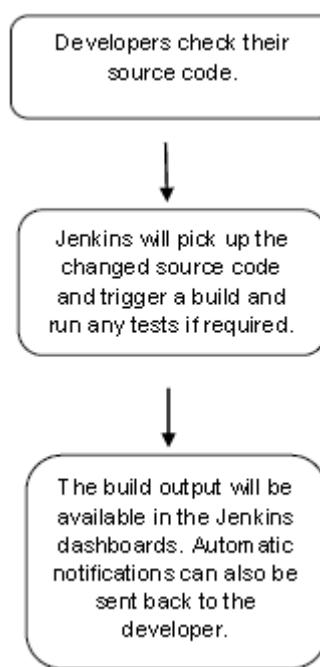


Jenkins - Quick Guide

Jenkins - Overview

Why Jenkins?

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



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

What is Continuous Integration?

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

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

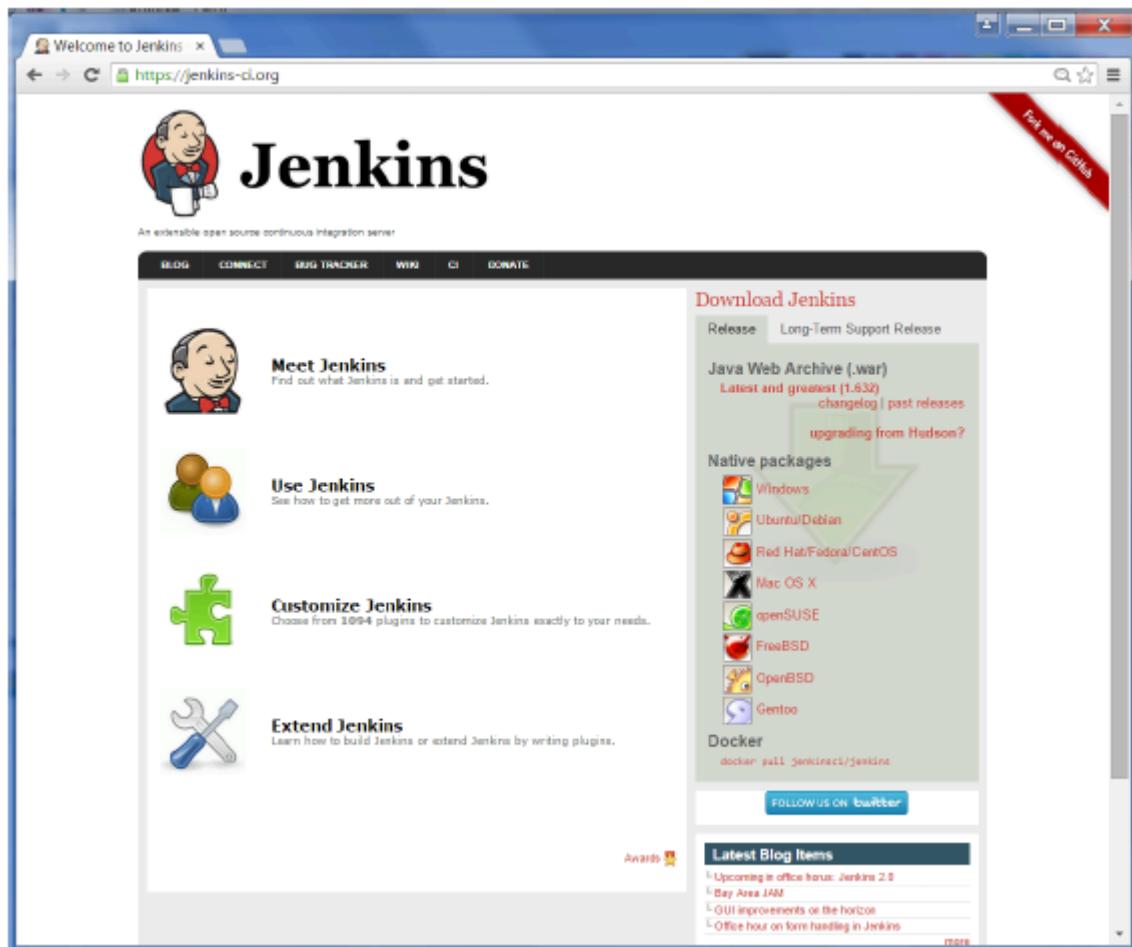
System Requirements

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

Jenkins - Installation

Download Jenkins

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



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



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

Starting Jenkins

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

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

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

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

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

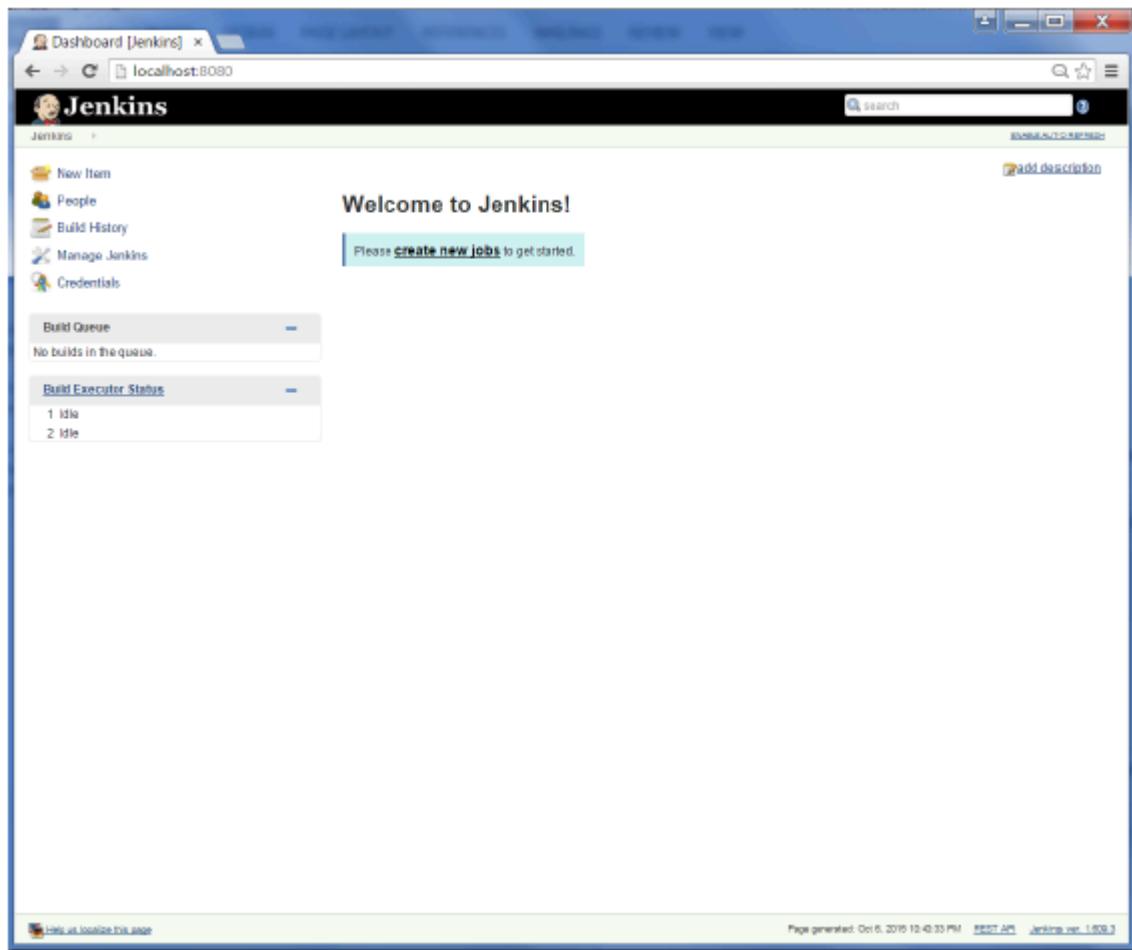
```
INFO: Jenkins is fully up and running
```

Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the link –

http://localhost:8080

This link will bring up the Jenkins dashboard.



Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

Step 1: Verifying Java Installation

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

OS	Task	Command
----	------	---------

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

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

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

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

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

Step 2: Verifying Java Installation

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

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

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

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

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

Step 3: Download Tomcat

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

The screenshot shows the Apache Tomcat homepage. On the left, there's a sidebar with links for Home, Taglibs, Maven Plugin, Download (with sub-links for Which version?, Tomcat 8.0, Tomcat 7.0, Tomcat 6.0, Tomcat Connectors, Tomcat Native, Taglibs, Archives), Documentation (with sub-links for Tomcat 8.0, Tomcat 7.0, Tomcat 6.0, Tomcat Connectors, Tomcat Native, Wiki, Migration guide), Problems? (with sub-links for Security Reports, Find help, FAQ, Mailing Lists, Bug Database, IRC), and Get Involved (with sub-links for Overview, SVN Repositories, Buildbot, Reviewboard, Tools). The main content area features the Apache Tomcat logo and a search bar. It highlights the "Apache Tomcat" section, which describes the software as an open-source implementation of Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. It mentions the Apache License version 2. Below this, it details the "Tomcat 8.0.27 Released" on 2015-10-01, noting fixes for various issues like URL escaping and TLD parsing. It also lists the "Tomcat 7.0.64 Released" on 2015-08-25, mentioning improvements for POJO WebSocket endpoints and async timeouts.

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

The screenshot shows the Apache Tomcat Tomcat 7.0.64 download page. The sidebar is identical to the homepage. The main content area is titled "Tomcat 7 Downloads". It welcomes visitors to the page and provides download links for the latest version of Tomcat 7.0.x. It includes sections for "Quick Navigation" (with links to KEYS, Browse, and Archives), "Release Integrity" (warning about verifying file integrity using OpenPGP signatures), "Mirrors" (listing the current mirror as http://www.us.apache.org/dist/ and providing a dropdown for other mirrors), and "7.0.64" (describing the distribution and linking to the README file). The "Binary Distributions" section lists various file formats available for download, including zip, tar.gz, and Windows zip files for both 32-bit and 64-bit architectures, along with an IIS installer and a Windows Service Installer.

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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

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

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

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

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

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



Jenkins - Git Setup

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

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The main header says "Jenkins". On the left, there's a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". Below the sidebar are two summary boxes: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). The central area has a "Welcome to Jenkins!" message with a button to "create new jobs". A note at the bottom says "Please [create new jobs](#) to get started." At the bottom right, it shows "Page generated: Oct 6, 2015 10:33:24 PM" and "Jenkins ver. 1.606.3".

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

The screenshot shows the "Manage Jenkins" screen at localhost:8080/jenkins/manage. The main header says "Manage Jenkins". The sidebar is identical to the dashboard. The central area lists several Jenkins management options: "Configure System", "Configure Global Security", "Reload Configuration from Disk", "Manage Plugins" (which is highlighted in red), "System Information", "System Log", "Load Statistics", "Jenkins CLI", "Script Console", "Manage Nodes", "Manage Credentials", "About Jenkins", "Manage Old Data", "In-Progress Script Approval", and "Prepare for Shutdown". There are also "Setup Security" and "Dismiss" buttons at the top right of the central area.

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

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

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

The screenshot shows the Jenkins Update Center interface. The title bar reads "Update Center [jenkins]" and the address bar shows "localhost:8080/jenkins/updateCenter/". The main content area is titled "Installing Plugins/Upgrades". On the left, there's a sidebar with links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area has two sections: "Preparation" (with a note about checking internet connectivity) and a list of pending plugin installations. The pending plugins listed are: Credentials Plugin, SSH Credentials Plugin, GIT client plugin, SCM API Plugin, Mailer Plugin, and GIT plugin. Each plugin entry includes a "Pending" status indicator. At the bottom of the page, 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 a "Help us improve this page" link and a timestamp "Page generated: Oct 8, 2015 10:42:53 PM".

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.

The screenshot shows the Jenkins 'New Item' configuration page. The 'Item name' field is set to 'Demo'. Below it, there are four project type options: 'Freestyle project', 'Maven project', 'External Job', and 'Multi-configuration project'. The 'Freestyle project' option is selected. A detailed description of what a Freestyle project is follows. On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below the sidebar, a 'Build Queue' section shows 'No builds in the queue.' and a 'Build Executor Status' section showing '1 Idle' and '2 Idle'. At the bottom right, there is an '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 'Demo Configuration' page. The 'Project name' is set to 'Demo'. In the 'Source Code Management' section, 'Git' is selected as the version control system. Other options like 'None', 'CVS', 'CVS Projectset', and 'Subversion' are also listed. There are sections for 'Build Triggers' and 'Build' with buttons for 'Add build step' and 'Post-build Actions'. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

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

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

The screenshot shows the Apache Maven download page at <https://maven.apache.org/download.cgi>. The left sidebar contains links for Support and Training, Documentation, Maven Plugins, Index (category), Running Maven, User Centre, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources, Security, Community, Community Overview, How to Contribute, Maven Repository, Getting Help, Issue Tracking, Source Repository, The Maven Team, Project Documentation, Project Information, Maven Projects, Ant Tasks, Archetype, Doxia, and I18N.

The main content area has sections for Memory, Disk, and Operating System requirements. It then lists 'Files' available for download:

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

Below the files section is a bulleted list of links:

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

The 'Previous Releases' section states that it is recommended to use the latest release version of Apache Maven. It also mentions that legacy archives are available for versions 3.0.4+.

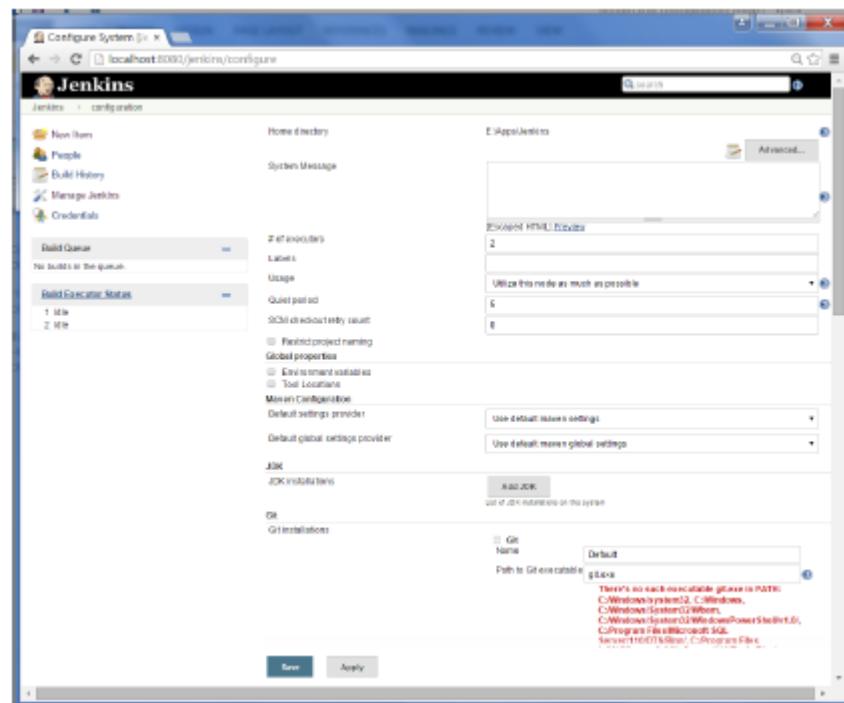
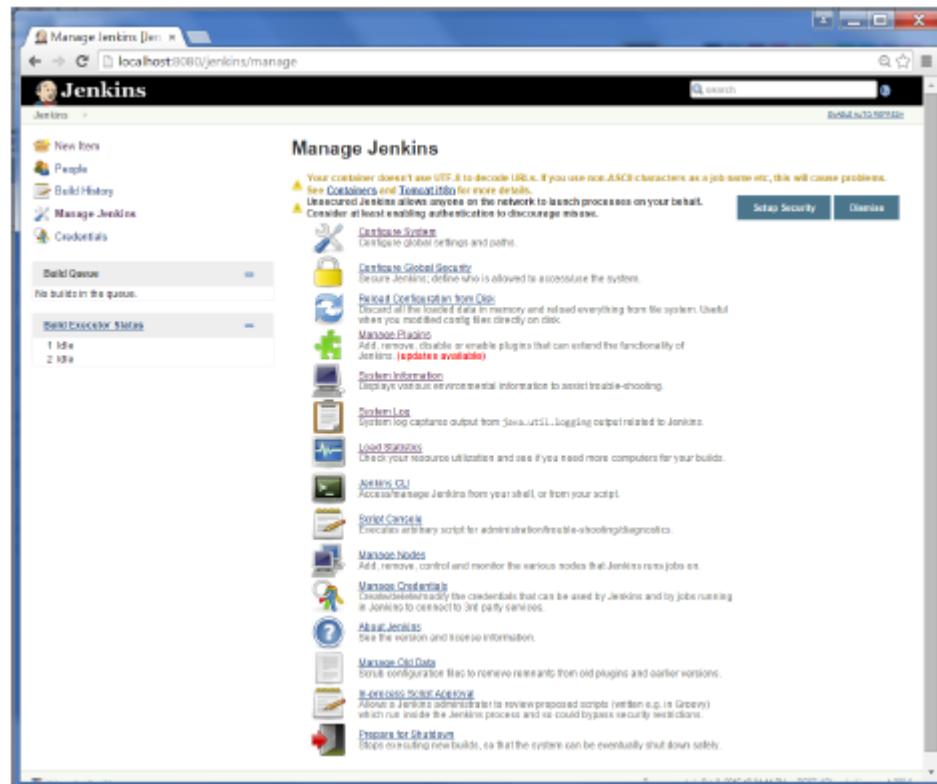
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

Step 2: Setting up Jenkins and Maven

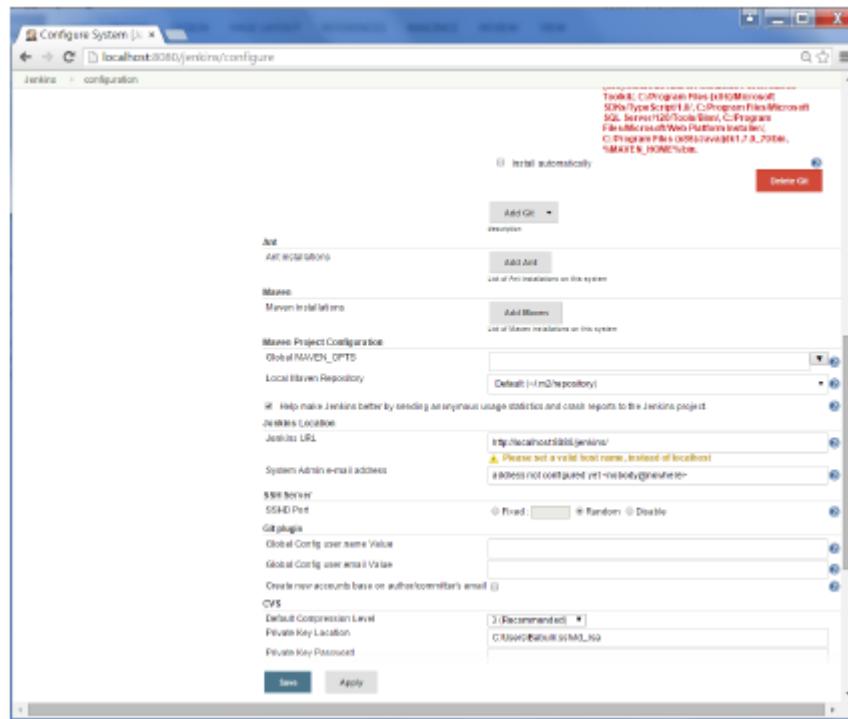
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.

The screenshot shows the Jenkins dashboard at <http://localhost:8080/jenkins/>. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. It also displays 'Build Queue' (0 items) and 'Build Executor Status' (2 items, both at 0%). The main content area says 'Welcome to Jenkins!' and prompts the user to 'create new jobs'. The 'Manage Jenkins' link is located in the sidebar.

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



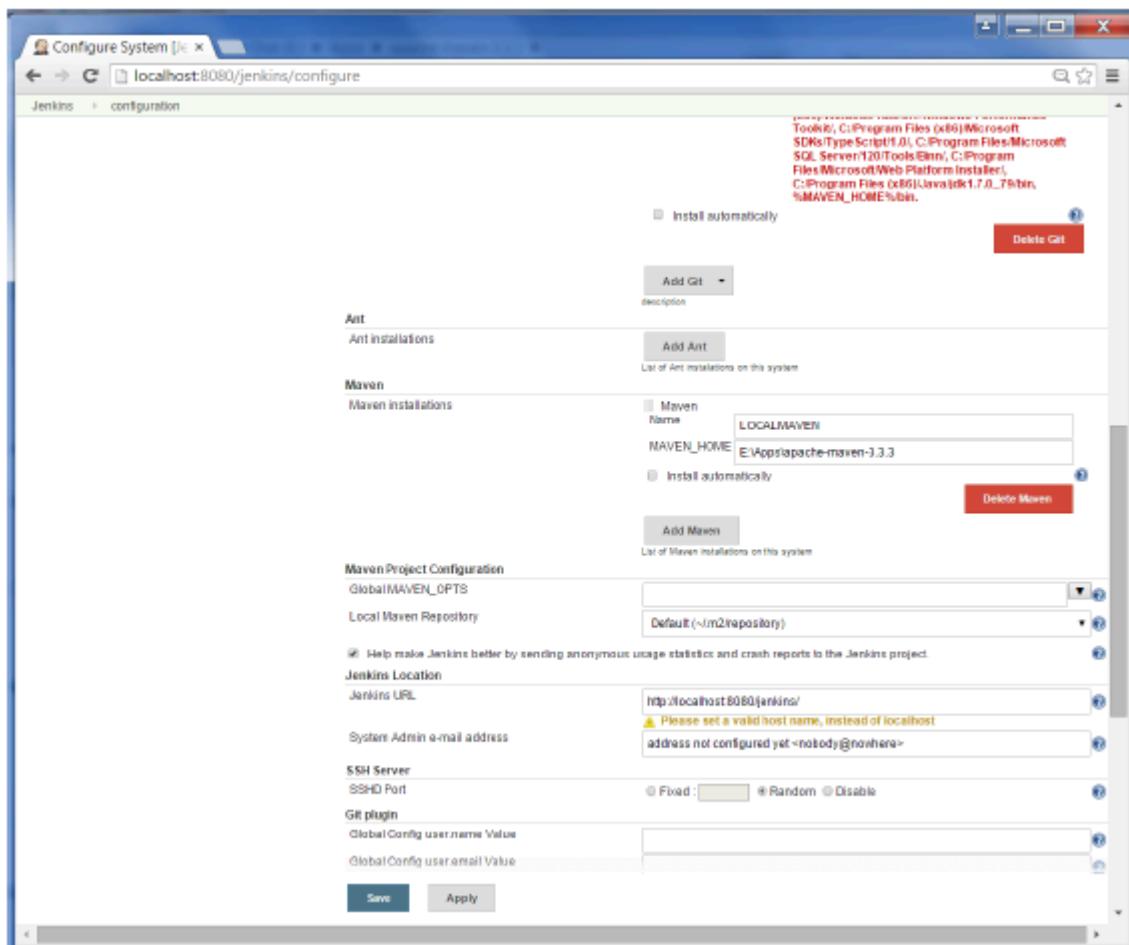
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



Uncheck the 'Install automatically' option.

