

Jenkins - Quick Guide

Advertisements



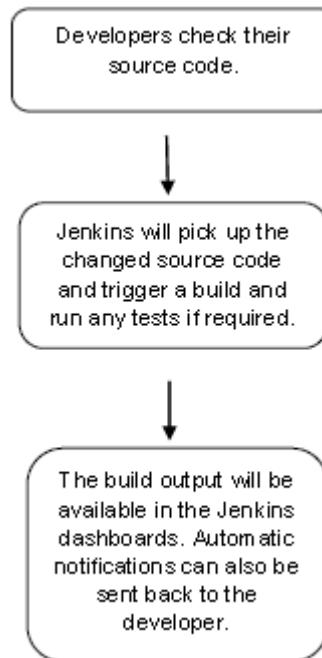
[Previous Page](#)

[Next Page](#)

Jenkins - Overview

Why Jenkins?

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



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

What is Continuous Integration?

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

System Requirements

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

Jenkins - Installation

Download Jenkins

The official website for Jenkins is [Jenkins](http://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 winstome.

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

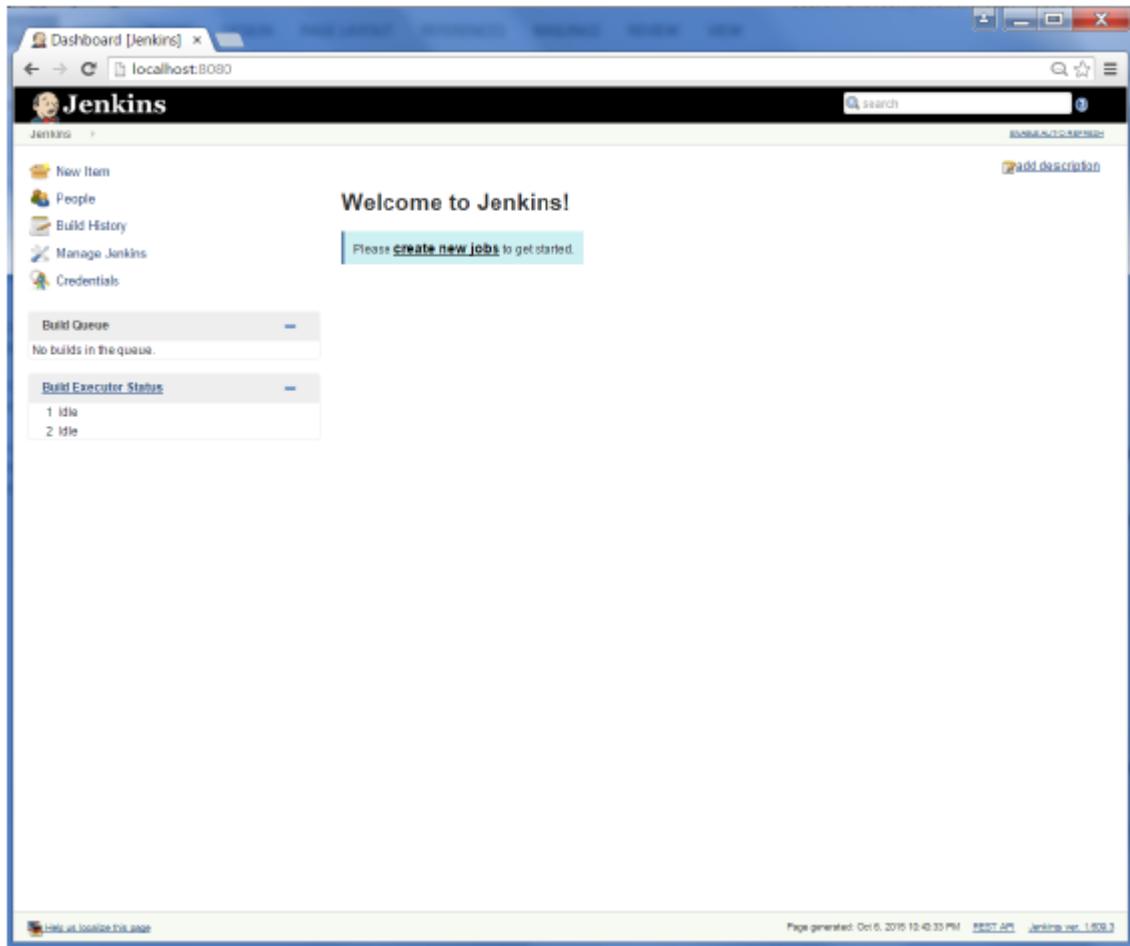
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

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

Accessing Jenkins

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

This link will bring up the Jenkins dashboard.



Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

Step 1: Verifying Java Installation

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

| os | Task | Command |
|---------|-----------------------|-----------------|
| Windows | Open command console | \>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 feather icon. It highlights the 'Tomcat 8.0.27 Released' on 2015-10-01, noting numerous fixes for issues identified in 8.0.26, along with other enhancements and changes. It also mentions the 'Tomcat 7.0.64 Released' on 2015-08-25, which includes fixes for POGO WebSocket endpoints and improved handling of async timeouts.

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

The screenshot shows the 'Tomcat 7 Downloads' page. The sidebar on the left is identical to the main homepage. The main content area has sections for 'Tomcat 7 Downloads', 'Quick Navigation' (with links for KEYS | 7.0.64 | Browse | Archives), 'Release Integrity' (warning about verifying file integrity using OpenPGP signatures), 'Mirrors' (listing mirrors and providing a dropdown for 'Other mirrors'), and '7.0.64' (which is currently selected). Under '7.0.64', it says to see the README file for packaging information. The 'Binary Distributions' section lists various download options, including 'Core' distributions (zip, tar.gz, 32-bit Windows zip, 64-bit Windows zip, 64-bit Itanium Windows zip, 32-bit/64-bit Windows Service Installer) and a link to 'Full documentation'.

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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

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

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

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

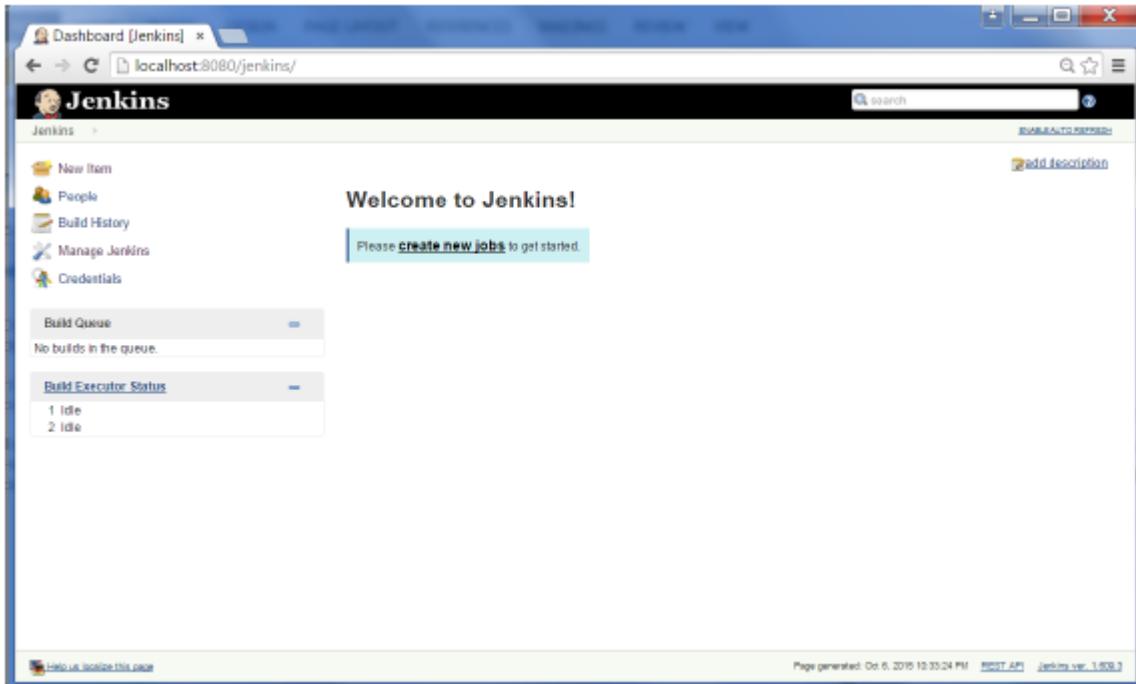
```
INFO: Server startup in 1302 ms
```

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

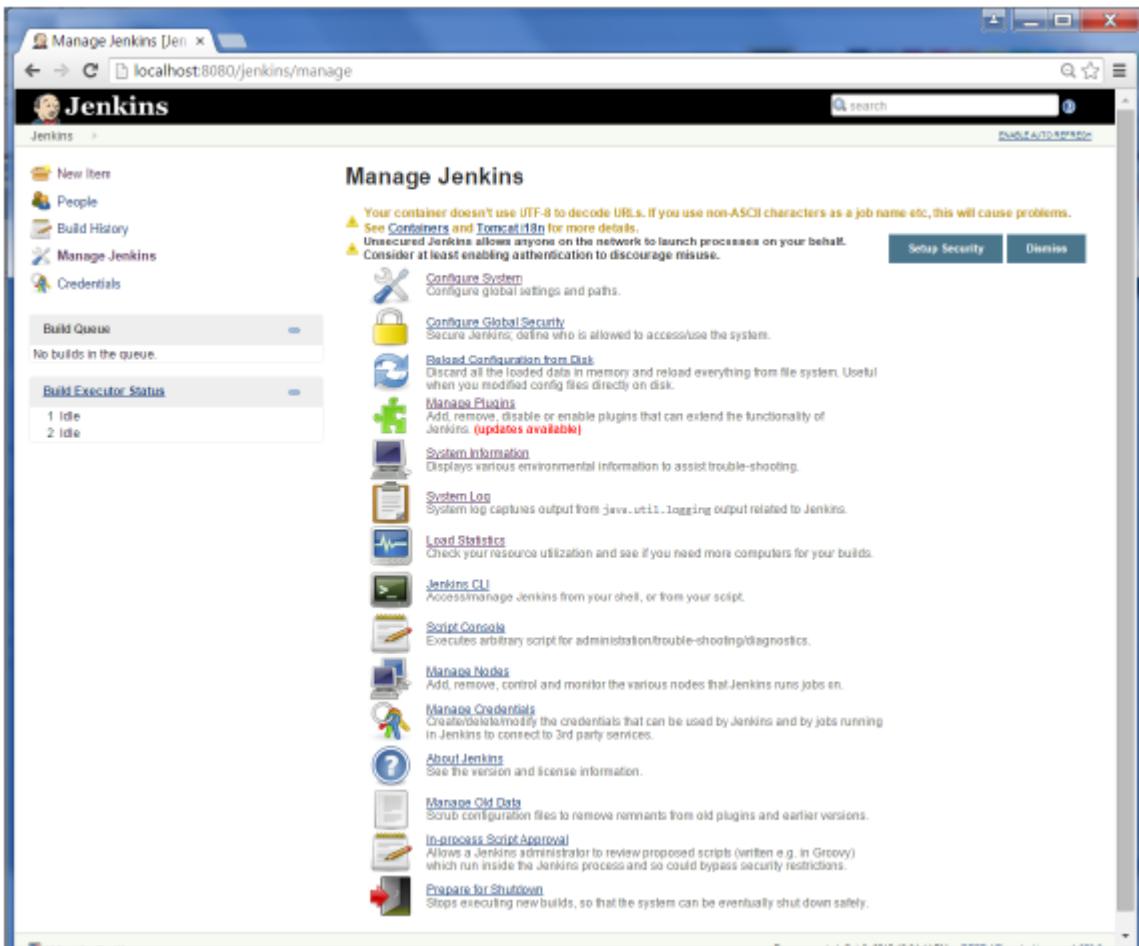


Jenkins - Git Setup

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



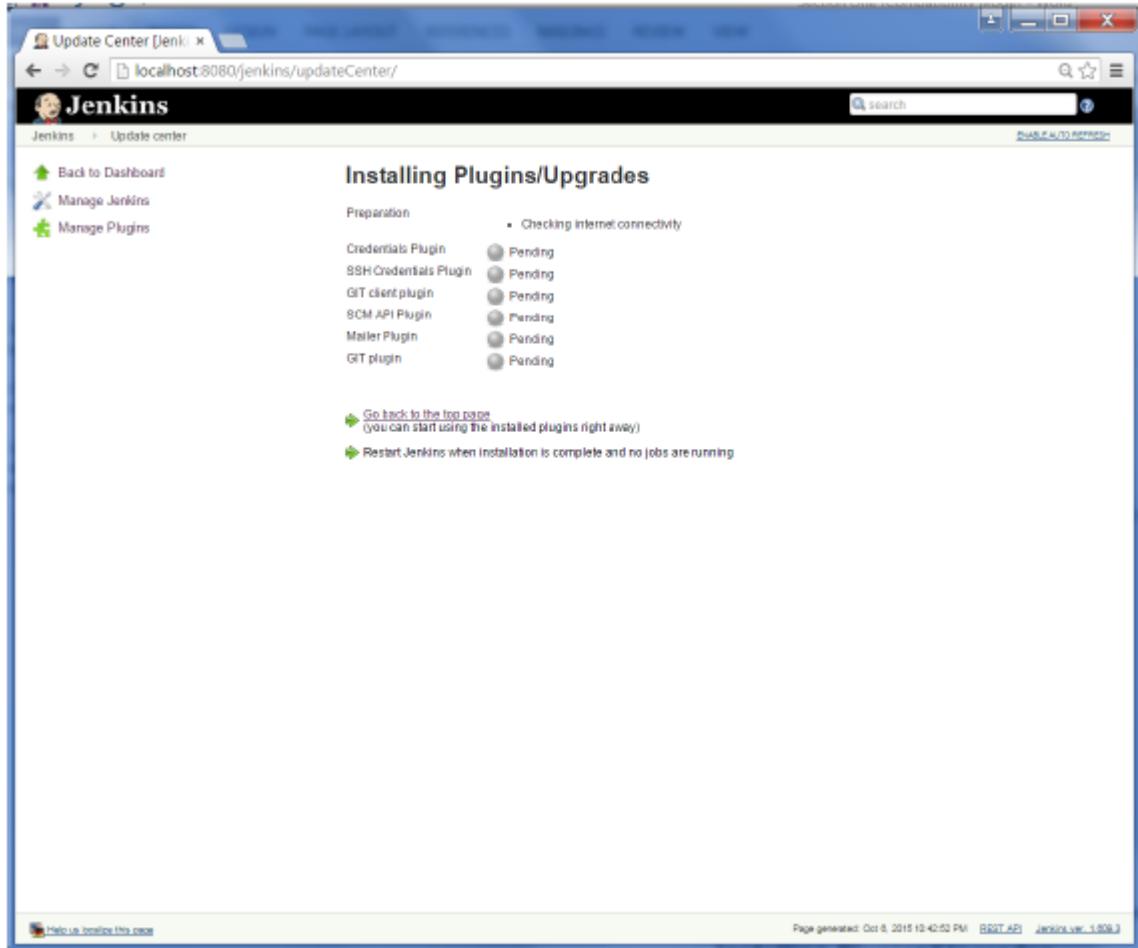
In the next screen, click the 'Manage Plugins' option.



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

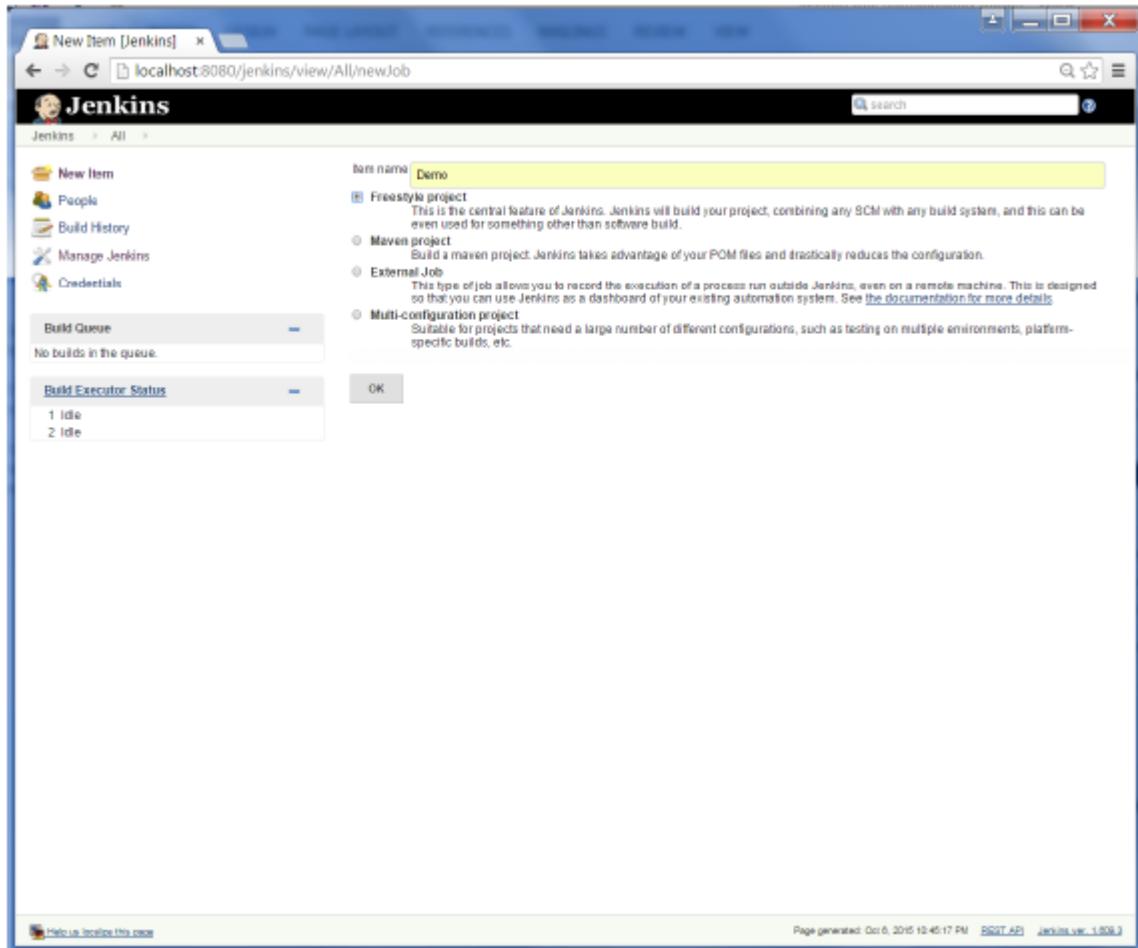
The list will then be filtered. Check the Git Plugin option and click on the button 'Install without restart'

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

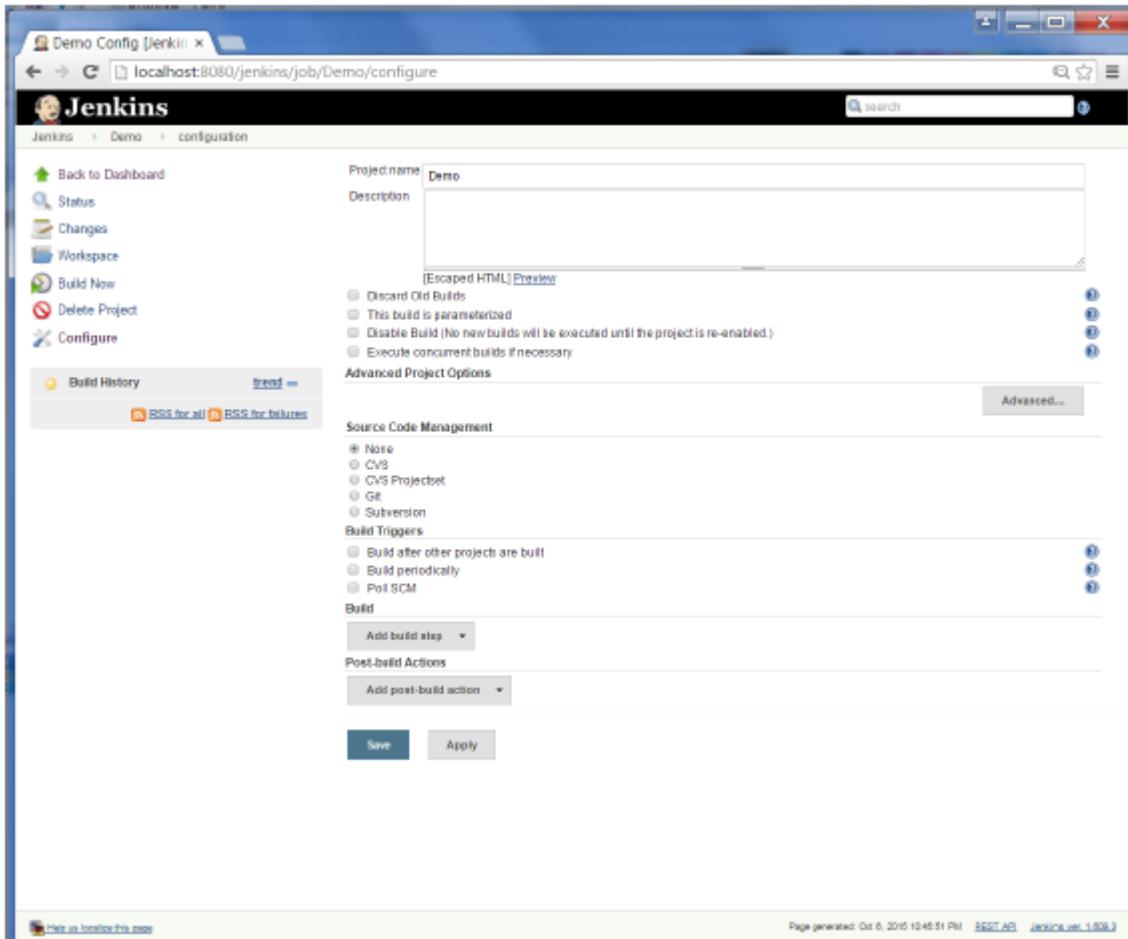


Once all installations are complete, restart Jenkins by issue the following command in the browser. **<http://localhost:8080/jenkins/restart>**

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



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



Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

The screenshot shows a web browser window titled "Maven - Download" with the URL <https://maven.apache.org/download.cgi>. The page features the Apache logo and the word "Maven" in large red letters. A sidebar on the left contains links for MAIN, Welcome, License, Download (which is highlighted), Install, Configure, Run, IDE Integration, ABOUT MAVEN, What is Maven?, Features, FAQ, Support and Training, DOCUMENTATION, Maven Plugins, Index (category), Running Maven, User Centre >, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources. The main content area has a heading "Downloading Apache Maven 3.3.3". It states that Apache Maven 3.3.3 is the latest release and recommended version for all users. It mentions the currently selected download mirror is <http://www.us.apache.org/dist/>. Below this is a "System Requirements" section 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.

| | Link | Checksum | Signature |
|---|--------------------------|-------------------------|---------------------|
| Maven 3.3.3 Complete Binary Distribution (Apache-Maven-3.3.3-bin.zip) | Download | SHA-256 | GPG |
| Maven 3.3.3 Complete Source Distribution (Apache-Maven-3.3.3-src.zip) | Download | SHA-256 | GPG |
| Maven 3.3.3 Complete Binaries (Apache-Maven-3.3.3-bin.tar.gz) | Download | SHA-256 | GPG |
| Maven 3.3.3 Complete Sources (Apache-Maven-3.3.3-src.tar.gz) | Download | SHA-256 | GPG |
| Maven 3.3.3 Complete Binaries (Apache-Maven-3.3.3-bin.zip) | Download | SHA-256 | GPG |
| Maven 3.3.3 Complete Sources (Apache-Maven-3.3.3-src.zip) | Download | SHA-256 | GPG |

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. On the left, there's a sidebar with navigation links like Support and Training, Documentation, Maven Plugins, Index (category), Running Maven, User Centre, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources, Security, Community Overview, How to Contribute, Maven Repository, Getting Help, Issue Tracking, Source Repository, The Maven Team, Project Documentation, Project Information, Maven Projects, Ant Tasks, Archetype, Doxia, and XPP. The main content area has sections for Memory (no minimum requirement), Disk (approx 10MB required for installation and repository), and Operating System (no minimum requirement). Below this is a 'Files' section with a table showing download links, checksums, and signatures for Binary tar.gz, Binary zip, Source tar.gz, and Source zip archives. A note says Maven is distributed in several formats for convenience. Another note advises verifying signatures. A list of distribution details follows:

- 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

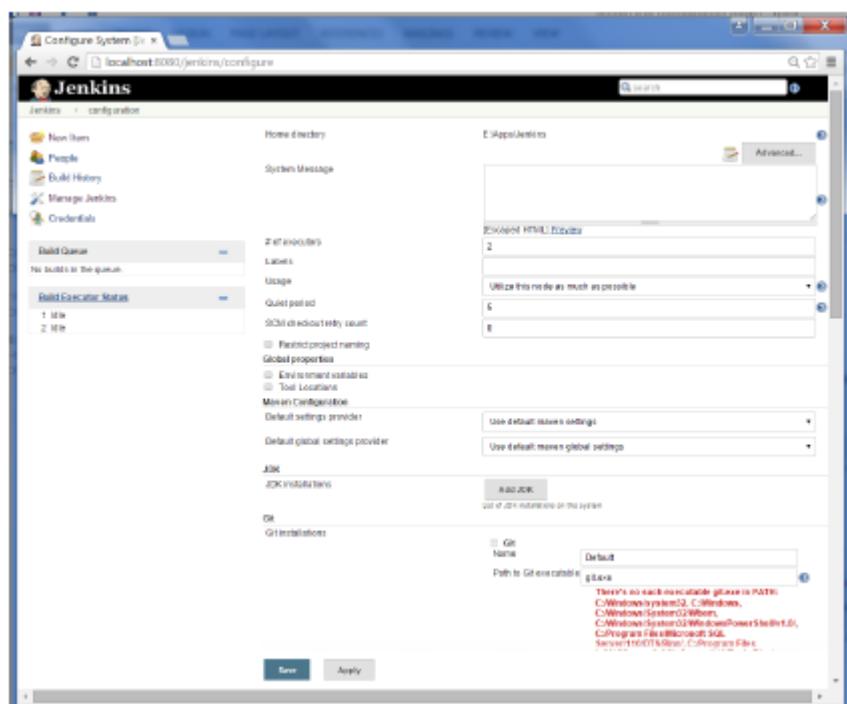
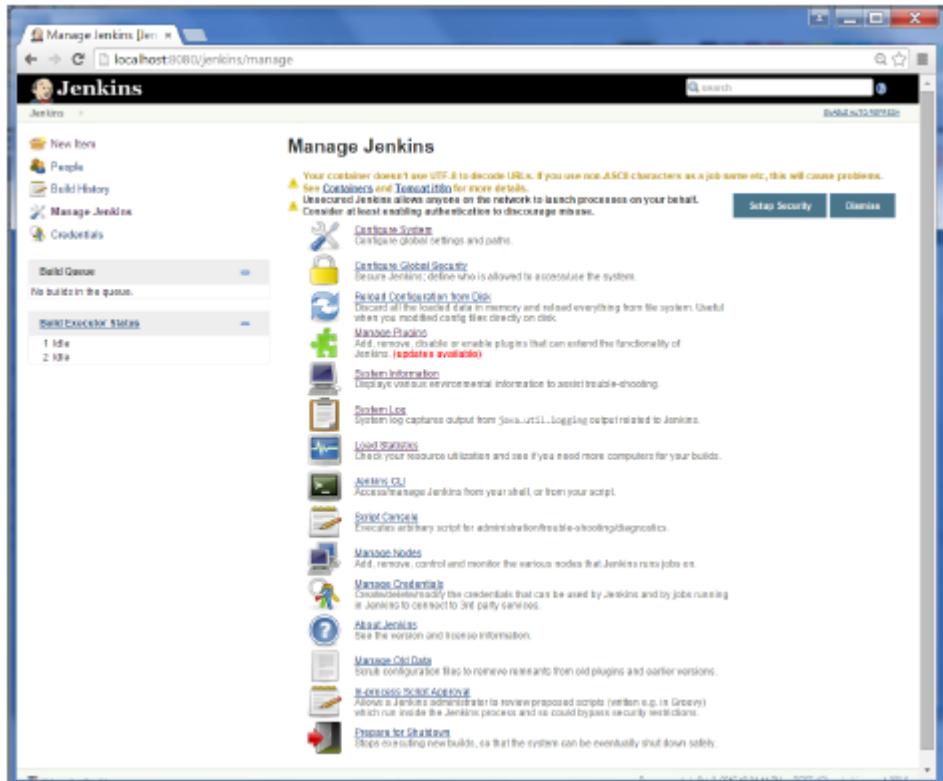
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

Step 2: Setting up Jenkins and Maven

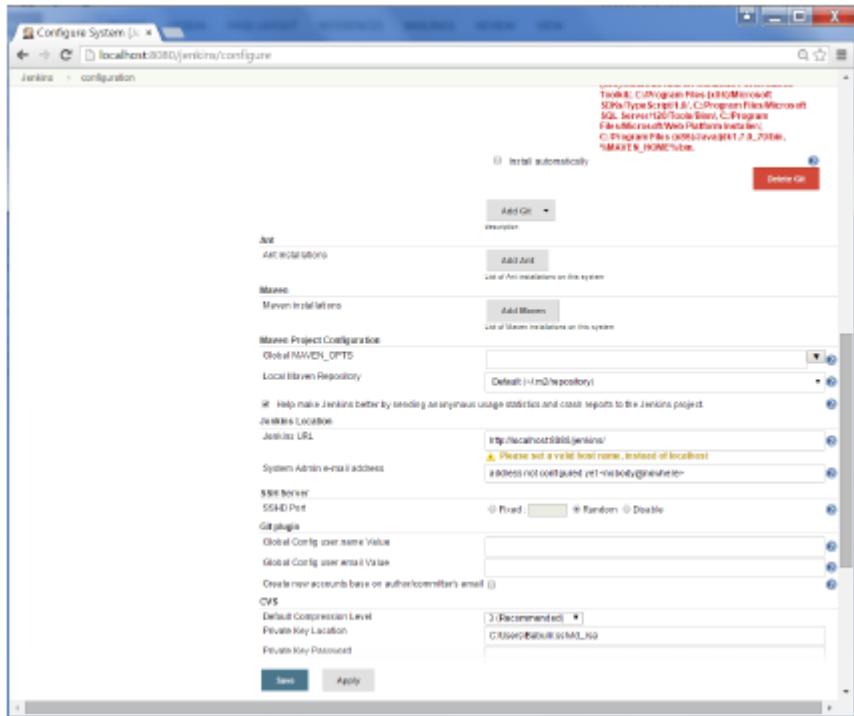
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.



