ZippyOPS

Making Automation Work

Jenkins Lab Guide

www.zippyops.com

172-172, 5th floor Old Mahabalipuram Road

(Above Axis Bank-PTC Bus Stop)

Thuraipakkam

Chennai 600097

✉**zippyops@gmail.com**

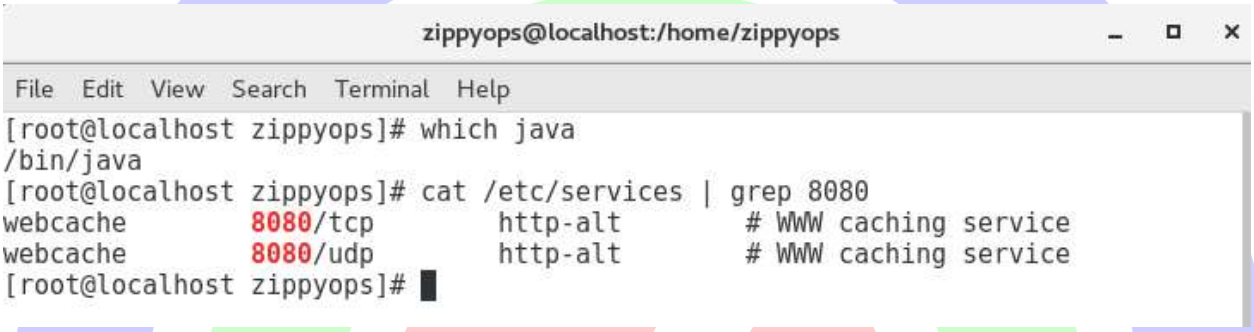
✆**+91 7010585768**

CONTENTS

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.



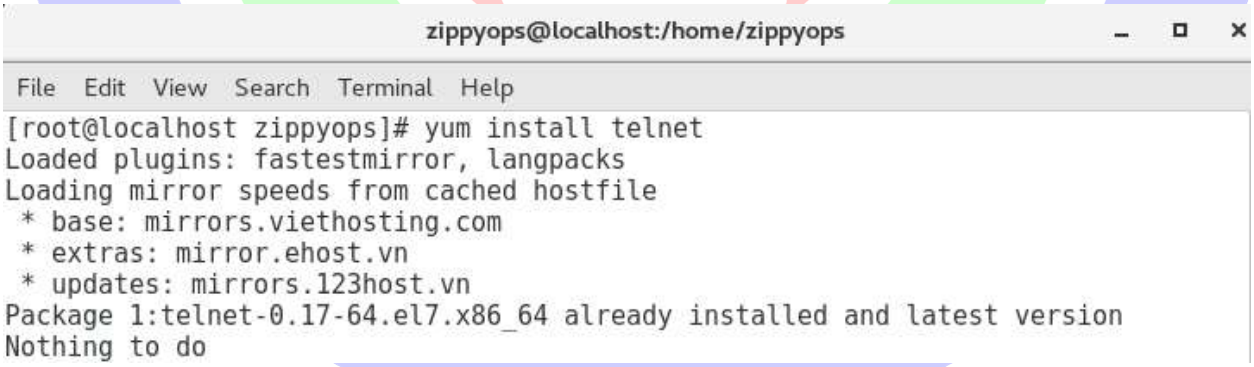
After that to check with port 8080 speak or not, if it is not speak , we need to install telnet,

