**https://wiki.jenkins-ci.org/display/JENKINS/Starting+and+Accessing+Jenkins**

**Installations on RedHat SYSTEM pl. refer the url info**

[**https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions**](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions)

**Run Jenkins**

* Jenkins can be deployed as a war file in the Apache Tomcat Server.
* It can be installed as a windows service
* It can be started as java jar application from command line

>java –jar Jenkins.war which runs Jenkins on default port =8080.

* To get help on Jenkins run from command line : java -jar jenkins.war --help

**The** [**RedHat Jenkins distribution**](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Red+Hat+distributions) **contains a startup script.**

**To change the port number**

* java -jar jenkins.war --httpPort=9090
* For working with ssl
* java -jar jenkins.war --httpsPort=9090
* Jenkins port settings
* open the jenkin.xml in the jenkins folder and change the port number:  
  httpPort=xxxx   
  to  
  httpPort=yyyy --- > restart the machine. it should change the setting permanently.

**On Windows (with Jenkins as Windows Service).**

Edit the file C:\Program Files (x86)\Jenkins\jenkins.xml with **8083** if you want 8083 port.

<arguments>-Xrs -Xmx256m -Dhudson.lifecycle=hudson.lifecycle.WindowsServiceLifecycle -jar "%BASE%\jenkins.war" --httpPort=8083</arguments>

**On RedHat systems**

vim /etc/sysconfig/jenkins

# Port Jenkins is listening on.

# Set to -1 to disable

#

JENKINS\_PORT="8080"

**For Fedora, RedHat, CentOS and alike, any customization should be done within /etc/sysconfig/jenkins** instead of /etc/init.d/jenkins.

The purpose of the first file is exactly the customization of the second file.

So, within /etc/sysconfig/jenkins, there is a the JENKINS\_PORT variable that holds the port number on which Jenkins is running.

Update port: change the '/etc/init.d/jenkins' shell

check\_tcp\_port "http" "$HTTP\_PORT" "8080" || return 1

To send the output of Jenkins to a log file,

java -jar jenkins.war --logfile=file.log

* java -jar jenkins.war --logfile=file.log --debug=9
* java -jar jenkins.war --httpPort=9090 --logfile=file.log
* java -jar jenkins.war --httpPort=9090 --logfile=file.log --debug=9

On Unix, to use **nohup**:

$ nohup java -jar jenkins.war > $LOGFILE 2>&1

**Using HTTPS with an existing certificate**

If you're setting up Jenkins using the built-in Winstone server and want to use an existing certificate for HTTPS:

--httpPort=-1 --httpsPort=443 --httpsKeyStore=path/to/keystore --httpsKeyStorePassword=keystorePassword

The keystore should be in JKS format (as created by the JDK 'keytool') and the keystore and target key must have the same password. (Placing the keystore arguments after Jenkins-specific parameters does not seem to work; either they are not forwarded to Winstone or Winstone ignores them coming after unknown parameters. So, make sure they are adjacent to the working --httpsPort argument.)

If your keystore contains multiple certificates (e.g. you are using CA signed certificate) Jenkins might end-up using a incorrect one. In this case you can [convert the keystore to PEM](http://stackoverflow.com/questions/7528944/convert-ca-signed-jks-keystore-to-pem) and use following command line options:

--httpPort=-1 --httpsPort=443 --httpsCertificate=path/to/cert --httpsPrivateKey=path/to/privatekey

**Passing the Command Line Parameters to an instance on a Mac OSX (Currently is Mavericks 10.9.4) that uses launchctl (rather than using Jenkins.jar to start up)**

In this example, we set the Jenkins server to listen for HTTPS on port 8443. Not we do not disable the httpPort by passing in -1. So in this example, your server would answer on both http and https. We also assume that the user has already created the keystore (see the "Using SSL" section from <http://wiki.wocommunity.org/display/documentation/Installing+and+Configuring+Jenkins>)

sudo launchctl unload /Library/LaunchDaemons/org.jenkins-ci.plist

sudo defaults write /Library/Preferences/org.jenkins-ci httpsPort 8443

sudo defaults write /Library/Preferences/org.jenkins-ci httpsKeyStore /path/to/your/keystore/file

sudo defaults write /Library/Preferences/org.jenkins-ci httpsKeyStorePassword <keystore password>

sudo launchctl load /Library/LaunchDaemons/org.jenkins-ci.plist

1. Start Jenkins by executing as you normally do, but add two extra parameters to the startup:

$ java -jar jenkins.war --argumentsRealm.passwd.*user*=*password* --argumentsRealm.roles.*user*=admin