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



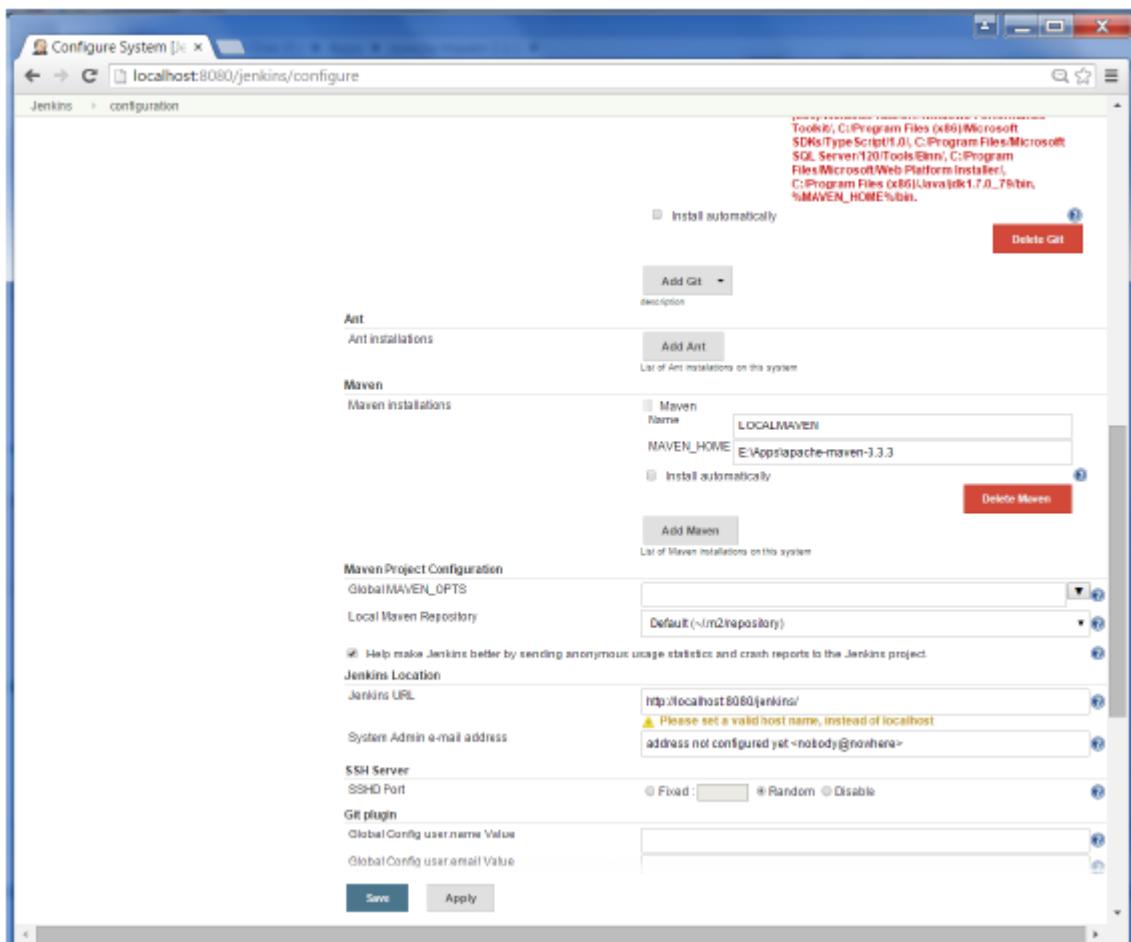
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



Uncheck the 'Install automatically' option.

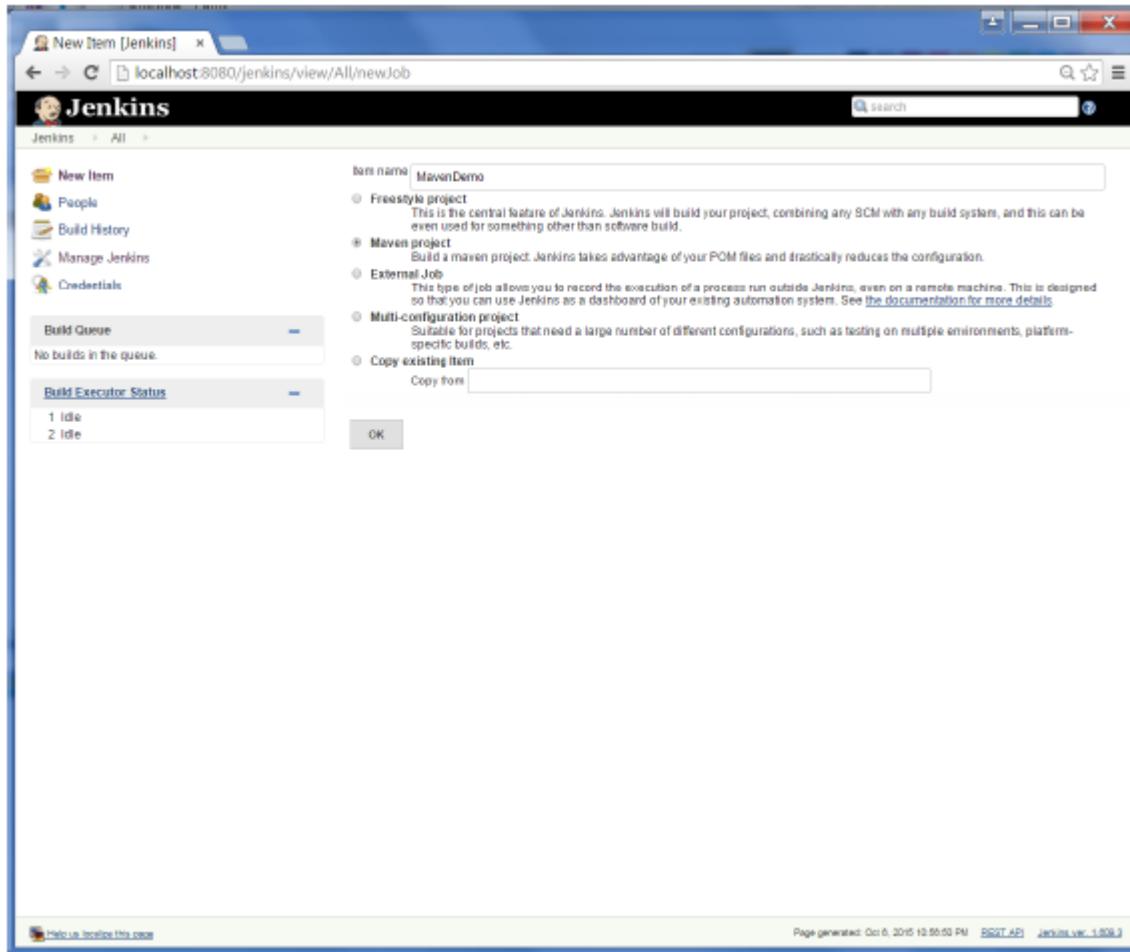
Add any name for the setting and the location of the MAVEN_HOME.

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



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

The screenshot shows the Jenkins dashboard at the URL localhost:8080/jenkins/. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area displays a table of jobs with columns: S, W, Name, Last Success, Last Failure, and Last Duration. A single job, 'Demo', is listed with an icon of a sun, indicating it is healthy ('W'). The 'Last Success' and 'Last Failure' fields show 'N/A'. The 'Last Duration' field is also 'N/A'. Below the table, there is a legend with three RSS feed icons labeled 'RSS for all', 'RSS for failures', and 'RSS for last failed builds'. At the bottom of the page, there is a link 'Help us improve this page' and a footer note 'Page generated: Oct 6, 2015 12:55:57 PM' along with Jenkins version information.



Jenkins - Configuration

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

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

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a table for the 'Demo' job, which is currently Idle. A legend indicates that grey means 'Idle' and yellow means 'Warning'. Below the table, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom, there's a link to 'Help Us Improve This Page' and a footer note about page generation.

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

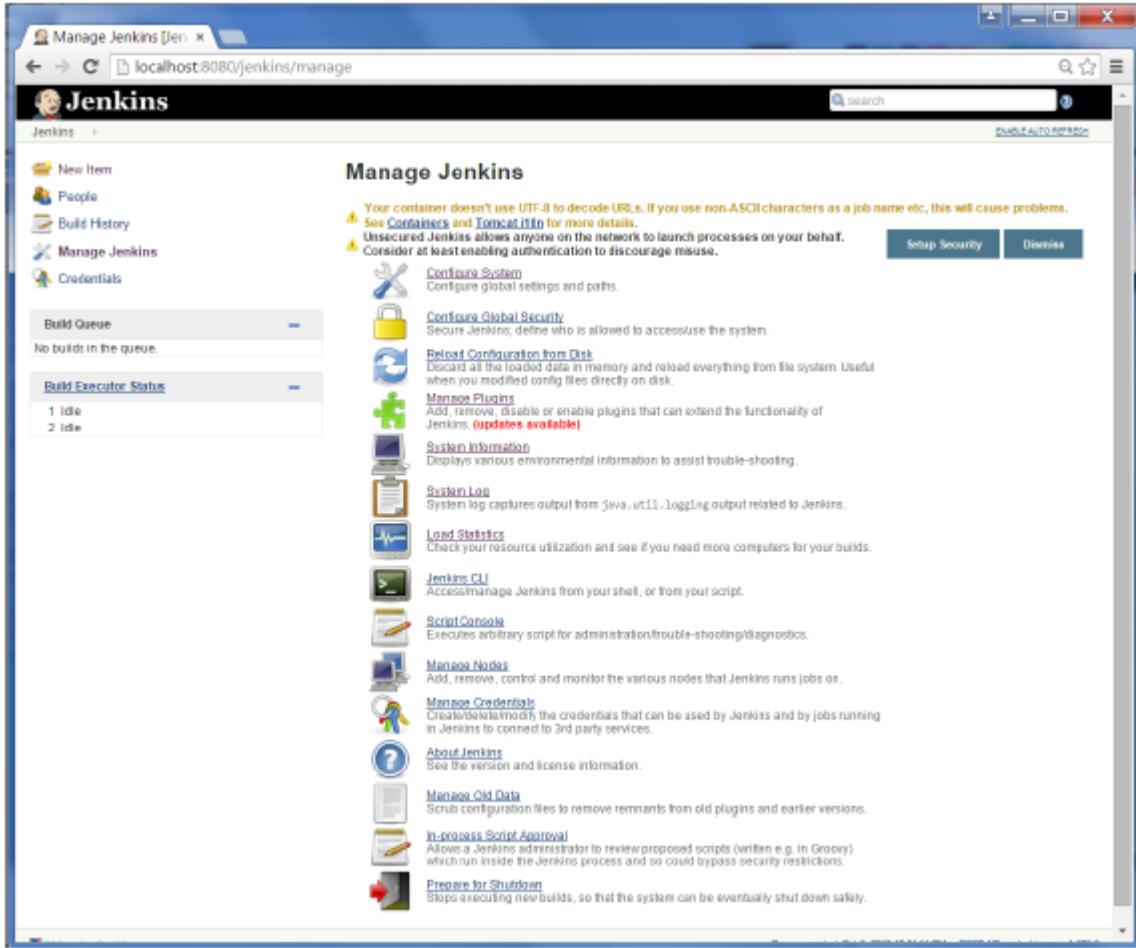
Legend: Idle Warning Failure Success

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle.

Help Us Improve This Page | Page generated: Oct 6, 2015 12:55:57 PM | REST API | Jenkins ver. 1.593

You will then be presented with the following screen –



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

Jenkins Home Directory

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

Set "JENKINS_HOME" environment variable to the new home directory before launching the servlet container.

Set "JENKINS_HOME" system property to the servlet container.

Set JNDI environment entry "JENKINS_HOME" to the new directory.

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

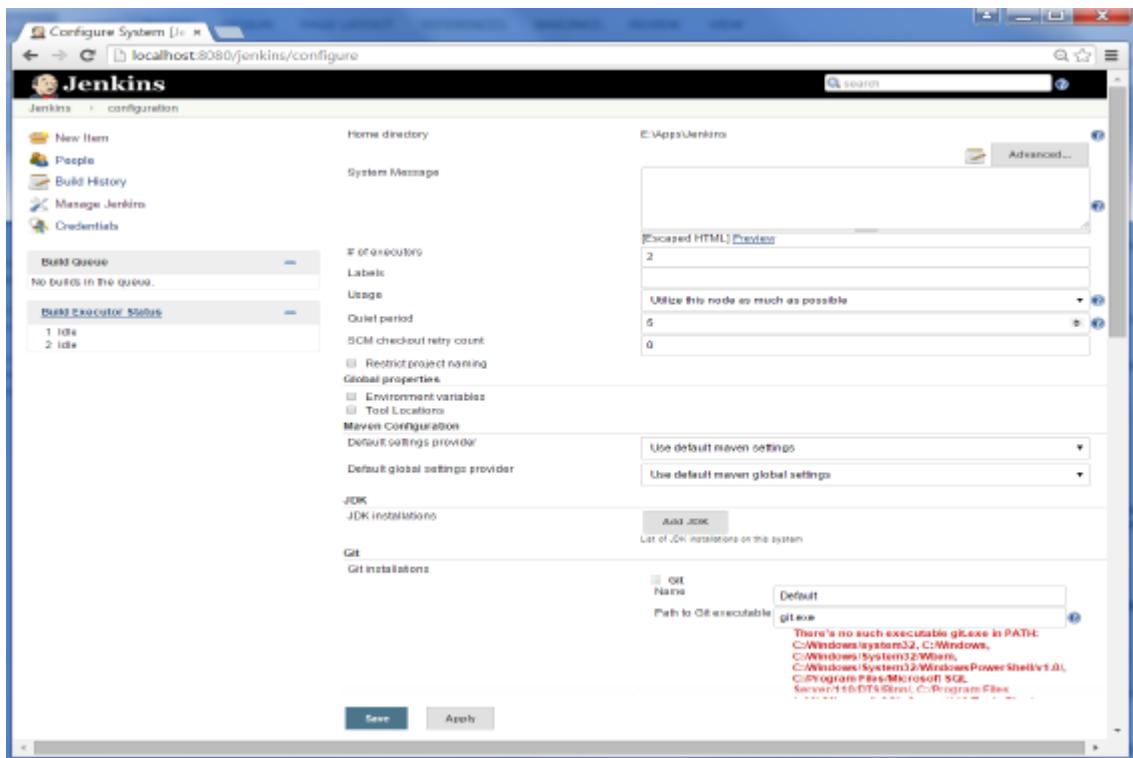
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/.jenkins` to this new directory.

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

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

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

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



of executors

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

Environment Variables

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

Jenkins URL

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

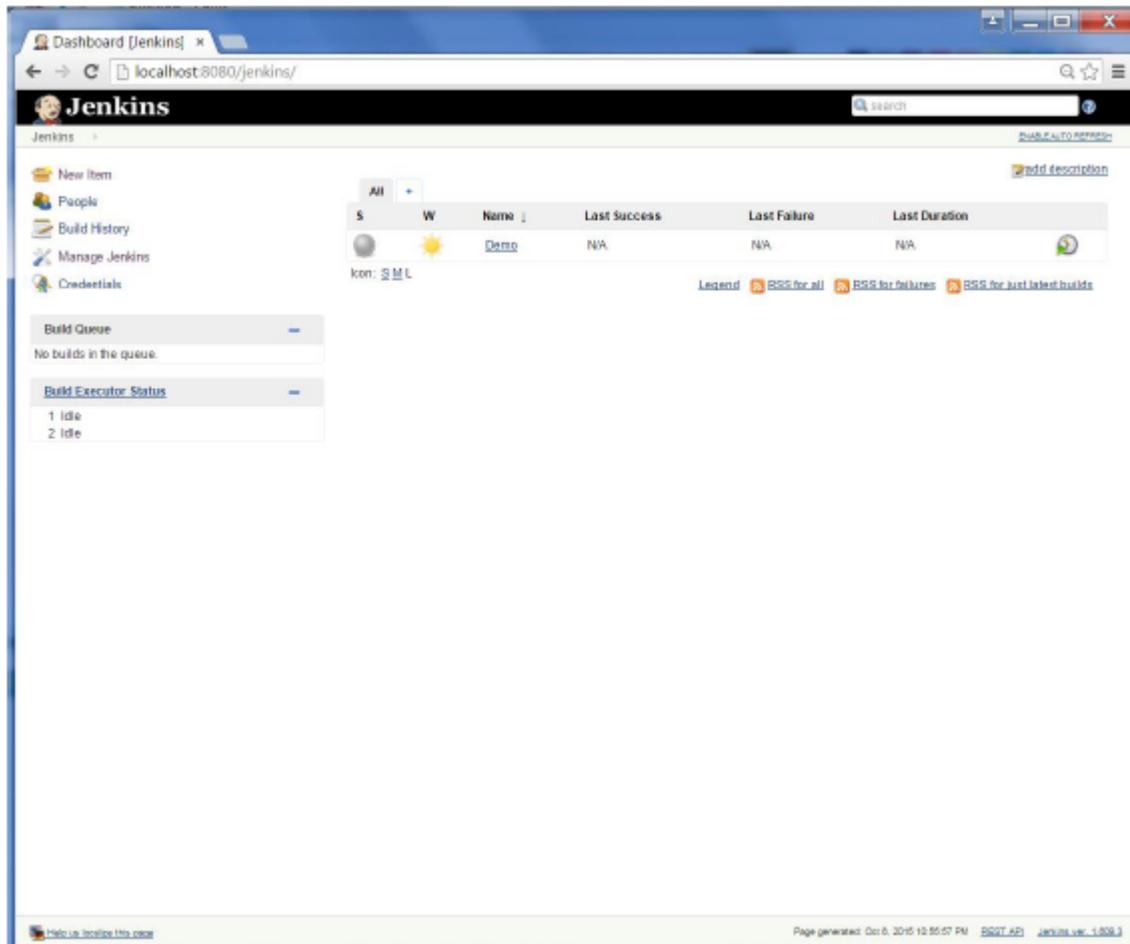
Email Notification

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

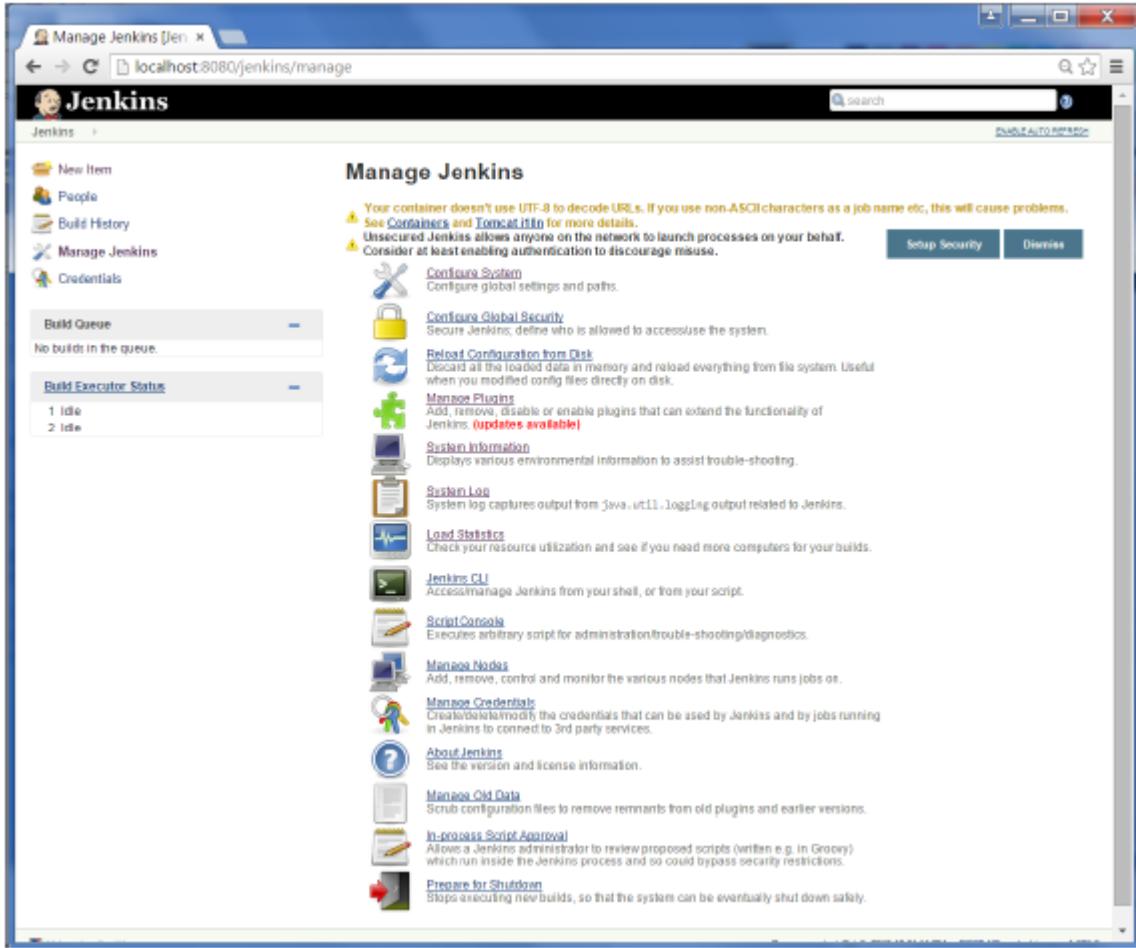
Jenkins - Management

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

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



You will then be presented with the following screen –



Some of the management options are as follows –

Configure System

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

Reload Configuration from Disk

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

Manage Plugin

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

The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes links for 'Back to Dashboard' and 'Manage Jenkins'. The main content area has tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Updates' tab is selected, displaying a list of available plugins:

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

Below the table are two buttons: 'Download now and install after restart' and 'Check now'. A status message indicates 'Update information obtained: 1 hr 36 min ago'. At the bottom, there's a note: 'Select All, None' and 'This page lists updates to the plugins you currently use.' The footer includes links for 'Help us localize this page' and 'Page generated: Oct 6 2015 11:08:25 PM EEST (EDT) Jenkins ver. 1.603'.

System Information

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

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

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

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

System Log

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

Load Statistics

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

Script Console

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

Manage nodes

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

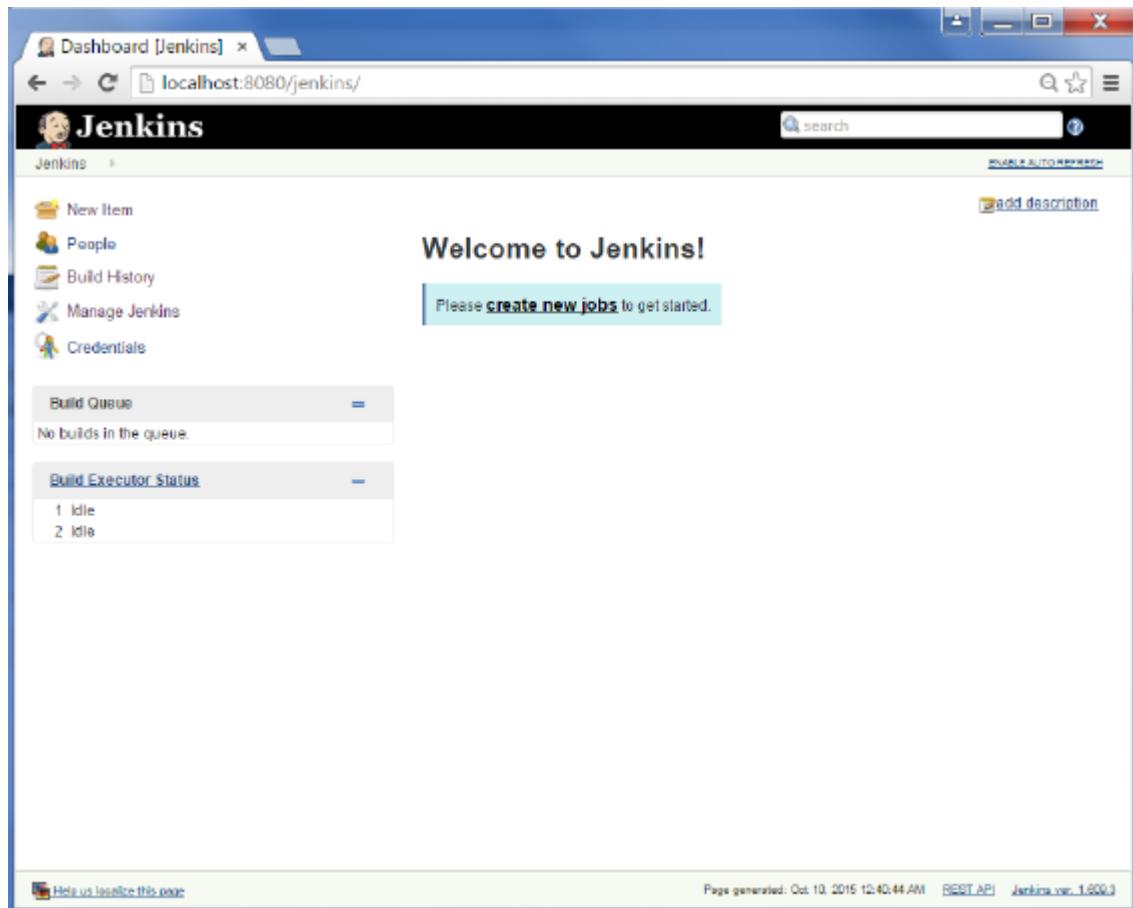
Prepare for Shutdown

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

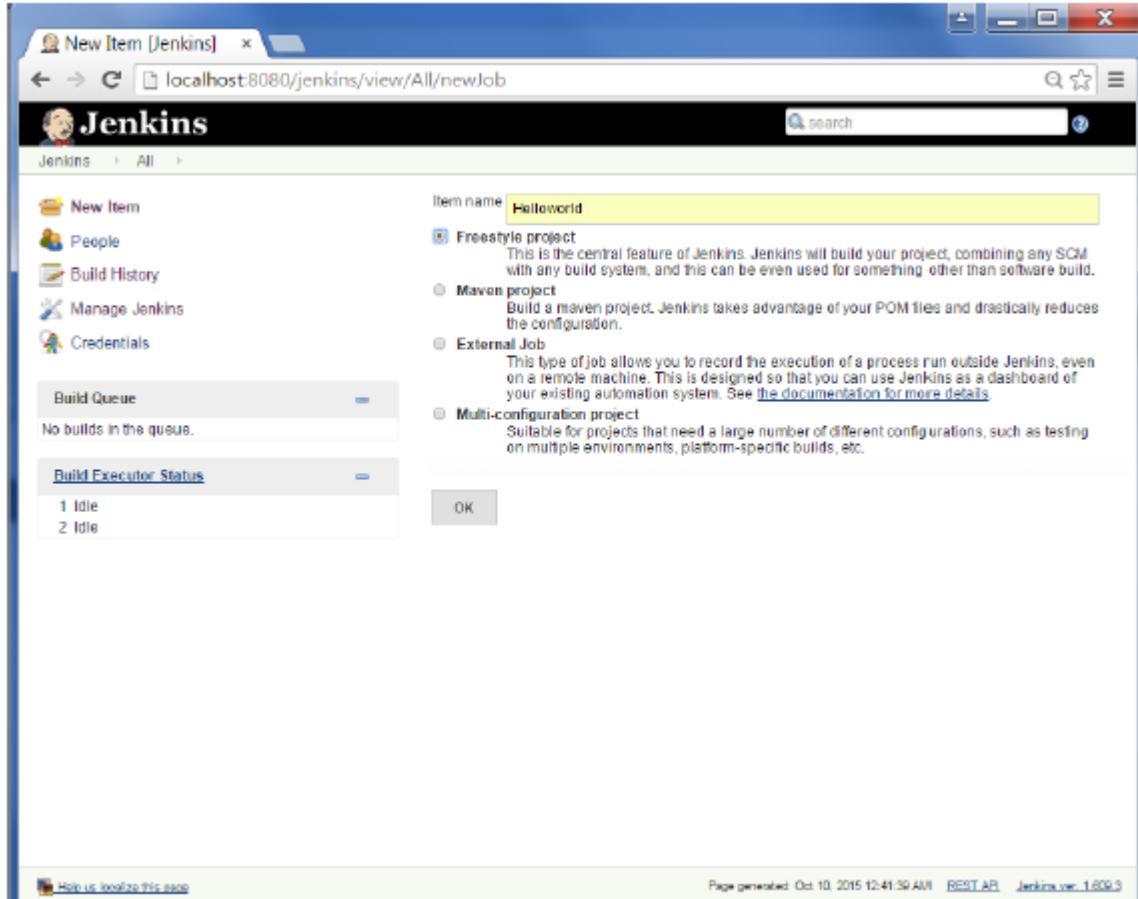
Jenkins - Setup Build Jobs

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

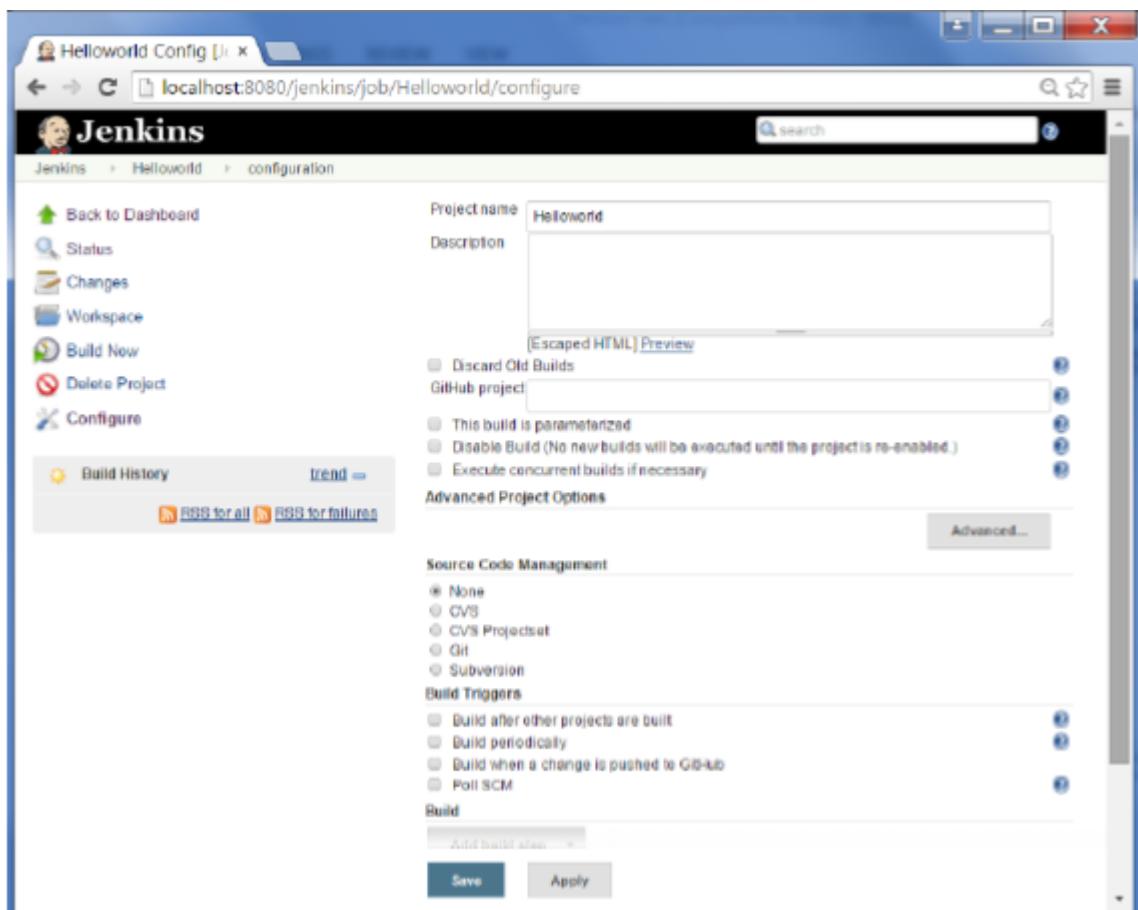
Step 1 – Go to the Jenkins dashboard and Click on New Item



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

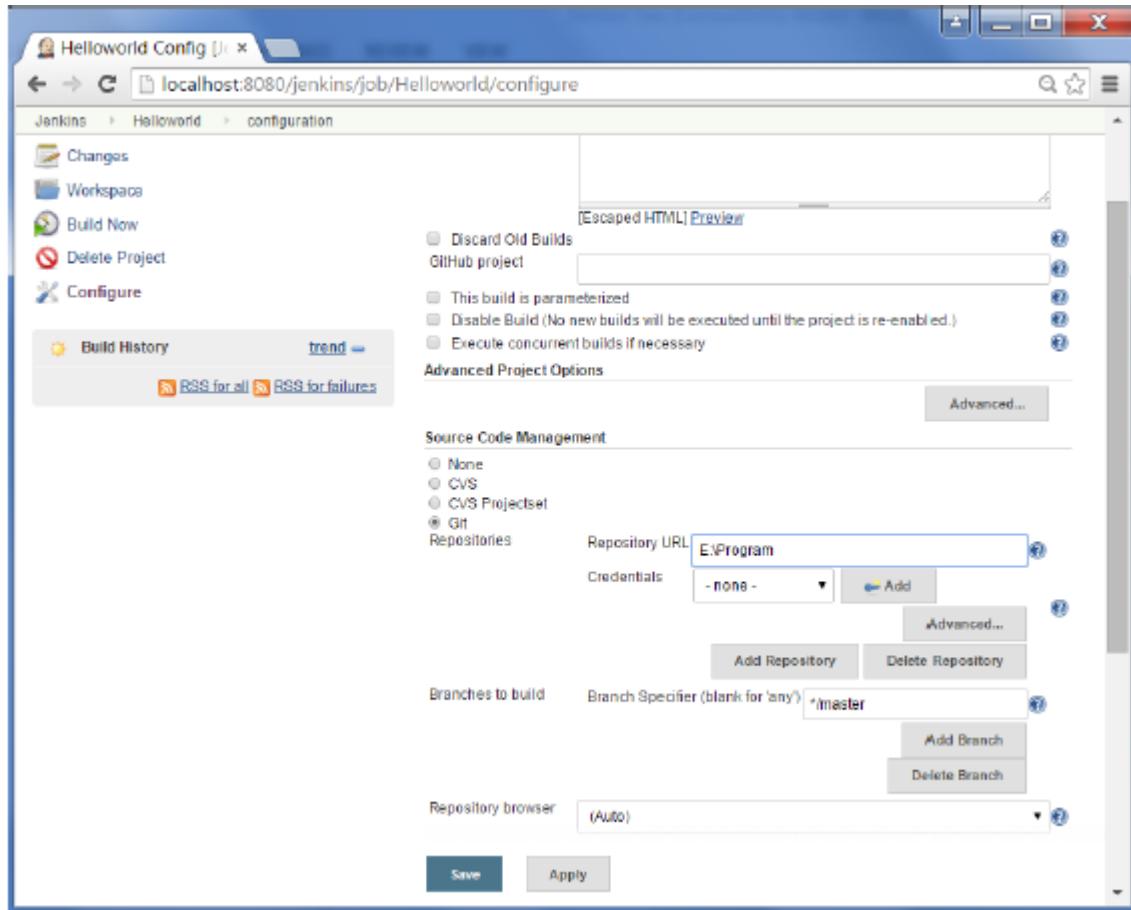


Step 3 – The following screen will come up in which you can specify the details of the job.