For that command as ,

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

To install telnet,

```
# yum install telnet
```



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]#
```

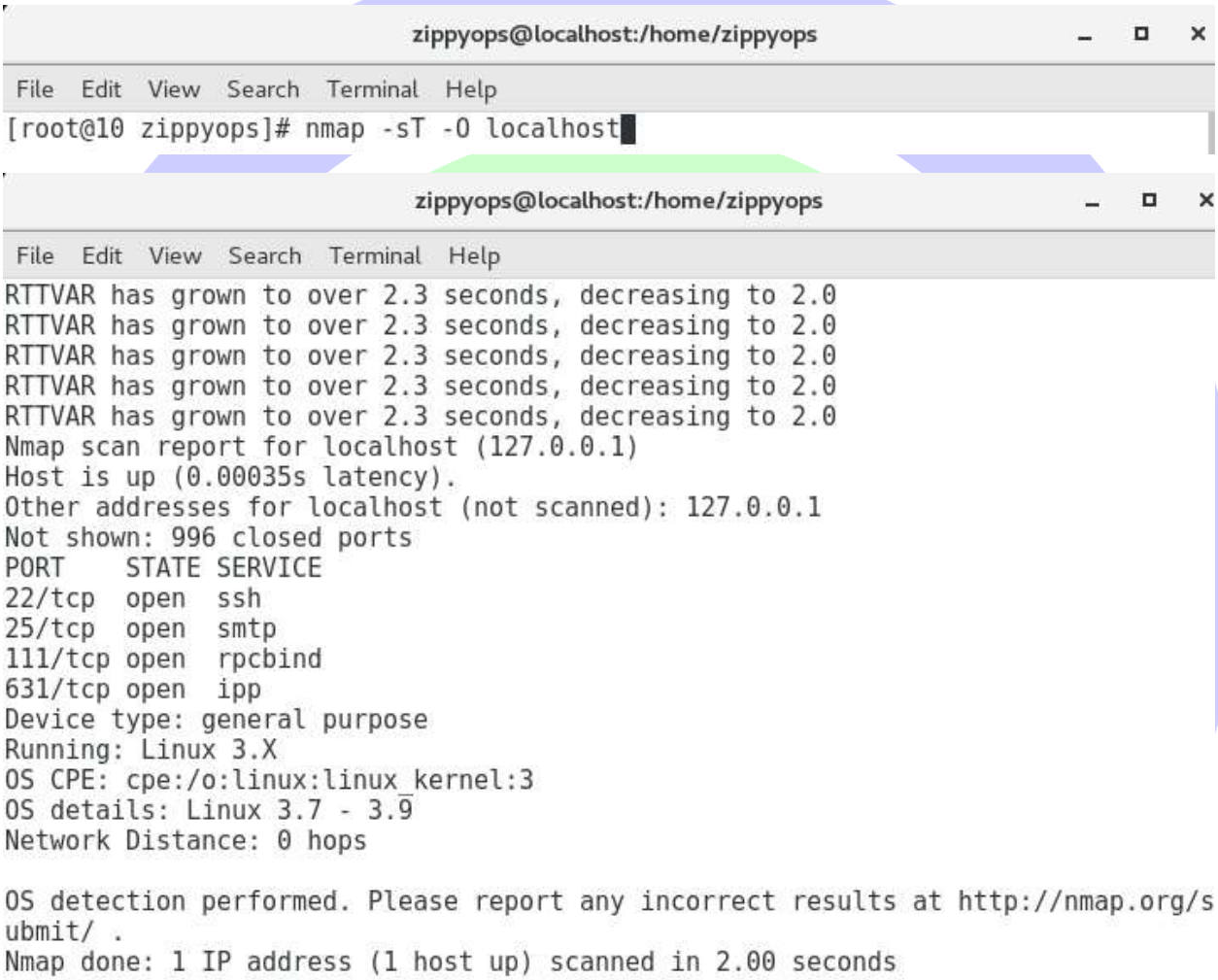
If refused the connection means nothing listening on us. So we have to know which one is running in local host port,

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"
```



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/e758a0de34e24606bca991d704f6dcbf/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
-----------------------------------------------
*  1           /opt/jdk1.7.0_71/bin/java
 + 2           /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
g/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... co
cted.
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,

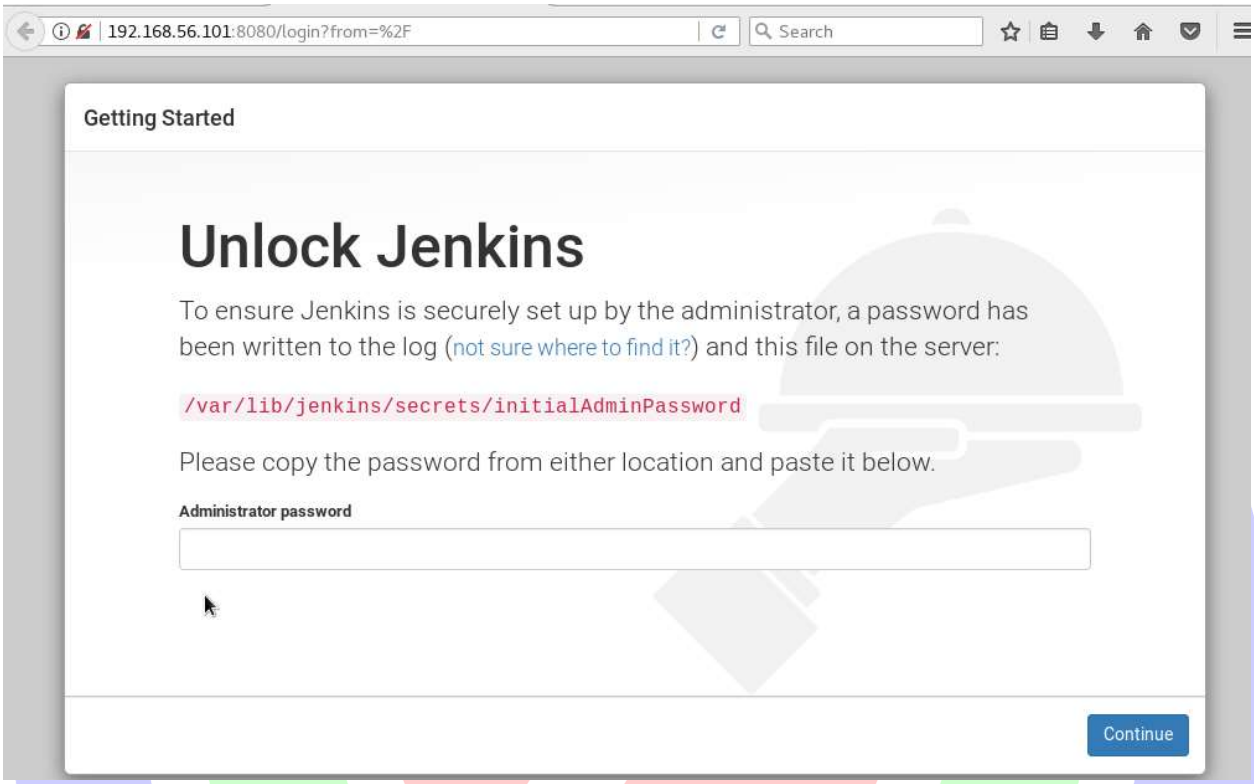
```
File  Edit  View  Search  Terminal  Help
[root@10 ~]# yum install jenkins
Loaded plugins: fastestmirror, langpacks
jenkins                                                  | 2.9 kB     00:00
jenkins/primary_db                                       | 103 kB   00:01
Loading mirror speeds from cached hostfile
 * base: mirrors.viethosting.com
 * elrepo: mirrors.thzhost.com
 * extras: mirror.ehost.vn
 * updates: mirrors.123host.vn
Resolving Dependencies
--> Running transaction check
---> Package jenkins.noarch 0:2.90-1.1 will be installed
```

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
```

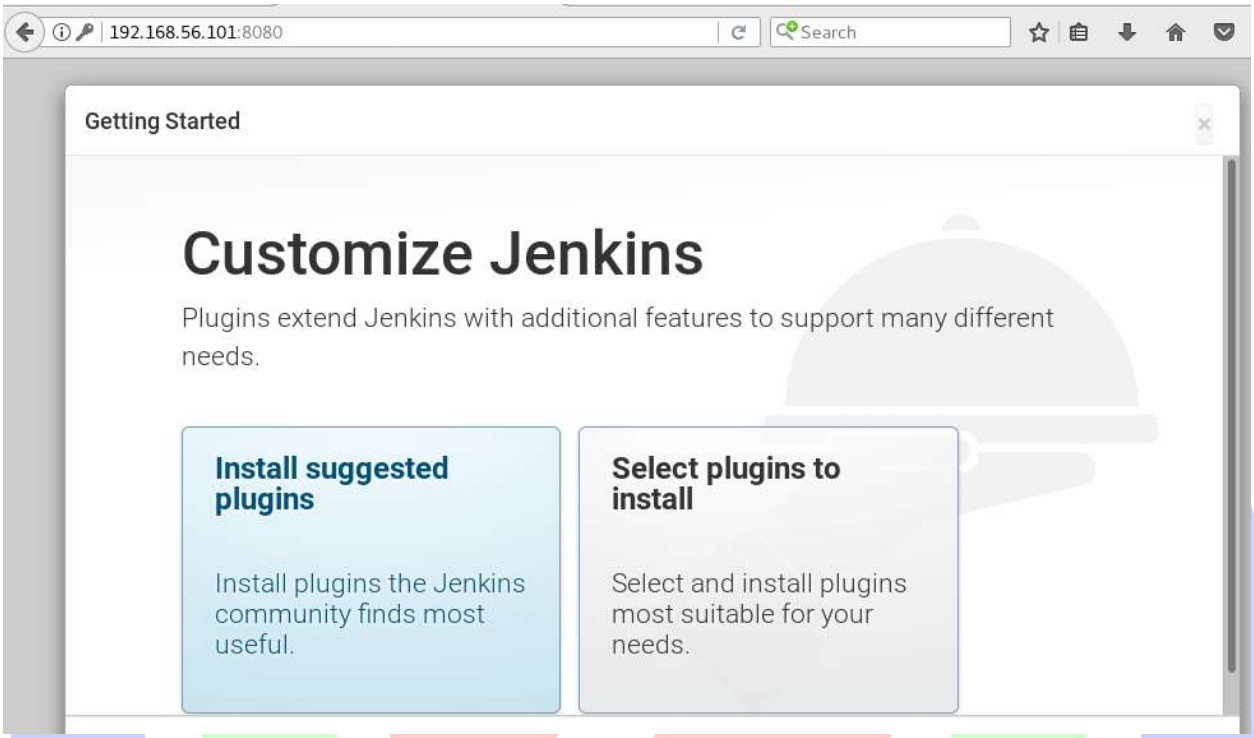
Once you see the active (running) prompt it is time to start using the Web UI. Open your web browser and visit http://ip_address:8080. Jenkins' dashboard is accessible via the public IP on port 8080.



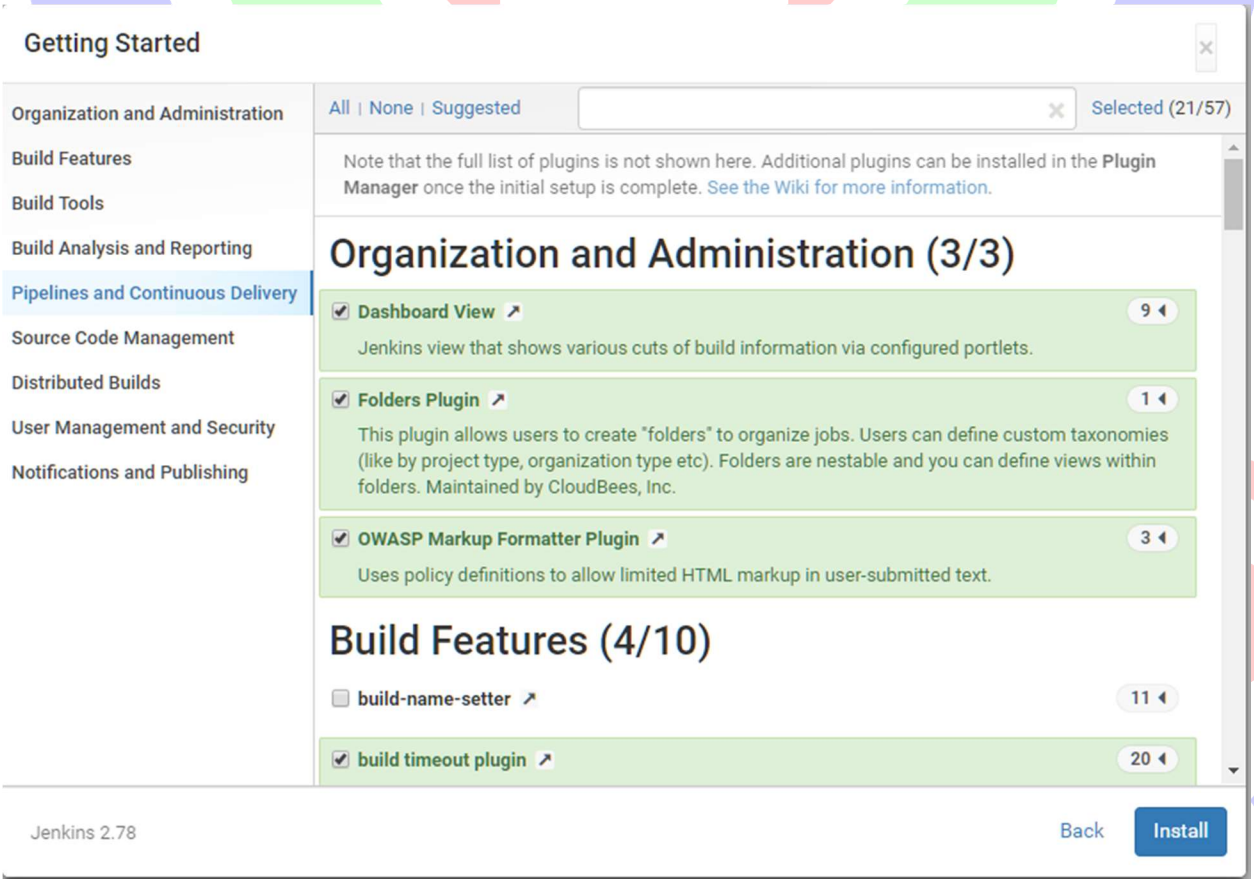
For the first time you login, you would be asked to unlock it, the password so let's get it from the path that is shown in the screen.

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

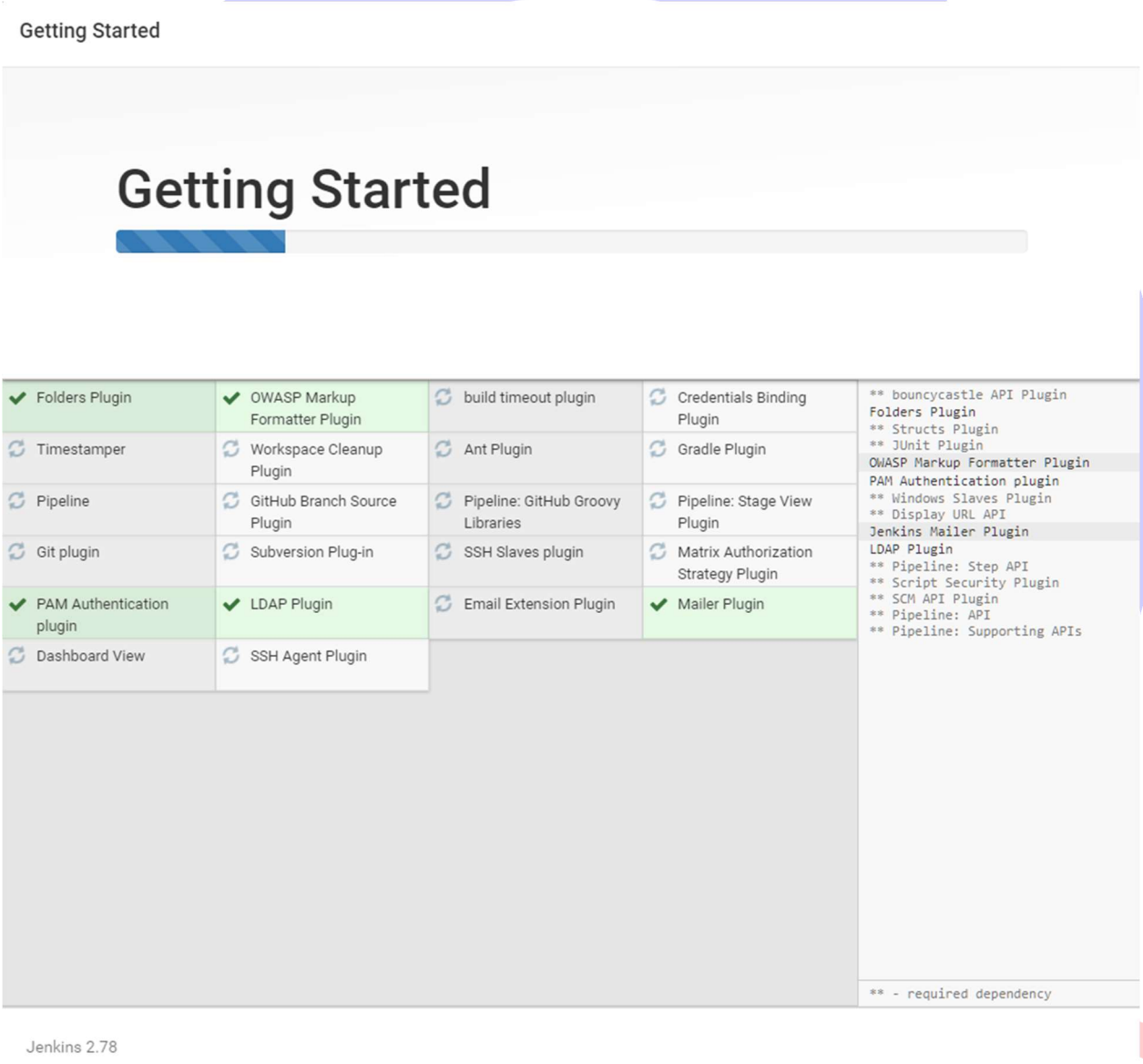
You can copy the output and use it to proceed with the Initial Set Up.



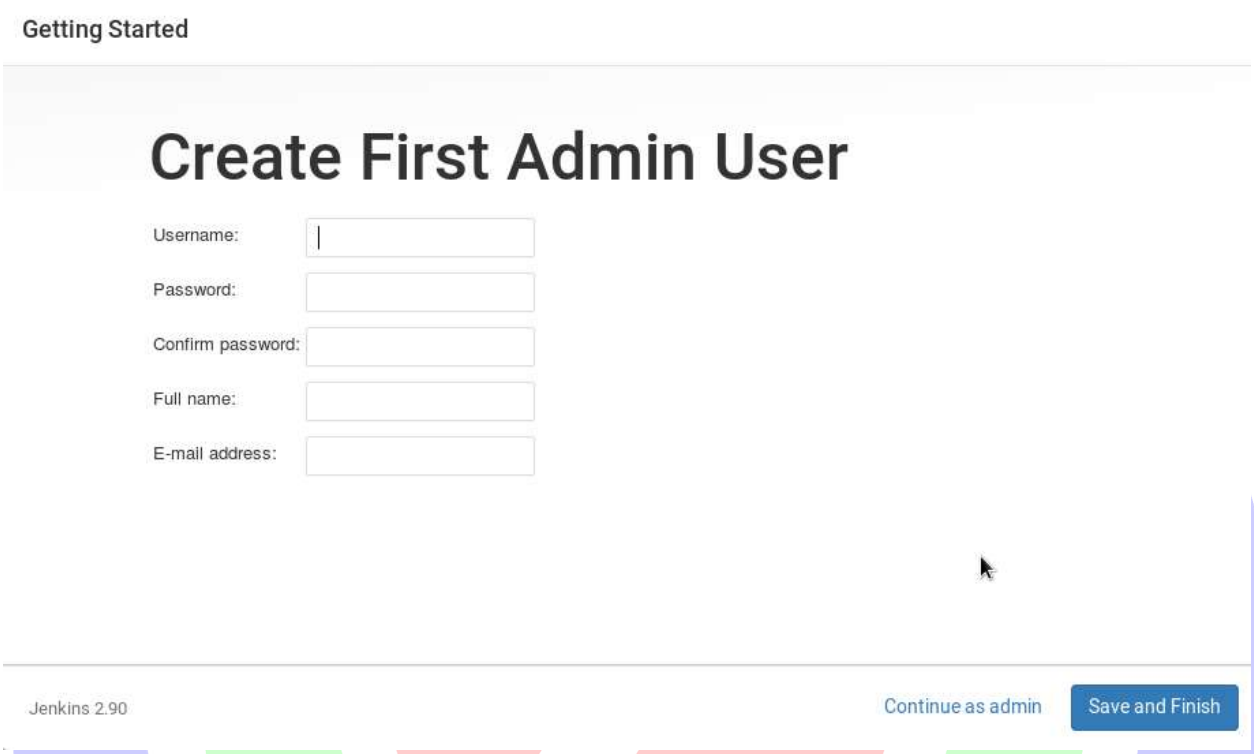
Here you may want to want to click on 'Select plugins' before installing them because Dashboard is not a part of 'Suggested Plugins', so click on the right button to select plugins.



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.

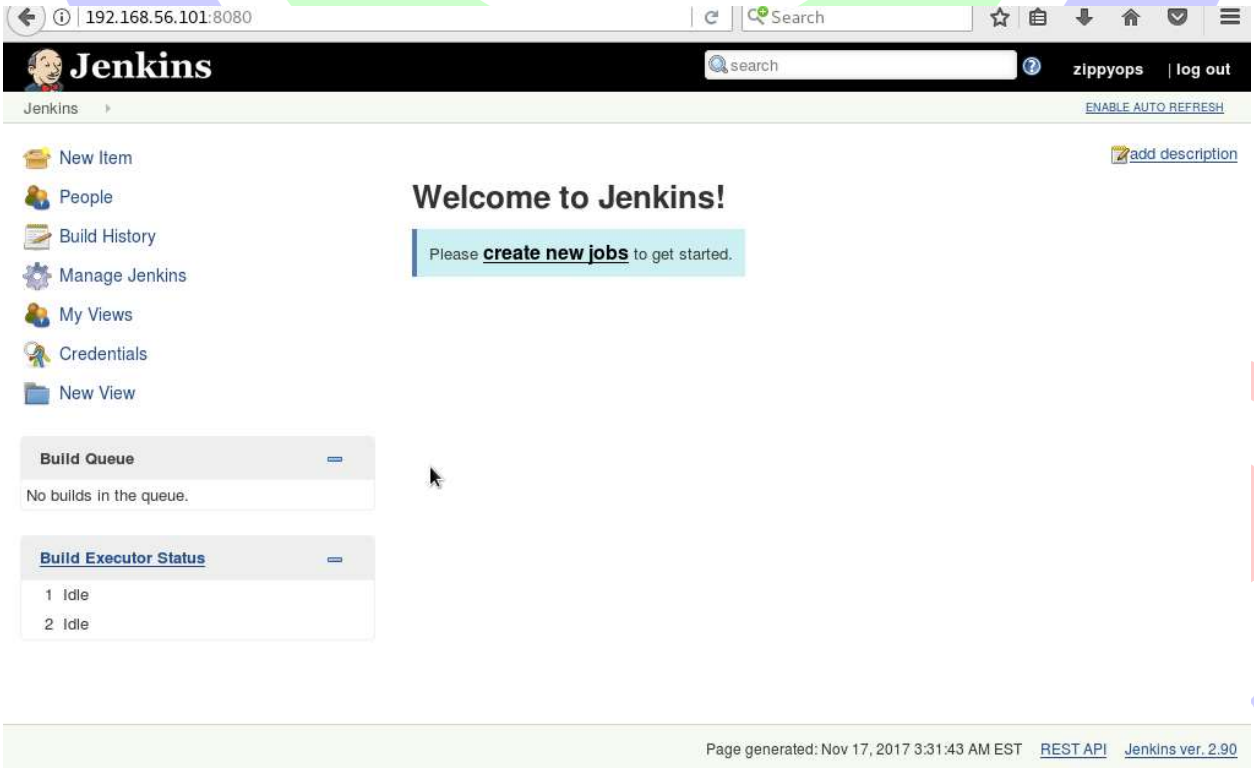


After this is done, you will be asked to set up your admin user and credentials. With that done, you are pretty much ready to import your existing software project or start a new one with Jenkins.

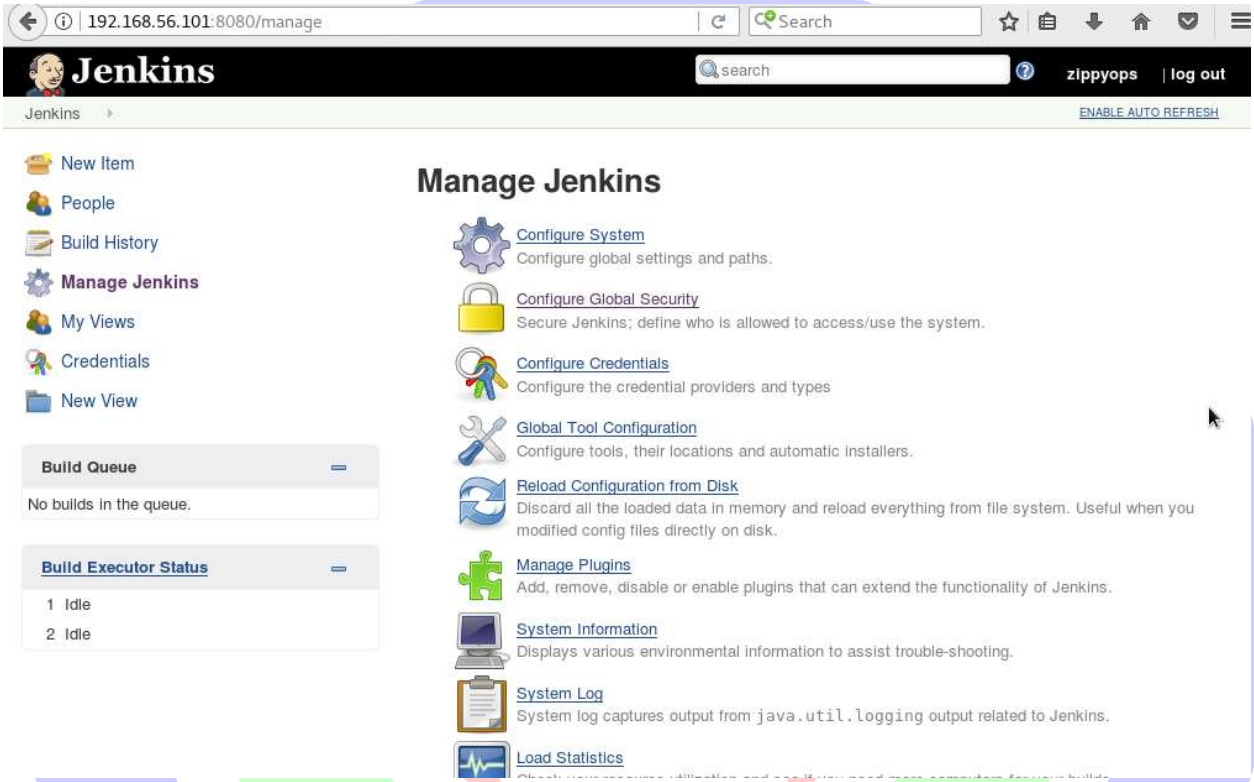


To give user name and password and email address to save and finish installation.

START USING JENKINS:



MANAGE JENKINS:



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:

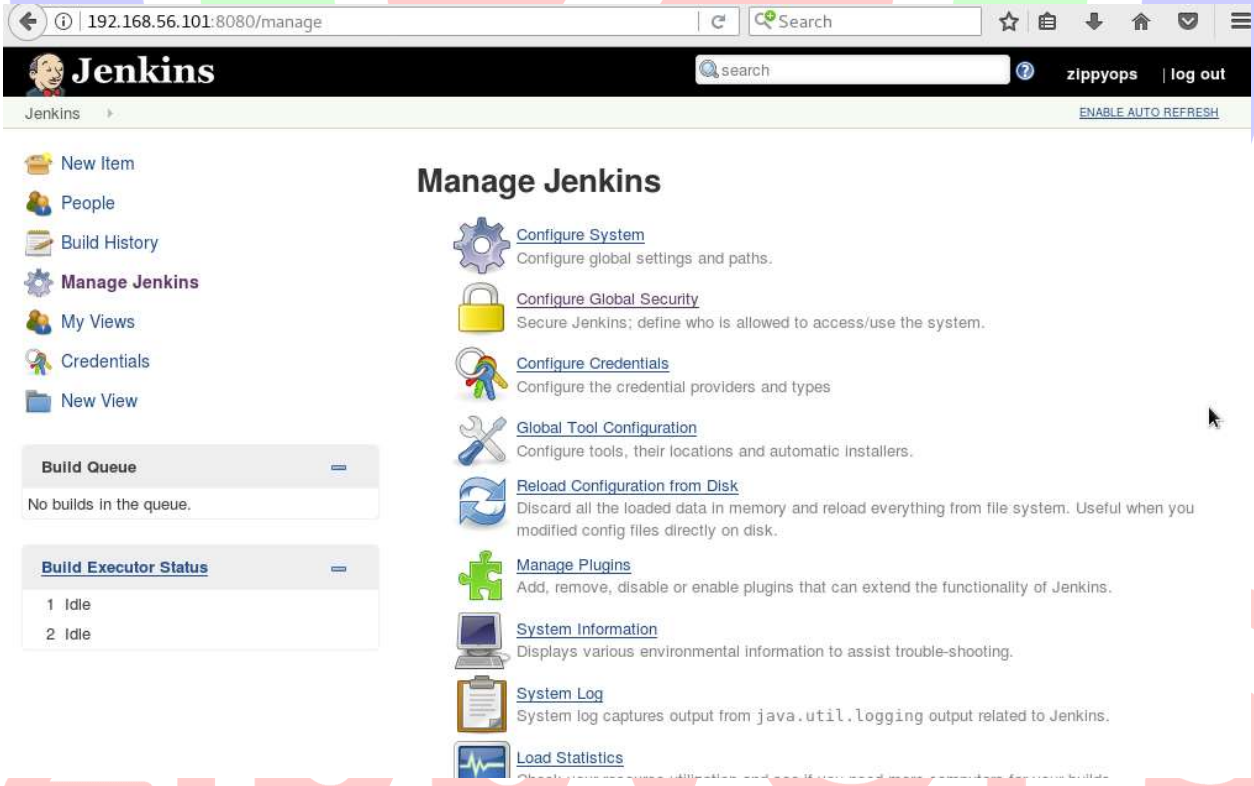
Jenkins handles parallel and distributed builds well. In this screen, you can configure how many builds you want. Jenkins runs simultaneously, and, if you are using distributed builds, set up build nodes. A build node is another machine that Jenkins can use to execute its builds.

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.