Note that *user* should be the name of the administrative user, and *password* should be the password for that user. Also please note that the user is specified twice - once in each argument.   
For example, I want to have a user *jenkins* with a password of *swordfish* as my administrative user:

$ java -jar jenkins.war --argumentsRealm.passwd.*jenkins*=*swordfish* --argumentsRealm.roles.*jenkins*=admin

Notice that the word *arguments* is plural in both --argumentsRealm command line parameters.

1. Enable the security setting, go to http://yourhost/jenkins/configureSecurity and select *enable security*, then choose *Delegate to servlet container* for security realm and *Legacy mode* for authorization strategy. Or, you may click on the **Manage Jenkins** link on the left side of the main *Jenkins* dashboard page to get to the configuration page, then click **Configure Global Security**.
2. To log into *Jenkins* go to http://yourhost/jenkins/loginEntry, or click on the **login** link located on the top right hand corner of any Jenkins Dashboard page.
3. This will bring you to the Login page. Enter the user name and password, and click on the **Submit** button. This will take you back to the main Jenkins dashboard. You will now see the **Manage Jenkins** link on the left side of the page.
4. To logout simply click the **logout** link located on the top right hand corner of any *Jenkins* dashboard page.

To restart Jenkins manually, use either of the following commands:

1. (jenkins\_url)/safeRestart - Allows all running jobs to complete. New jobs will remain in the queue to run after the restart is complete.
2. (jenkins\_url)/restart - Forces a restart without waiting for builds to complete.

(jenkins\_url)/{start|stop|status|restart|force-reload}

From Command path

java -jar jenkins-cli.jar -s http://[jenkins-server]/ restart

To shut down

http://[jenkins-server]/exit

http://[jenkins-server]/[command]

where [command] can be

* exit shutdown jenkins
* restart restart jenkins
* reload to reload the configuration

The **Remote Jenkins URL** can take any of the following types of URLs:

* <http://$JENKINS> - get all jobs on remote instance
* <http://$JENKINS/job/$JOBNAME> - get a single job
* <http://$JENKINS/view/$VIEWNAME> - get all jobs in a particular view

**Using Gmail smtp to send email**

1. [**SMTP relay service**](https://support.google.com/a/answer/2956491) –(smtp-relay.gmail.com) used to send mail from your organization by authenticating with the IP address(s). You can send messages to anyone inside or outside of your domain.
2. **Gmail SMTP server** – (smtp.gmail.com)requires authentication with your Gmail/Google Apps account and password. Messages can be sent to anyone inside or outside of your domain.
3. **Restricted Gmail SMTP server** – (aspmx.l.google.com)does not require authentication, and you will be restricted to send messages to Gmail or Google Apps users only.

# Jenkins Script Console

Jenkins features a nice Groovy script console which allows to run arbitrary scripts on the Jenkins server or on slave nodes. This feature can be accessed from the "manage Jenkins" link, typically at your [http://server/jenkins/script](http://server/hudson/script).

# Shortcut key on script console to submit

You can submit a script without mouse. Jenkins has a shortcut key which enables to submit with keyboard.

* Windows / Linux : Ctrl + Enter
* Mac : Command + Enter

# Remote access

User can execute groovy scripts remotely sending post request to /script/ url or /scriptText/ to have response returned without the html wrapping.

Send the script wirh curl rest command

$ curl -d "script=<your\_script\_here>" http://jenkins/script

$ # or

$ curl -d "script=<your\_script\_here>" http://jenkins/scriptText

Also, Jenkins CLI offers the possibility to execute groovy scripts remotely using groovy command or execute groovy interactivelly via groovysh.