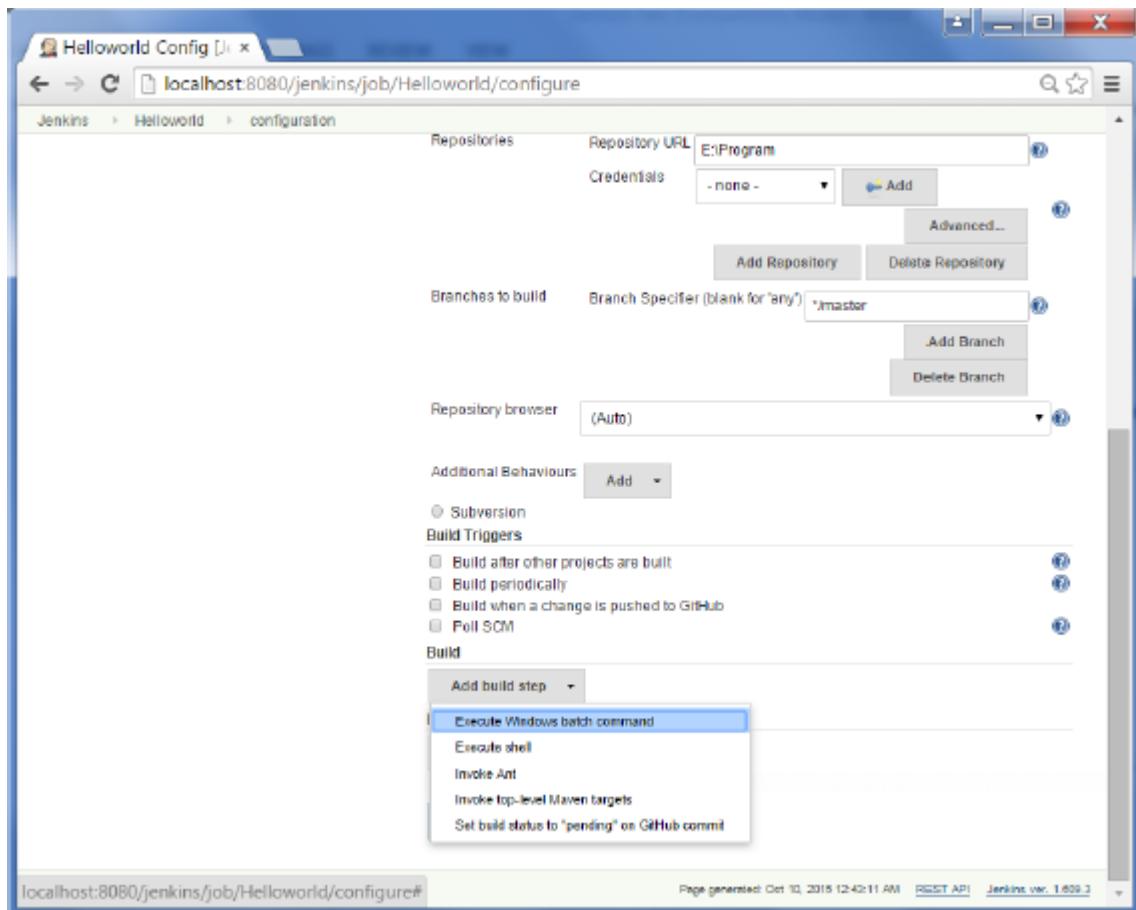


Step 4 – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

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



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



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

```
Java HelloWorld.java
```

```
Java HelloWorld
```

The screenshot shows the Jenkins configuration page for the 'Helloworld' job. At the top, there are tabs for 'Repository browser' (set to '(Auto)'), 'Delete Branch' (button), and a dropdown for 'Additional Behaviours' with 'Add' and 'Subversion' selected. Below this, the 'Build Triggers' section includes options for 'Build after other projects are built', 'Build periodically', 'Build when a change is pushed to GitHub', and 'Poll SCM'. The 'Build' section contains a step titled 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'. There are buttons for 'Save' and 'Apply' 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 'Project Helloworld'. On the left, a sidebar lists options: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area displays the project name 'Project Helloworld' and two links: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). On the right, there are buttons for 'Add description' and 'Disable Project'. At the bottom, there are sections for 'Build History' (with a sun icon), 'Permalinks' (with a link icon), and RSS feeds ('RSS for all' and 'RSS for failures'). The footer includes a 'Help us localize this page' link and a timestamp 'Page generated: Oct 10, 2015 12:42:11 AM'.

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'. On the right, there are buttons for 'Add description' and 'Disable Project'. Below the sidebar, there's a 'Build History' section showing one build (#1) from 'Oct 10, 2015 12:52 AM'. It has links for 'RSS for all' and 'RSS for failures'. To the right of the history is a 'Recent Changes' 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. 1.600.3'.

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 Project Helloworld dashboard. At the top right, there are links for "add description" and "Disable Project". On the left, a sidebar lists options: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. Below this is a "Build History" section showing one build (Oct 10, 2015 12:52 AM) with links for "RSS for all" and "RSS for failures". To the right, there are links for "Workspace" and "Recent Changes". A "Permalinks" section is also present.

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

The screenshot shows the Jenkins Build #1 (Oct 10, 2015 12:52:50 AM) details page. The build was started 4 min 40 sec ago and took 4.7 sec. The sidebar on the left includes options: Back to Project, Status, Changes, Console Output (which is selected), Edit Build Information, Delete Build, Git Build Data, and No Tags. The main content area displays build information: "No changes.", "Started by anonymous user", and "Revision: 42f9a82ffadd86fb5c3a9d8e40e731a907f5c8f · refs/remotes/origin/master".

The screenshot shows a Jenkins job named "Helloworld" with build number #12. The "Console Output" tab is selected. The output log is displayed, showing the build process starting from cloning the repository to running the Java application and exiting 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/origin^{commit}" # timeout=10
Checking out Revision 42f9a82ffad086f05c5a9dfeae40e731a907f5c8f
(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
42f9a82ffad086fb5c3a9dfeae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffad086fb5c3a9dfeae40e731a907f5c8f # 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

xunit - installations

| Month | Installations |
|-------|---------------|
| 10 | 11000 |
| 11 | 11200 |
| 12 | 11400 |
| 01 | 11600 |
| 02 | 11800 |
| 03 | 12000 |
| 04 | 12200 |
| 05 | 12400 |
| 06 | 12600 |
| 07 | 12800 |
| 08 | 13000 |
| 09 | 13200 |

Installations

- 2014-Oct 11692
- 2014-Nov 11557
- 2014-Dec 11631
- 2015-Jan 12105
- 2015-Feb 12262
- 2015-Mar 12891
- 2015-Apr 12894
- 2015-May 12716
- 2015-Jun 13143
- 2015-Jul 13470
- 2015-Aug 13192
- 2015-Sep 13663

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

CppUnit output

xUnit Plugin - Jenkins

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

Features

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

Supported tools

Embedded tools

- * JUnit itself
- * [JUnit](#)
- * [MSUnit](#) (imported from [MSTest Plugin](#))
- * [NUnit](#) (imported from [NUnit Plugin](#))
- * [UnitTest++](#)
- * [Boost Test Library](#)
- * [PHPUnit](#)
- * [Free Pascal Unit](#)
- * [CppUnit](#)
- * [MbUnit](#)
- * [GoogleTest](#)
- * [EmblUnit](#)
- * [gtest/glib](#)
- * [QTestLib](#)

Other plugins as an extension of the xUnit plugin:

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

External contributions

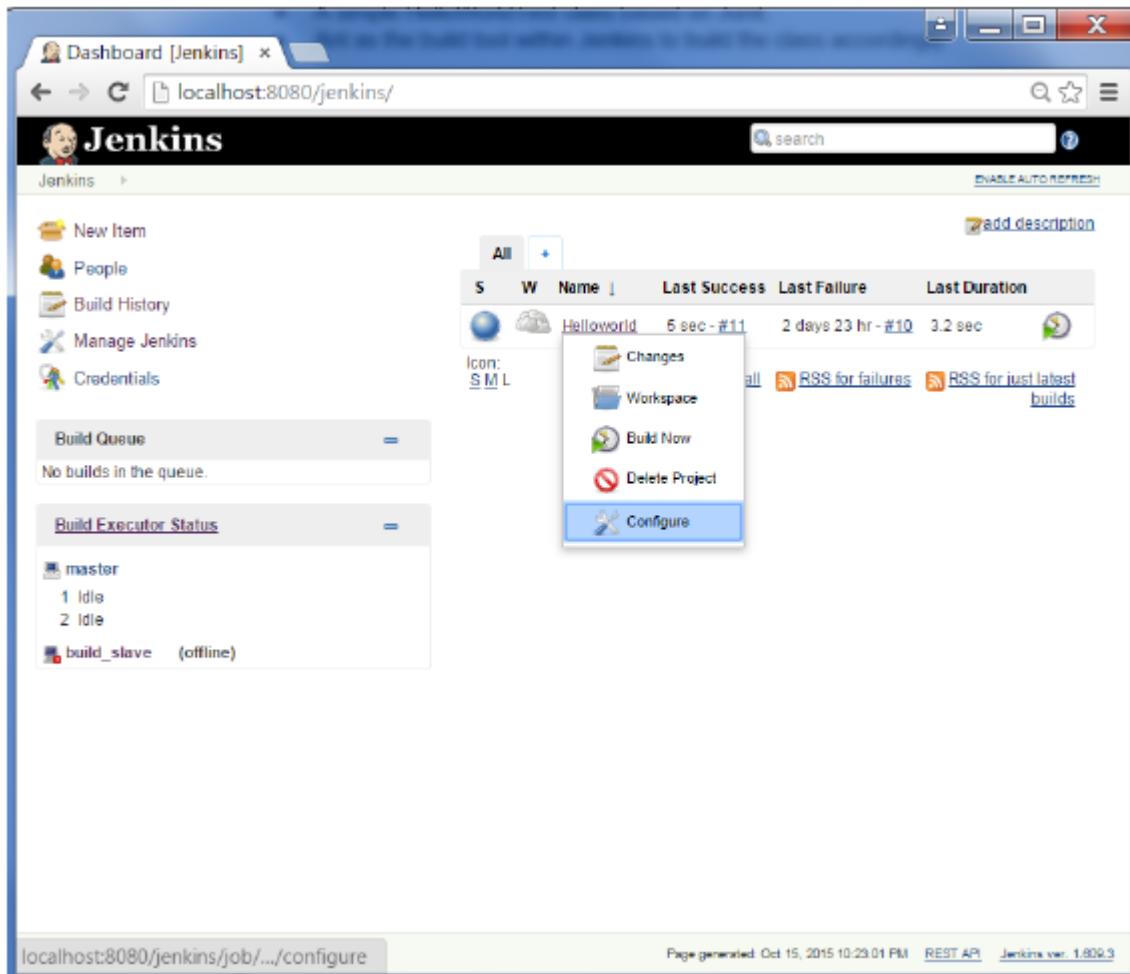
Example of a Junit Test in Jenkins

The following example will consider

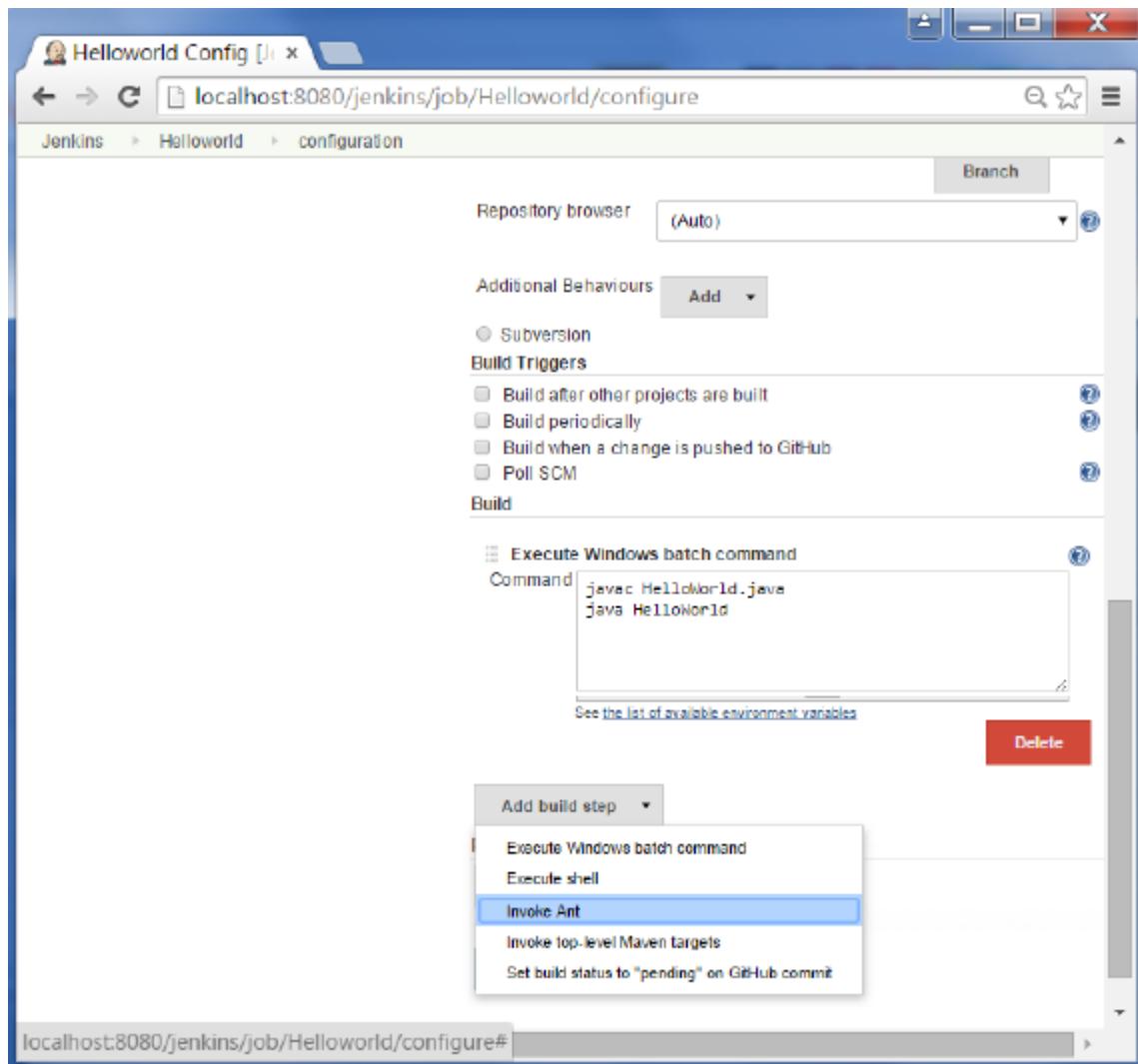
A simple HelloWorldTest class based on Junit.

Ant as the build tool within Jenkins to build the class accordingly.

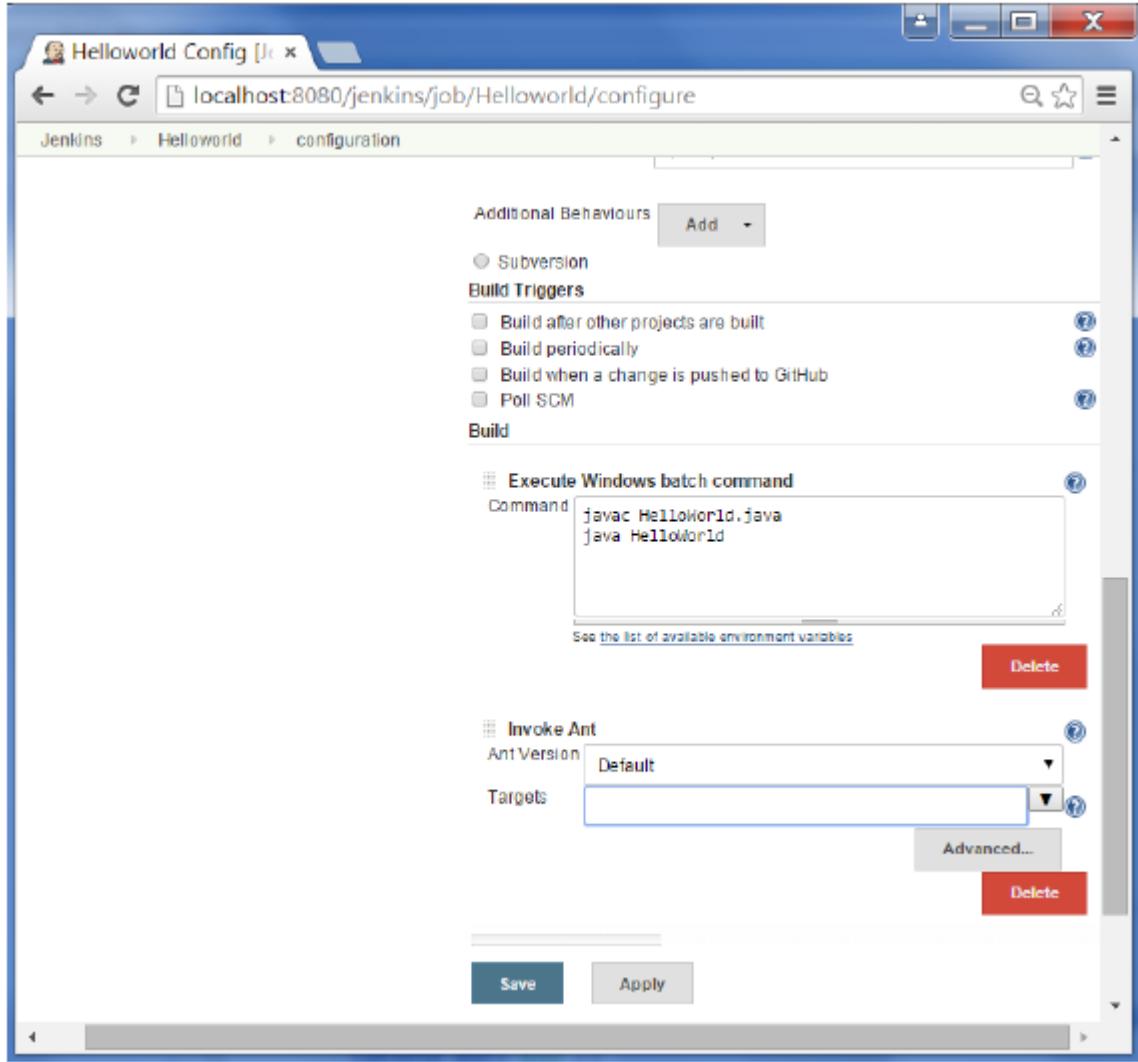
Step 1 – Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option



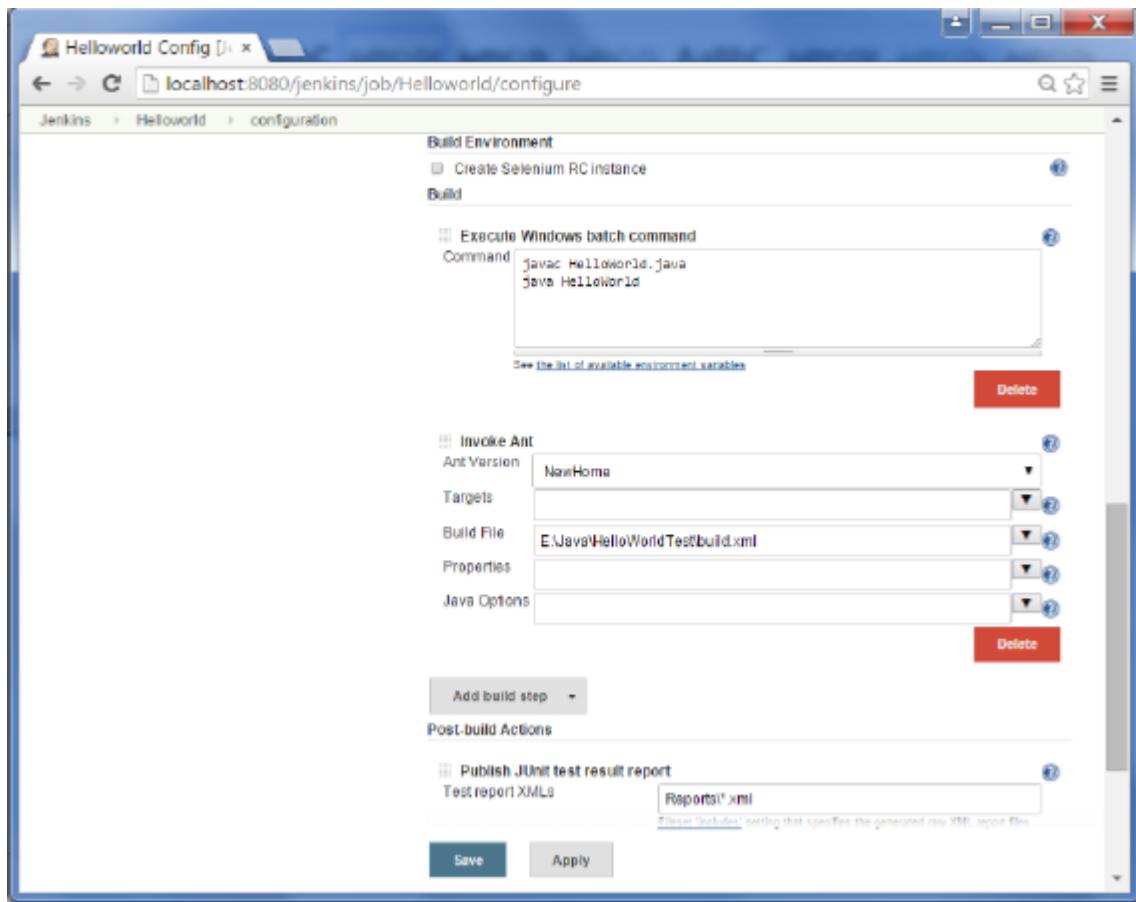
Step 2 – Browse to the section to Add a Build step and choose the option to Invoke Ant.



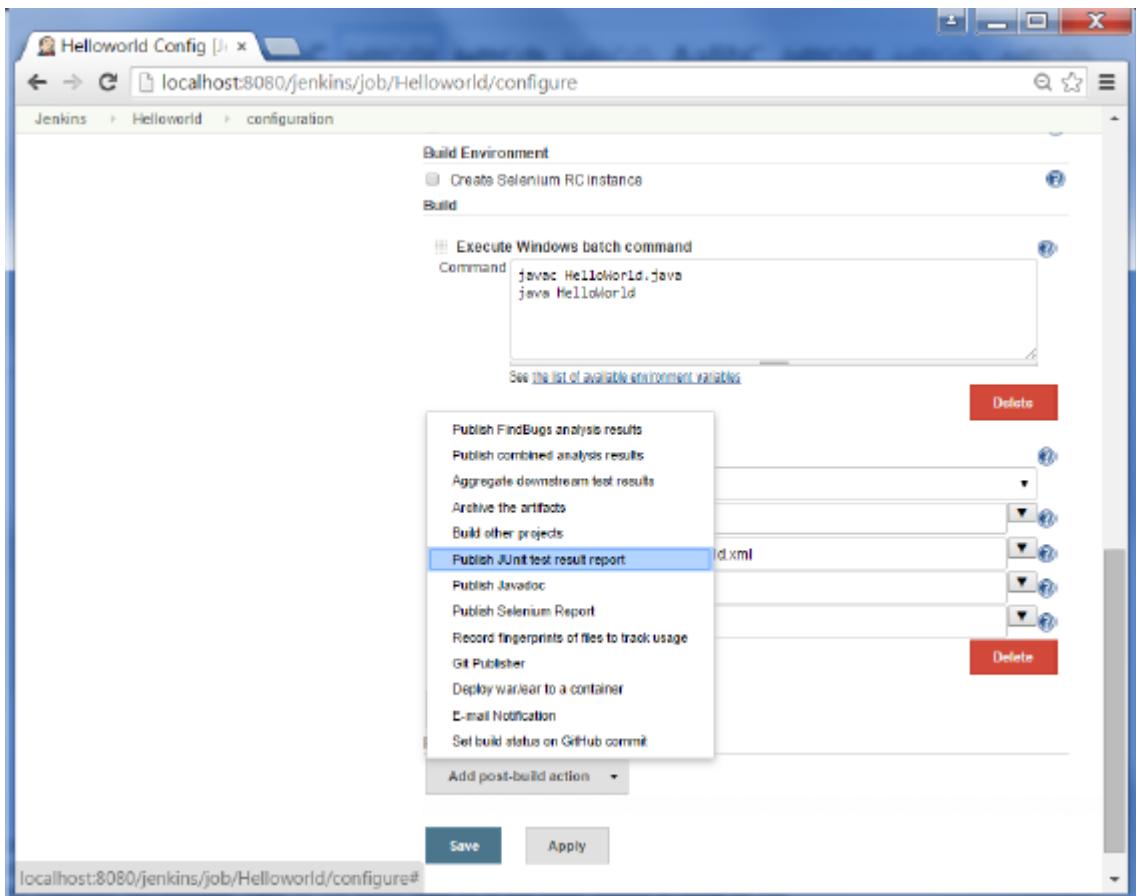
Step 3 – Click on the Advanced button.



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

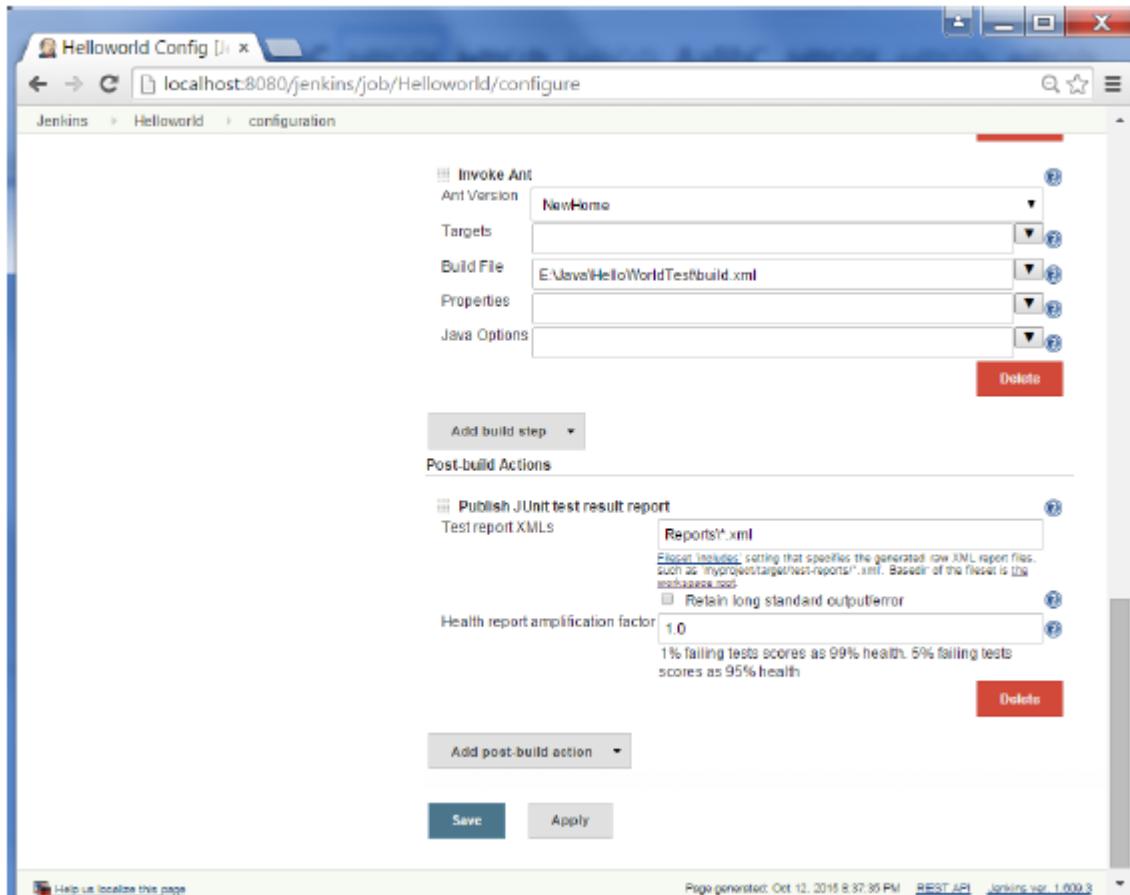


Step 5 – Next click the option to Add post-build option and choose the option of “Publish Junit test result report”



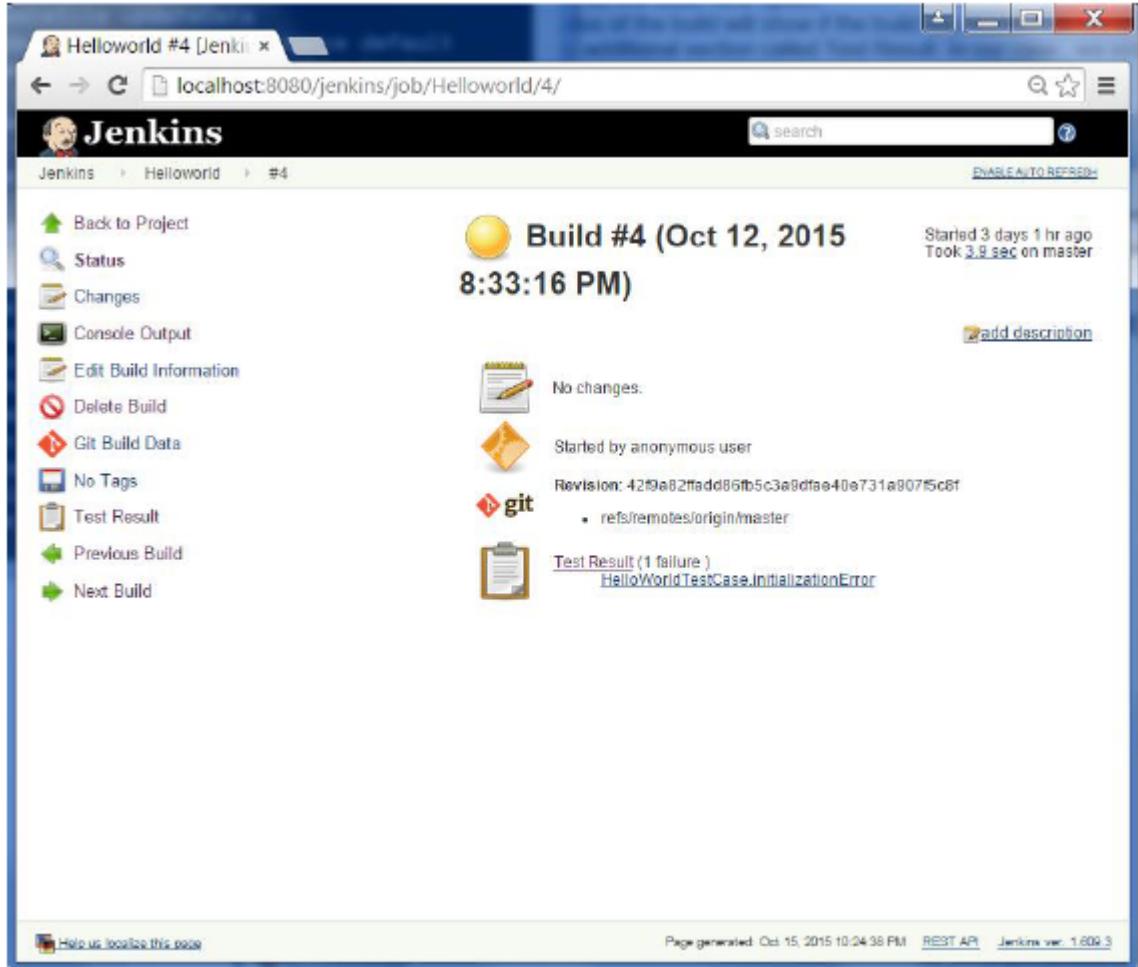
Step 6 – In the Test reports XML's, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which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



A screenshot of a web browser displaying a Jenkins job details page. The URL in the address bar is `localhost:8080/jenkins/job/Helloworld/4/`. The page title is "Helloworld #4 [Jenkins]". On the left, there's a sidebar with links like "Back to Project", "Status", "Changes", etc. The main content area shows "Build #4 (Oct 12, 2015 8:33:16 PM)". It includes a "No changes." message, a "Started by anonymous user" entry, a "git" commit log entry, and a "Test Result (1 failure)" section which lists "HelloWorldTestCase.InitializationError". The bottom of the page has a footer with links for "Help us localize this page", "Page generated: Oct 15, 2015 10:24:38 PM", "REST API", and "Jenkins ver. 1.609-3".

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

The screenshot shows the Jenkins Test Result page for the 'Helloworld' project, build #4. The main title is 'Test Result' with '1 failures'. A red bar indicates '1 tests Took 10 ms'. Below this, 'All Failed Tests' lists a single entry: 'HelloWorldTestCase InitializationError' with a duration of '10 ms'. The 'All Tests' section shows a summary table:

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

Page generated: Oct 12, 2015 9:45:49 PM | REST API | Jenkins ver. 1.606.3

Jenkins - Automated Testing

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

Step 1 – Go to Manage Plugins.

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
- Configure Global Security
- Reload Configuration from Disk
- Manage Plugins
- System Information
- System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes
- Manage Credentials
- About Jenkins

A warning message at the top right states: "Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat119](#) for more details." It also mentions "Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse." Buttons for 'Setup Security' and 'Dismiss' are present.

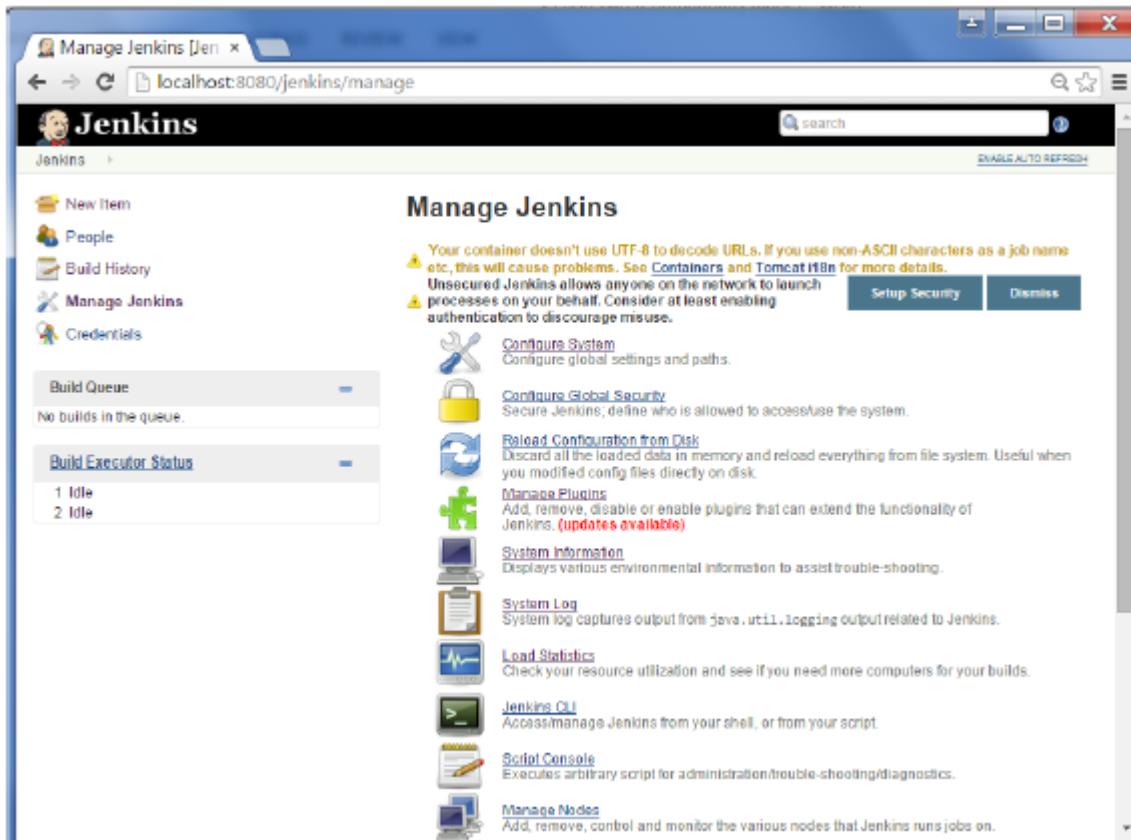
Step 2 – Find the Hudson Selenium Plugin and choose to install. Restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right is set to 'Filter: selenium'. The table lists various plugins:

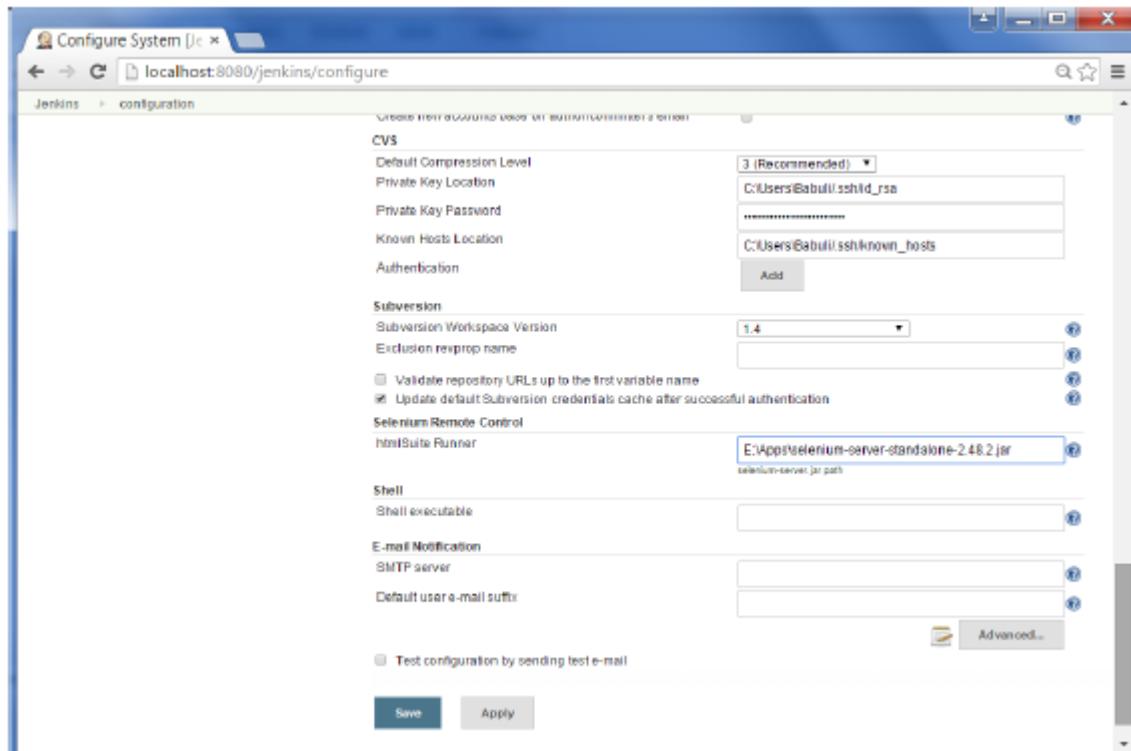
| Install | Name | Version |
|-------------------------------------|---------------------------------------|---------|
| <input type="checkbox"/> | Selenium Auto Exec Server(AES) plugin | 0.5 |
| <input checked="" type="checkbox"/> | Hudson Selenium plugin | 0.4 |
| <input type="checkbox"/> | Selenium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

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

Step 3 – Go to Configure system.



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



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

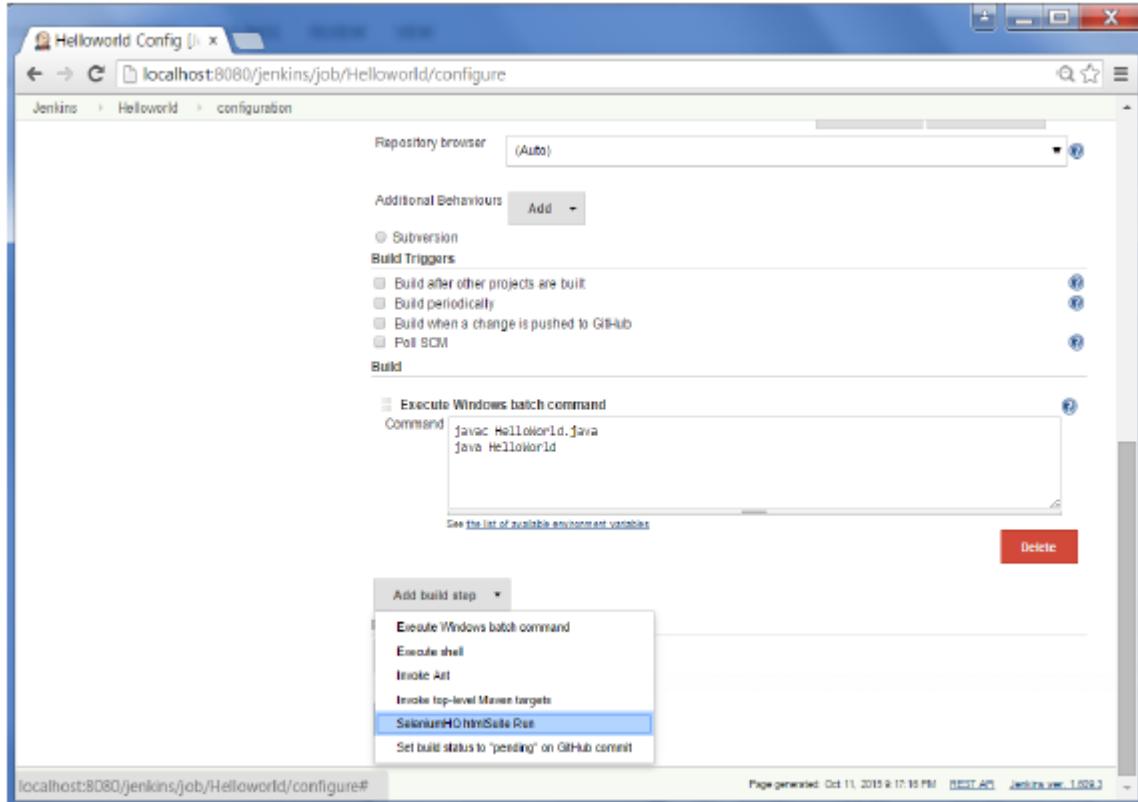
Click on the download for the Selenium standalone server.

The screenshot shows the SeleniumHQ website at www.seleniumhq.org/download/. The 'Download' tab is selected. On the left, there's a sidebar with links like 'Selenium Downloads', 'Latest Releases', 'Previous Releases', 'Source Code', and 'Maven Information'. Below that is a 'Donate to Selenium' section with a 'Donate' button and payment method icons (PayPal, Credit Card, VISA, MasterCard). Further down is a 'through sponsorship' section with a link to 'selenium project'. To the right, under 'Selenium Standalone Server', it 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 includes a 'Download version 2.48.2' link and a note about using it in a Grid configuration.

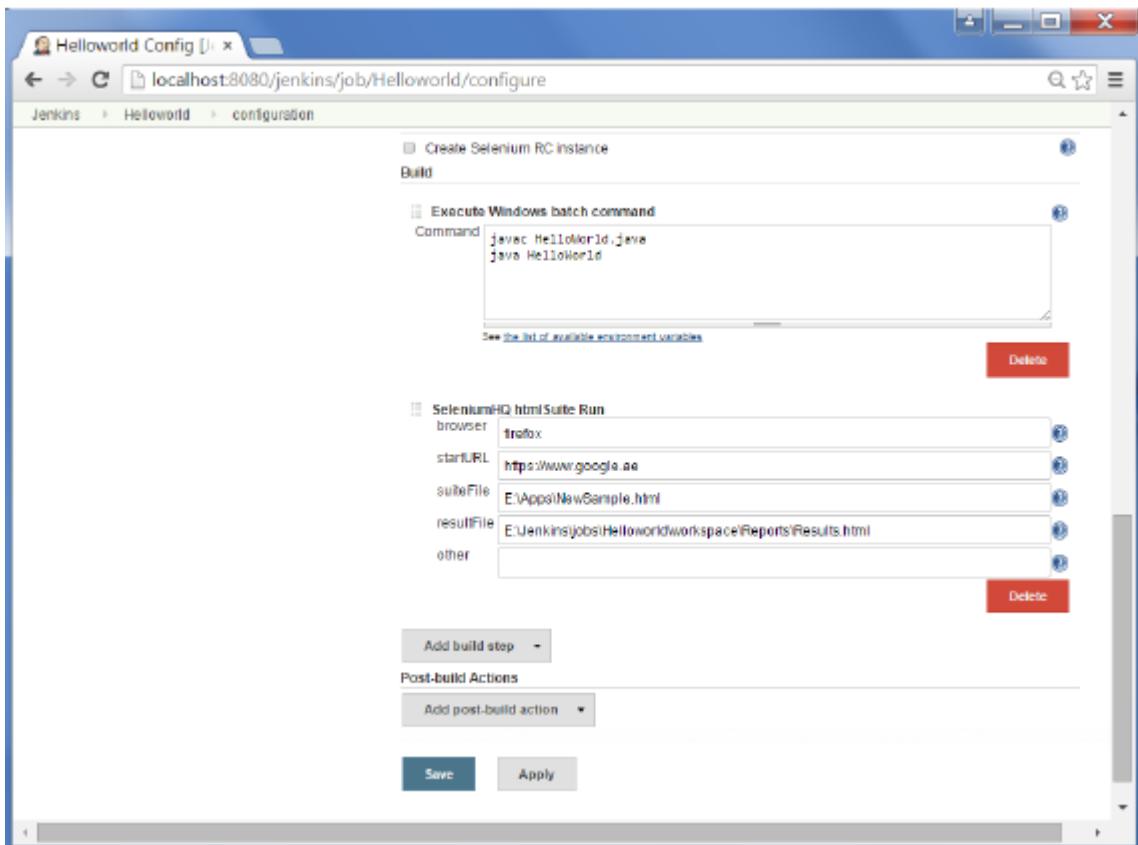
Step 5 – Go back to your dashboard and click on the Configure option for the HelloWorld project.

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The 'Jenkins' icon is selected in the sidebar. A context menu is open over the 'HelloWorld' project entry, which includes options like 'Changes', 'Workspace', 'Build Now', and 'Configure'. The 'Configure' option is highlighted with a blue border. The main table lists the 'HelloWorld' project with its last build status (23 hr - #12), last failure (23 hr - #10), and last duration (3.7 sec).

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



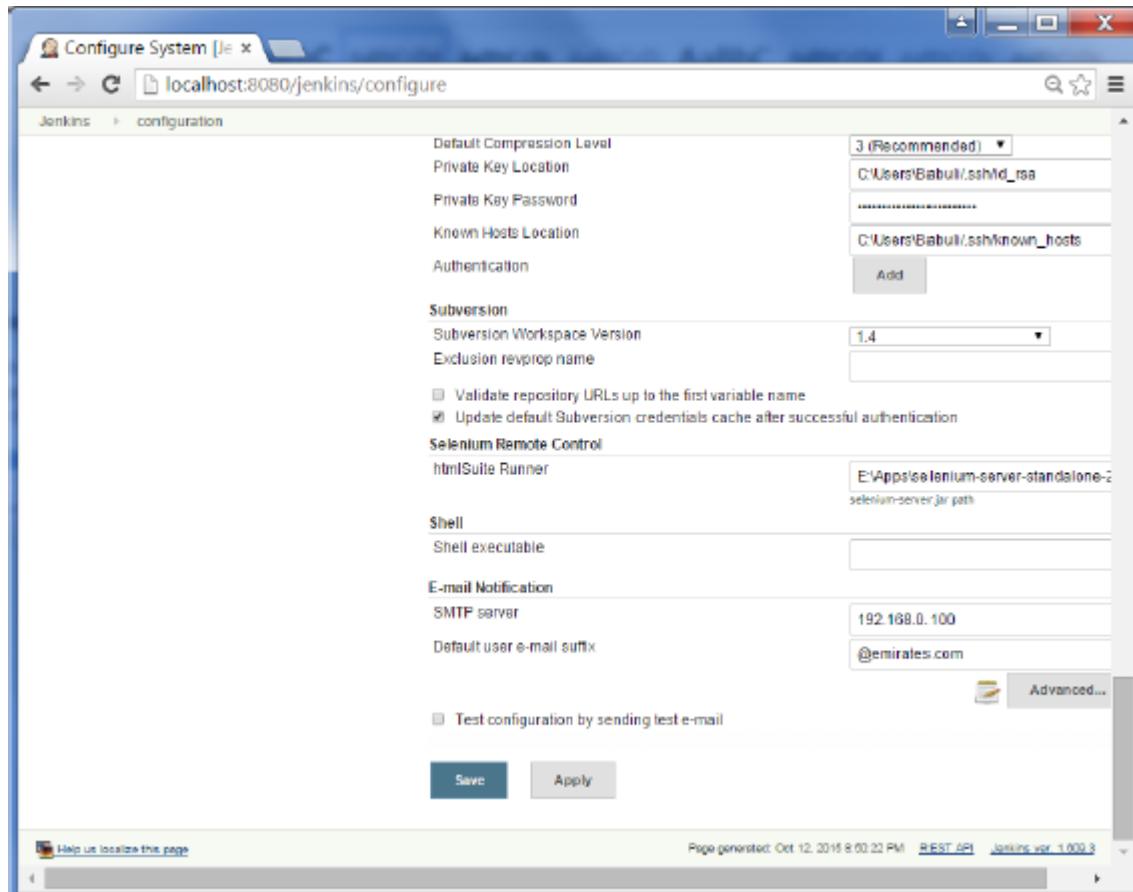
Step 7 – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



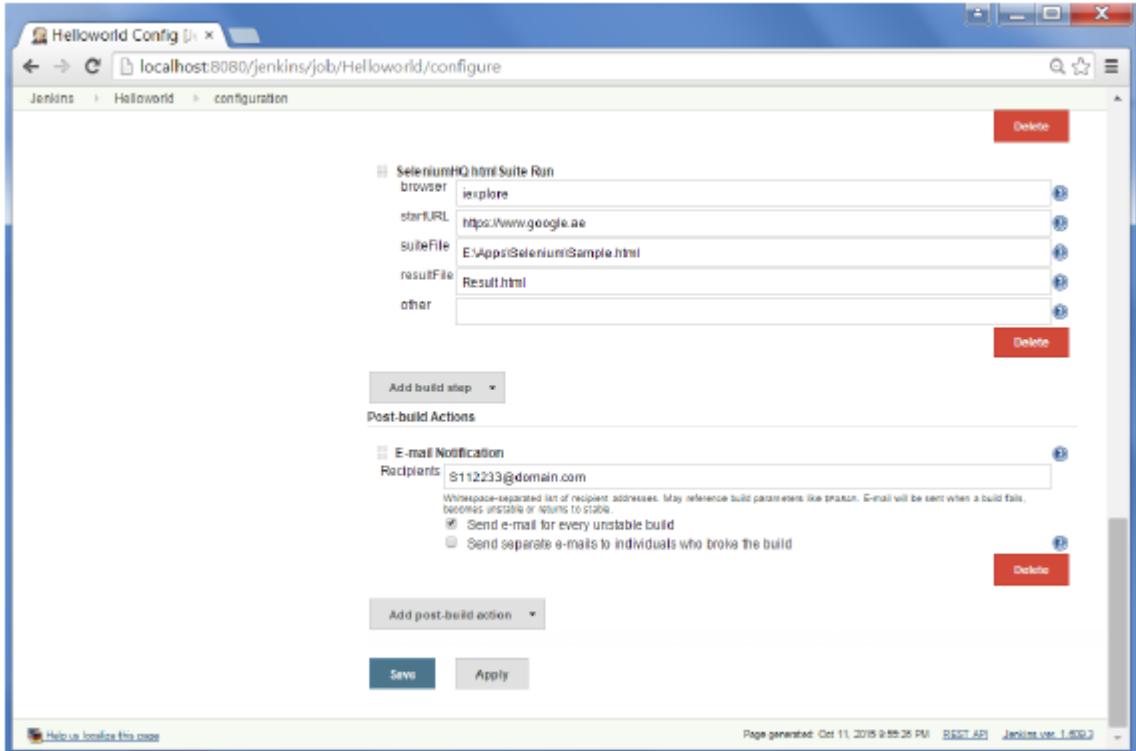
Jenkins - Notification

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

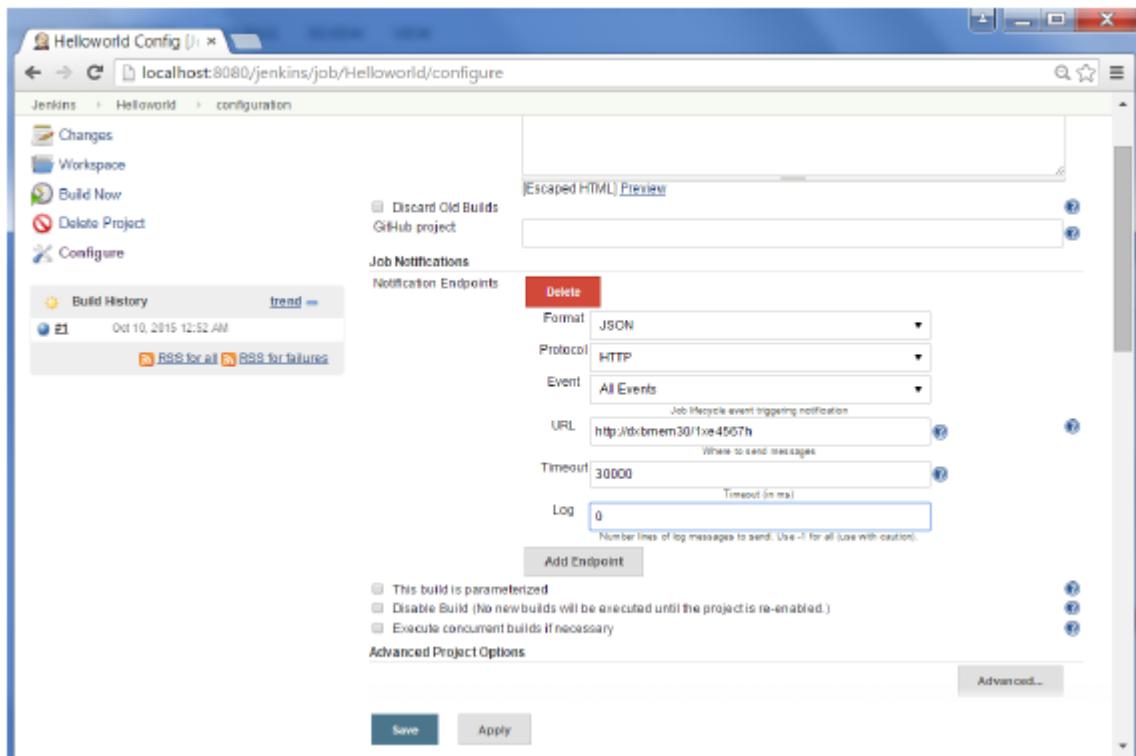
Step 1 – Configuring an SMTP server. Goto Manage Jenkins → Configure System. Go to the E-mail notification section and enter the required SMTP server and user email-suffix details.



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



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



Here are the details of each option –

"Format" – This is the notification payload format which can either be JSON or XML.

"Protocol" – protocol to use for sending notification messages, HTTP, TCP or UDP.

"Event" – The job events that trigger notifications: Job Started, Job Completed, Job Finalized or All Events (the default option).

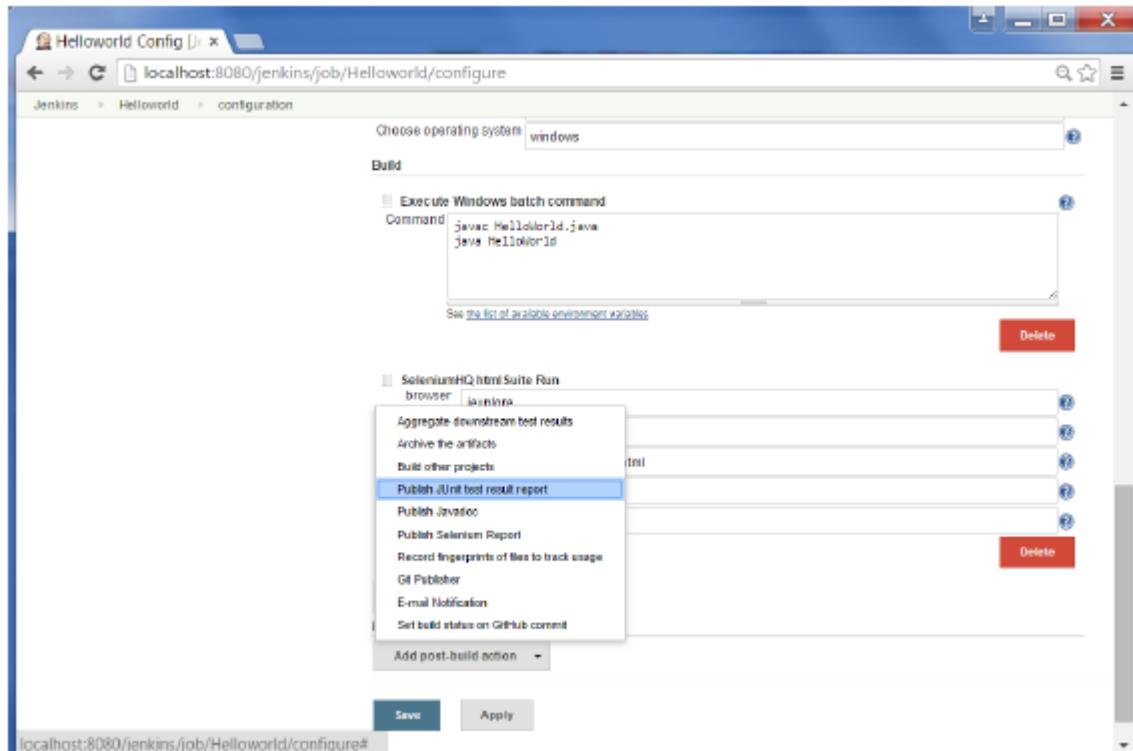
"URL" – URL to send notifications to. It takes the form of "http://host" for HTTP protocol, and "host:port" for TCP and UDP protocols.

"Timeout" – Timeout in milliseconds for sending notification request, 30 seconds by default.

Jenkins - Reporting

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

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



Jenkins - Code Analysis

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

The screenshot shows the Jenkins Static Code Analysis Plugins page. On the left, there's a sidebar with links like Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under 'Documents', there are links to Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes, and a note about waiting for wiki.jenkins-ci.org.

The main content area has a title 'Static Code Analysis Plug-ins' with a subtitle 'analysis-core'. It includes a 'Plugin Information' table with columns for Plugin ID (analysis-core), Changes (In Latest Release Since Latest Release), and Usage (analysis-core - installations). The usage chart shows a steady increase from approximately 25,000 in October 2014 to over 30,000 by September 2015. Below the table, there's a section for Source Code, Issue Tracking, Pull Requests, and Maintainer(s) (Ulli Hafner).

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

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

The plugins can provide information such as

- The total number of warnings in a job

- A showing of the new and fixed warnings of a build

- Trend Reports showing the number of warnings per build

- Overview of the found warnings per module, package, category, or type

- Detailed reports of the found warnings optionally filtered by severity (or new and fixed)

Jenkins - Distributed Builds

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

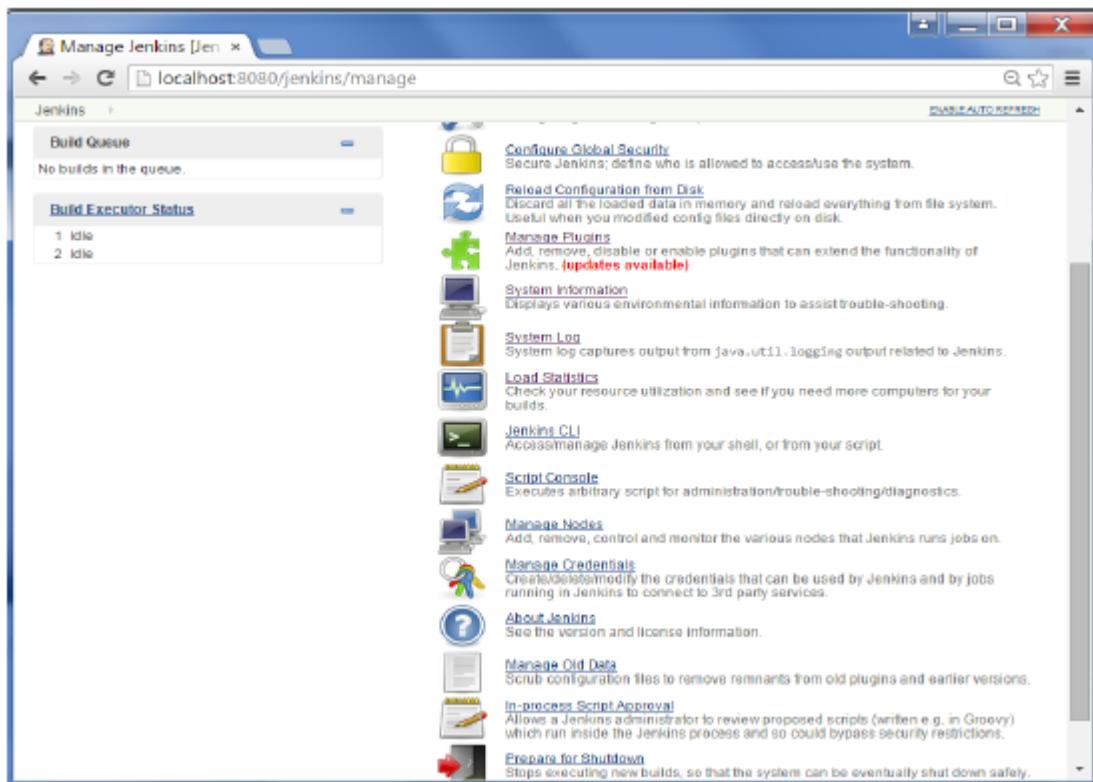
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

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

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

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



Step 2 – Click on New Node

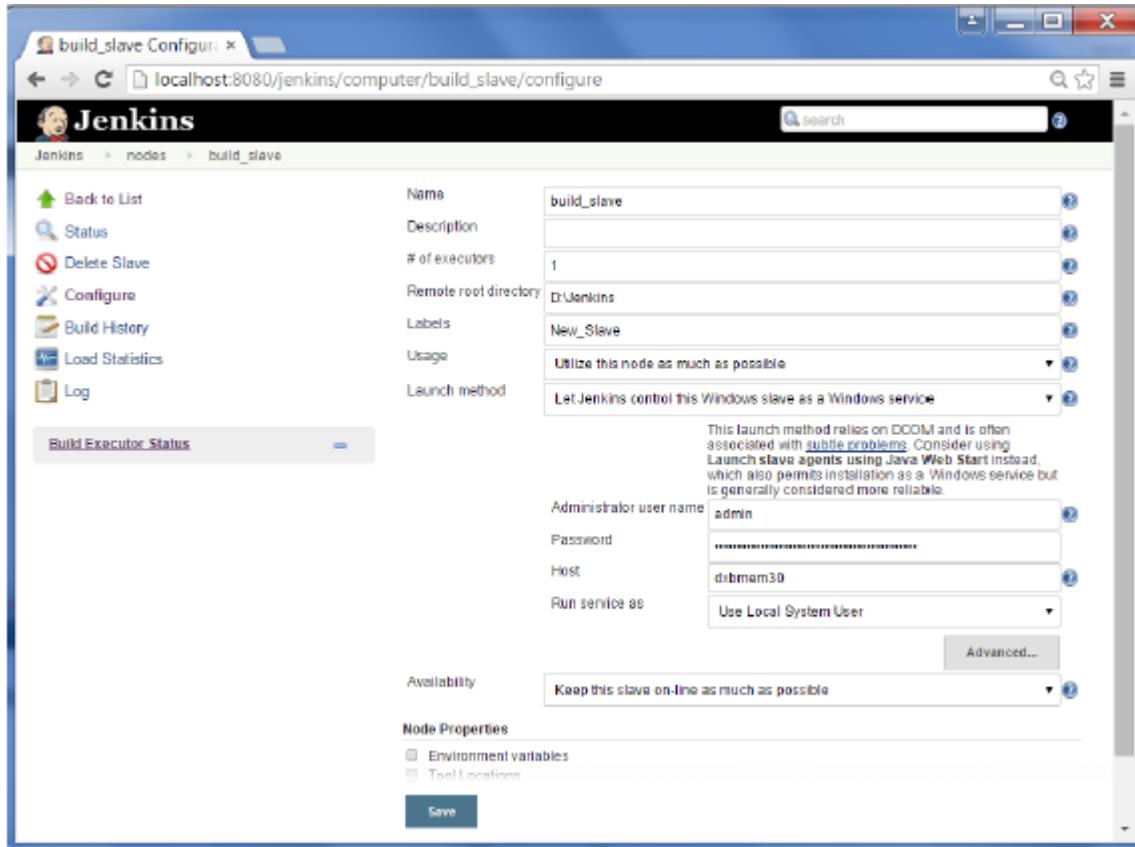
The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and an 'ENABLE AUTO REFRESH' checkbox. A table lists the nodes: one 'master' node running on Windows 7 (x86) is shown as 'In sync' with 229.89 GB free disk space and 12.13 GB free swap space. The status was last updated 3 min 11 sec ago. There are two buttons at the bottom right: 'Refresh status' and a large blue 'Add New Node' button.

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

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

Step 4 – Enter the details of the node slave machine. In the below example, we are considering the slave machine to be a windows machine, hence the option of "Let Jenkins

control this Windows slave as a Windows service" was chosen as the launch method. We also need to add the necessary details of the slave node such as the node name and the login credentials for the node machine. Click the Save button. The Labels for which the name is entered as "New_Slave" is what can be used to configure jobs to use this slave machine.



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

The screenshot shows the Jenkins interface for managing nodes. At the top, there's a navigation bar with links for 'Nodes [Jenkins]', 'localhost:8080/jenkins/computer/', 'search', and 'ENABLE AUTO REFRESH'. Below the header, the title 'Jenkins' is displayed above a breadcrumb trail: 'Jenkins > nodes >'. On the left, a sidebar contains links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. The main content area is titled 'Nodes' and shows a table with two rows. The columns are labeled 'Name', 'Architecture', 'Clock Difference', 'Free Disk Space', 'Free Swap Space', and 'Free Temp S'. The first row represents the 'build_slave' node, which is offline (indicated by a red icon) and running on Windows 7 (x86). The second row represents the 'master' node, which is online (green icon) and running on Windows 7 (x86). Below the table, there are sections for 'Build Queue' (which is empty) and 'Build Executor Status' (showing 1 idle and 2 idle executors). A 'Refresh status' button is located at the bottom right of the table area. At the very bottom of the page, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 12, 2016 9:31:43 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

Jenkins - Automated Deployment

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

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

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

| Install | Name | Version |
|-------------------------------------|--|---------|
| <input type="checkbox"/> | Artifact Deployer Plug-in 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 5.6. Also allows downloading packages from one CRX server and uploading them to one or more other CRX servers. | 1.3.2 |
| <input checked="" type="checkbox"/> | Deploy to container Plugin This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build | 1.10 |
| <input type="checkbox"/> | Deploy to WebSphere container Plugin This plugin is an extension of the Deploy Plugin . It takes a war/earfile 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 |

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

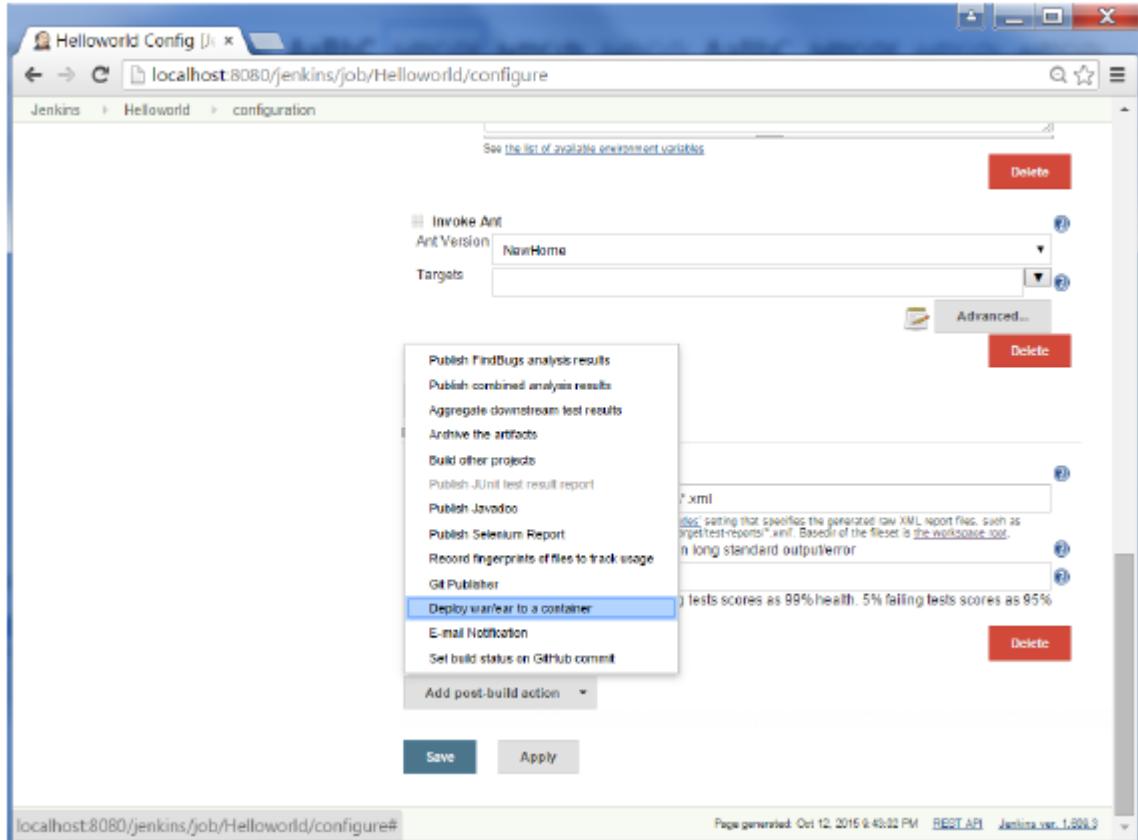
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

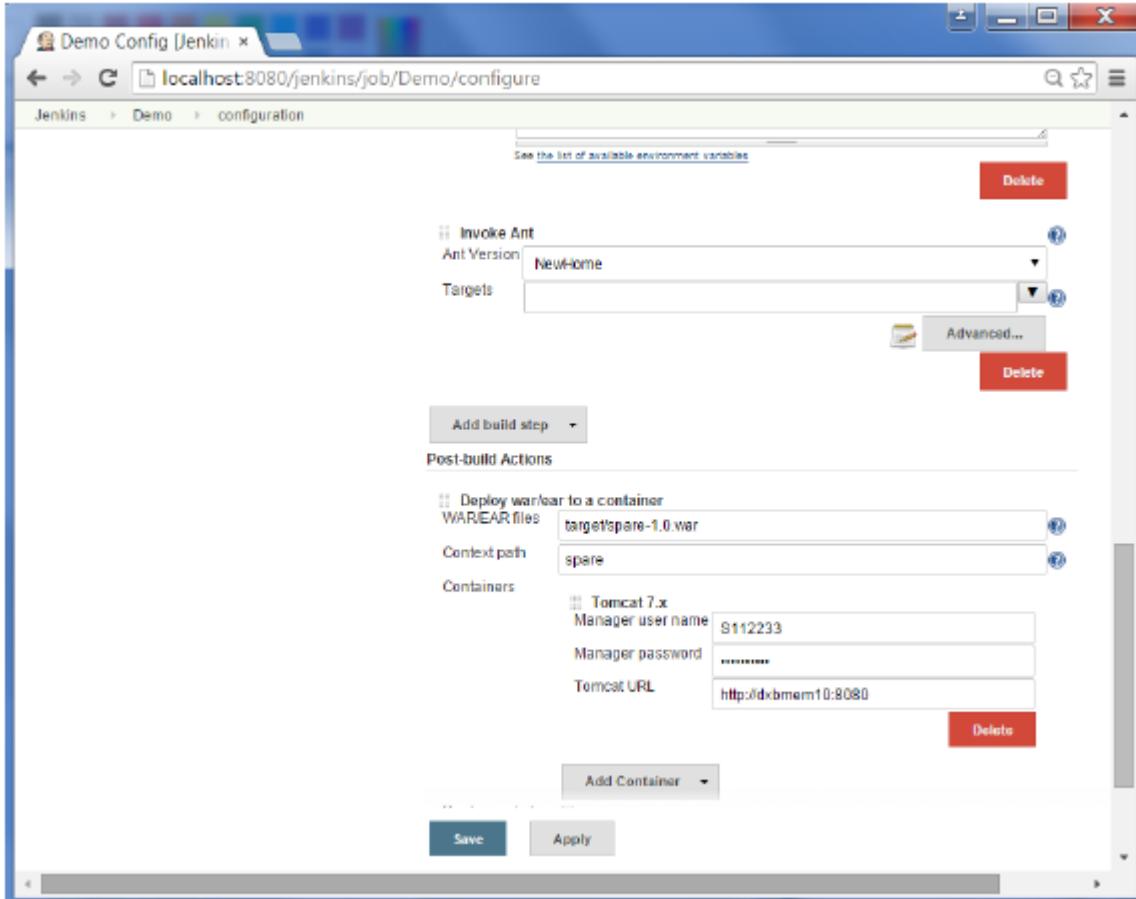
JBoss 3.x/4.x

Glassfish 2.x/3.x

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



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



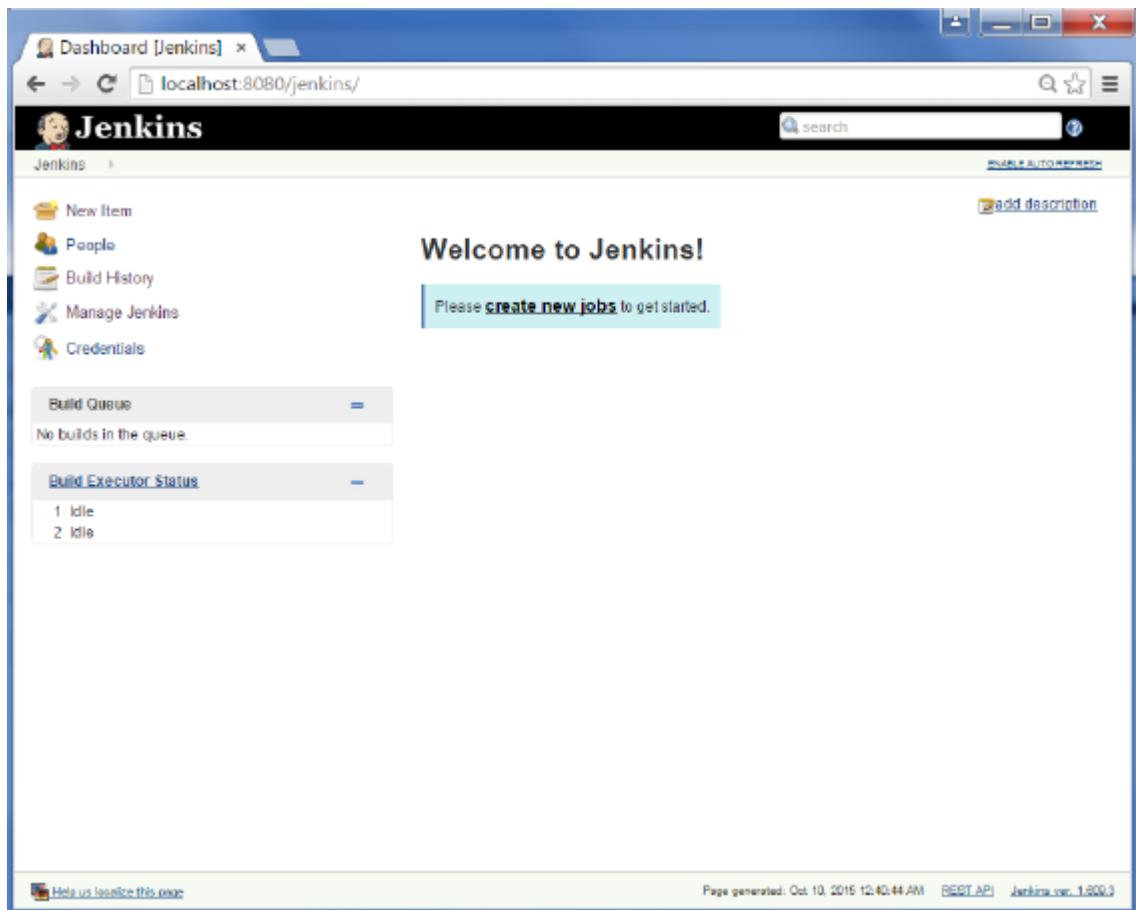
Jenkins - Metrics & Trends

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

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

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

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



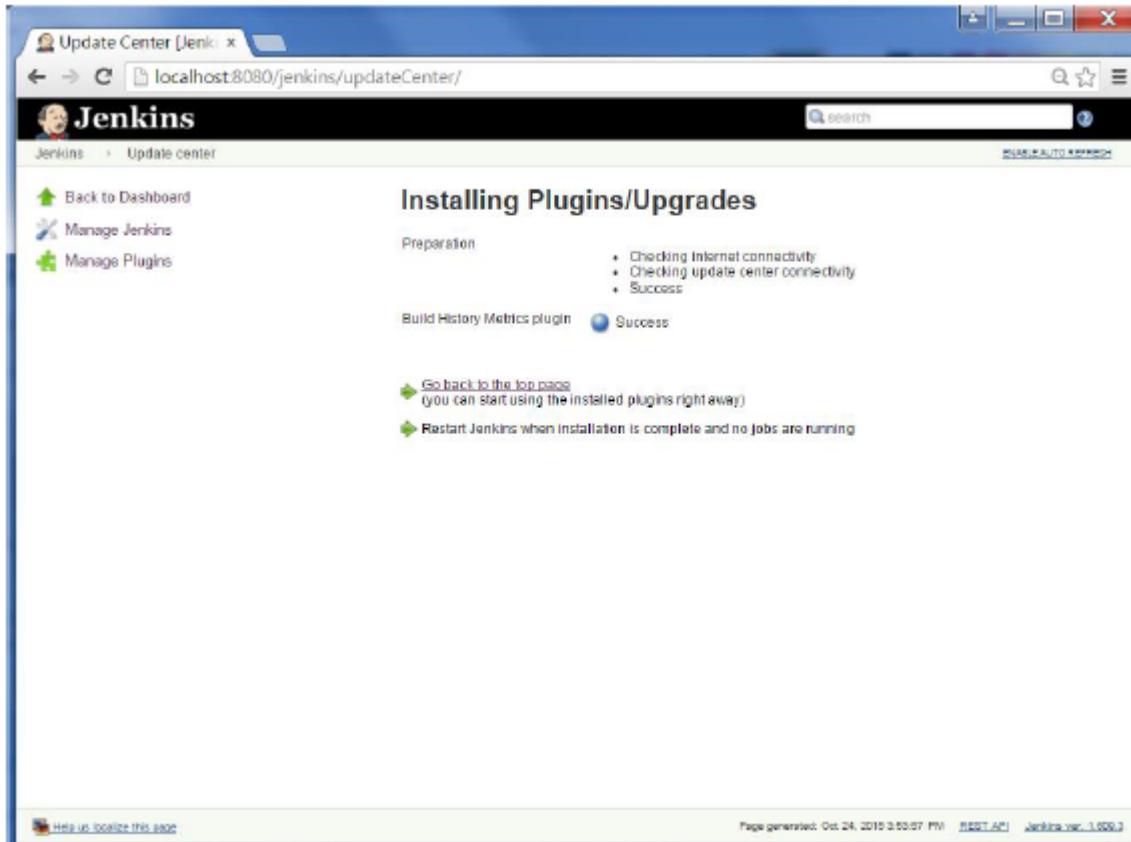
Step 2 – Go to the Manage Plugins option.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links for New Item, People, Build History, Manage Jenkins (which is selected), 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 main content area is titled "Manage Jenkins". It contains several configuration links with icons: Configure System (wrench icon), Configure Global Security (padlock icon), Reload Configuration from Disk (refresh icon), Manage Plugins (puzzle piece icon), System Information (monitor icon), System Log (document icon), Load Statistics (ECG icon), Jenkins CLI (terminal icon), Script Console (script icon), Manage Nodes (computer icon), Manage Credentials (key icon), and About Jenkins (person icon). A warning message at the top states: "Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat in 8](#) for more details." Below it says "Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse." With "Setup Security" and "Dismiss" buttons.

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 Available tab. The URL in the address bar is `localhost:8080/jenkins/pluginManager/available`. The interface includes a sidebar with Back to Dashboard and Manage Jenkins links. The main area has tabs for Updates, Available (which is selected), Installed, and Advanced. A filter bar at the top right shows the search term `build-history-metrics-plugin`. A table lists the plugin: Name is `Build History Metrics plugin`, Version is `1.2`, and the description is `Provides build metrics that encompass the history of all the runs`. At the bottom of the table are two buttons: `Install without restart` (highlighted in blue) and `Download now and install after restart`. A status message below the table 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 interface for the 'Helloworld' project. On the left, a sidebar provides links to 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. The main content area is titled 'Project Helloworld'. It features a 'Workspace' section with a folder icon, a 'Recent Changes' section with a document icon, and three tables for 'MTTR', 'MTTF', and 'Standard Deviation' metrics. Below these are 'Permalinks' to various build logs. At the bottom, there's a link to localize the page and footer information.

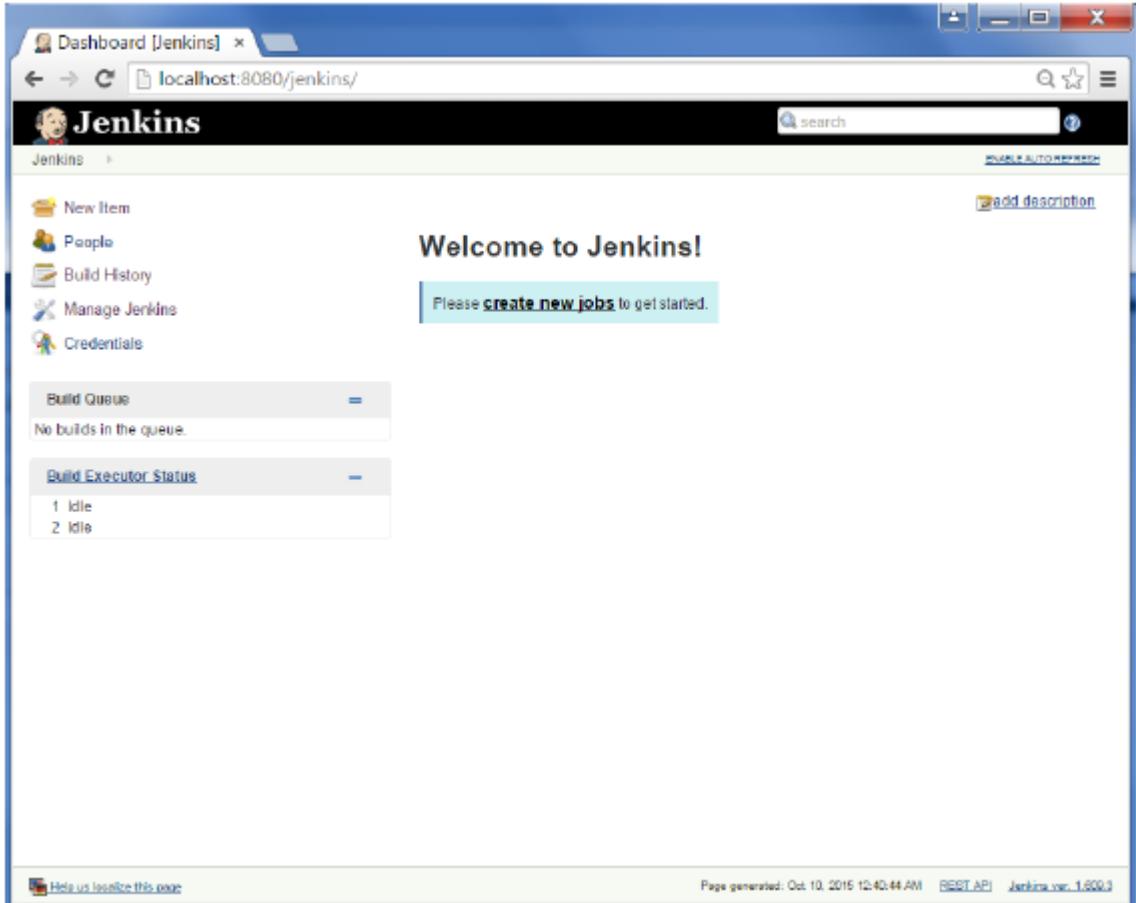
| | Last 7 Days | 0 ms |
|--------------------|--------------|-------------|
| MTTR | Last 30 Days | 23 hr |
| | All Time | 23 hr |
| | Last 7 Days | 0 ms |
| MTTF | Last 30 Days | 2 days 4 hr |
| | All Time | 2 days 4 hr |
| | Last 7 Days | 0 ms |
| Standard Deviation | Last 30 Days | 52 sec |
| | All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

Permalinks

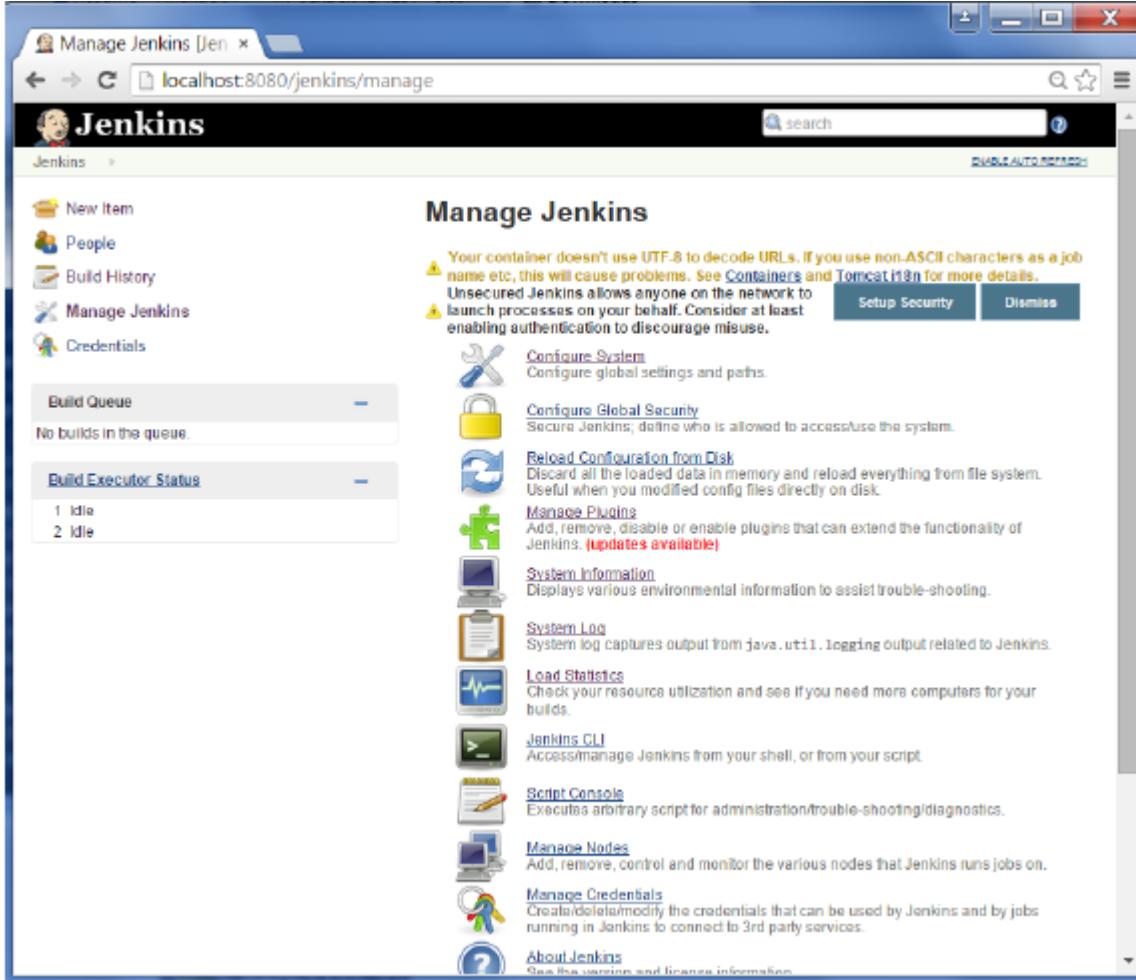
- [Last build \(#12\), 5.5 sec ago](#)
- [Last stable build \(#11\), 8 days 17 hr ago](#)
- [Last successful build \(#11\), 8 days 17 hr ago](#)
- [Last failed build \(#12\), 5.5 sec ago](#)
- [Last unstable build \(#4\), 11 days ago](#)
- [Last unsuccessful build \(#12\), 5.5 sec ago](#)

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

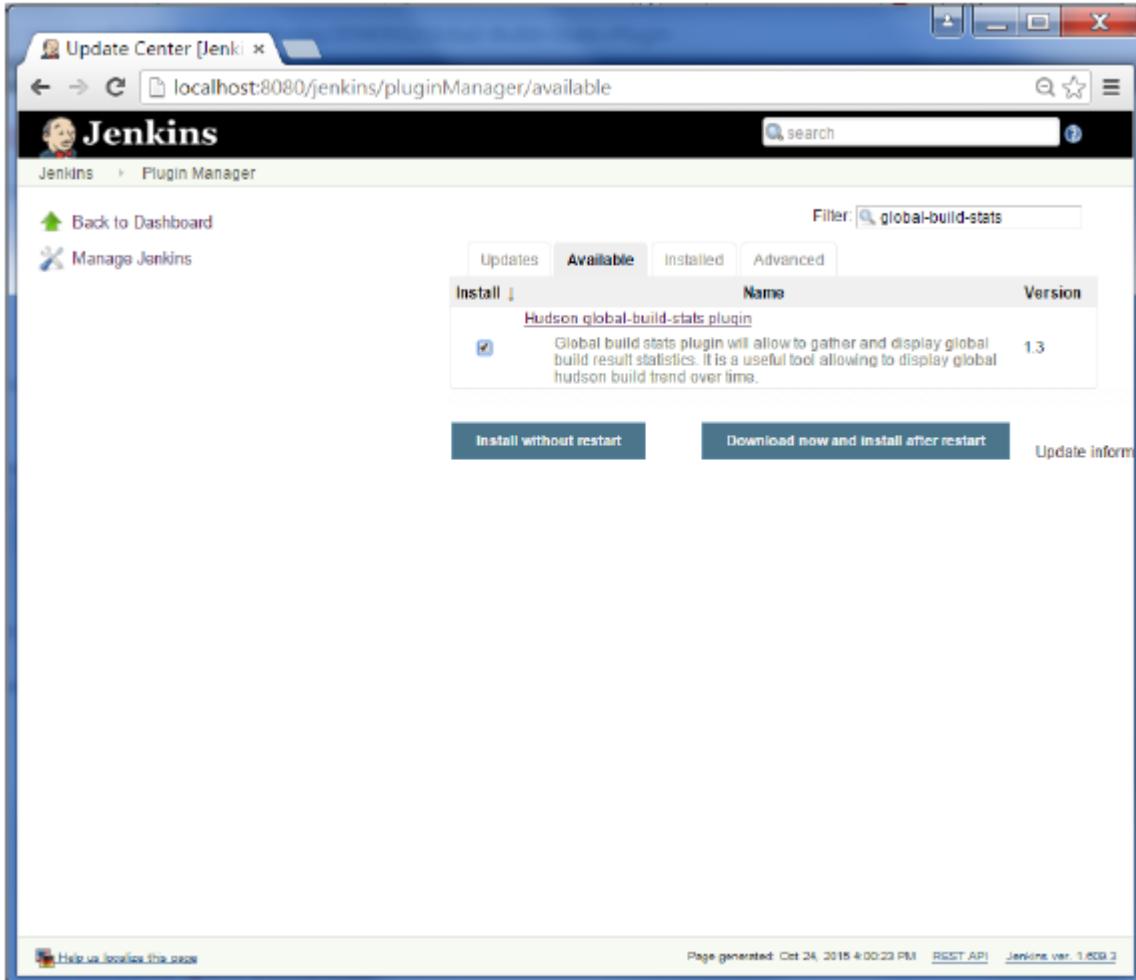
Step 1 – Go to the Jenkins dashboard and click on Manage Jenkins



Step 2 – Go to the Manage Plugins option



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 the Jenkins Update Center interface. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', and 'Manage Plugins'. The main title is 'Installing Plugins/Upgrades'. Below the title, under 'Preparation', there's a bulleted list: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. A success message for the 'Hudson global-build-stats plugin' is shown with a blue circular icon and the word 'Success'. At the bottom left, there's a link to 'Go back to the top page' with the note '(you can start using the installed plugins right away)'. At the bottom right, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 4:01:04 PM', 'REST API', and 'Jenkins ver. 1.80.3'.

