

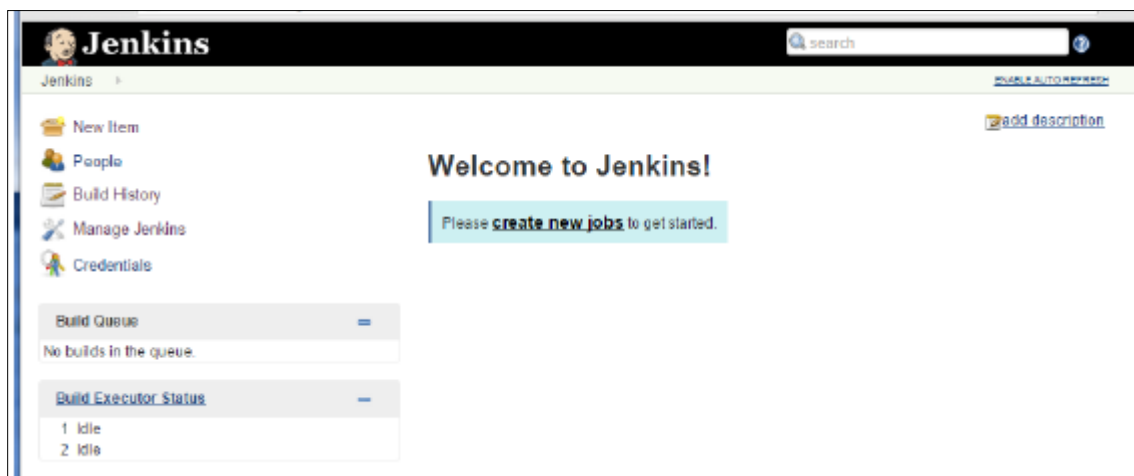
Jenkins

Jobs

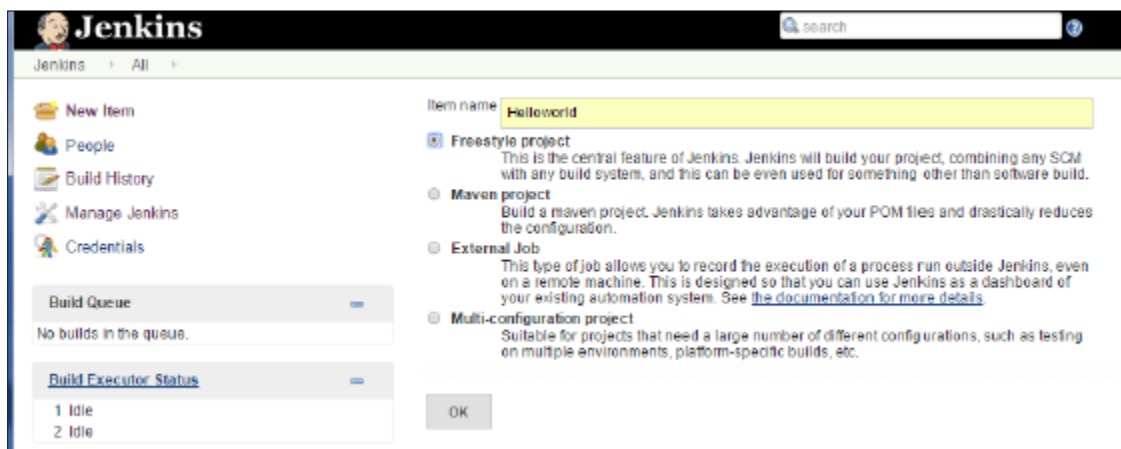
Jenkins job can be used to perform the typical build server work, such as doing continuous/official/nightly builds, run tests, or perform some repetitive batch tasks. This is called "free-style software project" in Jenkins.

Steps to create a Job

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.