|  |  |
| --- | --- |
|  |  |
|  |  |

# Jenkins Environment Variables

|  |  |
| --- | --- |
| **Environment Variable** | **Description** |
| BUILD\_NUMBER | The current build number, such as "153" |
| BUILD\_ID | The current build id, such as "2005-08-22\_23-59-59" (YYYY-MM-DD\_hh-mm-ss, [defunct](https://issues.jenkins-ci.org/browse/JENKINS-26520) since version 1.597) |
| BUILD\_URL | The URL where the results of this build can be found (e.g. http://buildserver/jenkins/job/MyJobName/666/) |
| NODE\_NAME | The name of the node the current build is running on. Equals 'master' for master node. |
| JOB\_NAME | Name of the project of this build. This is the name you gave your job when you first set it up. It's the third column of the Jenkins Dashboard main page. |
| BUILD\_TAG | String of jenkins-${JOB\_NAME}-${BUILD\_NUMBER}. Convenient to put into a resource file, a jar file, etc for easier identification. |
| JENKINS\_URL | Set to the URL of the Jenkins master that's running the build. This value is used by [Jenkins CLI](https://wiki.jenkins-ci.org/display/JENKINS/Jenkins+CLI) for example |
| EXECUTOR\_NUMBER | The unique number that identifies the current executor (among executors of the same machine) that's carrying out this build. This is the number you see in the "build executor status", except that the number starts from 0, not 1. |
| JAVA\_HOME | If your job is configured to use a specific JDK, this variable is set to the JAVA\_HOME of the specified JDK. When this variable is set, PATH is also updated to have $JAVA\_HOME/bin. |
| WORKSPACE | The absolute path of the workspace. |
| SVN\_REVISION | For Subversion-based projects, this variable contains the revision number of the module. If you have more than one module specified, this won't be set. |
| CVS\_BRANCH | For CVS-based projects, this variable contains the branch of the module. If CVS is configured to check out the trunk, this environment variable will not be set. |
| GIT\_COMMIT | For Git-based projects, this variable contains the Git hash of the commit checked out for the build (like ce9a3c1404e8c91be604088670e93434c4253f03) ﻿(all the GIT\_\* variables require git plugin) |
| GIT\_URL | For Git-based projects, this variable contains the Git url (like git@github.com:user/repo.git or [https://github.com/user/repo.git) ] |
| GIT\_BRANCH | For Git-based projects, this variable contains the Git branch that was checked out for the build (normally origin/master) |

# Jenkins Script Console

Jenkins features a nice Groovy script console which allows to run arbitrary scripts on the Jenkins server or on slave nodes. This feature can be accessed from the "manage Jenkins" link, typically at your [http://server/jenkins/script](http://server/hudson/script).

outputs all job names:

def hi = hudson.model.Hudson.instance

hi.getItems(hudson.model.Project).each {project ->

println(project.displayName)

}

# gets statuses for all the latest builds:

def hi = hudson.model.Hudson.instance

hi.getItems(hudson.model.Project).each {project ->

println(project.lastBuild.result)

}

**SVN Diff Viewer**

There is no such tool. All you can do is write such diff viewer by yourself or adapt to your needs existing web-based diff viewers, such as [websvn](http://www.websvn.info/) or [viewvc](http://www.viewvc.org/).

Actually, sending diffs via email is not a bad idea at all. You might install appropriate diff viewer ([freediff](http://www.freediff.com/), [winmerge](http://winmerge.org/) or any other; full list of popular diff viewers you can find [here](http://stackoverflow.com/questions/12625/best-diff-tool)) on your machine and create corresponding association between app and .diff extension. You will be able to open diffs directly from your e-mail.

# configure Jenkins to create links to a repository browser that can show diffs for you (e.g. FishEye, ViewSVN, Polarion, Sventon, etc.

|  |  |  |
| --- | --- | --- |
| Limited built-in support, or via third-party tools such as ViewVC | Only via third-party tools such as ViewVC | Only via third-party tools such as ViewVC |

# ViewVC Plugin

This plugin integrates ViewVC browser interface for CVS and Subversion with Hudson.  
Entries in change logs will by hyperlinked to the specified files/diffs in ViewVC.

**Working with Proxy**

Jenkins is a Java application, so it will inherit all HTTP(S) proxy related behaviour from the Java VM unless it's overwritten by any specific behaviour in the Jenkins or the plugin code itself.