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

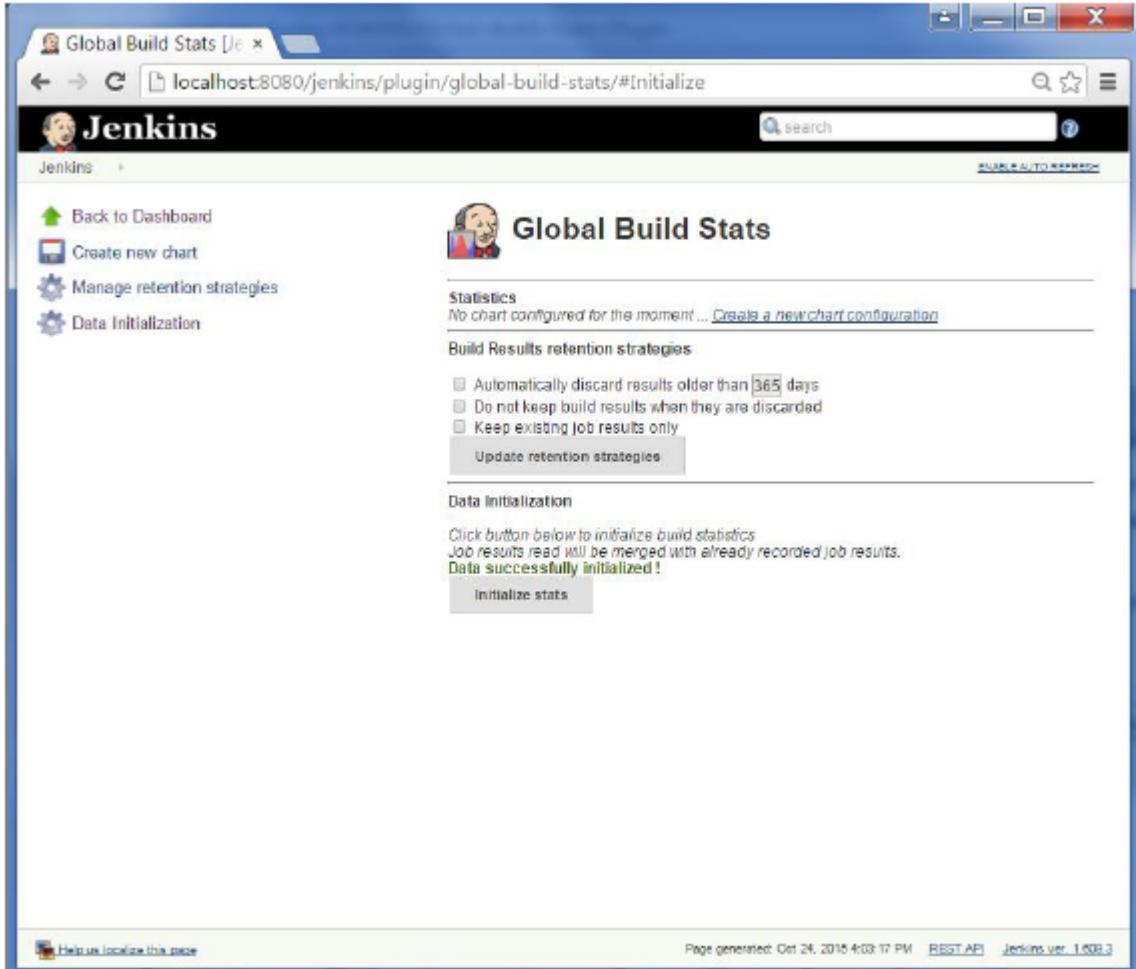
Step 5 – Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called ‘Global Build Stats’. Click on this link.



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

The screenshot shows a web browser window titled "Global Build Stats [Jenkins]" with the URL "localhost:8080/jenkins/plugin/global-build-stats/". 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", "Create new chart", "Manage retention strategies", and "Data Initialization". 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 it is a "Build Results retention strategies" section with three checkboxes: "Automatically discard results older than 30 days", "Do not keep build results when they are discarded", and "Keep existing job results only". A "Update retention strategies" button is below these checkboxes. At the bottom of the page 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 "Initialize stats" button. The footer includes links for "Help us localize this page", "Page generated: Oct 24, 2015 4:03:17 PM", "REST API", and "Jenkins ver. 1.809.3".

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



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

Title – Any title information, for this example is given as 'Demo'

Chart Width – 800

Chart Height – 600

Chart time scale – Daily

Chart time length – 30 days

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

Global Build Stats | Jenkins

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

Global Build Stats

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

Adding new chart

Title : Demo

Chart Width * Height : 800 * 600

Chart time scale : Daily

Chart time length : 30 days

Filters :

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

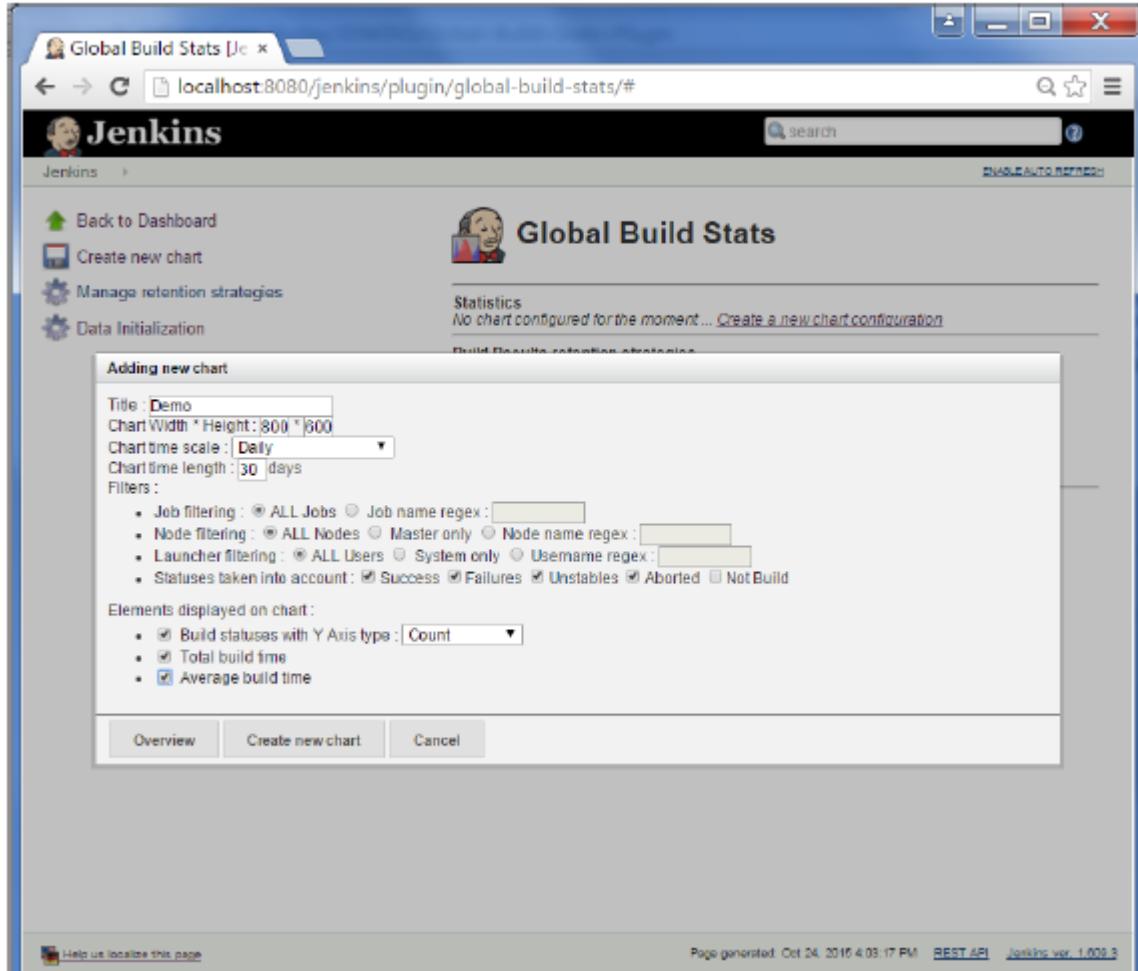
Elements displayed on chart:

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

Overview Create new chart Cancel

Help us localize this page

Page generated: Oct 24, 2016 4:03:17 PM [REST API](#) Jenkins ver. 1.609.3



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



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

The screenshot shows the Jenkins Global Build Search interface. At the top, there are navigation links for 'Back to Dashboard' and 'Back to Global Build Stats'. Below that is a search bar with placeholder text 'Search...'. Underneath the search bar is a 'Filters' section with several dropdown menus and checkboxes. The table below has columns for 'Status', 'Job name', '#', 'Date', 'Duration', 'Master name', and 'LastBuild # & log'. There are three entries in the table:

| Status | Job name | # | Date | Duration | Master name | LastBuild # & log |
|---------|-----------|---|--------------------------|----------|-------------|-------------------|
| Failure | Hellosend | 2 | Oct 11, 2015 19:04:57 PM | 5.6 sec | master | SH1STDR |
| Failure | Hellosend | 2 | Oct 11, 2015 19:51:37 PM | 4.6 sec | master | SH1STDR |
| Failure | Hellosend | 2 | Oct 11, 2015 19:48:21 PM | 4.2 sec | master | SH1STDR |

Jenkins - Server Maintenance

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

URL Options

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

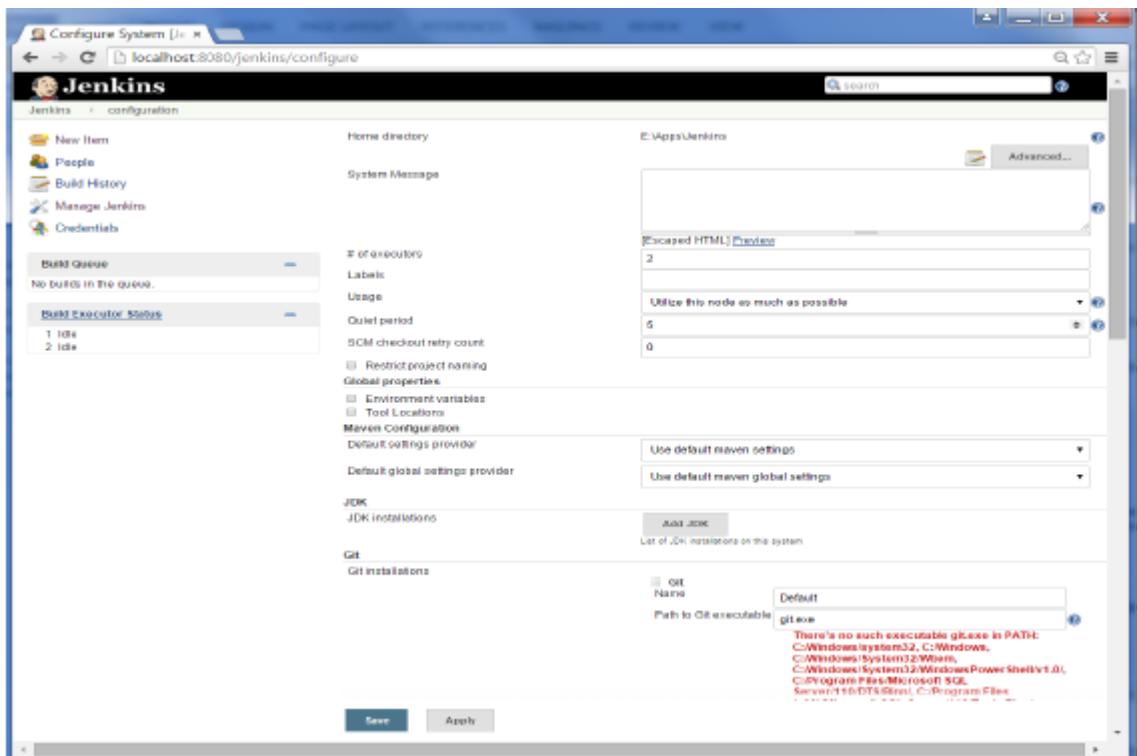
http://localhost:8080/jenkins/exit – shutdown jenkins

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

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

Backup Jenkins Home

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

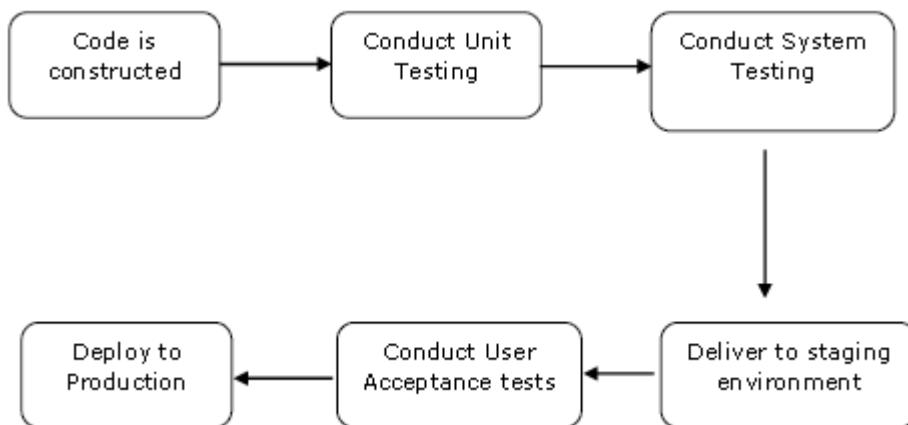


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

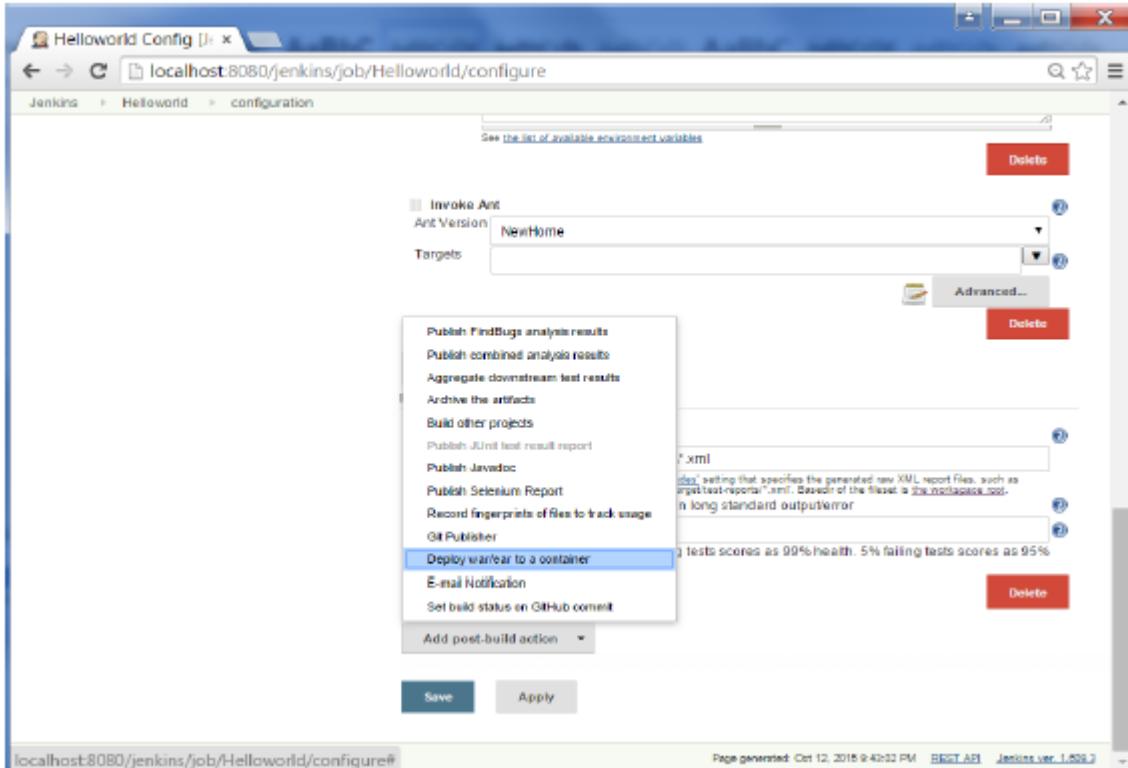
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

Jenkins - Continuous Deployment

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



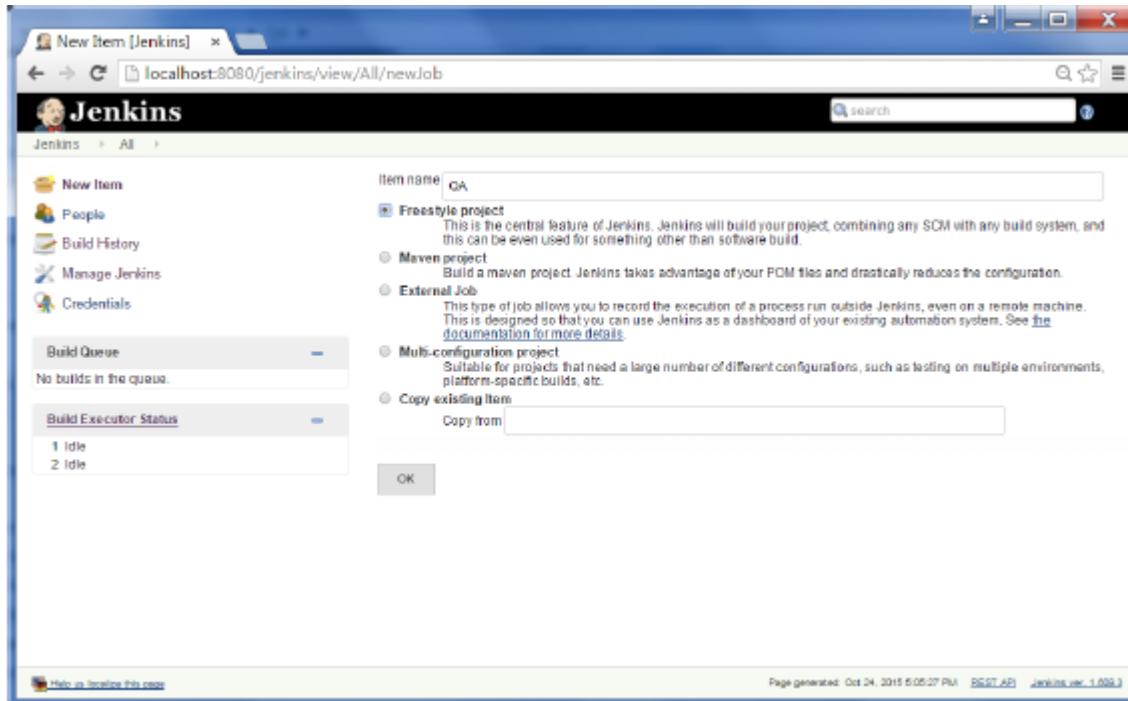
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



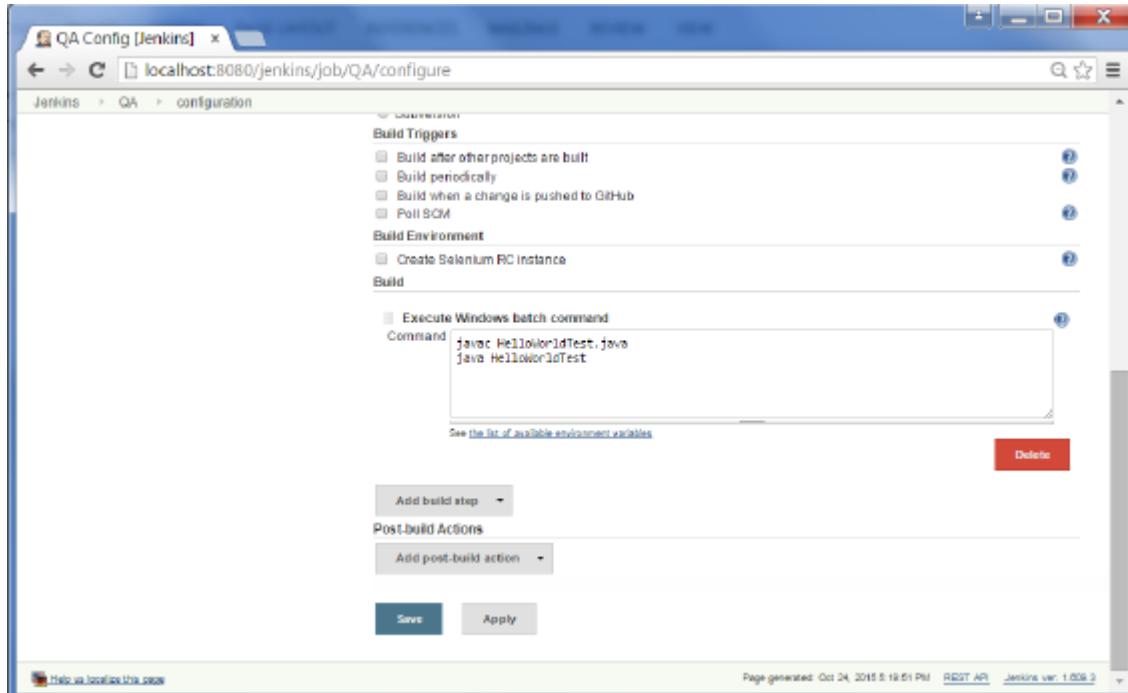
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

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

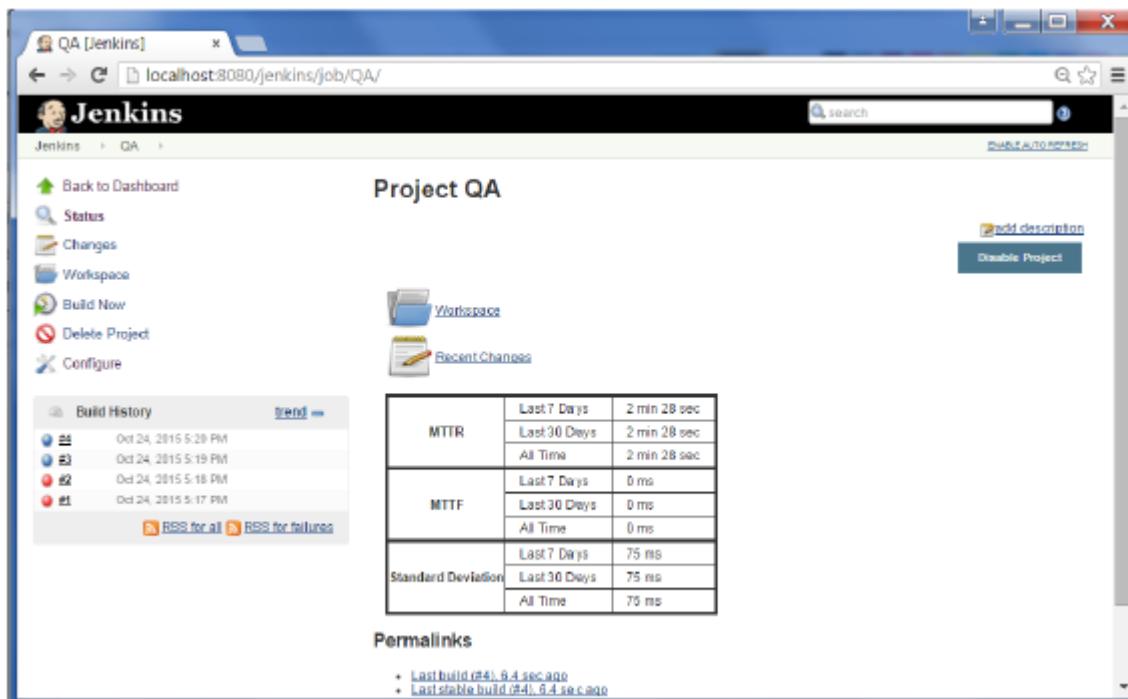
Step 1 – Go to the Jenkins dashboard and click on New Item. Choose a 'Freestyle project' and enter the project name as 'QA'. Click on the Ok button to create the project.



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



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



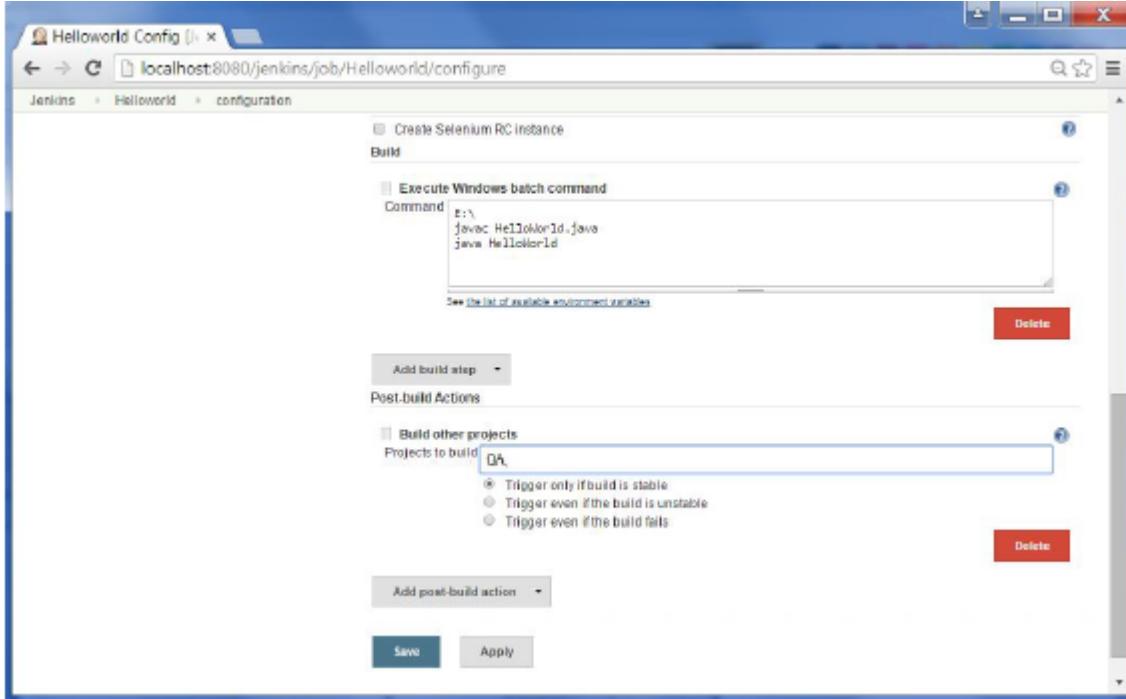
Step 3 – Now go to you Helloworld project and click on the Configure option

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' (1 idle). The main area displays a table of jobs. The 'Helloworld' job is listed, showing its last success was 12 days ago (build #15), its last failure was 12 days ago (build #14), and its duration was 6.6 sec. It has icons for 'Changes', 'Workspace', 'Build Now', and 'Delete Project'. A 'Configure' button is highlighted with a blue border. At the bottom, there are links for 'RSS for failures' and 'RSS for just latest builds'.

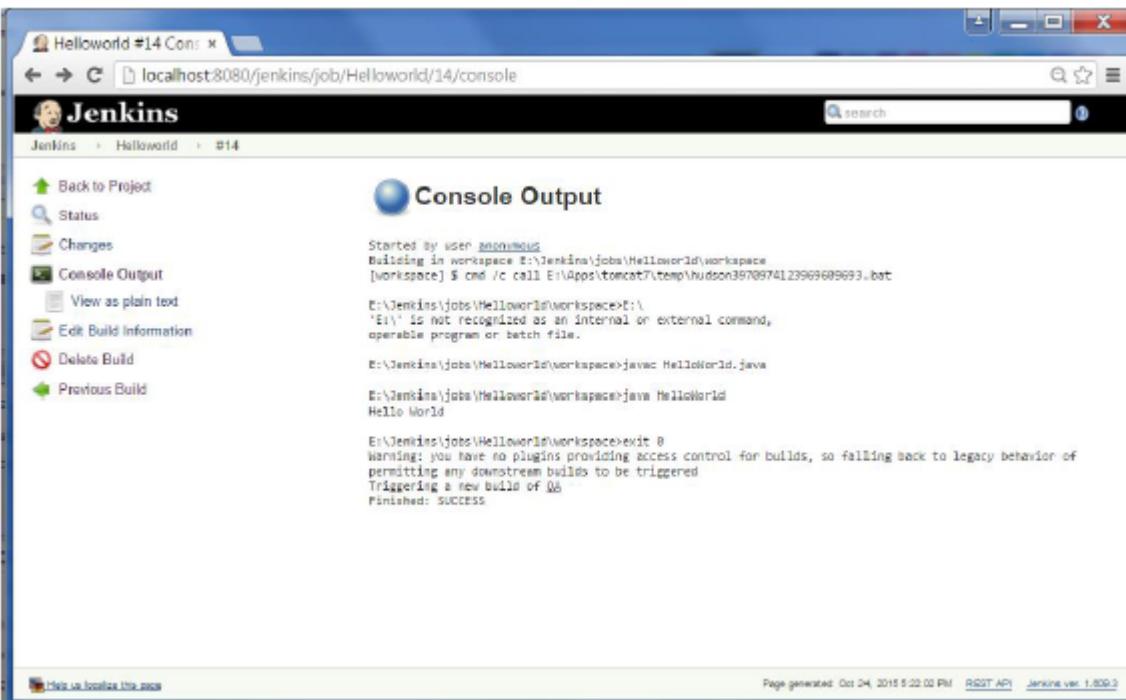
Step 4 – In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

The screenshot shows the 'Helloworld' project configuration page at localhost:8080/jenkins/job/Helloworld/configure. The 'Build other projects' option is highlighted with a blue selection bar. Other options shown include 'Aggregate downstream test results', 'Archive the artifacts', 'Publish JUnit test result report', 'Publish Javadoc', 'Record fingerprints of files to track usage', 'Git Publisher', and 'E-mail Notification'. Below the configuration area are 'Save' and 'Apply' buttons. The footer includes links for 'Help us localize this page', 'Page generated: Oct 25, 2015 9:58:29 AM', 'REST API', and 'Jenkins ver. 1.609.3'.

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.



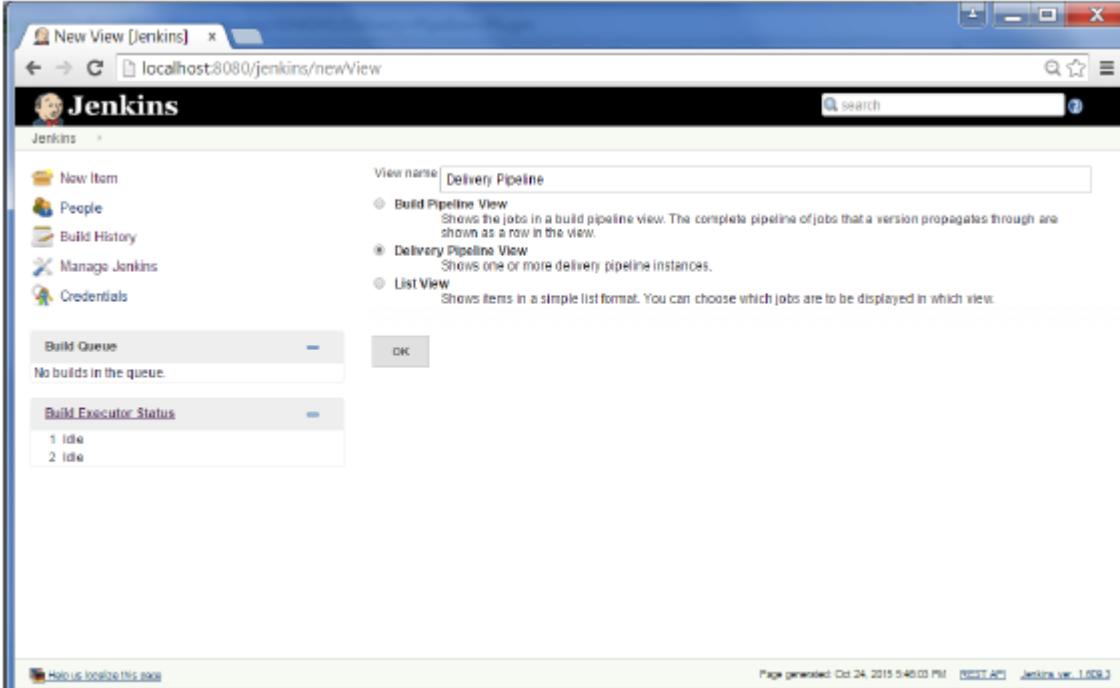
Step 7 – Let now install the Delivery pipeline plugin. Go to Manage Jenkins → Manage Plugins. In the available tab, search for 'Delivery Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The 'available' tab is selected. A list of plugins is displayed with their names, descriptions, and versions. One plugin, 'Delivery Pipeline Plugin', has a checked checkbox next to it, indicating it is selected for installation. Other visible plugins include 'ontrack Jenkins plug-in', 'Fail The Build Plugin', 'Runscope plugin', 'Build Graph View Plugin', 'Deployment Pipeline', 'CloudBees Docker Hub Notifications', 'Seed Jenkins plug-in', and 'Delivery Pipeline Plugin' (which is the one currently selected).

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. On the left, there is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below the sidebar, there are two sections: 'Build Queue' (which is empty) and 'Build Executor Status' (which shows 1 Idle and 2 Idle executors). The main area is titled 'All' and contains a table of build jobs. The table has columns: S, W, Name, Last Success, Last Failure, and Last Duration. Two rows are listed: 'HelloWorld' and 'DB'. Both rows have a green circular icon next to them, indicating success. The 'Last Success' column shows '25 min - #14' for HelloWorld and '25 min - #5' for DB. The 'Last Failure' column shows '1 hr 40 min - #12' for HelloWorld and '28 min - #2' for DB. The 'Last Duration' column shows '1.4 sec' for both. At the bottom of the table, there is a legend: 'Icon: S M L', 'Legend: RSB for all', 'RSB for failures', and 'RSB 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

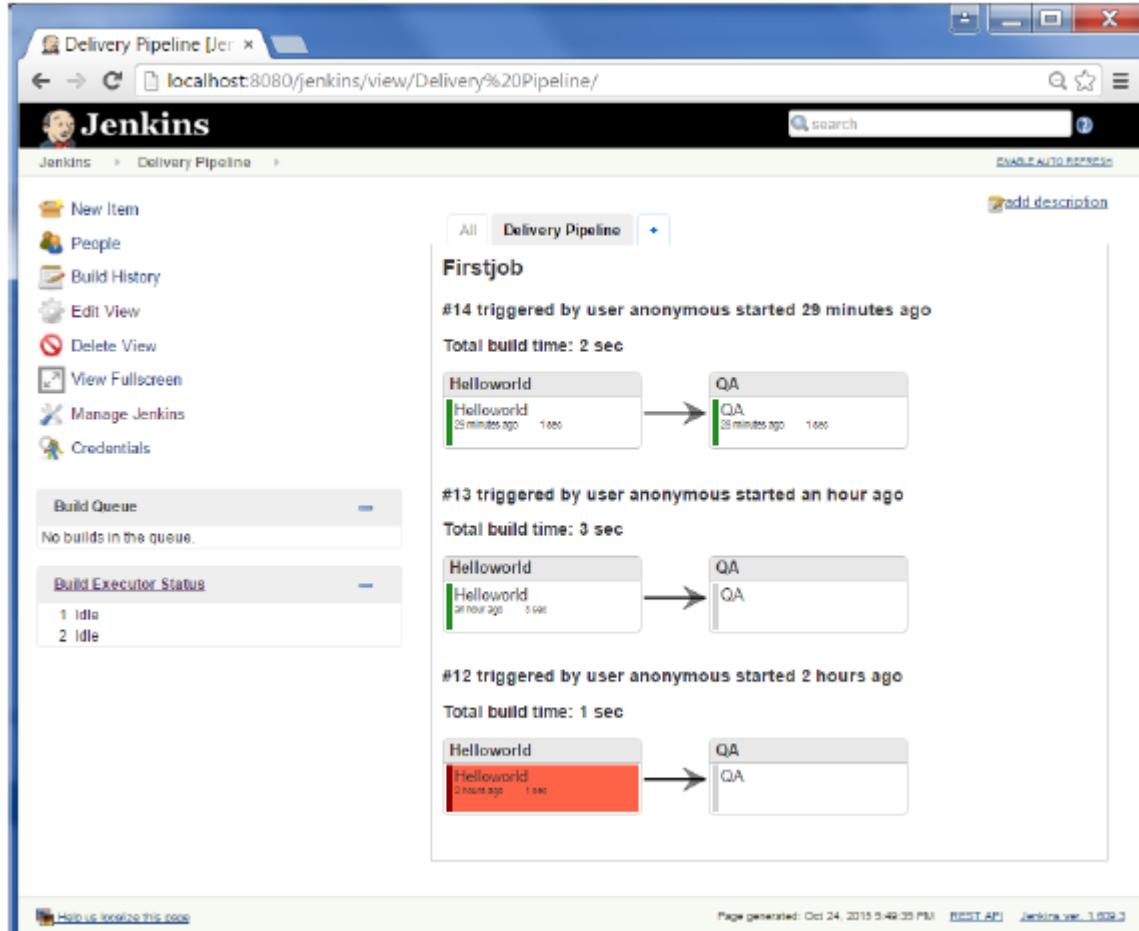
Click the OK button.

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

- Name:** Delivery Pipeline
- View settings:**
 - Number of pipeline instances per pipeline: 3
 - Display aggregated pipeline for each pipeline: checked
 - Number of columns: 1
 - Sorting: None
 - Update interval: 2
 - 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: (empty)
 - URL for custom CSS file (fullscreen): (empty)
- Pipelines:**
 - Components:
 - Name: FirstJob (Delete)
 - Initial Job: HelloWorld
 - Final Job (optional): (empty)
 - Add
 - Add
- Regular Expression:** (empty)

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.



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 'Available' tab is selected. A search bar at the top right contains the text 'Build pipeline'. Below the tabs, there is a table with columns: Name and Version. The table lists several plugins:

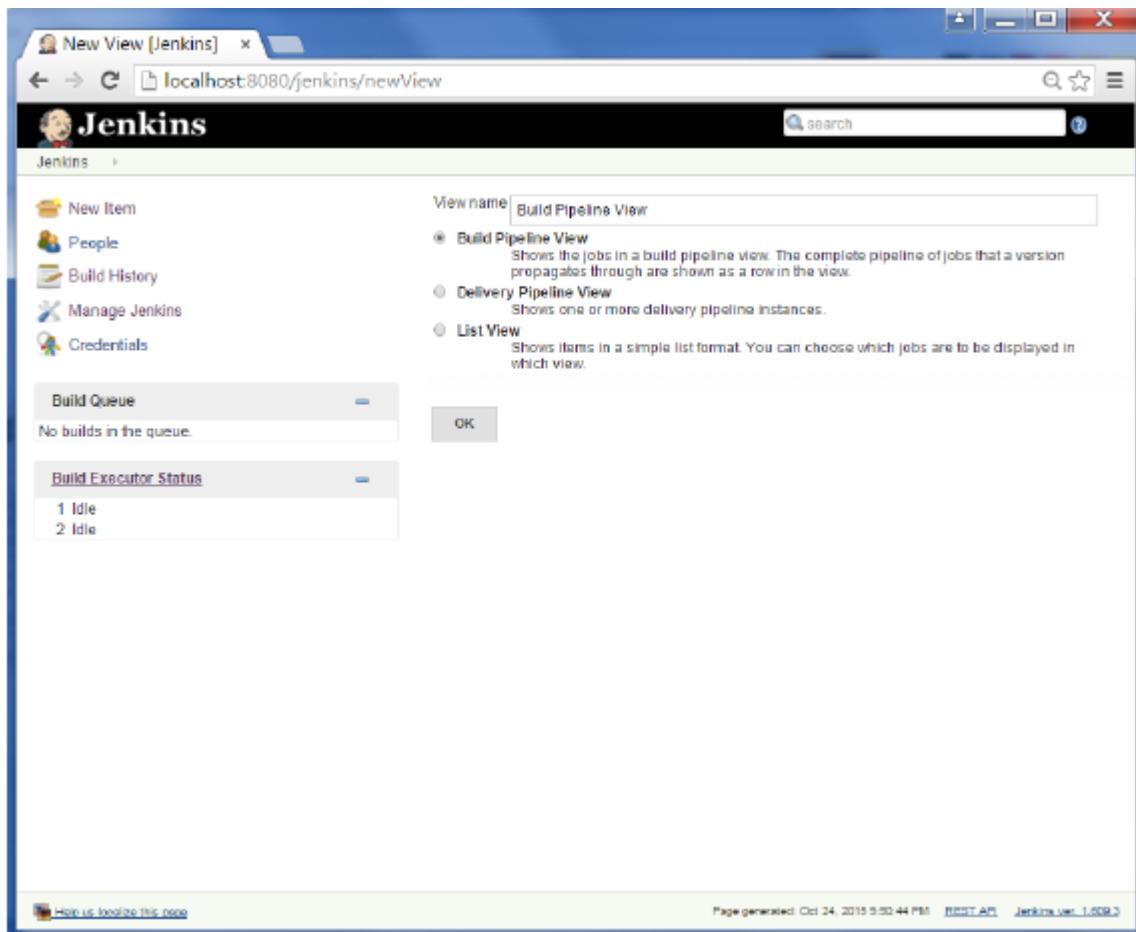
| | Name | Version |
|-------------------------------------|--|---------|
| <input checked="" type="checkbox"/> | Build Pipeline Plugin | 1.4.8 |
| <input type="checkbox"/> | Fail The Build Plugin | 1.0 |
| <input type="checkbox"/> | Runscope plugin | 1.44 |
| <input type="checkbox"/> | Build Graph View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

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

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

The screenshot shows the Jenkins Dashboard. On the left, there is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main area has a table with columns: #, W, Name, Last Success, Last Failure, and Last Duration. It lists two items: 'HelloWorld' and 'QA'. At the bottom of the table, there is a legend with three items: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'. The 'All' tab is highlighted with a blue background.

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 configuration interface for a 'Build Pipeline View'. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Edit View', 'Delete View', '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 right panel has tabs for 'Name' (Build Pipeline View), 'Description', 'Filter build queue', 'Filter build executors', 'Build Pipeline View Title', and 'Layout'. Under 'Layout', it says 'Based on upstream/downstream relations' with a note: 'This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.' It includes a dropdown for 'Select Initial Job' set to 'HelloWorld', a dropdown for 'No Of Displayed Builds' set to '1', and several radio button groups for various pipeline-related settings. 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 dashboard for the 'Build Pipeline View'. It features a header with the Jenkins logo and the title 'Build Pipeline View'. Below the header is a toolbar with icons for New Item, Help, Configure, Add Job, Delete, Create, and Manage. The main area is titled 'Build Pipeline' and displays three stages of the pipeline: 'Pipeline #14' (grey box), '#14 HelloWorld' (green box), and '#5 QA' (green box). Each stage has a timestamp: '02/24/2016 3:13 PM', '02/24/2016 3:13 PM', and '02/24/2016 3:13 PM' respectively. The green boxes indicate successful builds.

Jenkins - Managing Plugins

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

The screenshot shows the Jenkins Plugins page at <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The left sidebar 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:

| |
|--|
| 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

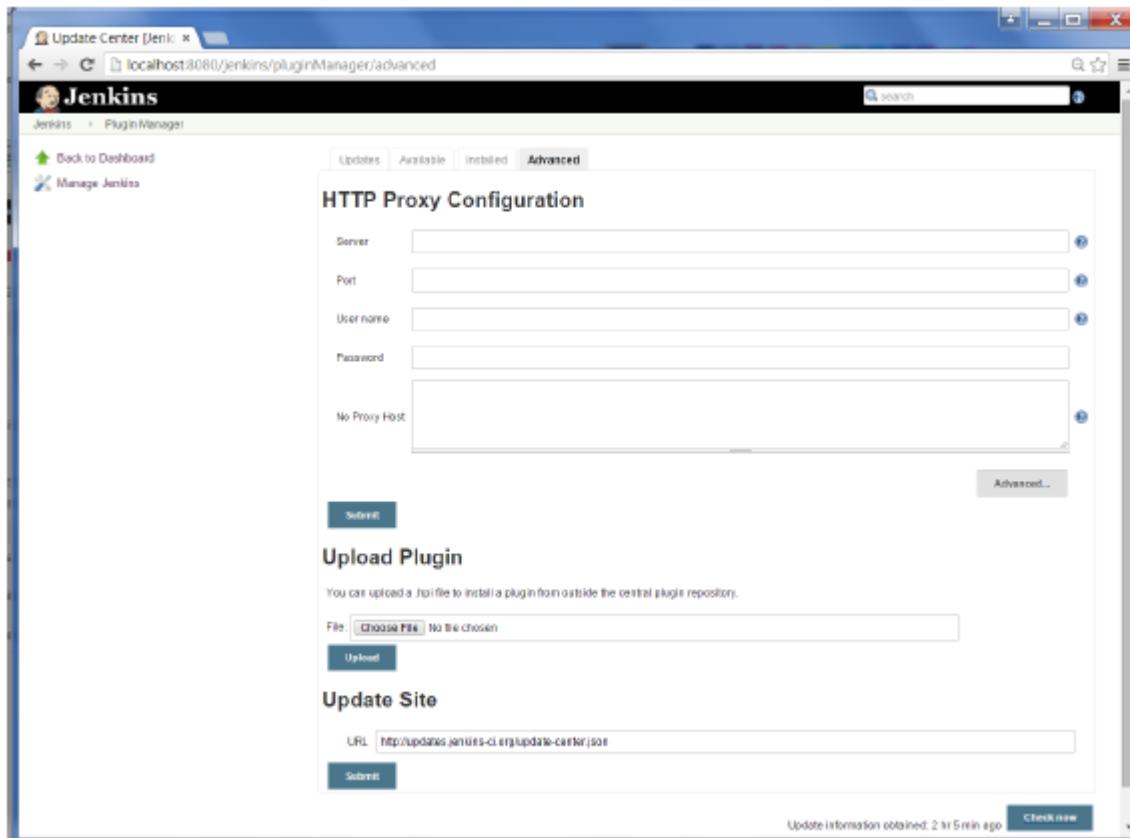
To uninstall a plugin, Go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.

The screenshot shows the Jenkins Plugin Manager at <http://localhost:8080/jenkins/pluginManager/installed>. The left sidebar has 'Back to Dashboard' and 'Manage Jenkins'. The main content area shows the 'Installed' tab of the plugin manager. A table lists installed plugins:

| Enabled | Name | Version | Previously installed version | Pinned | Uninstall |
|---------|----------------------------------|---------|------------------------------|-----------------------|---------------------------|
| ✓ | Ant Plugin | 1.2 | | | Uninstall |
| ✓ | Built History Metrics Plugin | 1.2 | | | Uninstall |
| ✓ | Build Pipeline Plugin | 1.4.8 | | | Uninstall |
| ✓ | Credentials Plugin | 1.22 | Downgrade to 1.18 | Unpin | Uninstall |
| ✓ | CVS Plugin | 2.13 | | | Uninstall |
| ✓ | Delivery Pipeline Plugin | 0.8.7 | | | Uninstall |
| ✓ | Deploy to container Plugin | 1.10 | | | Uninstall |
| ✓ | External Monitor Job Type Plugin | 1.4 | | | Uninstall |
| ✓ | External Notification Plugin | 1.1 | | | Uninstall |
| ✓ | FindBugs Plugin | 4.62 | | | Uninstall |

Installing another Version of a Plugin

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

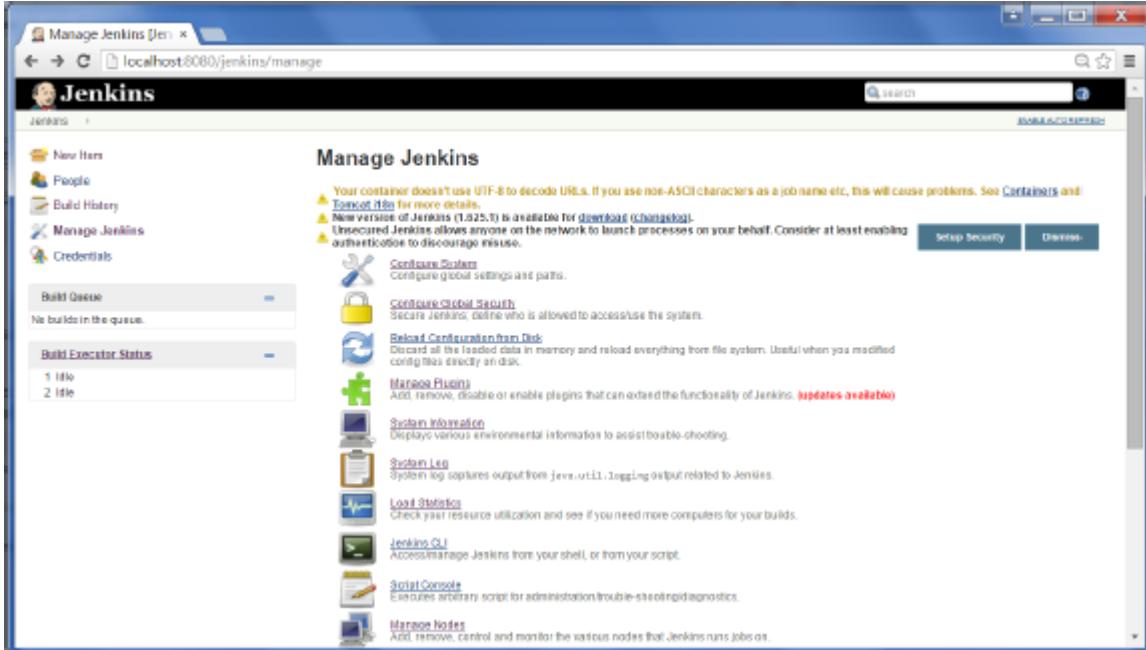


Jenkins - Security

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

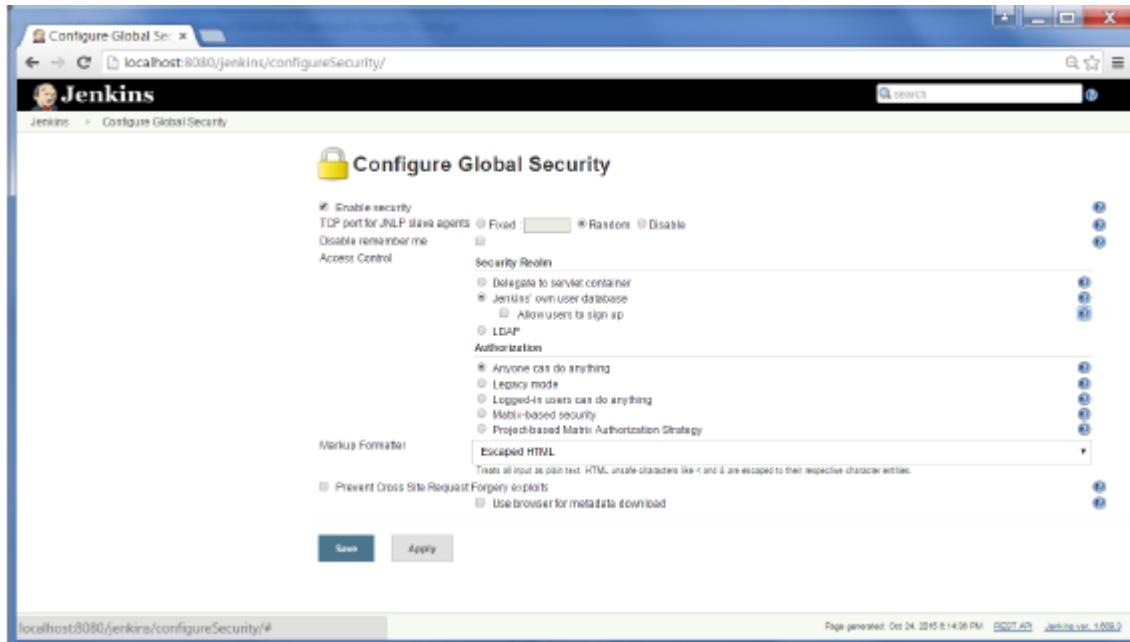
To configure Security in Jenkins, follow the steps given below.

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

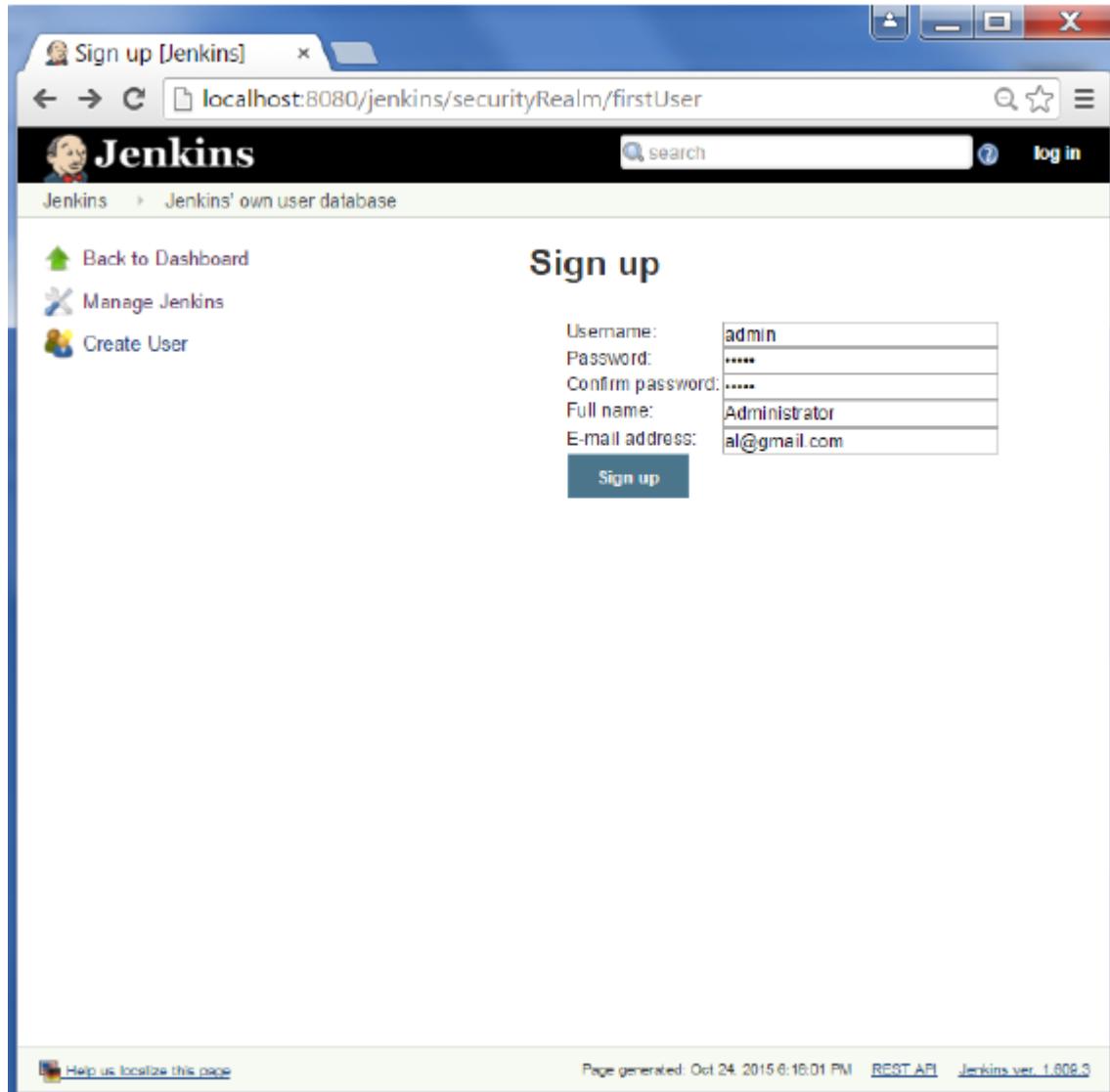


Step 2 – Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain it's own database of users, so in the Security Realm, choose the option of 'Jenkins' own user database'.

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 the Jenkins 'Sign up' page. At the top, there's a navigation bar with links for 'Sign up [Jenkins]', 'localhost:8080/jenkins/securityRealm/firstUser', 'log in', and a search bar. Below the header, the Jenkins logo is on the left, and a 'Back to Dashboard' link is available. On the right, there's a 'Sign up' button. The main form contains fields for 'Username' (admin), 'Password' (****), 'Confirm password' (****), 'Full name' (Administrator), and 'E-mail address' (al@gmail.com). The 'Sign up' button is located at the bottom of the form.

Sign up

Username: admin

Password: ****

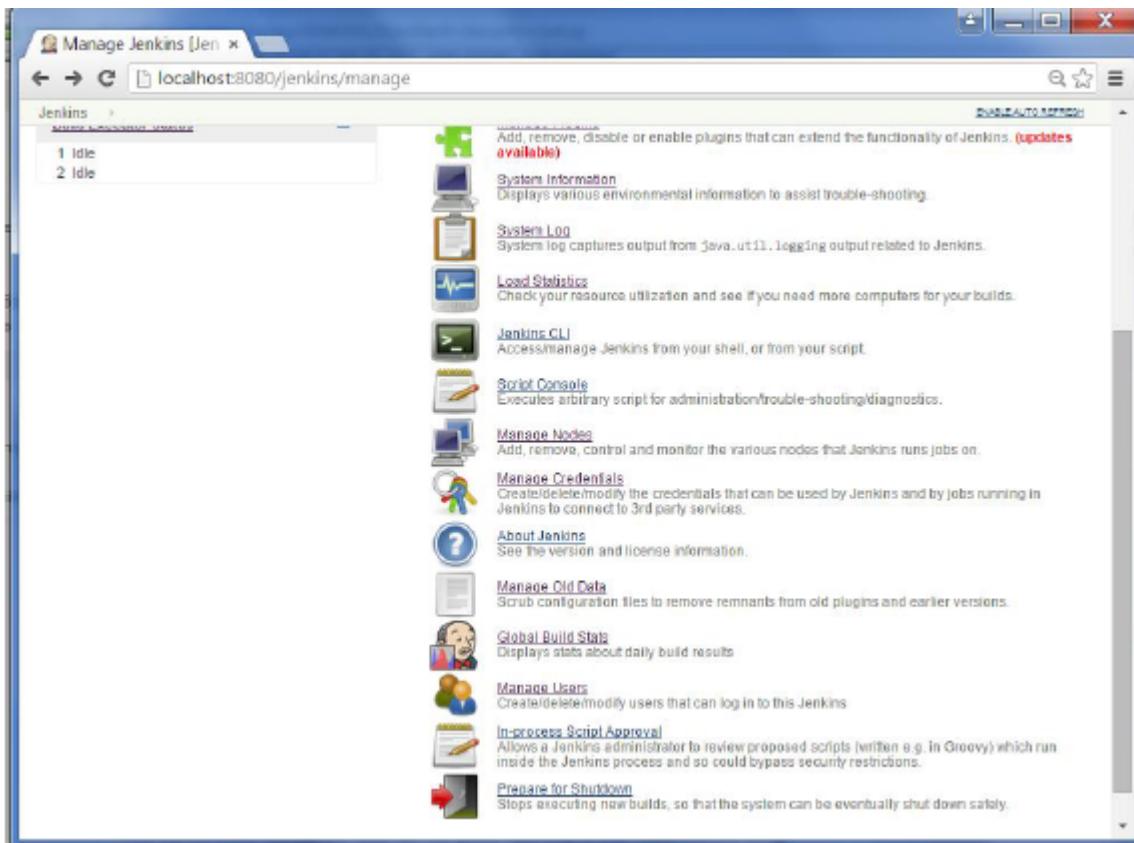
Confirm password: ****

Full name: Administrator

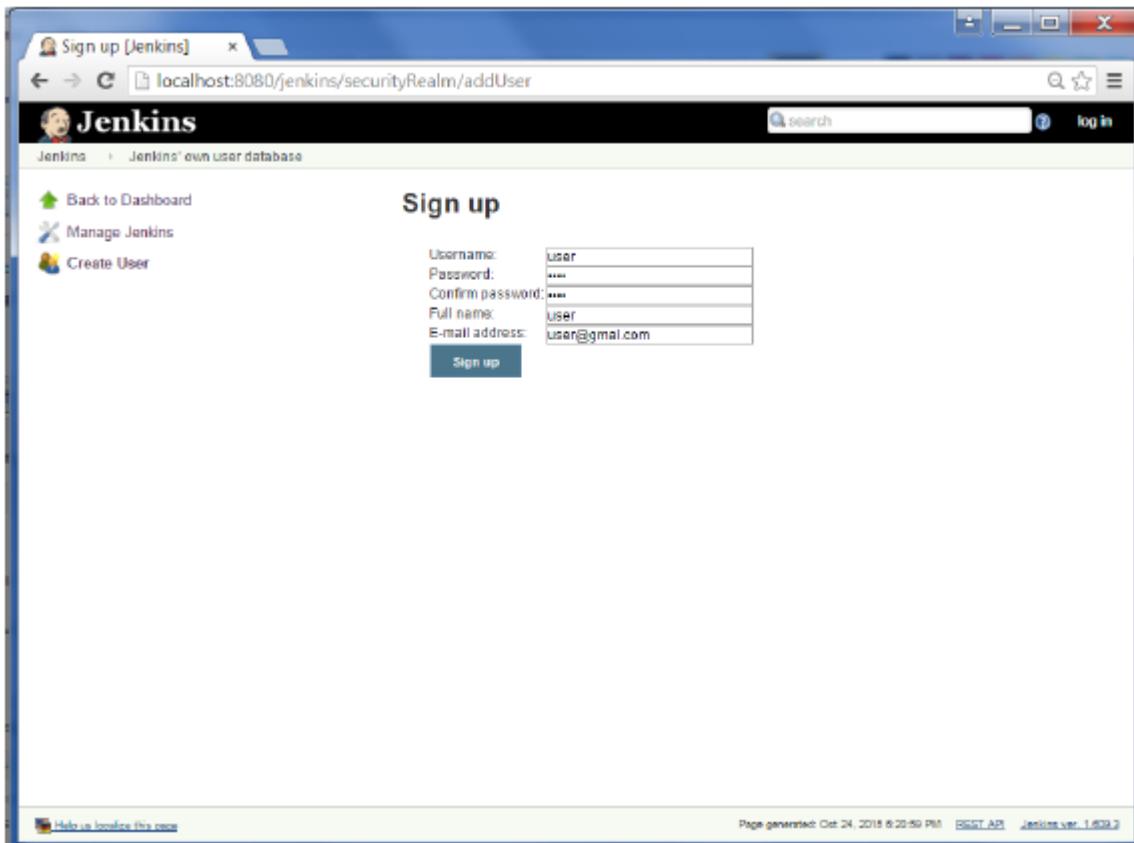
E-mail address: al@gmail.com

Sign up

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

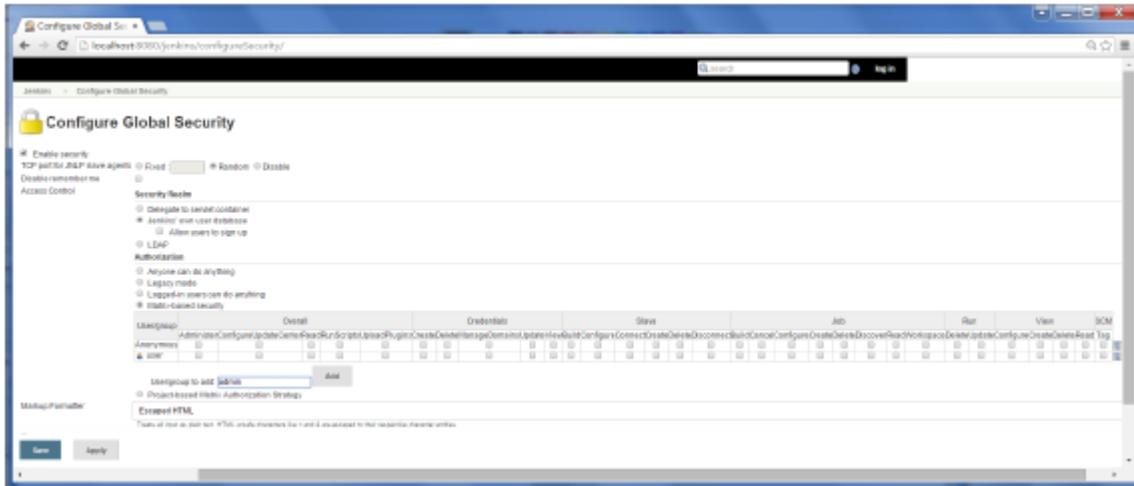


Step 5 – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.



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

Now in the Authorization section, click on 'Matrix based security'



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

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

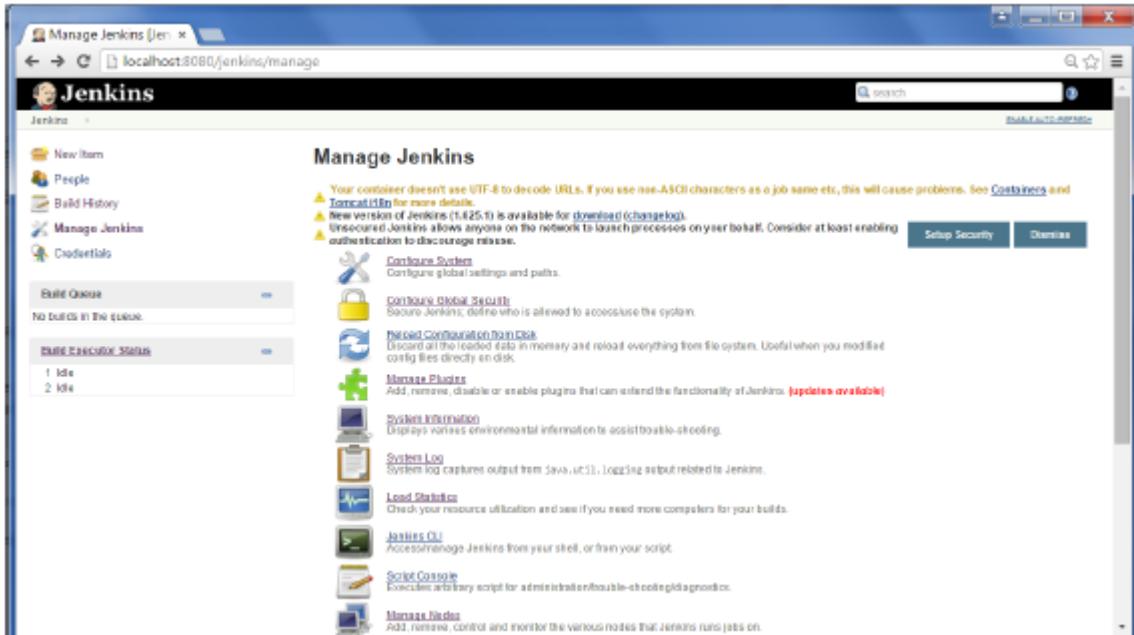
Your Jenkins security is now setup.

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

Jenkins - Backup Plugin

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

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



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

Available Tab (Top Screenshot):

| Name | Version |
|-----------------------------------|---------|
| Backup plugin | 1.6.1 |
| Backup and interrupt job plugin | 1.0 |
| CloudBees Jenkins Enterprise | 15.05.1 |
| CloudBees Free Enterprise Plugins | 5.0 |
| Periodic Backup | 1.3 |
| ThinBackup | 1.7.4 |

Installing Plugins/Upgrades Page (Bottom Screenshot):

Preparation:

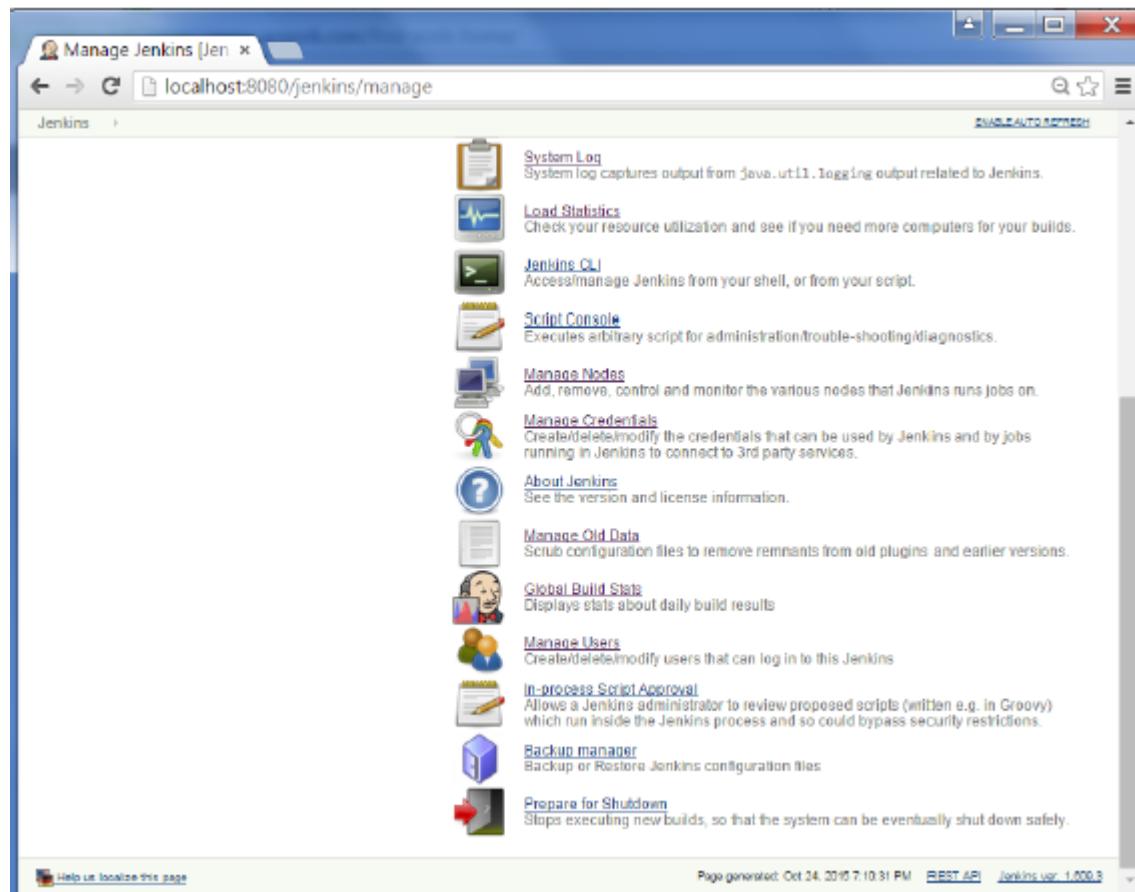
- Checking internet connectivity
- Checking update center connectivity
- Success

Backup plugin: Success

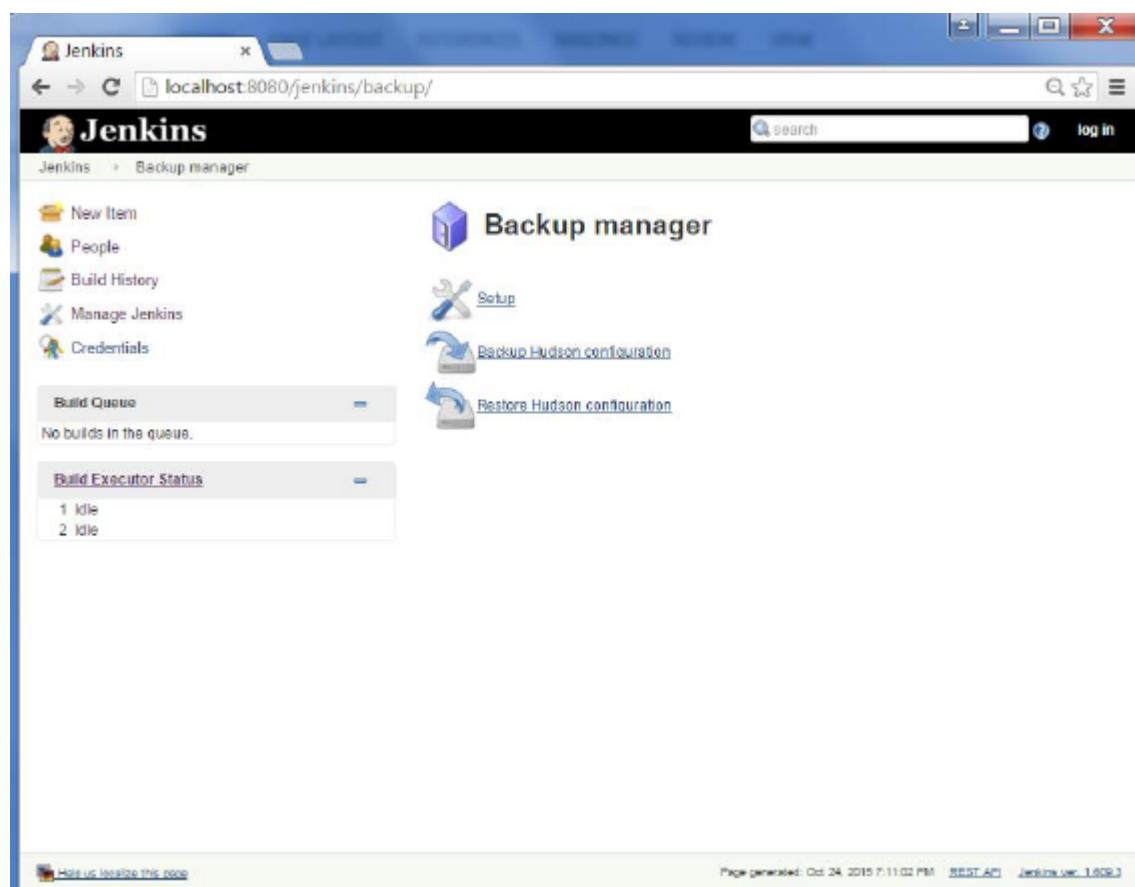
Success Messages:

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

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

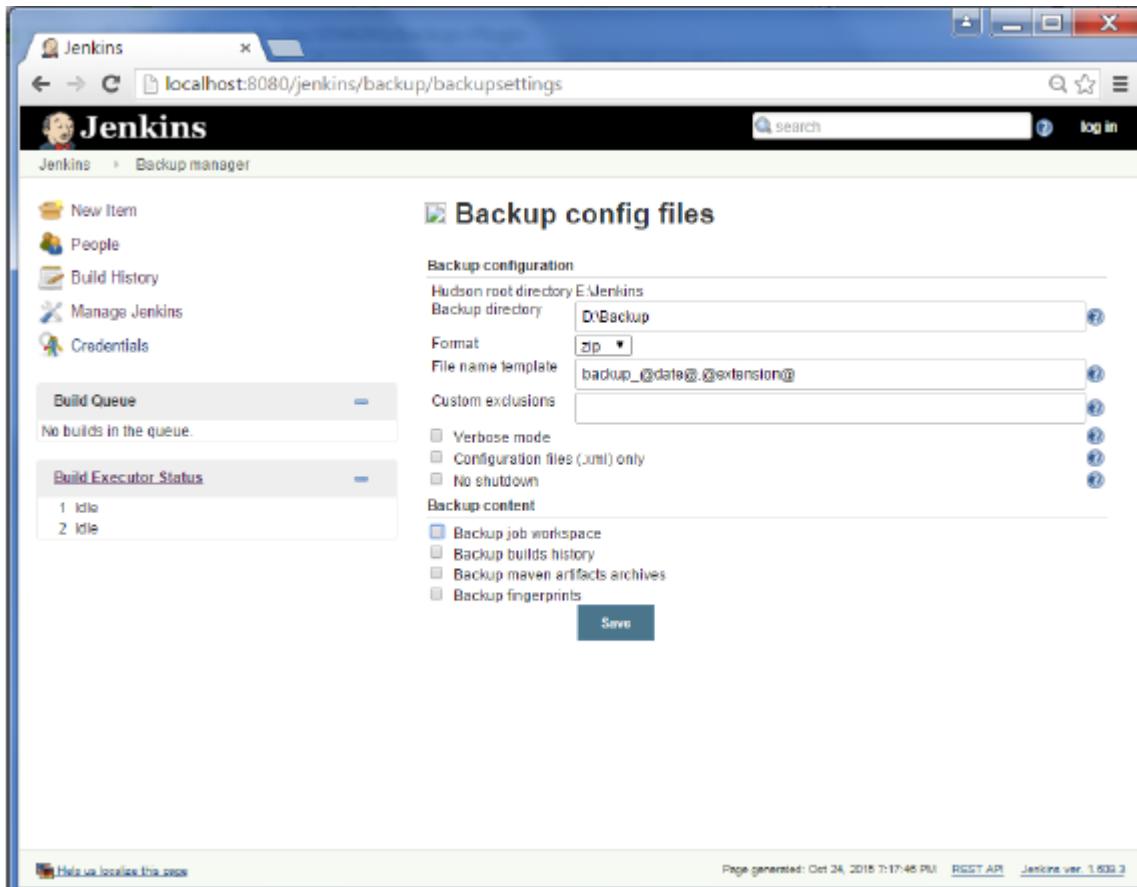


Step 4 – Click on Setup.



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

on the Save button.



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. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below these are two dropdown menus: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main area has a title 'Backup manager' with a blue cube icon. It contains three buttons: 'Setup' (gear icon), 'Backup Hudson configuration' (cloud icon), and 'Restore Hudson configuration' (arrow icon). At the bottom of the page, there's a link 'Help us localize this page' and a footer with the date 'Page generated: Oct 24, 2015 7:11:02 PM' and version 'Jenkins ver. 1.609.3'.

The next screen will show the status of the backup

The screenshot shows the Jenkins Backup manager log screen. The interface is similar to the previous one, with a sidebar and a main area. A red banner at the top right says 'Jenkins is going to shut down'. In the main area, there's a log window displaying the following text:

```
[ INFO] Backup started at [18/24/15 19:19:31]
[ INFO] Setting hudson in shutdown mode to avoid files corruptions.
[ INFO] Waiting all jobs end...
[ INFO] Number of running jobs detected : 0
[ INFO] All jobs finished.
[ INFO] Full backup file name : D:\Backup\backup_20151024_1919.zip
[ INFO] Saved files : 911
[ INFO] Number of errors : 0
[ INFO] Cancel hudson shutdown mode
[ INFO] Backup end at [18/24/15 19:19:50]
[ INFO] [19.524s]
```

At the bottom, there's a link 'Help us localize this page' and a footer with the date 'Page generated: Oct 24, 2015 7:19:31 PM' and version 'Jenkins ver. 1.609.3'.

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

The screenshot shows the Jenkins Backup manager page. At the top, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below the sidebar, there are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). On the right side, there's a large icon for 'Backup manager' and three buttons: 'Setup', 'Backup Hudson configuration', and 'Restore Hudson configuration'. At the bottom, there's a link 'Help us localize this page' and footer text 'Page generated: Oct 24, 2015 7:11:02 PM REST API Jenkins ver. 1.606.3'.

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

This screenshot shows the same Jenkins Backup manager page as above, but with a specific backup file highlighted. The backup file 'backup_20151024_1918.zip' is listed under 'Available backup in D:\Backup :'. A blue button labeled 'Launch restore' is positioned below the backup file name. The rest of the interface is identical to the first screenshot, including the sidebar, build queue, executor status, and footer information.

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 interface with the title 'Manage Jenkins [Jenkins]'. The URL in the address bar is 'dxbmem20:8080/jenkins/manage'. The page contains several links:

- 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.
- 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.

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 interface with the title 'Nodes [Jenkins]'. The URL in the address bar is 'dxbmem20:8080/jenkins/computer/'. The page displays a list of nodes:

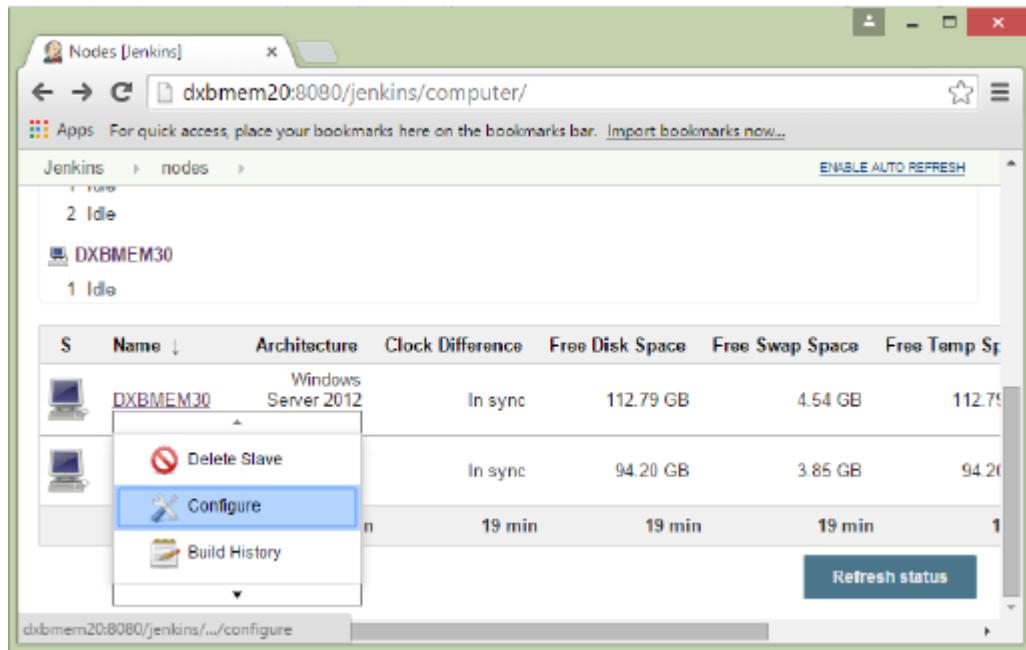
- 1 idle
- 2 idle
- DXBMEM30**
 - 1 idle

A table below lists the details for each node:

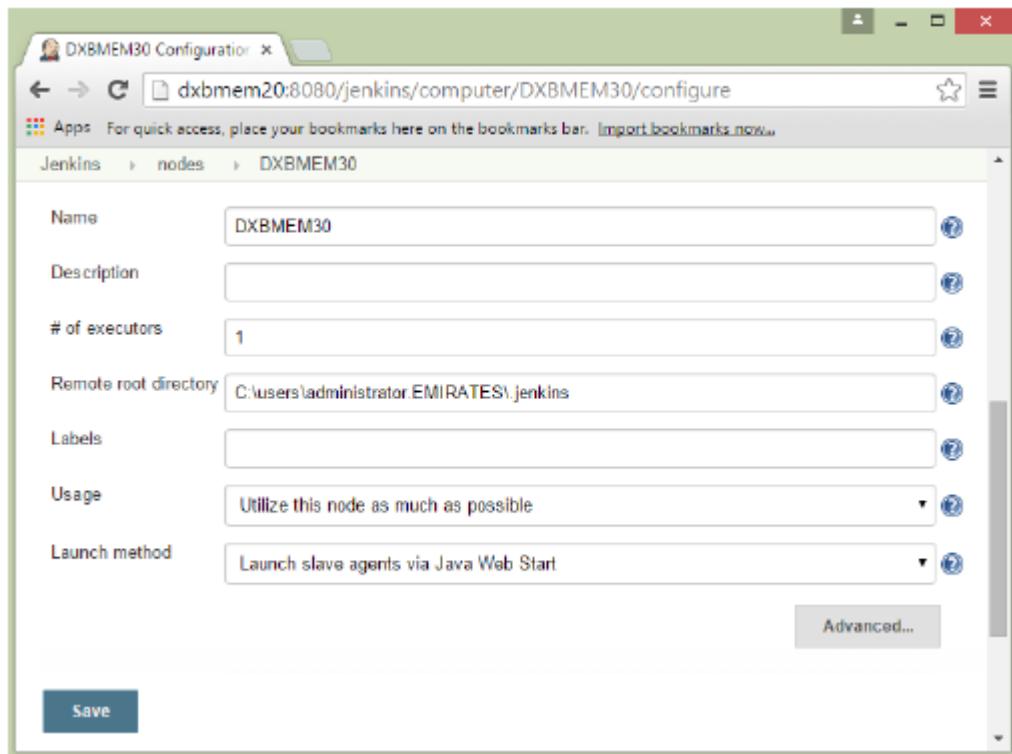
| S | Name | Architecture | Clock Difference | Free Disk Space | Free Swap Space | Free Temp Sp |
|---|---------------|---------------------------|------------------|-----------------|-----------------|---------------|
| | DXBMEM30 | Windows Server 2012 (x86) | In sync | 112.79 GB | 4.54 GB | 13 min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

At the bottom right is a blue button labeled 'Refresh status'.

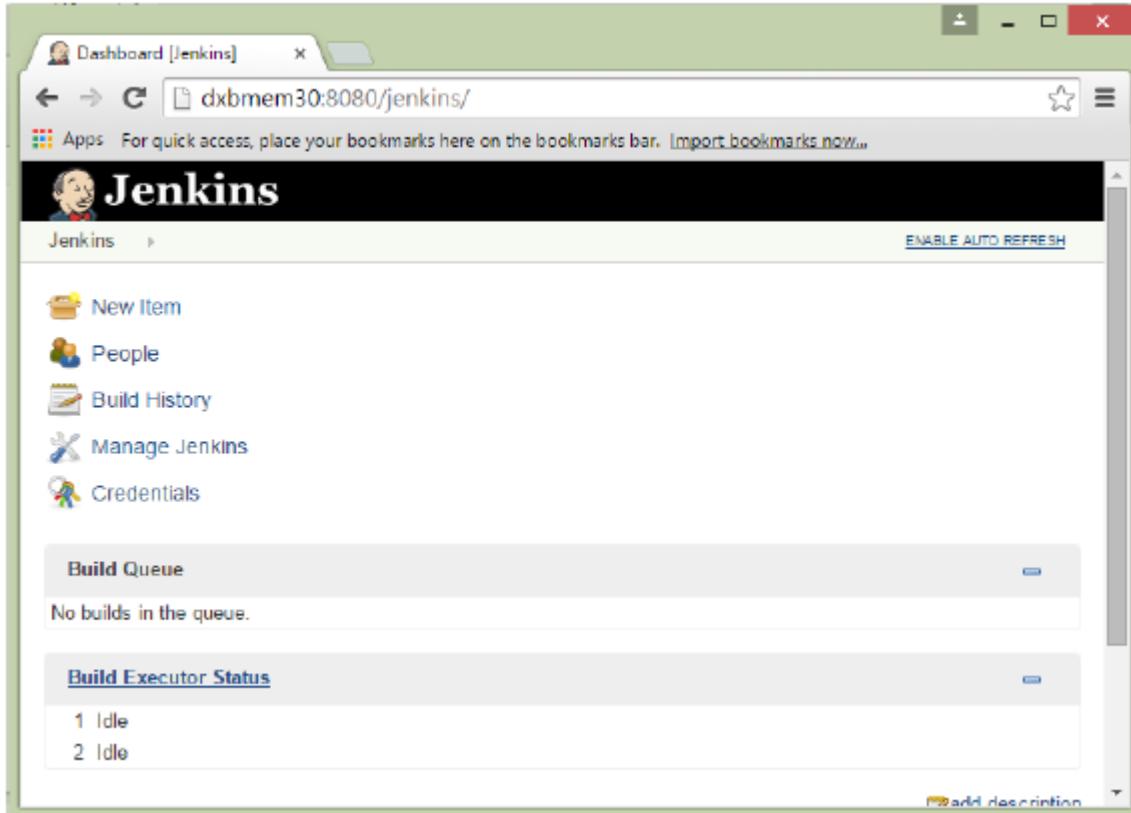
Step 2 – Click on configure for the DXBMEM30 slave machine.



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.

This screenshot shows the 'Nodes' page in Jenkins. The URL in the address bar is 'dxbmemp20:8080/jenkins/computer/'. The page displays a 'Build Executor Status' section where the 'DXBMEM30' node is listed as '(offline)'. Below this is a table of 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 'Refresh status' button is located at the bottom right of the table. A tooltip 'Data obtained' is visible near the bottom left of the table.

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

The screenshot shows a web browser window with the URL `dxbmemp20:8080/jenkins/computer/DXBMEM30/`. The page title is "Jenkins > nodes > DXBMEM30". A link "ENABLE AUTO REFRESH" is visible in the top right. Below the title, an error stack trace is shown:

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

Text below the stack trace says "Connect slave to Jenkins one of these ways:" followed by a list:

- * 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`

Text below the list says "Created by anonymous user".

Projects tied to DXBMEM30

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

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

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

The dialog box is titled "Security Warning". It asks "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.

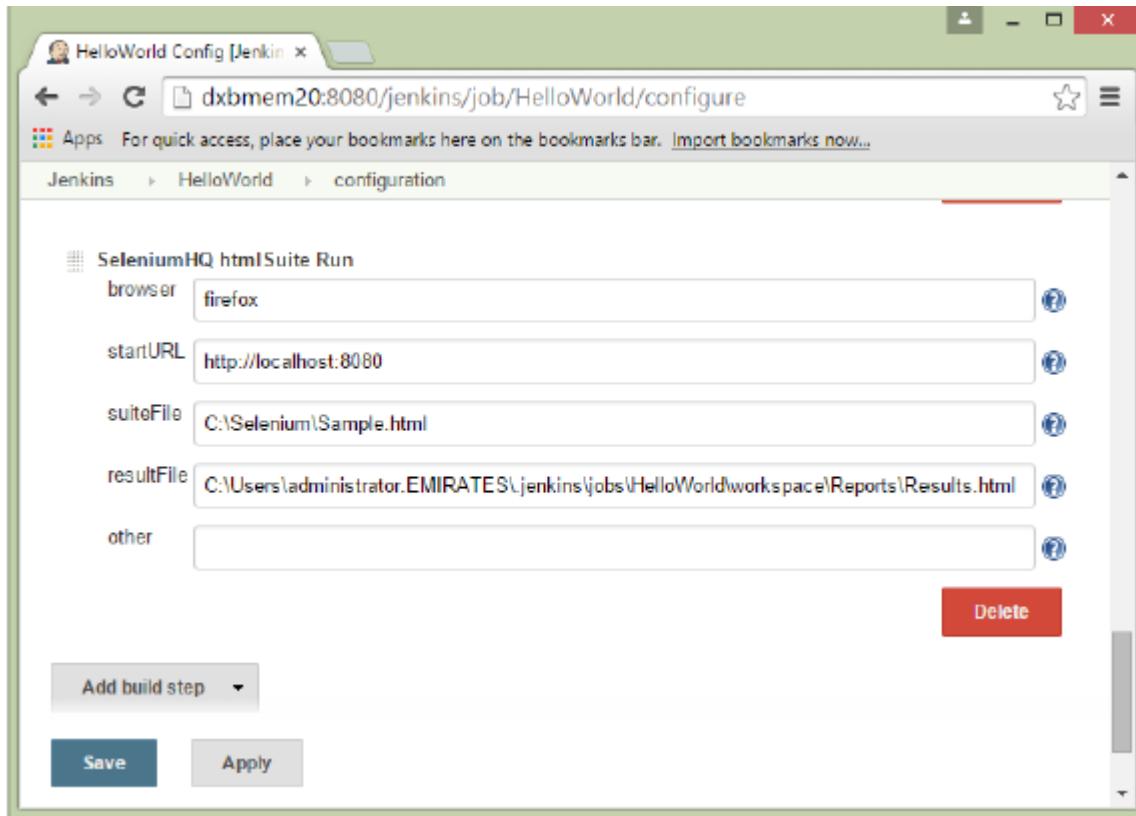


Step 8 – Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

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.

A screenshot of a web browser displaying the Jenkins job configuration for "HelloWorld". The URL in the address bar is "dxbmem20:8080/jenkins/job/HelloWorld/configure". The configuration page shows several build options: "Discard Old Builds", "This build is parameterized", "Disable Build", "Execute concurrent builds if necessary", and "Restrict where this project can be run" (which is checked). Below these options is a "Label Expression" field containing "DXBMEM30". A note below the field says "Slaves in label: 1". At the bottom of the configuration page 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.



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.

[Previous Page](#)

[Next Page](#)

Advertisements



[FAQ's](#) [Cookies Policy](#) [Contact](#)

© Copyright 2018. All Rights Reserved.

Enter email for newsletter

go