The screenshot shows the Jenkins configuration page for a job named 'Helloworld'. The page is divided into several sections:

- Left Sidebar:** Contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build New', 'Delete Project', and 'Configure'. Below these is a 'Build History' section with a 'trend' link and two links: 'RSS for all' and 'RSS for failures'.
- Main Content Area:**
 - Project name:** A text field containing 'Helloworld'.
 - Description:** A large text area for the job description.
 - Discard Old Builds:** A checkbox that is currently unchecked.
 - GitHub project:** A text field for the GitHub repository path.
 - Build Options:** Three checkboxes: 'This build is parameterized' (unchecked), 'Disable Build (No new builds will be executed until the project is re-enabled)' (unchecked), and 'Execute concurrent builds if necessary' (unchecked).
 - Advanced Project Options:** A section with an 'Advanced...' button.
 - Source Code Management:** A section with radio buttons for 'None' (selected), 'CVS', 'CVS Projectset', 'Git', and 'Subversion'.
 - Build Triggers:** A section with checkboxes: 'Build after other projects are built' (unchecked), 'Build periodically' (unchecked), 'Build when a change is pushed to GitHub' (unchecked), and 'Poll SCM' (unchecked).
 - Build:** A section with a 'Build' button.
- Bottom:** Two buttons: 'Save' and 'Apply'.

Step-4 We need to specify the location of files which need to be built. In this example, we will assume that a local svn or git repository has been setup which contains a 'HelloWorld.java' file.

Source Code Management

☐ None

☐ CVS

☐ CVS Projectset

☒ Subversion

Modules

Repository URL

http://192.168.242.134:90/repo/branches/R1/

Local module directory (optional)

.

Repository depth

infinity

Ignore externals

☐

Add more locations...

Check-out Strategy

Use 'svn update' as much as possible

Use 'svn update' whenever possible, making the build **faster**. But this causes the artifacts from the previous build to remain when a new build starts.

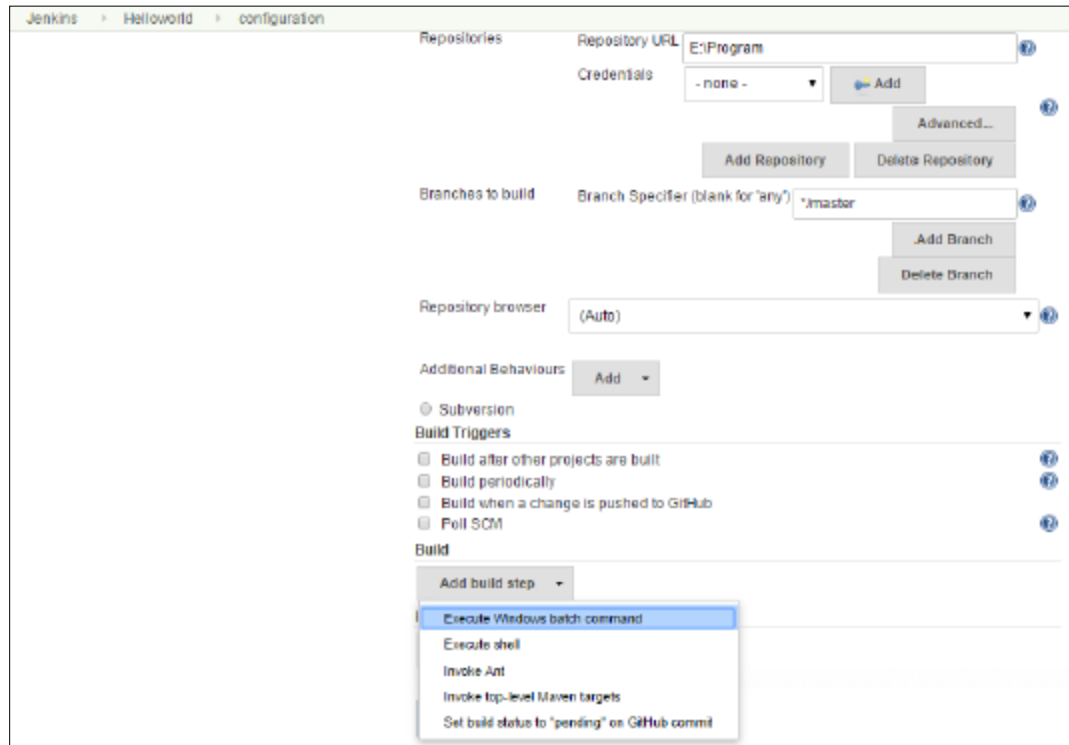
Repository browser

(Auto)

Save

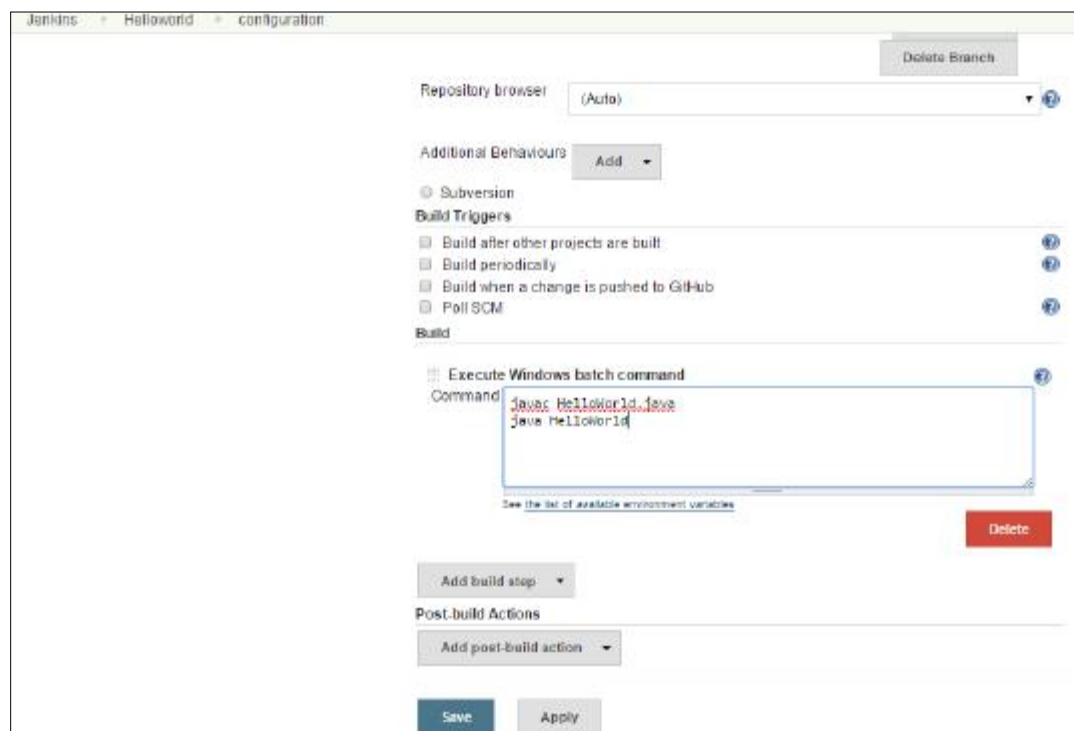
Apply

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

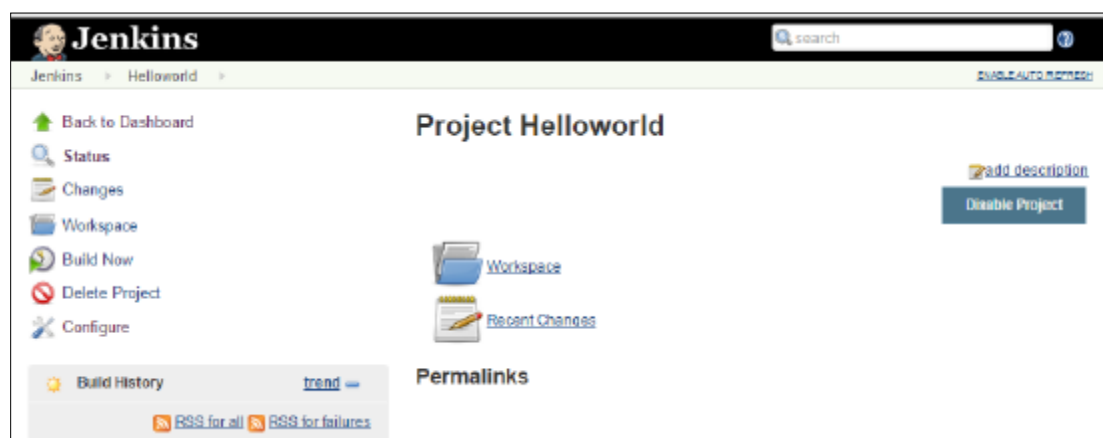


Step 6 – In the command window, enter the following commands and then click on the Save button.

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



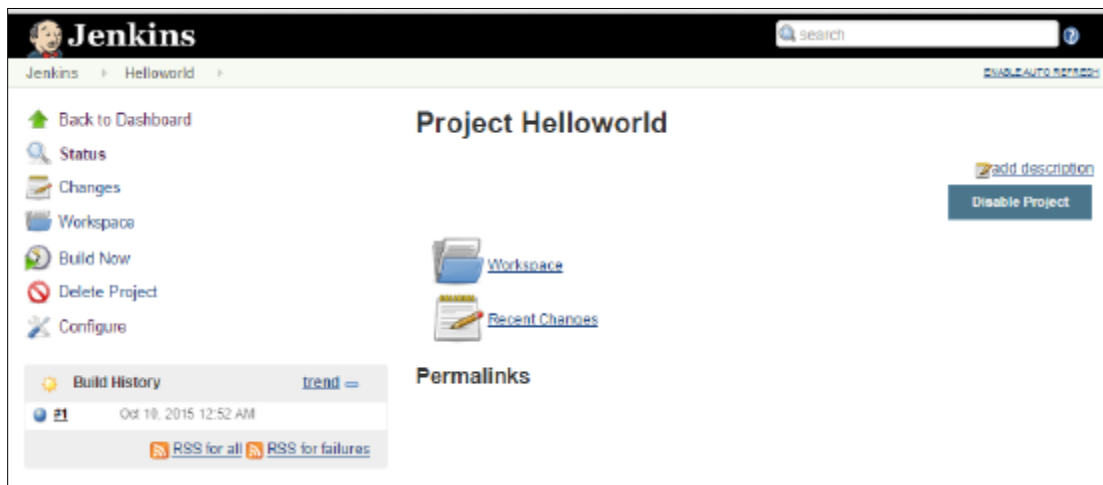
Step 7 – Once saved, you can click on the Build Now option to see if you have successfully defined the job.



Step 8- Click on the build now button to run the job, then we can see the following Build history section shows that a build is in progress.



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

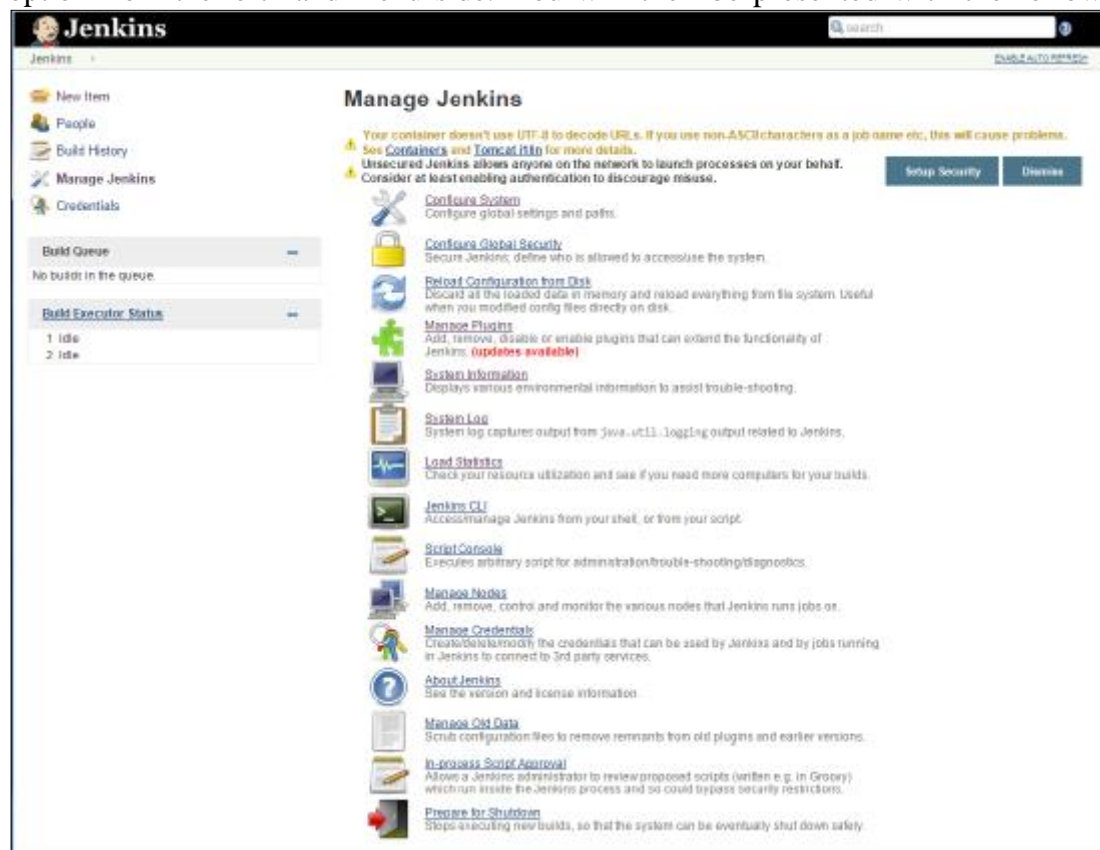


Step 10 – Click on the Console Output link to see the details of the build



Jenkins Management

Step 1- 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

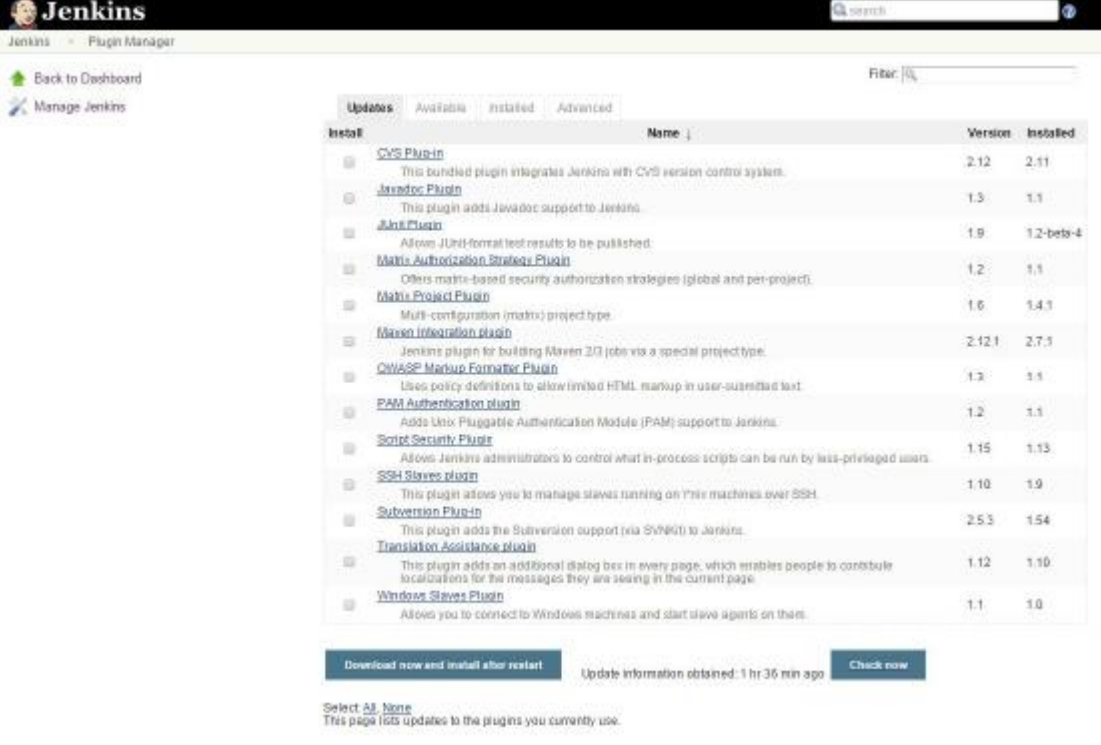
This is where one can manage paths to the various tools to use in builds, such as the JDKs, the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. When plugins are installed, Jenkins will add the required configuration fields dynamically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



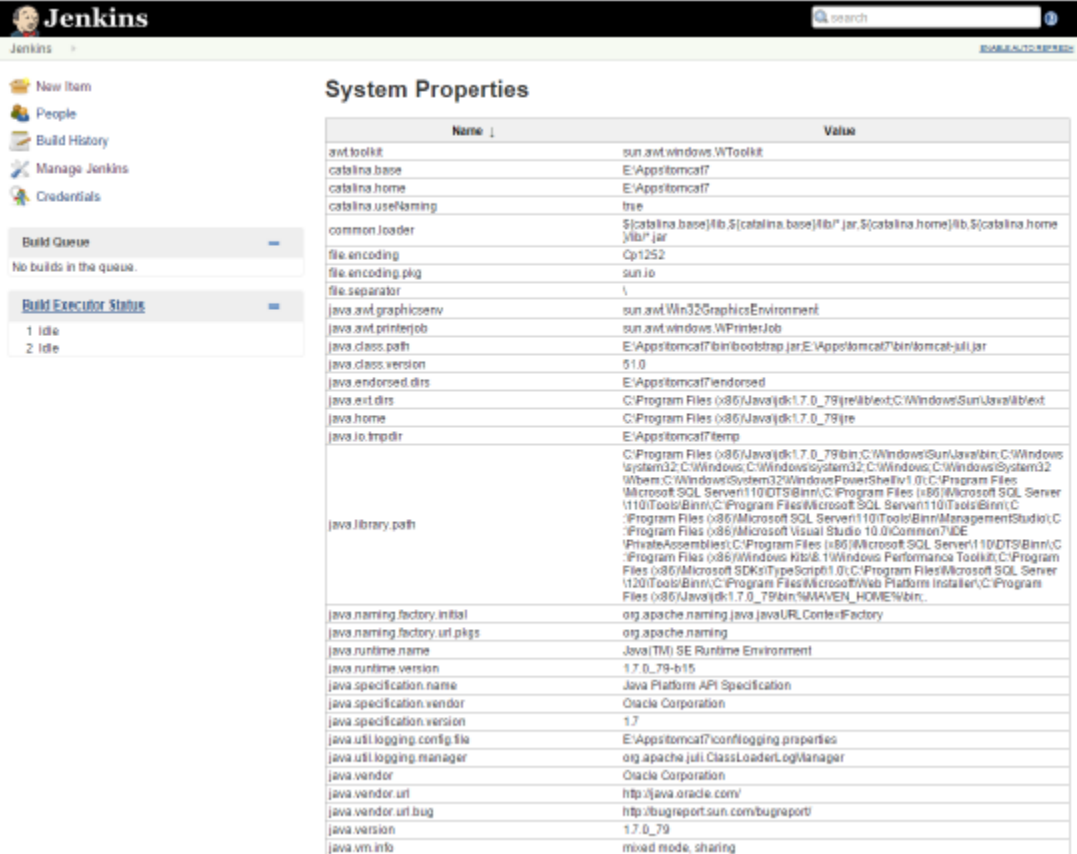
The screenshot displays the Jenkins Plugin Manager interface. At the top, there's a search bar and a filter dropdown. Below the navigation links (Back to Dashboard, Manage Jenkins), there are tabs for Updates, Available, Installed, and Advanced. The main content area shows a list of plugins under the 'Install' tab. Each plugin entry includes a checkbox, a link to the plugin, a brief description, and columns for 'Version' and 'Installed'.