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

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



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

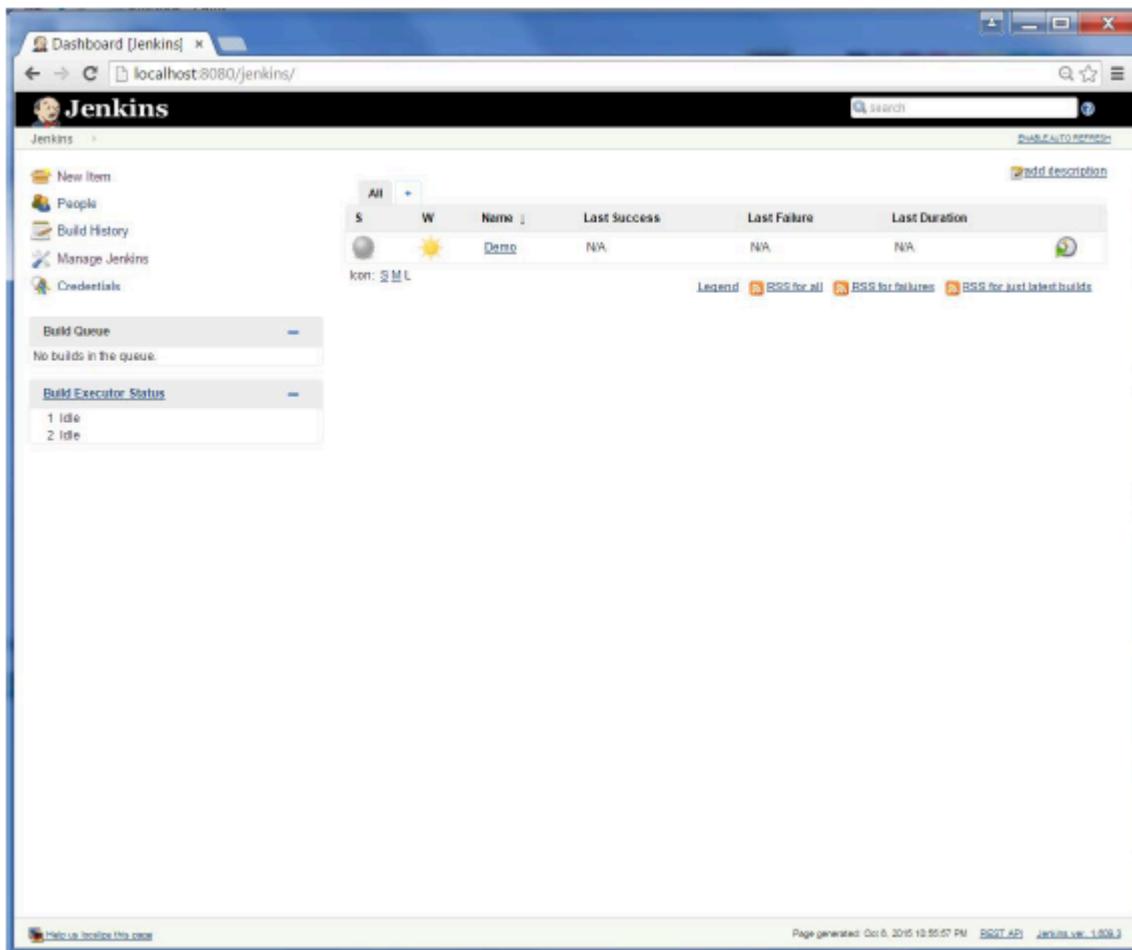
The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The main content area displays a table of build items. There is one item named "Demo" with a yellow sun icon, indicating it is currently building. The table columns are S, W, Name, Last Success, Last Failure, and Last Duration. A legend at the bottom right shows three RSS feed icons: RSS for all, RSS for failures, and RSS for just latest builds.

The screenshot shows the "New Item" dialog in Jenkins at localhost:8080/jenkins/view/All/newJob. The "Item name" field is filled with "MavenDemo". Below it, there are five options: "Freestyle project", "Maven project", "External Job", "Multi-configuration project", and "Copy existing item". The "Freestyle project" option is selected. The "OK" button is visible at the bottom right of the dialog.

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 –

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are 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 links with icons:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins, define who is allowed to access the system.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from java.util.Logging output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/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.
- Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.
- In-progress Script Approval**: Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Prepare for Shutdown**: Stops executing new builds, so that the system can be eventually shut down safely.

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

Jenkins Home Directory

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

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

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

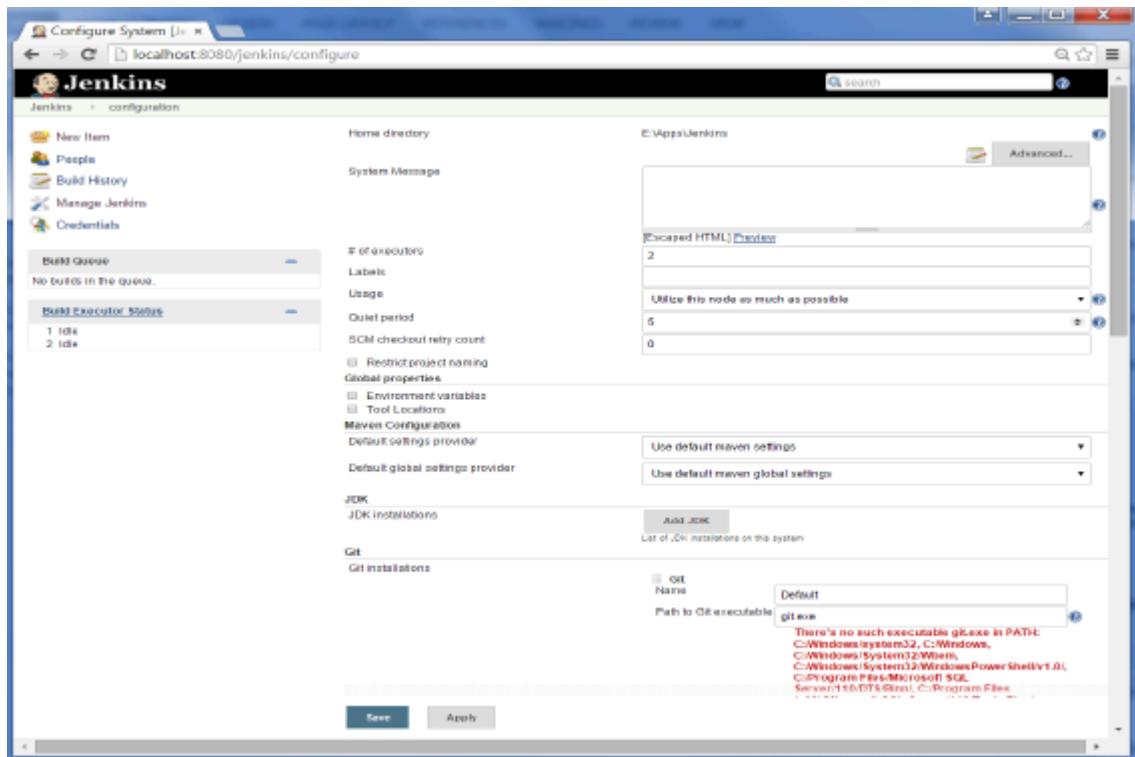
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/jenkins` to this new directory.

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

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

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

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



of executors

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

Environment Variables

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

Jenkins URL

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

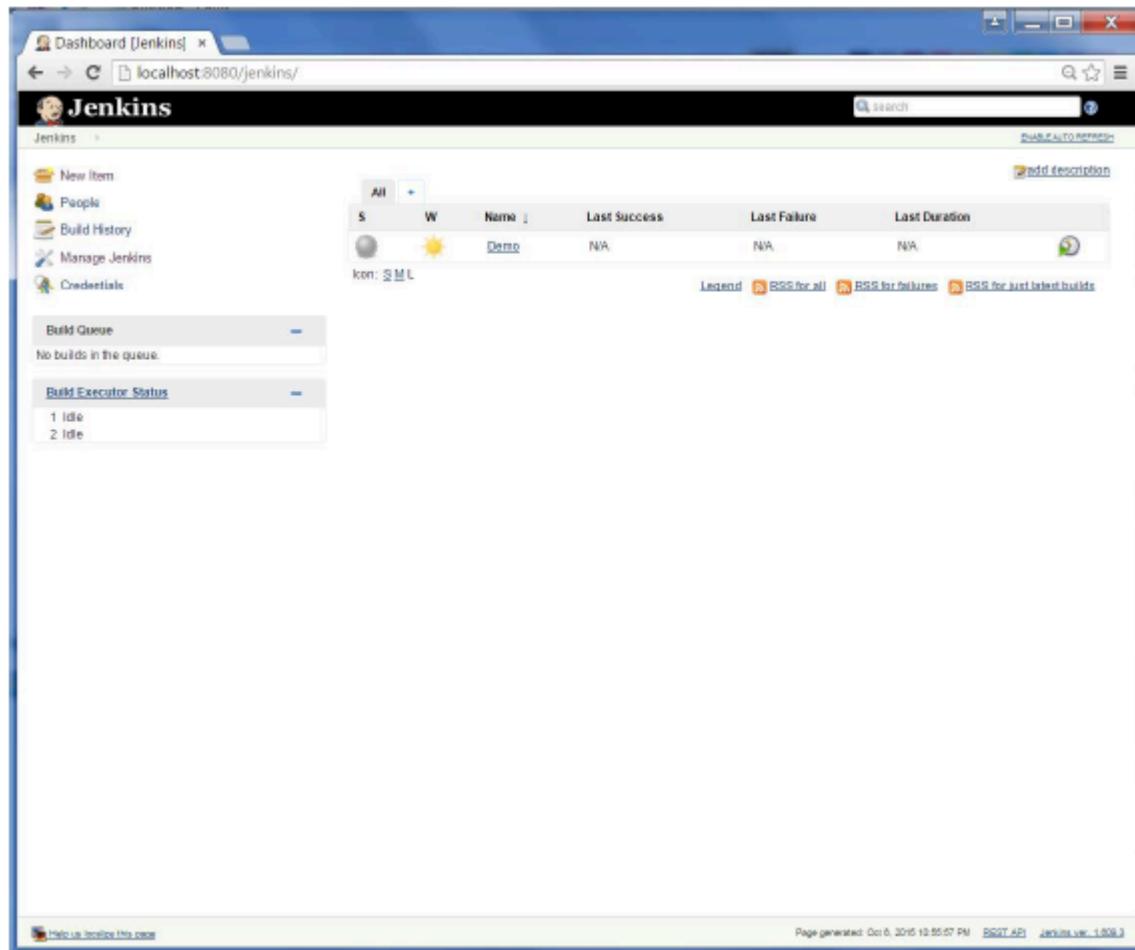
Email Notification

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

Jenkins - Management

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

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



You will then be presented with the following screen –

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are two 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 a warning about non-ASCII characters in job names. It lists several management options with icons and descriptions:

- Configure System: Configure global settings and paths.
- Configure Global Security: Secure Jenkins; define who is allowed to access the system.
- Reload Configuration from Disk: Reload 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/Manage 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.
- Manage Old Data: Scrub configuration files to remove remnants from old plugins and earlier versions.
- In-progress Script Approval: Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Prepare for Shutdown: Stops executing new builds, so that the system can be eventually shut down safely.

Some of the management options are as follows –

Configure System

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

Reload Configuration from Disk

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

Manage Plugin

Here one can install a wide variety of third-party plugins right from different Source code management tools such as Git, Mercurial or ClearCase, to code quality and code

coverage metrics reporting. Plugins can be installed, updated and removed through the Manage Plugins screen.

The screenshot shows the Jenkins Update Center interface. At the top, there's a navigation bar with links for 'Back to Dashboard' and 'Manage Jenkins'. Below this is a search bar and a 'Filter' input field. The main content area has tabs for 'Updates', 'Available', 'Installed', and 'Advanced', with 'Updates' selected. A table lists various Jenkins plugins:

Name	Version	Installed
CVS Plug-in	2.12	2.11
Javadoc Plugin	1.3	1.1
JUnit Plugin	1.9	1.2-beta-4
Matrix Authorization Strategy Plugin	1.2	1.1
Matrix Project Plugin	1.6	1.4.1
Maven Integration plugin	2.12.1	2.7.1
OWASP Markup Formatter Plugin	1.3	1.9
PAM Authentication plugin	1.2	1.1
Script Security Plugin	1.15	1.13
SSH Slaves plugin	1.10	1.9
This plugin allows you to manage slaves running on Unix machines over SSH.		
Subversion Plugin	2.5.3	1.54
This plugin adds the Subversion support (via SVNKIT) to Jenkins.		
Translation Assistance plugin	1.12	1.10
This plugin adds an additional dialog box in every page, which enables people to contribute localizations for the messages they are seeing in the current page.		
Windows Slaves Plugin	1.1	1.0
Allows you to connect to Windows machines and start slave agents on them.		

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

Below the table, there's a note: 'Select All None This page lists updates to the plugins you currently use.'

At the very bottom of the page, there are links for 'Help us Improve TutorialsPoint', 'Page generated: Oct 6, 2016 11:08:25 PM', 'REST API', and 'Jenkins ver. 1.808.3'.

System Information

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

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

The screenshot shows the Jenkins System Information page at localhost:8080/jenkins/systemInfo. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. Under Build Queue and Build Executor Status, there are no builds in the queue or idle executors. The main content is the 'System Properties' table:

Name	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Appstomcat7
catalina.home	E:\Appstomcat7
catalina.useNaming	true
common.loader	\$catalina.base\$lib;\$catalina.base\$lib\$jar;\$catalina.home\$lib;\$catalina.home\$lib\$jar
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Appstomcat7\bin\bootstrap.jar;E:\Appstomcat7\bin\tomcat-juli.jar
java.class.version	51.0
java.endorsed.dirs	E:\Appstomcat7\endorsed
java.ext.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\SunJava\lib\ext
java.home	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.io.tmpdir	E:\Appstomcat7\temp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\SunJava\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32;C:\Windows\System32\Wbem;C:\Windows\System32\WMI;C:\Windows\System32\Threading;C:\Windows\System32\RPC;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\ManagementStudio;C:\Program Files (x86)\Microsoft Visual Studio\10.0\Common7\IDE\PrivateAssemblies;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files (x86)\Windows Kits\8.1\Windows Performance Toolkit;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin;
java.naming.factory.initial	org.apache.naming.java.javaURLContextFactory
java.naming.factory.url.pkgs	org.apache.naming
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.7_0_79-b15
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.7
java.util.logging.config.file	E:\Appstomcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url_bug	http://bugreport.sun.com/bugreport/
java.version	1.7.0_79
java.vm.info	mixed mode, sharing

System Log

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

Load Statistics

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

Script Console

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

Manage nodes

Jenkins is capable of handling parallel and distributed builds. In this screen, you can configure how many builds you want. Jenkins runs simultaneously, and, if you are using

distributed builds, set up build nodes. A build node is another machine that Jenkins can use to execute its builds.

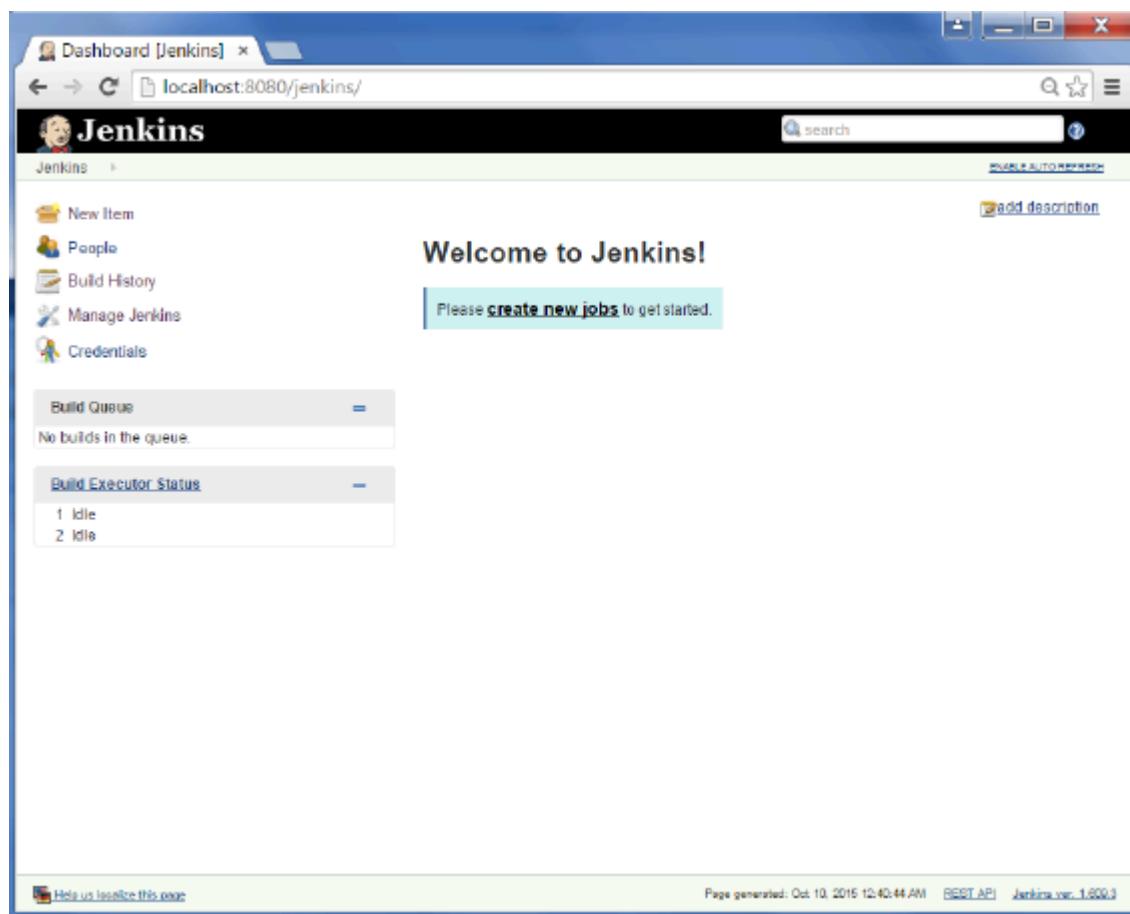
Prepare for Shutdown

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

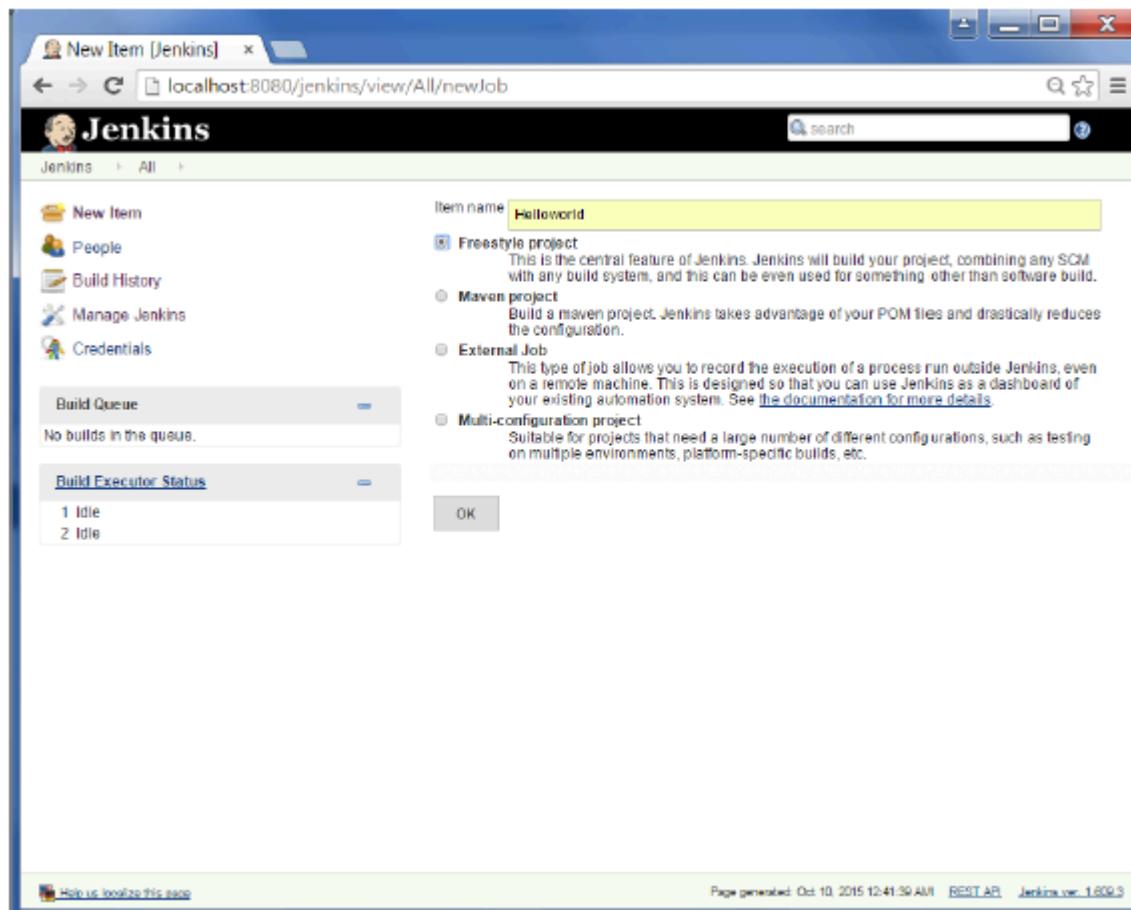
Jenkins - Setup Build Jobs

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

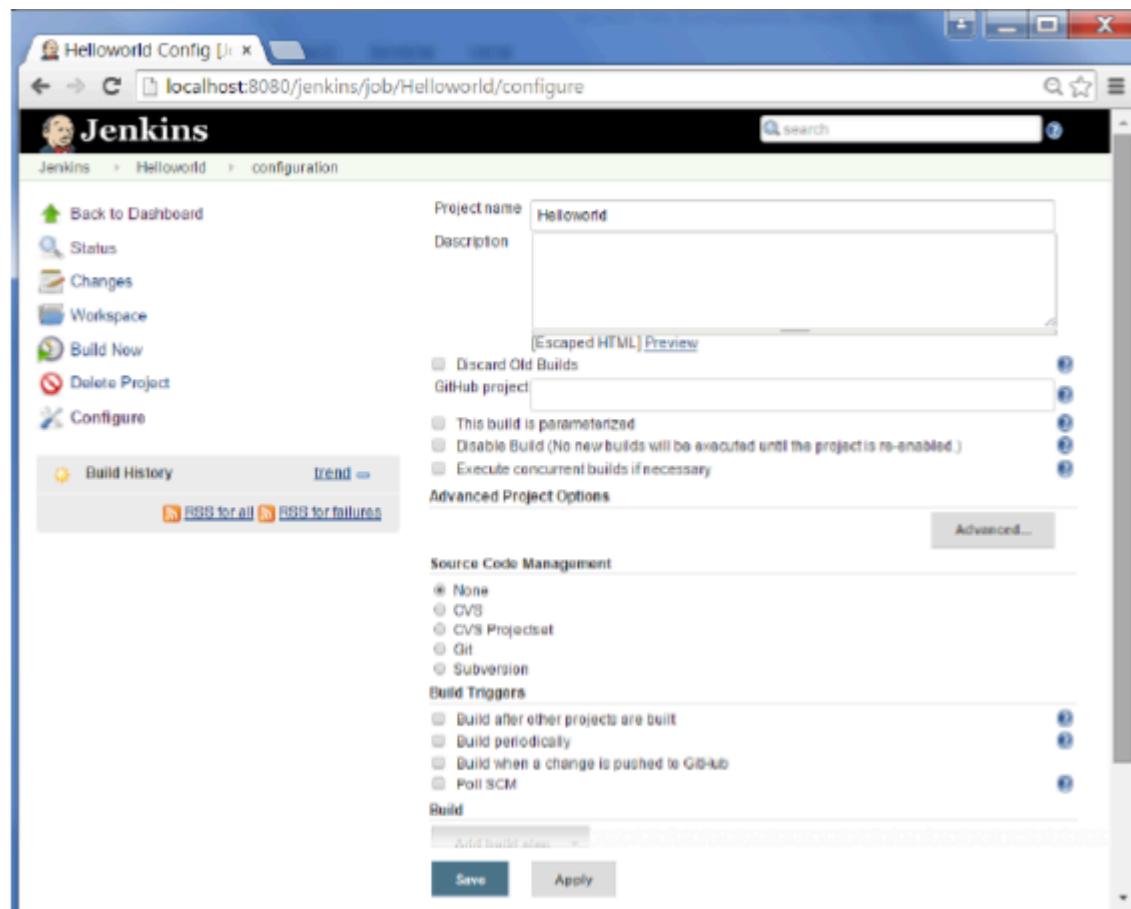
Step 1 – Go to the Jenkins dashboard and Click on New Item



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

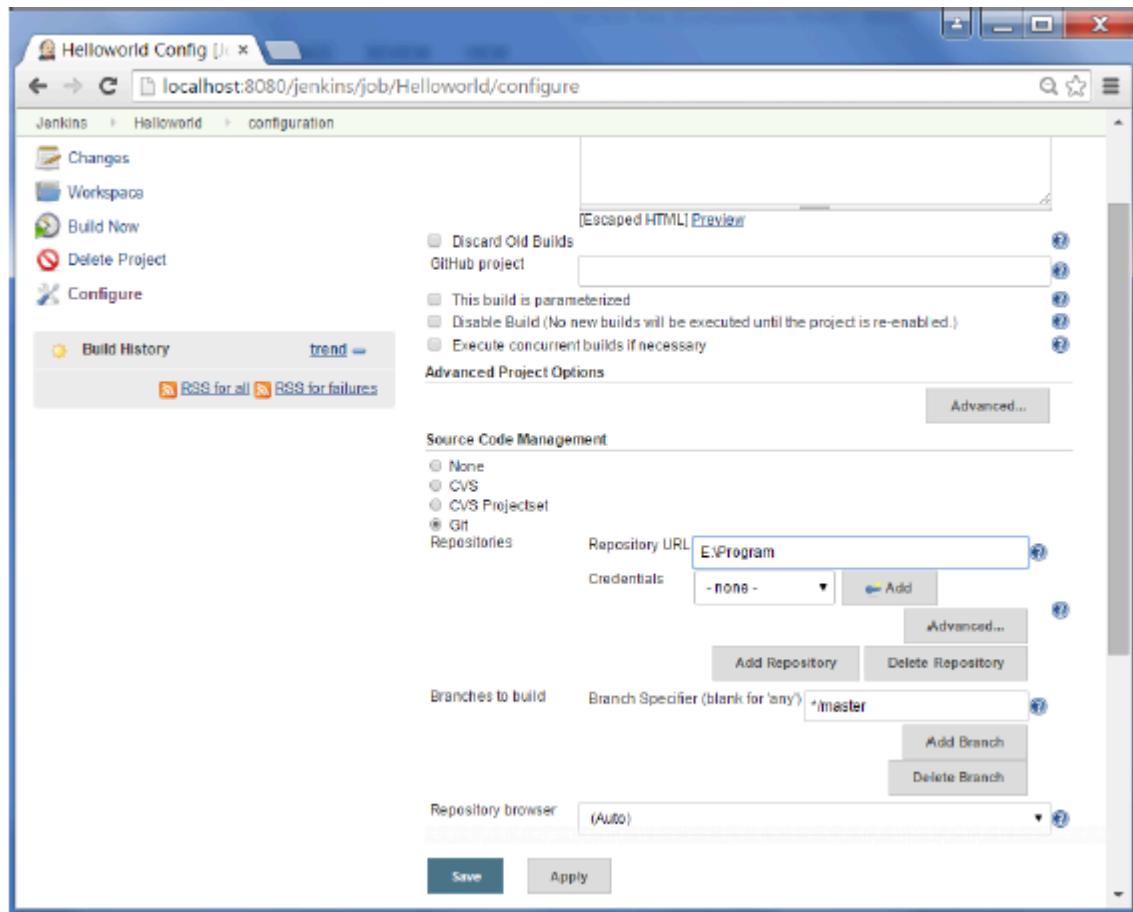


Step 3 – The following screen will come up in which you can specify the details of the job.