In the simplest possible case, it actually comes down to setting some system properties using for example -D when starting Jenkins, i.e.:

java -Dhttp.proxyHost=some.proxy.host -Dhttp.proxyPort=1234 -jar jenkins.war

Note that if you need to access HTTPS sites as well from Jenkins, you also need to setup the HTTPS related system properties, even if your HTTP and HTTPS proxy are the same machine, which is often the case:

java -Dhttp.proxyHost=some.proxy.host -Dhttp.proxyPort=1234 -Dhttps.proxyHost=some.otherorsame.host -Dhttps.proxyPort=2345 -jar jenkins.war

There are some other properties you may need or want to set, such as **http.nonProxyHosts** for example. Make sure to properly read the documentation provided in the link above.

## 

## Configuring a Proxy from Console

In most enterprise environments, your Jenkins server will be situated behind a firewall, and will not have direct access to the Internet. Jenkins needs Internet access to download plugins and updates, and also to install tools such as the JDK, Ant and Maven from remote sites. If you need to go through an HTTP proxy server to get to the Internet, you can configure the connection details (the server and port, and if required the username and password) in the Advanced tab on the Plugin Manager screen

Finally, if you are setting up Proxy access on your Jenkins build server, remember that all of the other tools running on this server will need to know about the proxy as well. In particular, this may include tools such as Subversion (if you are accessing an external repository) and Maven (if you are not using an Enterprise Repository Manager).

If your proxy is using Microsoft’s NTLM authentication scheme, then you will need to provide a domain name as well as a username. You can place both in the User name field: just enter the domain name, followed by a back-slash (\), followed by the username, such as “MyDomain\Joe Bloggs”.

back-slash (\), followed by the username, such as “MyDomain\Joe Bloggs”.

**Disable Security**

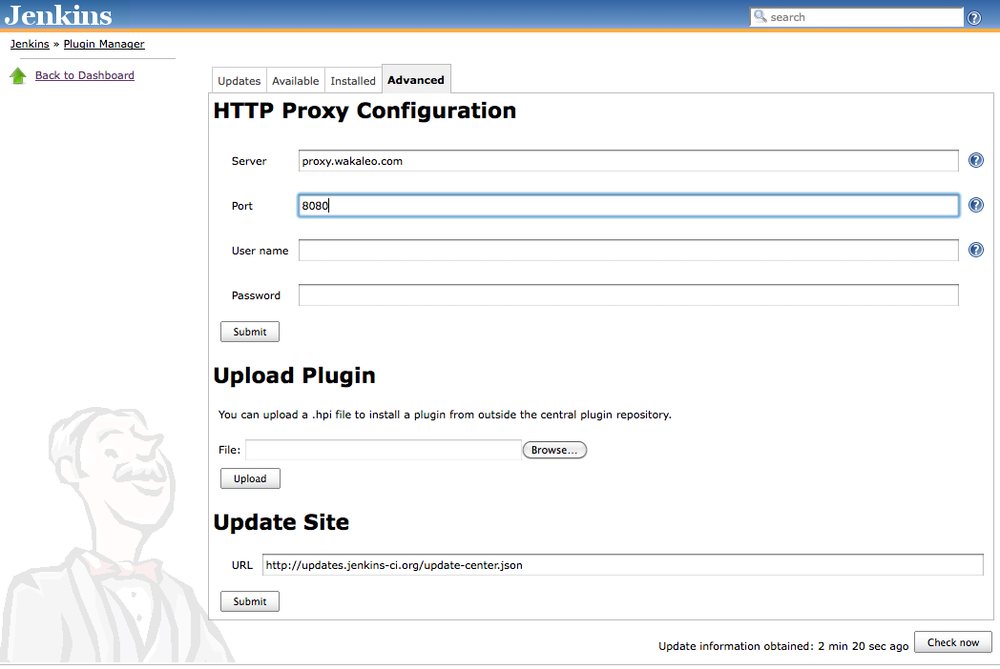
**One may accidentally set up security realm / authorization in such a way that you may no longer able to reconfigure Jenkins.**

When this happens, you can fix this by the following steps:

1. Stop Jenkins (the easiest way to do this is to kill the servlet container.)
2. Go to $JENKINS\_HOME in the file system and find config.xml file.
3. Open this file in the editor.
4. Look for the <useSecurity>true</useSecurity> element in this file.
5. Replace true with false
6. Remove the elements authorizationStrategy and securityRealm
7. Start Jenkins

When Jenkins comes back, it's in the unsecured mode where everyone gets full access to the system.

If this is still not working, trying renaming or deleting config.xml.

Figure  : Configuring Jenkins to use a proxy

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |