



jenkins

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies. In this tutorial, we would explain how you can use Jenkins to build and test your software projects continuously.

Audience

This tutorial is going to help all those software testers who would like to learn how to build and test their projects continuously in order to help the developers to integrate the changes to the project as quickly as possible and obtain fresh builds.

Prerequisites

Jenkins is a popular tool for performing continuous integration of software projects. This is a preliminary tutorial that covers the most fundamental concepts of Jenkins. Any software professional having a good understanding of Software Development Life Cycle should benefit from this tutorial.

Disclaimer & Copyright

© Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute, or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness, or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

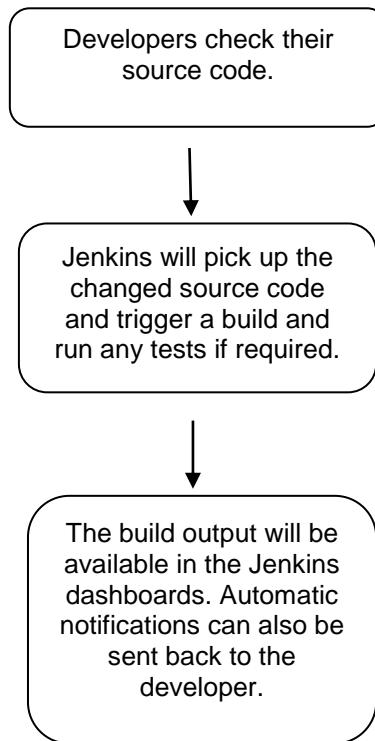
About the Tutorial	i
Audience.....	i
Prerequisites.....	i
Disclaimer & Copyright	i
Table of Contents.....	ii
 1. JENKINS – OVERVIEW.....	1
Why Jenkins?	1
What is Continuous Integration?	1
System Requirements	2
 2. JENKINS – INSTALLATION.....	3
Download Jenkins.....	3
Starting Jenkins.....	4
Accessing Jenkins.....	5
 3. JENKINS – TOMCAT SETUP.....	6
 4. JENKINS – GIT SETUP.....	10
 5. JENKINS – MAVEN SETUP.....	16
 6. JENKINS – CONFIGURATION.....	23
 7. JENKINS – MANAGEMENT.....	27
Configure System.....	28
Reload Configuration from Disk	28
Manage Plugins	29
System Information	29
 8. JENKINS – SETUP BUILD JOBS.....	32

9.	JENKINS – UNIT TESTING.....	43
	Example of a Junit Test in Jenkins	44
10.	JENKINS – AUTOMATED TESTING.....	53
11.	JENKINS – NOTIFICATION.....	60
12.	JENKINS – REPORTING	63
13.	JENKINS – CODE ANALYSIS.....	64
14.	JENKINS – DISTRIBUTED BUILDS.....	65
15.	JENKINS – AUTOMATED DEPLOYMENT	70
16.	JENKINS – METRICS AND TRENDS	73
17.	JENKINS – SERVER MAINTENANCE.....	87
	URL Options.....	87
	Backup Jenkins Home	87
18.	JENKINS – CONTINUOUS DEPLOYMENT	89
19.	JENKINS – MANAGING PLUGINS.....	102
	Uninstalling Plugins	102
	Installing another Version of a Plugin	103
20.	JENKINS – SECURITY.....	104
21.	JENKINS – BACKUP PLUGIN	109
22.	JENKINS – REMOTE TESTING.....	117

1. 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.

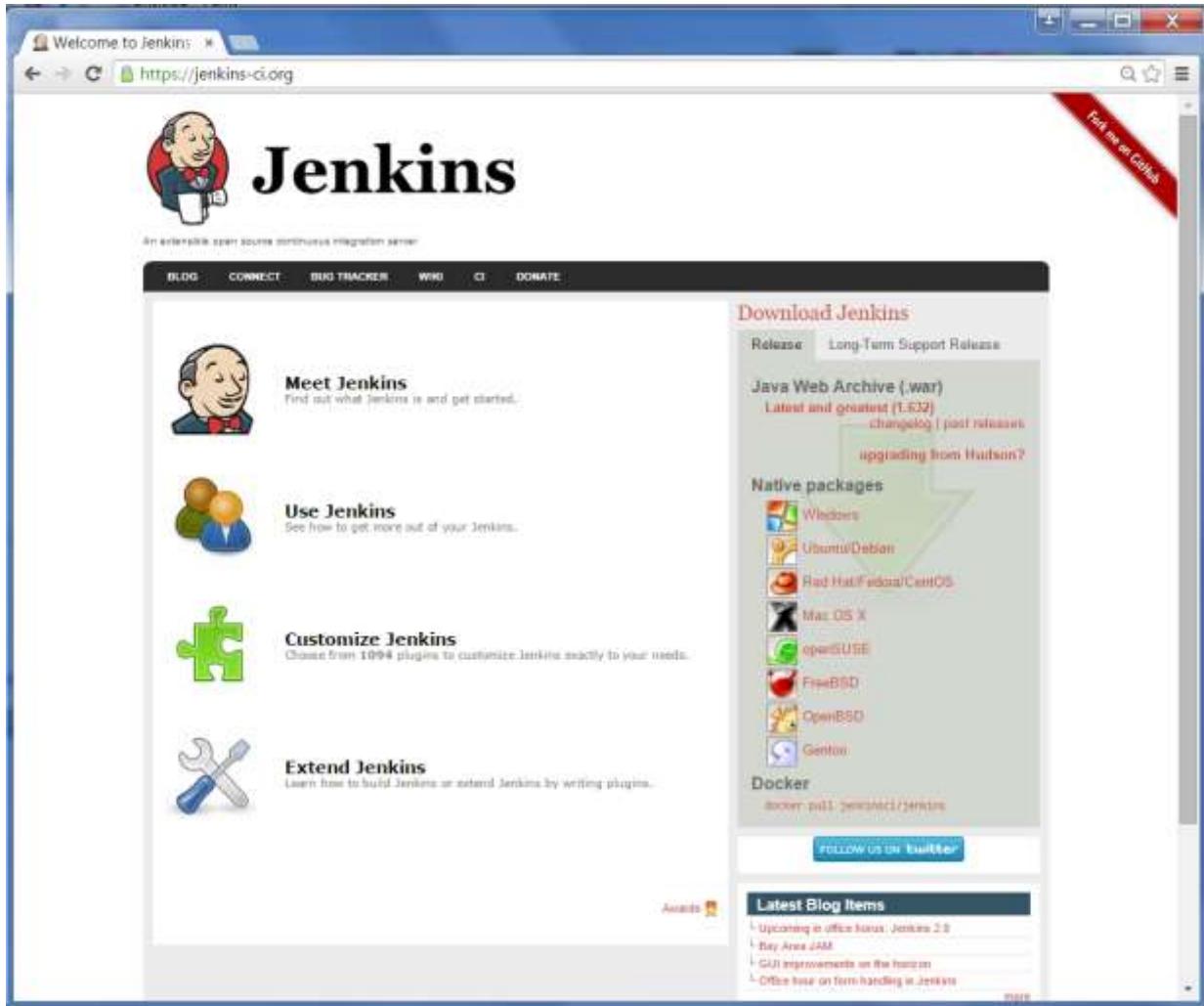
System Requirements

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).

2. Jenkins – Installation

Download Jenkins

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



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



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

Starting Jenkins

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

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

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

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

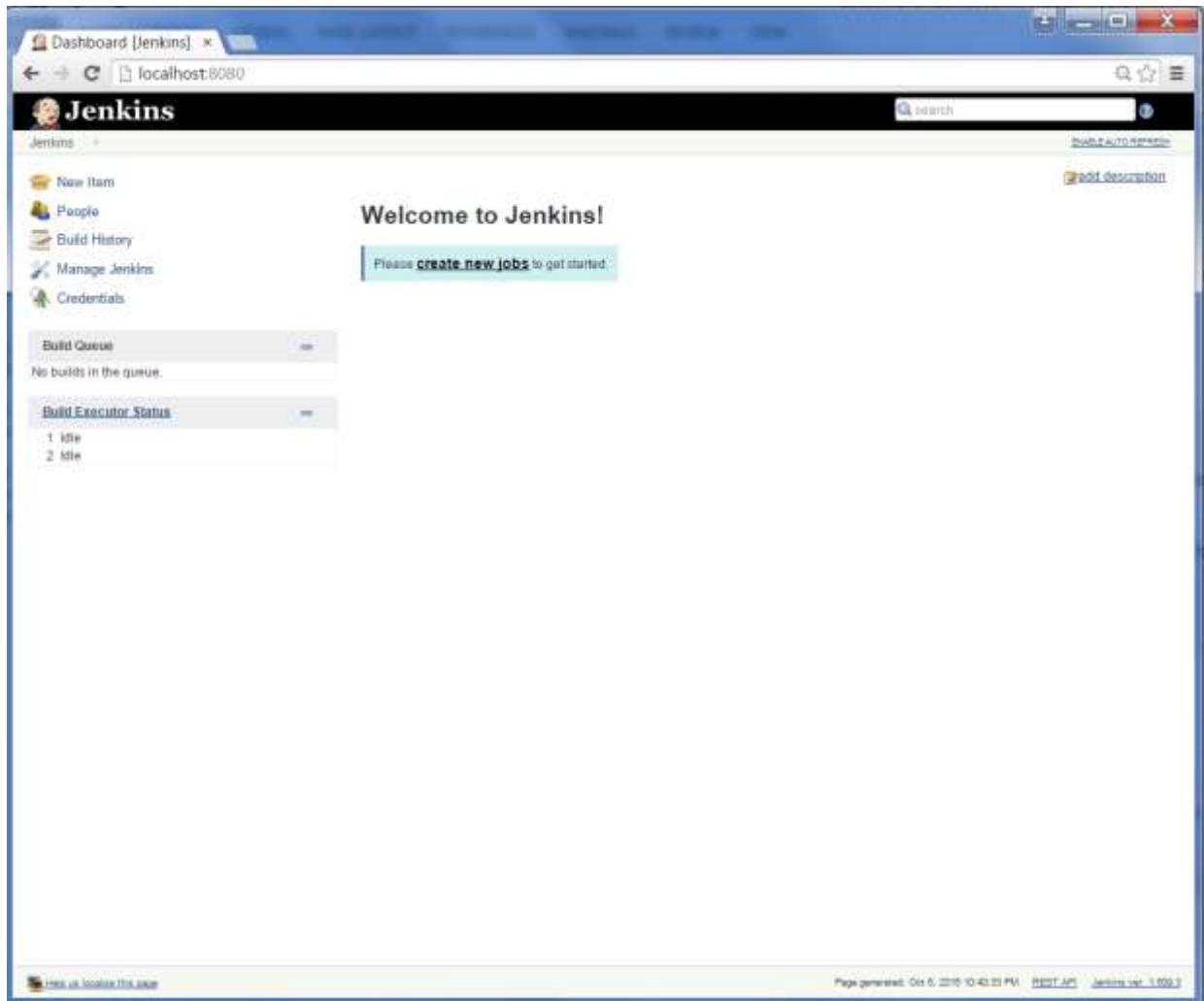
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

INFO: Jenkins is fully up and running

Accessing Jenkins

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

This link will bring up the Jenkins dashboard.



3. Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

Step 1: Verifying Java Installation

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

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

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

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

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

In case you do not have Java JDK, you can download it from the link <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

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	<code>export JAVA_HOME=/usr/local/java-current</code>
-------	---

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	<code>export PATH=\$PATH:\$JAVA_HOME/bin/</code>

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

Step 3: Download Tomcat

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

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

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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

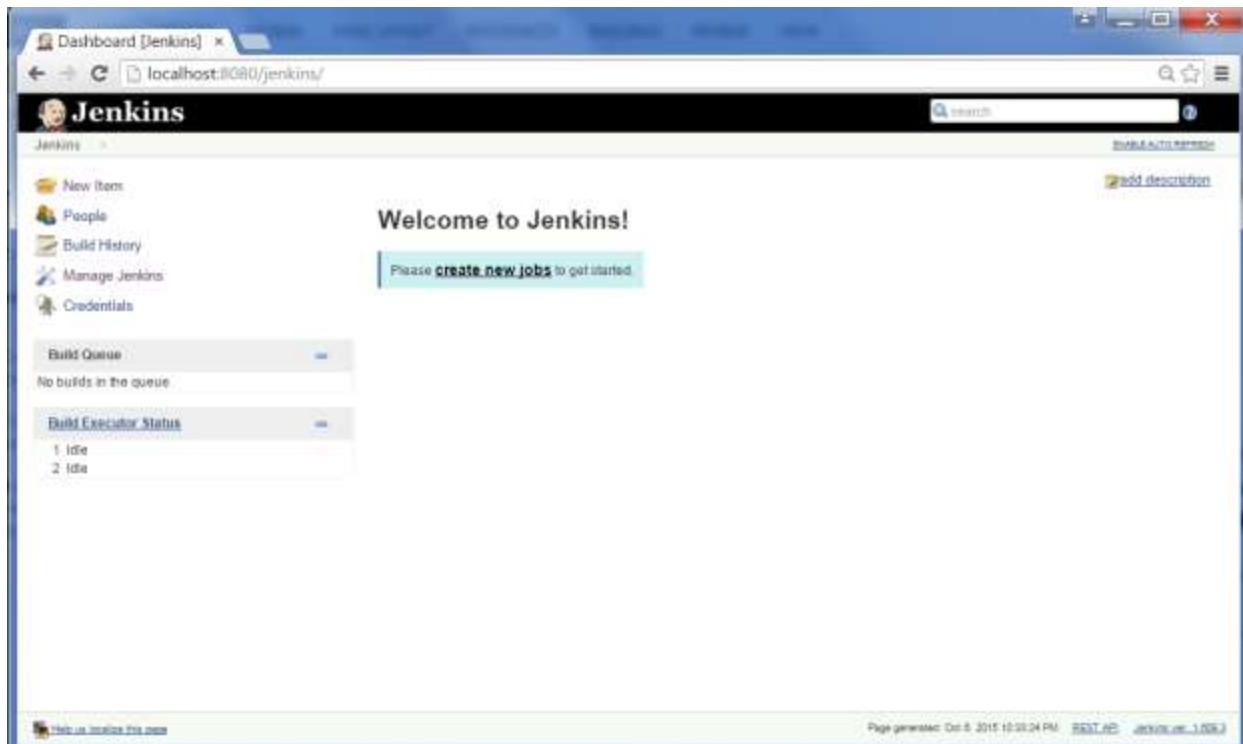
Now open the command prompt. From the command prompt, browse to the directory where the tomcat7 folder is location. Browse to the bin directory in this folder and run the start.bat file

```
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/jenkins>. Jenkins will be up and running on tomcat.

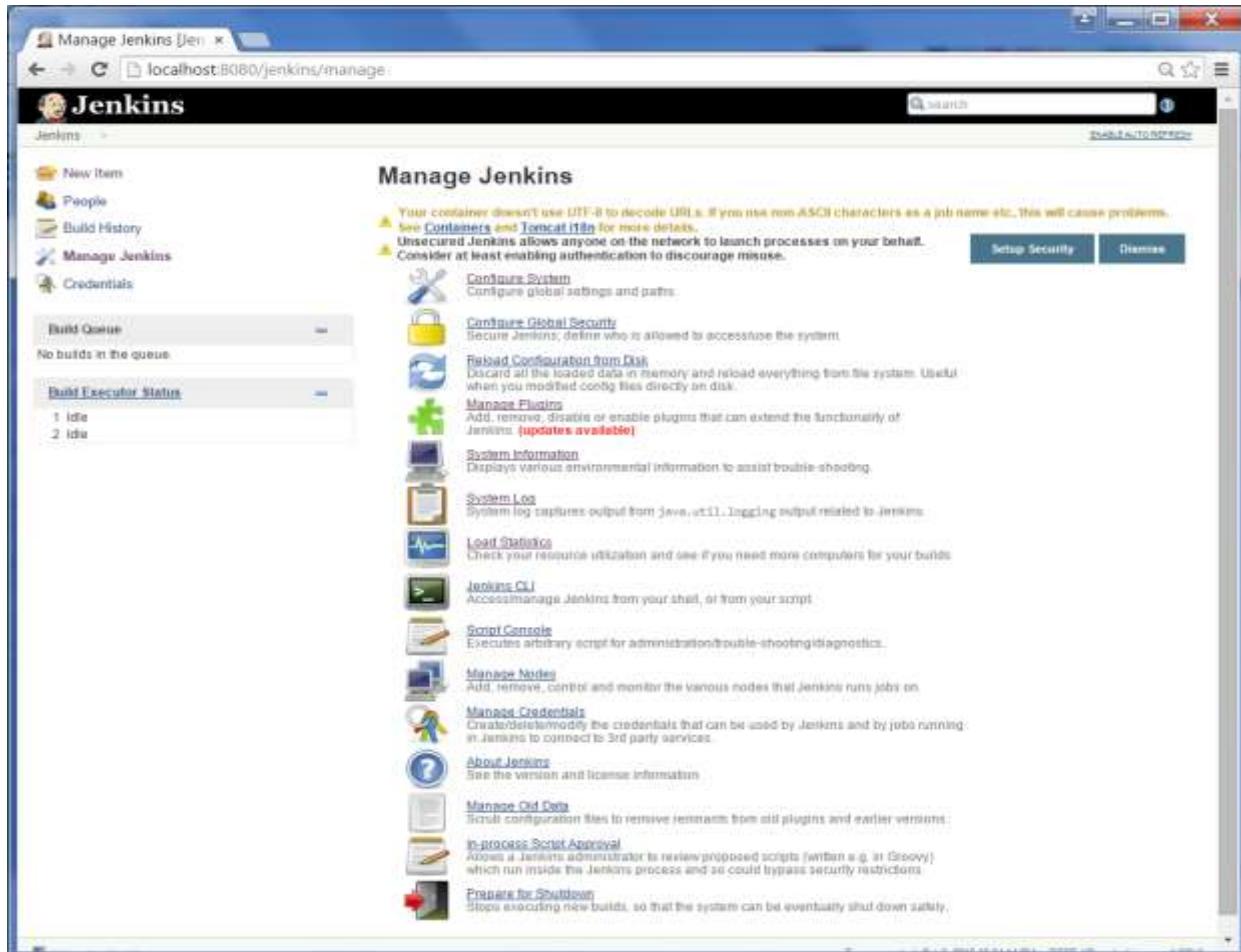


4. Jenkins – Git Setup

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.



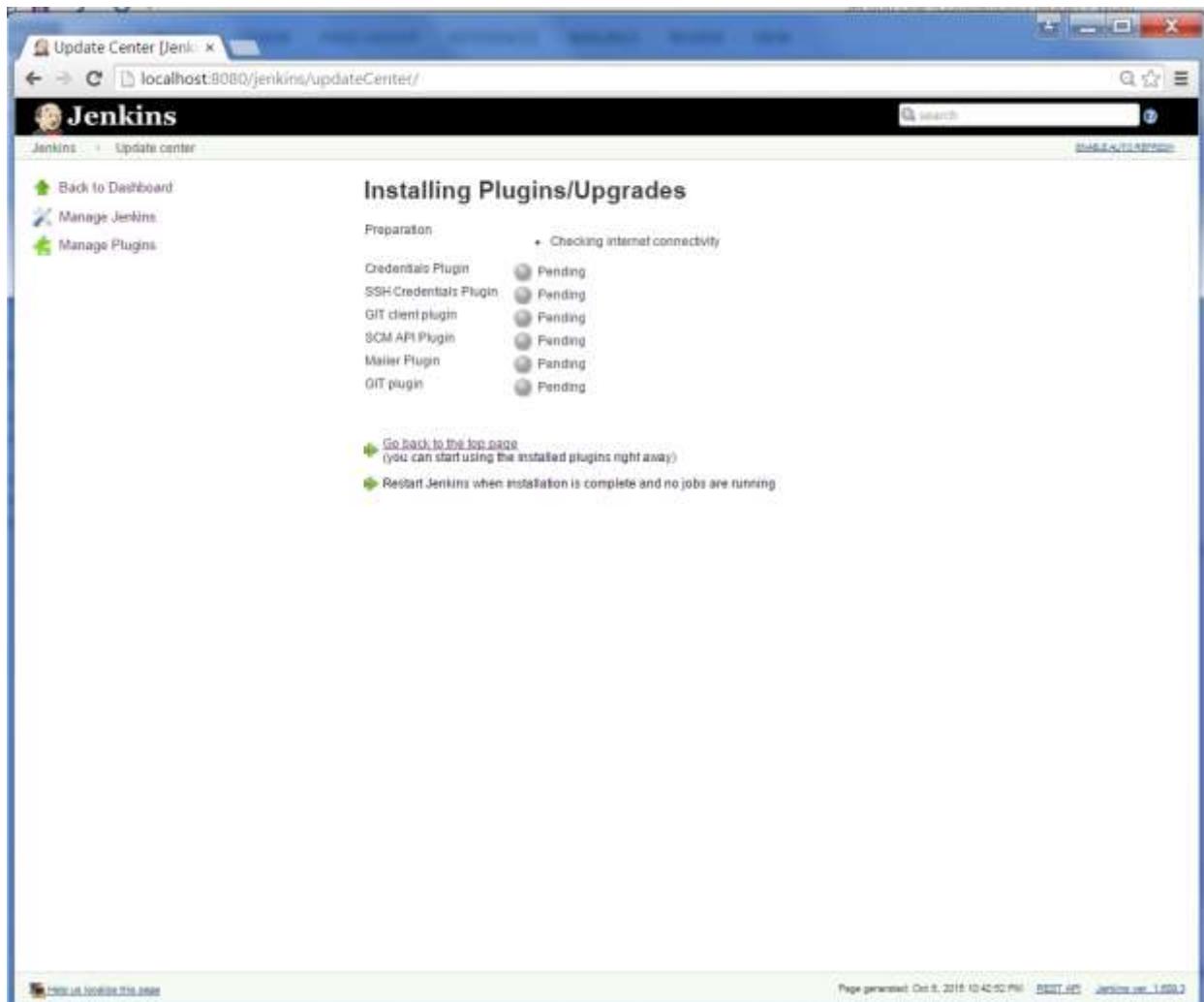
In the next screen, click the Available tab. This tab will give a list of plugins which are available for downloading. In the 'Filter' tab type 'Git plugin'

The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'Git plugin'. Below the search bar, there is a table with columns: Name, Version, and several other columns that are mostly hidden or not visible in the screenshot. One row in the table is highlighted with a blue background, corresponding to the 'GIT plugin' entry. At the bottom of the page, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'.

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 it, showing the Jenkins Plugin Manager with the 'Available' tab selected and a filter for 'Git plugin'. The 'GIT plugin' option is highlighted with a blue background. The same three buttons at the bottom ('Install without restart', 'Download now and install after restart', 'Check now') are present.

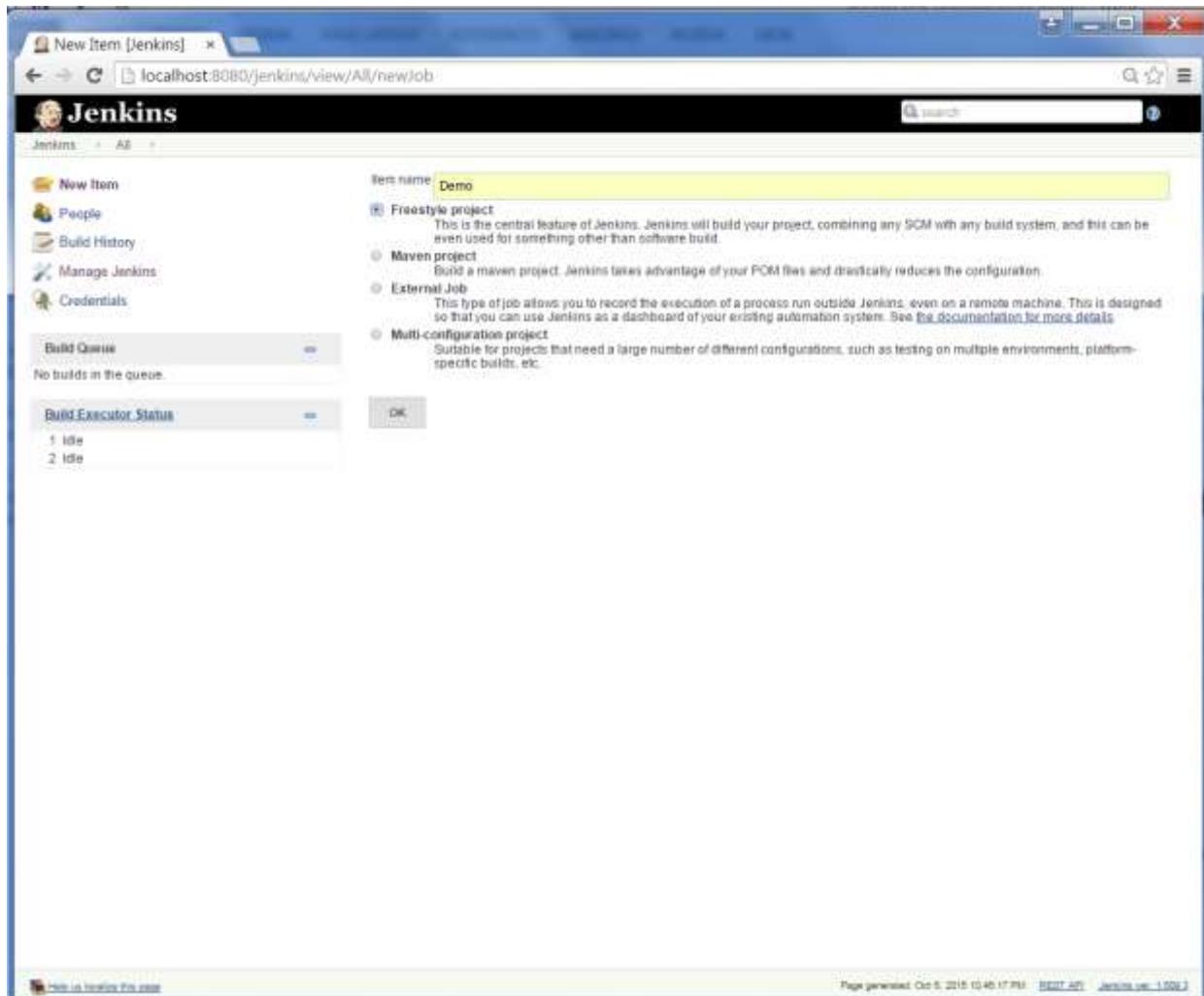
The installation will then begin and the screen will be refreshed to show the status of the download.



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.

The screenshot shows the Jenkins configuration interface for a job named 'Demo'. The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main content area has sections for Project name (set to 'Demo'), Description, Advanced Project Options (with options like Discard Old Builds, This build is parameterized, Disable Build, and Execute concurrent builds if necessary), Source Code Management (with Git selected), Build Triggers (with options like Build after other projects are built, Build periodically, and Poll SCM), and Build (with an 'Add build step' dropdown). Post-build Actions are also present with an 'Add post-build action' dropdown. At the bottom are Save and Apply buttons.

5. Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

The screenshot shows a web browser window displaying the Apache Maven Project download page at <http://maven.apache.org/download.cgi>. The page features the Apache logo and the large 'Maven' logo. A sidebar on the left contains links for Apache, Maven, and Download Apache Maven. The main content area is titled 'Downloading Apache Maven 3.3.3'. It includes a note about the latest release, a mirror selection dropdown set to 'http://www.eu.apache.org/dist/' with a 'Change' button, and a 'System Requirements' section with details for Java, Development Kit (JDK), Memory, Disk, and Operating System. Below this is a 'Files' section with a table showing download links, checksums, and signatures for various Maven distributions. The table has columns for Link, Checksum, and Signature.

	Link	Checksum	Signature
Maven 3.3.3 Complete Distribution (binaries)	Download	SHA-1	GPG
Maven 3.3.3 Complete Distribution (source)	Download	SHA-1	GPG
Maven 3.3.3 Source Only (binaries)	Download	SHA-1	GPG
Maven 3.3.3 Source Only (source)	Download	SHA-1	GPG
Maven 3.3.3 Binaries Only (binaries)	Download	SHA-1	GPG
Maven 3.3.3 Binaries Only (source)	Download	SHA-1	GPG

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

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

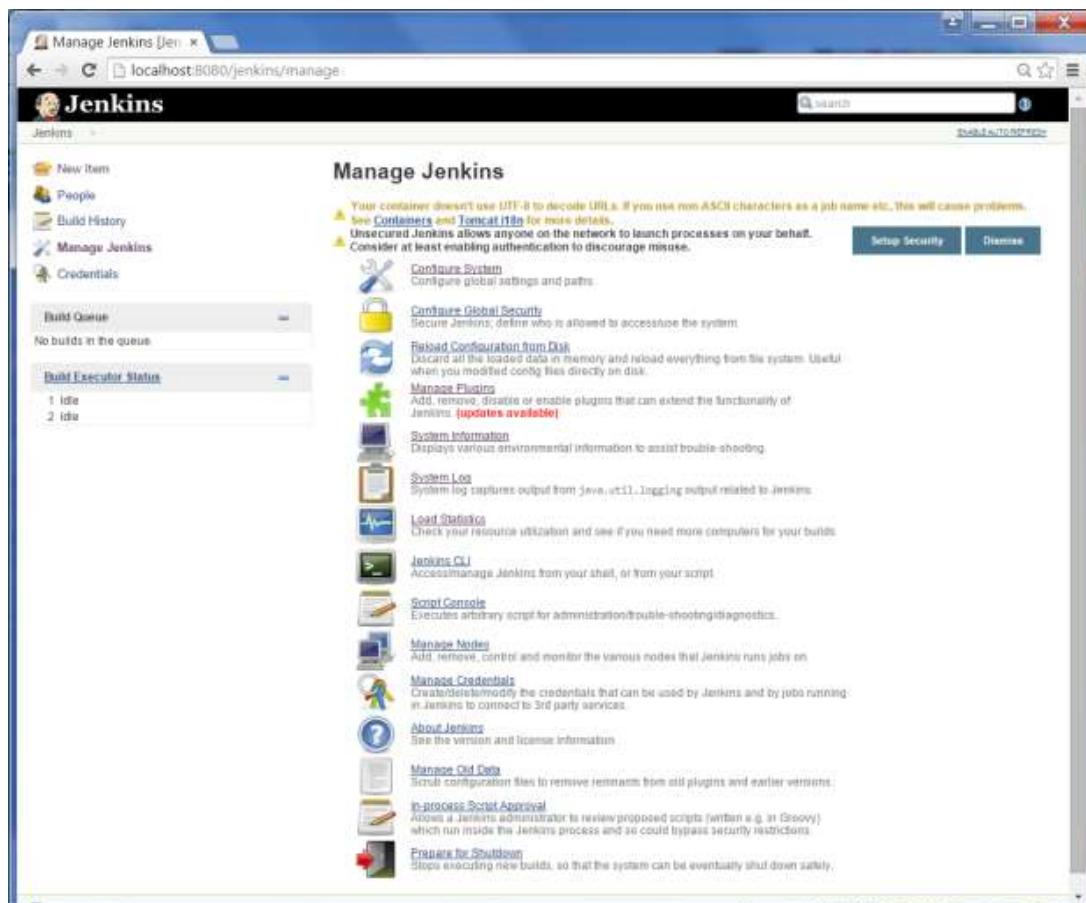
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

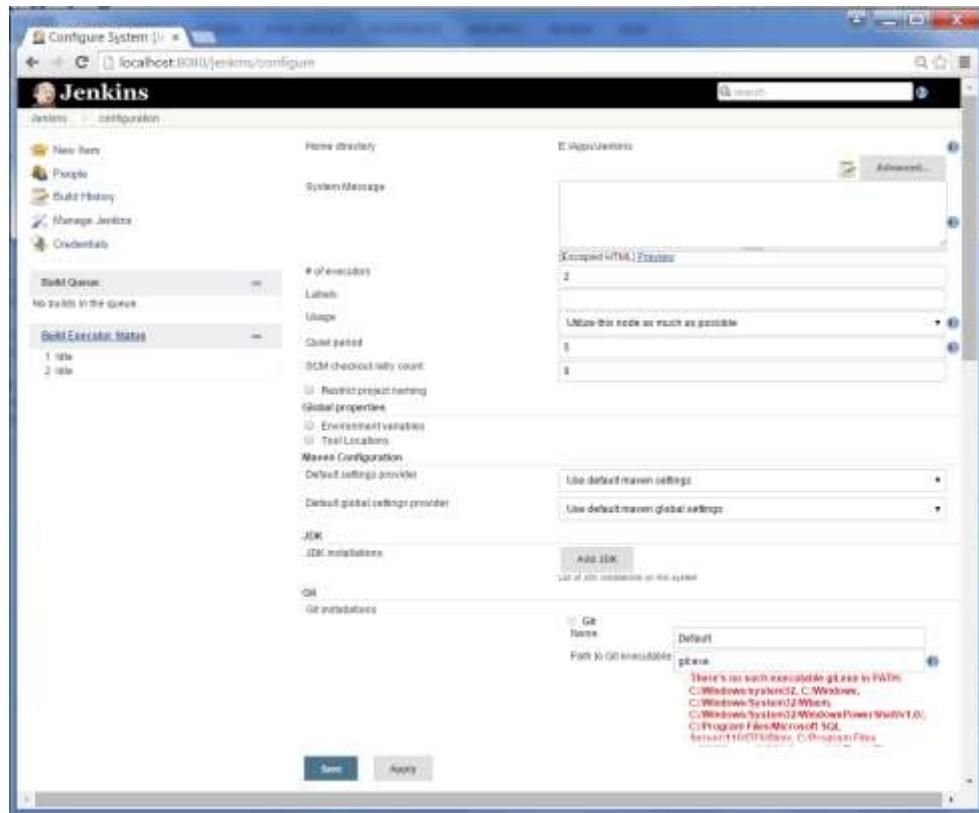
Step 2: Setting up Jenkins and Maven

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

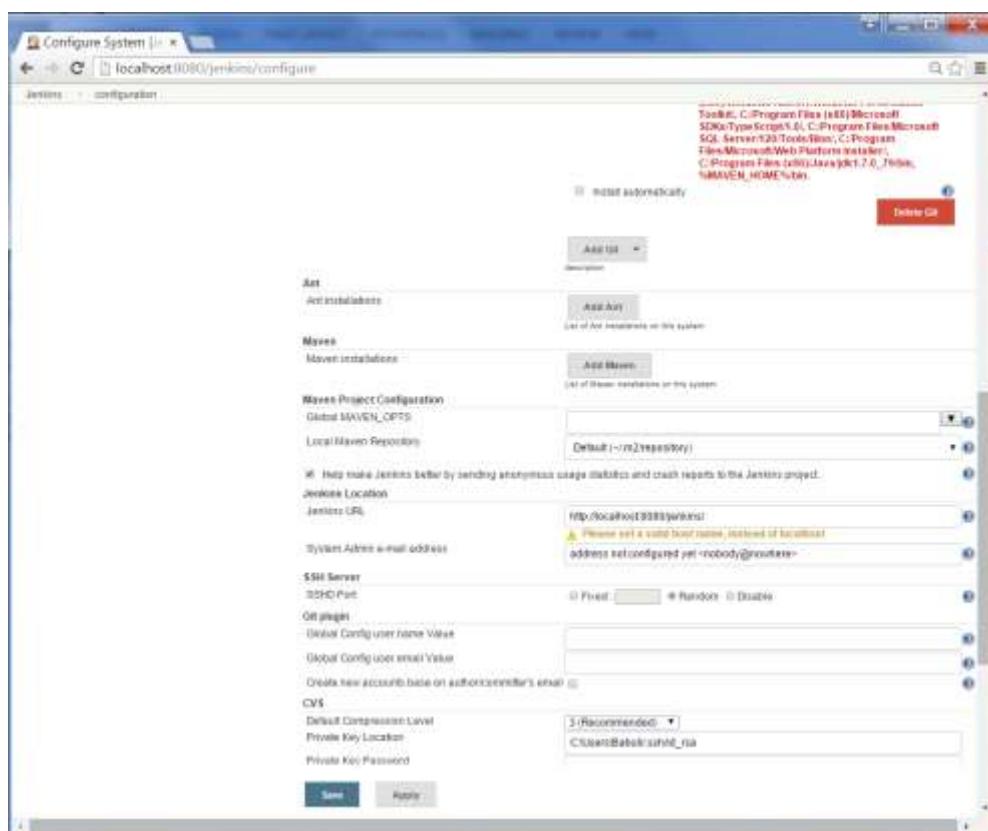


Then, click on 'Configure System' from the right hand side.





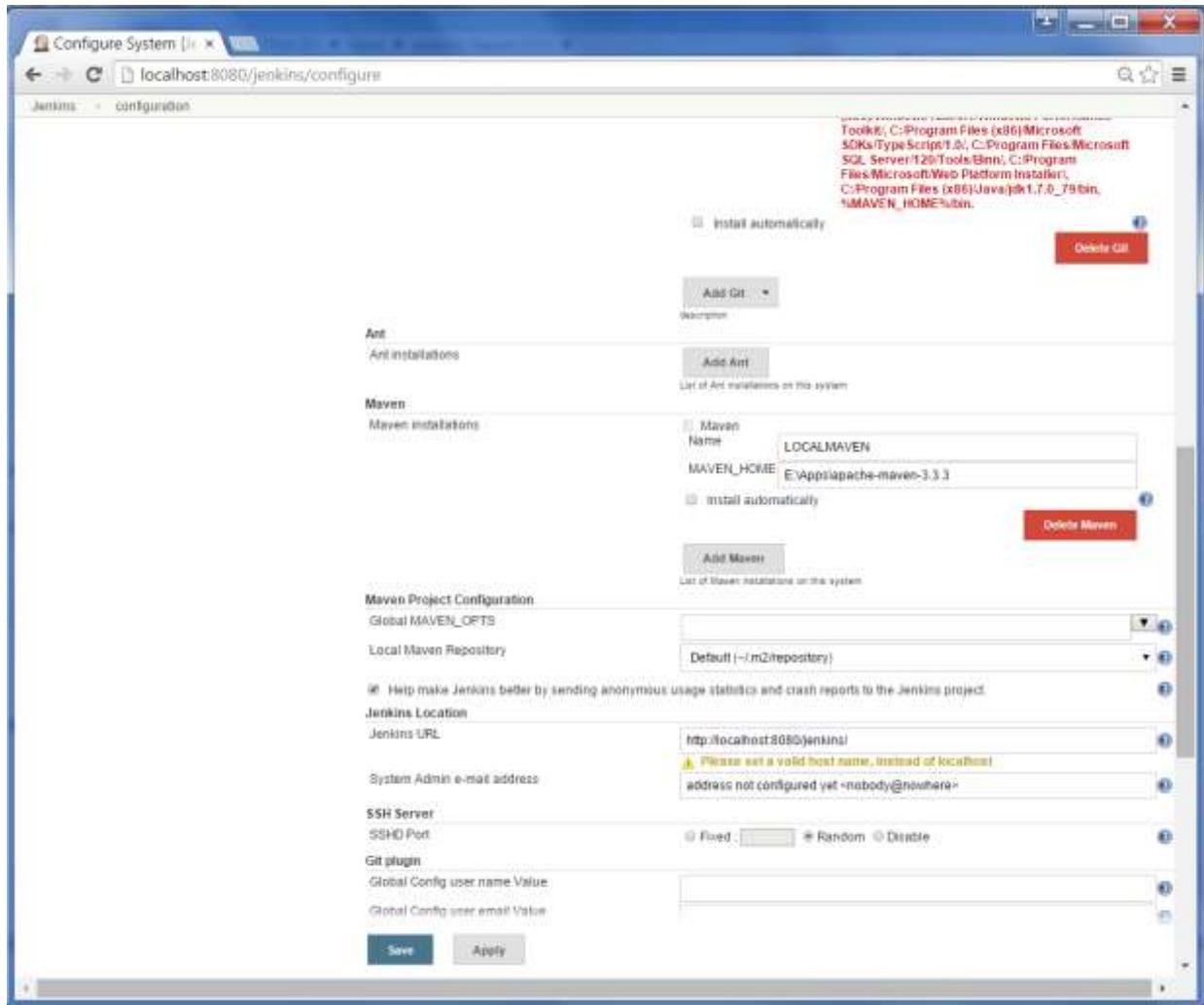
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



Uncheck the 'Install automatically' option.

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

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



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

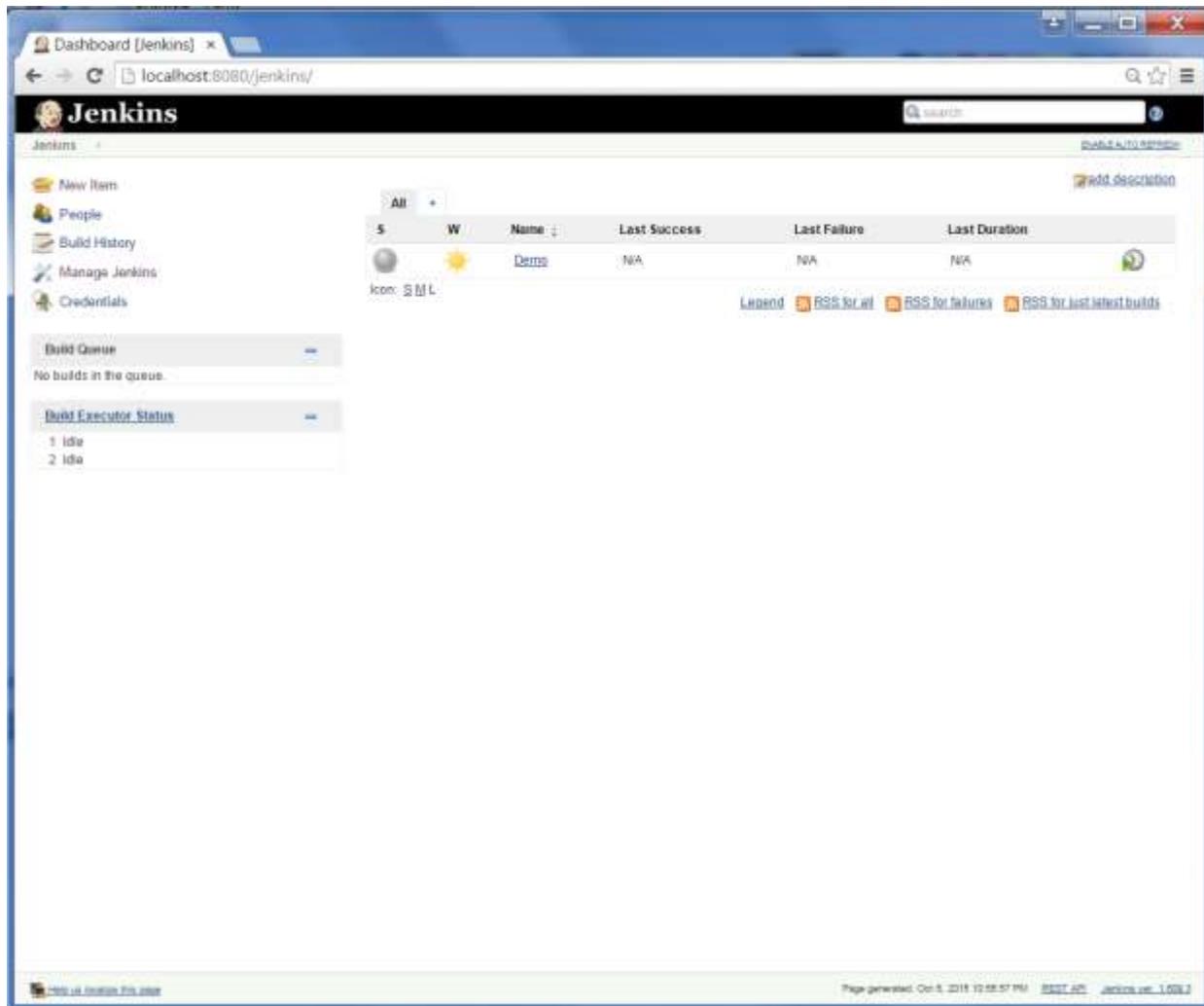
The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a table with one item: 'Demo' (status: W, Last Success: N/A, Last Failure: N/A, Last Duration: N/A). Below the table are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 idle, 2 idle). At the bottom, there are links for 'Help us translate this page', 'Page generated: Oct 5, 2018 12:58:57 PM', 'REST API', and 'Jenkins ver: 1.603.2'.

The screenshot shows the Jenkins web interface for creating a new job. The URL in the browser is `localhost:8080/jenkins/view/All/newJob`. 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, it shows "1 idle" and "2 idle". The main content area is titled "New Item [Jenkins]" and contains a form for creating a Maven project. The "Item name" field is set to "MavenDemo". Below it, there are five radio button options: "Freestyle project", "Maven project" (which is selected), "External Job", "Multi-configuration project", and "Copy existing item". The "Maven project" option is described as building a Maven project using POM files. The "External Job" option is described as recording a process run outside Jenkins. The "Multi-configuration project" option is for testing multiple environments. The "Copy existing item" option allows copying from another Jenkins item. At the bottom right of the form is an "OK" button.

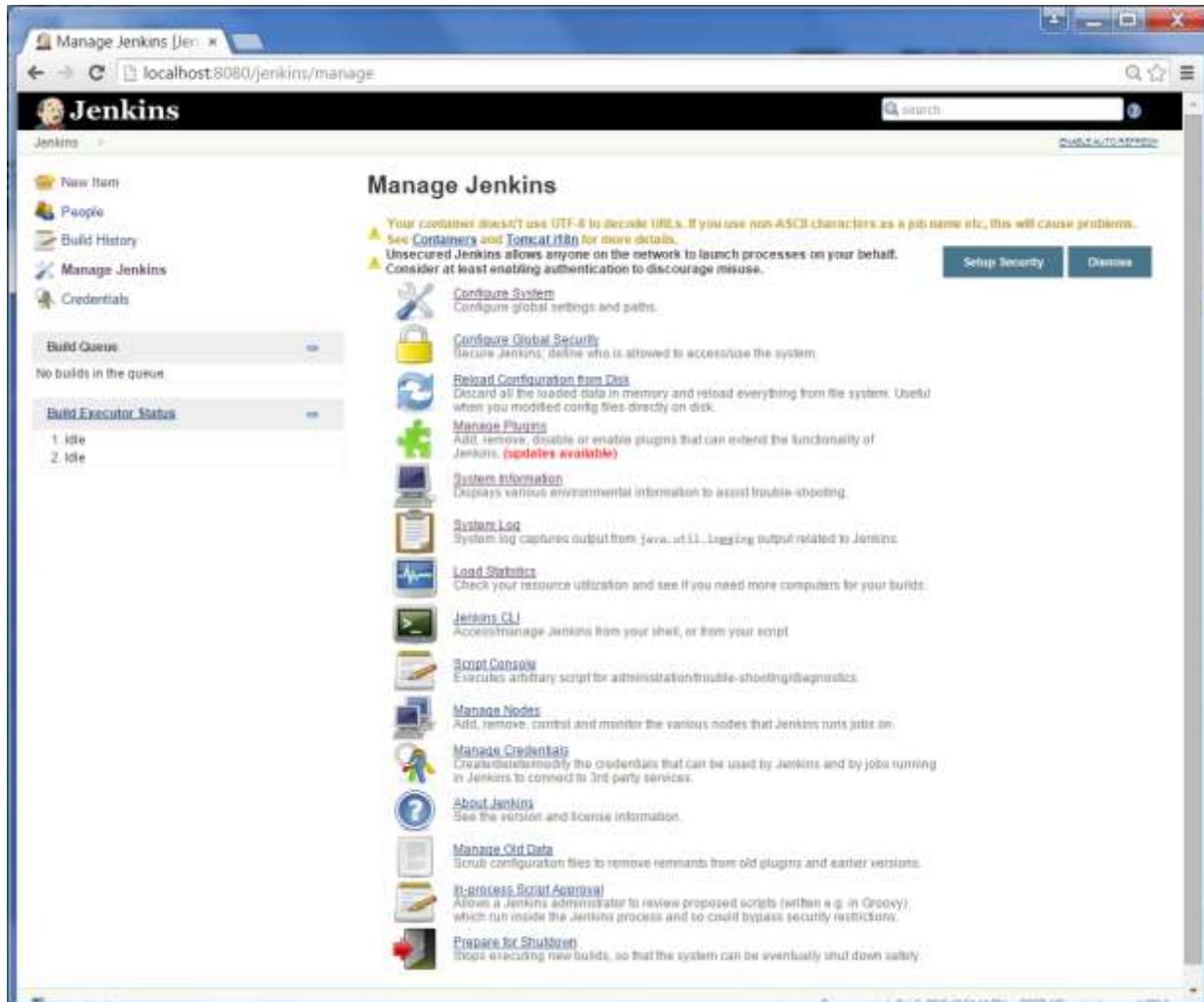
6. Jenkins – Configuration

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

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



You will then be presented with the following screen:



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

Jenkins Home Directory

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

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

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

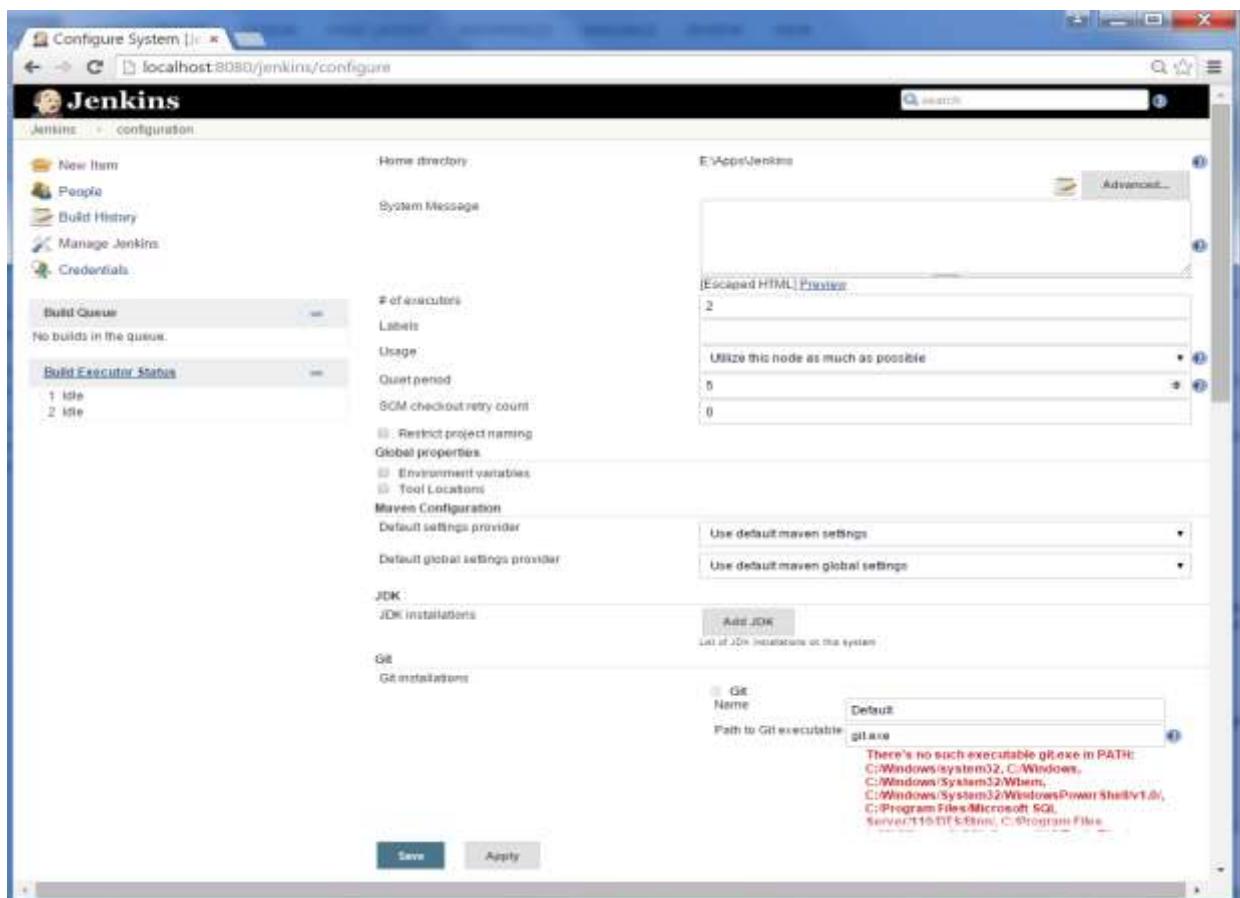
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/jenkins` to this new directory.

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

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

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

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



of executors

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

Environment Variables

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

Jenkins URL

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

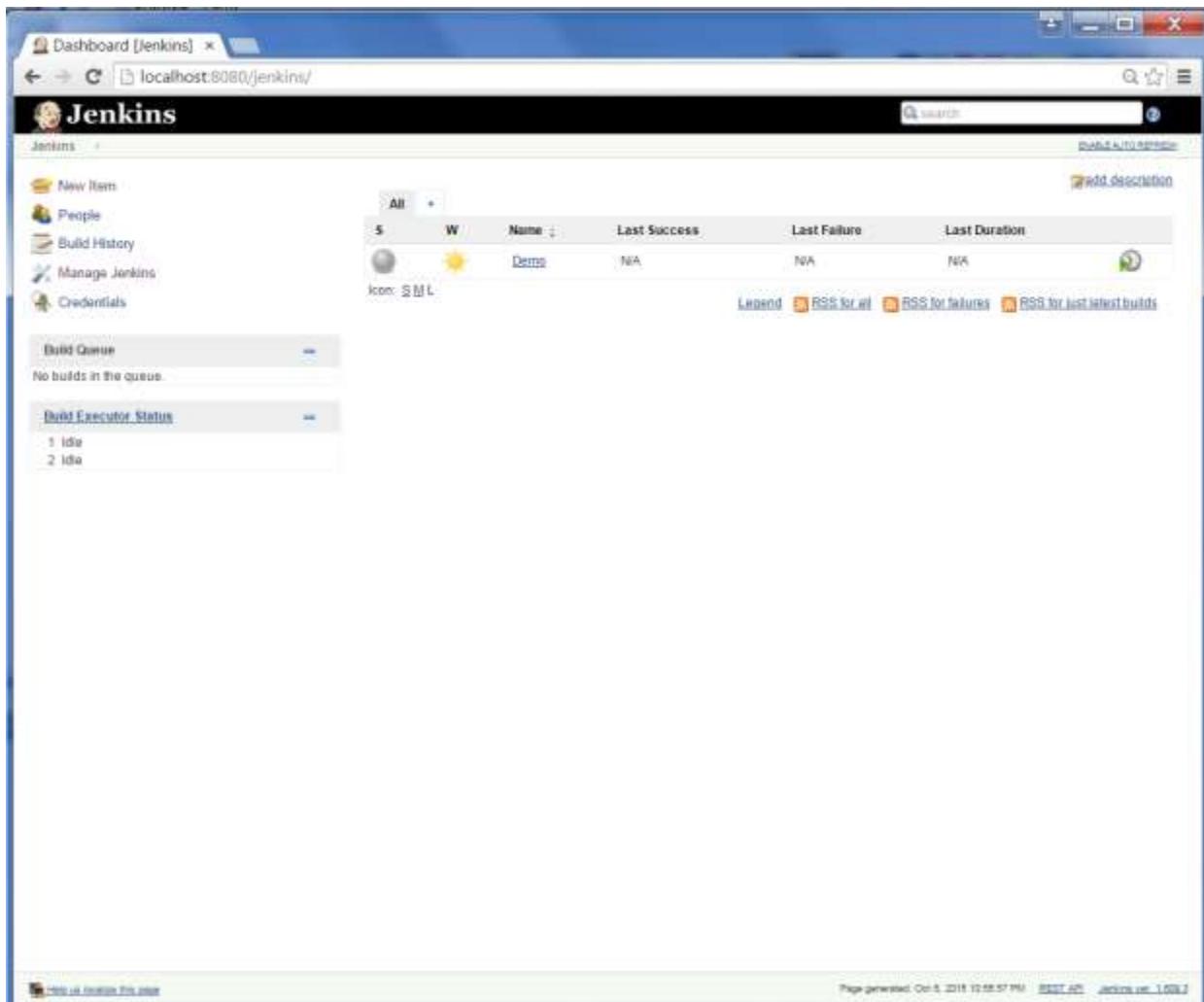
Email Notification

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

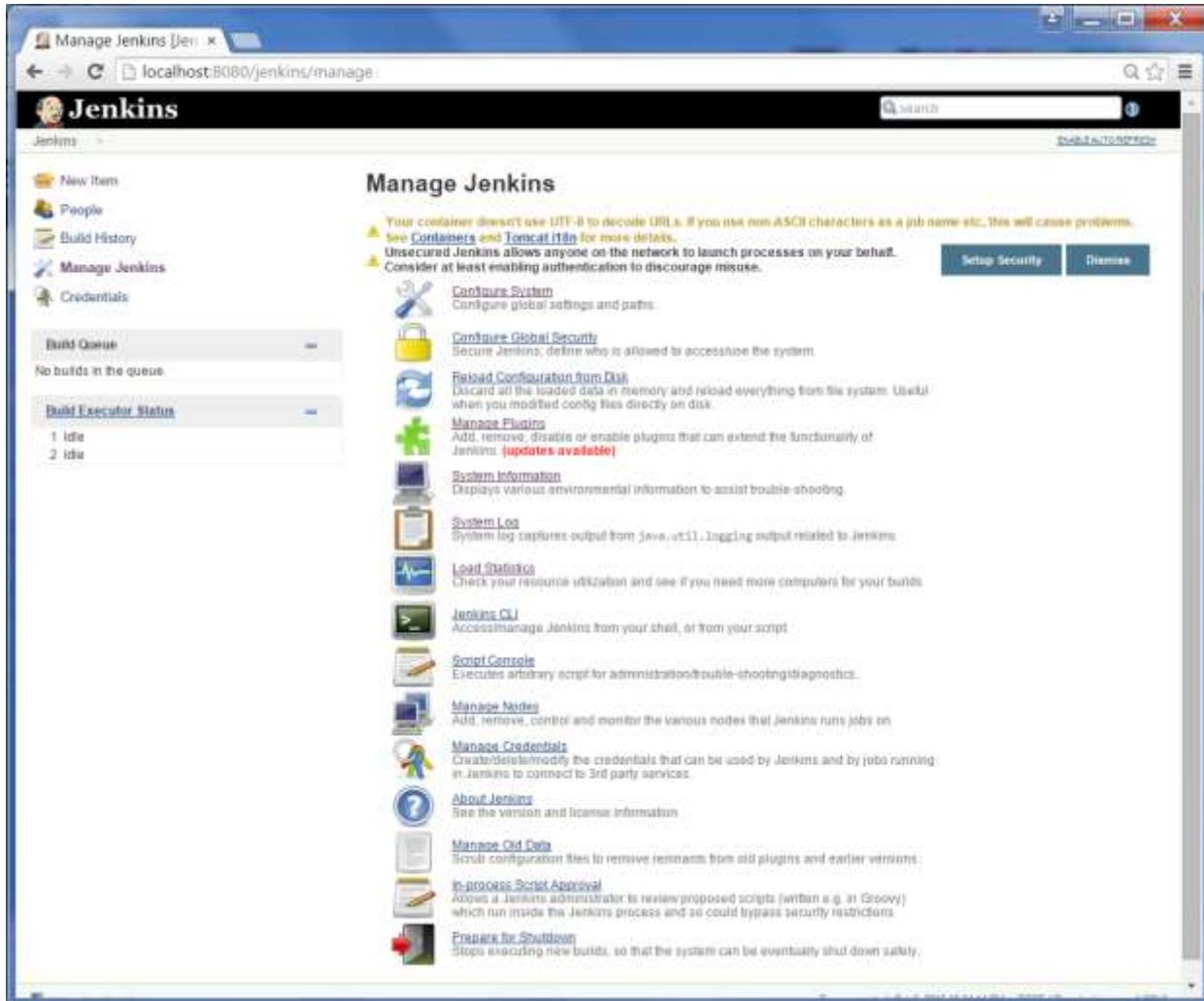
7. Jenkins – Management

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 Plugins

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 'Manage Plugins' interface. The 'Updates' tab is selected, displaying a list of available plugins:

Name	Version	Installed
CVS Plugin	2.12	2.11
Javadoc Plugin	1.3	1.1
JUnit Plugin	1.9	1.2-beta-4
Matrix Authorization Strategy Plugin	1.2	1.1
Matrix Project Plugin	1.6	1.4.1
Maven Integration plugin	2.12.1	2.7.1
OWASP Markup Formatter Plugin	1.3	1.1
PAM Authentication plugin	1.2	1.1
Script Security Plugin	1.15	1.13
SSH Slaves plugin	1.10	1.8
Subversion Plugin	2.5.3	1.54
Translation Assistance plugin	1.12	1.10
Windows Slaves Plugin	1.1	1.0

Below the table, there are buttons for 'Download now and install after restart' and 'Check now'. A note says 'Select All. None' and 'This page lists updates to the plugins you currently use'. At the bottom, it shows 'Page generated: Oct 6, 2015 11:06:28 PM' and 'Jenkins ver: 1.603.3'.

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 interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', 'Build Queue' (empty), and 'Build Executor Status' (two idle). The main area is titled 'System Properties' and contains a table with over 50 system properties and their values. Some key entries include:

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*.jar
file.encoding	Cp1252
file.encoding.pkg	sun.cs
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Appstromcat7\lib\bootstrap.jar;E:\Appstromcat7\bin\Normal\juli.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\SunJavaUpdate\lib
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\SunJava\bin;C:\Windows\System32;C:\Windows\bin;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files\Microsoft SQL Server\110\DTSP\Binn\;C:\Program Files\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files\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.

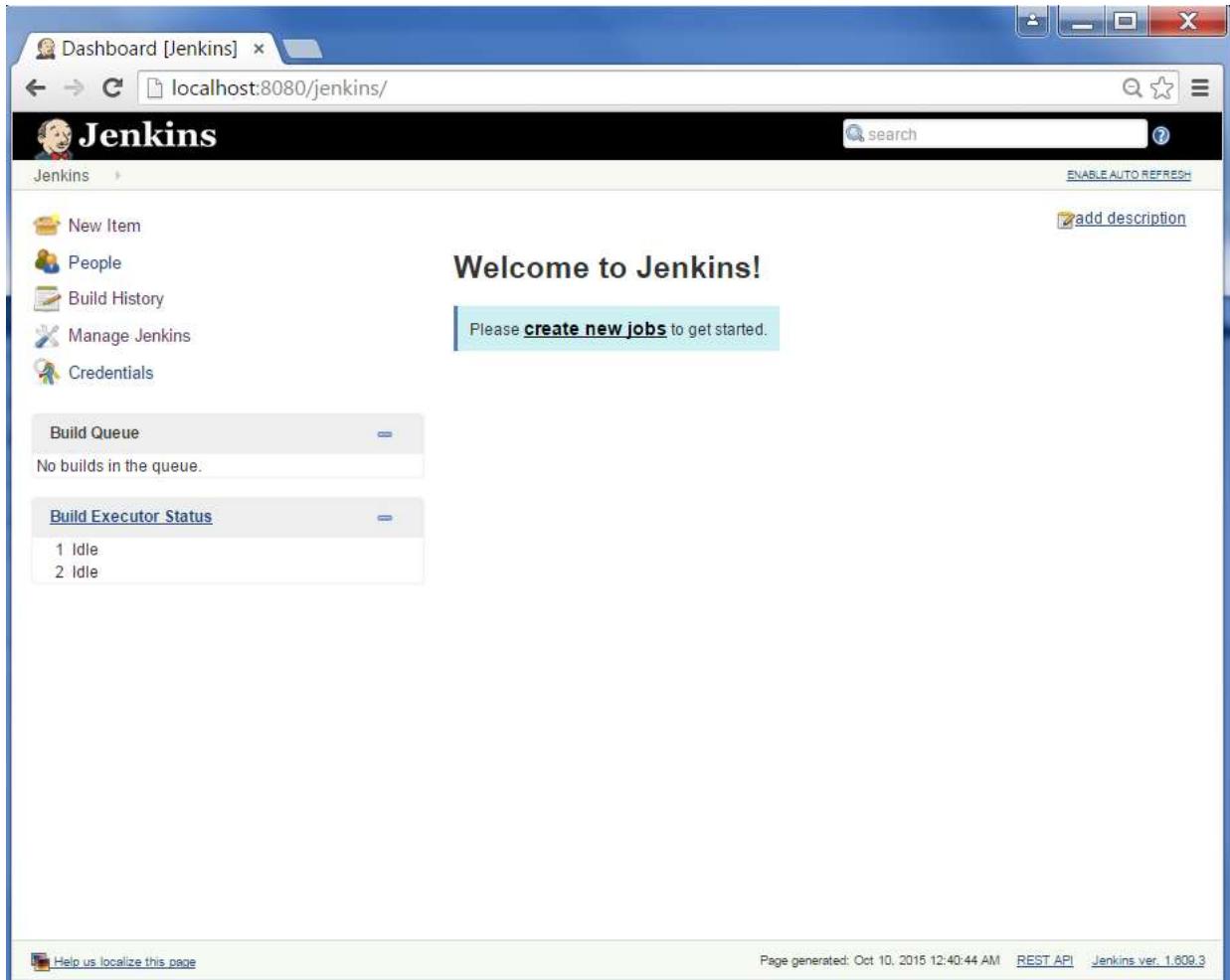
Prepare for Shutdown

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.

8. 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'

New Item [Jenkins] ×

localhost:8080/jenkins/view/All/newJob

Jenkins

search

Item name: **Helloworld**

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

OK

Help us localize this page

Page generated: Oct 10, 2015 12:41:39 AM REST API Jenkins ver. 1.609.3

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

The screenshot shows the Jenkins configuration interface for a project named 'Helloworld'. The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main panel displays the following configuration details:

- Project name:** Helloworld
- Description:** (Empty text area)
- [Escaped HTML] Preview:** (Empty text area)
- Advanced Project Options:**
 - Discard Old Builds
 - GitHub project
 - This build is parameterized
 - Disable Build (No new builds will be executed until the project is re-enabled.)
 - Execute concurrent builds if necessary
- Source Code Management:**
 - None
 - CVS
 - CVS Projectset
 - Git
 - Subversion
- Build Triggers:**
 - Build after other projects are built
 - Build periodically
 - Build when a change is pushed to GitHub
 - Poll SCM
- Build:**
 - Add build step

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

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.

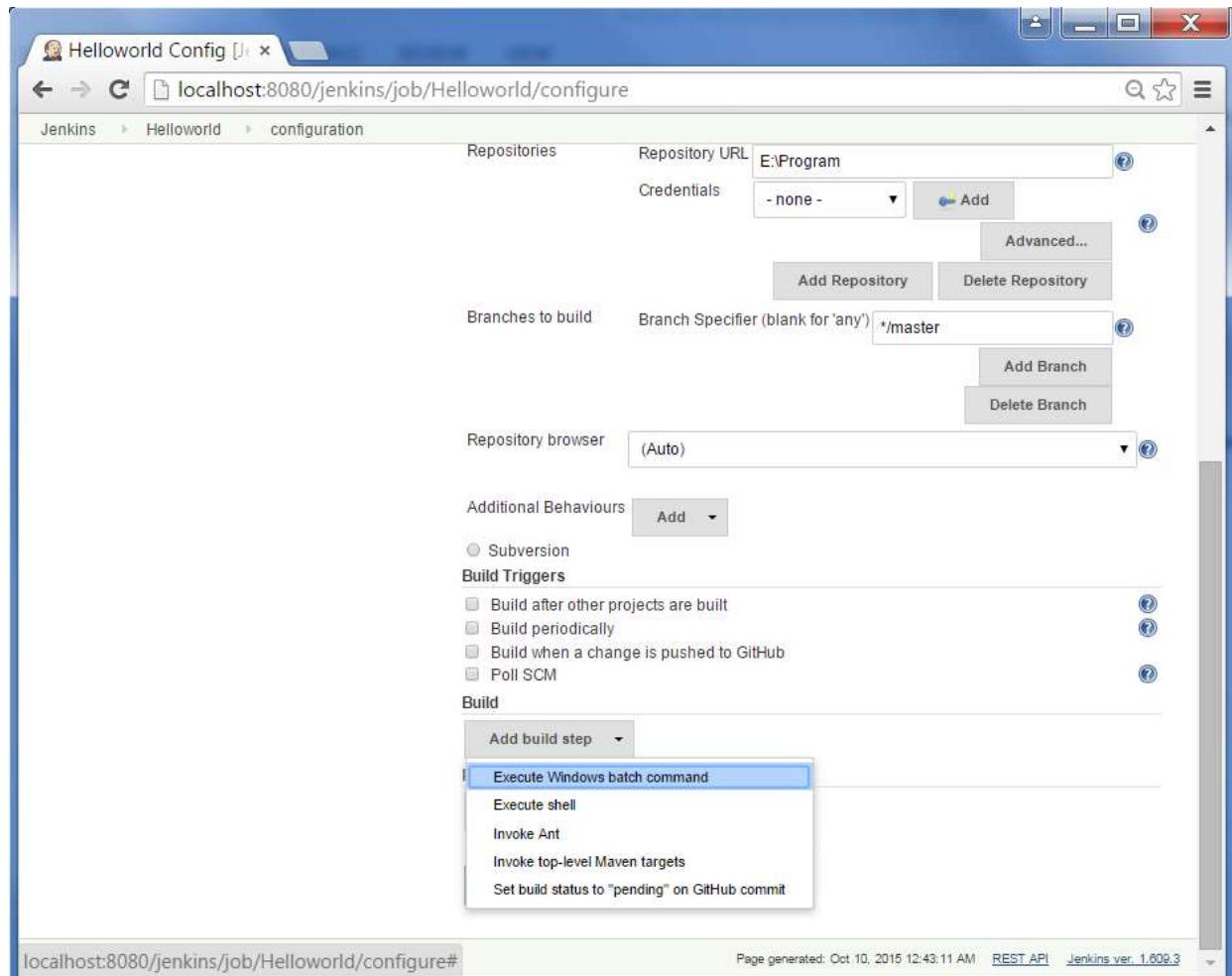
Note – If your repository is hosted on GitHub, you can also enter the URL of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the GitHub repository so that the code can be picked up from the remote repository.

The screenshot shows the Jenkins configuration interface for a job named 'Helloworld'. The left sidebar includes links for Changes, Workspace, Build Now, Delete Project, and Configure. The main panel displays several configuration sections:

- Escaped HTML Preview:** A large text area for previewing escaped HTML.
- Advanced Project Options:** Includes checkboxes for Discard Old Builds, GitHub project, This build is parameterized, Disable Build, and Execute concurrent builds if necessary. There is also an **Advanced...** button.
- Source Code Management:** Set to **Git**. The **Repository URL** is set to `E:\Program`, and the **Credentials** dropdown is set to `- none -`. Buttons for **Add**, **Advanced...**, **Add Repository**, and **Delete Repository** are available.
- Branches to build:** The **Branch Specifier** is set to `*/master`. Buttons for **Add Branch** and **Delete Branch** are present.
- Repository browser:** Set to `(Auto)`.

At the bottom are **Save** and **Apply** buttons.

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 a job named "Helloworld". The top navigation bar includes links for "Repository browser", "Delete Branch", "Additional Behaviours", "Add", "Subversion", "Build Triggers", and "Build". Under "Build", there is a section for "Execute Windows batch command" with the command text:
javac HelloWorld.java
java HelloWorld

Below the build steps, there are sections for "Post-build Actions" and "Add post-build action". At the bottom of the page 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 interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below this is a search bar and a 'Project Helloworld' title. To the right of the title are buttons for 'add description' and 'Disable Project'. Underneath the title, there are two icons: 'Workspace' (a folder icon) and 'Recent Changes' (a notebook icon). At the bottom left, there are 'Build History' and 'RSS' links. The bottom right corner displays the page generation information: 'Page generated: Oct 10, 2015 12:51:53 AM REST API Jenkins ver. 1.609.3'.

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

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. On the right, there are buttons for 'Add description' and 'Disable Project'. Below the navigation, the title 'Project Helloworld' is displayed. A sidebar on the left lists 'Build History' (with one entry for build #1), 'trend', and links for 'RSS for all' and 'RSS for failures'. The main content area features two icons: 'Workspace' (a folder icon) and 'Recent Changes' (a notepad icon). At the bottom, there are 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 9 : Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below this is a 'Build History' section with a single entry: '#1 Oct 10, 2015 12:52 AM'. To the right of the history is a 'Permalinks' section with links for 'Workspace' and 'Recent Changes'. At the bottom of the page, there are links for 'RSS for all' and 'RSS for failures'. The footer includes a link to 'Help us localize this page' and copyright information: 'Page generated: Oct 10, 2015 12:51:53 AM REST API 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 web interface for a job named "Helloworld". The main title bar says "Helloworld #1 [Jenki...]" and the URL is "localhost:8080/jenkins/job/Helloworld/1/". The left sidebar has links: Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Git Build Data, and No Tags. The main content area shows "Build #1 (Oct 10, 2015 12:52:50 AM)". It includes a "Status" icon (blue circle with a white dot), a "Duration" of "Started 4 min 40 sec ago Took 4.7 sec", and a "Changes" section with a note "No changes.". It also shows the "Started by anonymous user" and the "Revision: 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f refs/remotes/origin/master" information. A "git" icon is present. At the bottom, there's a "Help us localize this page" link and footer text "Page generated: Oct 10, 2015 12:57:31 AM REST API Jenkins ver. 1.809.3".

The screenshot shows the Jenkins interface for a build named "Helloworld #12". The left sidebar contains links like "Back to Project", "Status", "Changes", "Console Output" (which is 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 build log:

```

Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program
timeout=10
Fetching upstream changes from E:\Program
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program +refs/heads/*:refs/remotes/origin/*
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9d9ae40e731a907f5c8f
(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
42f9a82ffadd86fb5c3a9d9ae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9d9ae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat

E:\Jenkins\jobs\Helloworld\workspace>javac HelloWorld.java

E:\Jenkins\jobs\Helloworld\workspace>java HelloWorld
Hello World

E:\Jenkins\jobs\Helloworld\workspace>exit 0
Finished: SUCCESS

```

At the bottom, there's a link "Help us localize this page" and footer text "Page generated: Oct 10, 2015 10:14:21 PM REST API Jenkins ver. 1.609.3".

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

9. 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.

The screenshot shows the Jenkins xUnit Plugin page. The left sidebar has a 'Jenkins' section with links like Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, Wiki Site Map, and Documents (Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes). The main content area has a title 'xUnit Plugin' with a red icon. Below it is a message: '7 Added by Gregory Boissinot, last edited by Gregory Boissinot on Oct 08, 2015 (view change)'. A 'Plugin Information' table shows details: Plugin ID: xunit, Latest Release: 1.98 (archives), Latest Release Date: Oct 09, 2015, Required Core: 1.580.1, Dependencies: junit (version: 1.6). It also shows Source Code (GitHub), Issue Tracking (Open Issues, Pull Requests), and Maintainer(s) (Gregory Boissinot). A 'Usage' section contains a chart titled 'xunit - installations' showing the number of installations from October 2014 to September 2015. The chart shows a steady increase from approximately 12,000 to over 13,000. A note below the chart states: 'This plugin makes it possible to publish the test results of an execution of a testing tool in Jenkins.' At the bottom, there's a 'CppUnit output' section with a yellow progress bar.

Features

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

Supported tools

Embedded tools

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

Other plugins as an extension of the xUnit plugin:

- * Gallio (Gallio plugin)
- * Parasoft C++Test tool (Cpptest Plugin)
- * JSUnit (JSUnit Plugin)
- * JBehave
- * TestComplete (TestComplete xUnit Plugin)

External contributions

Example of a Junit Test in Jenkins

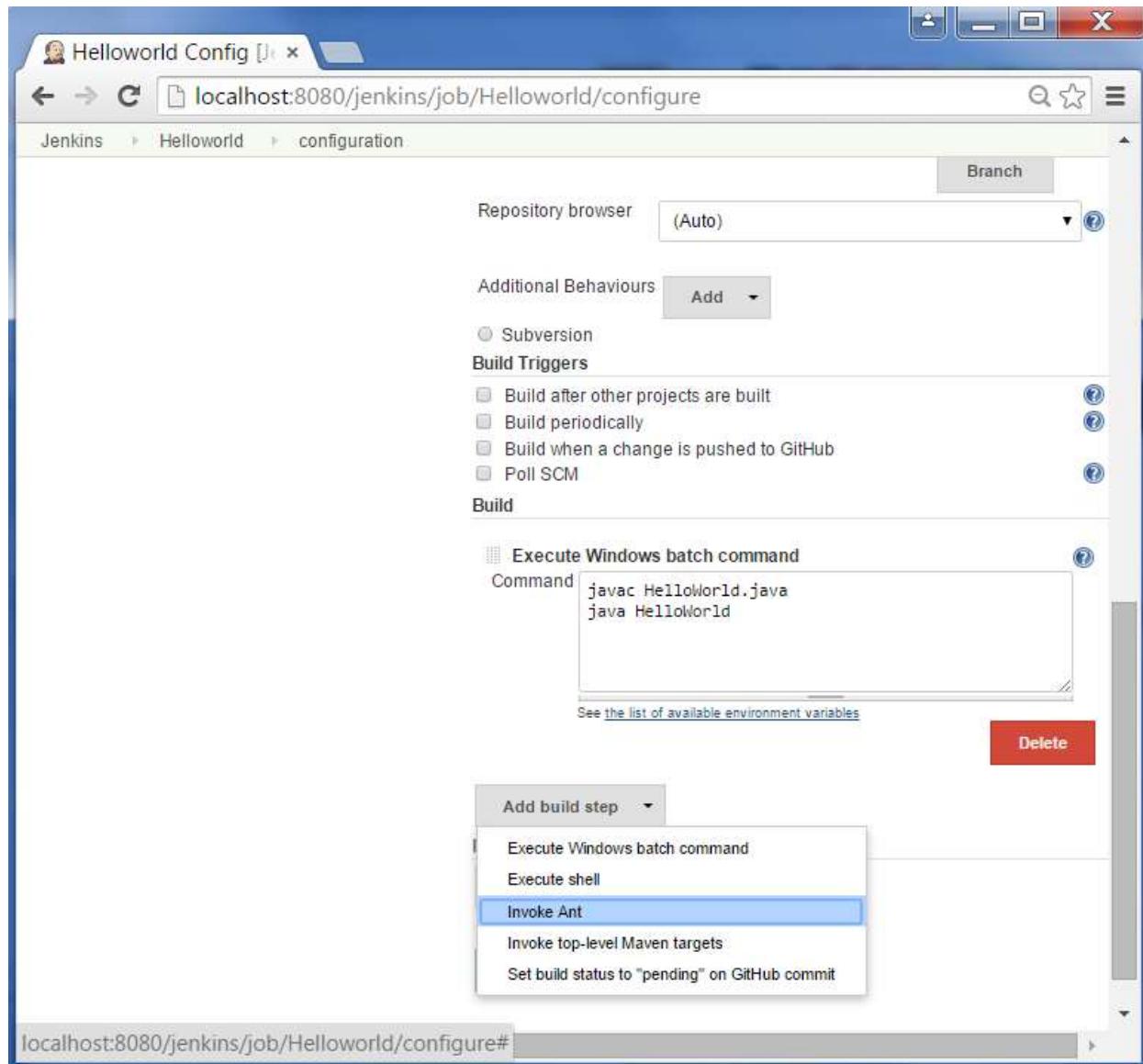
The following example will consider

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

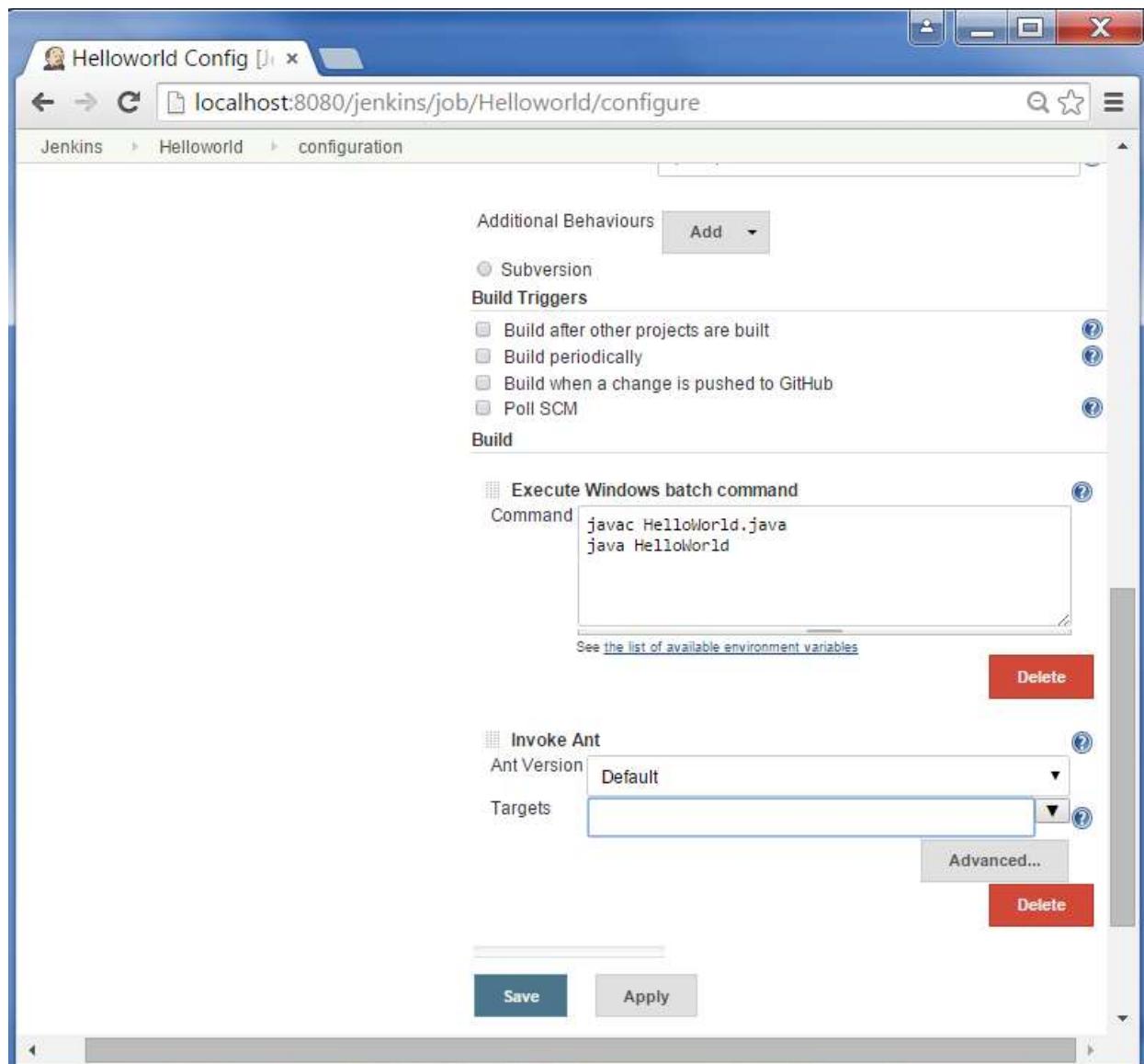
Step 1 : Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (showing 'master' with 1 idle and 2 idle executors, and 'build_slave' which is offline). The main area displays a table of projects. One project, 'Helloworld', is selected, showing its last build details: '5 sec - #11' (Last Success), '2 days 23 hr - #10' (Last Failure), and '3.2 sec' (Last Duration). A context menu is open over this project, with 'Configure' highlighted in blue. Other options in the menu include 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'RSS for failures' and 'RSS for just latest builds'. At the bottom, the URL is 'localhost:8080/jenkins/job/.../configure' and the page footer says 'Page generated: Oct 15, 2015 10:23:01 PM REST API Jenkins ver. 1.609.3'.

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



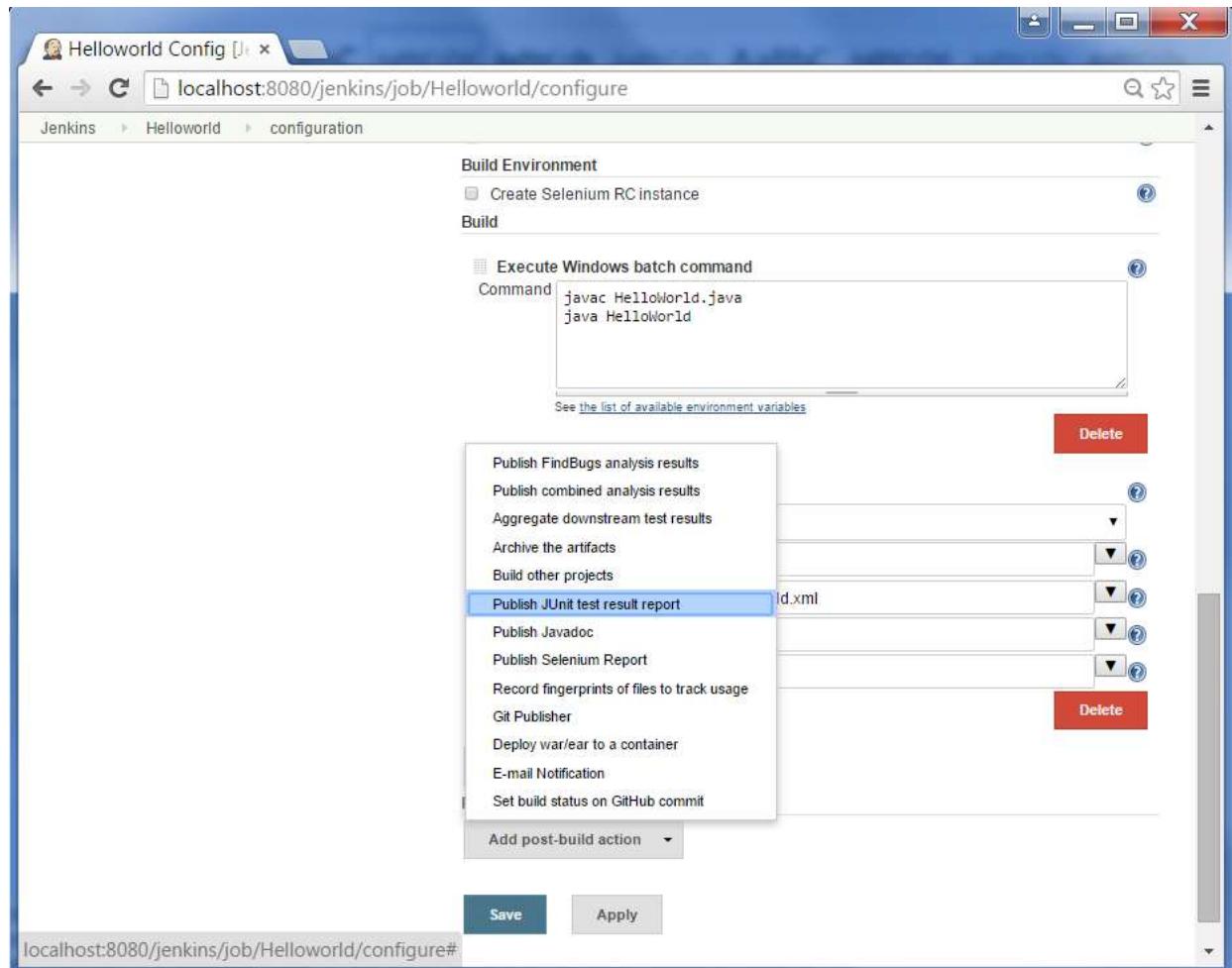
Step 3 : Click on the Advanced button.



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

The screenshot shows the Jenkins configuration interface for a job named "Helloworld". The "Build Environment" section contains a "Create Selenium RC instance" checkbox and a "Build" section with an "Execute Windows batch command" step. The command is set to "javac HelloWorld.java" and "java HelloWorld". Below this is an "Invoke Ant" step with "Ant Version" set to "NewHome", "Targets" empty, "Build File" set to "E:\Java\HelloWorldTest\build.xml", and "Properties" and "Java Options" both empty. A "Delete" button is visible next to the Ant step. At the bottom, there is a "Post-build Actions" section with a "Publish JUnit test result report" step, where the "Test report XMLs" field is set to "Reports*.xml". There are "Save" and "Apply" buttons at the bottom of the configuration panel.

Step 5 : Next click the option to Add post-build option and choose the option of "Publish Junit test result report"



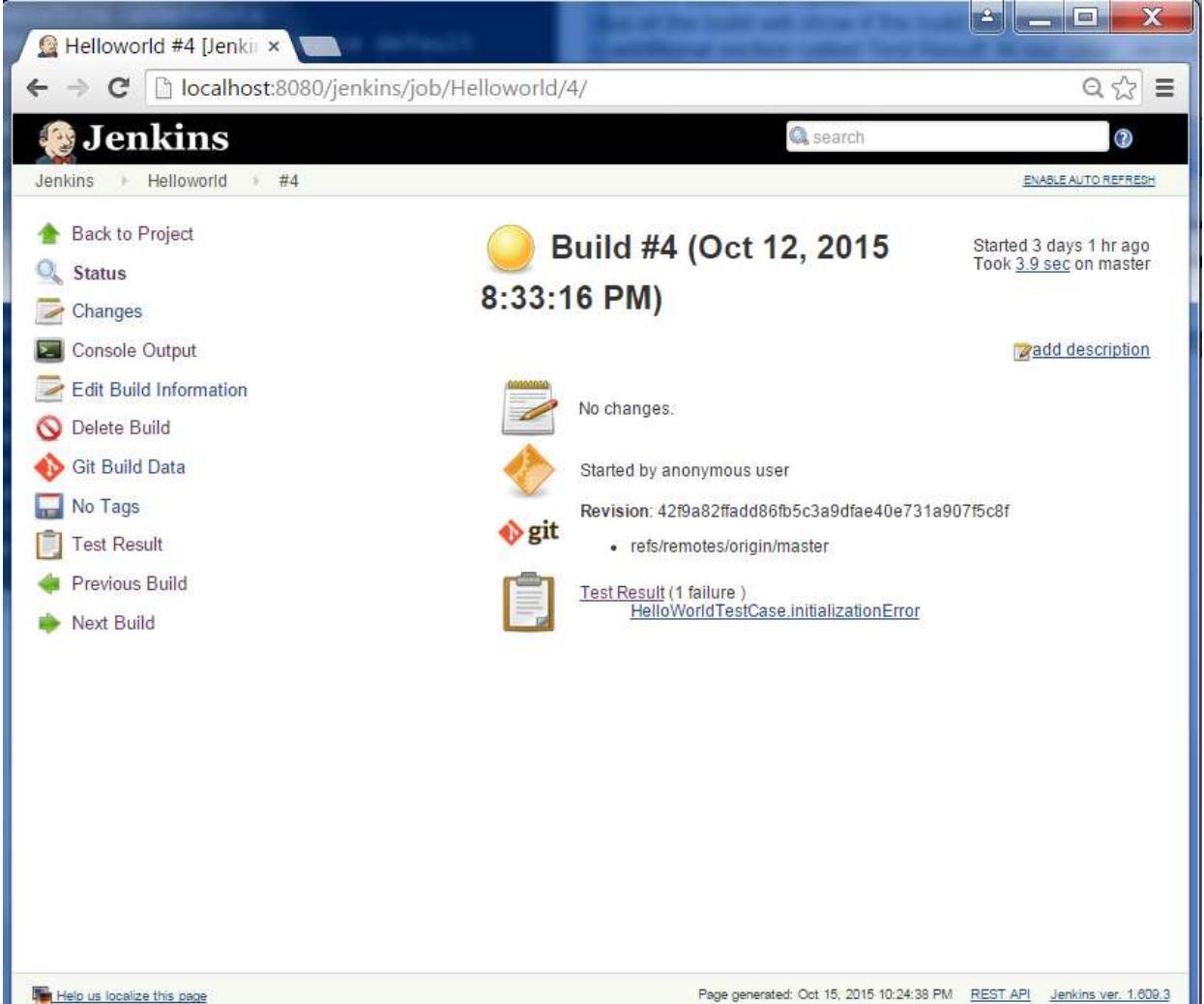
Step 6 : In the Test reports XML's, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which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

Once done, click the Save option at the end.

The screenshot shows the Jenkins configuration page for the 'Helloworld' job. The top section is titled 'Invoke Ant' with fields for 'Ant Version' (set to 'NewHome'), 'Targets' (empty), 'Build File' (set to 'E:\Java\HelloWorldTestbuild.xml'), 'Properties' (empty), and 'Java Options' (empty). Below this is a 'Delete' button. A 'Post-build Actions' section contains a 'Publish JUnit test result report' step. This step has a 'Test report XMLs' field set to 'Reports*.xml'. A tooltip for 'Fileset "includes"' explains it as specifying raw XML report files. There is also a checked checkbox for 'Retain long standard output/error'. A 'Health report amplification factor' field is set to '1.0'. A note below states '1% failing tests scores as 99% health. 5% failing tests scores as 95% health'. Below the actions are 'Add post-build action' and 'Delete' buttons. At the bottom are 'Save' and 'Apply' buttons. The footer includes links for localization, page generation time (Oct 12, 2015 8:37:35 PM), REST API, and Jenkins version (1.609.3).

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

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



The screenshot shows the Jenkins interface for a build named "Helloworld #4". The main title is "Build #4 (Oct 12, 2015) 8:33:16 PM". Key details include:

- Status: Started 3 days 1 hr ago, Took 3.9 sec on master.
- No changes.
- Started by anonymous user.
- Revision: 42f9a82ffadd86fb5c3a9dfa40e731a907f5c8f
refs/remotes/origin/master
- Test Result (1 failure)
[HelloWorldTestCase.initializationError](#)

The left sidebar lists various Jenkins management options: 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 the Helloworld #4 Test build. The left sidebar contains links like Back to Project, Status, Changes, Console Output, Edit Build Information, History, Git Build Data, No Tags, Test Result (which is selected), and Previous Build. The main content area has a title 'Test Result' and a message '1 failures'. Below it is a table with one row:

Test Name	Duration	Age
HelloWorldTestCase.initializationError	10 ms	1

Below the table is another section titled 'All Tests' with a table:

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
(root)	10 ms	1	+1	0	0	0	0	1	+1

At the bottom of the page are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 8:45:49 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

10. Jenkins – Automated Testing

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

Step 1 : Go to Manage Plugins.

The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected), and Credentials. Below that are sections for Build Queue (empty) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Manage Jenkins". It has two warning messages at the top:

- Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat i18n](#) for more details.
- Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.

Below these are several management options with icons:

- Configure System: Configure global settings and paths.
- Configure Global Security: Secure Jenkins, define who is allowed to access/use the system.
- Reload Configuration from Disk: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**
- System Information: Displays various environmental information to assist trouble-shooting.
- System Log: System log captures output from `java.util.logging` 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.

The screenshot shows the Jenkins Plugin Manager interface. The URL in the browser is `localhost:8080/jenkins/pluginManager/available`. The 'Available' tab is selected. A search bar at the top right contains the text 'selenium'. A table lists several plugins:

Install	Name	Version
<input type="checkbox"/>	Selenium Auto Exec Server(AES) plugin	0.5
<input checked="" type="checkbox"/>	Hudson Seleniumhq plugin	0.4
<input type="checkbox"/>	Selenium HTML report	0.94
<input type="checkbox"/>	TestingBot plugin	1.11
<input type="checkbox"/>	TestLink Plugin	3.10
<input type="checkbox"/>	Nirvana Plugin for Jenkins	1.02.06
<input type="checkbox"/>	Sauce OnDemand plugin	1.141
<input type="checkbox"/>	Selenium Builder plugin	1.14
<input type="checkbox"/>	SeleniumRC plugin	

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

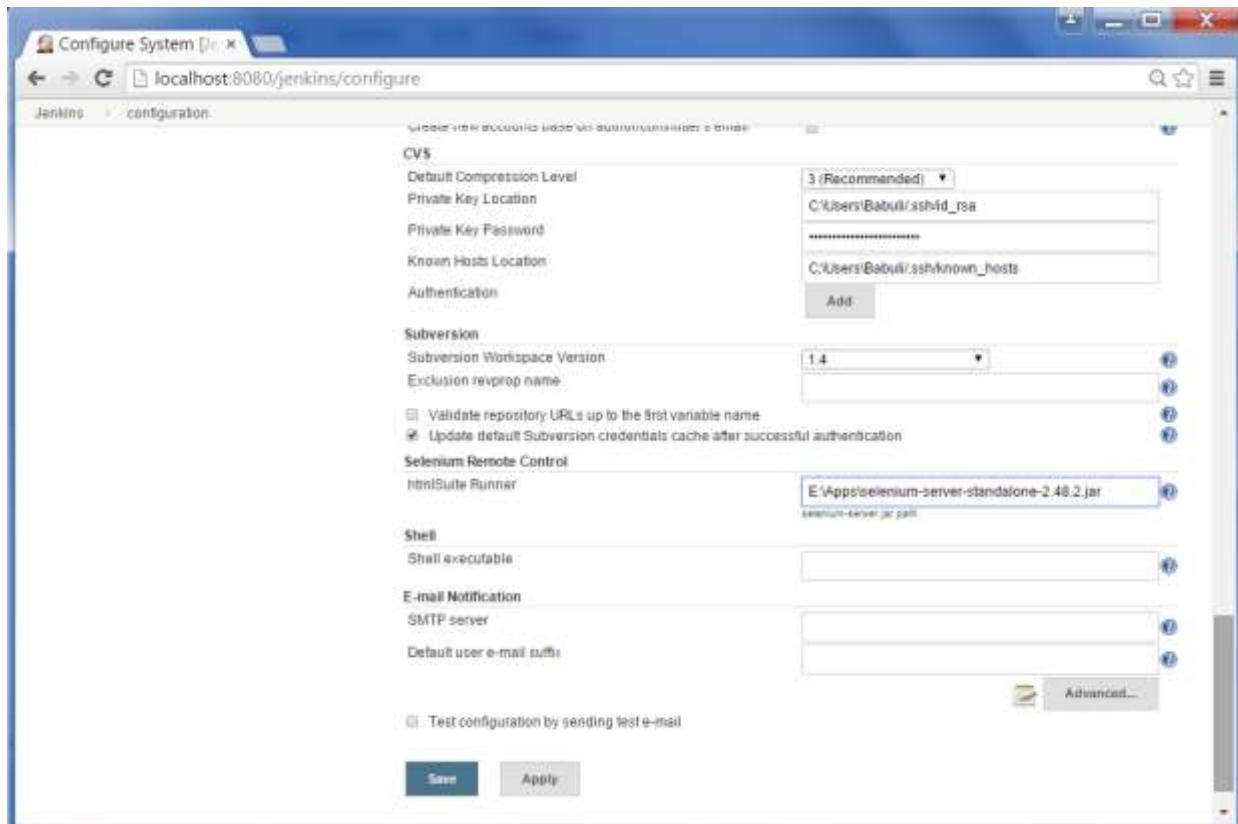
Step 3 : Go to Configure system.

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

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

At the top right, there are 'Setup Security' and 'Dismiss' buttons. A warning message at the top states: 'Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See Containers and Tomcat i18n for more details.' Below it says 'Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.'

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



Note: The selenium jar file can be downloaded from the location <http://www.seleniumhq.org/download/>

Click on the download for the Selenium standalone server.

Selenium Downloads

Latest Releases

Previous Releases

Source Code

Maven Information

Donate to Selenium

with PayPal

Donate

through sponsorship

You can [sponsor the Selenium project](#) if you'd like some public recognition of your generous contribution.

Selenium Sponsors

See who [supports the Selenium project](#).

SeleniumHQ Browser Automation

[edit this page](#) [search selenium:](#) [Go](#)

[Projects](#) [Download](#) [Documentation](#) [Support](#) [About](#)

Downloads

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

Selenium Standalone Server

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

Download version [2.48.2](#)

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

The Internet Explorer Driver Server

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

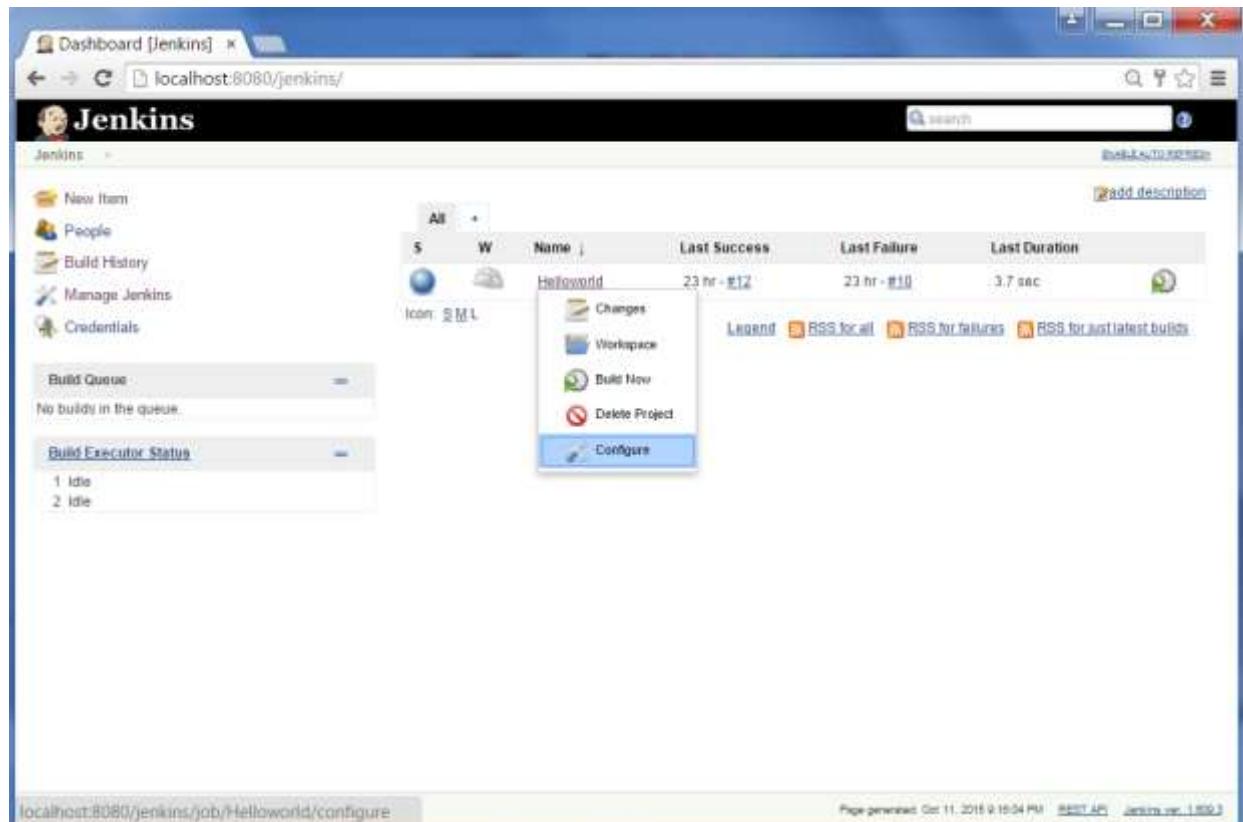
Download version 2.48.0 for (recommended) [32 bit Windows IE](#) or [64 bit Windows IE CHangelog](#)

Selenium Client & WebDriver Language Bindings

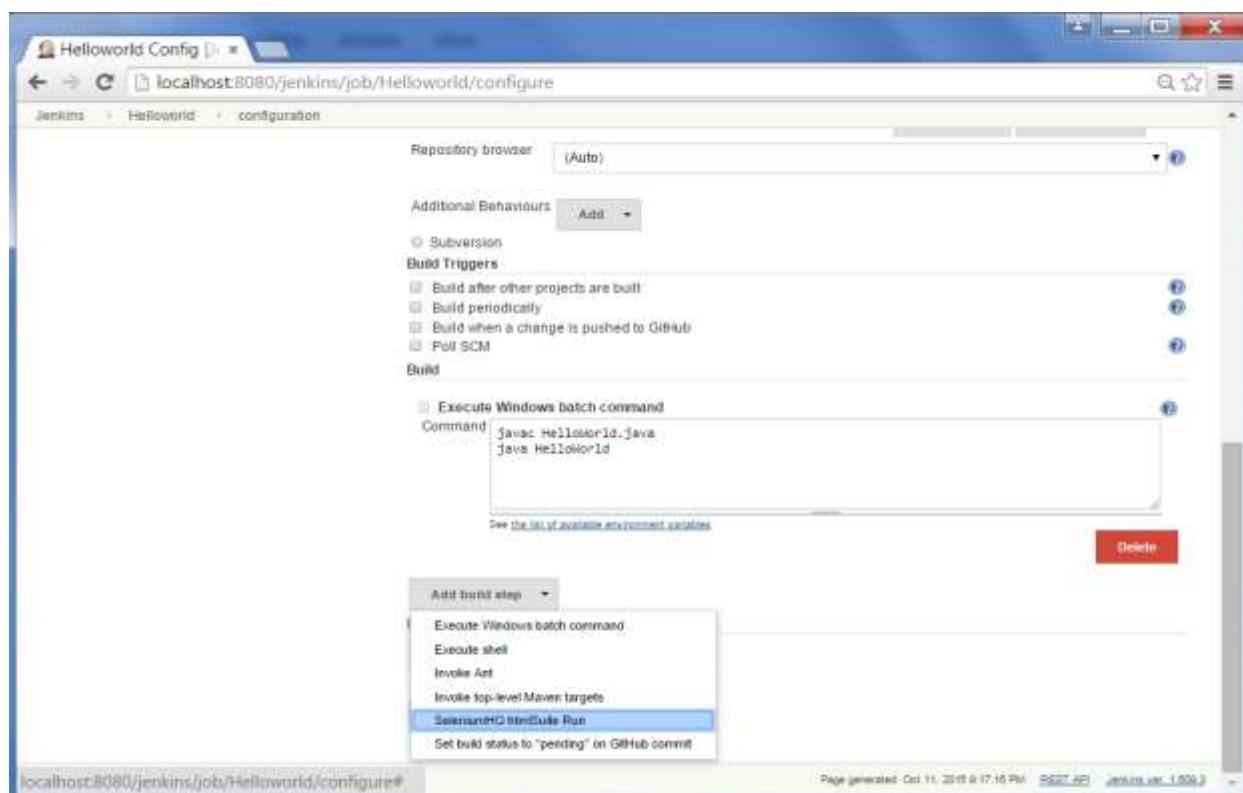
In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

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

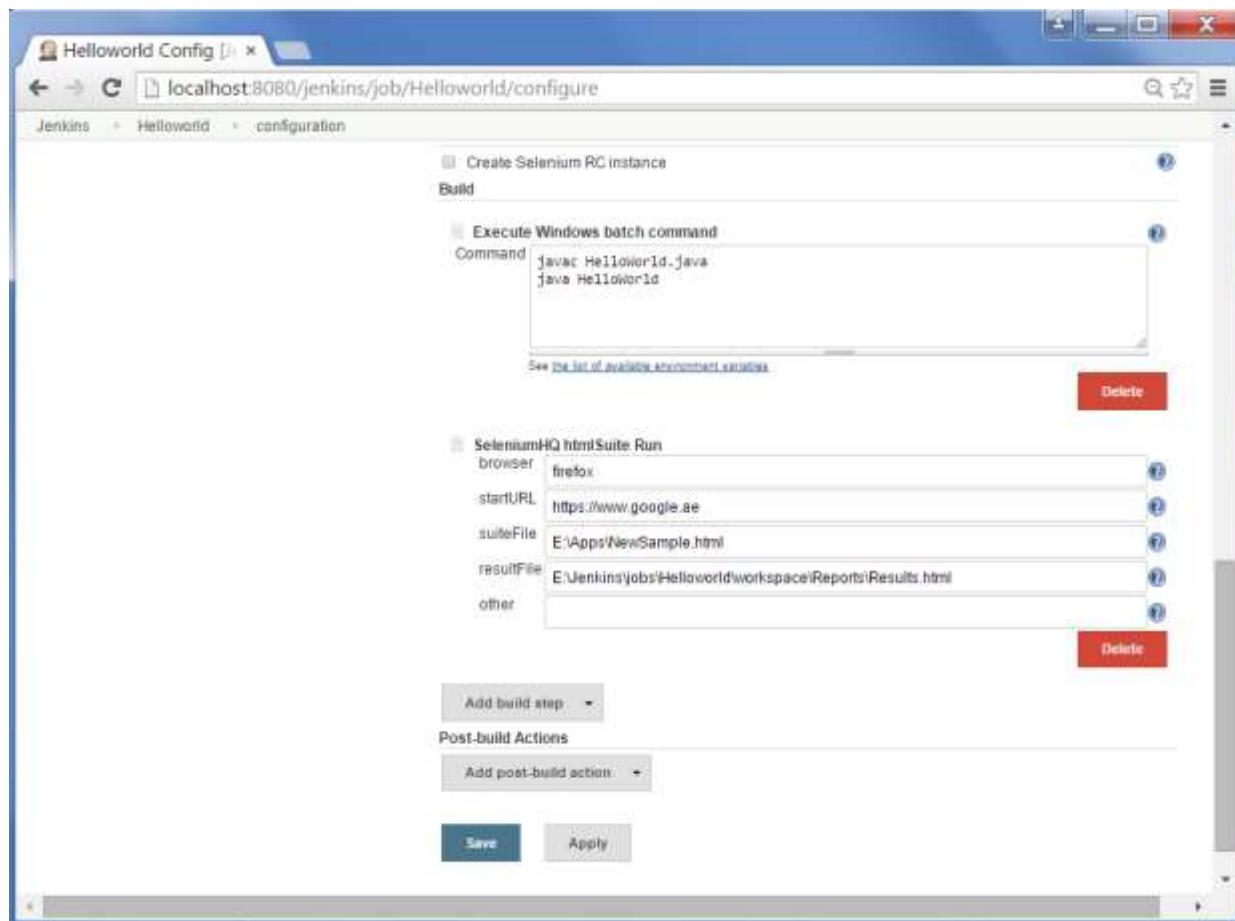
Step 5 : Go back to your dashboard and click on the Configure option for the HelloWorld project.



Step 6 : Click on Add build step and choose the option of "SelecniumHQ htmlSuite Run"



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



11. Jenkins – Notification

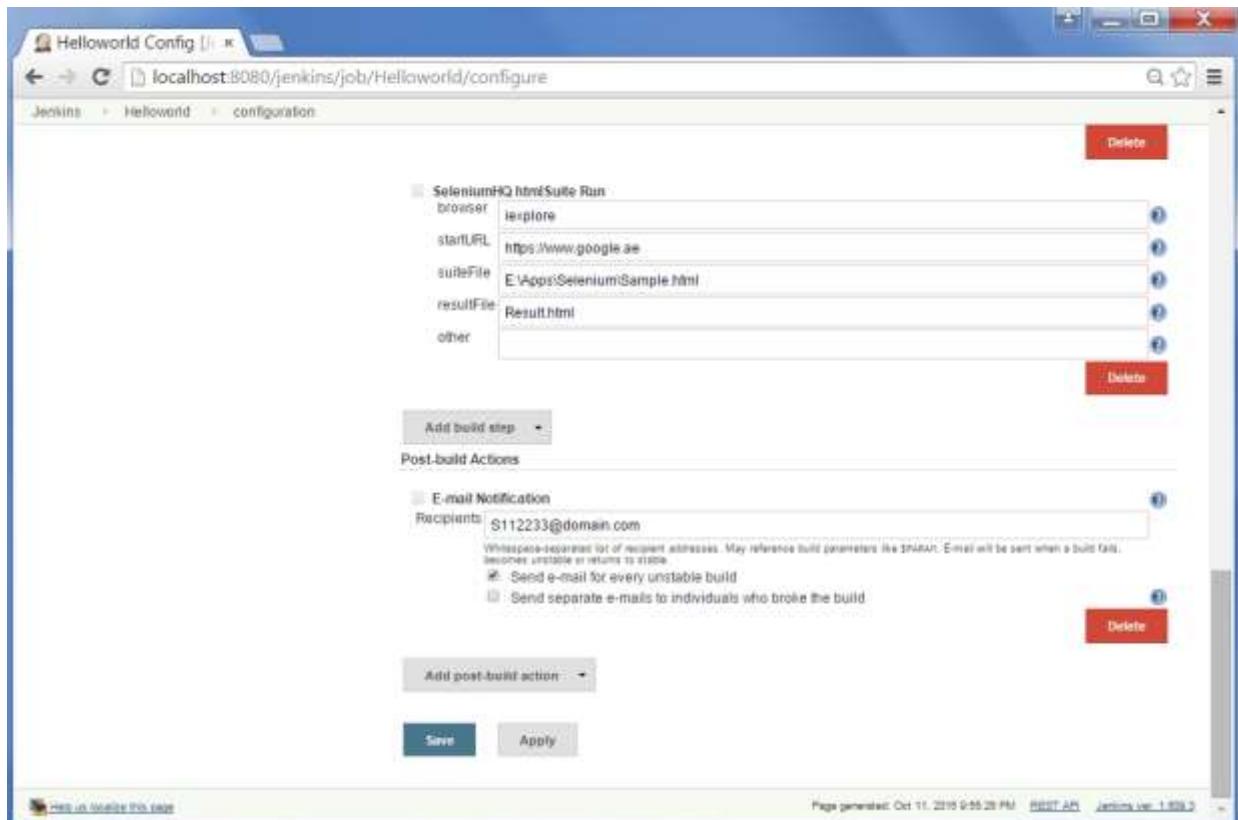
Jenkins comes with an out of box facility to add an email notification for a build project.

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.

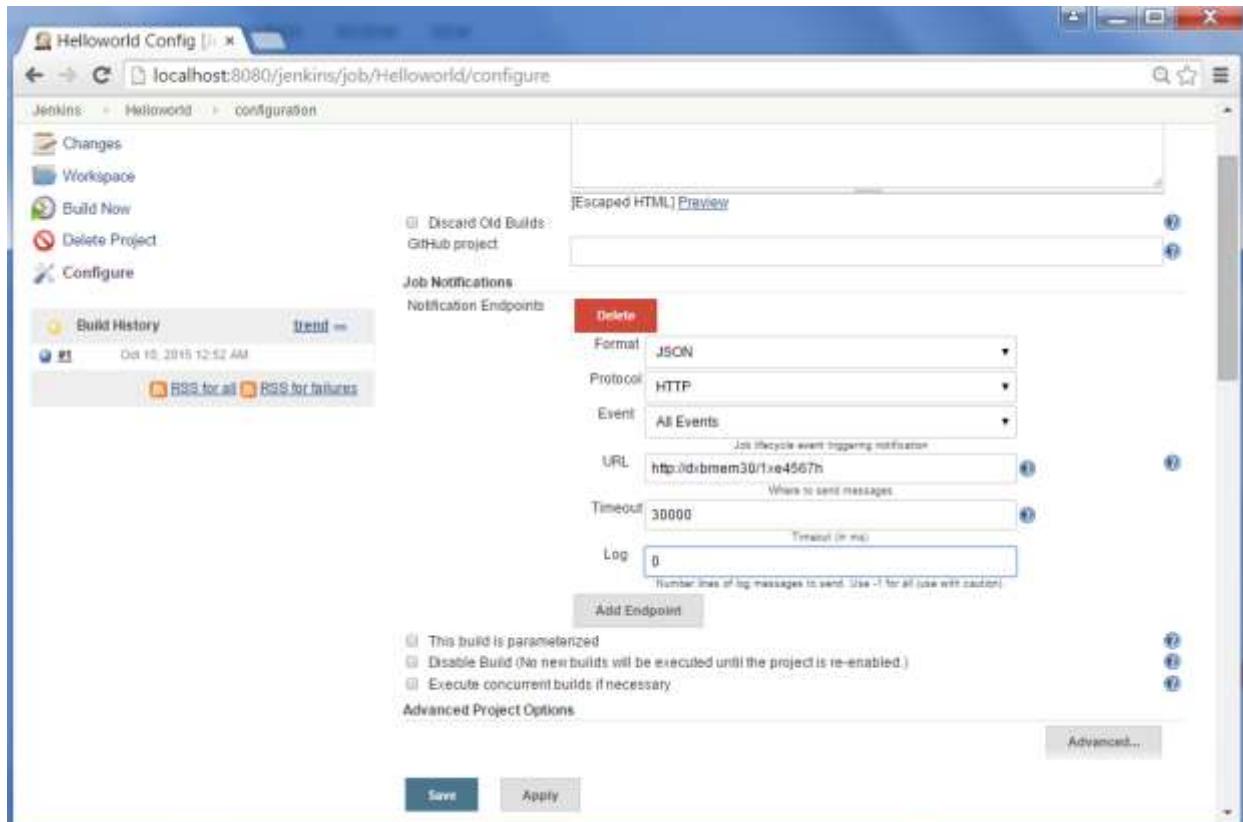
The screenshot shows the Jenkins 'Configure System' page at localhost:8080/jenkins/configure. The 'E-mail Notification' section is highlighted, showing the configuration for an SMTP server (192.168.0.100) and a default user e-mail suffix (@emirates.com). Other sections like Subversion, Selenium Remote Control, and Shell are also visible.

Section	Setting	Value
E-mail Notification	SMTP server	192.168.0.100
E-mail Notification	Default user e-mail suffix	@emirates.com

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



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



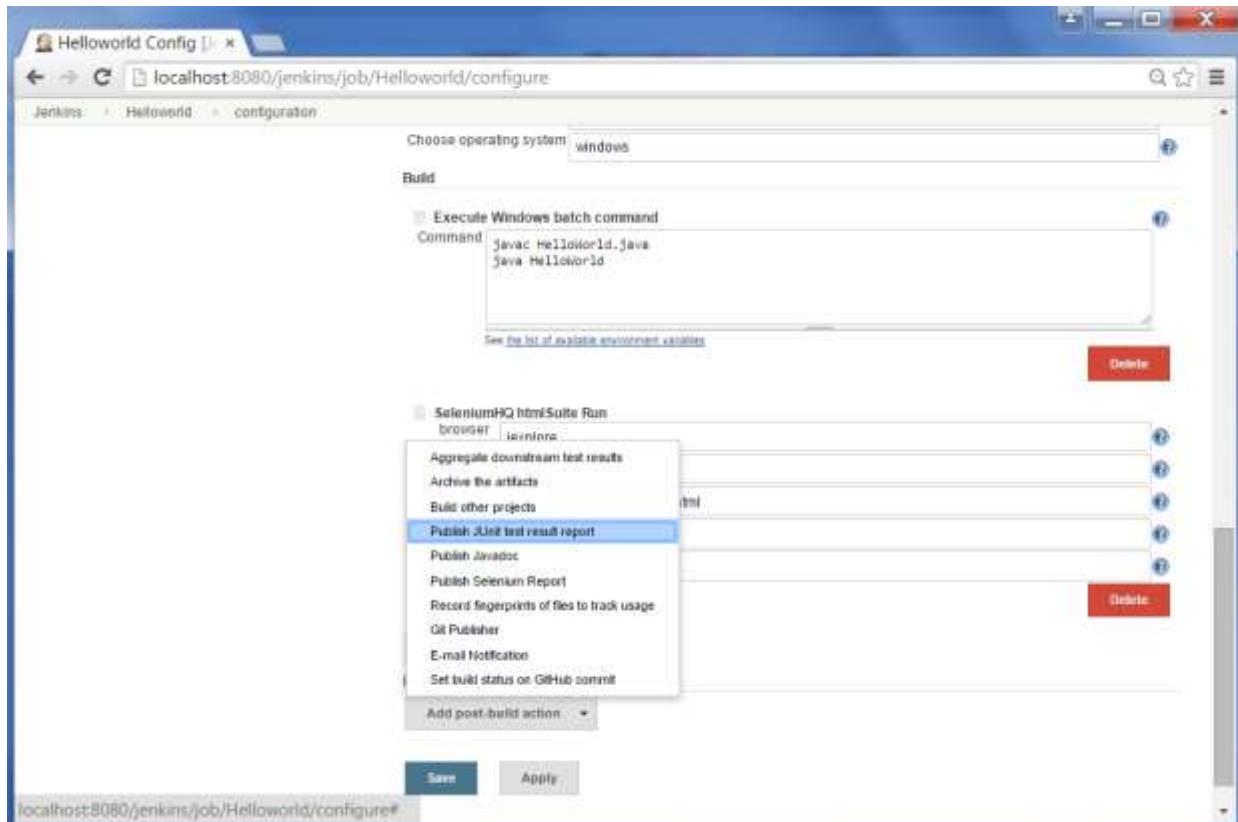
Here are the details of each option:

- "**Format**" : This is the notification payload format which can either be JSON or XML.
- "**Protocol**": protocol to use for sending notification messages, HTTP, TCP or UDP.
- "**Event
- "**URLhttp://host" for HTTP protocol, and "host:port" for TCP and UDP protocols.**
- "**Timeout****

12. 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.



13. 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 is titled 'Static Code Analysis Plug-ins' and shows the 'analysis-core' plugin. It includes a 'Plugin Information' table with details like Plugin ID (analysis-core), Latest Release (1.74), Latest Release Date (Sep 07, 2015), Required Core Dependencies (ant, token-macro, maven-plugin, matrix-project, dashboard-view), and a 'Changes' section with GitHub links. Below this is a 'Usage' section with a line graph titled 'analysis-core - installations' showing the number of installations over time from October 2014 to September 2015. The graph shows a steady increase from approximately 25,000 to 30,000 installations. To the right of the graph is a 'Installations' table listing the count for each month.

Month	Installations
2014-Oct	26415
2014-Nov	26297
2014-Dec	26166
2015-Jan	27193
2015-Feb	27184
2015-Mar	28580
2015-Apr	28399
2015-May	28172
2015-Jun	28906
2015-Jul	29519
2015-Aug	29170
2015-Sep	29058

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)

14. Jenkins – Distributed Builds

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

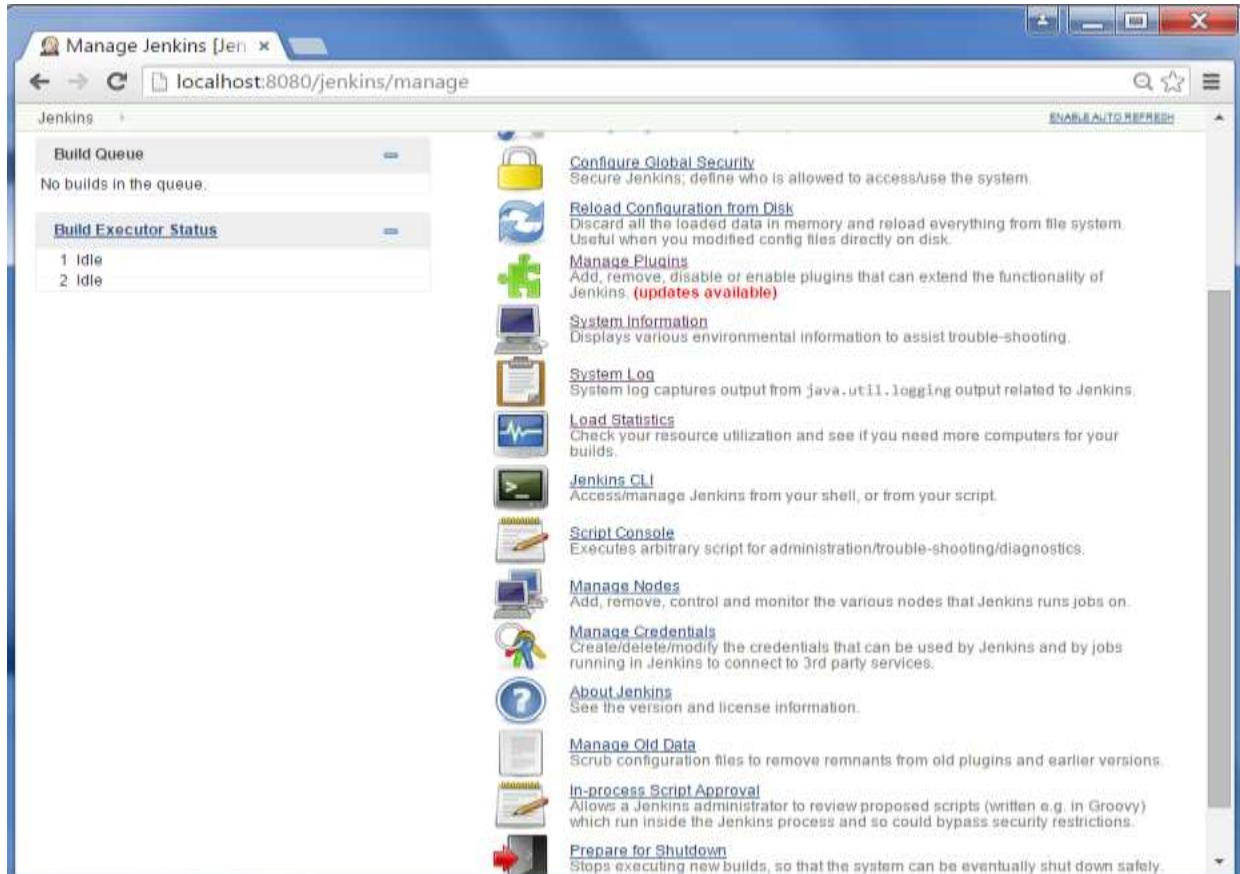
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

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

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

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



Step 2: Click on New Node

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and a 'Refresh status' button. The main content area displays a table of nodes. The table has columns for 'S', 'Name', 'Architecture', 'Clock Difference', 'Free Disk Space', 'Free Swap Space', and 'Free'. There is one entry in the table:

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

Below the table, there are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:23:44 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

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

The screenshot shows the Jenkins 'New Node' configuration page. The 'Node name' field contains 'build_slave'. The 'Dumb Slave' radio button is selected, with a tooltip explaining it adds a plain, dumb slave to Jenkins. Below the form are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom right are links for localization, page generation details, REST API, and Jenkins version.

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.

The screenshot shows the Jenkins 'build_slave Configuration' page. The URL is `localhost:8080/jenkins/computer/build_slave/configure`. The left sidebar shows links like Back to List, Status, Delete Slave, Configure, Build History, Load Statistics, and Log. The main form has the following fields:

Name	build_slave
Description	
# of executors	1
Remote root directory	D:\Jenkins
Labels	New_Slave
Usage	Utilize this node as much as possible
Launch method	Let Jenkins control this Windows slave as a Windows service

A note below the launch method says: "This launch method relies on DCOM and is often associated with subtle problems. Consider using Launch slave agents using Java Web Start instead, which also permits installation as a Windows service but is generally considered more reliable." Below this are fields for Administrator user name (admin), Password (redacted), Host (dxbmem30), and Run service as (Use Local System User). There is an "Advanced..." button. Under Availability, it says "Keep this slave on-line as much as possible". At the bottom, there is a "Node Properties" section with checkboxes for Environment variables and Tool Locations, and a "Save" button.

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

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and a 'Refresh status' button. The main content area has three sections: 'Build Queue' (No builds in the queue), 'Build Executor Status' (master: 1 Idle, 2 Idle; build_slave: (offline)), and a 'Help us localize this page' link. On the right, there's a table showing node details:

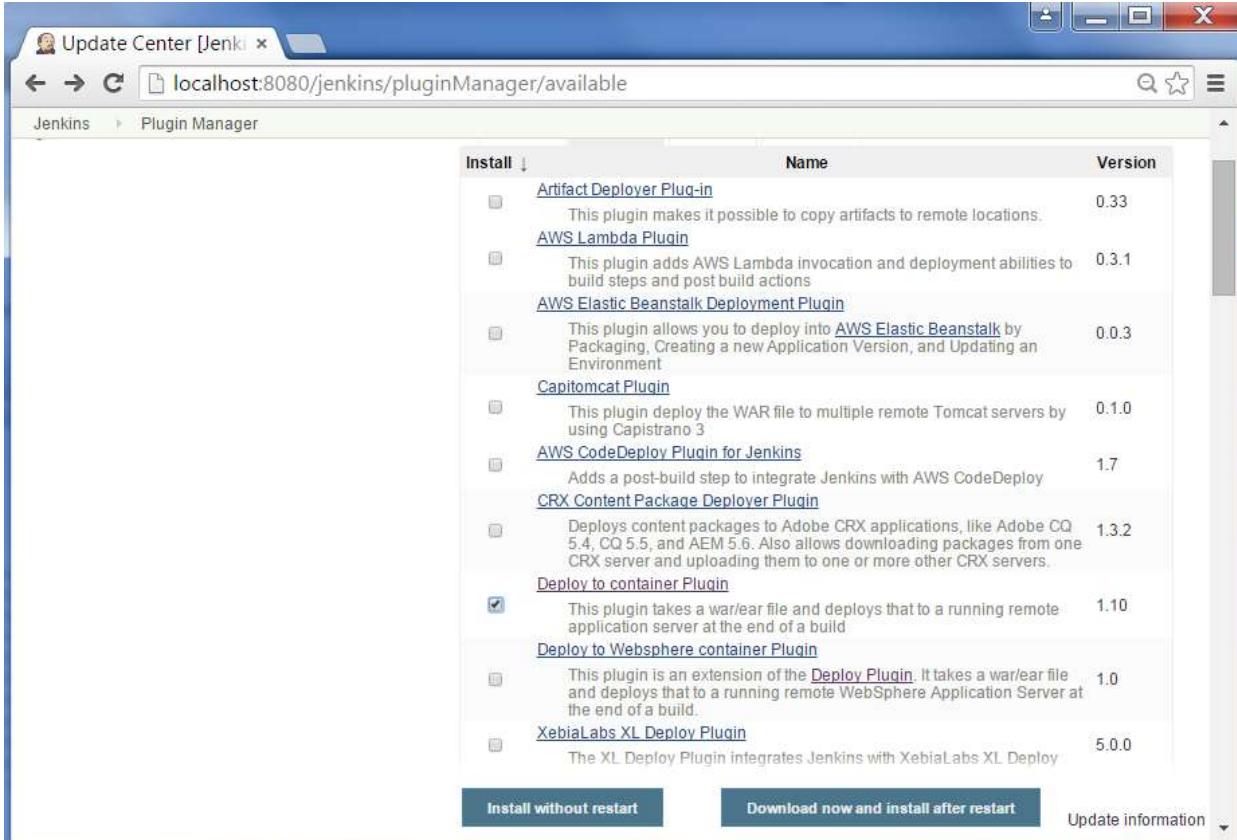
\$	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space
	build_slave		N/A	N/A	N/A	
	master	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	229.8
		Data obtained	3 ms	2 ms	1 ms	11 min

Page generated: Oct 12, 2015 9:31:43 PM | REST API | Jenkins ver. 1.626.3

15. Jenkins – Automated Deployment

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

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



The screenshot shows the Jenkins Update Center interface. The URL in the address bar is `localhost:8080/jenkins/pluginManager/available`. The page displays a list of available Jenkins plugins, with the 'Deploy to container Plugin' checked for installation. Other visible plugins include 'Artifact Deployer Plug-in', 'AWS Lambda Plugin', 'AWS Elastic Beanstalk Deployment Plugin', 'Capitomcat Plugin', 'AWS CodeDeploy Plugin for Jenkins', 'CRX Content Package Deployer Plugin', 'Deploy to container Plugin' (selected), 'Deploy to Websphere container Plugin', and 'XebiaLabs XL Deploy Plugin'. At the bottom, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Update information'.

Install ↓	Name	Version
<input type="checkbox"/>	Artifact Deployer Plug-in This plugin makes it possible to copy artifacts to remote locations.	0.33
<input type="checkbox"/>	AWS Lambda Plugin This plugin adds AWS Lambda invocation and deployment abilities to build steps and post build actions	0.3.1
<input type="checkbox"/>	AWS Elastic Beanstalk Deployment Plugin This plugin allows you to deploy into AWS Elastic Beanstalk by Packaging, Creating a new Application Version, and Updating an Environment	0.0.3
<input type="checkbox"/>	Capitomcat Plugin This plugin deploys the WAR file to multiple remote Tomcat servers by using Capistrano 3	0.1.0
<input type="checkbox"/>	AWS CodeDeploy Plugin for Jenkins Adds a post-build step to integrate Jenkins with AWS CodeDeploy	1.7
<input type="checkbox"/>	CRX Content Package Deployer Plugin Deploys content packages to Adobe CRX applications, like Adobe CQ 5.4, CQ 5.5, and AEM 5.6. Also allows downloading packages from one CRX server and uploading them to one or more other CRX servers.	1.3.2
<input checked="" type="checkbox"/>	Deploy to container Plugin This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build	1.10
<input type="checkbox"/>	Deploy to Websphere container Plugin This plugin is an extension of the Deploy Plugin . It takes a war/ear file and deploys that to a running remote WebSphere Application Server at the end of a build.	1.0
<input type="checkbox"/>	XebiaLabs XL Deploy Plugin The XL Deploy Plugin integrates Jenkins with XebiaLabs XL Deploy	5.0.0

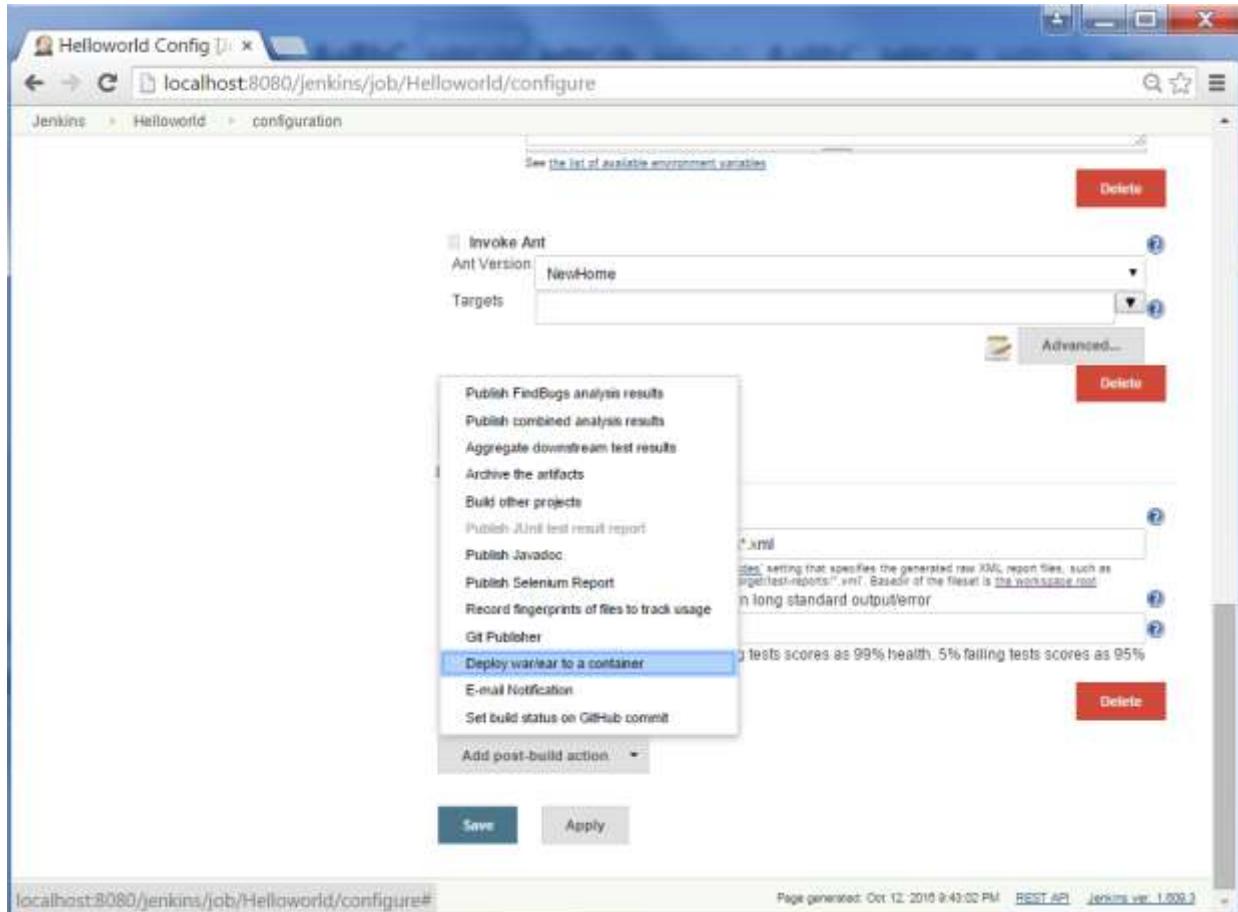
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

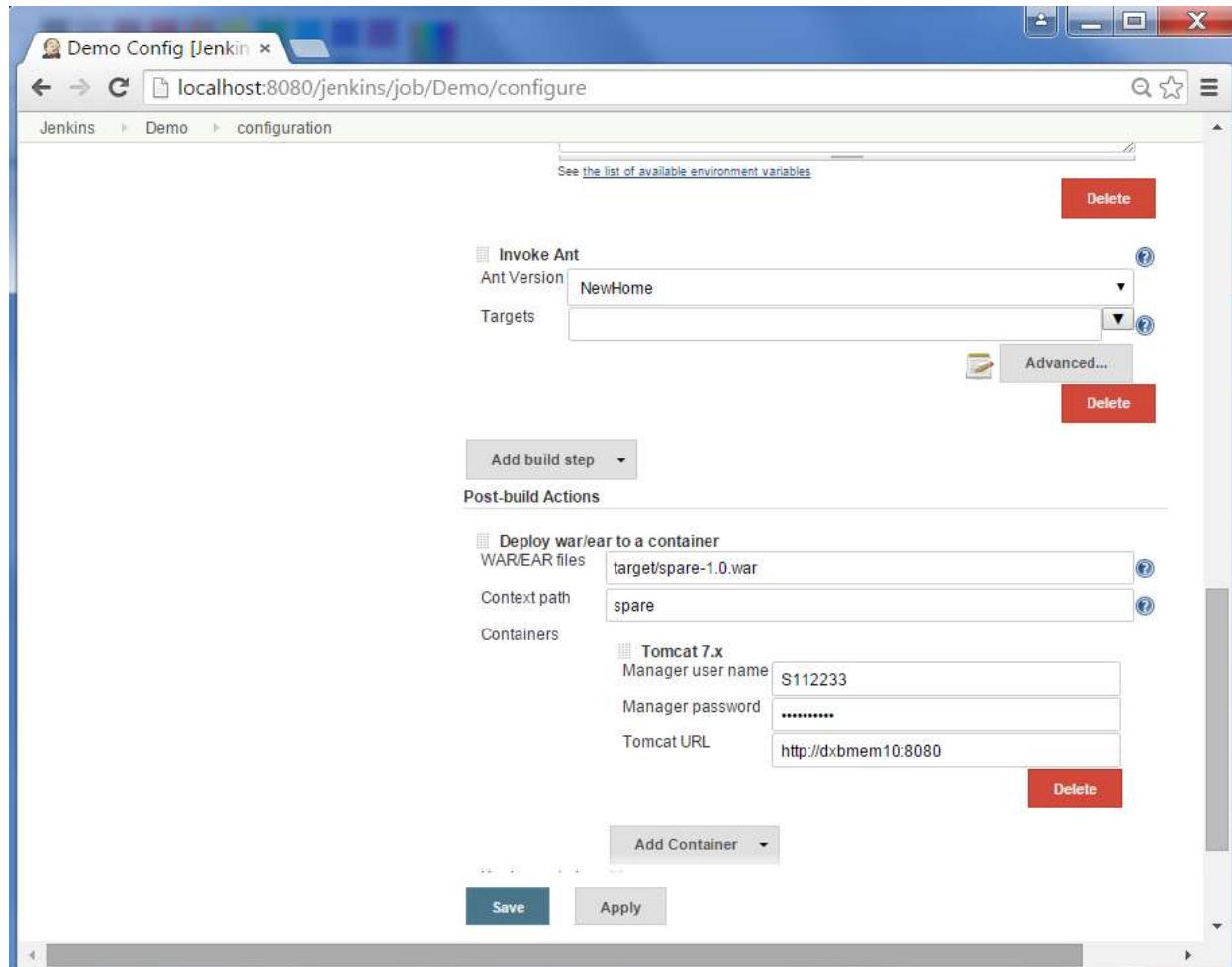
JBoss 3.x/4.x

Glassfish 2.x/3.x

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



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



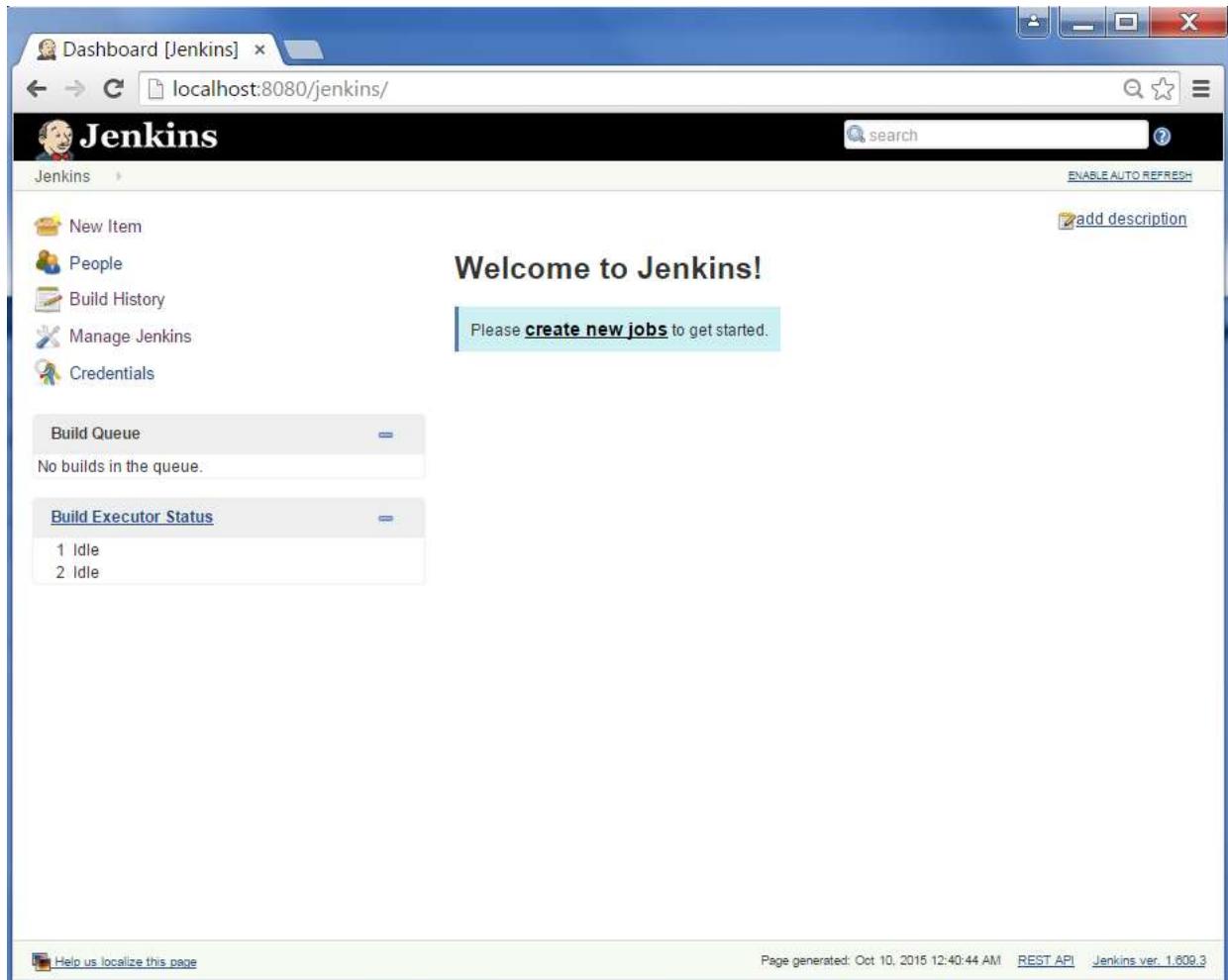
16. Jenkins – Metrics and Trends

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

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

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

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



Step 2 : Go to the Manage Plugins option.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), and 'Credentials'. Below this are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains several configuration options. At the top right of this area, there are two buttons: 'Setup Security' and 'Dismiss'. Below these are several items with icons and descriptions:

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

Step 3 : Go to the Available tab and search for the plugin 'Build History Metrics plugin' and choose to 'install without restart'.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jen]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main header has tabs: "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the filter "build-history-metrics-plugin". Below the tabs is a table with columns: "Name" and "Version". One row in the table is highlighted, showing "Build History Metrics plugin" and "1.2". A note below the table says "Provides build metrics that encompass the history of all the runs". At the bottom of the table are two buttons: "Install without restart" (highlighted in blue) and "Download now and install after restart". To the right of these buttons is the text "Update information obtained: 2 mi". At the very bottom of the page, there is a footer with links: "Help us localize this page", "Page generated: Oct 24, 2015 3:53:24 PM", "REST API", and "Jenkins ver. 1.609.3".

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

The screenshot shows the Jenkins Update Center interface. The title bar reads "Update Center [Jen...]" and the address bar shows "localhost:8080/jenkins/updateCenter/". The main content area has a header "Installing Plugins/Upgrades". On the left, there's a sidebar with links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area includes a "Preparation" section with a bulleted list: "Checking Internet connectivity", "Checking update center connectivity", and "Success". Below this, a specific plugin, "Build History Metrics plugin", is shown with a "Success" status indicator. At the bottom, there are two green checkmark icons with instructions: "Go back to the top page (you can start using the installed plugins right away)" and "Restart Jenkins when installation is complete and no jobs are running". The footer contains links for "Help us localize this page", "Page generated: Oct 24, 2015 3:03:57 PM", "REST API", and "Jenkins ver. 1.600.3".

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

The screenshot shows the Jenkins interface for the 'Helloworld' project. On the left, there is a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. Below this is the 'Build History' section, which lists the last 12 builds. The history shows builds #12 (failed), #11, #10, #9, #8, #7, #6, #5, #4, #3, #2, and #1. The most recent build (#12) was successful. There are also links for RSS feeds for all and failures.

The main content area is titled 'Project Helloworld'. It features three buttons: 'Workspace', 'Recent Changes', and 'Disable Project'. To the right of these buttons is a 'add description' link. Below these are three tables showing performance metrics:

	Last 7 Days	0 ms
MTTR	Last 30 Days	23 hr
	All Time	23 hr

	Last 7 Days	0 ms
MTTF	Last 30 Days	2 days 4 hr
	All Time	2 days 4 hr

	Last 7 Days	0 ms
Standard Deviation	Last 30 Days	52 sec
	All Time	52 sec

Below the tables is a section titled 'Permalinks' containing a bulleted list of links to specific builds:

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

At the bottom of the page, there is a link to help localize the page and footer information: 'Help us localize this page', 'Page generated: Oct 24, 2015 3:57:10 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

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

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

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The main content area displays the message "Welcome to Jenkins!" and a note: "Please [create new jobs](#) to get started." On the left sidebar, there are links for "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". Below these are two expandable sections: "Build Queue" (which shows "No builds in the queue.") and "Build Executor Status" (which shows "1 Idle" and "2 Idle"). At the bottom of the page, there are links for "Help us localize this page", "Page generated: Oct 10, 2015 12:40:44 AM", "REST API", and "Jenkins ver. 1.609.3".

Step 2 : Go to the Manage Plugins option

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), and 'Credentials'. Below this are two expandable sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains several configuration options. Two yellow warning messages are displayed at the top:

- Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat i18n](#) for more details.
- Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.

Below these are various management links with icons:

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

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

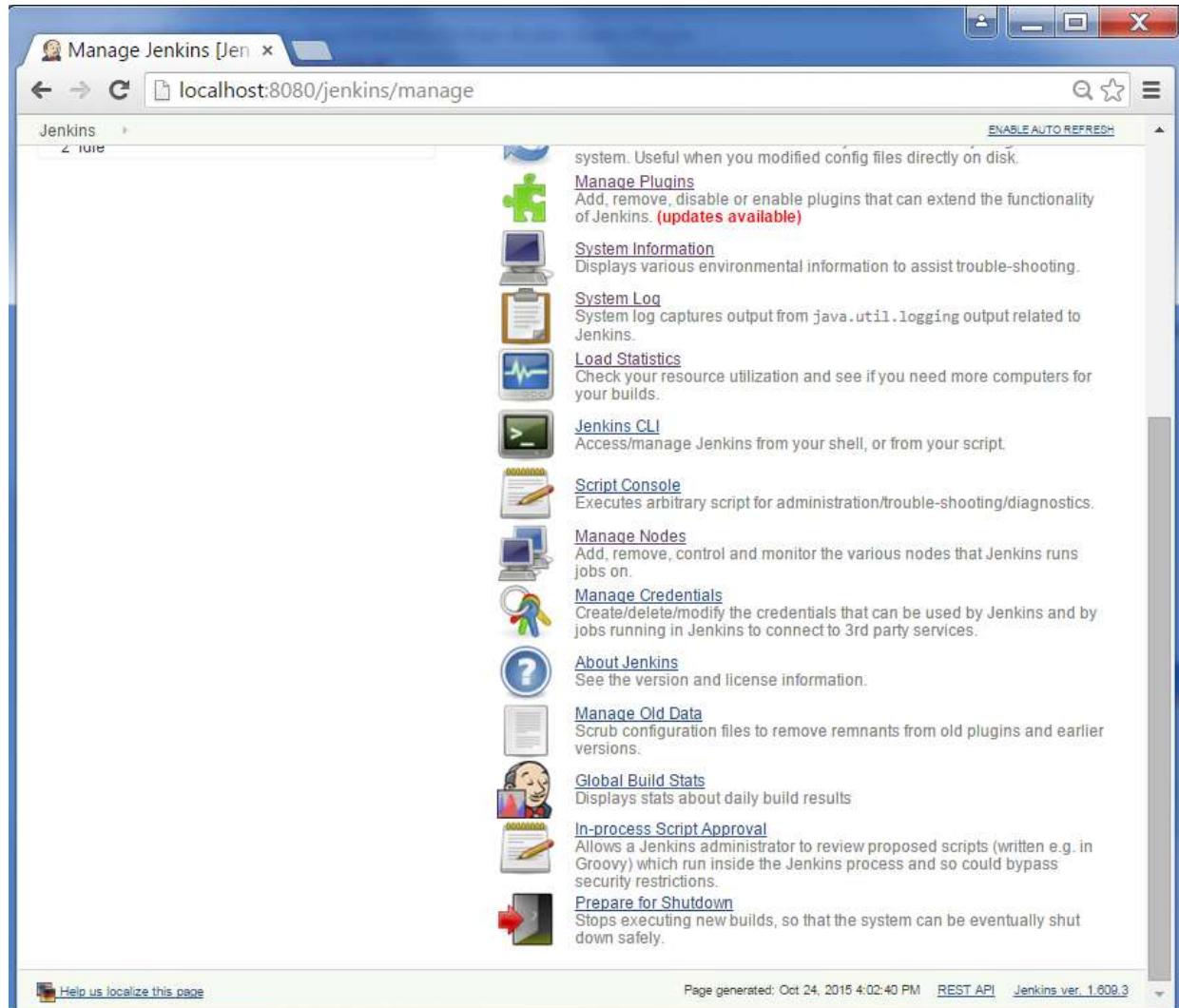
The screenshot shows the Jenkins Plugin Manager interface. The browser address bar indicates the URL is `localhost:8080/jenkins/pluginManager/available`. The main title is "Jenkins" and the sub-page title is "Plugin Manager". On the left, there are links for "Back to Dashboard" and "Manage Jenkins". The top navigation bar has tabs: "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the text "global-build-stats". Below the tabs is a table with columns: "Install ↓", "Name", and "Version". One row in the table is highlighted for the "Hudson global-build-stats plugin", version 1.3. The description for this plugin states: "Global build stats plugin will allow to gather and display global build result statistics. It is a useful tool allowing to display global hudson build trend over time..". At the bottom of the table are two buttons: "Install without restart" (in a blue box) and "Download now and install after restart". To the right of these buttons is a link "Update inform". At the very bottom of the page, there are links for "Help us localize this page", "Page generated: Oct 24, 2015 4:00:23 PM", "REST API", and "Jenkins ver. 1.609.3".

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

The screenshot shows the Jenkins Update Center interface. The title bar says "Update Center [Jenk] x" and the address bar shows "localhost:8080/jenkins/updateCenter/". The main content area has a header "Installing Plugins/Upgrades". On the left, there's a sidebar with links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area shows a section titled "Preparation" with a bulleted list: "Checking internet connectivity", "Checking update center connectivity", and "Success". Below this, it shows "Hudson global-build-stats plugin" with a "Success" status indicator. 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 links for "Help us localize this page", "Page generated: Oct 24, 2015 4:01:04 PM", "REST API", and "Jenkins ver. 1.609.3".

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

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



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

The screenshot shows the Jenkins Global Build Stats page. On the left sidebar, there are links: Back to Dashboard, Create new chart, Manage retention strategies, and Data Initialization. The main content area has a title 'Global Build Stats' with a cartoon character icon. Below it is a 'Statistics' section stating 'No chart configured for the moment ... [Create a new chart configuration](#)'. Under 'Build Results retention strategies', there are three checkboxes: 'Automatically discard results older than 365 days', 'Do not keep build results when they are discarded', and 'Keep existing job results only'. A 'Update retention strategies' button is below these checkboxes. In the 'Data Initialization' section, there is a note: 'Click button below to initialize build statistics. Job results read will be merged with already recorded job results.' followed by a large 'Initialize stats' button. At the bottom of the page, there is a link 'Help us localize this page' and footer text 'Page generated: Oct 24, 2015 4:03:17 PM REST API Jenkins ver. 1.609.3'.

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 interface. On the left, there's a sidebar with links: 'Back to Dashboard', 'Create new chart' (which is highlighted in blue), 'Manage retention strategies', and 'Data Initialization'. The main content area is titled 'Global Build Stats' and features a 'Statistics' section with a message: 'No chart configured for the moment ... [Create a new chart configuration](#)'. Below this is a 'Build Results retention strategies' section with three options: '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 next to the third option. At the bottom of the page, under 'Data Initialization', there's a note: 'Click button below to initialize build statistics. Job results read will be merged with already recorded job results.' followed by a green success message: 'Data successfully initialized !'. A 'Initialize stats' button is present. The bottom of the page includes a 'Help us localize this page' link, a timestamp 'Page generated: Oct 24, 2015 4:03:17 PM', and API links 'REST API' and 'Jenkins ver. 1.609.3'.

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

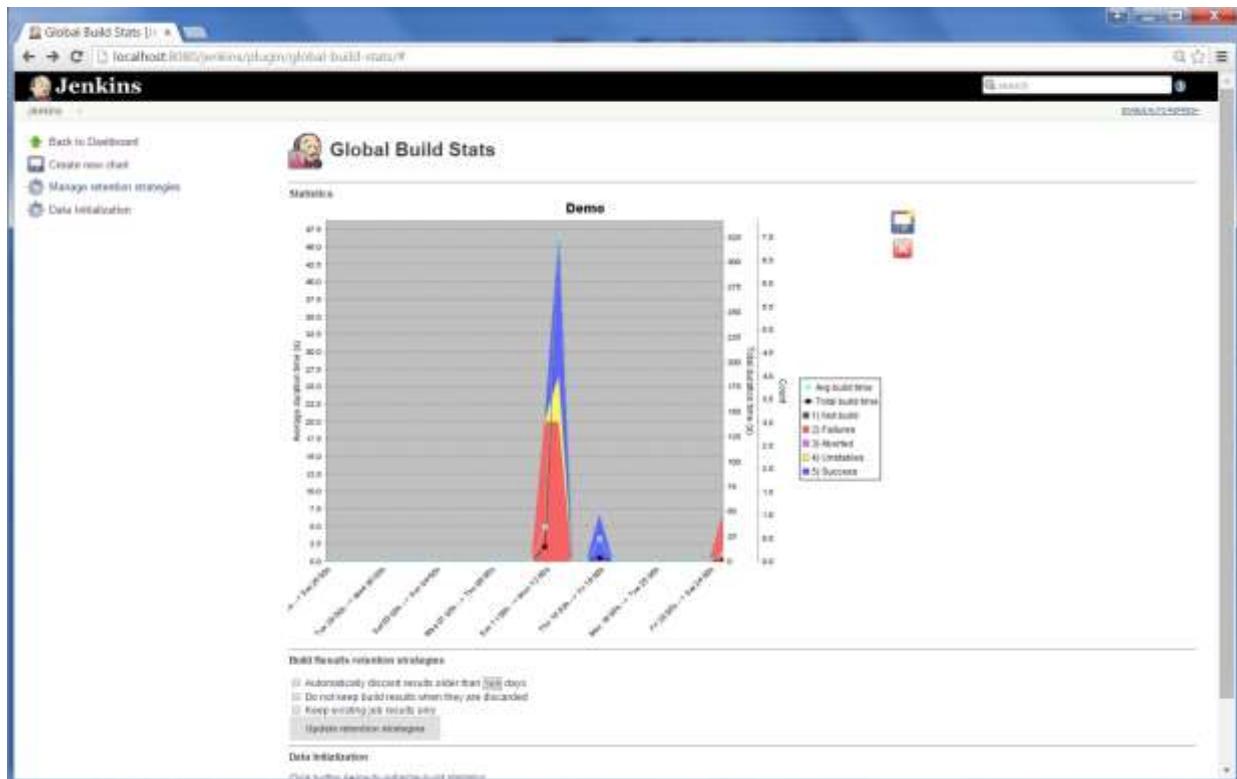
The rest of the information can remain as it is. Once the information is entered, click on Create New chart.

The screenshot shows the Jenkins Global Build Stats configuration interface. At the top, there's a navigation bar with links to 'Back to Dashboard', 'Create new chart', 'Manage retention strategies', and 'Data Initialization'. A search bar and an 'ENABLE AUTO REFRESH' button are also present.

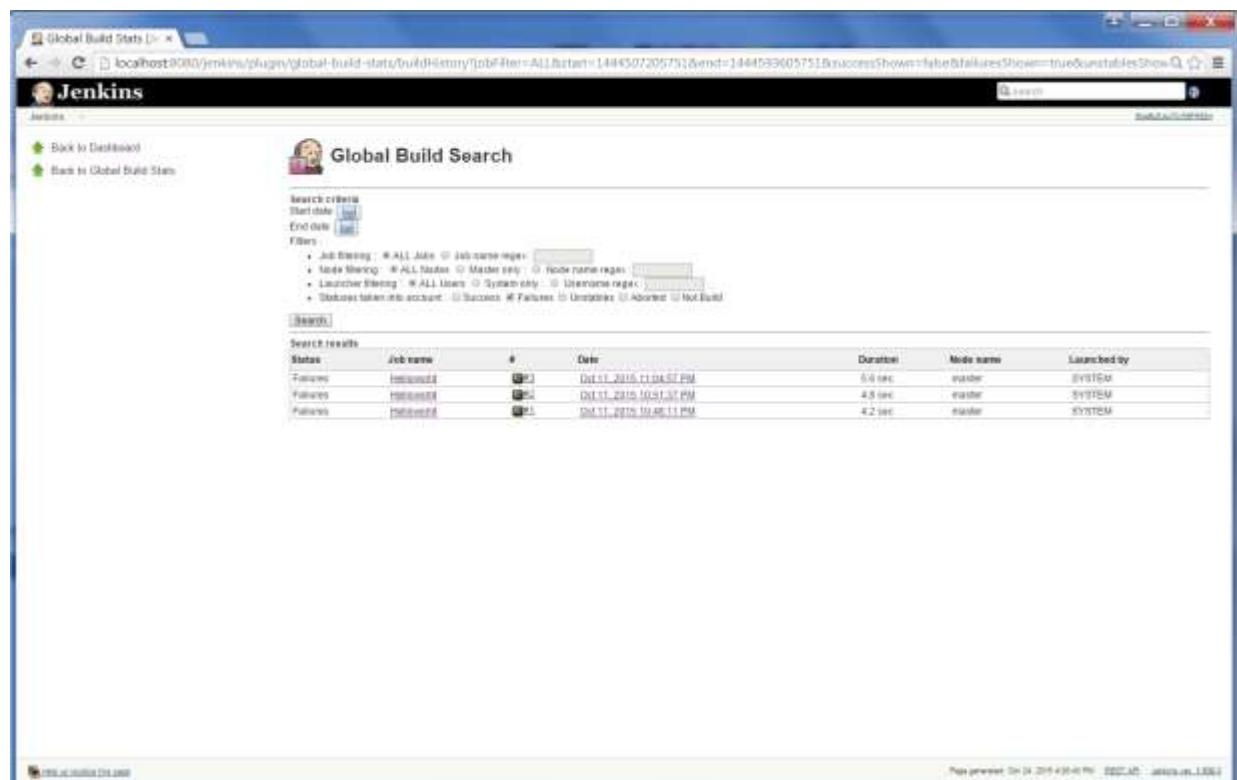
The main title is 'Global Build Stats' with a subtitle 'Statistics'. It displays the message 'No chart configured for the moment ... [Create a new chart configuration](#)'. Below this, there are sections for 'Chart Width * Height', 'Chart time scale', 'Chart time length', and 'Filters'. The 'Filters' section contains several filtering options: Job filtering (radio buttons for 'ALL Jobs' and 'Job name regex'), Node filtering (radio buttons for 'ALL Nodes' and 'Master only'), Launcher filtering (radio buttons for 'ALL Users' and 'System only'), and Statuses taken into account (checkboxes for Success, Failures, Unstables, Aborted, and Not Build). There are also sections for 'Elements displayed on chart' with checkboxes for 'Build statuses with Y Axis type' (Count), 'Total build time', and 'Average build time'.

At the bottom, there are buttons for 'Overview', 'Create new chart', and 'Cancel'. The footer includes links for 'Help us localize this page', 'Page generated: Oct 24, 2015 4:03:17 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

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



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



17. 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.

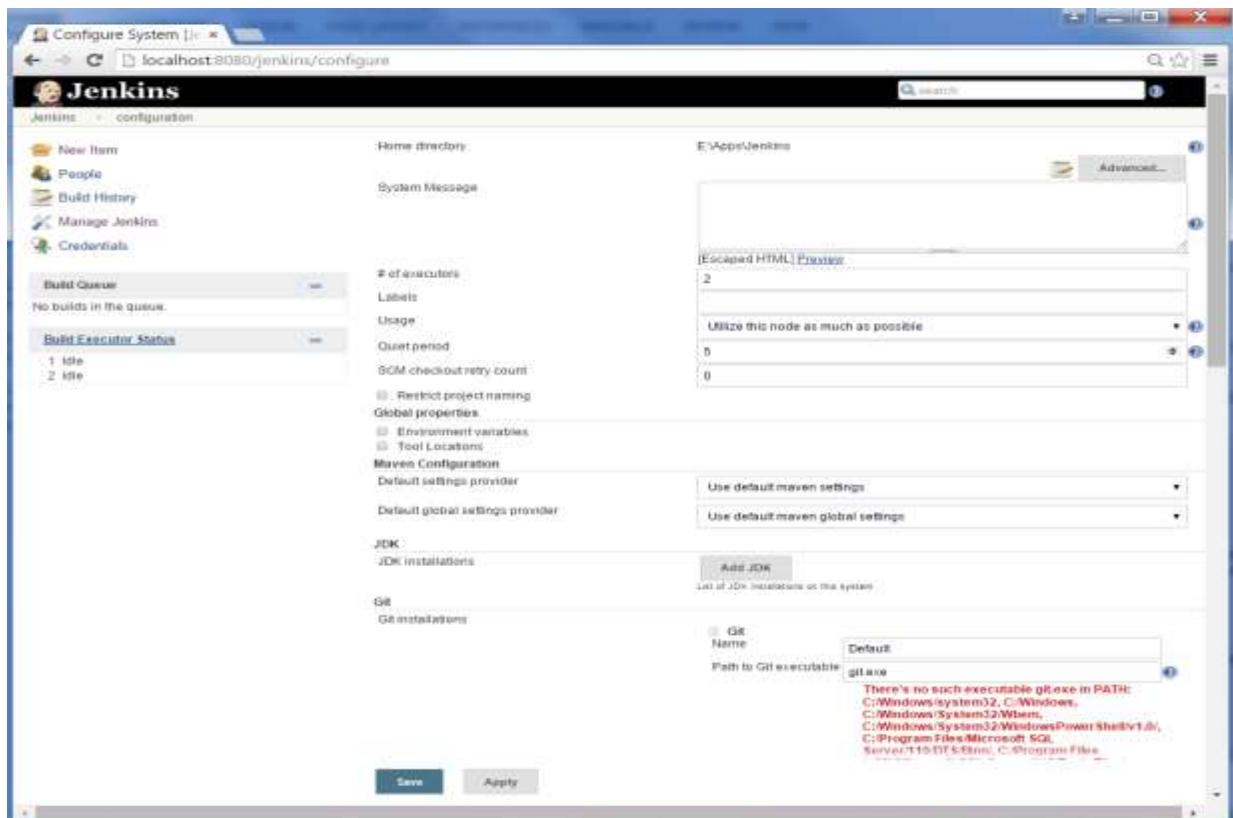
<http://localhost:8080/jenkins/exit> - shutdown jenkins

<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.

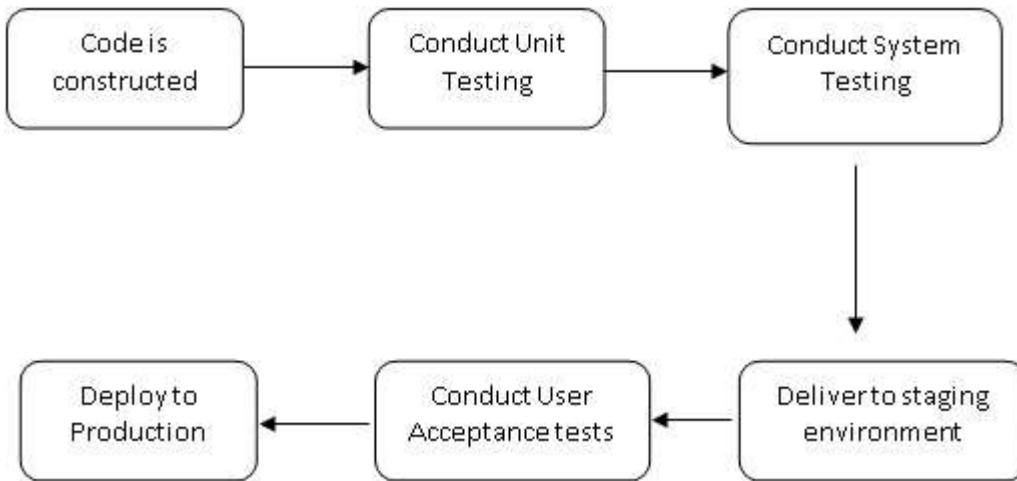


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

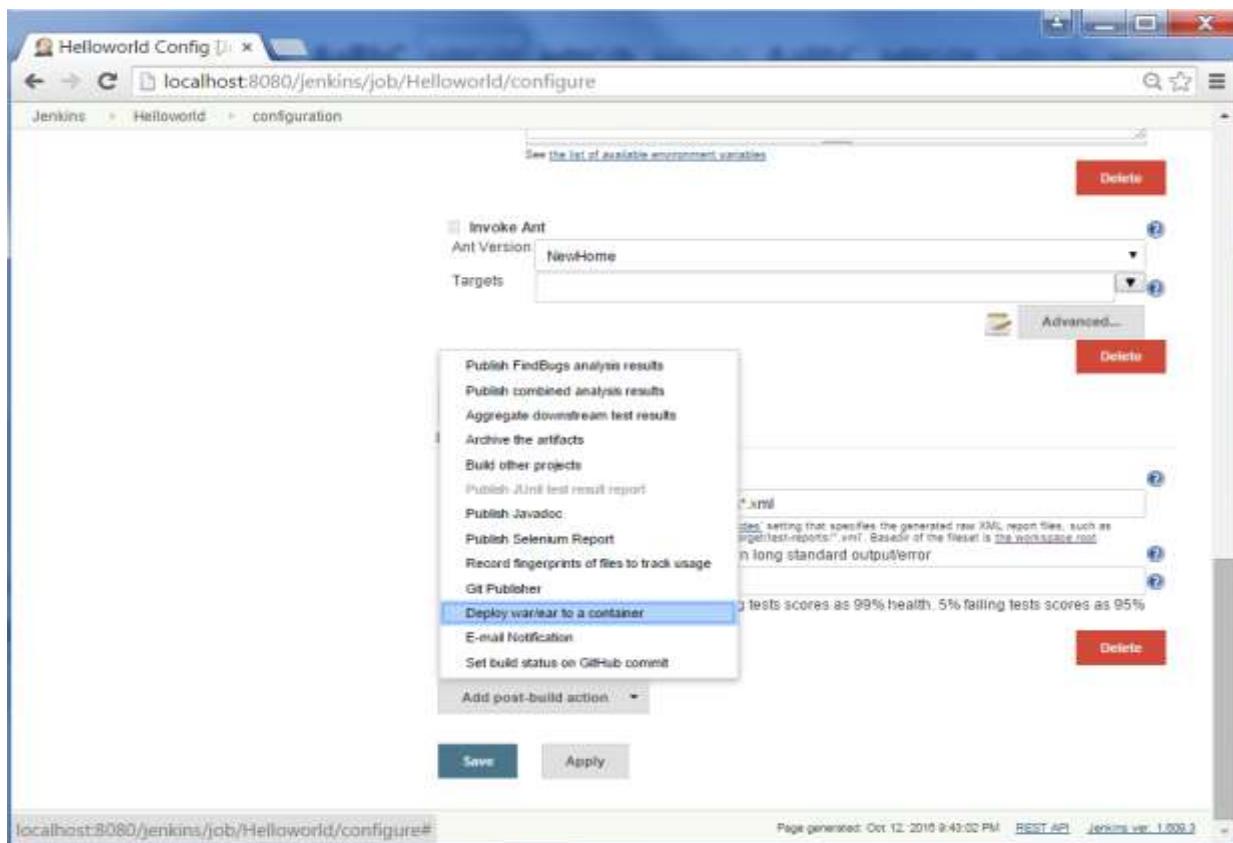
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

18. 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.



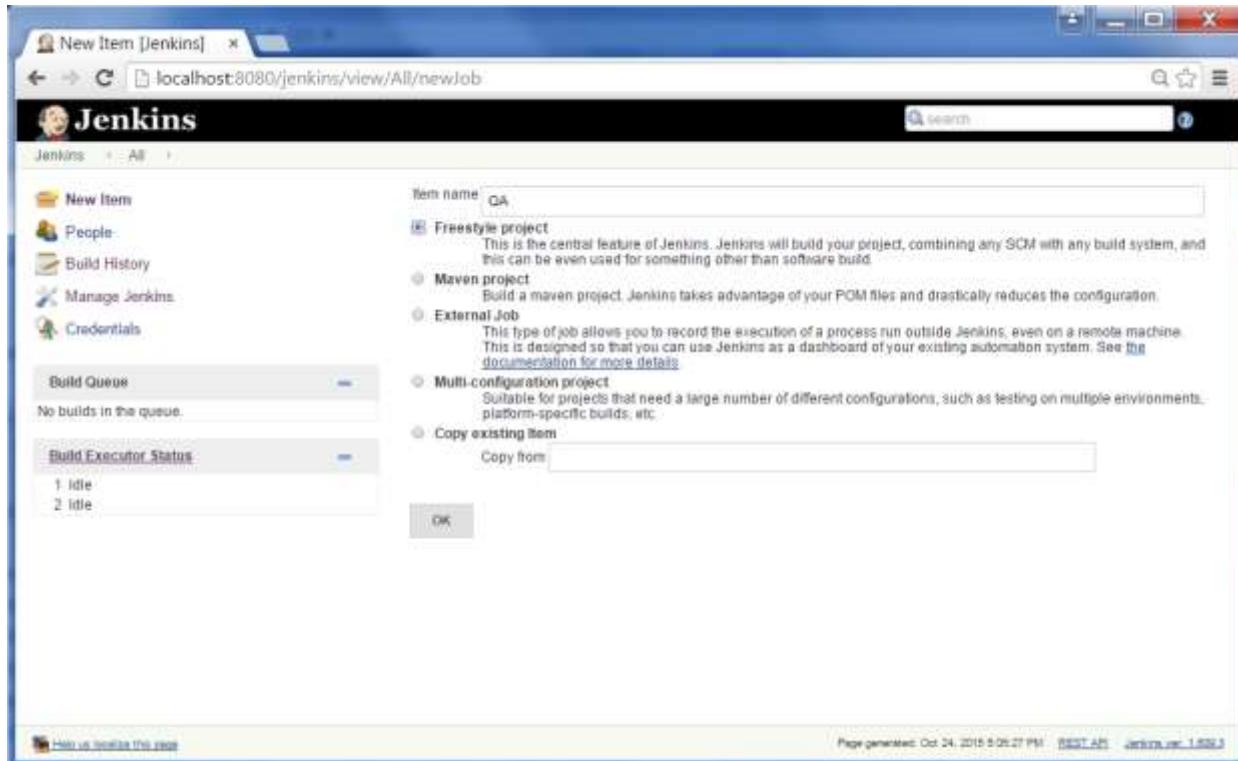
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



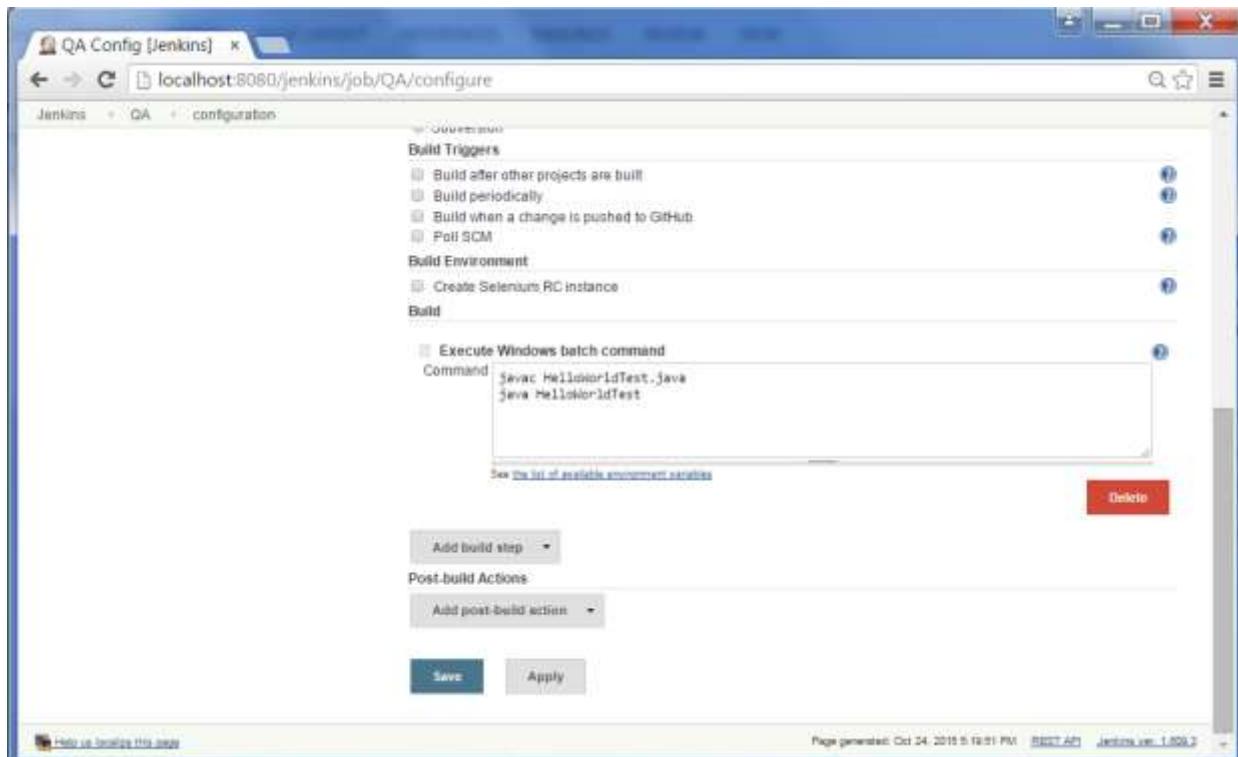
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

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.

	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 with the 'Helloworld' project listed. A tooltip for the 'Configure' button is displayed, indicating it will lead to the project configuration page.

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

Configure

Step 4 : In the project configuration, choose the 'Add post-build action' and choose 'Build other projects'

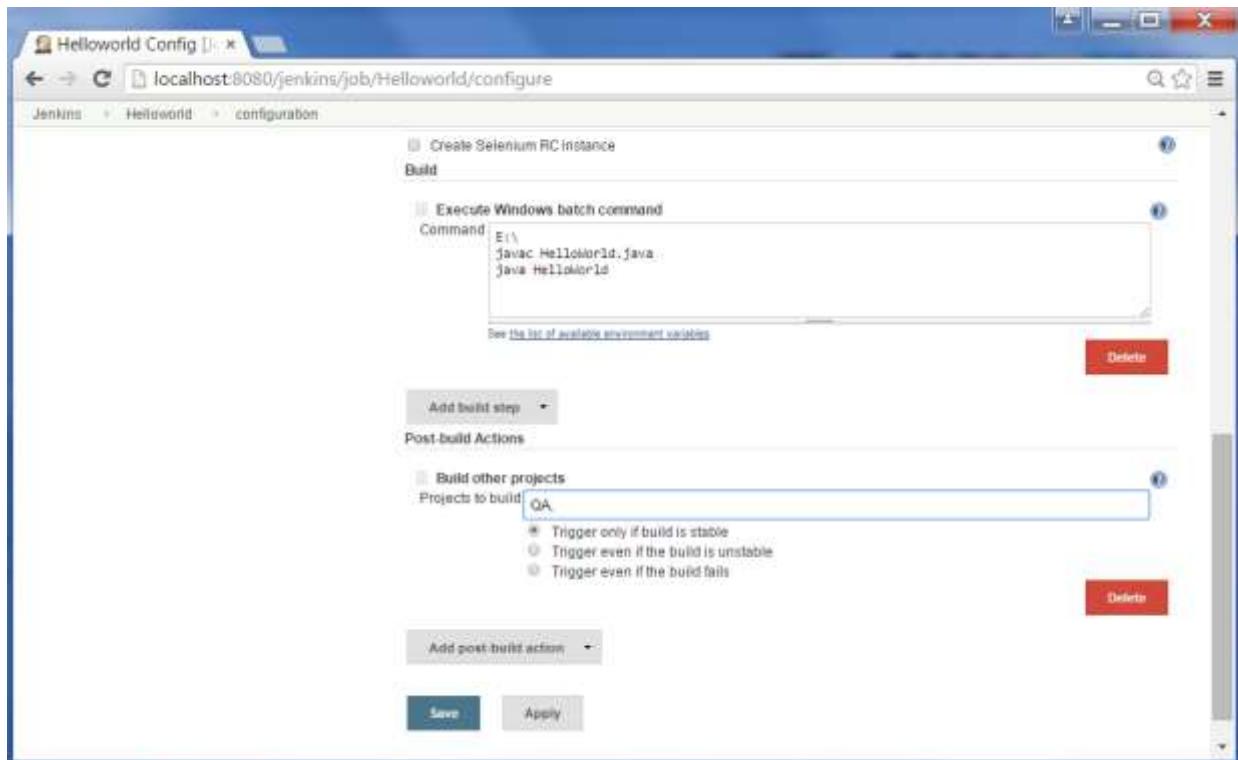
The screenshot shows the 'Helloworld' configuration page. A dropdown menu under 'Add post-build action' has 'Build other projects' selected. A tooltip for this action is shown, explaining it builds downstream projects based on workspace root.

Add post-build action

- 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

Save **Apply**

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



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



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

Install	Name	Version
ontrack Jenkins plugin	This plug-in allows to connect to an Ontrack server in order to enable traceability and monitoring of your continuous delivery pipeline(s).	2.15.0
Fail The Build Plugin	Set or change the build result to test job configurations - catchers, publishers, promotions, build pipelines, etc.	1.0
Runscope plugin	This plugin allows you to add a Runscope API test in a build step into your Jenkins build pipeline. The plugin will trigger a specific API test (via Trigger URL) and wait for the test to complete. If the test passes, the build steps will continue. However, if it fails, the build will be marked as failed.	1.44
Build Graph View Plugin	Shows a graph of builds that relates together (aka "built pipeline").	1.1.1
Deployment Schema	Jenkins plugin to have a bird's eye view of your continuous deployment pipeline.	0.1.105
CloudBees Docker Hub Notification	This plugin provides integration between Jenkins and Docker Hub, utilizing a Docker Hub hook to trigger one (or more) Jenkins job(s). This allows you to implement continuous delivery pipelines based on Docker in Jenkins.	1.0.2
Seed Jenkins plugin	The Seed project aims to help automating the generation and management of pipelines for branches of a project in Jenkins.	0.17.0
Delivery Pipeline Plugin	Visualization of Delivery/Build Pipelines. renders pipelines based on upstream/downstream jobs. Full screen view for information radiators.	0.9.7

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

Update information obtained: 1 hr 36 min ago

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

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

Icon:

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

Build Queue Status: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle

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



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

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

The screenshot shows the Jenkins 'Edit View [Delivery Pipeline]' configuration page. The left sidebar lists various Jenkins management options like 'New Item', 'People', 'Build History', etc. 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:** None
- Update interval:** 2
- Build Queue:** No builds in the queue.
- Build Executor Status:** 1 Idle, 2 Idle
- Pipelines:** Components
 - Name: Firstjob (highlighted with a red 'Delete' button)
 - Initial Job: HelloWorld
 - Final Job (optional):
- Regular Expression:** (empty field)

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'. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area is titled 'Firstjob' and shows three stages:

- #14 triggered by user anonymous started 29 minutes ago**: Total build time: 2 sec. Shows a 'Helloworld' stage (green bar) and a 'QA' stage (green bar). A green arrow points from Helloworld to QA.
- #13 triggered by user anonymous started an hour ago**: Total build time: 3 sec. Shows a 'Helloworld' stage (green bar) and a 'QA' stage (green bar). A green arrow points from Helloworld to QA.
- #12 triggered by user anonymous started 2 hours ago**: Total build time: 1 sec. Shows a 'Helloworld' stage (red bar) and a 'QA' stage (green bar). A red arrow points from Helloworld to QA.

At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 5:49:35 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

Another famous plugin is the **build pipeline plugin**. Let's take a look at this.

Step 1 : Go to Manage Jenkins->Manage Plugin's. In the available tab, search for 'Build Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main content area has a "Available" tab selected. A search bar at the top right contains the text "Build pipeline". Below it is a table with columns: "Install ↓", "Name", and "Version". The table lists five plugins:

Install ↓	Name	Version
<input checked="" type="checkbox"/>	Build Pipeline Plugin This plugin provides a _Build Pipeline View_ of upstream and downstream connected jobs that typically form a build pipeline.	1.4.8
<input type="checkbox"/>	Fail The Build Plugin Set or change the build result to test job configurations – notifiers, publishers, promotions, build pipelines, etc.	1.0
<input type="checkbox"/>	Runscope plugin This plugin allows you to add a Runscope API test as a build step into your Jenkins build pipeline. The plugin will trigger a specific API test (via Trigger URL) and wait for the test to complete. If the test passes, the build steps will continue. However, if it fails, the build will be marked as failed.	1.44
<input type="checkbox"/>	Build Graph View Plugin Shows a graph of builds that relates together (aka "build pipeline").	1.1.1
<input type="checkbox"/>	Delivery Pipeline Plugin Visualisation of Delivery/Build Pipelines, renders pipelines based on upstream/downstream jobs. Full screen view for information radiators.	0.9.7

At the bottom are two buttons: "Install without restart" and "Download now and install after restart". There is also a "Update info" link.

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

The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [Jenkins]". The address bar shows "localhost:8080/jenkins/". The main content area has a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". Below the sidebar are two expandable sections: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). The main area has a table titled "All" with columns: S, W, Name, Last Success, Last Failure, and Last Duration. It lists two jobs:

S	W	Name	Last Success	Last Failure	Last Duration
		Helgeundf	25 min - #14	1 hr 49 min - #12	1.4 sec
		GA	25 min - #5	28 min - #2	1.4 sec

At the bottom are three RSS feed links: "RSS for all", "RSS for failures", and "RSS for just latest builds".

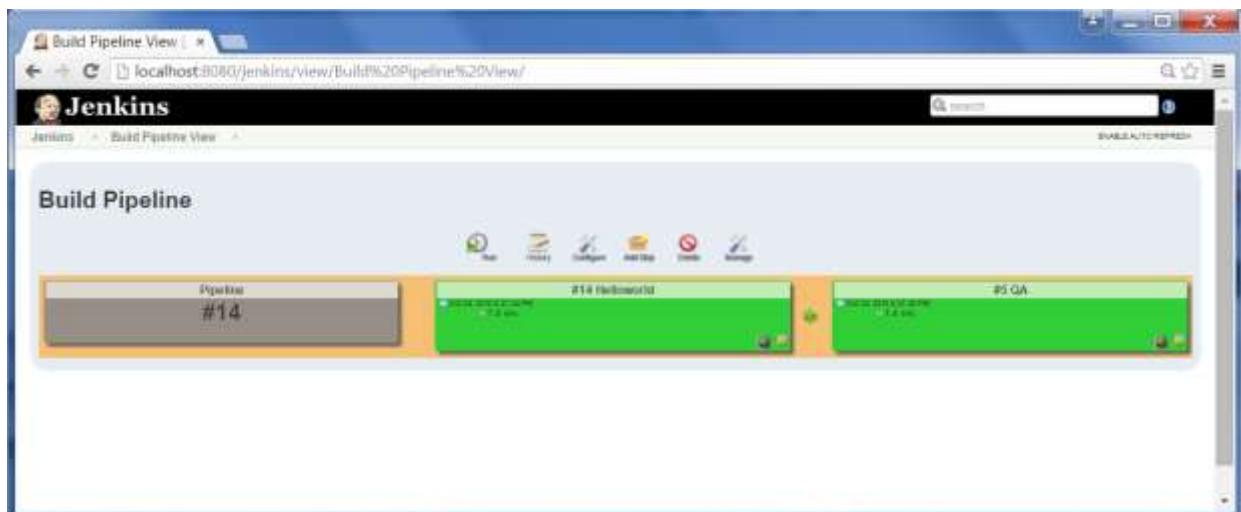
Step 3 : Enter any name for the View name and choose the option 'Build Pipeline View'.

The screenshot shows the Jenkins 'New View' configuration dialog. The title bar says 'New View [Jenkins]'. The URL in the address bar is 'localhost:8080/jenkins/newView'. The main content area has a 'View name' field containing 'Build Pipeline View'. There are three radio button options: 'Build Pipeline View' (selected), 'Delivery Pipeline View', and 'List View'. The 'Build Pipeline View' option is described as showing jobs in a build pipeline view, displaying the complete pipeline of jobs that a version propagates through. The 'Delivery Pipeline View' option is described as showing one or more delivery pipeline instances. The 'List View' option is described as showing items in a simple list format where jobs can be chosen. On the left, there's a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below the sidebar are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom right is an 'OK' button. The footer includes links for 'Help us localize this page', 'Page generated: Oct 24, 2015 5:50:44 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

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 configuration interface for a 'Build Pipeline View'. The left sidebar contains links for New Item, People, Build History, Edit View, Delete View, Manage Jenkins, and Credentials. The main configuration area includes fields for Name (Build Pipeline View), Description, Filter build queue, Filter build executors, and Build Pipeline View Title. Under Layout, it says 'Based on upstream/downstream relations' and provides options for selecting initial jobs, restrict triggers, and show pipeline headers. The 'Select Initial Job' dropdown is set to 'Helloworld'. 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.



19. Jenkins – Managing Plugins

To get the list of all plugins available within Jenkins, one can visit the link -

<https://wiki.jenkins-ci.org/display/JENKINS/Plugins>

The screenshot shows a web browser window with the address bar containing <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The page title is "Plugins". The left sidebar has sections for "Jenkins" (Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, Wiki Site Map) and "Documents" (Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes). The main content area is titled "How to Install plugins" and includes sub-sections like "Using the interface", "Installing the newest version", "Installing a specific version", "By hand", "Getting notified of plugin releases", "Developers", and "Plugins by topic". Under "Plugins by topic", there are 22 numbered items from 1 to 22, each with a link to a sub-page.

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

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

Uninstalling Plugins

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

Enabled	Name	Version	Previously installed version	Pinned	Uninstall
✓	AnsiColor	1.2			Uninstall
✓	Build-Header-MavenPlugin	1.2			Uninstall
✓	Build-Step-Email-Plugin	2.4.8			Uninstall
✓	Credentials Plugin	1.23	Downgrade to 1.18	Unpin	Uninstall
✓	CVS Plugin	2.11			Uninstall
✓	Delivery Pipeline	0.0.7			Uninstall
✓	Deploy-to-container-Plugin	1.10			Uninstall
✓	External-Monitor-Job-Type-Plugin	1.0			Uninstall
✓	Extreme-Notification-Plugin	1.1			Uninstall
✓	FindBugs-Plugin	4.0.2			Uninstall

Installing another Version of a Plugin

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

HTTP Proxy Configuration

Server:

Port:

User name:

Password:

No Proxy Host:

[Advanced...](#)

Upload Plugin

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

File: [Choose File](#)

Upload

Update Site

URL:

Submit

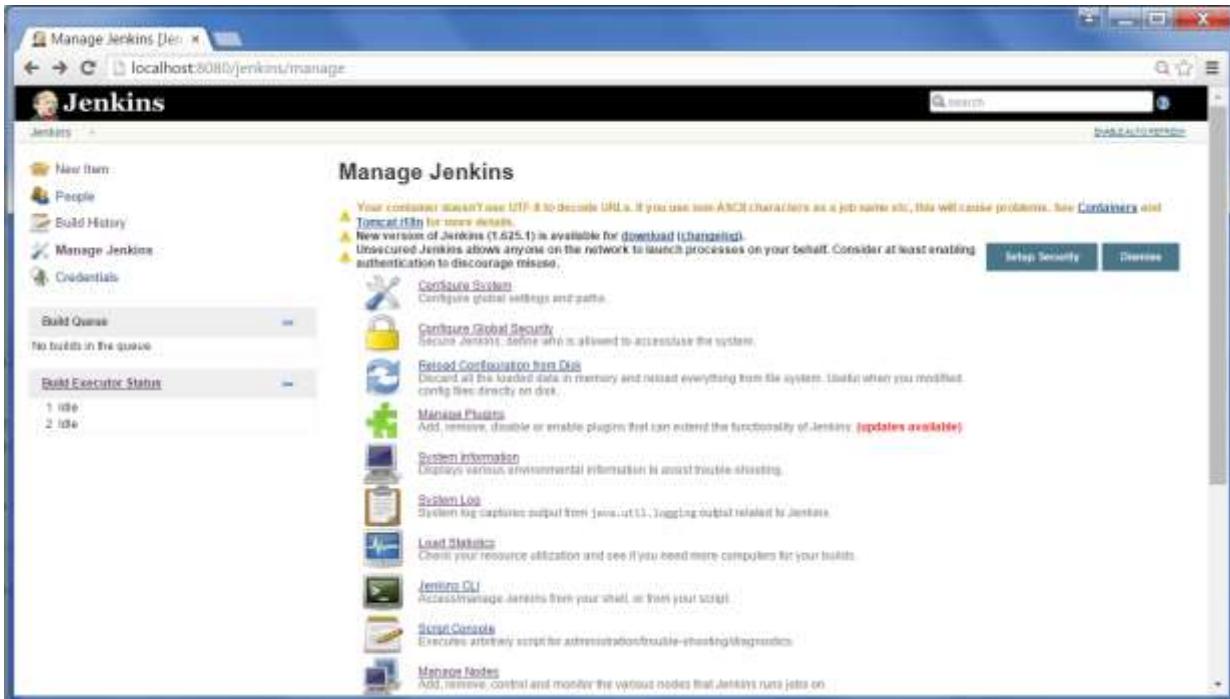
Update information obtained: 2 hr 5 min ago [Check now](#)

20. 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.



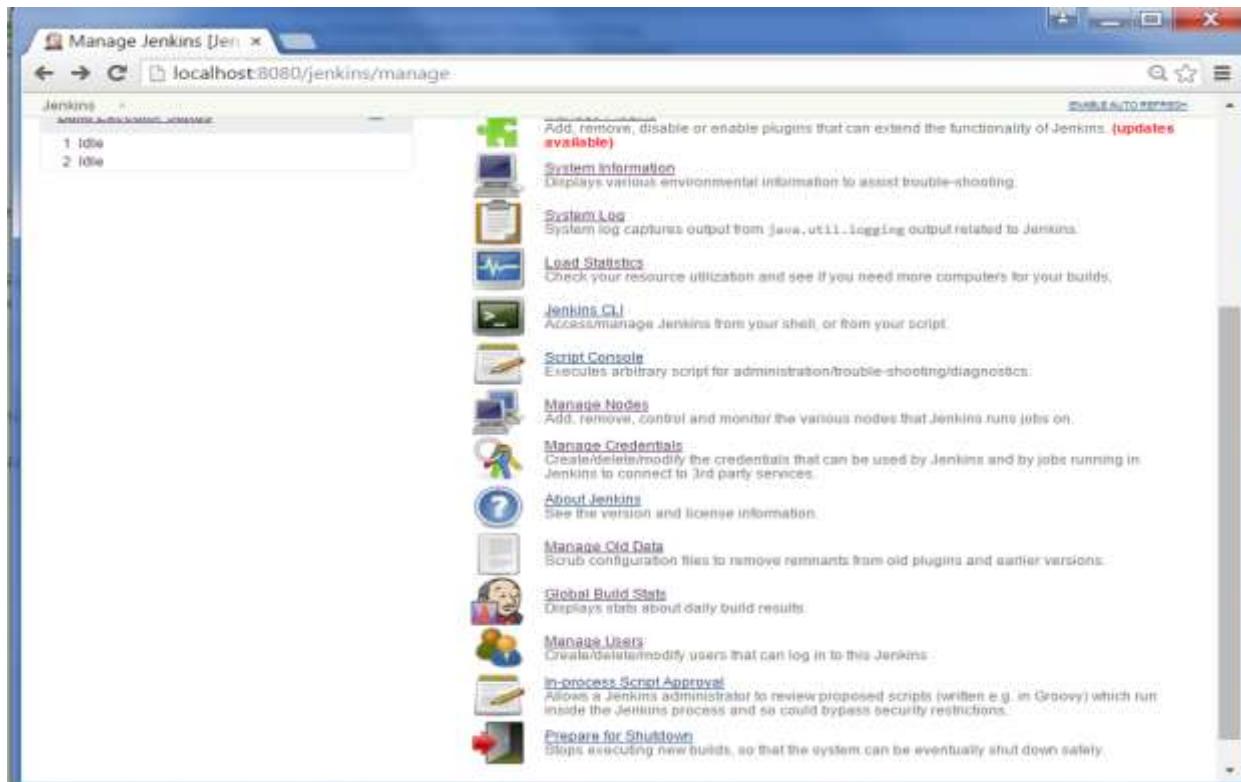
Step 3 : You will be prompted to add your first user. As an example, we are setting up an admin users for the system.

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

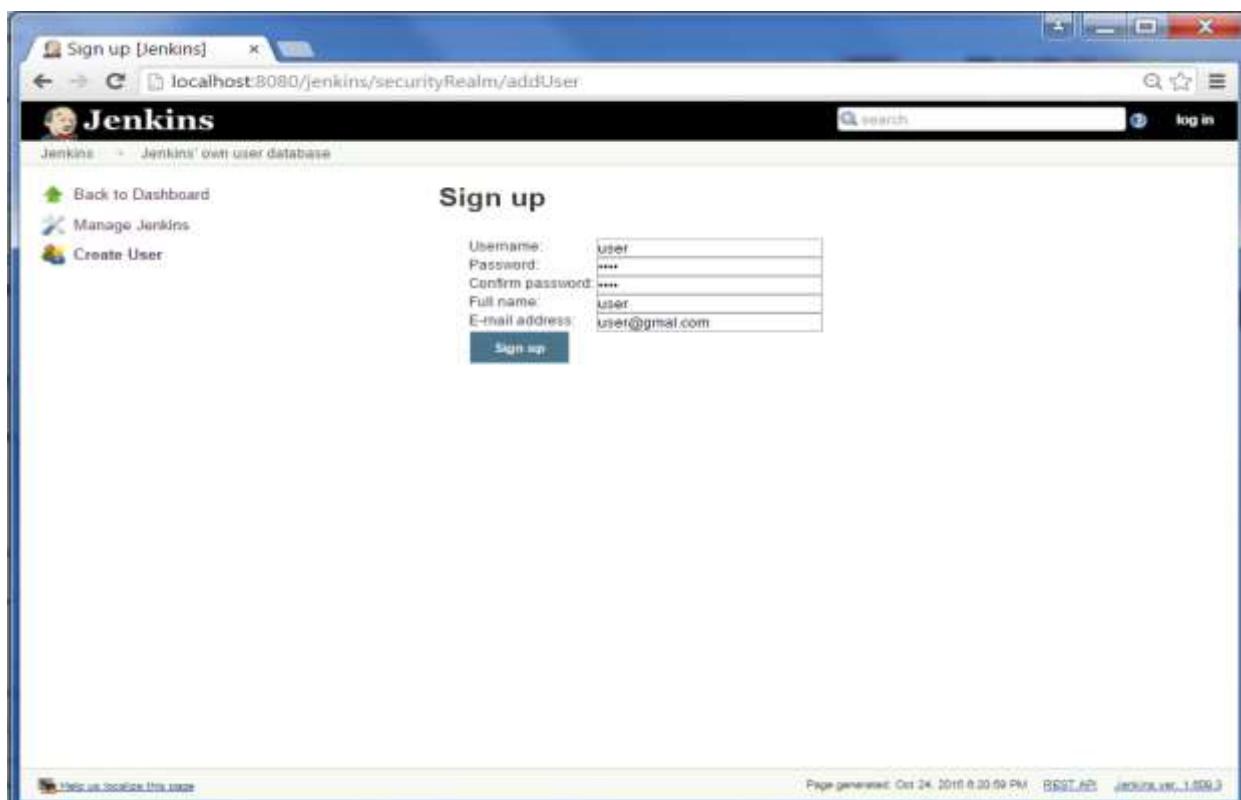
Username:	admin
Password:
Confirm password:
Full name:	Administrator
E-mail address:	al@gmail.com

Below the form is a blue 'Sign up' button. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 6:18:01 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

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.

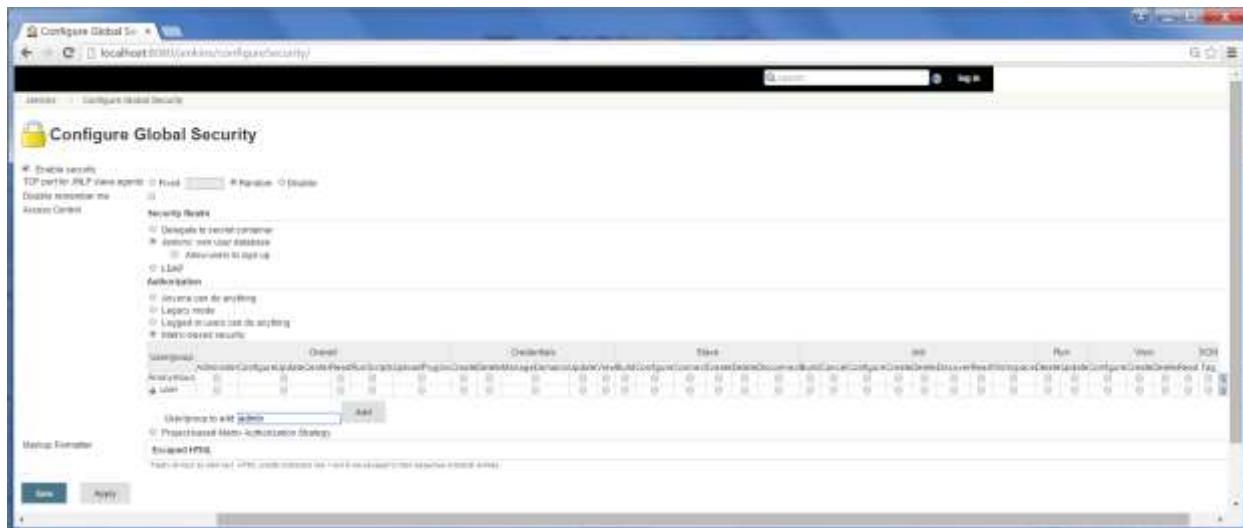


Step 5 : Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called 'user'.



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.

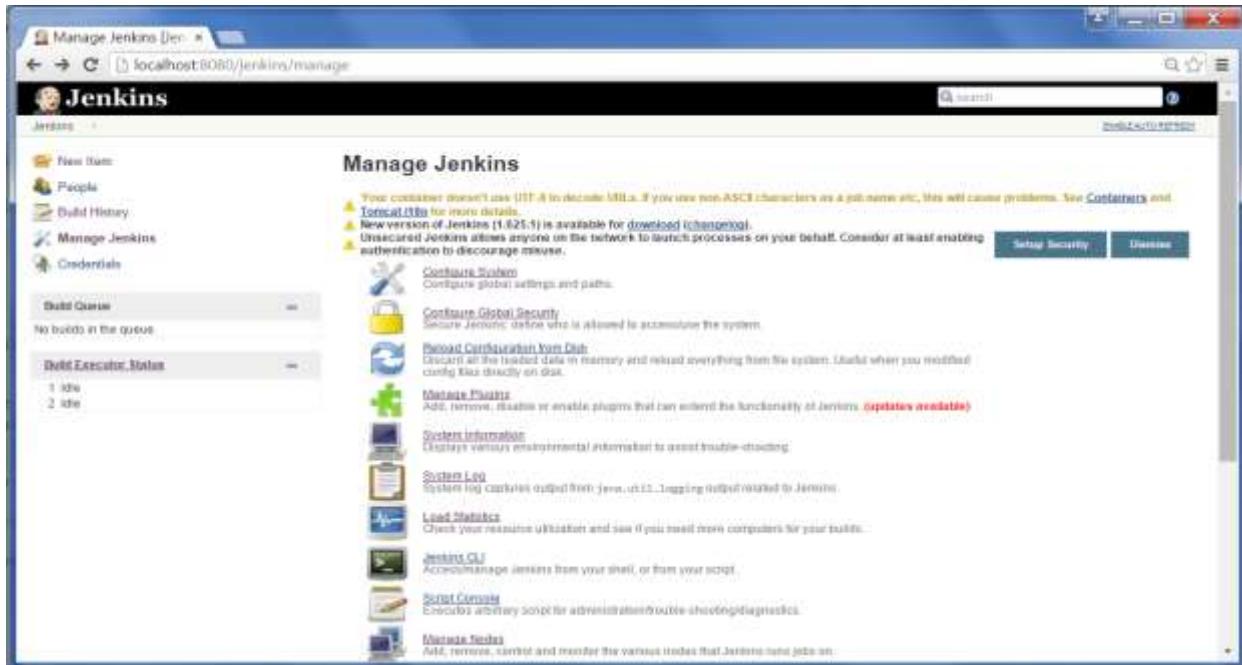
Your Jenkins security is now setup.

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

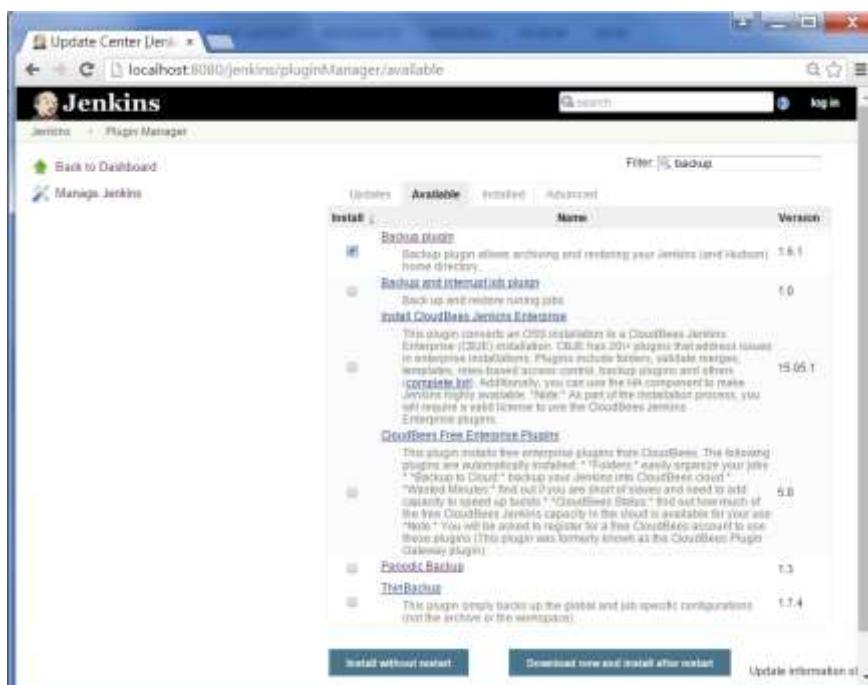
21. Jenkins – Backup Plugin

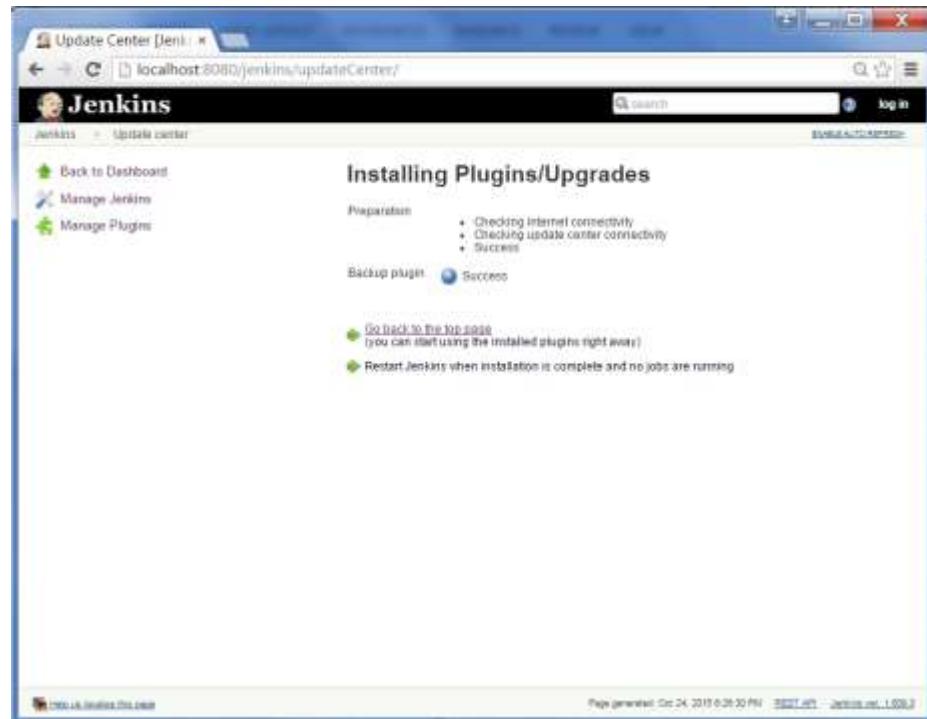
Jenkins has a backup plugin which can be used to backup critical configuration settings related to Jenkins. Follow the steps given below to have a backup in place.

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

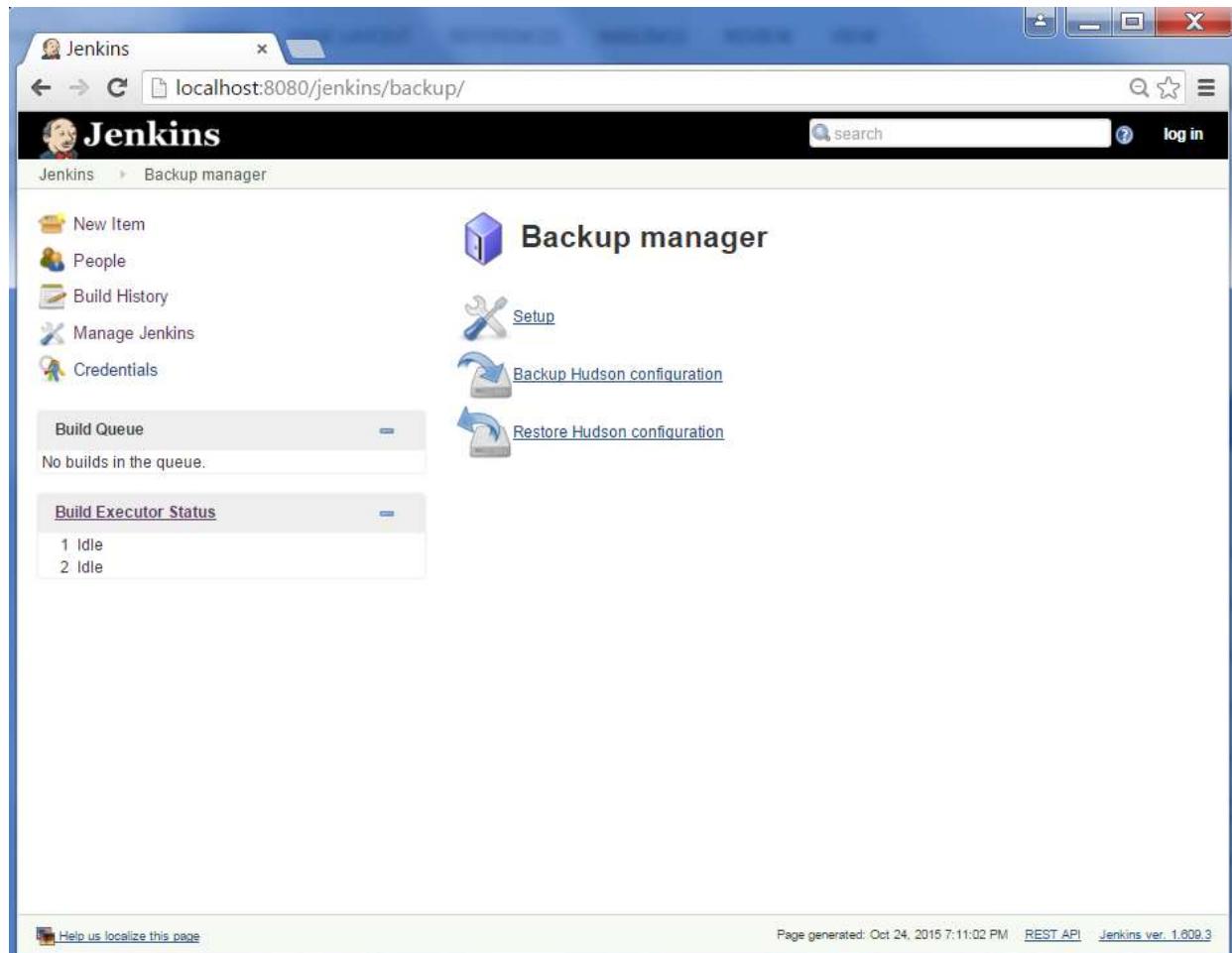




Step 3 : Now when you go to Manage Jenkins, and scroll down you will see 'Backup Manager' as an option. Click on this option.

The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with links like System Log, Load Statistics, Jenkins CLI, Script Console, Manage Nodes, Manage Credentials, About Jenkins, Manage Old Data, Global Build Stats, Manage Users, In-process Script Approval, Backup manager, and Prepare for Shutdown. The "Backup manager" link is highlighted with a red arrow pointing to it. The footer includes a "Help us localize this page" link and a "Page generated: Oct 24, 2015 7:10:31 PM REST API Jenkins ver. 1.609.3" message.

Step 4 : Click on Setup.



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

The screenshot shows the Jenkins interface for managing backup configurations. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins, and Credentials. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (2 Idle). The main content area is titled "Backup config files". It contains two main sections: "Backup configuration" and "Backup content". In "Backup configuration", the Hudson root directory is set to E:\Jenkins and the backup directory is set to D:\Backup, with the format set to zip and the file name template to backup_@date@.extension@. There are also options for Custom exclusions, Verbose mode, Configuration files (.xml) only, and No shutdown. In "Backup content", there are checkboxes for Backup job workspace, Backup builds history, Backup maven artifacts archives, and Backup fingerprints. A "Save" button is located at the bottom of the configuration section. At the very bottom of the page, there are links for Help us localize this page, Page generated: Oct 24, 2015 7:17:46 PM, REST API, and Jenkins ver. 1.609.3.

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

The screenshot shows the Jenkins interface with the title bar "Jenkins" and the URL "localhost:8080/jenkins/backup/". The main content area is titled "Backup manager" and contains the following elements:

- A sidebar on the left with icons for "New Item", "People", "Build History", "Manage Jenkins", and "Credentials".
- A central section titled "Backup manager" with three options:
 - "Setup" (with a wrench icon)
 - "Backup Hudson configuration" (with a blue arrow icon)
 - "Restore Hudson configuration" (with a grey arrow icon)
- Two expandable sections:
 - "Build Queue" which is collapsed and displays "No builds in the queue."
 - "Build Executor Status" which is collapsed and displays "1 Idle" and "2 Idle".
- A footer bar at the bottom with links for "Help us localize this page", "Page generated: Oct 24, 2015 7:11:02 PM", "REST API", and "Jenkins ver. 1.609.3".

The next screen will show the status of the backup.

The screenshot shows the Jenkins interface for a 'Backup manager log' page. The URL in the browser is `localhost:8080/jenkins/backup/backup`. The main content area displays a log message: 'Jenkins is going to shut down' followed by a series of INFO-level log entries detailing the backup process. The log entries include:

```
[ INFO] Backup started at [10/24/15 19:19:31]
[ INFO] Setting hudson in shutdown mode to avoid files corruptions.
[ INFO] Waiting all jobs end...
[ INFO] Number of running jobs detected : 0
[ INFO] All jobs finished.
[ INFO] Full backup file name : D:\Backup\backup_20151024_1919.zip
[ INFO] Saved files : 911
[ INFO] Number of errors : 0
[ INFO] Cancel hudson shutdown mode
[ INFO] Backup end at [10/24/15 19:19:50]
[ INFO] [19.524s]
```

The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 7:19:31 PM REST API Jenkins ver. 1.600.3', and a 'log in' link.

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

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 and Build Executor Status, both indicating 'No builds in the queue.' and '1 Idle' respectively. The main content area is titled 'Backup manager' and contains three items: 'Setup' (with a wrench icon), 'Backup Hudson configuration' (with a blue arrow icon), and 'Restore Hudson configuration' (with a grey arrow icon). At the bottom of the page, there is a link 'Help us localize this page' and footer text 'Page generated: Oct 24, 2015 7:11:02 PM REST API Jenkins ver. 1.609.3'.

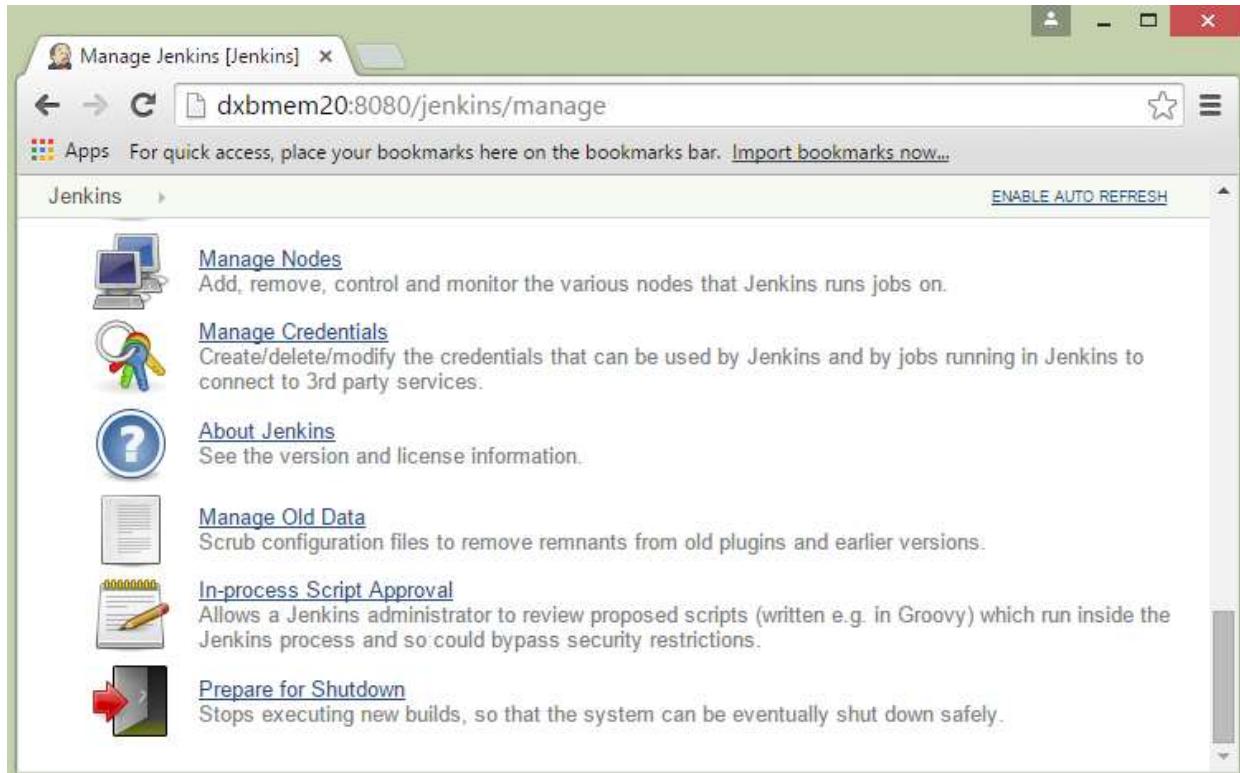
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.

The screenshot shows the Jenkins Backup manager interface. At the top, there is a navigation bar with links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below this, the main content area is titled 'Backup manager' and displays the message 'Available backup in D:\Backup :'. A single backup entry, 'backup_20151024_1919.zip', is listed with a 'Launch restore' button next to it. On the left side, there are two collapsed sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which shows '1 Idle' and '2 Idle'). At the bottom of the page, there is a footer with links for 'Help us localize this page', 'Page generated: Oct 24, 2015 7:20:45 PM', 'REST API', and 'Jenkins ver. 1.609.3'.

22. Jenkins – Remote Testing

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

Step 1 : Ensuring your master slave configuration is in place. Got to your master Jenkins server. Go to Manage Jenkins->Manage Nodes.



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

The screenshot shows the Jenkins 'Nodes' page. At the top, there are links for 'Jenkins', 'nodes', and 'ENABLE AUTO REFRESH'. Below this, there are two sections: 'idle' (2 Idle) and 'DXBMEM30' (1 Idle). A table lists the details for each node:

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
	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 located at the bottom right of the table.

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

The screenshot shows the same Jenkins 'Nodes' page as before, but with a context menu open over the DXBMEM30 node. The menu options are: 'Delete Slave', 'Configure' (which is highlighted with a blue selection box), and 'Build History'. The 'Configure' option is intended to be clicked to proceed with the configuration of the slave machine.

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

The screenshot shows the 'DXBMEM30 Configuration' page in a browser. The URL is `dxbmemp30:8080/jenkins/computer/DXBMEM30/configure`. The page displays the following configuration details for the node 'DXBMEM30':

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

A 'Save' button is visible at the bottom left, and an 'Advanced...' button is at the bottom right.

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

The screenshot shows the Jenkins Dashboard under the 'Manage Jenkins' section. The URL is `dxbmemp30:8080/jenkins/`. The dashboard includes:

- A main menu with links: New Item, People, Build History, Manage Jenkins, and Credentials.
- A 'Build Queue' section stating 'No builds in the queue.'
- A 'Build Executor Status' section showing '1 Idle' and '2 Idle' executors.

Step 5 : Click on the DXBMEM30 instance.

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a header bar with the Jenkins logo and a user icon. Below it is a browser-style address bar with the URL 'dxbmemp20:8080/jenkins/computer/'. The main content area has a breadcrumb navigation path: 'Jenkins > nodes >'. To the right of the path is a link 'ENABLE AUTO REFRESH'. Below the path, a message says 'No builds in the queue.' Underneath, there's a section titled 'Build Executor Status' which lists two nodes: 'master' and 'DXBMEM30 (offline)'. A table follows, providing detailed information for each node:

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

A blue button labeled 'Refresh status' is located at the bottom right of the table area. The entire interface has a light blue header and a white body with some shadows.

Step 6 : Scroll down and you will see the Launch option which is the option to Start 'Java Web Start'

The screenshot shows the Jenkins web interface. At the top, there's a header bar with a user icon, the text 'DXBMEM30 [Jenkins]', and standard window controls (minimize, maximize, close). Below the header is a browser-style address bar with the URL 'dxbmeme20:8080/jenkins/computer/DXBMEM30/'. The main content area has a breadcrumb navigation path: 'Jenkins > nodes > DXBMEM30'. To the right of the path is a link 'ENABLE AUTO REFRESH'. The page displays a stack trace starting with 'at org.jenkinsci.remoting.nio.NioChannelHub.run(NioChannelHub.java:561)' and '... 6 more'. Below the stack trace, a section titled 'Connect slave to Jenkins one of these ways:' lists three methods:

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

Below this, it says 'Created by anonymous user'.

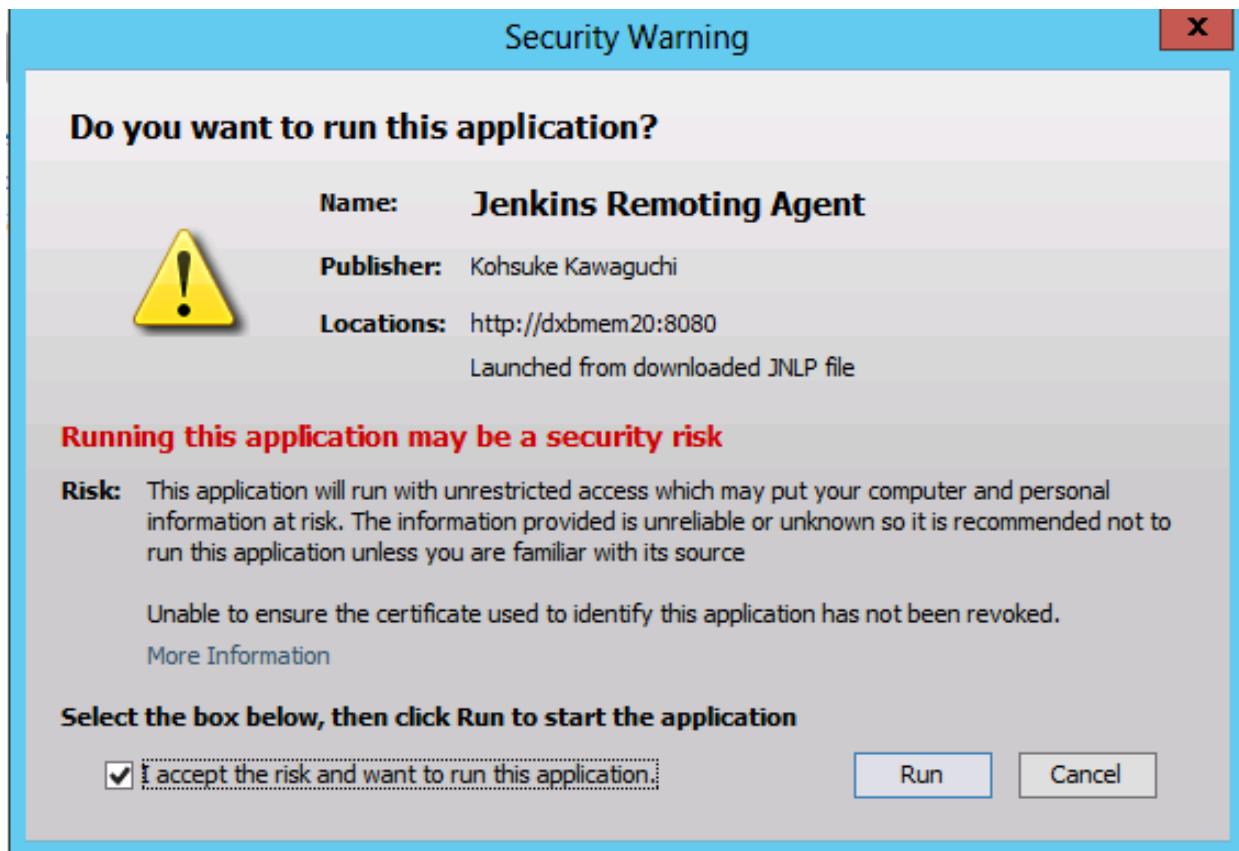
Projects tied to DXBMEM30

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

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

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

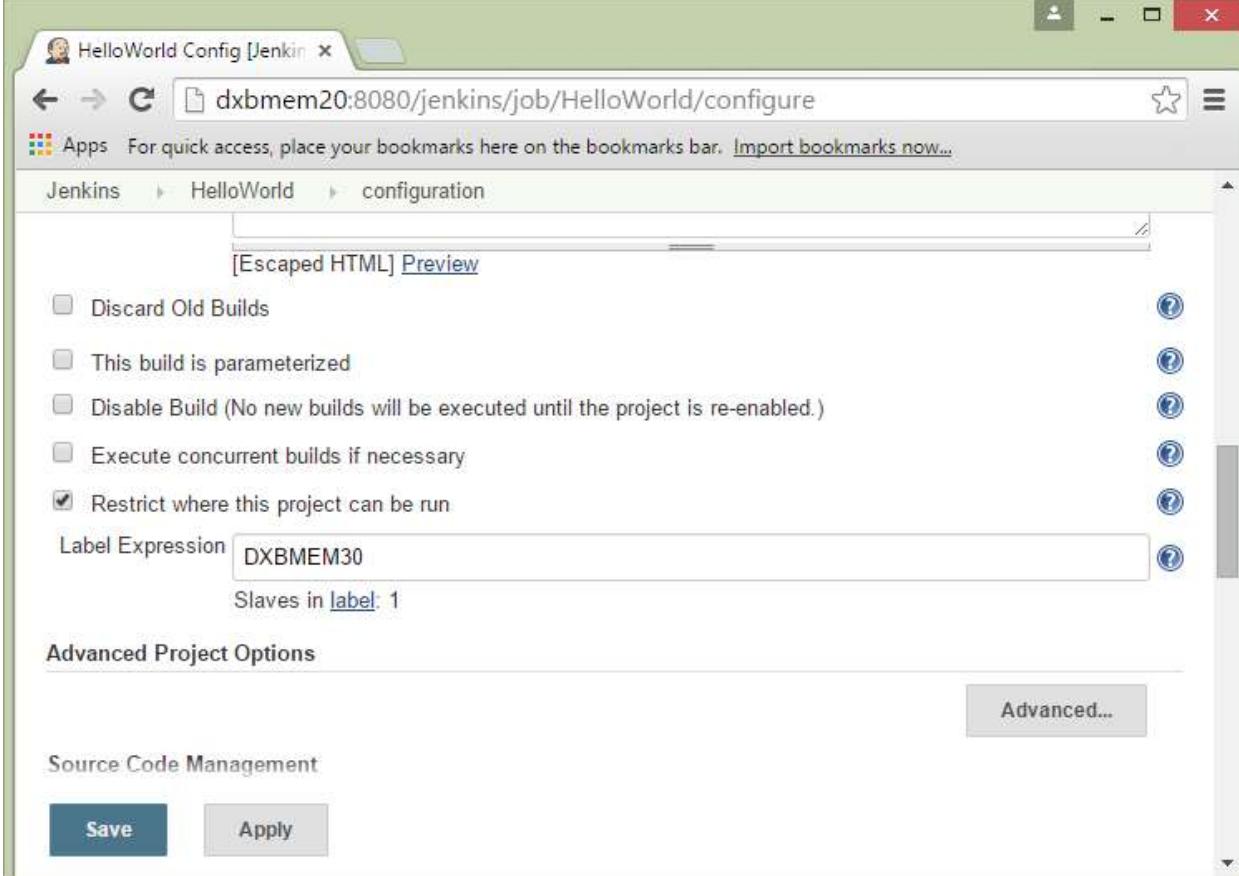


You will now see a Jenkins Slave window opened and now connected.



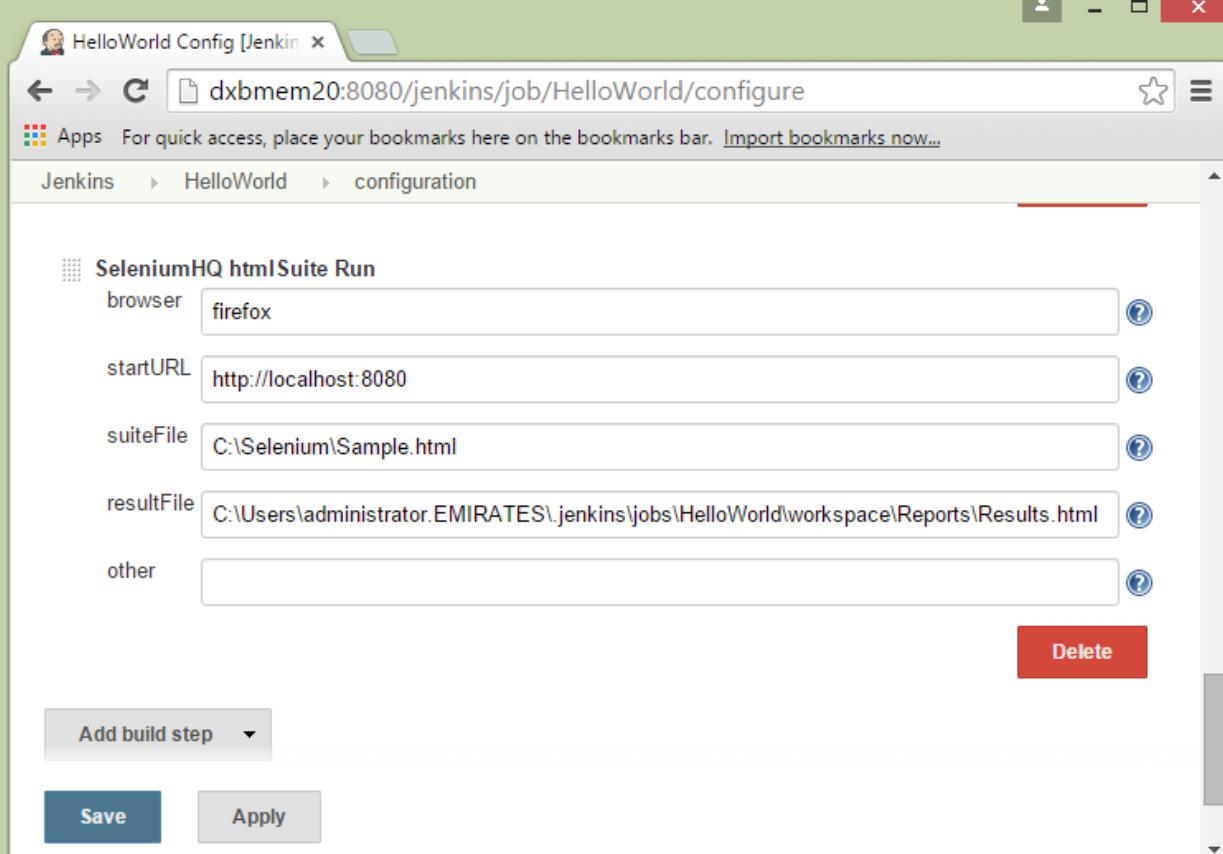
Step 8 : Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

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



The screenshot shows the Jenkins job configuration page for 'HelloWorld'. The URL in the browser is `dxbmcm20:8080/jenkins/job/HelloWorld/configure`. The configuration section is titled 'configuration'. Under 'Advanced Project Options', the 'Label Expression' field contains 'DXBMEM30'. The 'Restrict where this project can be run' checkbox is checked, and the 'Label Expression' field shows 'Slaves in label: 1'. At the bottom, there are 'Save' and 'Apply' buttons, and an 'Advanced...' button.

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 interface for the 'HelloWorld' job. The configuration step selected is 'SeleniumHQ htmlSuite Run'. The configuration parameters are as follows:

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

At the bottom of the configuration panel, there are 'Save' and 'Apply' buttons, and a 'Delete' button for the selected step.

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.