



# Jenkins Lab Guide

**[www.zippyops.com](http://www.zippyops.com)**

172-172, 5th floor Old Mahabalipuram Road  
(Above Axis Bank-PTC Bus Stop)

Thuraipakkam

Chennai 600097

**✉ [zippyops@gmail.com](mailto:zippyops@gmail.com)**

**⌚ +91 7010585768**

## CONTENTS

Installing Jenkins on a CentOS : .....	2
Setup JAVA_HOME, JRE_HOME and PATH environment variables.....	4
Installing JENKINS:.....	5
Enable Jenkins and start service .....	6
Start using JENKINS:.....	10
Manage JENKINS: .....	11
Configure SYSTEM:.....	11
Reload Configuration from DISK:.....	11
Manage Plugins:.....	12
System INFORMATION:.....	12
System Log: .....	12
Load Statistics: .....	12
Script CONSOLE:.....	12
Manage NODES:.....	13
Prepare for Shutdown:.....	13
Configure Security in JENKINS: .....	13
Manage credentials: .....	16
Plugin management:.....	20
Uninstalling Plugins:.....	20
Backup plugin:.....	21
Configure Jenkins Slave and Connect to Master: .....	27
Set Up Jenkins Access to GitHub :.....	32
Create a New Pipeline in Jenkins: .....	34
Install Apache Maven on CentOS 7 : .....	38
Setting up Jenkins and MAVEN:.....	39

**ZippyOPS**  
Making Automation Work

## INSTALLING JENKINS ON A CENTOS:

First, we need to get the only dependency we need to get Jenkins to take off, and that is Java (more precisely, Java8),

Before that, we can do check port 8080 available or not in our system.

```
zippyops@localhost:/home/zippyops
File Edit View Search Terminal Help
[root@localhost zippyops]# which java
/bin/java
[root@localhost zippyops]# cat /etc/services | grep 8080
webcache      8080/tcp      http-alt      # WWW caching service
webcache      8080/udp      http-alt      # WWW caching service
[root@localhost zippyops]#
```

After that to check with port 8080 speak or not, if it is not speak , we need to install telnet,

For that command as ,

```
# telnet localhost 8080 -> if command not found means we need to install telnet.
```

To install telnet,

```
# yum install telnet
```

```
zippyops@localhost:/home/zippyops
File Edit View Search Terminal Help
[root@localhost zippyops]# yum install telnet
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.viethosting.com
 * extras: mirror.ehost.vn
 * updates: mirrors.123host.vn
Package 1:telnet-0.17-64.el7.x86_64 already installed and latest version
Nothing to do
```

After that telnet installation to verify telnet,

```
[root@localhost zippyops]# telnet localhost 8080
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
[root@localhost zippyops]#
```

If refused the connection means nothing listening on us. So we have to know which one is running in local host port,

To know that give a command as

```
" # nmap -sT -O localhost "
```

If you do not have nmap so you need to install that too, for that command as

```
"yum install nmap"
```

```
zippyops@localhost:/home/zippyops
File Edit View Search Terminal Help
[root@10 zippyops]# nmap -sT -O localhost
zippyops@localhost:/home/zippyops
File Edit View Search Terminal Help
RTTVar has grown to over 2.3 seconds, decreasing to 2.0
RTTVar has grown to over 2.3 seconds, decreasing to 2.0
RTTVar has grown to over 2.3 seconds, decreasing to 2.0
RTTVar has grown to over 2.3 seconds, decreasing to 2.0
RTTVar has grown to over 2.3 seconds, decreasing to 2.0
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00035s latency).
Other addresses for localhost (not scanned): 127.0.0.1
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
111/tcp   open  rpcbind
631/tcp   open  ipp
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.7 - 3.9
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at http://nmap.org/s
ubmit/
Nmap done: 1 IP address (1 host up) scanned in 2.00 seconds
```

In above snapshot shown, port 22, 25,111, etc. But port 8080 is not shown in that so, to check particular port give a command as

```
[root@10 zippyops]# netstat -anp | grep 8080
```

After that you go on install java, first you have to go cd /opt/

Then give a command to download java file from oracle,

```
# wget --no-cookies --no-check-certificate --header "Cookie:
gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie"
http://download.oracle.com/otn-pub/java/jdk/8u151-b12/e758a0de34e24606bca991d704f6dcfb/jdk-8u151-
linux-x64.tar.gz
```

To unzip the jdk file,

```
# tar xzf jdk-8u151-linux-x64.tar.gz
```

After extracting archive file use alternatives command to install it. Alternatives command is available in chkconfig package.

```
# cd /opt/jdk1.8.0_151/  
# alternatives --install /usr/bin/java java /opt/jdk1.8.0_151/bin/java 2  
# alternatives --config java
```

There are 3 programs which provide 'java'.

Selection	Command
*	/opt/jdk1.7.0_71/bin/java
+	/opt/jdk1.8.0_45/bin/java
3	/opt/jdk1.8.0_144/bin/java
4	/opt/jdk1.8.0_151/bin/java

Enter to keep the current selection[+], or type selection number: 4

At this point JAVA 8 has been successfully installed on your system. We also recommend to setup javac and jar commands path using alternatives,

```
# alternatives --install /usr/bin/jar jar /opt/jdk1.8.0_151/bin/jar 2  
# alternatives --install /usr/bin/javac javac /opt/jdk1.8.0_151/bin/javac 2  
# alternatives --set jar /opt/jdk1.8.0_151/bin/jar  
# alternatives --set javac /opt/jdk1.8.0_151/bin/javac
```

Check the installed Java version on your system using following command.

```
# java -version  
  
java version "1.8.0_151"  
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

## SETUP JAVA\_HOME, JRE\_HOME AND PATH ENVIRONMENT VARIABLES

```
# export JAVA_HOME=/opt/jdk1.8.0_151  
# export JRE_HOME=/opt/jdk1.8.0_151/jre  
# export PATH=$PATH:/opt/jdk1.8.0_151/bin:/opt/jdk1.8.0_151/jre/bin
```

Also put all above environment variables in /etc/environment file and cd /etc/rc.d/ vi rc.local for auto loading on system boot.

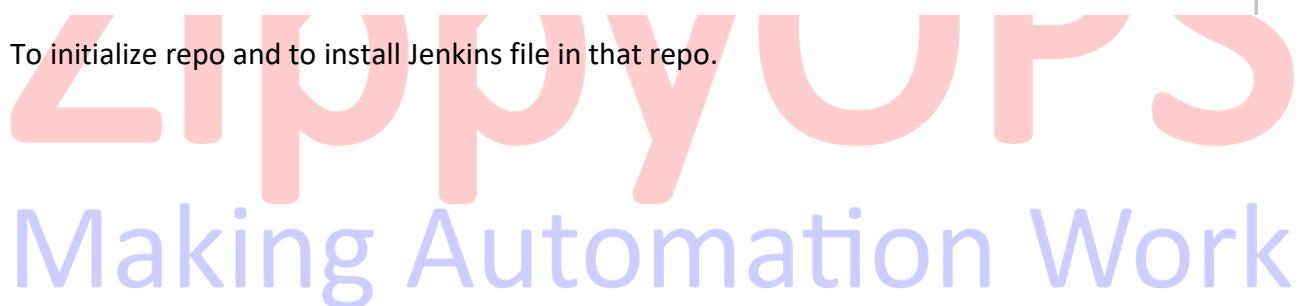
```
File Edit View Search Terminal Help  
#!/bin/bash  
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES  
#  
# It is highly advisable to create own systemd services or udev rules  
# to run scripts during boot instead of using this file.  
#  
# In contrast to previous versions due to parallel execution during bo  
# this script will NOT be run after all other services.  
#  
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensur  
# that this script will be executed during boot.  
  
touch /var/lock/subsys/local  
  
export JAVA_HOME=/opt/jdk1.8.0_151  
export JRE_HOME=/opt/jdk1.8.0_151/jre  
export PATH=$PATH:/opt/jdk1.8.0_151/bin:/opt/jdk1.8.0_151/jre/bin  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
"rc.local" 17L, 613C  
  
Now java is installed successfully.
```

## INSTALLING JENKINS:

Installing Jenkins is really as simple as fetching the package info to your environment along with the necessary cryptographic key.

```
zippyops@localhost:~ - □ ×  
File Edit View Search Terminal Help  
[root@10 ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.or  
g/redhat/jenkins.repo
```

To initialize repo and to install Jenkins file in that repo.



```
File Edit View Search Terminal Help
[root@10 ~]# sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
--2017-11-17 02:39:05--  http://pkg.jenkins-ci.org/redhat/jenkins.repo
Resolving pkg.jenkins-ci.org (pkg.jenkins-ci.org)... 52.202.51.185
Connecting to pkg.jenkins-ci.org (pkg.jenkins-ci.org)|52.202.51.185|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 71
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[=====] 71          --.-K/s   in 0s

2017-11-17 02:39:06 (11.7 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [71/7]

[root@10 ~]# rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
[root@10 ~]#
```

To import key from Jenkins website,

To install Jenkins to give a command,

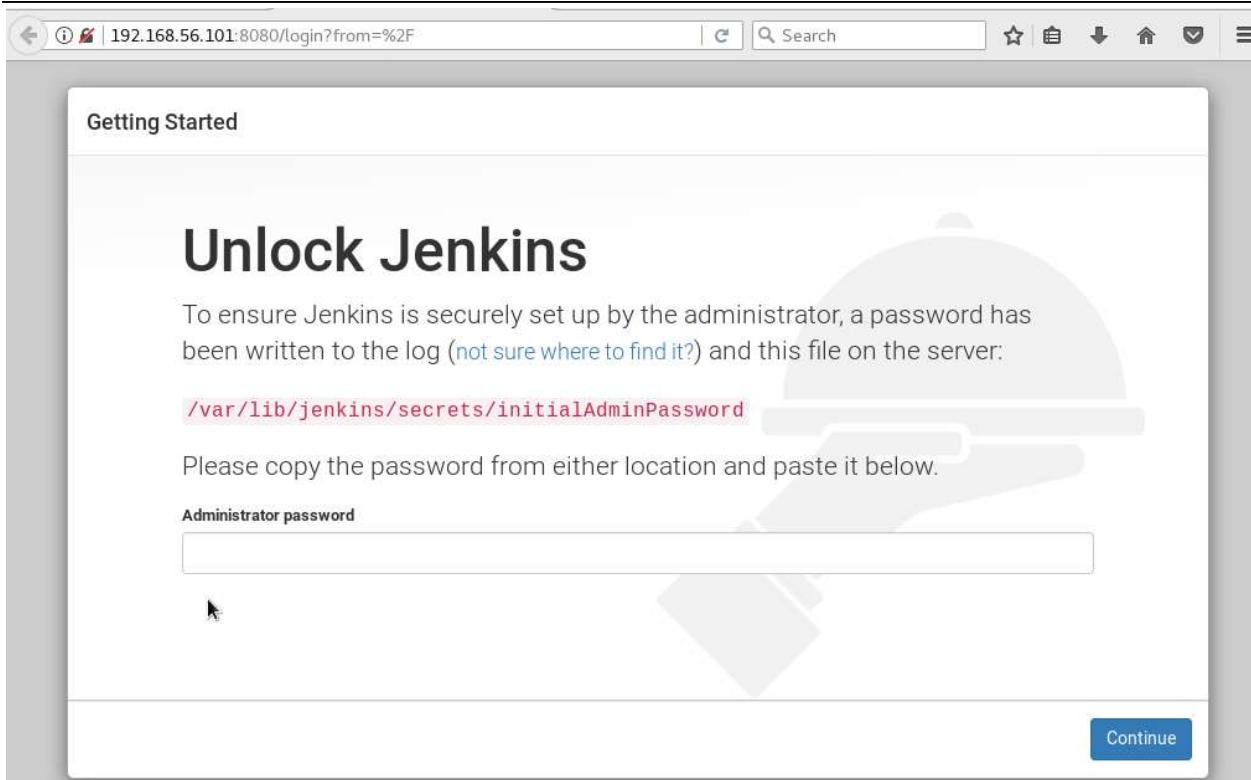
```
File Edit View Search Terminal Help
[root@10 ~]# yum install jenkins
Loaded plugins: fastestmirror, langpacks
jenkins
jenkins/primary_db
Loading mirror speeds from cached hostfile
 * base: mirrors.viethosting.com
 * elrepo: mirrors.thzhost.com
 * extras: mirror.ehost.vn
 * updates: mirrors.l23host.vn
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.90-1.1 will be installed
[root@10 ~]#
```

## ENABLE JENKINS AND START SERVICE

We begin with starting up the Jenkins service by running the command:

```
[root@10 ~]# systemctl enable jenkins.service
jenkins.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig jenkins on
[root@10 ~]# systemctl restart jenkins.service
```

Once you see the active (running) prompt it is time to start using the Web UI. Open your web browser and visit [http://ip\\_address:8080](http://ip_address:8080). Jenkins' dashboard is accessible via the public IP on port 8080.



For the first time you login, you would be asked to unlock it, the password so let's get it from the path that is shown in the screen.

```
[root@10 ~]# cat /var/lib/jenkins/secrets/initialAdminPassword  
bf912f4c76ee4f84b1b2c11150f2170f  
[root@10 ~]#
```

You can copy the output and use it to proceed with the Initial Set Up.

# ZippyOPS

Making Automation Work

**Customize Jenkins**

Plugins extend Jenkins with additional features to support many different needs.

**Install suggested plugins**  
Install plugins the Jenkins community finds most useful.

**Select plugins to install**  
Select and install plugins most suitable for your needs.

Here you may want to click on 'Select plugins' before installing them because Dashboard is not a part of 'Suggested Plugins', so click on the right button to select plugins.

**Organization and Administration (3/3)**

- Dashboard View** ↗ 9 ↘ Jenkins view that shows various cuts of build information via configured portlets.
- Folders Plugin** ↗ 1 ↘ This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.
- OWASP Markup Formatter Plugin** ↗ 3 ↘ Uses policy definitions to allow limited HTML markup in user-submitted text.

**Build Features (4/10)**

- build-name-setter** ↗ 11 ↘
- build timeout plugin** ↗ 20 ↘

Jenkins 2.78

Back Install

The dashboard view is not selected by default, you may want to select that and review other plugins as per the need of your organization. In this example, we would keep the list of other selected plugins as is, and click on install. It may take a while but eventually, all the plugins would get installed.

Getting Started

## Getting Started

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	⌚ build timeout plugin	⌚ Credentials Binding Plugin	** bouncycastle API Plugin Folders Plugin ** Structs Plugin ** JUnit Plugin OWASP Markup Formatter Plugin PAM Authentication plugin ** Windows Slaves Plugin ** Display URL API Jenkins Mailer Plugin LDAP Plugin ** Pipeline: Step API ** Script Security Plugin ** SCM API Plugin ** Pipeline: API ** Pipeline: Supporting APIs
⌚ Timestamper	⌚ Workspace Cleanup Plugin	⌚ Ant Plugin	⌚ Gradle Plugin	
⌚ Pipeline	⌚ GitHub Branch Source Plugin	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View Plugin	
⌚ Git plugin	⌚ Subversion Plug-in	⌚ SSH Slaves plugin	⌚ Matrix Authorization Strategy Plugin	
✓ PAM Authentication plugin	✓ LDAP Plugin	⌚ Email Extension Plugin	✓ Mailer Plugin	
⌚ Dashboard View	⌚ SSH Agent Plugin			** - required dependency

Jenkins 2.78

After this is done, you will be asked to set up your admin user and credentials. With that done, you are pretty much ready to import your existing software project or start a new one with Jenkins.

# ZippyU's

## Making Automation Work

## Getting Started

### Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.90

Continue as admin

Save and Finish

To give user name and password and email address to save and finish installation.

### START USING JENKINS:

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', and 'New View'. Below the navigation bar, there's a search bar and a user icon labeled 'zippyops | log out'. A button for 'ENABLE AUTO REFRESH' is also visible. The main content area features a large 'Welcome to Jenkins!' heading and a message encouraging users to 'create new jobs'. Two expandable sections, 'Build Queue' and 'Build Executor Status', are shown, both currently displaying 'No builds in the queue.' and '1 Idle' respectively. At the bottom of the page, a footer notes 'Page generated: Nov 17, 2017 3:31:43 AM EST REST API Jenkins ver. 2.90'.

## MANAGE JENKINS:

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), 'My Views', 'Credentials', and 'New View'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main content area is titled 'Manage Jenkins' and lists several configuration options with icons: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), 'System Information' (monitor icon), 'System Log' (clipboard icon), and 'Load Statistics' (graph icon).

This screen lets you configure different aspects of your Jenkins server. Each link on this page takes you to a dedicated configuration screen, where you can manage different parts of the Jenkins server.

## CONFIGURE SYSTEM:

This is where you manage paths to the various tools you use in your builds, such as JDKs, and versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. Many of the plugins that you install will also need to be configured here—Jenkins will add the fields dynamically when you install the plugins.

## RELOAD CONFIGURATION FROM DISK:

As we saw in the previous chapter, Jenkins stores all its system and build job configuration details as XML files stored in the Jenkins home directory (see [The Jenkins Home Directory](#)). It also stores all of the build history in the same directory. 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. This process can be a little slow if there is a lot of build history, as Jenkins loads not only the build configurations but also all of the historical data as well.

## MANAGE PLUGINS:

One of the best features of Jenkins is its extensible architecture. There is a large ecosystem of third-party open source plugins available, enabling you to add extra features to your build server, from support for different SCM tools such as Git, Mercurial or ClearCase, to code quality and code coverage metrics reporting. We will be looking at many of the more popular and useful plugins throughout this book. Plugins can be installed, updated and removed through the Manage Plugins screen. Note that removing plugins needs to be done with some care, as it can sometimes affect the stability of your Jenkins instance—we will look at this in more detail in Migrating Build Jobs.

## SYSTEM INFORMATION:

This screen displays a list of all the current Java system properties and system environment variables. Here, you can check exactly what version of Java Jenkins is running in, what user it is running under, and so forth. You can also check that Jenkins is using the correct environment variable settings. Its main use is for troubleshooting, so that you can make sure that your server is running with the system properties and variables you think it is.

## 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.

You can also subscribe to RSS feeds for various levels of log messages. For example, as a Jenkins administrator, you might want to subscribe to all the ERROR and WARNING log messages.

## LOAD STATISTICS:

Jenkins keeps track of how busy your server 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 you an idea of whether you need to add extra capacity or extra build nodes to your infrastructure.

## 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, it is mainly useful for plugin developers and the like.

## MANAGE NODES:

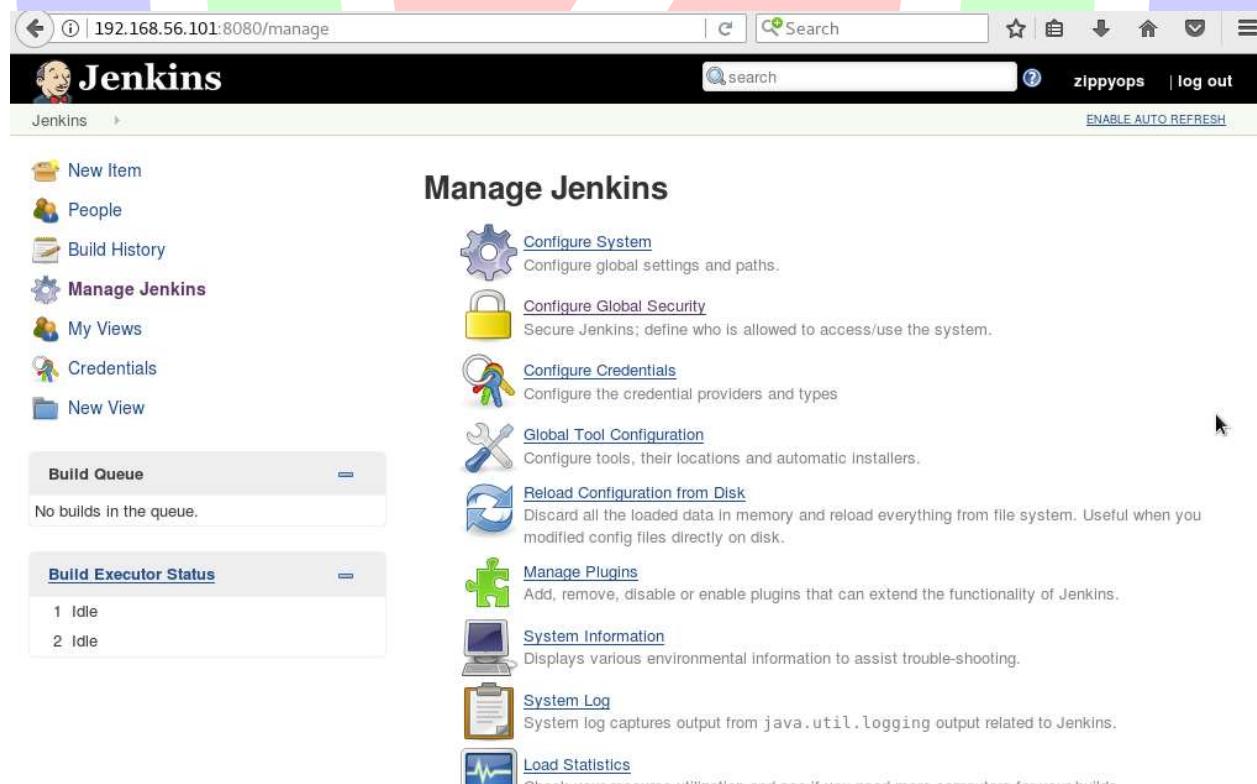
Jenkins handles parallel and distributed builds well. In this screen, you can configure how many builds you want. Jenkins runs simultaneously, and, if you are using distributed builds, set up build nodes. A build node is another machine that Jenkins can use to execute its builds.

## PREPARE FOR SHUTDOWN:

If you 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, you will be able to shut down Jenkins cleanly.

## CONFIGURE SECURITY IN JENKINS:

Click on Manage Jenkins and choose the ‘Configure Global Security’ option.

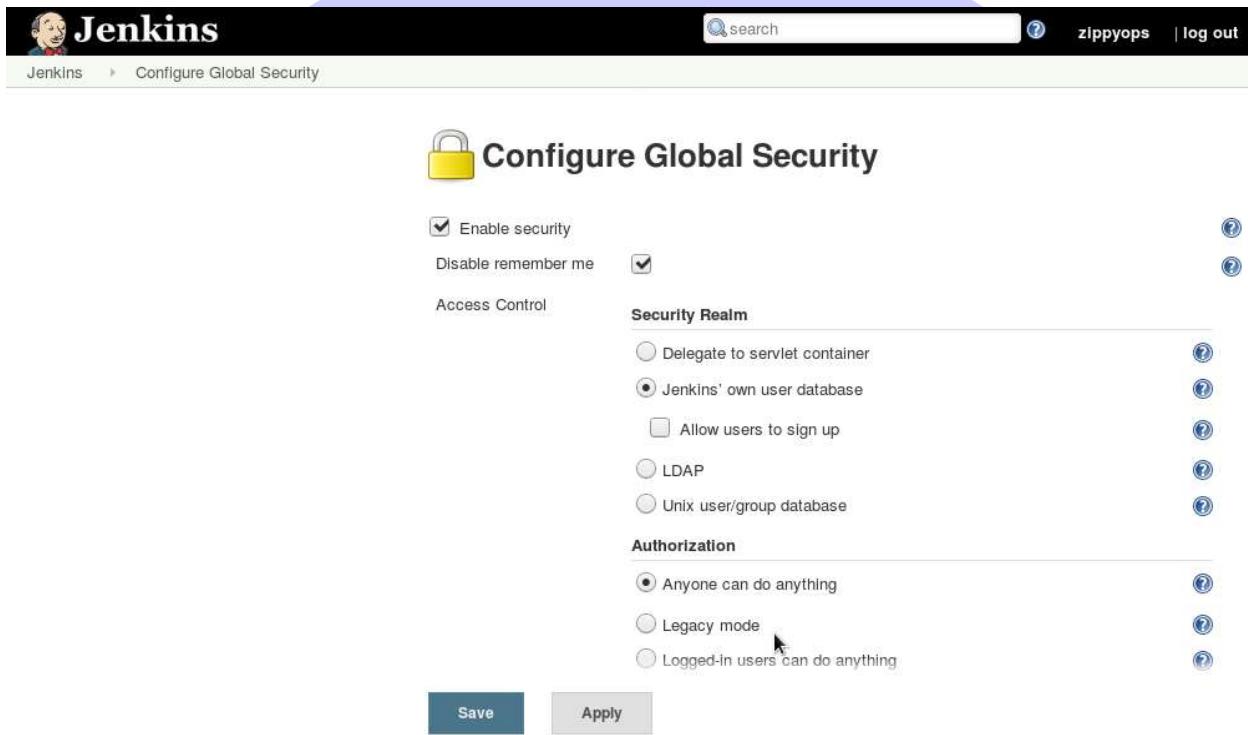


The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is currently selected), 'My Views', 'Credentials', and 'New View'. Below this are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main content area is titled 'Manage Jenkins' and lists several configuration options: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon, highlighted with a red box), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), 'System Information' (monitor icon), 'System Log' (clipboard icon), and 'Load Statistics' (heartbeat icon). The 'Configure Global Security' link is underlined and has a tooltip: 'Secure Jenkins; define who is allowed to access/use the system.'

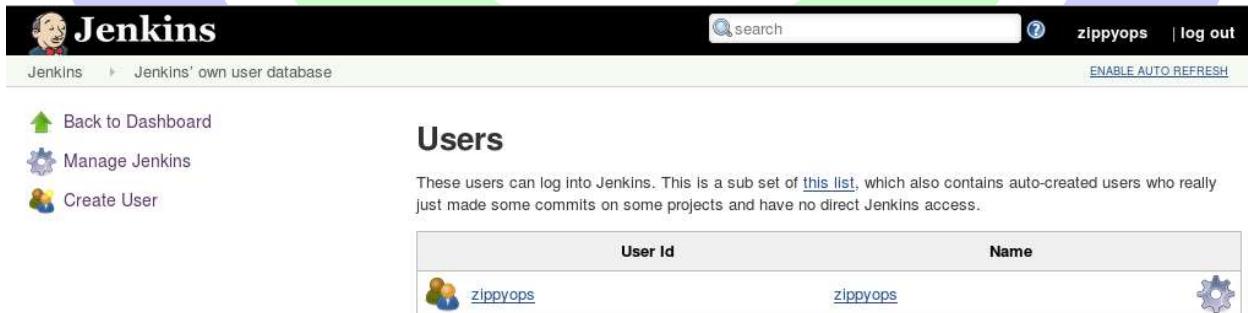
Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain its own database of users, so in the Security Realm, choose the option of ‘Jenkins’ own user database’.

# Making Automation Work

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.



First you can click manage Jenkins, then entered to manager user.



There are only one user in Jenkins, for example I am going to create new for it, that is test user.

# ZippyOPS

## Making Automation Work

Start creating other users for the system. As an example, we are just creating another user called 'test'.

Now new user has been created.

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'

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.

## MANAGE CREDENTIALS:

The screenshot shows the Jenkins interface with the 'Credentials' section selected. A table lists a single global credential:

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	c6d31dc1-9dfe-43d3-b84a-a04f519cf2de	<a href="#">jenkins (Jenkins System Service Account)</a>

Below the table, a section titled 'Stores scoped to Jenkins' shows:

P	Store ↓	Domains
	<a href="#">Jenkins</a>	(global)

To add global credential, first we have to run ssh key for Jenkins,

So we have edit passwd file to bash ,

```
zippyops@localhost:~  
File Edit View Search Terminal Help  
[root@10 ~]# vi /etc/passwd
```

# ZippyOPS

## Making Automation Work

```
File Edit View Search Terminal Help
saslauth:x:996:76:Saslauthd user:/run/saslauthd:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
chrony:x:995:993::/var/lib/chrony:/sbin/nologin
qemu:x:107:107:qemu user:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcscd daemon:/dev
/null:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
geoclue:x:994:991:User for geoclue:/var/lib/geoclue:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
radvd:x:75:75:radvd user:/:/sbin/nologin
setroubleshoot:x:993:990::/var/lib/setroubleshoot:/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:992:987::/run/gnome-initial-setup:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
zippyops:x:1000:1000:zippyops:/home/zippyops:/bin/bash
vboxadd:x:991:1::/var/run/vboxadd:/bin/bash
jenkins:x:990:985:Jenkins Automation Server:/var/lib/jenkins:/bin/false
-- TNSFRT --
```

```
File Edit View Search Terminal Help
saslauth:x:996:76:Saslauthd user:/run/saslauthd:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
chrony:x:995:993::/var/lib/chrony:/sbin/nologin
qemu:x:107:107:qemu user:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcscd daemon:/dev
/null:/sbin/nologin
usbmuxd:x:113:113:usbmuxd user:/:/sbin/nologin
geoclue:x:994:991:User for geoclue:/var/lib/geoclue:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
radvd:x:75:75:radvd user:/:/sbin/nologin
setroubleshoot:x:993:990::/var/lib/setroubleshoot:/sbin/nologin
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:992:987::/run/gnome-initial-setup:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
zippyops:x:1000:1000:zippyops:/home/zippyops:/bin/bash
vboxadd:x:991:1::/var/run/vboxadd:/bin/bash
jenkins:x:990:985:Jenkins Automation Server:/var/lib/jenkins:/bin/bash
```

## Making Automation Work

To change false to bash ,

After that we can edit /etc/sudoers file ,

```
File Edit View Search Terminal Help  
[root@10 ~]# vi /etc/sudoers  
  
File Edit View Search Terminal Help  
  
#  
# Adding HOME to env_keep may enable a user to run unrestricted  
# commands via sudo.  
#  
# Defaults env_keep += "HOME"  
  
Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin  
  
## Next comes the main part: which users can run what software on  
## which machines (the sudoers file can be shared between multiple  
## systems).  
## Syntax:  
##  
##       user      MACHINE=COMMANDS  
##  
## The COMMANDS section may have other options added to it.  
##  
## Allow root to run any commands anywhere  
root    ALL=(ALL)        ALL  
jenkins ALL=(ALL)        NOPASSWD: ALL  
  
## Allows members of the 'sys' group to run networking, software,  
  
There to be add Jenkins user too in that and save it as no passwd too.  
Then you can generate ssh key in command line as ,
```

# ZippyOPS

## Making Automation Work

```
zippyops@localhost:~  
File Edit View Search Terminal Help  
[root@10 ~]# ssh-keygen -jenkins  
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):  
/root/.ssh/id_rsa already exists.  
Overwrite (y/n)? y  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /root/.ssh/id_rsa.  
Your public key has been saved in /root/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:KHy0iXWpvCG2bhM/USfizSJ0JJxLKF2BITGEXm3B48g root@10.0.2.15  
The key's randomart image is:  
+--- [RSA 2048] ---+  
|**.=o.  
|+oB..=  
|0.0++ .  
.oEo.oo.  
. o==oS  
o++Oo  
++*o.  
.0000  
00...  
+--- [SHA256] ---+
```

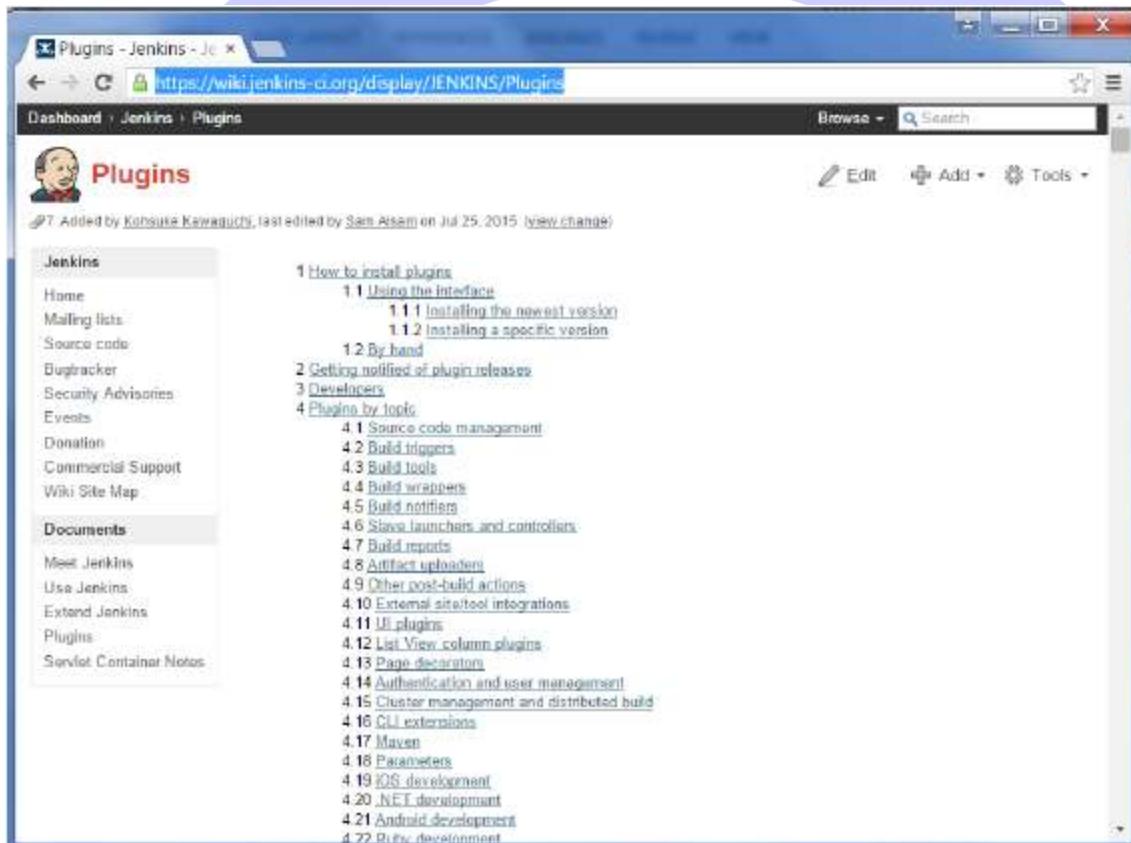
After that go to localhost:8080 ,

The screenshot shows the Jenkins Global Credentials configuration page. A new credential is being added for the 'jenkins' user. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains 'jenkins'. Under 'Private Key', the 'Enter directly' option is selected, and the key value is pasted from the terminal output above. The key value is a long string of characters starting with 'AAAAB3NzaC1yc2EAAAQABAAQC4l...' and ending with 'Bi1e4t7IAeokxOihd/OPrCQbiTk0EvF8eEwlCs+NtIWY1IOEWfE4blZN6Ps2J'. Below the key input, there are three radio button options: 'From a file on Jenkins master', 'From the Jenkins master ~/.ssh', and 'Passphrase', which is currently filled with a series of dots. The 'ID' field is set to 'c6d31dc1-9dfe-43d3-b84a-a04f519cf2de', and the 'Description' field is 'Jenkins System Service Account'. At the bottom right, there is a 'Save' button.

I have to add my ssh key directly to global security and credential .

## PLUGIN MANAGEMENT:

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

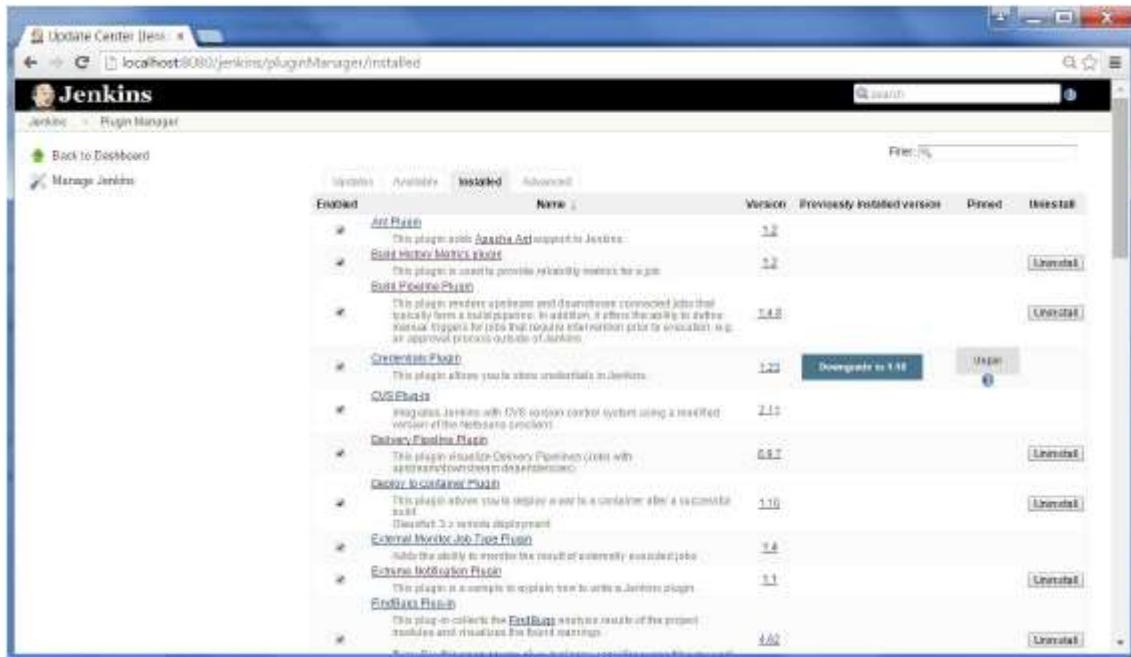


Let's look at some other maintenance tasks with regards to plugins

## 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.

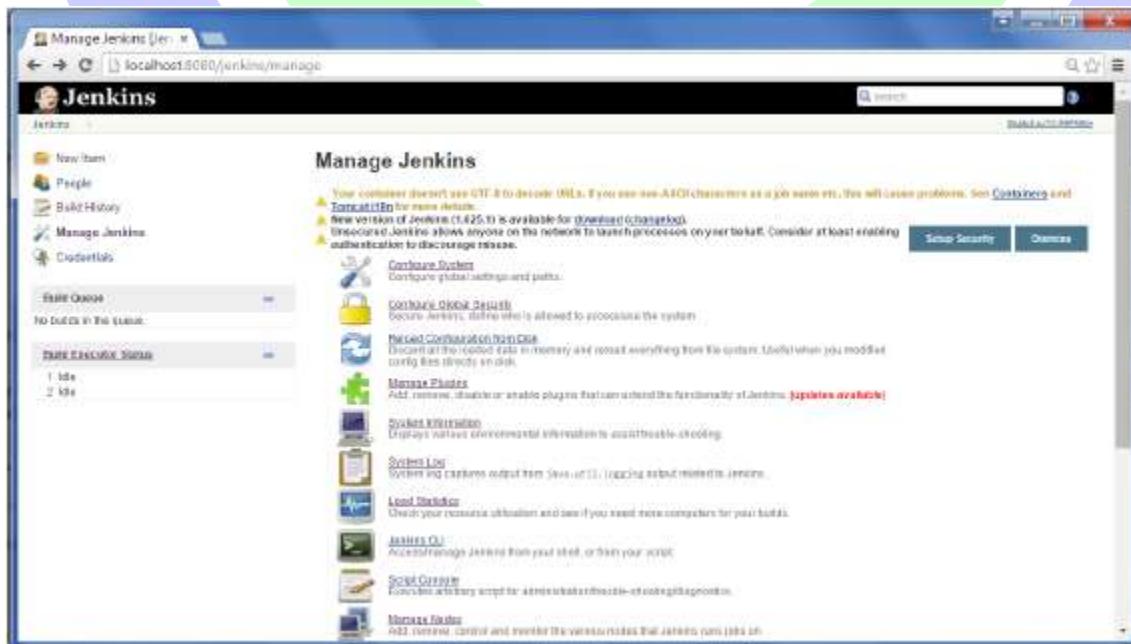
Making Automation Work



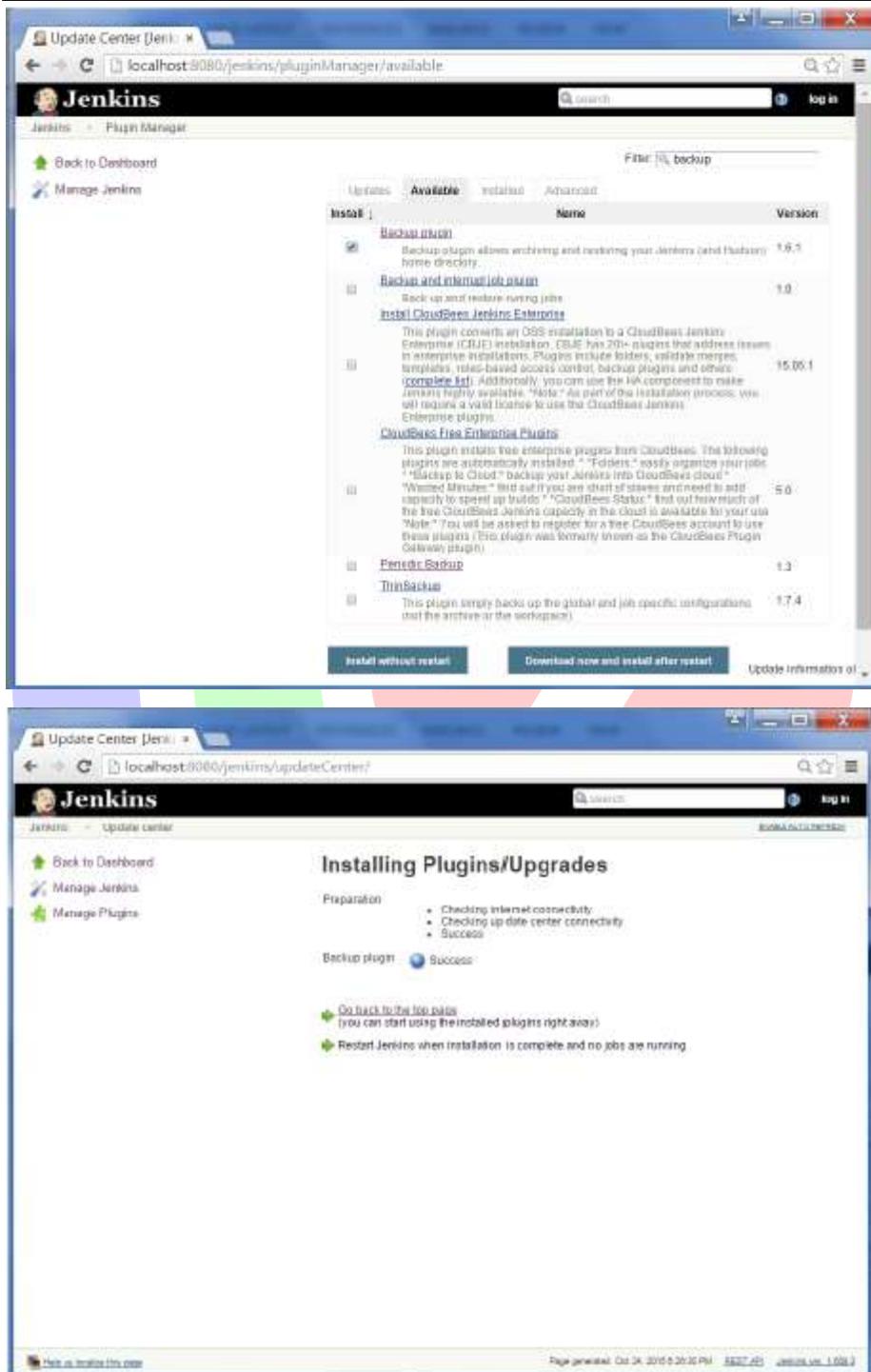
## BACKUP PLUGIN:

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.

Click on Manage Jenkins and choose the ‘Manage Plugins’ option.



In the available tab, search for ‘Backup Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance



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

# Making Automation Work



Click on Setup.

The screenshot shows the Jenkins Backup manager page. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area features a large blue cube icon labeled "Backup manager". Below it are three buttons: "Setup" (highlighted in red), "Backup Hudson configuration", and "Restore Hudson configuration".

Building Automation Work

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.

To run command line to create directory for backup,

```
# mkdir Jenkins_backup  
# chown Jenkins:Jenkins /Jenkins_backup/
```

To create back up directory.

The screenshot shows the Jenkins interface with the 'Backup manager' page selected. On the left, there is a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Credentials', and 'New View'. Below these are sections for 'Build Queue' (no builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Backup config files' and contains 'Backup configuration' settings. It includes fields for 'Hudson root directory' (set to '/var/lib/jenkins'), 'Backup directory' (set to '/jenkins\_backup'), 'Format' (set to 'tar.gz'), 'File name template' (set to 'backup\_@date@@extension@'), and 'Custom exclusions'. There are also checkboxes for 'Verbose mode', 'Configuration files (.xml) only', and 'No shutdown'. Below this is a 'Backup content' section with a checkbox for 'Backup job workspace' and fields for 'Includes' and 'Excludes'.

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

# ZippyOPS

## Making Automation Work

The screenshot shows the Jenkins Backup manager interface. 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, 2 idle). The main area has a title 'Backup manager' with a blue cube icon. It contains three buttons: 'Setup' (with a wrench icon), 'Backup Hudson configuration' (with a blue arrow icon), and 'Restore Hudson configuration' (with a grey arrow icon). A large purple circle graphic is overlaid on the right side of the page.

The next screen will show the status of the backup

This screenshot shows the same Jenkins Backup manager interface as above, but with a red banner at the top stating 'Jenkins is going to shut down'. Below the banner, a log window displays the following text:

```
[INFO] Backup started at [10/24/15 19:19:31]
[INFO] Setting hudson in shutdown mode to avoid files corruptions.
[INFO] Waiting all jobs end...
[INFO] Number of running jobs detected : 0
[INFO] All jobs finished.
[INFO] Full backup file name : D:\Backup\beckm_20151024_1919.zip
[INFO] Saved file(s) : 911
[INFO] Number of errors : 0
[INFO] Cancel hudson shutdown mode
[INFO] Backup end at [10/24/15 19:19:38]
[INFO] [19.524s]
```

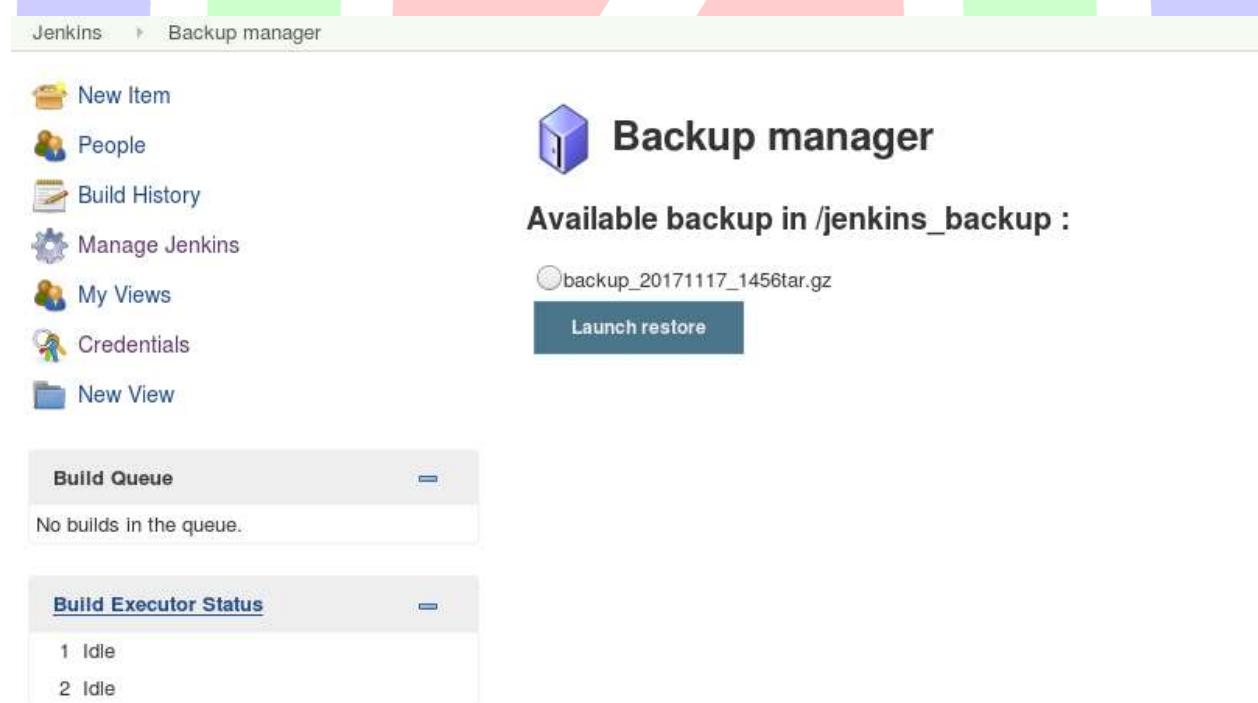
A large green circle graphic is overlaid on the right side of the page.

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

# Making Automation Work



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.



**ZippyUrs**  
Making Automation Work

## CONFIGURE JENKINS SLAVE AND CONNECT TO MASTER:

On Jenkins master go to Manage Jenkins > Manage Nodes

The screenshot shows the Jenkins 'Manage Nodes' page. On the left sidebar, there are links: Back to Dashboard, Manage Jenkins, New Node, and Configure. The main area displays a table of nodes. One row is highlighted for the 'master' node, which is listed as 'Linux (amd64)', 'In sync', with 13.22 GB free disk space, 2.05 GB free swap space, and 38 min free temp. Below the table is a 'Refresh status' button. Under the table, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle).

Select New Node and enter node name. select Permanent Agent -> Press OK.

The screenshot shows the Jenkins 'New Node' configuration page. The 'Node name' field is set to 'slave'. A checkbox labeled 'Permanent Agent' is checked. A tooltip explains: 'Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.' At the bottom right is an 'OK' button.

On the next page, fill in the following fields:

- Set a number of executors (one or more) as needed. Default is 1.
- Set a Remote FS Root, a home directory for the master on the slave machine. i.e Linux Slave: "/opt/jenkins/"
- Set Usage to "Use this node as much as possible".
- Choose Launch method as "Launch agent via Java Web Start".
- Set Availability, -> Keep this slave online as much as possible.
- Press Save.

# ZippyOPS

## Making Automation Work

Jenkins > Nodes >

<a href="#">Back to Dashboard</a>	Name	slave
<a href="#">Manage Jenkins</a>	Description	
<a href="#">New Node</a>	# of executors	1
<a href="#">Configure</a>	Remote root directory	/opt/jenkins/
<b>Build Queue</b>	Labels	
No builds in the queue.	Usage	Use this node as much as possible
<b>Build Executor Status</b>	Launch method	Launch agent via execution of command on the master
1 Idle 2 Idle	Launch command	<b>No launch command specified</b>
	Availability	Keep this agent online as much as possible

**Node Properties**

Now for connecting to slave to the master by

- Open a browser on the slave machine and go to the Jenkins - master server url (<http://yourjenkinsmaster:8080>).
- Go to Manage Jenkins > Manage Nodes, click on the newly -created slave machine.
- Connect agent to Jenkins by one of these ways.
- Launch Agent from Browser.
- Run from agent command line.
- We have chosen second option, login to agent node -> download the slave.jar file from Jenkins Master UI to slave machine then while executing the command, please specify download path of slave.jar file.

Jenkins > Nodes > slave

<a href="#">Back to List</a>	<a href="#">Status</a>	<a href="#">Delete Agent</a>	<a href="#">Configure</a>	<a href="#">Build History</a>	<a href="#">Load Statistics</a>	<a href="#">Log</a>
<b>Build Executor Status</b>						

**Agent slave**

**Mark this node temporarily offline**

**This agent is offline because Jenkins failed to launch the agent process on it. See log for more details**

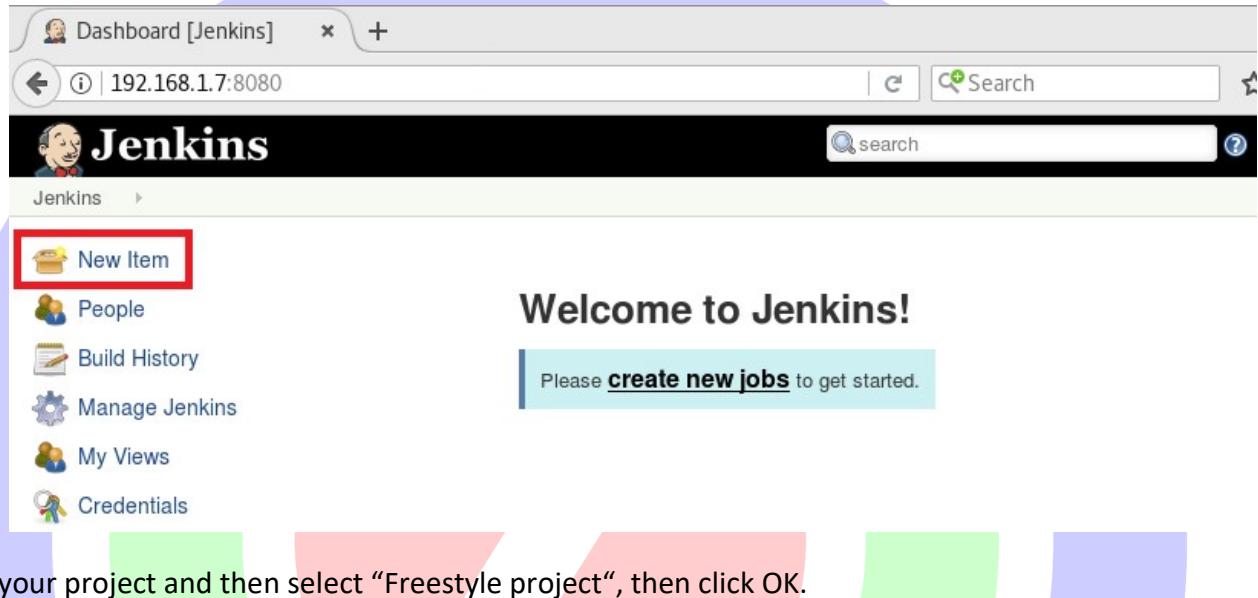
[Launch agent](#)

**Projects tied to slave**

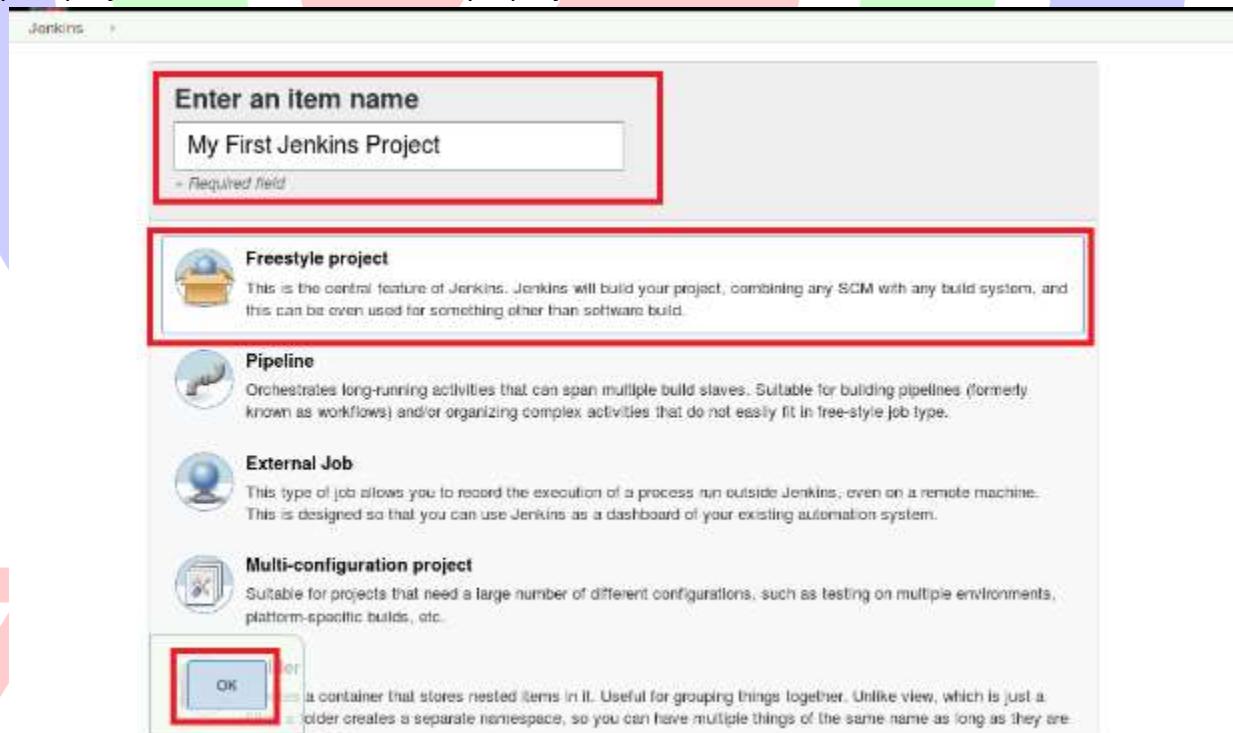
None

Create your first Jenkins project:

Let's create a sample build using shell command. To do that, go to Jenkins Web Portal → New Item or create new jobs.



Name your project and then select "Freestyle project", then click OK.



On the project configuration page, scroll down and choose "Execute shell" in the "Build" section

# Making Automation Work

Enter a shell command to execute while building the project.



Click on “SAVE”

The project is now ready to build. Click on “Build Now” in the left pane to build the project.

# ZippyOPS

Making Automation Work

Click on "Console Icon" to see the output of your build.

Jenkins > My First Jenkins Project

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure

Build History trend =

find

RSS for all RSS for failures

Workspace

Recent Changes

Permalinks

# Making Automation Work

Console output:

Jenkins > My First Jenkins Project > #1

Back to Project | Status | Changes | **Console Output** | View as plain text | Edit Build Information | Delete Build

## Console Output

Started by user  
Building in workspace /var/lib/jenkins/workspace/My First Jenkins Project  
[My First Jenkins Project] \$ /bin/sh -xe /tmp/jenkins8568335340419869315.sh  
+ echo This is My First Jenkins Project  
This is My First Jenkins Project  
Finished: SUCCESS

## SET UP JENKINS ACCESS TO GITHUB :

Back in the main Jenkins dashboard, click Manage Jenkins in the left hand menu:



In the list of links on the following page, click Configure System:

## Manage Jenkins



### Configure System

Configure global settings and paths.



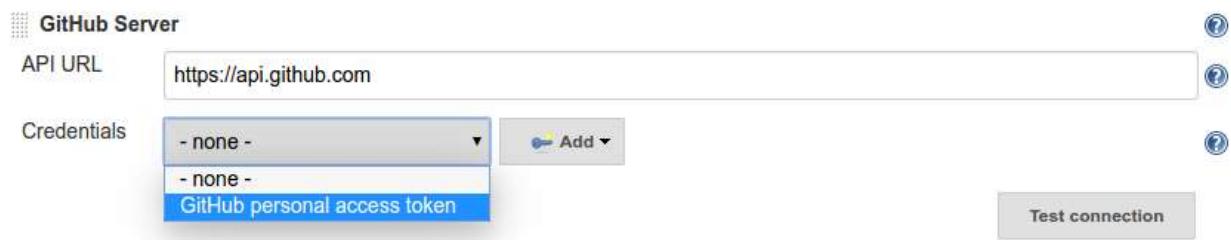
### Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.

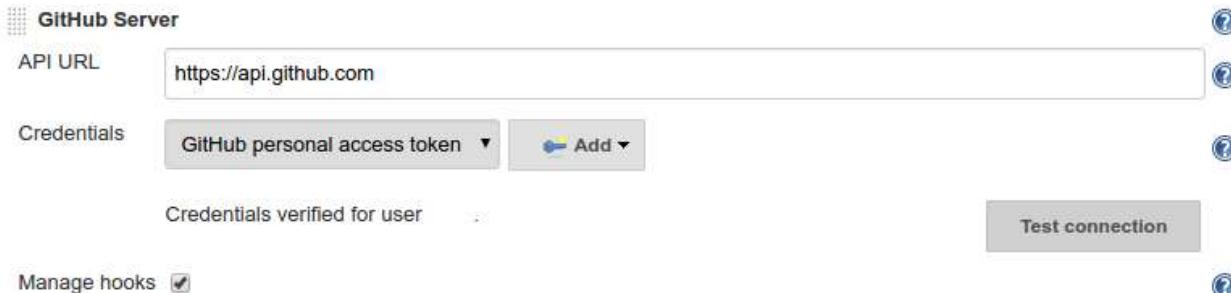
Scroll through the options on the next page until you find the GitHub section. Click the Add GitHub Server button and then select GitHub Server:



The section will expand to prompt for some additional information. In the Credentials drop down menu, select your GitHub personal access token that you added in the last section:



Click the Test connection button. Jenkins will make a test API call to your account and verify connectivity:



When you are finished, click the Save button to implement your changes.

# ZippyOPS

## Making Automation Work

## CREATE A NEW PIPELINE IN JENKINS:

Next, we can set up Jenkins to use the GitHub personal access token to watch our repository.

Back in the main Jenkins dashboard, click New Item in the left hand menu:



Enter a name for your new pipeline in the Enter an item name field. Afterwards, select Pipeline as the item type:

The screenshot shows a 'Enter an item name' dialog box. In the input field, the word 'Hello' is typed. Below the input field, a note says '» Required field'. Below the dialog, two item types are listed: 'Freestyle project' and 'Pipeline'. The 'Pipeline' item type is selected, indicated by a blue border around its icon and text.

**Enter an item name**

Hello

» Required field

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

**Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Click the OK button at the bottom to move on.

On the next screen, check the GitHub project box. In the Project url field that appears, enter your project's GitHub repository URL.

The screenshot shows a configuration dialog for a GitHub project. At the top, there is a checkbox labeled 'GitHub project' which is checked. Below it, there is a 'Project url' field containing the URL 'https://github.com/YOUR\_GITHUB\_NAME\_HERE/hello\_hapi/'. There is also an 'Advanced...' button and a small 'M' icon on the left.

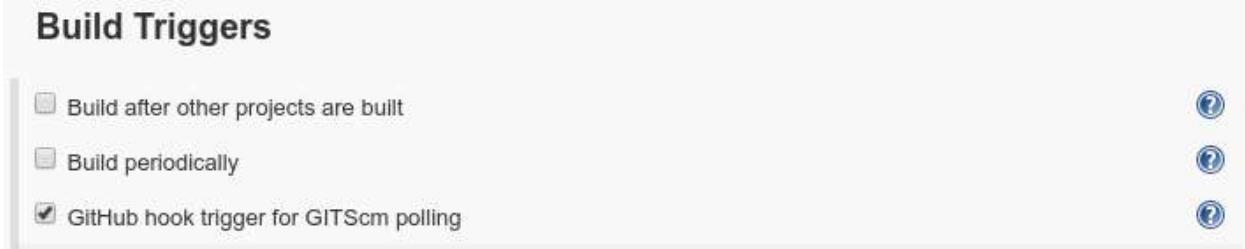
GitHub project

Project url

https://github.com/YOUR\_GITHUB\_NAME\_HERE/hello\_hapi/

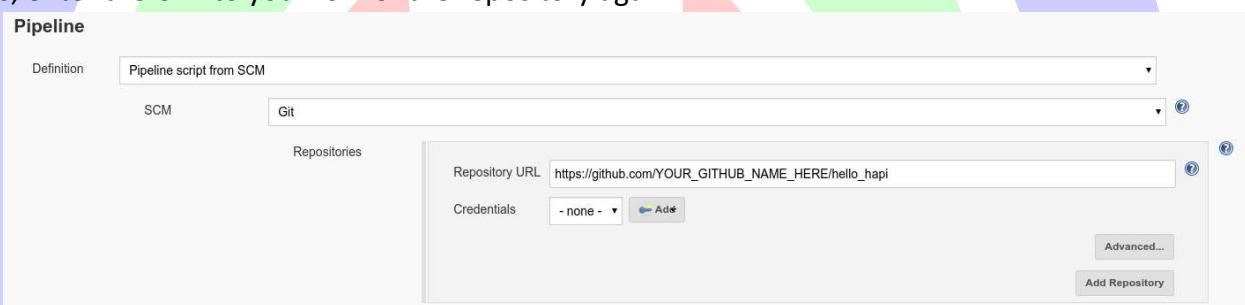
Advanced...

Next, in the Build Triggers section, check the GitHub hook trigger for GITScm polling box:



In the Pipeline section, we need to tell Jenkins to run the pipeline defined in the Jenkinsfile in our repository. Change the Definition type to Pipeline script from SCM.

In the new section that appears, choose Git in the SCM menu. In the Repository URL field that appears, enter the URL to your fork of the repository again:



When you are finished, click the Save button at the bottom of the page.

In your pipeline's main page, click Build Now in the left hand menu:



A new build will be scheduled. In the Build History box in the lower left corner, a new build should appear in a moment. Additionally, a Stage View will begin to be drawn in the main area of the interface. This will track the progress of your testing run as the different stages are completed:

The screenshot shows the Jenkins pipeline interface. On the left, there's a sidebar with icons for Full Stage View, GitHub, Pipeline Syntax, and GitHub Hook Log. Below that is the Build History section, which has a search bar with 'find' and a list item '#1'. At the bottom of the history section are links for 'RSS for all' and 'RSS for failures'. To the right of the sidebar is the Stage View, which displays four stages: Declarative: Checkout SCM (409ms), Declarative: Agent Setup (724ms), Build (2s), and Test (2s). Below the Stage View is a section titled 'Permalinks' containing a bulleted list of links to the last build, stable build, successful build, and completed build.

- [Last build \(#1\), 1 min 53 sec ago](#)
- [Last stable build \(#1\), 1 min 53 sec ago](#)
- [Last successful build \(#1\), 1 min 53 sec ago](#)
- [Last completed build \(#1\), 1 min 53 sec ago](#)

In the Build History box, click on the number associated with the build to go to the build detail page. From here, you can click the Console Output button in the left hand menu to see details of the steps that were run:

The screenshot shows the Jenkins build detail page for build #1. On the left, there's a sidebar with various options: Back to Project, Status, Changes, Console Output (which is selected and highlighted in blue), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, Docker Fingerprints, Replay, and Pipeline Steps. The main content area is titled 'Console Output' and shows the Jenkins pipeline logs. The logs detail the steps taken during the build, including cloning the repository from GitHub, fetching upstream changes, and checking out the specified revision.

```
Started by user
Obtained Jenkinsfile from git https://github.com/
[Pipeline] node
Running on master in /var/lib/jenkins/workspace/Hello Hapi
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/
Fetching upstream changes from https://github.com/
# timeout=10
> git --version # timeout=10
> git fetch --tags --progress https://github.com/
+refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision f45d1e0e71a4d4e9058f1f249b6be5c113d98415 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f f45d1e0e71a4d4e9058f1f249b6be5c113d98415
First time build. Skipping changelog.
[Pipeline]
```

Click the Back to Project item in the left hand menu when you are finished in order to return to the main pipeline view.

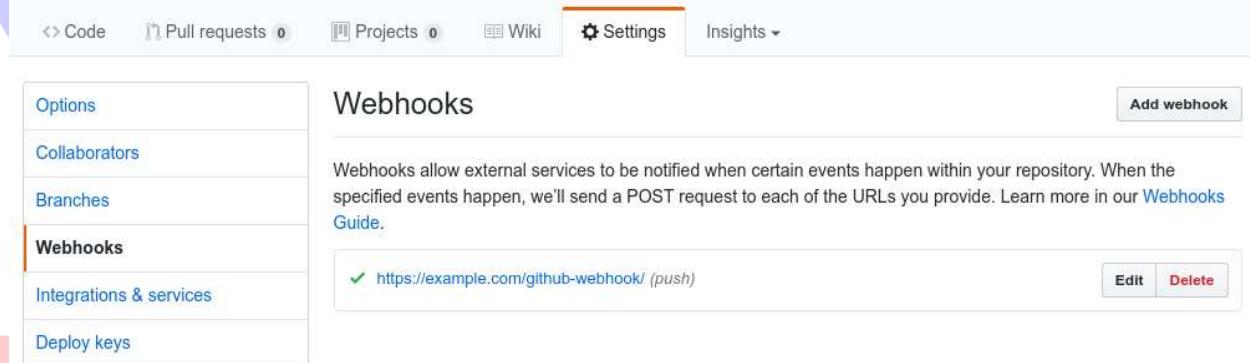
Now that we've built the project once, we can have Jenkins create the webhooks for our project. Click Configure in the left hand menu of the pipeline:

# Making Automation Work

-  Back to Dashboard
-  Status
-  Changes
-  Build Now
-  Delete Pipeline
-  Configure
-  Full Stage View
-  GitHub
-  Pipeline Syntax
-  GitHub Hook Log

No changes are necessary on this screen, just click the Save button at the bottom. Now that the Jenkins has information about the project from the initial build process, it will register a webhook with our GitHub project when you save the page.

You can verify this by going to your GitHub repository and clicking the Settings button. On the next page, click Webhooks from the side menu. You should see your Jenkins server webhook in the main interface:



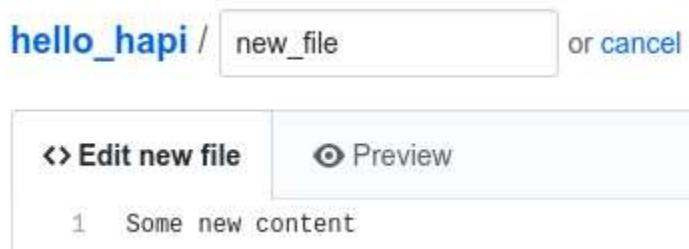
The screenshot shows the GitHub repository settings page. The top navigation bar includes links for Code, Pull requests, Projects, Wiki, Settings (which is highlighted in orange), and Insights. The left sidebar menu lists Options, Collaborators, Branches, Webhooks (which is selected and highlighted in orange), Integrations & services, and Deploy keys. The main content area is titled "Webhooks" and contains the following text: "Webhooks allow external services to be notified when certain events happen within your repository. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#)." Below this text is a table with one row, showing a webhook entry: "https://example.com/github-webhook" (push). To the right of the URL are "Edit" and "Delete" buttons.

Now, when you push new changes to your repository, Jenkins will be notified. It will then pull the new code and retest it using the same procedure.

To approximate this, in our repository page on GitHub, you can click the Create new file button to the left of the green Clone or download button:



On the next page, choose a filename and some dummy contents:



Click the Commit new file button at the bottom when you are finished.

If you return to your Jenkins interface, you will see a new build automatically started:

Declarative: Checkout SCM	Declarative: Agent Setup	Build	Test
410ms	832ms	2s	2s
411ms	941ms	2s	2s
409ms	724ms	2s	2s

**Build History**

- #2 2017 8:08 PM
- #1 2017 7:53 PM

**Permalinks**

- Last build (#1), 15 min ago
- Last stable build (#1), 15 min ago
- Last successful build (#1), 15 min ago
- Last completed build (#1), 15 min ago

You can kick off additional builds by making commits to a local copy of the repository and pushing it back up to GitHub.

## INSTALL APACHE MAVEN ON CENTOS 7 :

First you have got to “ /opt/ ” directory,

Give a command to download maven for ,

```
# wget http://www-eu.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
```

To Extract it using following command :

```
# tar zxvf apache-maven-3.3.9-bin.tar.gz
```

Move the extracted contents using following command:

```
# mv apache-maven-3.3.9/ /opt/maven
```

Make a symbolink to maven/bin folder as shown below :

```
# ln -s /opt/maven/bin/mvn /usr/bin/mvn
```

Create a file and add the content as below to setup Maven environment variable:

```
# vim /etc/profile.d/maven.sh
```

Add the following contents:

```
#!/bin/bash  
MAVEN_HOME=/opt/maven  
PATH=$MAVEN_HOME/bin:$PATH  
export PATH MAVEN_HOME  
export CLASSPATH=.
```

Make it executable using the following command.

```
# chmod +x /etc/profile.d/maven.sh
```

Then, set the environment variables permanently by running the following command:

```
# source /etc/profile.d/maven.sh
```

You can check version using following command:

```
# mvn -version
```

Check the environment variables:

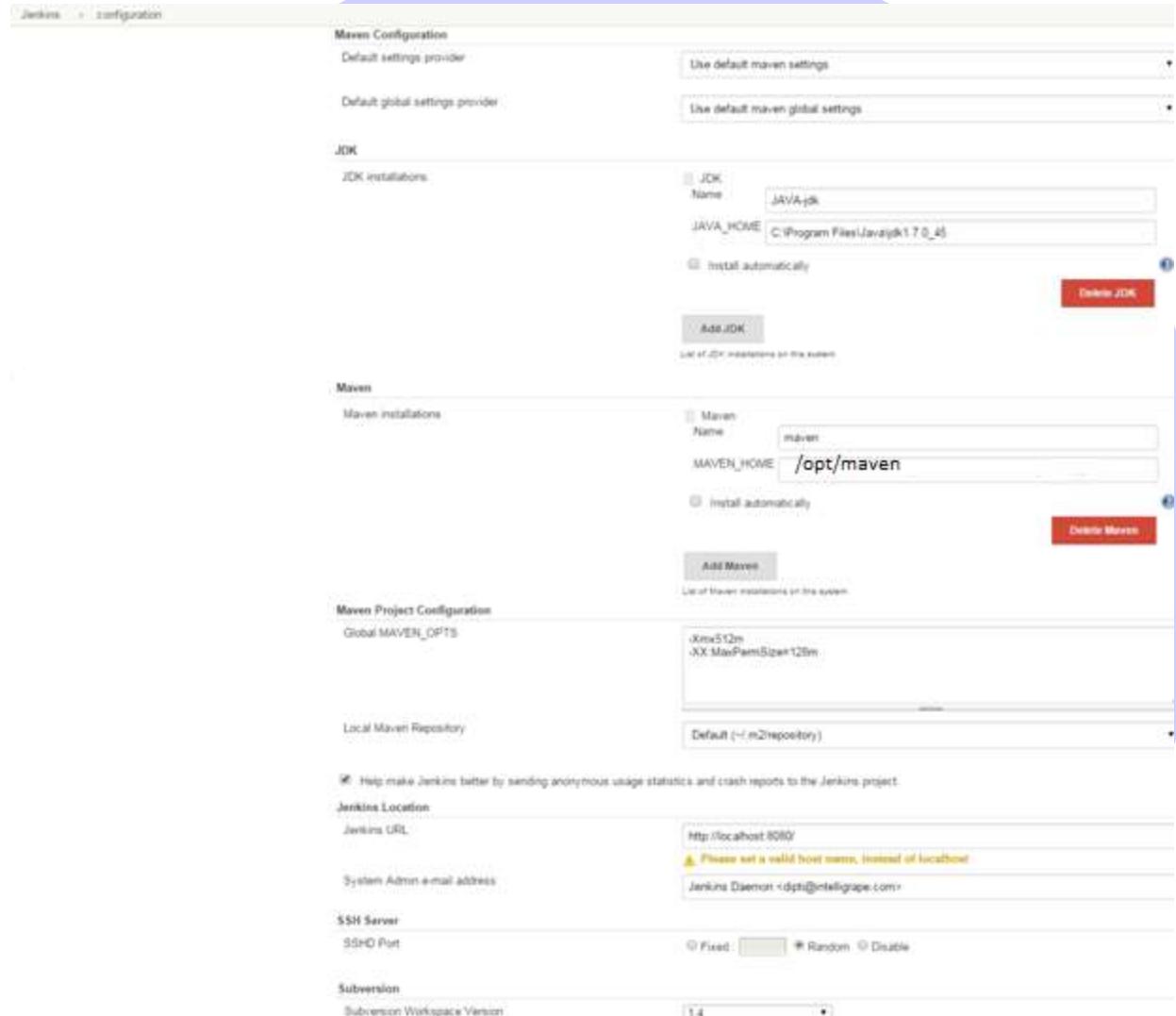
```
# echo $MAVEN_HOME
```

## SETTING UP JENKINS AND MAVEN:

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



Then, click on 'Global tool configuration' from the right hand side.



In the global tools configuration, scroll down till you see the Maven section and then click on the 'Add Maven' button.

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.

# Making Automation Work

Enter an item name

Mavenproject

» Required field

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

**Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

If you want to create a new item from other existing, you can use this option:

Jenkins

Mavenproject

search zippyops

General Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Build Settings

Maven project name: Mavenproject

Description: This is my maven project

(Plain text) Preview

Post-build Actions

Discard old builds  GitHub project  This project is parameterized  Throttle builds  Disable this project  Execute concurrent builds if necessary

Advanced...

Source Code Management

None  Git...

Save Apply Repository URL: https://github.com/Zippyops/mavenproject.git

# MAKING AUTOMATION WORK

To select source code management, I have selected source code management as git.

Jenkins > Mavenproject >

**General**   **Source Code Management**   Build Triggers   Build Environment   Pre Steps   Build   Post Steps   Build Settings

**Post-build Actions:**

**Source Code Management**

- None
- Git

**Repositories**

Repository URL: <https://github.com/Zippyops/mavenproject.git>

Credentials: - none - [Add](#)

[Advanced...](#)   [Add Repository](#)

**Branches to build**

Branch Specifier (blank for 'any'): \*/master

[Add Branch](#)

**Repository browser**: (Auto)

**Additional Behaviours**: [Add](#)

Subversion

**Build Triggers**

[Save](#)   [Apply](#)

Active Go to

I have added my github repository URL and repository branch as master.

**General**   **Source Code Management**   **Build Triggers**   **Build Environment**   Pre Steps   Build   Post Steps   Build Settings

**Post-build Actions:**

- Delete workspace before build starts
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Use secret text(s) or file(s)
- With Ant

**Pre Steps**

[Add pre-build step](#)

**Build**

Root POM: pom.xml

Goals and options: clean

[Advanced...](#)

**Post Steps**

Run only if build succeeds    Run only if build succeeds or is unstable    Run regardless of build result

Should the post-build steps run only for successful builds, etc.

[Add post-build step](#)

[Save](#)   [Apply](#)

Active Go to

MANAGING AUTOMATION WORK

In build Jenkins understands Maven pom files and project structures and added my goal as clean and save the project.

The screenshot shows the Jenkins interface for a Maven project named "Mavenproject". On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Build (disabled), Configure, and Modules. Below this is a "Build History" section with a search bar and a single entry for build #2, which was run on Oct 16, 2017 at 3:09 PM. At the bottom of the sidebar are RSS links for all and failures. To the right, the main content area has a title "Maven project Mavenproject" and a subtitle "This is my maven project". It features two links: "Workspace" and "Recent Changes". Below these are "Permalinks" for the workspace and recent changes.

I have clicked build button and build is in process.

The screenshot shows the Jenkins "Console Output" page for build #2. The left sidebar includes links for Back to Project, Status, Changes, Console Output (which is selected), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, Redeploy Artifacts, and See Fingerprints. The main content area displays the build log, starting with "Started by user zippyops" and "Building in workspace C:\Program Files (x86)\Jenkins\workspace\Mavenproject". The log continues through various Maven commands like git.exe rev-parse, git.exe config, git.exe fetch, and git.exe checkout, followed by a commit message and the start of the build. It also shows the discovery of a new module and the execution of Maven commands like "mvn clean". The log concludes with a warning about illegal reflective access operations and a final "[INFO] Scanning for projects..." message. A watermark for "Activate Windows" is visible in the bottom right corner.

Build is completed and maven project is executed and clean the project.

The screenshot shows the Jenkins interface for a Maven project named "Mavenproject". The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Maven project, Configure, and Modules. A search bar and a build history section (#2, Oct 16, 2017 3:09 PM) are also present. The main content area displays the project name "Maven project Mavenproject" and a message "This is my maven project". It features two links: "Workspace" and "Recent Changes". Below these are "Permalinks" to four recent builds. A large circular progress bar at the bottom indicates a success rate of 100%.

Maven project Mavenproject

This is my maven project

Workspace

Recent Changes

Permalinks

- Last build (#2), 2 min 34 sec ago
- Last stable build (#2), 2 min 34 sec ago
- Last successful build (#2), 2 min 34 sec ago
- Last completed build (#2), 2 min 34 sec ago

Build History

find

#2 Oct 16, 2017 3:09 PM

RSS for all RSS for failures

Output of maven project is success.

# ZippyOPS

Making Automation Work