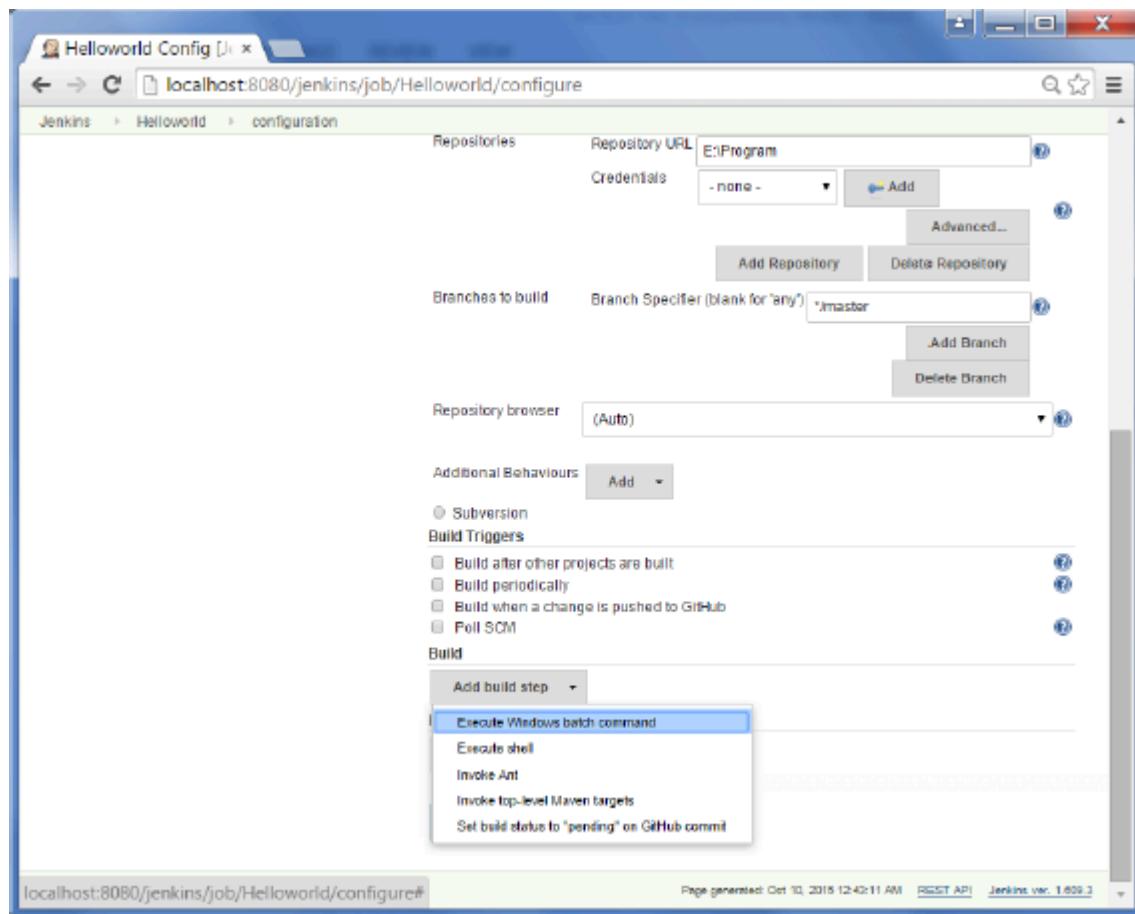


Step 4 – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

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



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



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

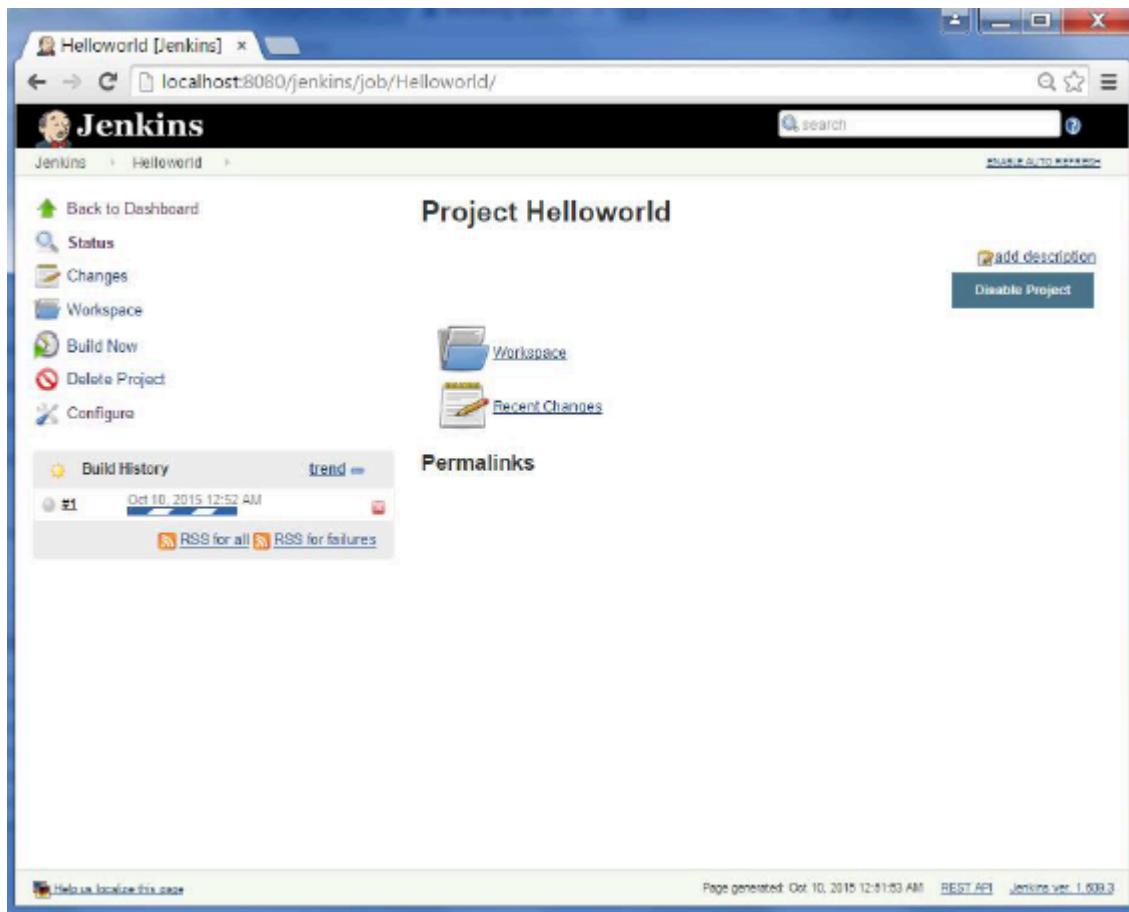
```
javac HelloWorld.java  
java HelloWorld
```

The screenshot shows the Jenkins configuration interface for the 'Helloworld Config' job. The 'Build' section is expanded, showing a 'Command' field containing the Java command to build the project. The 'Save' and 'Apply' buttons are visible at the bottom.

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

The screenshot shows the Jenkins project page for 'Helloworld'. It displays the 'Project Helloworld' header, a sidebar with navigation links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below the sidebar are 'Build History' and 'Permalinks' sections, along with RSS feed links. A 'Disable Project' button is located in the top right corner.

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. The title bar says 'Helloworld [Jenkins]'. The URL in the address bar is 'localhost:8080/jenkins/job/Helloworld/'. The main content area is titled 'Project Helloworld'. On the left, there's a sidebar with links: '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 'Workspace' icon and a 'Recent Changes' link. At the bottom of the page, there are links for 'RSS for all' and 'RSS for failures'. The footer contains links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.600.3'.

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 like '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. Underneath it, there are two sections: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). A 'Permalinks' section follows, showing a build history entry for 'Oct 10, 2015 12:52 AM' with an 'RSS for all' and 'RSS for failures' link. At the bottom, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.609.3'.

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

The screenshot shows the Jenkins interface for the first build of the 'Helloworld' project. The title is 'Build #1 (Oct 10, 2015 12:52:50 AM)'. It indicates the build started 4 min 40 sec ago and took 4.7 sec. The build status is green. The left sidebar includes links for 'Back to Project', 'Status', 'Changes', 'Console Output' (which is highlighted in blue), 'Edit Build Information', 'Delete Build', 'Git Build Data', and 'No Tags'. The main content area shows 'No changes.' (with a document icon), 'Started by anonymous user' (with a person icon), and the git revision '42f0a82ffadd86fb5c3a9dfae40e731a907f5c8f' with a commit message 'refs/remotes/origin/master'. The footer is identical to the previous screenshot.