	Name	Version	Installed
<input type="checkbox"/>	CVS Plugin This bundled plugin integrates Jenkins with CVS version control system.	2.12	2.11
<input type="checkbox"/>	Javadoc Plugin This plugin adds Javadoc support to Jenkins.	1.3	1.1
<input type="checkbox"/>	JUnit Plugin Allows JUnit-format test results to be published.	1.9	1.2-beta-4
<input type="checkbox"/>	Matrix Authorization Strategy Plugin Offers matrix-based security authorization strategies (global and per-project).	1.2	1.1
<input type="checkbox"/>	Matrix Project Plugin Multi-configuration (matrix) project type.	1.6	1.4.1
<input type="checkbox"/>	Maven Integration plugin Jenkins plugin for building Maven 2/3 jobs via a special project type.	2.12.1	2.7.1
<input type="checkbox"/>	OWASP Markup Formatter Plugin Uses policy definitions to allow limited HTML markup in user-submitted text.	1.3	1.1
<input type="checkbox"/>	PAM Authentication plugin Adds Unix Pluggable Authentication Module (PAM) support to Jenkins.	1.2	1.1
<input type="checkbox"/>	Script Security Plugin Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users.	1.15	1.13
<input type="checkbox"/>	SSH Slaves plugin This plugin allows you to manage slaves running on remote machines over SSH.	1.10	1.9
<input type="checkbox"/>	Subversion Plugin This plugin adds the Subversion support (via SVNKit) to Jenkins.	2.5.3	1.54
<input type="checkbox"/>	Translation Assistance plugin This plugin adds an additional dialog box in every page, which enables people to contribute localizations for the messages they are seeing in the current page.	1.12	1.10
<input type="checkbox"/>	Windows Slaves Plugin Allows you to connect to Windows machines and start slave agents on them.	1.1	1.0

At the bottom, there are buttons for 'Download now and install after restart', 'Update information obtained: 1 hr 36 min ago', and 'Check now'. Below these, a note says 'Select All: None' and 'This page lists updates to the plugins you currently use.'

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.



Name	Value
ant.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\App\Tomcat7
catalina.home	E:\App\Tomcat7
catalina.useNaming	true
common.loader	\$(catalina.base)\lib\\$(catalina.base)\lib*.jar;\$(catalina.home)\lib\\$(catalina.home)\lib*.jar
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\App\Tomcat7\bin\bootstrap.jar;E:\App\Tomcat7\bin\tomcat-juli.jar
java.class.version	51.0
java.endorsed.dirs	E:\App\Tomcat7\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:\App\Tomcat7\temp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\Sun\Java\bin;C:\Windows\System32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\10\DTSBinn\;C:\Program Files (x86)\Microsoft SQL Server\10\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\10\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files (x86)\Microsoft SQL Server\10\DTSBinn\;C:\Program Files (x86)\Windows Kits\8.1\Windows Performance Toolkit\;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\Java\jdk1.7.0_79\bin\NAVEN_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:\App\Tomcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url.bug	http://bugreport.sun.com/bugreport/
java.version	1.7.0_79
java.vm.info	mixed mode, sharing

System Log

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

Load Statistics

This page displays graphical data on how busy the Jenkins instance is in terms of the number of concurrent builds and the length of the build queue which gives an idea of how long your

builds need to wait before being executed. These statistics can give a good idea of whether extra capacity or extra build nodes is required from an infrastructure perspective.

Script Console

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

Manage nodes

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

Prepare for Shutdown

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