The screenshot shows a Jenkins job named 'Helloworld' with build number '#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 command-line log for the build. The log shows the Jenkins user anonymous starting the build, cloning the repository from 'origin/master', and then executing Java code to print 'Hello World'. The build completed successfully.

```
Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program +refs/heads/*:refs/remotes/origin/*
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffad08ef05c3a9dfeae40e731a907f5c8f
(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
42f9a82ffad08ef05c3a9dfeae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffad08ef05c3a9dfeae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat

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

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

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

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

Jenkins - Unit Testing

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

xUnit Plugin - Jenkins

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

Dashboard > Jenkins > Plugins > xUnit Plugin

Browse Search

xUnit Plugin

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

Jenkins

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

Plugin Information

Plugin ID	xunit	Changes	In Latest Release Since Latest Release																									
Latest Release	1.98 (archives)																											
Latest Release Date	Oct 09, 2015																											
Required Core Dependencies	junit (version:1.6)																											
Usage	<p>xunit - installations</p> <table border="1"> <caption>xunit - installations</caption> <thead> <tr> <th>Month</th> <th>Installations</th> </tr> </thead> <tbody> <tr><td>10</td><td>12000</td></tr> <tr><td>11</td><td>11500</td></tr> <tr><td>12</td><td>11500</td></tr> <tr><td>01</td><td>11500</td></tr> <tr><td>02</td><td>11500</td></tr> <tr><td>03</td><td>11500</td></tr> <tr><td>04</td><td>11500</td></tr> <tr><td>05</td><td>11500</td></tr> <tr><td>06</td><td>11500</td></tr> <tr><td>07</td><td>11500</td></tr> <tr><td>08</td><td>11500</td></tr> <tr><td>09</td><td>11500</td></tr> </tbody> </table>	Month	Installations	10	12000	11	11500	12	11500	01	11500	02	11500	03	11500	04	11500	05	11500	06	11500	07	11500	08	11500	09	11500	Installations
Month	Installations																											
10	12000																											
11	11500																											
12	11500																											
01	11500																											
02	11500																											
03	11500																											
04	11500																											
05	11500																											
06	11500																											
07	11500																											
08	11500																											
09	11500																											
			2014-Oct 11692 2014-Nov 11557 2014-Dec 11631 2015-Jan 12106 2015-Feb 12262 2015-Mar 12891 2015-Apr 12894 2015-May 12716 2015-Jun 13143 2015-Jul 13470 2015-Aug 13192 2015-Sep 13563																									

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

CppUnit output

xUnit Plugin - Jenkins

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

Features

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

Supported tools

Embedded tools

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

Other plugins as an extension of the xUnit plugin:

- * Gallo (Gallo plugin)
- * Parasoft C++Test tool (CppUnit Plugin)
- * JSUnit (JSUnit Plugin)
- * JBehave
- * TestComplete (TestComplete xUnit Plugin)

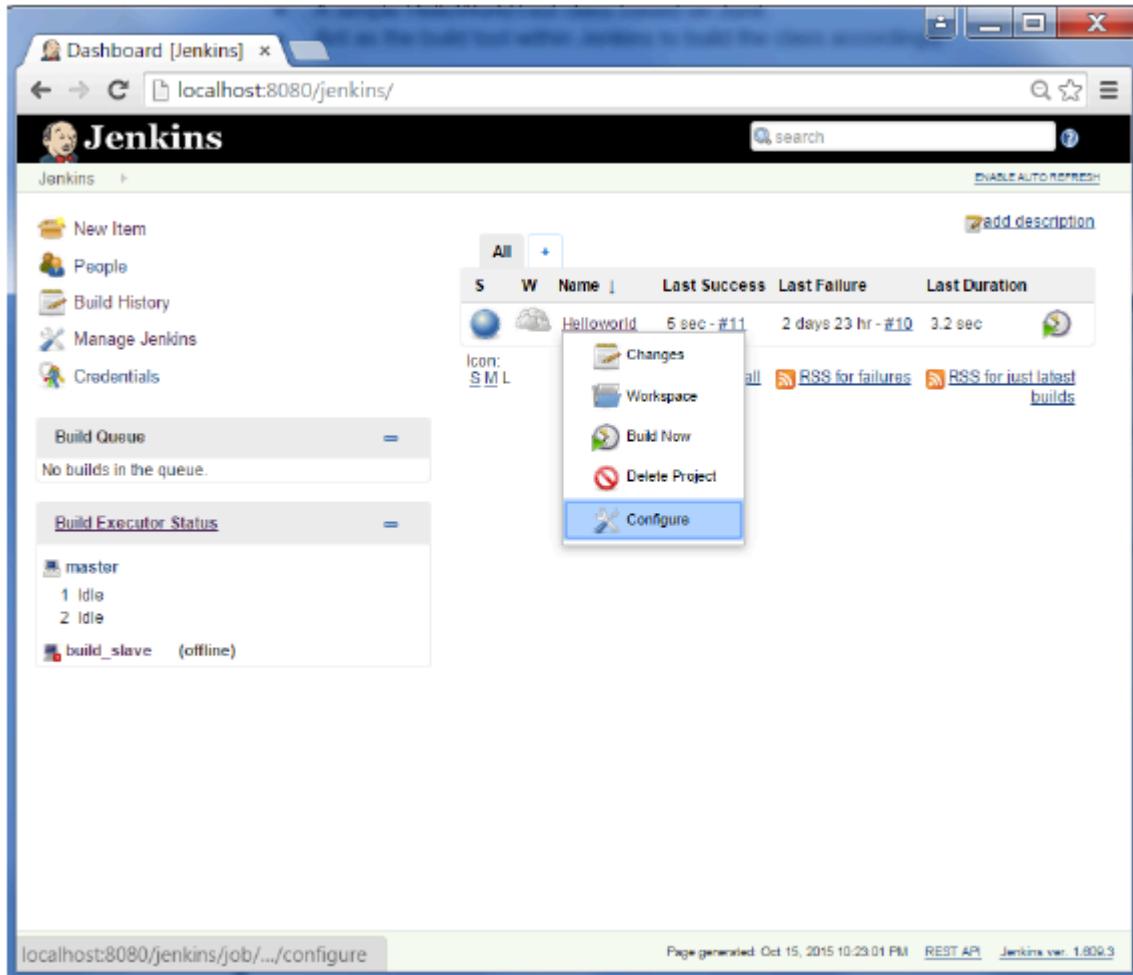
External contributions

Example of a Junit Test in Jenkins

The following example will consider

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

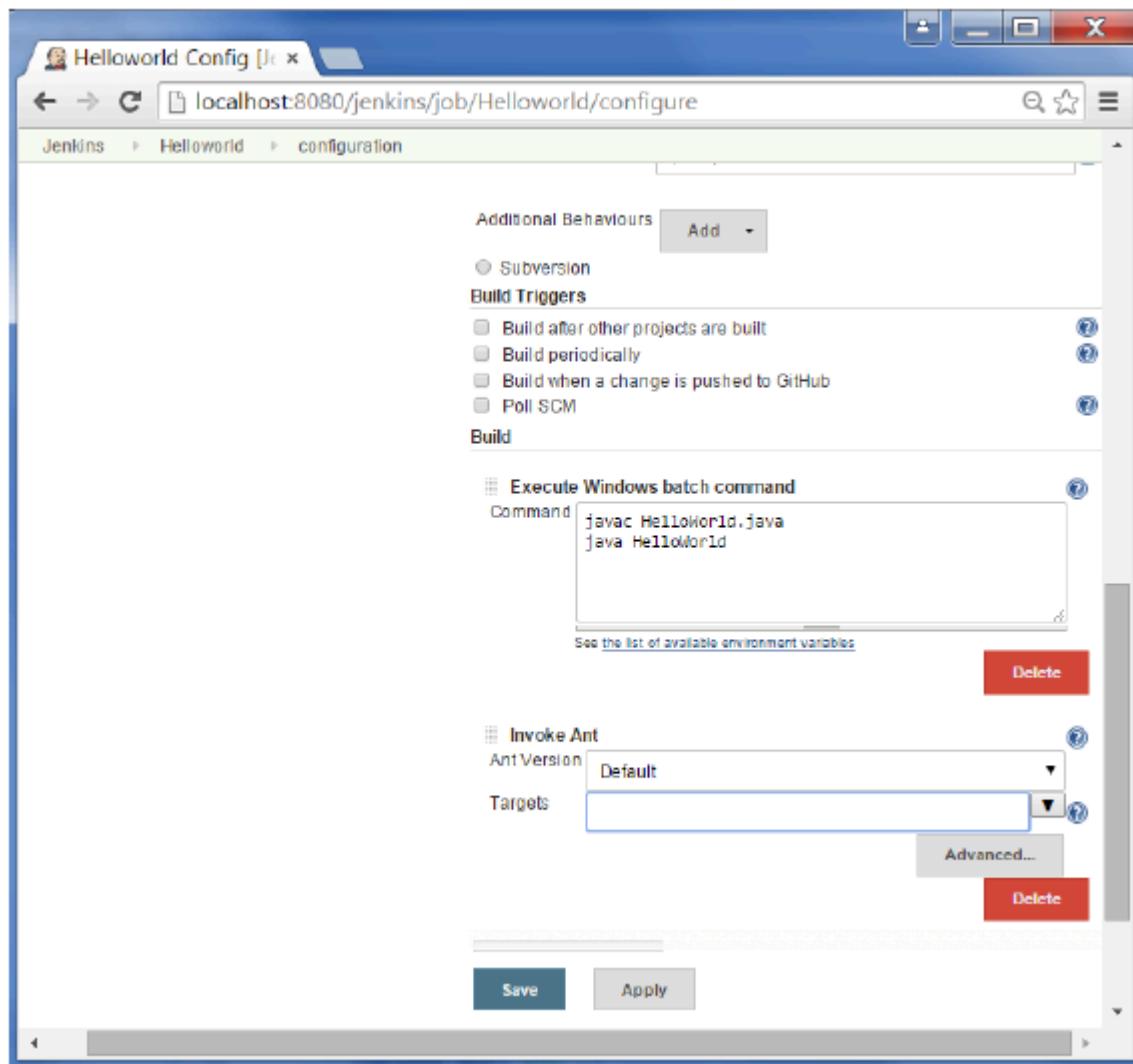
Step 1 – Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option



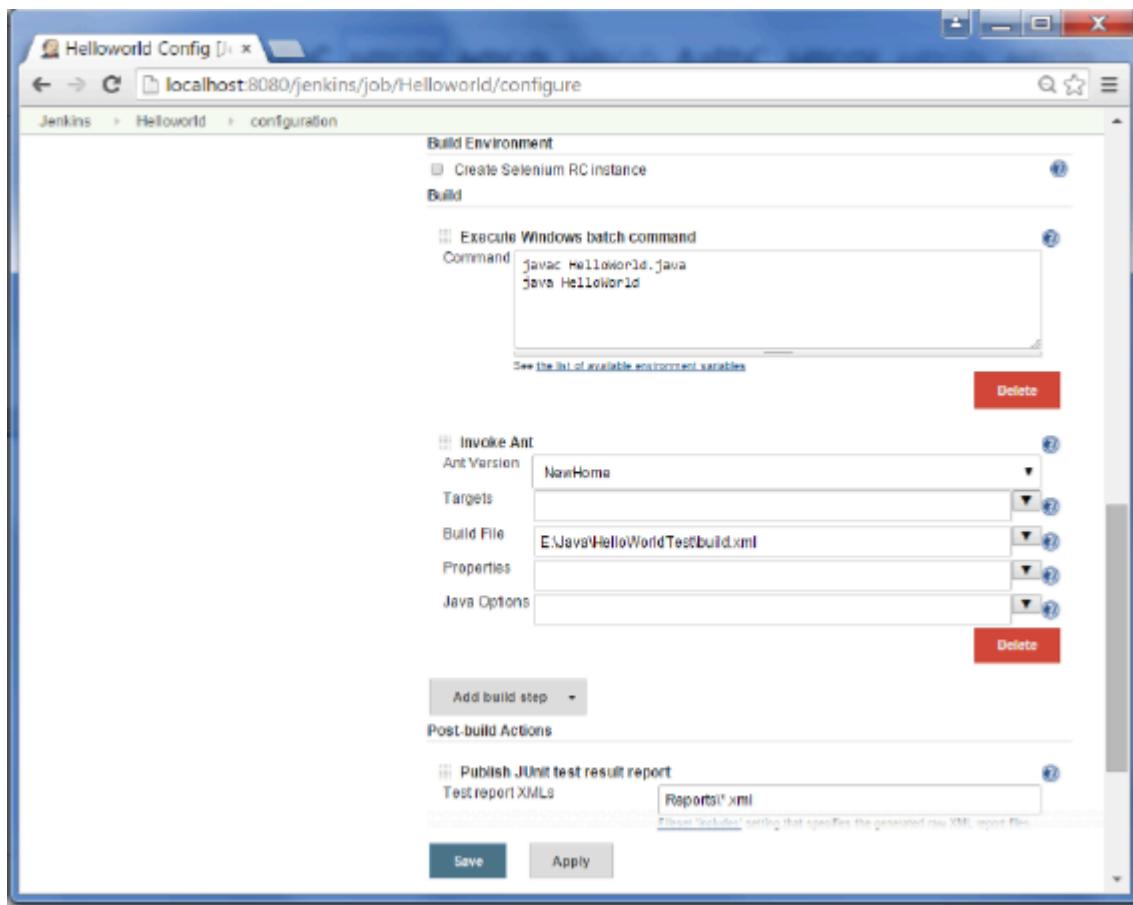
Step 2 – Browse to the section to Add a Build step and choose the option to Invoke Ant.

The screenshot shows the Jenkins configuration page for a job named "Helloworld". The URL is `localhost:8080/jenkins/job/Helloworld/configure`. The "Build" section is expanded, showing a command input field containing `javac HelloWorld.java
java HelloWorld`. A tooltip for environment variables is visible below the command. A "Delete" button is located to the right of the command field. A dropdown menu titled "Add build step" is open, with "Invoke Ant" selected. Other options in the dropdown include "Execute Windows batch command", "Execute shell", "Invoke top-level Maven targets", and "Set build status to 'pending' on GitHub commit".

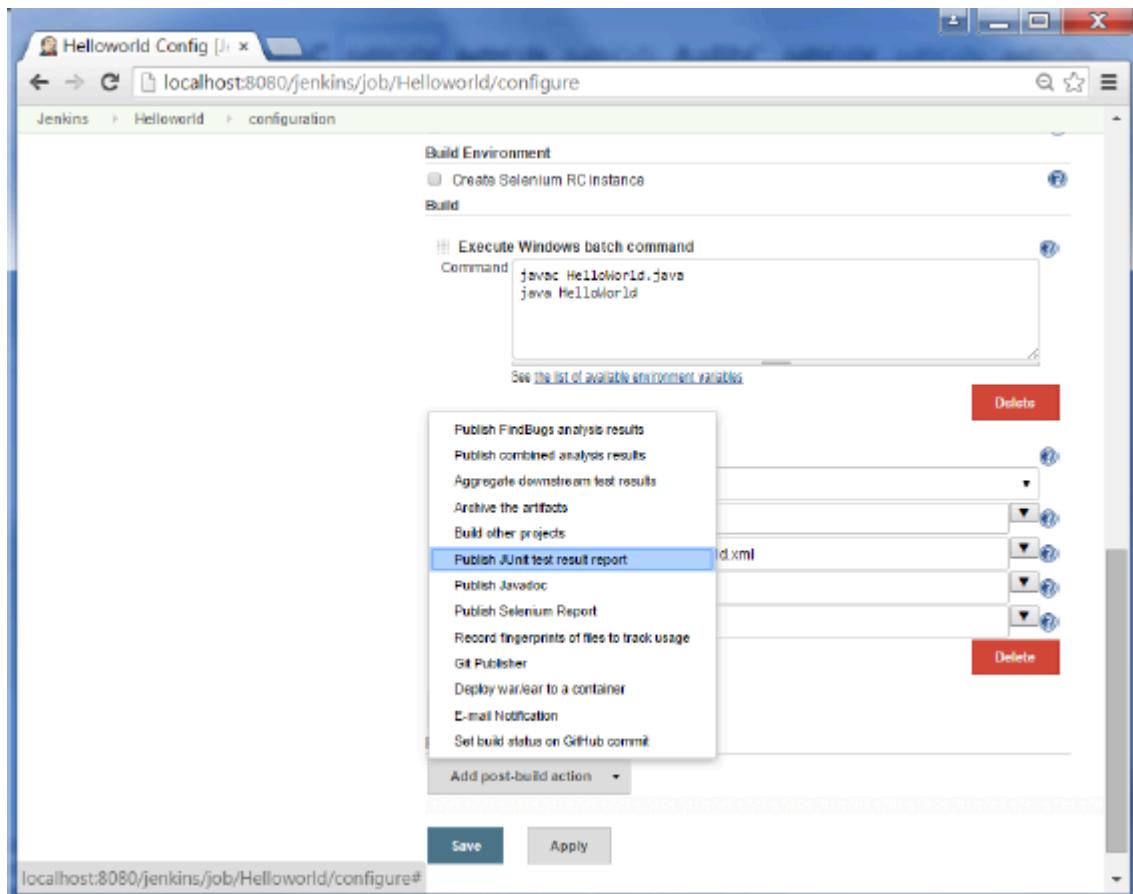
Step 3 – Click on the Advanced button.



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

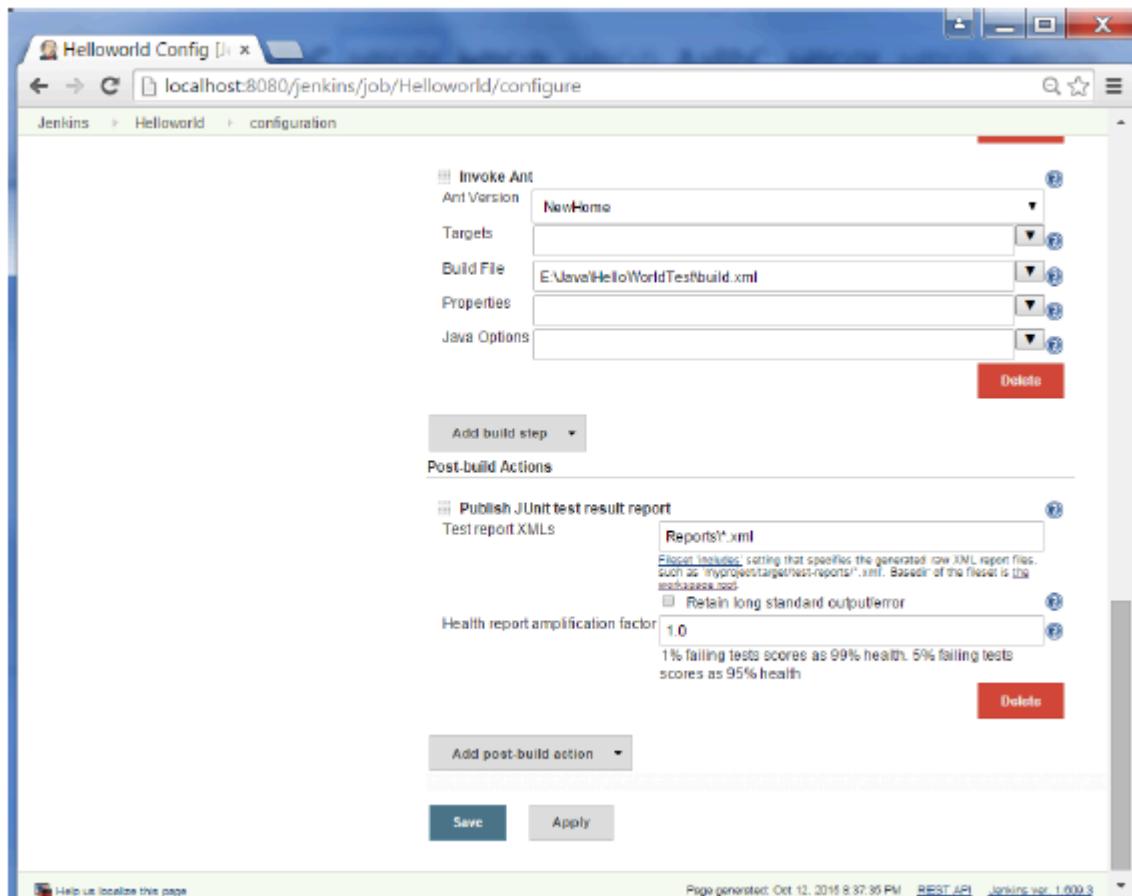


Step 5 – Next click the option to Add post-build option and choose the option of “Publish Junit test result report”



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

Once done, click the Save option at the end.



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

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

The screenshot shows the Jenkins interface for a build named "Helloworld #4". The main title is "Build #4 (Oct 12, 2015) 8:33:16 PM". It indicates the build was started 3 days 1 hr ago and took 3.9 sec on master. A "Status" icon shows "No changes." and it was started by an anonymous user. The revision is 42f9a82ffadd86fb5c3a0dfa40e731a8075c81, with a link to "refs/remotes/origin/master". Under "Test Result", there is one failure: "HelloWorldTestCase.InitializationError". The footer includes links for "Help us localize this page", "Page generated: Oct 15, 2015 10:24:38 PM", "REST API", and "Jenkins ver. 1.602.3".

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

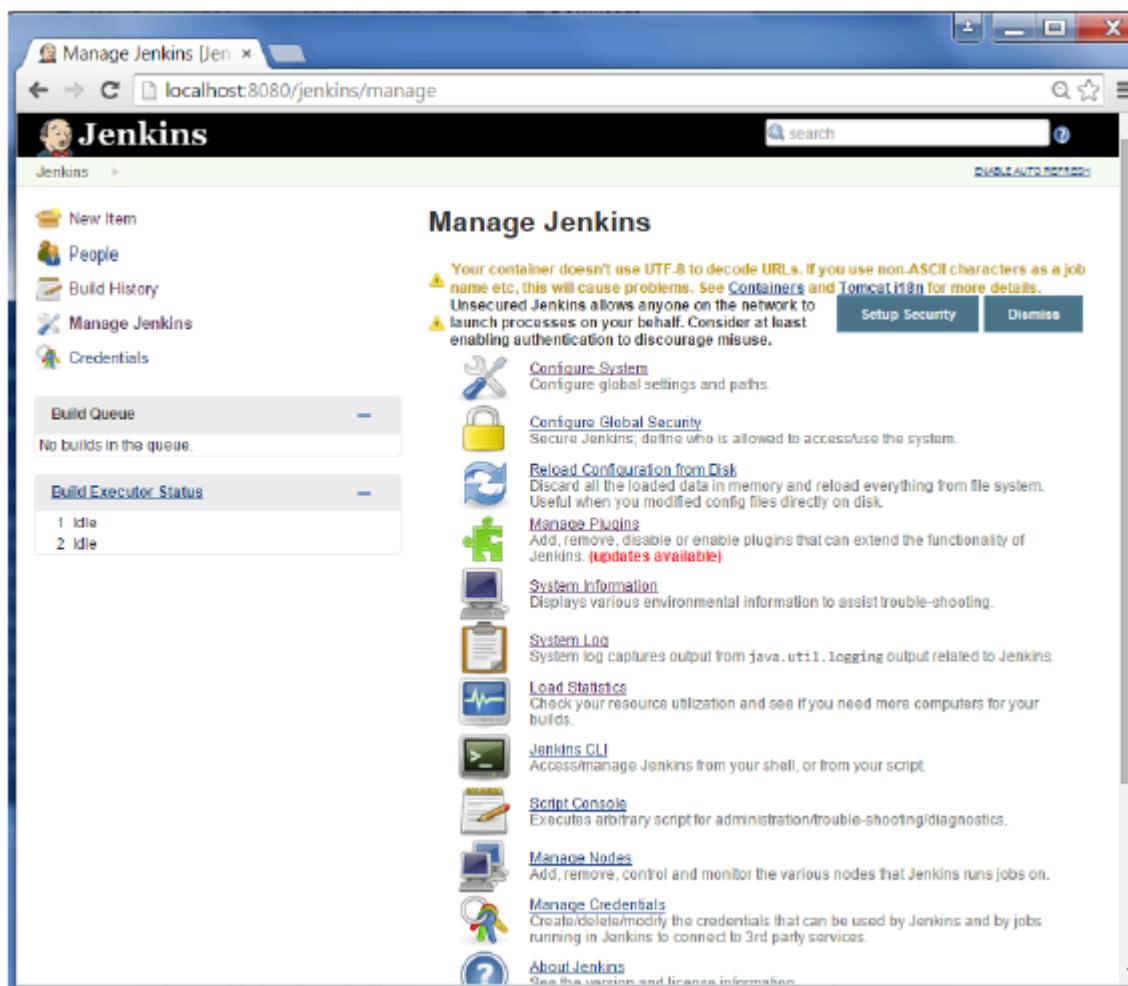
The screenshot shows the Jenkins interface for the "Test Results" of build "Helloworld #4". The main title is "Test Result" with "1 failures". It shows a single failed test: "HelloWorldTestCase.InitializationError" which took 10 ms. Below this, there is a section for "All Failed Tests" and a summary table for "All Tests". The table shows the following data:

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

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.



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

The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search bar at the top right contains the text 'selenium'. A list of available plugins is displayed, filtered by the search term. The visible plugins include:

- Hudson SeleniumHQ plugin**: This plugin allows you to run and load HTML Seleneese suites result generates by Selenium Server from [seleniumhq](#). Jenkins will generate the trend report of test result. The SeleniumHQ plug in can be [downloaded here](#).
- Selenium HTML report**: This plugin visualizes the results of selenium tests.
- TestingBot plugin**: This plugin allows for integration of [TestingBot](#) Selenium in Jenkins. TestingBot provides cross browser testing in the cloud.
- TestLink Plugin**: This plug-in integrates Jenkins and [TestLink](#) and generates reports on automated test execution. With this plug-in you can manage your tests in TestLink, schedule and control in Jenkins, and execute using your favorite test execution tool (TestPartner, Selenium, TestNG, PHPUnit, among others).
- Nirvana Plugin for Jenkins**: The Nirvana Jenkins plugin allows you to automate functional and cross browser Selenium testing of your web applications in [Nirvana cloud](#).
- Sauce OnDemand plugin**: This plugin allows you to integrate [Sauce Selenium Testing](#) with Jenkins.
- Selenium Builder plugin**: Invokes [Selenium Builder](#) scripts from a Jenkins build.
- SeleniumRC plugin**

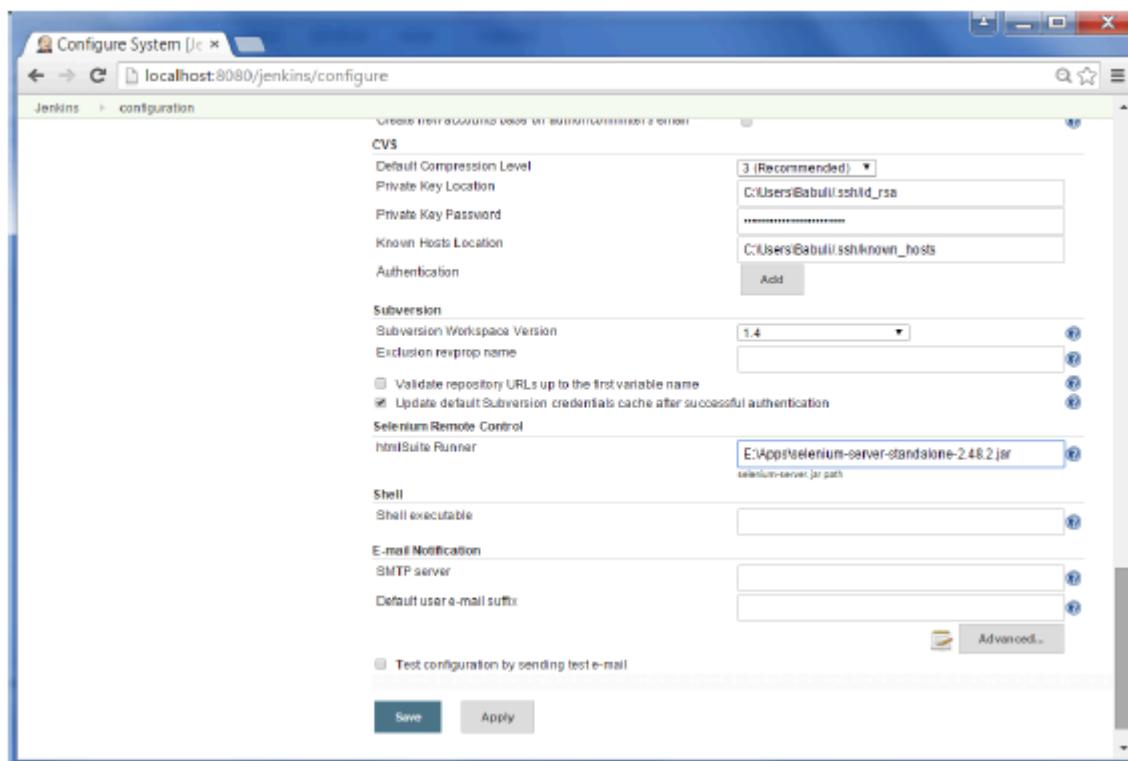
At the bottom of the list are three buttons: 'Install without restart', 'Download now and install after restart', and 'Update information obtained'.

Step 3 – Go to Configure system.

The screenshot shows the Jenkins 'Manage Jenkins' page. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected and highlighted in blue), and 'Credentials'. The main content area is titled 'Manage Jenkins' and contains several configuration links:

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

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



Note – The selenium jar file can be downloaded from the location [SeleniumHQ](#)

Click on the download for the Selenium standalone server.

A screenshot of a web browser window titled "Downloads" showing the SeleniumHQ website at "www.seleniumhq.org/download/". The main content area is titled "Downloads" and contains information about the Selenium Standalone Server. It states: "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)." Below this, there is a section for the "Selenium Standalone Server" which says: "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." It provides a link to "Download version 2.48.2". Further down, it says: "To use the Selenium Server in a Grid configuration see the [wiki page](#)". There are other sections for "The Internet Explorer Driver Server" and "Selenium Client & WebDriver Language Bindings". A sidebar on the left includes links for "Selenium Downloads", "Latest Releases", "Previous Releases", "Source Code", and "Maven Information". There are also sections for "Donate to Selenium" (with PayPal and other payment options) and "through sponsorship". A logo for "BrowserStack" is visible at the bottom.

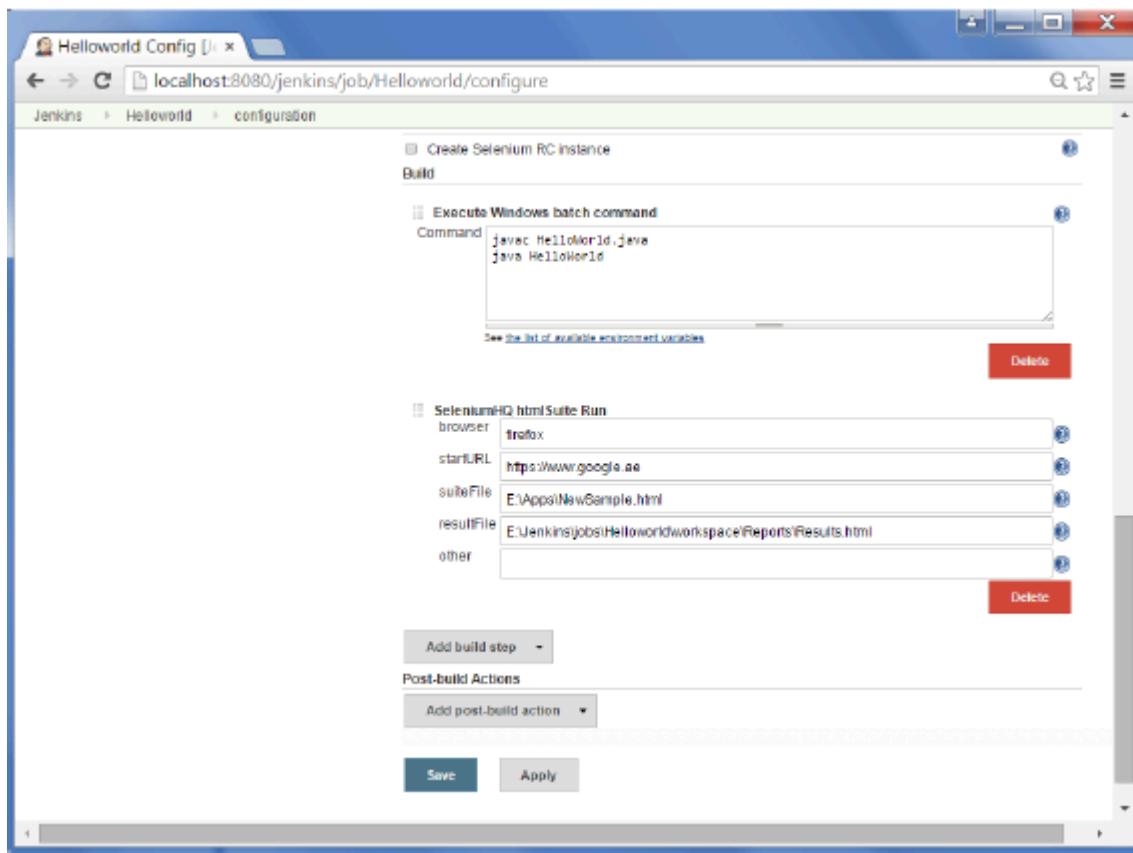
Step 5 – Go back to your dashboard and click on the Configure option for the HelloWorld project.

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area displays a table for the 'Helloworld' project, which has been last successful 23 hours ago, last failed 23 hours ago, and had a duration of 3.7 seconds. Below the table are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). A 'Configure' button is highlighted in blue. At the bottom, the URL is localhost:8080/jenkins/job/Helloworld/configure, and the page footer indicates it was generated on Oct 11, 2015 at 16:04 PM.

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

The screenshot shows the 'Helloworld' configuration page at localhost:8080/jenkins/job/Helloworld/configure. It includes sections for 'Repository browser' (set to '(Auto)'), 'Additional Behaviours' (with 'Subversion' selected), 'Build Triggers' (checkboxes for 'Build after other projects are built', 'Build periodically', 'Build when a change is pushed to GitHub', and 'Poll SCM'), and 'Build' (checkbox for 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'). A 'Delete' button is visible. A dropdown menu under 'Add build step' shows options like 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke top-level Maven targets', and 'SeleniumHQ htmlSuite Run', with 'SeleniumHQ htmlSuite Run' currently selected. The page footer shows the URL localhost:8080/jenkins/job/Helloworld/configure#.

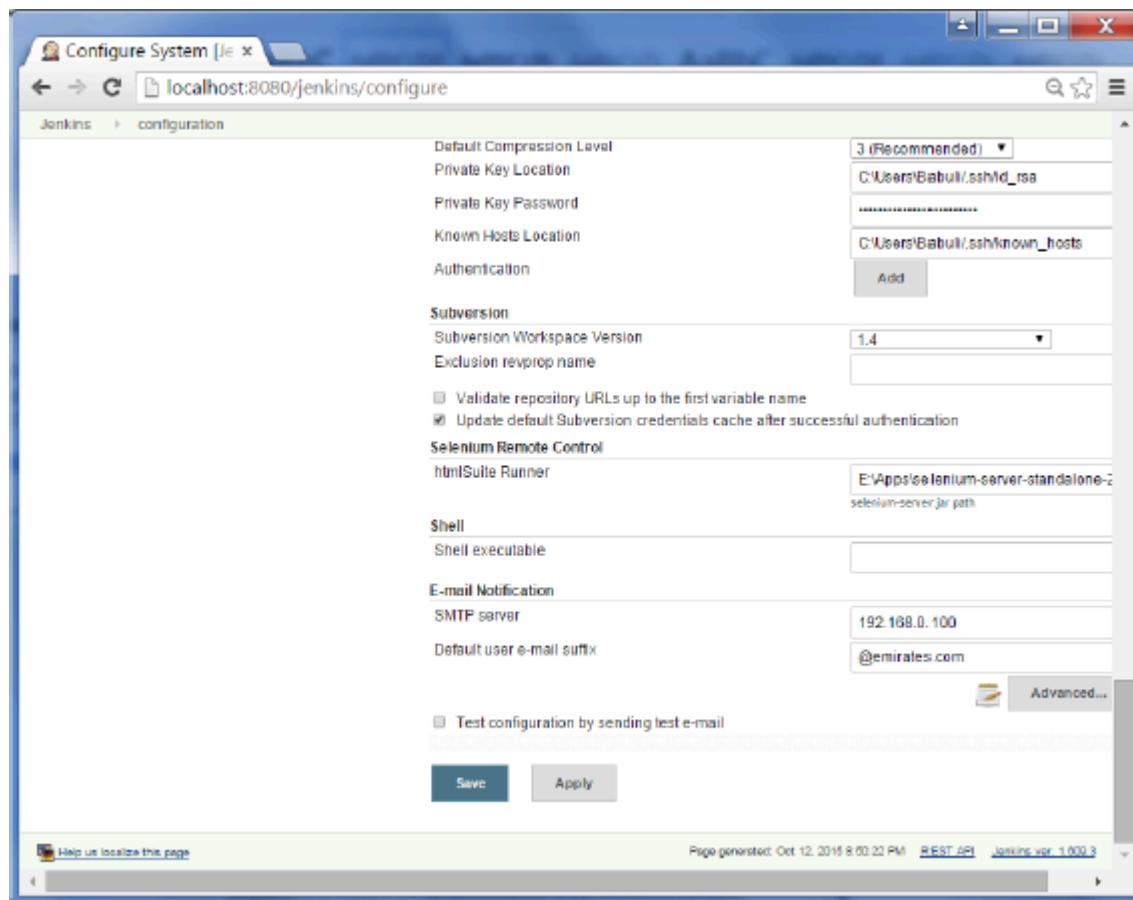
Step 7 – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



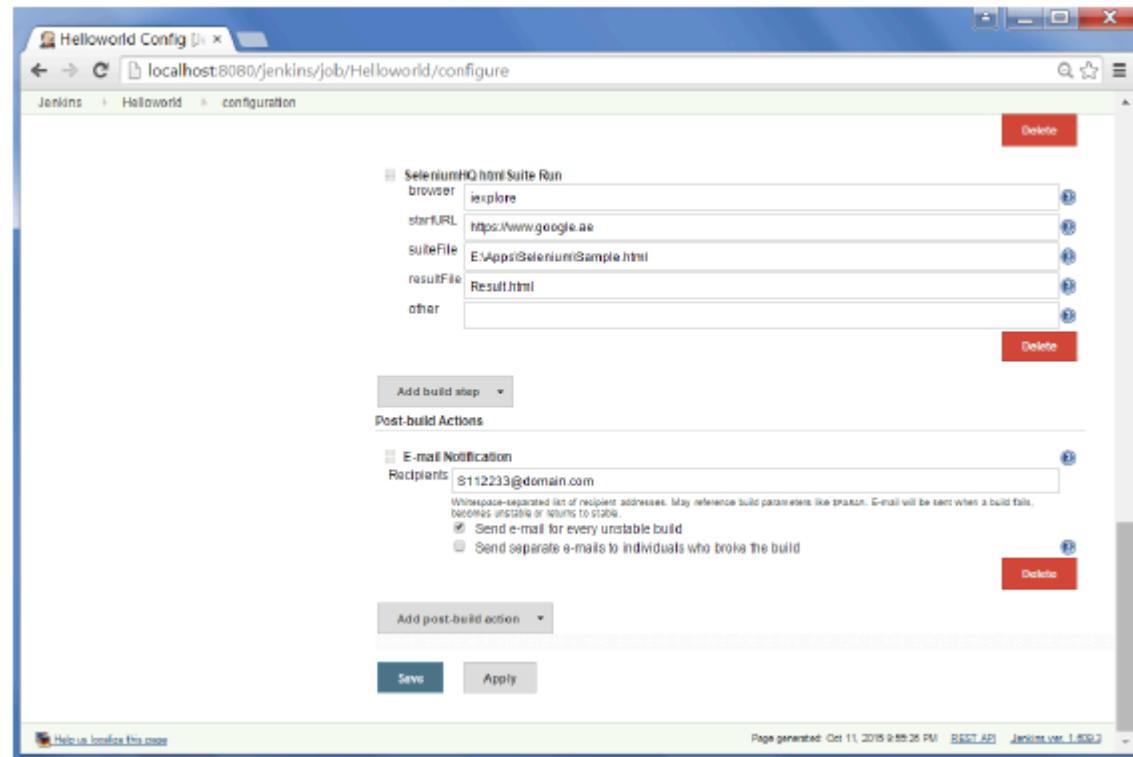
Jenkins - Notification

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

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

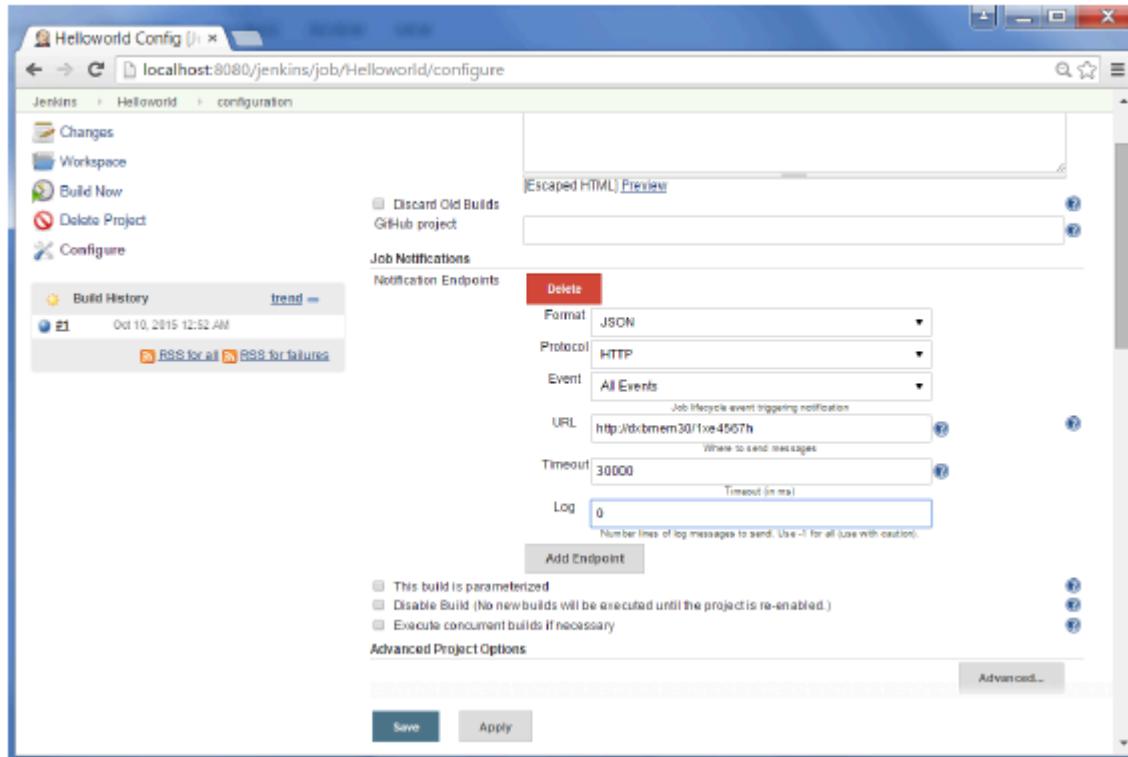


Step 2 – Configure the recipients in the Jenkins project - When you configure any Jenkins build project, right at the end is the ability to add recipients who would get email notifications for unstable or broken builds. Then click on the Save button.



Apart from the default, there are also notification plugin's available in the market. An example is the notification plugin from Tikal Knowledge which allows sending Job Status

notifications in JSON and XML formats. This plugin enables end-points to be configured as shown below.



Here are the details of each option –

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

Jenkins - Reporting

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

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

The screenshot shows the Jenkins configuration interface for a job named 'Helloworld'. In the 'Build' section, there is a step titled 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'. Below this, a context menu is open over a 'SeleniumHQ HTML Suite Run' step, showing options like 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects', and 'Publish JUnit test result report'. The 'Publish JUnit test result report' option is highlighted. At the bottom of the screen, there are 'Save' and 'Apply' buttons.

Jenkins - Code Analysis

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

The screenshot shows the 'Static Code Analysis Plug-ins' page on the Jenkins Wiki. The page title is 'Static Code Analysis Plug-ins'. It features a sidebar with links like 'Home', 'Mailing lists', 'Source code', 'Bugtracker', 'Security', 'Advisories', 'Events', 'Donation', 'Commercial Support', and 'Wiki Site Map'. The main content area displays 'Plugin Information' for the 'analysis-core' plugin. It includes details such as the latest release (1.74), latest release date (Sep 07, 2015), required core dependencies (ant, token-macro, maven-plugin, matrix-project, dashboard-view), and maintainers (Ulli Hafner). A chart titled 'analysis-core - installations' shows the number of installations per month from October 2014 to September 2015. The chart shows a steady increase from approximately 25,000 to 29,000 installations.

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

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

The plugins can provide information such as

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

Jenkins - Distributed Builds

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

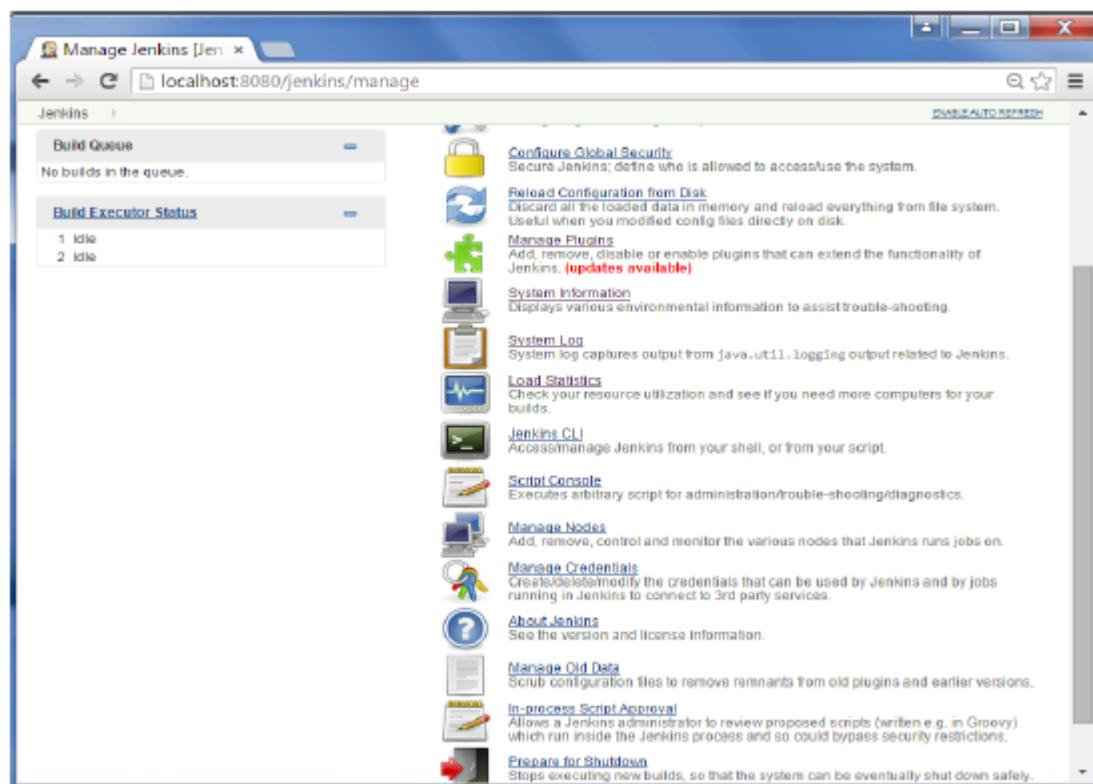
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

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

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

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



Step 2 – Click on New Node

A screenshot of the Jenkins Nodes page. The left sidebar includes 'Back to Dashboard', 'Manage Jenkins', 'New Node' (which is highlighted in blue), and 'Configure'. The main content area shows a table of nodes:

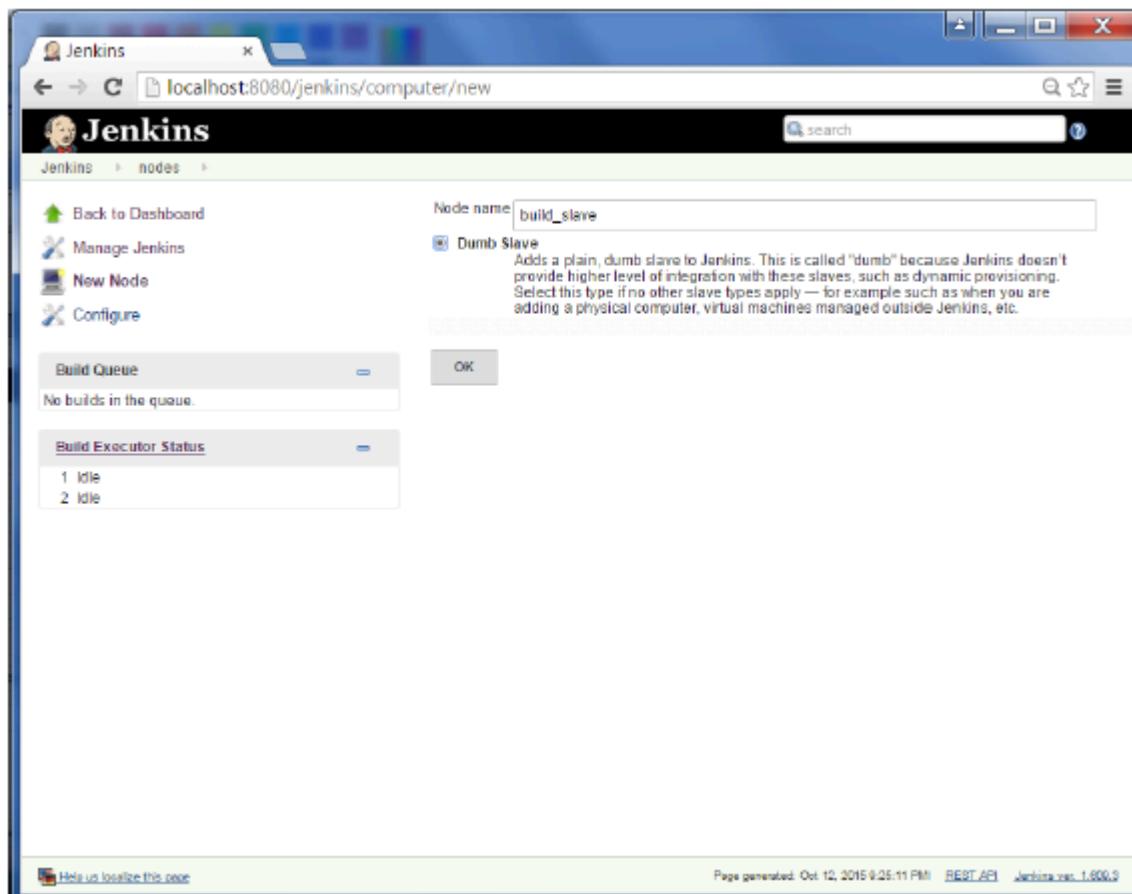
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Fre
	master	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	

Data obtained 3 min 11 sec 3 min 12 sec 3 min 12 sec 3 min 12 sec

[Refresh status](#)

At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:22:44 PM', 'REST API', and 'Jenkins ver. 1.800.2'.

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



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

The screenshot shows the Jenkins 'build_slave Configure' page. The left sidebar includes links for Back to List, Status, Delete Slave, Configure, Build History, Load Statistics, and Log. The main form fields are:

- Name:** build_slave
- Description:** (empty)
- # 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, Host (dbmm30), and Run service as (Use Local System User). There is also an "Advanced..." button.

Availability: Keep this slave on-line as much as possible

Node Properties: Environment variables, Test locations

Save button at the bottom.

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 [jenkins]' page. The left sidebar includes links for Back to Dashboard, Manage Jenkins, New Node, and Configure. The main area displays a table of nodes:

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

Buttons include Refresh status and Refresh AUTO REFRESH. The bottom footer includes links for Help us localize this page, Page generated: Oct 12, 2015 9:31:43 PM, REST API, Jenkins ver. 1.600.3, and navigation arrows.

Jenkins - Automated Deployment

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

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

Install	Name	Version
<input type="checkbox"/>	Artifact Deployer Plugin 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"/>	Capistrano 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 6.6. 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"

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the 'Post-build Actions' section, the 'Deploy war/ear to a container' option is selected. A dropdown menu is open, showing various actions like 'Publish FindBugs analysis results', 'Archive the artifacts', and 'Deploy war/ear to a container'. The 'Deploy war/ear to a container' option is highlighted. Other options include 'Publish JUnit test result report', 'Publish Javadoc', 'Publish Selenium Report', 'Record fingerprints of files to track usage', 'Git Publisher', 'E-mail Notification', and 'Set build status on GitHub commit'. Below the dropdown, there is an 'Add post-build action' button. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons.

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

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

- WAR/EAR files: target/spare-1.0.war
- Context path: spare
- Containers:
 - Tomcat 7.x
 - Manager user name: 8112233
 - Manager password: (redacted)
 - Tomcat URL: http://dxbmcm10:8080

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

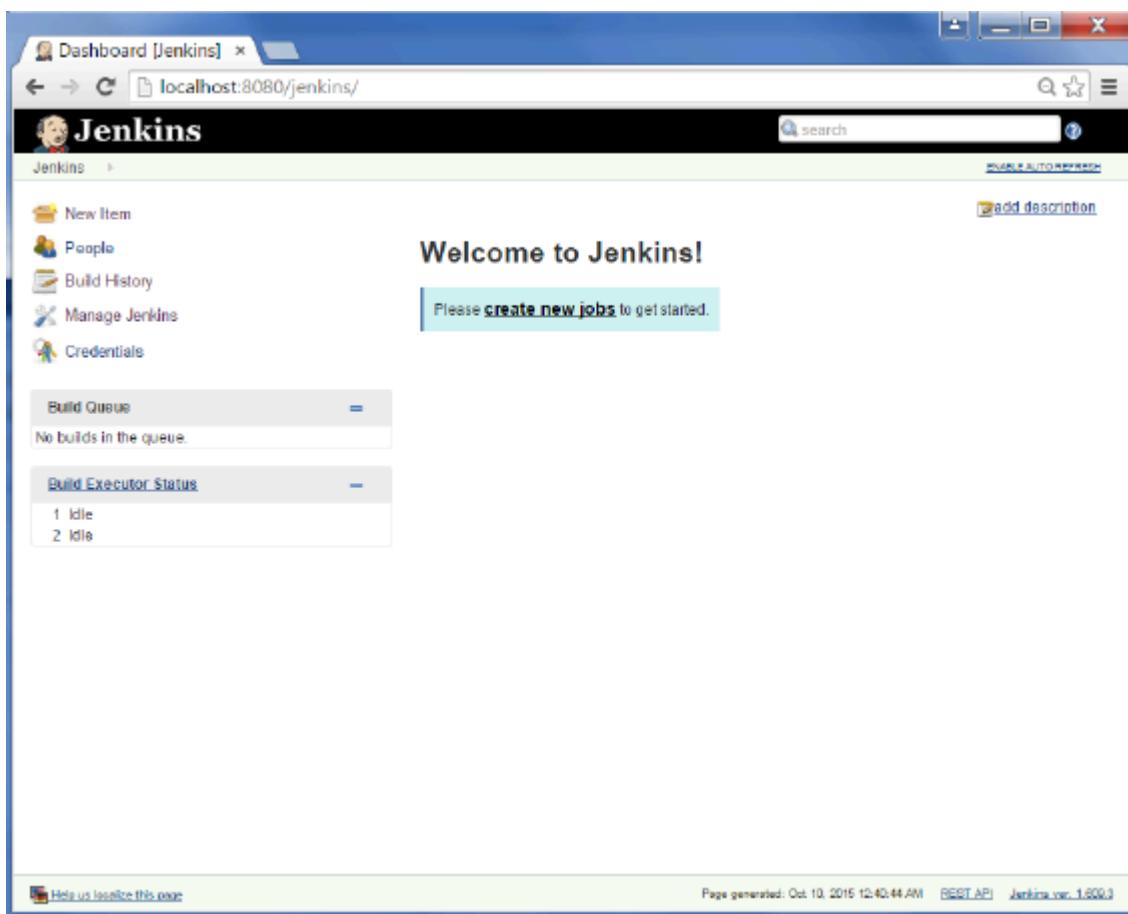
Jenkins - Metrics & Trends

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

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

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

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



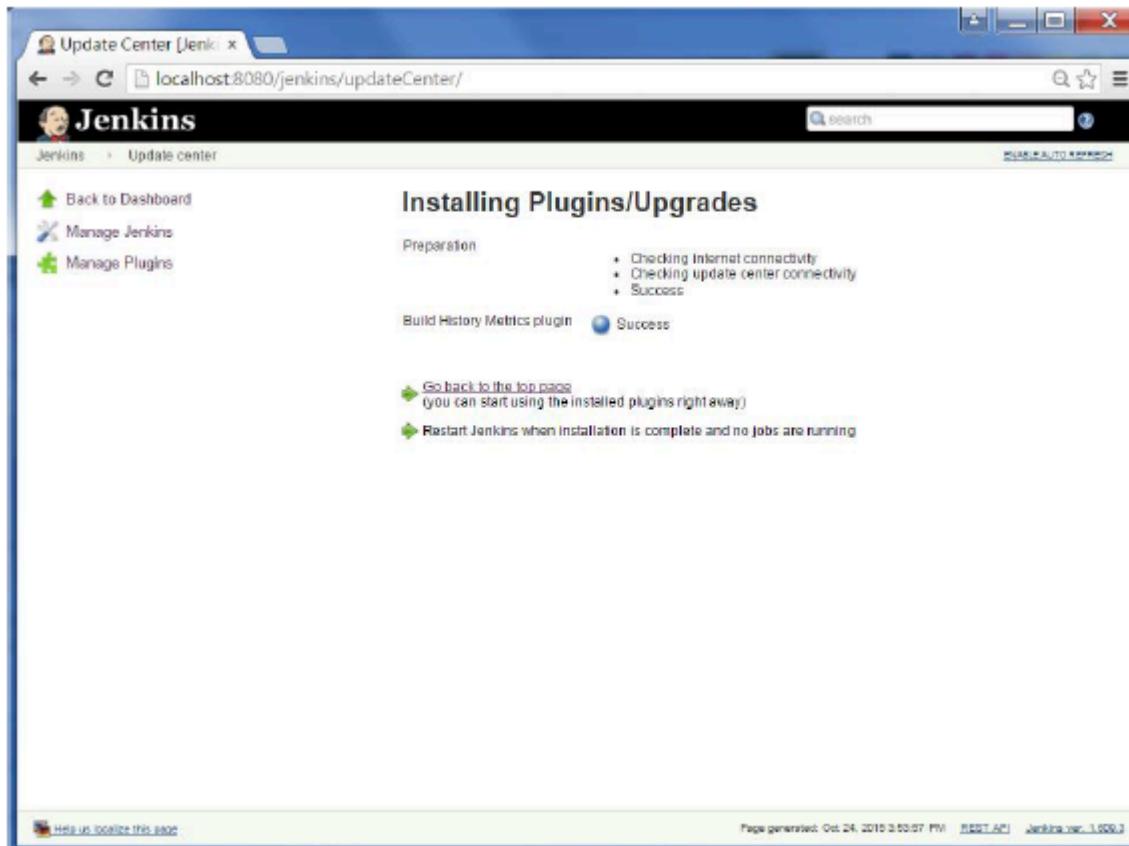
Step 2 – Go to the Manage Plugins option.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are two dropdown menus: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains several configuration links with icons: 'Configure System' (warning icon), 'Configure Global Security' (lock icon), 'Reload Configuration from Disk' (disk icon), 'Manage Plugins' (puzzle icon), 'System Information' (monitor icon), 'System Log' (document icon), 'Load Statistics' (heartbeat icon), 'Jenkins CLI' (terminal icon), 'Script Console' (script icon), 'Manage Nodes' (computer icon), 'Manage Credentials' (key icon), and 'About Jenkins' (info icon). A 'Setup Security' button and a 'Dismiss' button are located at the top right of this section.

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

The screenshot shows the Jenkins Plugin Manager on the 'Available' tab. It lists the 'Build History Metrics plugin' by 'jenkinsci-build-history-metrics'. The plugin details are: Name: Build History Metrics plugin, Version: 1.2, Description: Provides build metrics that encompass the history of all the runs. There are two buttons at the bottom: 'Install without restart' (selected) and 'Download now and install after restart'. A status message says 'Update information obtained: 2 mi'.

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



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

The screenshot shows the Jenkins Project Helloworld dashboard. 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 'ENABLE AUTO REFRESH' button. A 'Project Helloworld' title is centered above a 'Workspace' icon and a 'Recent Changes' link. To the right, there are 'add description' and 'Disable Project' buttons. A large table displays build statistics: MTTR (Mean Time To Recovery) with rows for Last 7 Days (0 ms), Last 30 Days (23 hr), and All Time (23 hr); MTTF (Mean Time To Failure) with rows for Last 7 Days (0 ms), Last 30 Days (2 days 4 hr), and All Time (2 days 4 hr); and Standard Deviation with rows for Last 7 Days (0 ms), Last 30 Days (52 sec), and All Time (52 sec). Below the table is a 'Permalinks' section with a bulleted list of links to various builds. At the bottom, there are links for 'RSS for all' and 'RSS for failures', along with a 'Help us localize this page' link and footer text indicating the page was generated on Oct 24, 2015 at 9:57:10 PM.

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

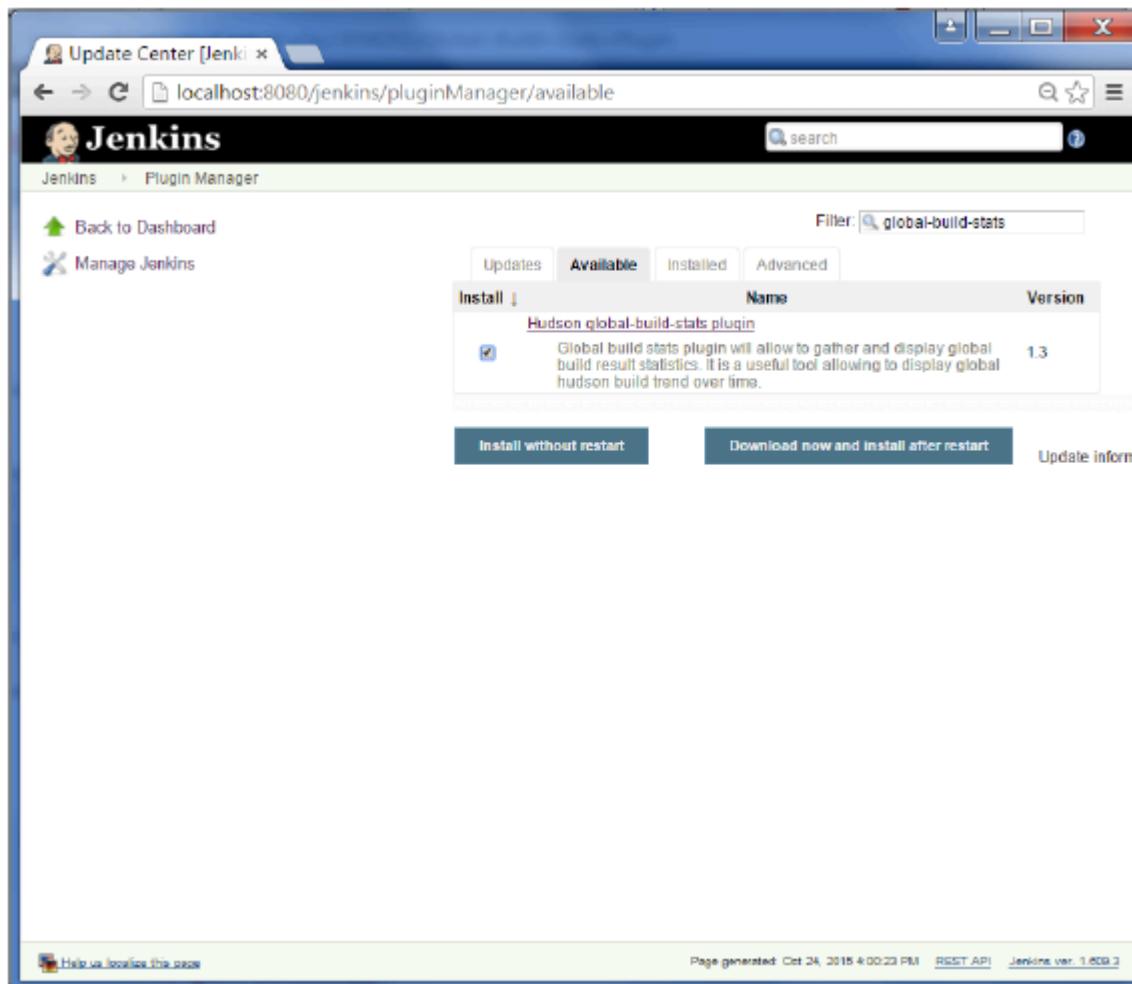
Step 1 – Go to the Jenkins dashboard and click on Manage Jenkins

The screenshot shows the Jenkins Dashboard at localhost:8080/jenkins/. The main header says "Jenkins". On the left, there's a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". Below the sidebar are two collapsed sections: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). The central area has a "Welcome to Jenkins!" message with a button to "create new jobs". A search bar is at the top right. At the bottom, there are links for "Help us localize this page", "Page generated: Oct 10, 2015 12:40:44 AM", "REST API", and "Jenkins ver. 1.600".

Step 2 – Go to the Manage Plugins option

The screenshot shows the "Manage Jenkins" page at localhost:8080/jenkins/manage. The main header says "Jenkins". The sidebar is identical to the dashboard. The central area has a "Manage Jenkins" title. It displays several configuration options with icons: "Configure System" (wrench), "Configure Global Security" (padlock), "Reload Configuration from Disk" (refresh), "Manage Plugins" (puzzle piece), "System Information" (monitor), "System Log" (document), "Load Statistics" (graph), "Jenkins CLI" (terminal), "Script Console" (script), "Manage Nodes" (computer), "Manage Credentials" (key), and "About Jenkins" (info). A warning message at the top right says: "Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat i18n](#) for more details." Buttons for "Setup Security" and "Dismiss" are also present.

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



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

The screenshot shows a web browser window for the Jenkins Update Center. The URL is `localhost:8080/jenkins/updateCenter/`. The main title is "Installing Plugins/Upgrades". On the left sidebar, there are links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area has a section titled "Preparation" with a bulleted list: "Checking internal connectivity", "Checking update center connectivity", and "Success". Below this, it says "Hudson global-build-stats plugin" followed by a blue circular icon with a white checkmark and the word "Success". At the bottom, there are two green icons with arrows pointing right: "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". A small note at the bottom left says "Help us localize this page". The bottom right corner shows the page was generated on Oct 24, 2015 at 4:01:04 PM, and the Jenkins version is 1.809.3.

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

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

The screenshot shows the Jenkins 'Manage Jenkins' interface at the URL localhost:8080/jenkins/manage. The page lists several management functions with corresponding icons:

- Manage Plugins**: Adds, removes, disables or enables 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**: Checks your resource utilization and sees 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**: Adds, removes, controls and monitors the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Creates/deletes/modifies 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.
- Manage Old Data**: Scrubs configuration files to remove remnants from old plugins and earlier versions.
- Global Build Stats**: Displays stats about daily build results.
- In-process Script Approval**: Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Prepare for Shutdown**: Stops executing new builds, so that the system can be eventually shut down safely.

At the bottom left is a link to "Help us localize this page". At the bottom right, it says "Page generated: Oct 24, 2010 4:02:40 PM REST API Jenkins ver. 1.600.3".

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 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 content area is titled 'Global Build Stats' and features a 'Statistics' section with a note: 'No chart configured for the moment ... [Create a new chart configuration](#)'. Below this is a 'Build Results retention strategies' section with three checkboxes: 'Automatically discard results older than 365 days', 'Do not keep build results when they are discarded', and 'Keep existing job results only'. An 'Update retention strategies' button is located below these checkboxes. Further down is a 'Data initialization' section with a note: 'Click button below to initialize build statistics. Job results read will be merged with already recorded job results.' and a large 'Initialize stats' button. At the bottom left is a link to 'Help us localize this page!' and at the bottom right are links for 'Page generated: Oct 24, 2015 4:09:17 PM', 'REST API', and '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 page at localhost:8080/jenkins/plugin/global-build-stats/#Initialize. The left sidebar has links for Back to Dashboard, Create new chart, Manage retention strategies, and Data Initialization. The main content area is titled "Global Build Stats" with a subtitle "Statistics". It says "No chart configured for the moment ... [Create a new chart configuration](#)". Below that is a section for "Build Results retention strategies" with three checkboxes: "Automatically discard results older than [365] days", "Do not keep build results when they are discarded", and "Keep existing job results only". A "Update retention strategies" button is next to the checkboxes. At the bottom is a "Data Initialization" section with a note: "Click button below to initialize build statistics. Job results read will be merged with already recorded job results." A "Data successfully initialized!" message is displayed above a "Initialize stats" button. The bottom of the page includes links for "Help us localize this page", "Page generated: Oct 24, 2015 4:03:17 PM", "REST API", and "Jenkins ver. 1.603.3".

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

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

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

The screenshot shows the Jenkins Global Build Stats configuration interface. At the top, there's a navigation bar with links like 'Back to Dashboard', 'Create new chart', 'Manage retention strategies', and 'Data Initialization'. On the right, there's a search bar and a 'DISABLE AUTO REFRESH' button. The main title is 'Global Build Stats' with a small icon of a person holding a chart. Below it, a message says 'Statistics' and 'No chart configured for the moment ... Create a new chart configuration'. A large central window is titled 'Adding new chart' and contains the following fields:

- Title:** Demo
- Chart Width * Height:** 800 * 600
- Chart time scale:** Daily
- Chart time length:** 30 days

Filters:

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

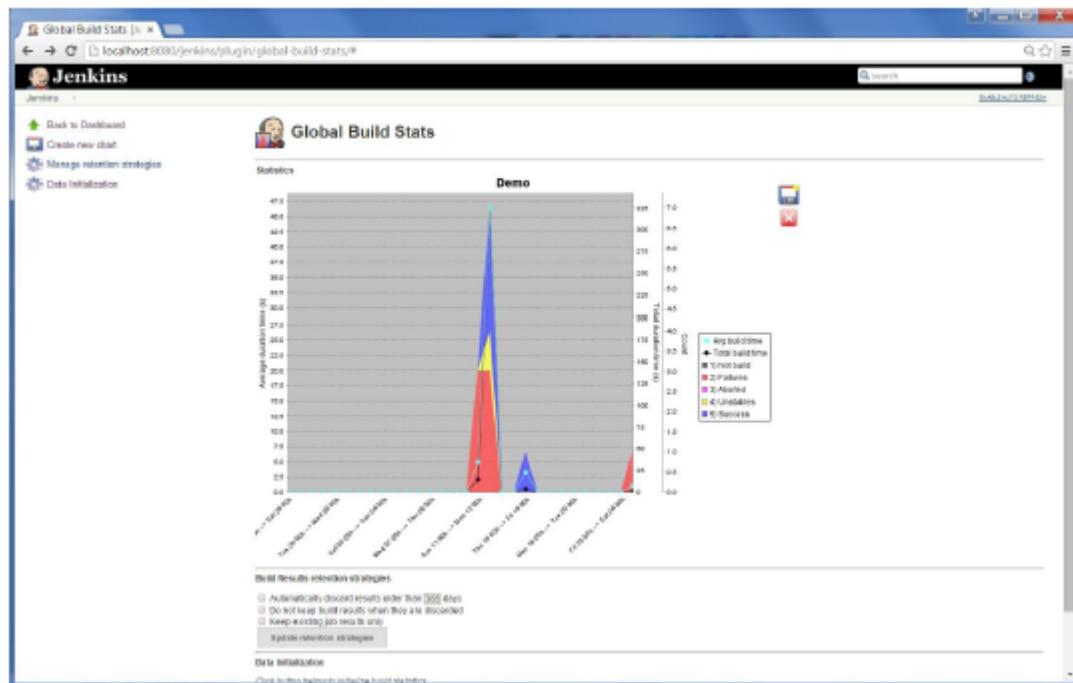
Elements displayed on chart:

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

At the bottom of this window are buttons for 'Overview', 'Create new chart', and 'Cancel'.

At the very bottom of the page, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 24, 2016 4:03:17 PM', 'REST API', and 'Jenkins ver. 1.602.3'.

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



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

The screenshot shows the Jenkins Global Build Search interface. At the top, there are filter options: 'Job Name' (set to 'Hellmund'), 'Status' (set to 'Failure'), and a search bar. Below these are sections for 'Search results' and 'Recent builds'. The 'Search results' section displays a table with three rows of build data:

Status	Job name	Date	Duration	Mode name	Launcher ID	
Failure	Hellmund	[Build #13]	Oct 11, 2015 11:04:57 PM	5.6 sec	master	SYSTEM
Failure	Hellmund	[Build #12]	Oct 11, 2015 19:51:37 PM	4.6 sec	master	SYSTEM
Failure	Hellmund	[Build #11]	Oct 11, 2015 19:48:11 PM	0.2 sec	master	SYSTEM

Jenkins - Server Maintenance

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

URL Options

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

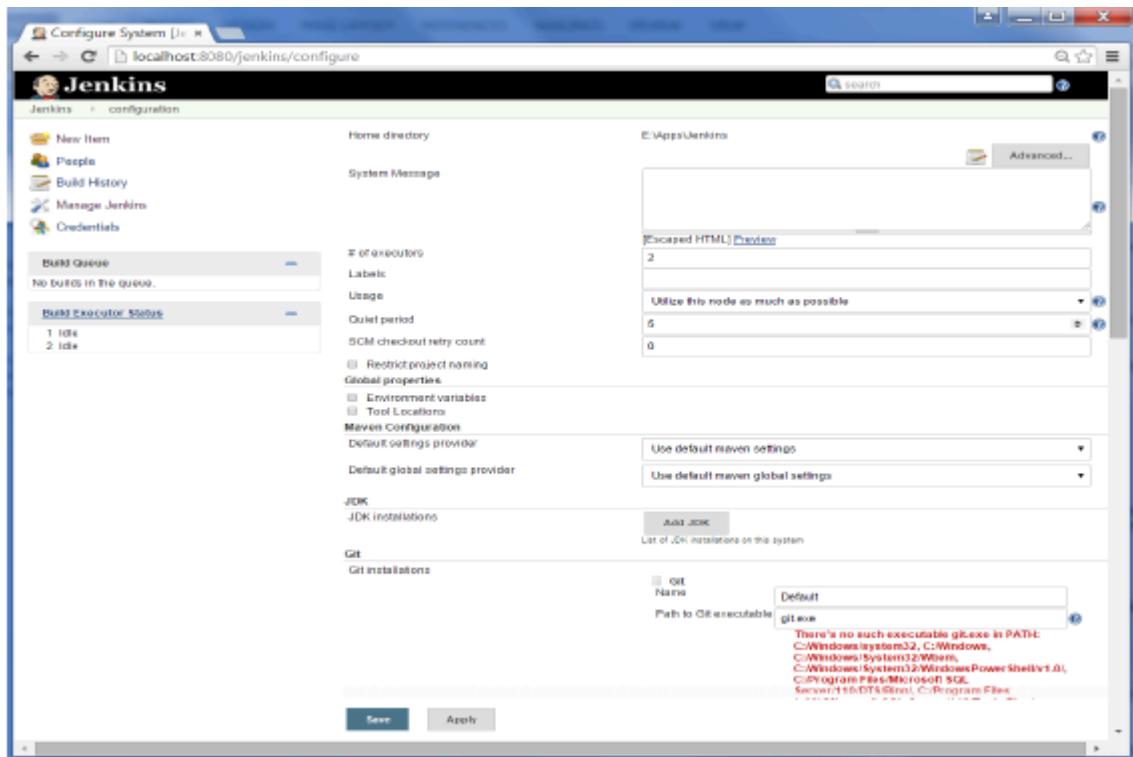
<http://localhost:8080/jenkins/exit> – shutdown jenkins

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

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

Backup Jenkins Home

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

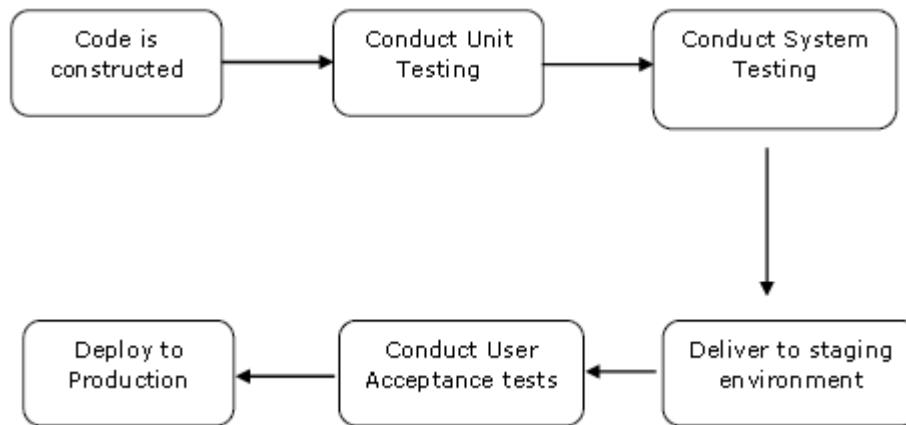


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

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

Jenkins - Continuous Deployment

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



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

The screenshot shows the Jenkins configuration interface for a job named 'Helloworld'. In the main pane, there's a section for 'Invoke Ant' with 'Ant Version' set to 'NewHome' and 'Targets' dropdown. Below it, a context menu is open over the 'Deploy war/ear to a container' action, listing various publishing and reporting options. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons.

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.

The screenshot shows the 'New Item' dialog in Jenkins. The 'Item name' field is filled with 'QA'. The 'Freestyle project' radio button is selected. To the right, there are detailed descriptions for other project types: 'Maven project', 'External Job', 'Multi-configuration project', and 'Copy existing item'. At the bottom left, the 'Build Executor Status' section shows '1 Idle' and '2 Idle' executors. At the bottom right, there is an 'OK' button.

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

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

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

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

The screenshot shows the Jenkins dashboard for the 'QA' project. On the left sidebar, there are links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The 'Configure' link is currently selected. In the center, there is a 'Project QA' summary with sections for Workspace and Recent Changes. Below this is a table showing performance metrics:

	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

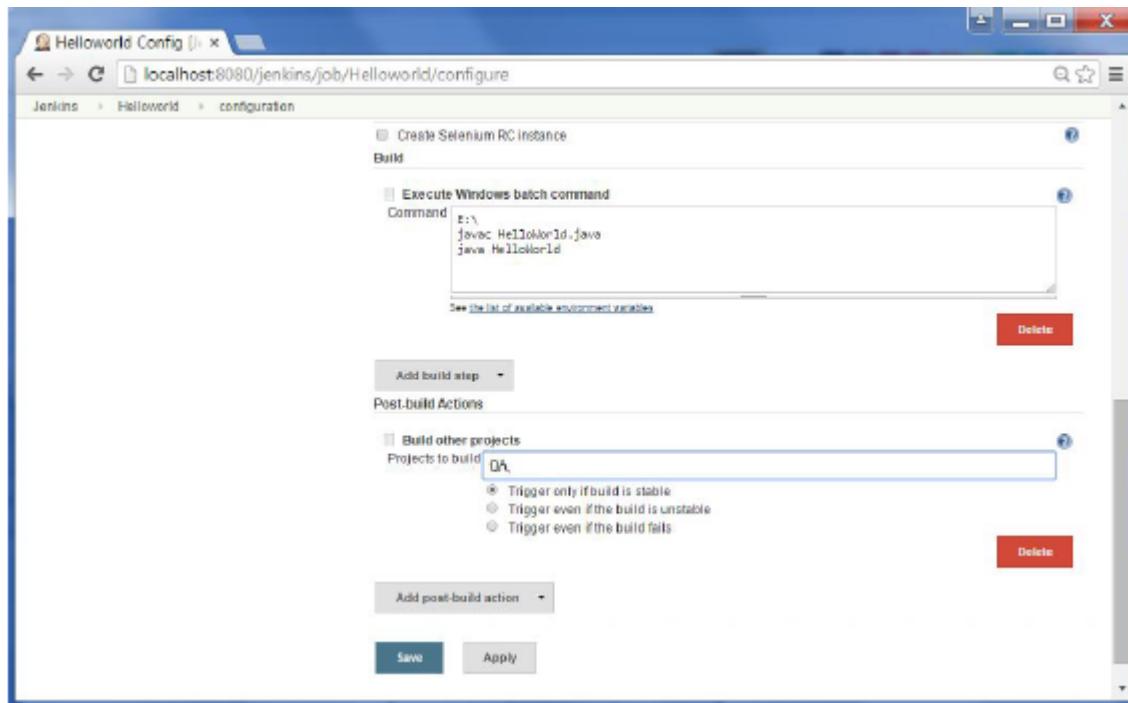
Step 3 – Now go to your Helloworld project and click on the Configure option

The screenshot shows the Jenkins dashboard 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' (one idle). The main area displays a table of jobs. The 'Helloworld' job is selected, showing its status as '12 days - #15' with a 'Changes' link. A context menu is open over the 'Configure' button, with options like 'Changes', 'Workspace', 'Build Now', and 'Delete Project'. At the bottom of the page, there are links for 'RSS for failures' and 'RSS for just latest builds'.

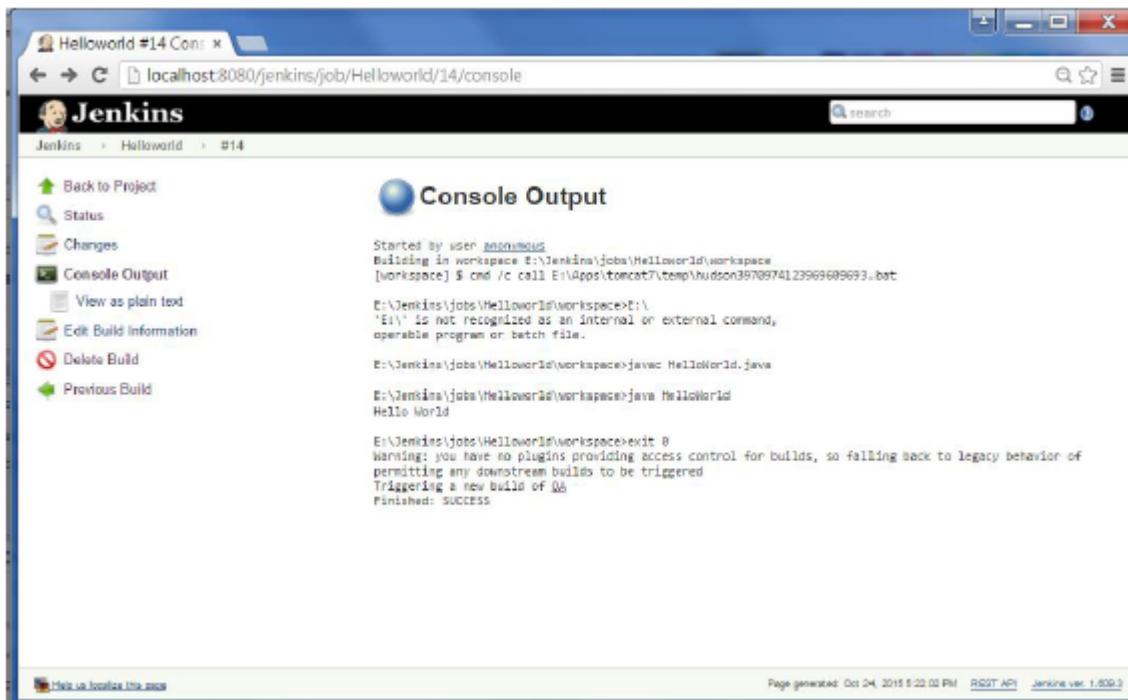
Step 4 – In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

The screenshot shows the 'Helloworld' project configuration page at localhost:8080/jenkins/job/Helloworld/configure. The 'Post-build Actions' section is expanded, showing various options like 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects' (which is highlighted with a blue selection bar), 'Publish JUnit test result report', 'Publish Javadoc', 'Record fingerprints of files to track usage', 'Git Publisher', and 'E-mail Notification'. A 'Delete' button is visible next to the 'Build other projects' item. At the bottom, there are 'Save' and 'Apply' buttons.

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 Plugins. In the available tab, search for 'Delivery Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Update Center interface. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main content area is titled "Install" and lists several Jenkins plugins:

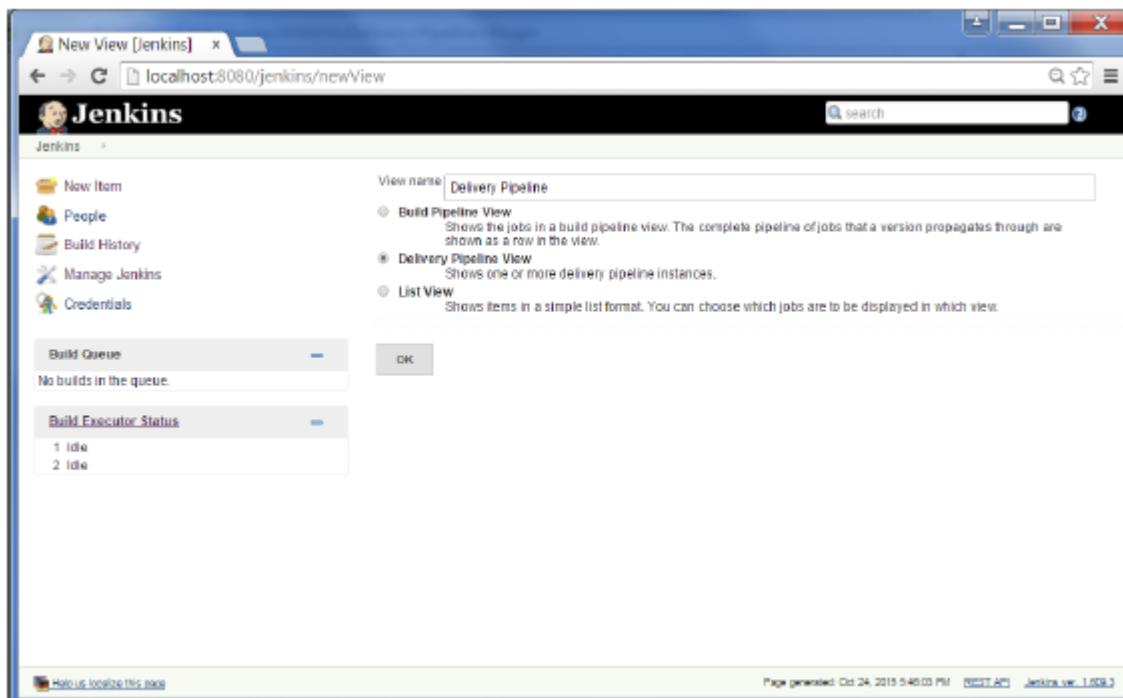
	Name	Version
<input type="radio"/>	Ontrack Jenkins plugin	2.15.0
<input type="radio"/>	Fail The Build Plugin	1.0
<input type="radio"/>	Binscope plugin	1.44
<input type="radio"/>	Build Graph View Plugin	1.1.1
<input type="radio"/>	Deployment Status	0.1.106
<input type="radio"/>	CloudBees Docker Hub Notification	1.0.2
<input type="radio"/>	Seed Jenkins plugin	0.17.0
<input checked="" type="radio"/>	Delivery Pipeline Plugin	0.9.7

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

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

The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [Jenkins]". The address bar shows "localhost:8080/jenkins/". The left sidebar has links: "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". The main content area has tabs: "All" (highlighted in blue), "Build Queue" (grey), and "Build Executor Status" (grey). Below the tabs is a table showing build status for two projects: "HelloWorld" and "QA". At the bottom, there is a legend for build status colors: R88 for all, R88 for failures, and R88 for just latest builds.

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 [Jenkins]' configuration page for 'Delivery Pipeline'. The left sidebar includes links for New Item, People, Build History, Edit View, Delete View, View Fullscreen, Manage Jenkins, and Credentials. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', there are 1 Idle and 2 Idle executors. The main configuration area has the following 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
 - Enable start of new pipeline build: checked
 - Enable manual triggers: checked
 - Enable rebuild: checked
 - Show avatars: checked
 - Show commit messages: checked
 - Show job description: checked
 - Show job promotions: checked
 - Show JUnit results: checked
 - Show static analysis results: checked
 - Show total build time: checked
 - URL for custom CSS file (fullscreen): (empty)
- Pipelines:**
 - Components:**

Name: FirstJob	Delete
Initial Job: HelloWorld	▼
Final Job (optional):	▼

Add

Add

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

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

The screenshot shows the Jenkins interface for a 'Delivery Pipeline' job named 'Firstjob'. The pipeline consists of three stages: 'HelloWorld', 'QA', and another 'QA' stage. Stage 1, 'HelloWorld', is triggered by an anonymous user and completed 29 minutes ago. Stage 2, 'QA', is triggered by an anonymous user and completed 28 minutes ago. Stage 3, another 'QA' stage, is triggered by an anonymous user and completed an hour ago. The total build time for the entire pipeline is 2 seconds. The Jenkins sidebar on the left includes links for New Item, People, Build History, Edit View, Delete View, View Fullscreen, Manage Jenkins, and Credentials. It also displays the Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle).

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

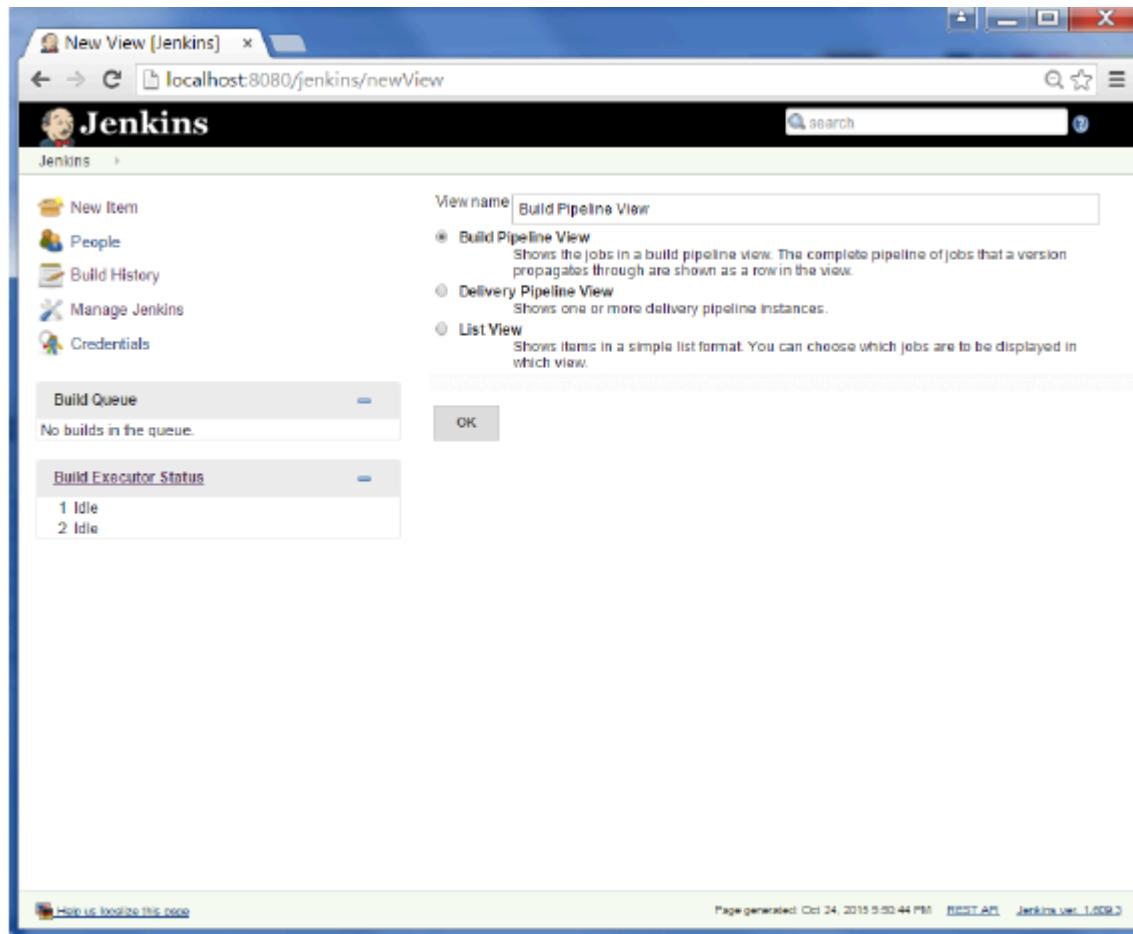
Step 1 – Go to Manage Jenkins → Manage Plugin's. In the available tab, search for 'Build Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

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 header is "Jenkins" with a "Plugin Manager" link. Below it are links to "Back to Dashboard" and "Manage Jenkins". A search bar has "Build pipeline" typed into it. There are four tabs at the top: "Updates", "Available" (which is selected), "Installed", and "Advanced". A "Filter" input field also contains "Build pipeline". The main content area is titled "Install" and lists several plugins. One plugin is highlighted: "Build Pipeline Plugin" (version 1.4.8). Its description says: "This plugin provides a _Build Pipeline View_ of upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins." Other listed plugins include "Fail The Build Plugin", "Runscope plugin", "Build Graph View Plugin", and "Delivery Pipeline Plugin". At the bottom are three buttons: "Install without restart", "Download now and install after restart" (which is highlighted in blue), and "Update info".

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

The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [Jenkins]". The address bar shows "localhost:8080/jenkins/". The main header is "Jenkins". On the left, there's a sidebar with links: "New item", "People", "Build History", "Manage Jenkins", and "Credentials". Below that are sections for "Build Queue" (empty) and "Build Executor Status" (1 Idle, 2 Idle). The main content area has a table titled "All" showing two build items: "HelloWorld" and "QA". The table columns are: S, W, Name, Last Success, Last Failure, and Last Duration. The "HelloWorld" row shows "25 min - #14" for Last Success, "1 hr 49 min - #12" for Last Failure, and "1.4 sec" for Last Duration. The "QA" row shows "25 min - #5" for Last Success, "28 min - #2" for Last Failure, and "1.4 sec" for Last Duration. There are "add description" and "Icon: SML" buttons above the table. A legend at the bottom right indicates: RSS for all, RSS for failures, and RSS for just latest builds. The footer includes a "Help us localize this page" link and "Page generated: Oct 24, 2015 4:48:09 PM REST API Jenkins ver. 1.809.3".

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



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

The screenshot shows the Jenkins 'Edit View [Jenkins]' configuration page for 'Build Pipeline View'. The left sidebar lists various Jenkins management options. The main configuration area includes fields for 'Name' (Build Pipeline View), 'Description', 'Filter build queue', 'Filter build executors', 'Build Pipeline View Title', 'Layout' (set to 'Based on upstream/downstream relations'), 'Select Initial Job' (HelloWorld), and 'Refresh frequency (in seconds)' (set to 3). Buttons for 'OK' and 'Apply' are at the bottom.

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

The screenshot shows the Jenkins 'Build Pipeline View' page. It displays the 'Build Pipeline' interface with three stages: 'Pipeline #14' (grey bar), '#14 HelloWorld' (green bar), and '#5 QA' (green bar). Each stage has a small Jenkins icon and a 'Run' button above it. The Jenkins logo is visible in the top left corner of the browser window.

Jenkins - Managing Plugins

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

The screenshot shows the Jenkins Plugins page on a web browser. The URL is <https://wiki.jenkins-ci.org/display/JENKINS/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 "Plugins" and contains a table of contents for plugin installation and usage. The table of contents includes:

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

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

Uninstalling Plugins

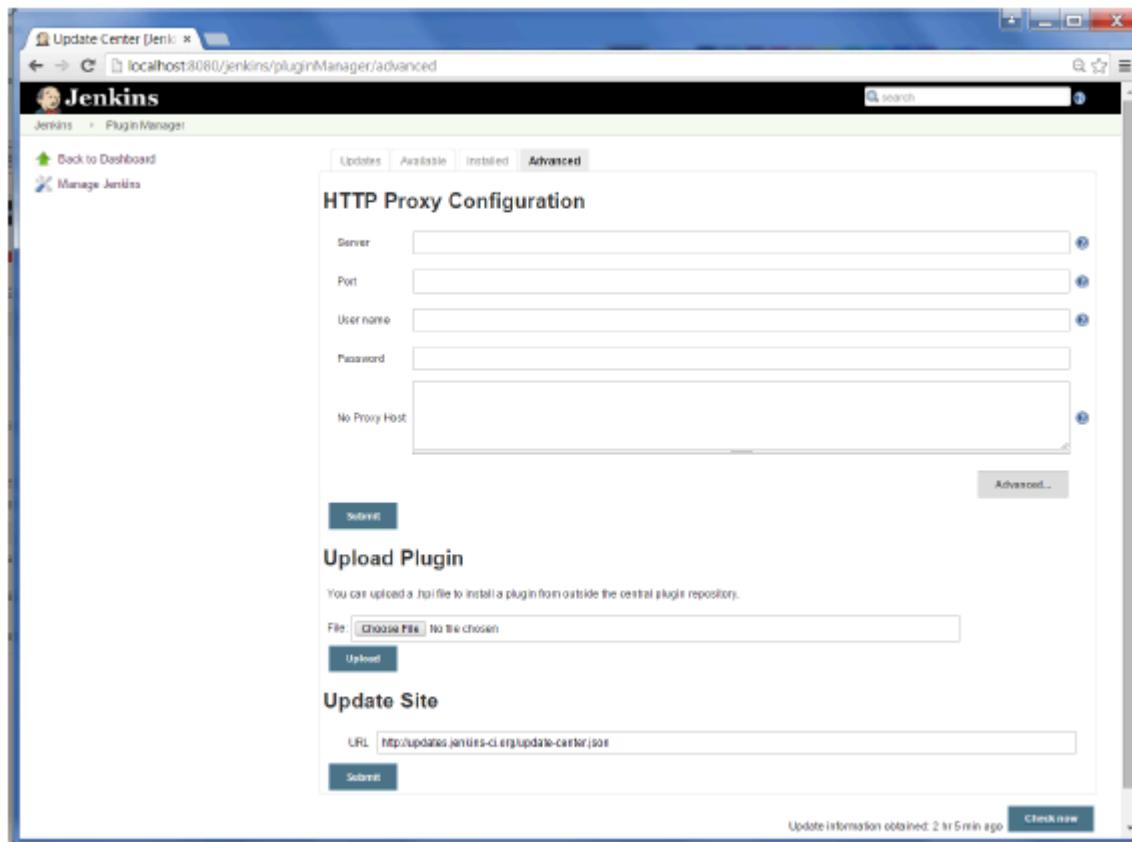
To uninstall a plugin, Go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [jenk]" and the URL is localhost:8080/jenkins/pluginManager/installed. The left sidebar has "Back to Dashboard" and "Manage Jenkins". The main area has tabs for "Updates", "Available", "Installed" (which is selected), and "Advanced". A search bar is at the top right. The "Installed" tab displays a list of installed plugins with their names, versions, previous versions, pinned status, and uninstall buttons. The plugins listed are:

Enabled	Name	Version	Previously installed version	Pinned	Uninstall
<input checked="" type="checkbox"/>	Ant Plugin	1.2			<button>Uninstall</button>
<input checked="" type="checkbox"/>	Built History Metrics Plugin	1.2			<button>Uninstall</button>
<input checked="" type="checkbox"/>	Built Pipeline Plugin	1.4.8			<button>Uninstall</button>
<input checked="" type="checkbox"/>	Credentials Plugin	1.23	Downgrade to 1.18	<input checked="" type="checkbox"/>	<button>Unpin</button>
<input checked="" type="checkbox"/>	CVS Plugin	2.15			<button>Uninstall</button>
<input checked="" type="checkbox"/>	Delivery Pipeline Plugin	0.8.1			<button>Uninstall</button>
<input checked="" type="checkbox"/>	Deploy to container Plugin	1.10			<button>Uninstall</button>
<input checked="" type="checkbox"/>	External Monitor Job Type Plugin	1.4			<button>Uninstall</button>
<input checked="" type="checkbox"/>	External Notification Plugin	1.1			<button>Uninstall</button>
<input checked="" type="checkbox"/>	FindBugs Plugin	4.60			<button>Uninstall</button>

Installing another Version of a Plugin

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

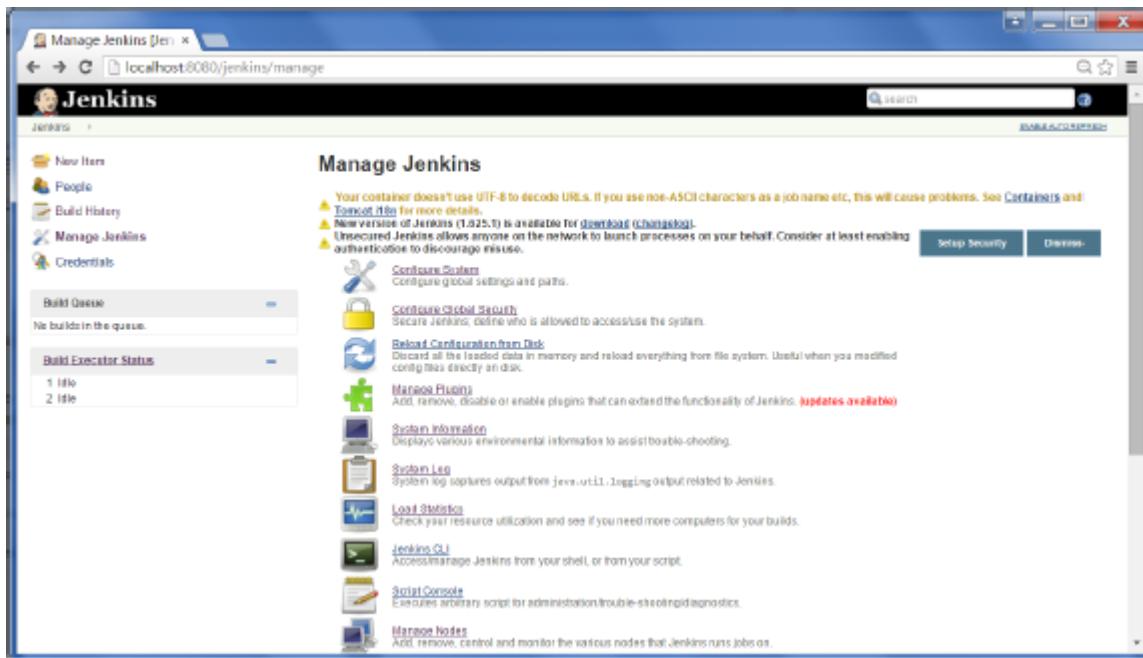


Jenkins - Security

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

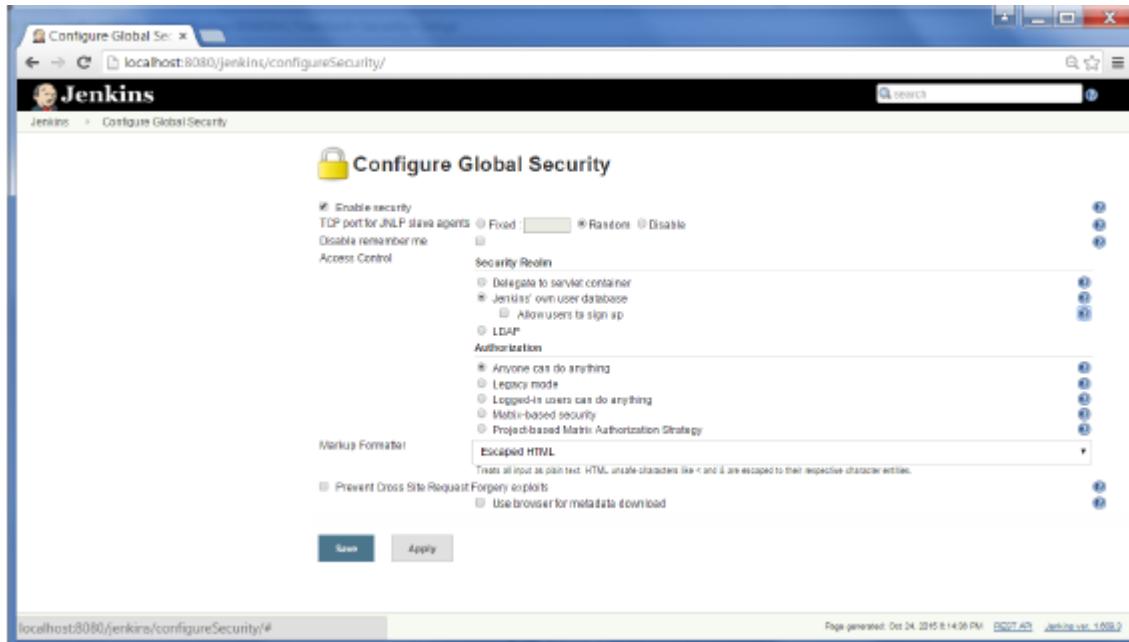
To configure Security in Jenkins, follow the steps given below.

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



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

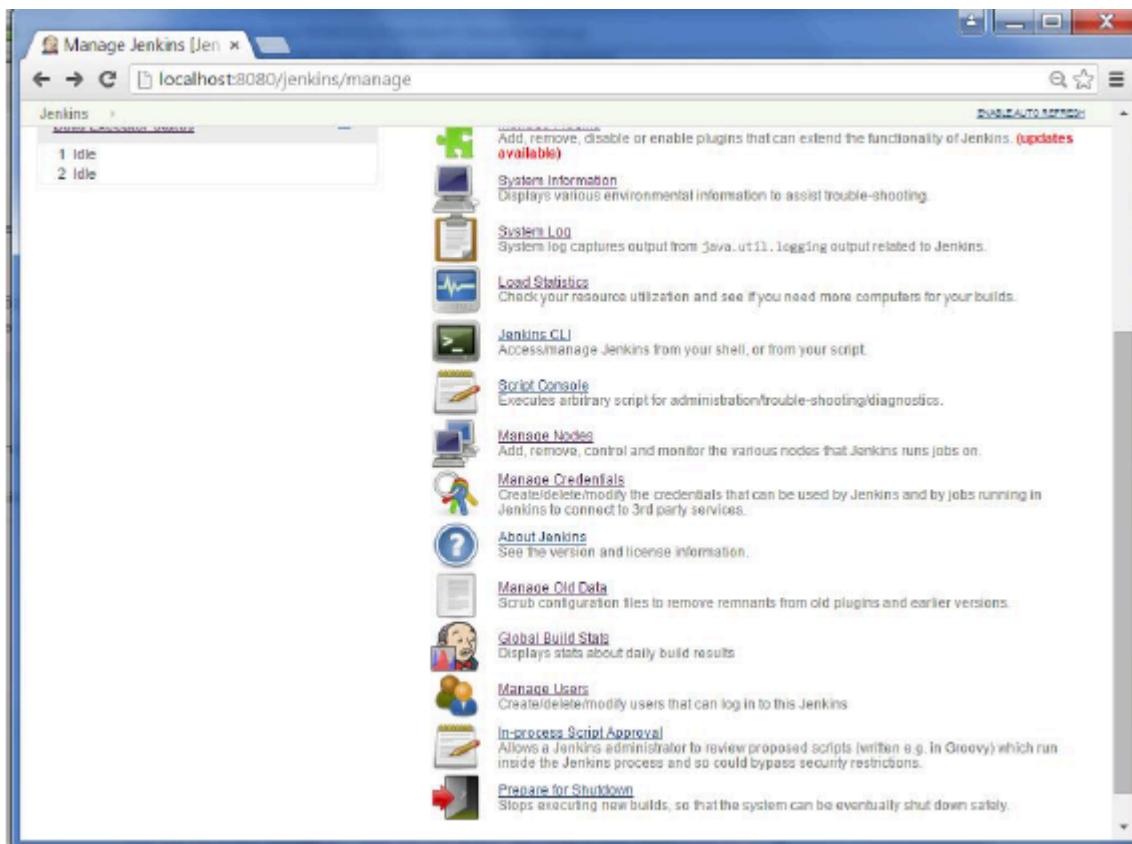
By default you would want a central administrator to define users in the system, hence ensure the 'Allow users to sign up' option is unselected. You can leave the rest as it is for now and click the Save button.



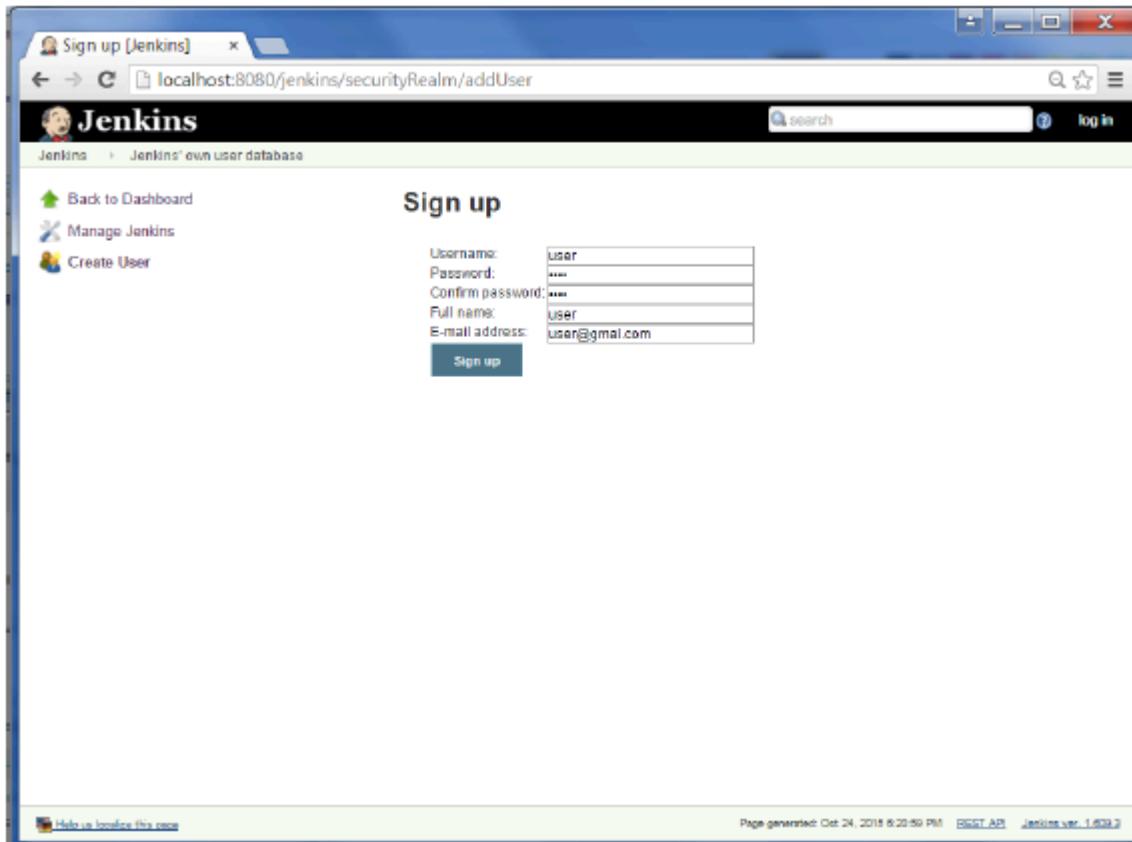
Step 3 – You will be prompted to add your first user. As an example, we are setting up an admin users for the system.

The screenshot shows a web browser window with the Jenkins 'Sign up' page. The URL in the address bar is `localhost:8080/jenkins/securityRealm/firstUser`. The page has a dark header with the Jenkins logo and a search bar. On the left, there's a sidebar with links: 'Back to Dashboard', 'Manage Jenkins', and 'Create User'. The main content area is titled 'Sign up' and contains four input fields: 'Username' (filled with 'admin'), 'Password' (filled with '****'), 'Confirm password' (filled with '****'), and 'E-mail address' (filled with 'al@gmail.com'). Below these fields 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 6:16:01 PM', 'REST API', and 'Jenkins ver. 1.608.3'.

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

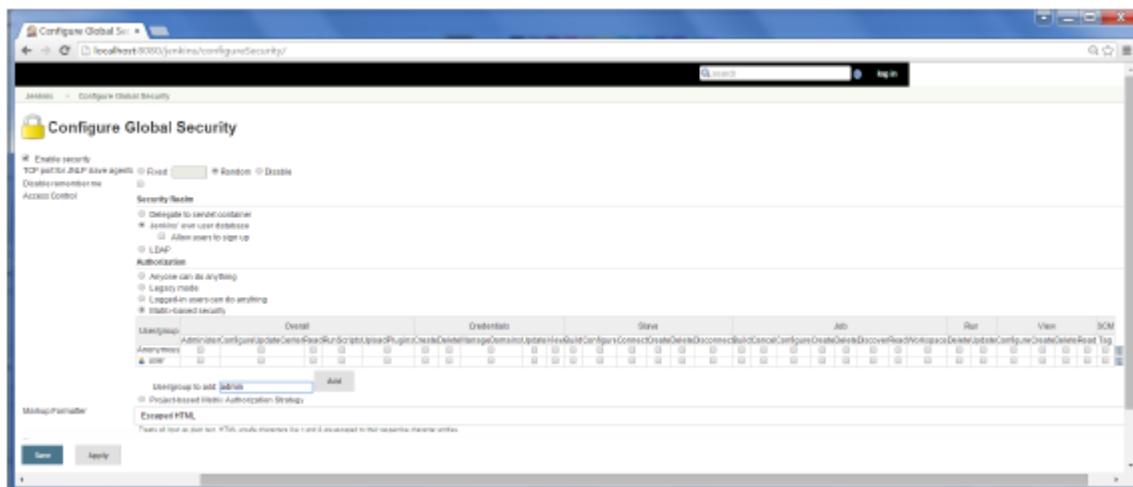


Step 5 – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.



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

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



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

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

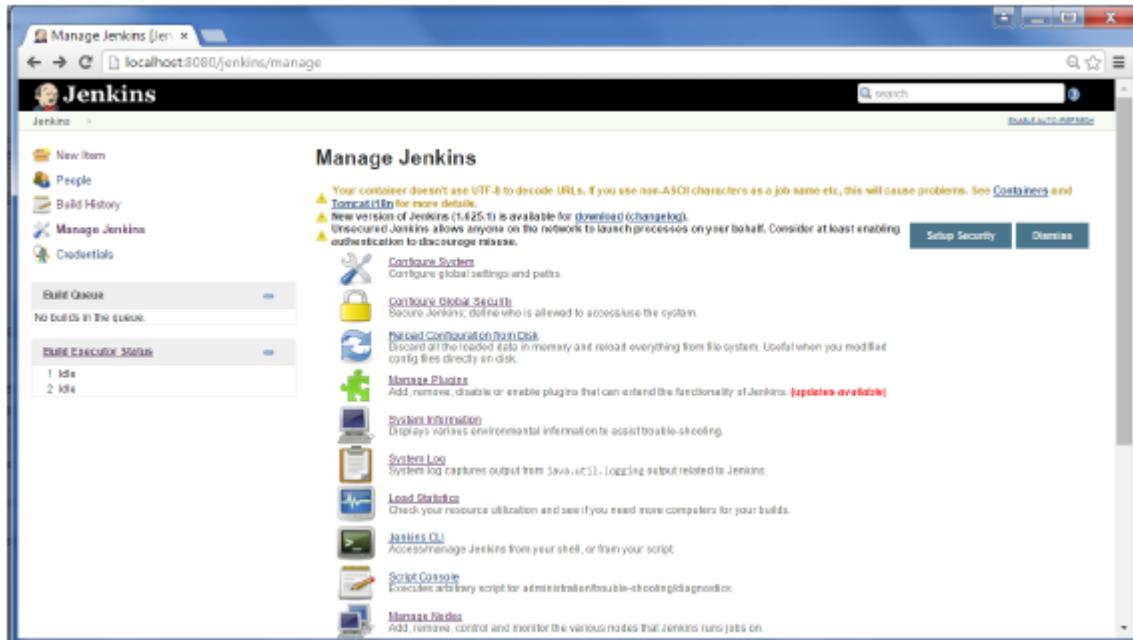
Your Jenkins security is now setup.

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

Jenkins - Backup Plugin

Jenkins has a backup plugin which can 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

Update Center [Jenkins] < localhost:8080/jenkins/pluginManager/available

Jenkins > Plugin Manager

Back to Dashboard | Manage Jenkins | Manage Plugins

Updates Available Installed Advanced

Filter: backup

Install	Name	Version
<input checked="" type="checkbox"/>	Backup plugin	1.6.1
<input type="checkbox"/>	Backup and interrupt job plugin	1.0
<input type="checkbox"/>	CloudBees Jenkins Enterprise	15.05.1
CloudBees Free Enterprise Plugins		
<input type="checkbox"/>	This plugin installs free enterprise plugins from CloudBees. The following plugins are automatically installed: "Folders," easily organize your jobs "Backup to Cloud," backup your Jenkins into CloudBees cloud "Wasted Minutes," find out if you are short of slaves and need to add capacity quickly "CloudBees Status," find out how much of the free CloudBees Jenkins capacity in the cloud is available for your use "Note." You will be asked to register for a free CloudBees account to use these plugins (This plugin was formerly known as the CloudBees Plugin Gateway plugin)	5.0
<input type="checkbox"/>	Periodic Backup	1.3
<input type="checkbox"/>	ThinBackup	1.7.4

Install without restart | Download now and install after restart | Update information of

Update Center [Jenkins] < localhost:8080/jenkins/updateCenter/

Jenkins > Update center

Back to Dashboard | Manage Jenkins | Manage Plugins

Installing Plugins/Upgrades

Preparation

- Checking Internet connectivity
- Checking update center connectivity
- Success

Backup plugin Success

Go back to the top page
(you can start using the installed plugins right away)

Restart Jenkins when installation is complete and no jobs are running

Help us localize this page | Page generated: Oct 24, 2015 8:26:30 PM | REST API | Jenkins ver. 1.603

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 interface at the URL localhost:8080/jenkins/manage. The page lists several management functions:

- 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 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.
- Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.
- Global Build Stats**: Displays stats about daily build results.
- Manage Users**: Create/delete/modify users that can log in to this Jenkins.
- In-process Script Approval**: Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Backup manager**: Backup or Restore Jenkins configuration files.
- Prepare for Shutdown**: Stops executing new builds, so that the system can eventually shut down safely.

At the bottom, there are links for "Help us localize this page", "Page generated: Oct 24, 2015 7:10:31 PM", "REST API", and "Jenkins ver. 1.600.3".

Step 4 – Click on Setup.

The screenshot shows the Jenkins Backup manager interface at the URL localhost:8080/jenkins/backup/. The page has a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. It also displays the Build Queue and Build Executor Status. The main content area is titled "Backup manager" and contains three options:

- Setup**: Represented by a wrench icon.
- Backup Hudson configuration**: Represented by a blue arrow icon.
- Restore Hudson configuration**: Represented by a red arrow icon.

At the bottom, there are links for "Help us localize this page", "Page generated: Oct 24, 2015 7:11:02 PM", "REST API", and "Jenkins ver. 1.600.3".

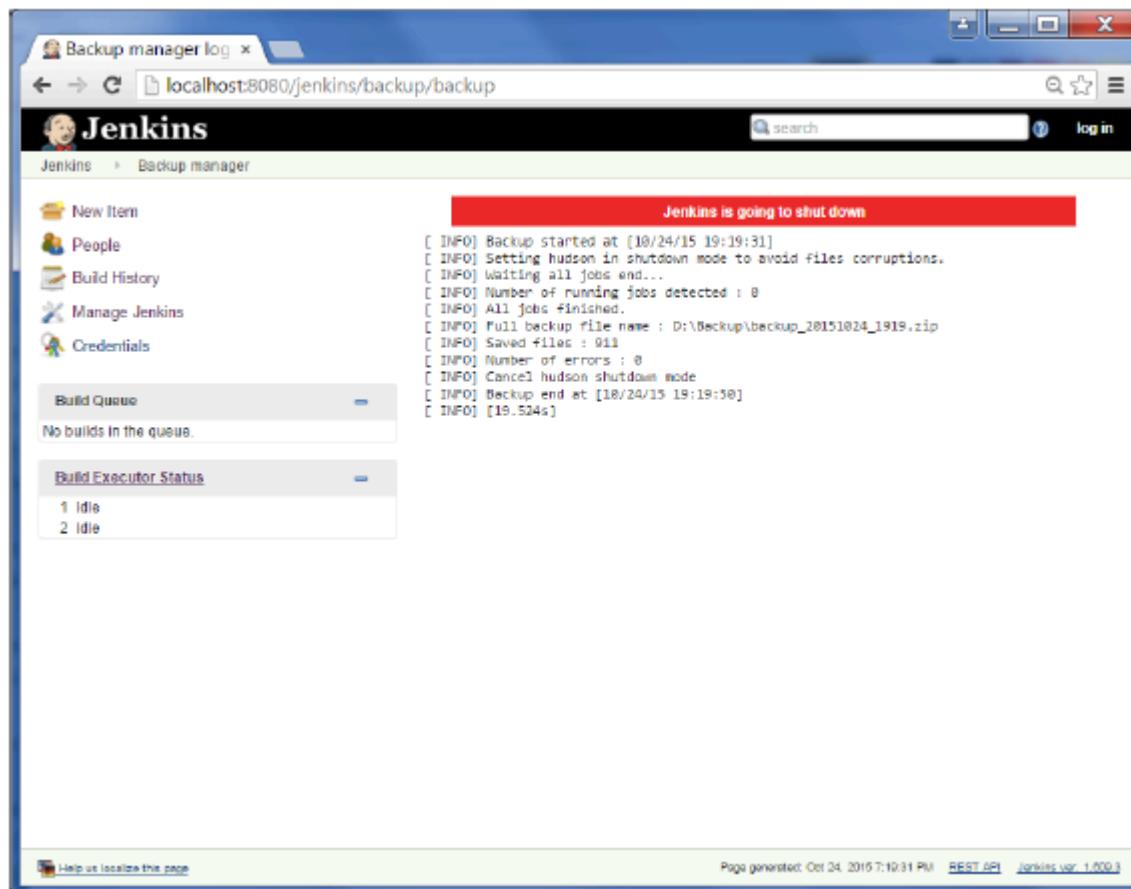
Step 5 – Here, the main field to define is the directory for your backup. Ensure it's on another drive which is different from the drive where your Jenkins instance is setup. Click on the Save button.

The screenshot shows the Jenkins 'Backup config files' configuration page. 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 has a title 'Backup config files'. It contains sections for 'Backup configuration' (Hudson root directory: E:\Jenkins, Backup directory: D:\Backup, Format: zip, File name template: backup_@date@.@extension@, Custom exclusions: none), 'Backup content' (Backup job workspace, Backup builds history, Backup maven artifacts archives, Backup fingerprints), and a 'Save' button. At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 7:17:45 PM', 'REST API', and 'Jenkins ver. 1.603.3'.

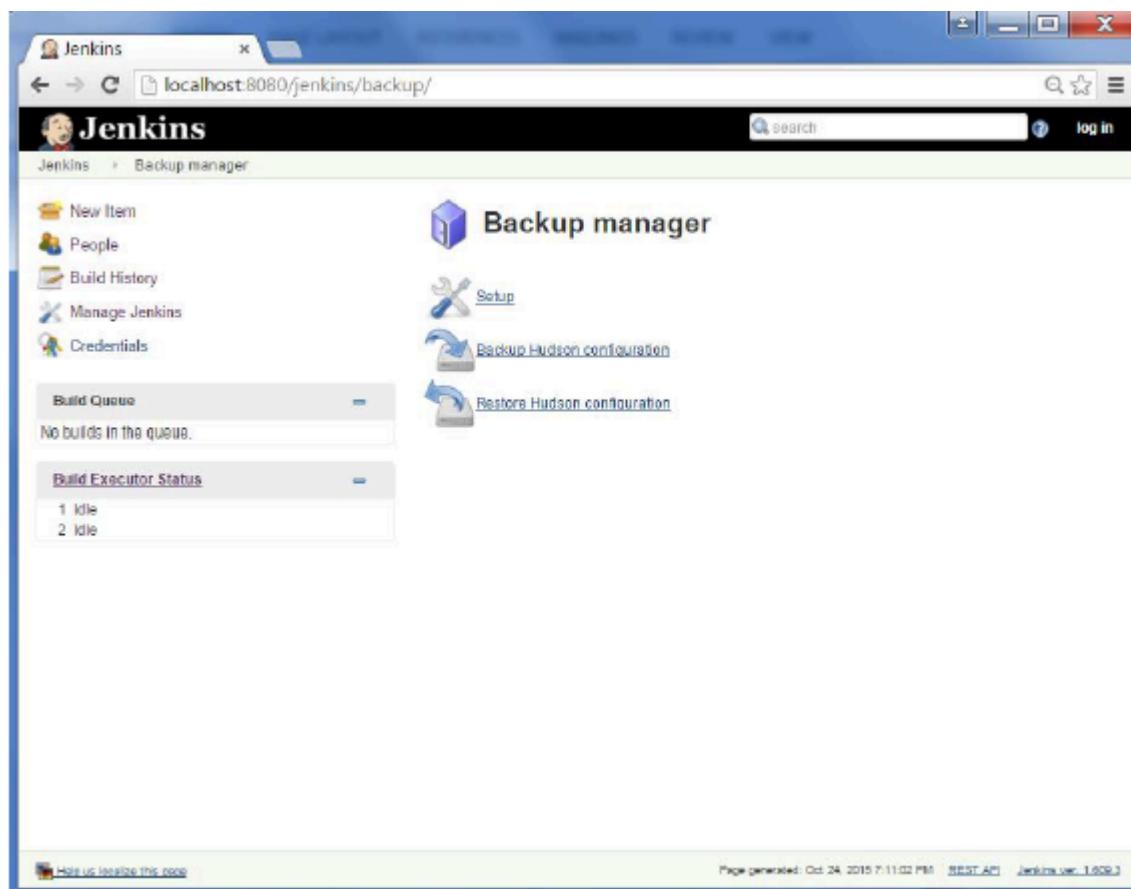
Step 6 – Click on the ‘Backup Hudson configuration’ from the Backup manager screen to initiate the backup.

The screenshot shows the Jenkins 'Backup manager' screen. The sidebar is identical to the previous one. The main area has a title 'Backup manager' and three options: '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, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 7:11:02 PM', 'REST API', and 'Jenkins ver. 1.603.3'.

The next screen will show the status of the backup



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



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

The screenshot shows the Jenkins Backup manager interface. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below these 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 "Backup manager" and displays "Available backup in D:\Backup :". A link "Launch restore" is shown next to a file entry "backup_20151024_1919.zip". At the bottom, there's a footer with links for Help, Localization, Page generated time, REST API, and Jenkins version.

Jenkins - Remote Testing

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

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

The screenshot shows the Jenkins Manage Jenkins page. The left sidebar lists several options: Manage Nodes, Manage Credentials, About Jenkins, Manage Old Data, In-process Script Approval, and Prepare for Shutdown. The "Manage Nodes" option is highlighted with a blue icon and a callout box. The main content area provides a brief description of each option. At the top of the main area, there are links for Apps, Import bookmarks now..., and ENABLE AUTO REFRESH.

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, it says '2 Idle'. Under the heading 'DXBMEM30', it shows '1 Idle'. A table lists the nodes:

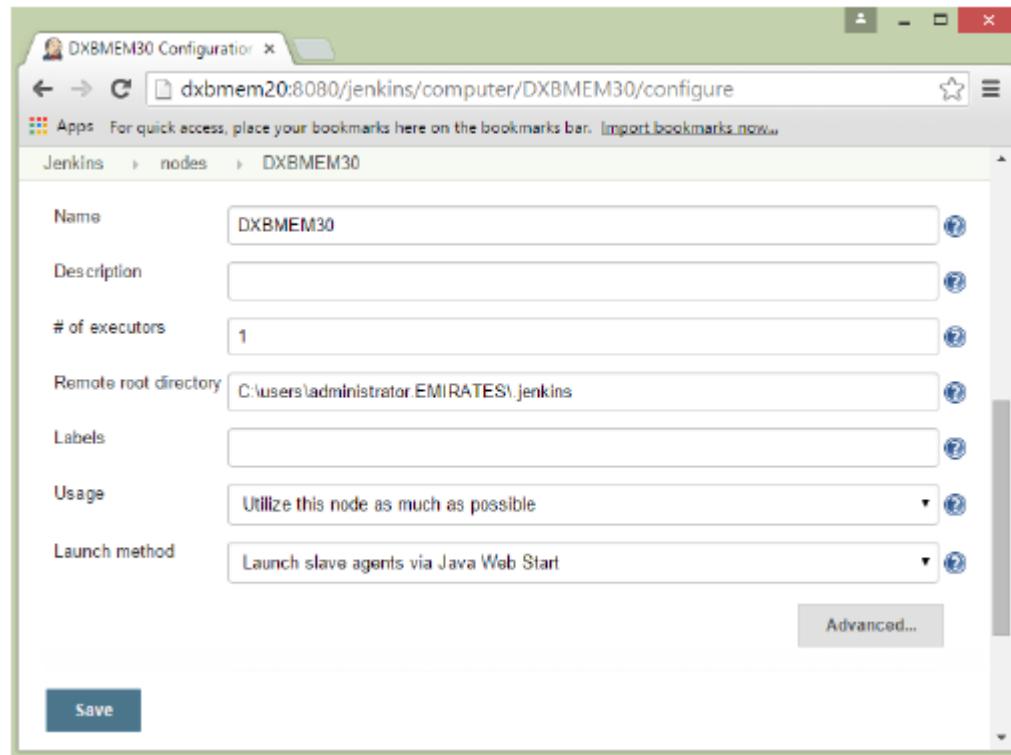
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
1	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB	13 min
2	master	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB	94.20

At the bottom right of the table is a 'Refresh status' button. The URL in the browser bar is 'dxbmem20:8080/jenkins/computer/'.

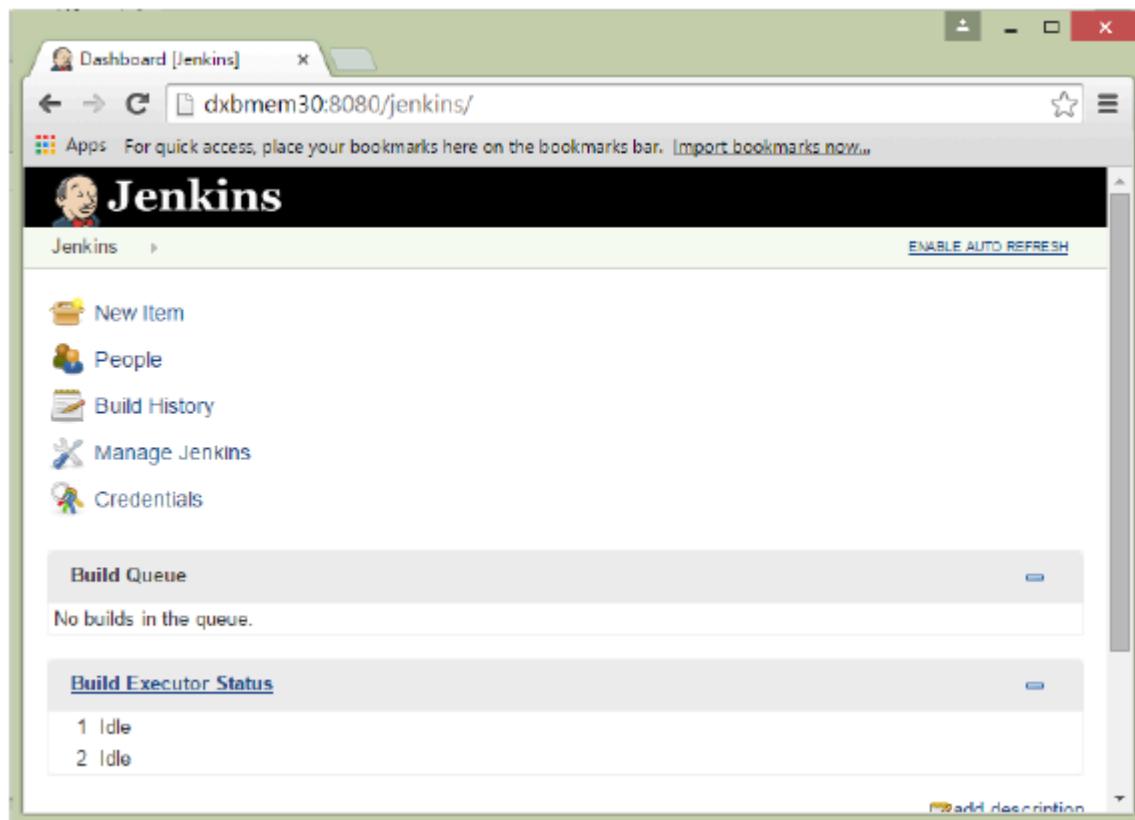
Step 2 – Click on configure for the DXBMEM30 slave machine.

The screenshot shows the same Jenkins 'Nodes' page as before. The 'DXBMEM30' node is selected. A context menu is open over the 'DXBMEM30' row, with the 'Configure' option highlighted. Other options in the menu are 'Delete Slave' and 'Build History'. The URL in the browser bar is 'dxbmem20:8080/jenkins/computer/.../configure'.

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



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



Step 5 – Click on the DXBMEM30 instance.

The screenshot shows the Jenkins interface for managing nodes. At the top, there's a header bar with the title 'Nodes [Jenkins]' and a URL 'dxbmeme20:8080/jenkins/computer/'. Below the header, the navigation path is 'Jenkins > nodes'. There's also a link to 'ENABLE AUTO REFRESH'.

The main content area displays the 'Build Executor Status' for two nodes:

- master**: Shows 1 Idle and 2 Idle executors.
- DXBMEM30**: Listed as (offline).

Below this, a table provides detailed information about the nodes:

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.

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

Connect slave to Jenkins one of these ways:

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

Created by anonymous user

Projects tied to DXBMEM30

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

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

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

Do you want to run this application?

Name: Jenkins Remoting Agent

Publisher: Kohsuke Kawaguchi

Locations: <http://dxbmemp20:8080>
Launched from downloaded JNLP file

Running this application may be a security risk

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

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

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

I accept the risk and want to run this application.

Run **Cancel**

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



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

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

The screenshot shows the Jenkins job configuration interface for a job named 'HelloWorld'. In the 'Label Expression' field, the value 'DXBMEM30' is entered. The 'Restrict where this project can be run' checkbox is checked. Below the configuration area, there are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins configuration interface for the 'HelloWorld' job. The browser title is 'HelloWorld Config [Jenkin]'. The URL in the address bar is 'dxbmemo20:8080/jenkins/job/HelloWorld/configure'. The page displays the 'SeleniumHQ htmlSuite Run' build step configuration. The configuration fields are:

- 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)

Below the configuration form are buttons for 'Delete', 'Add build step', 'Save', and 'Apply'.

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