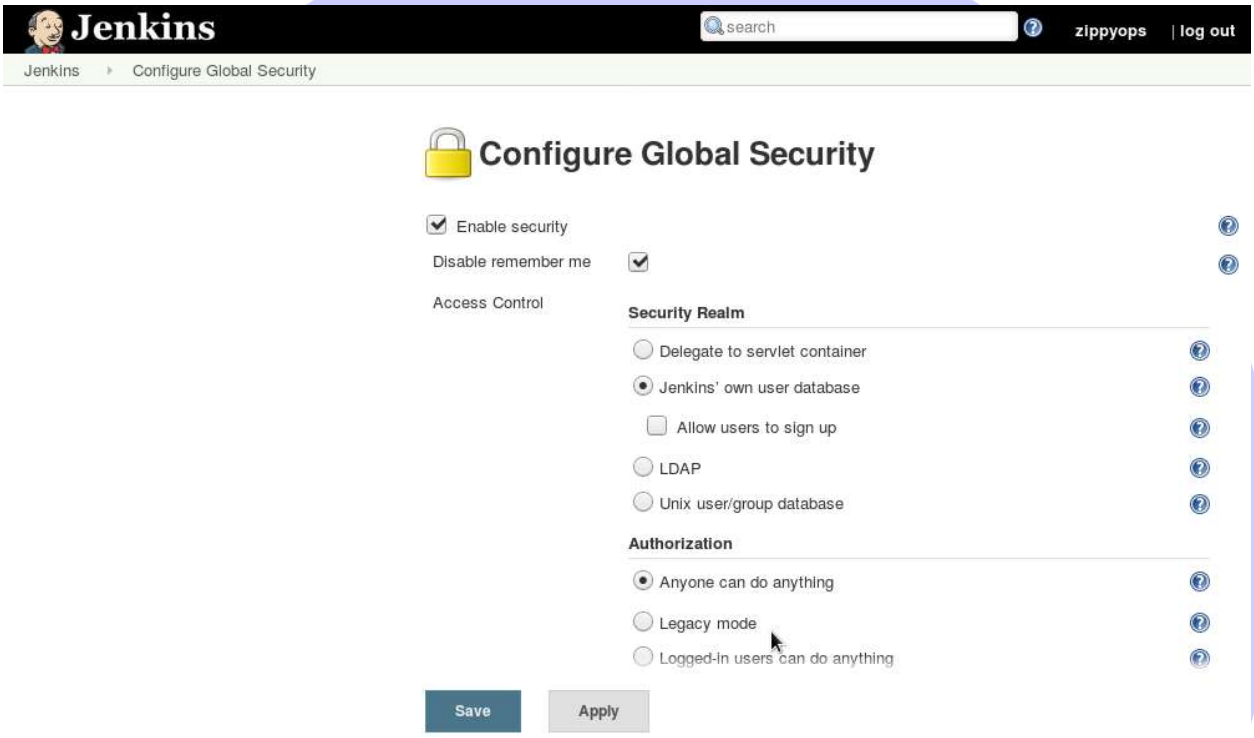
CONFIGURE SECURITY IN JENKINS:

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

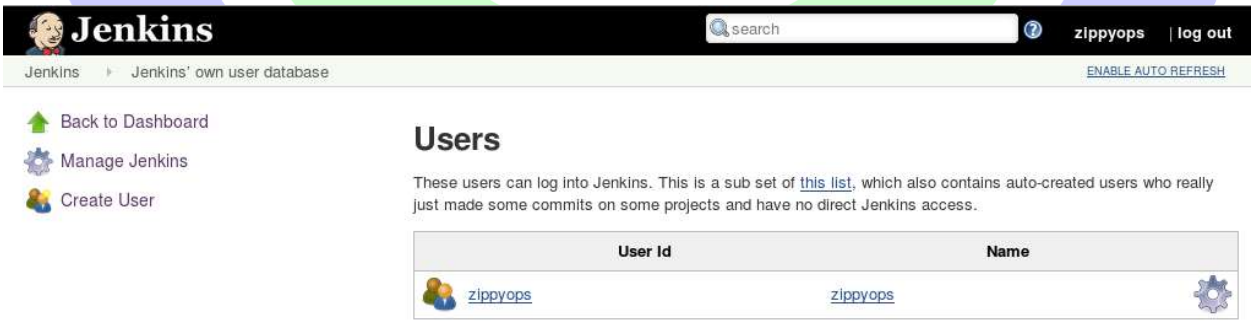


Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain it's own database of users, so in the Security Realm, choose the option of 'Jenkins' own user database'.

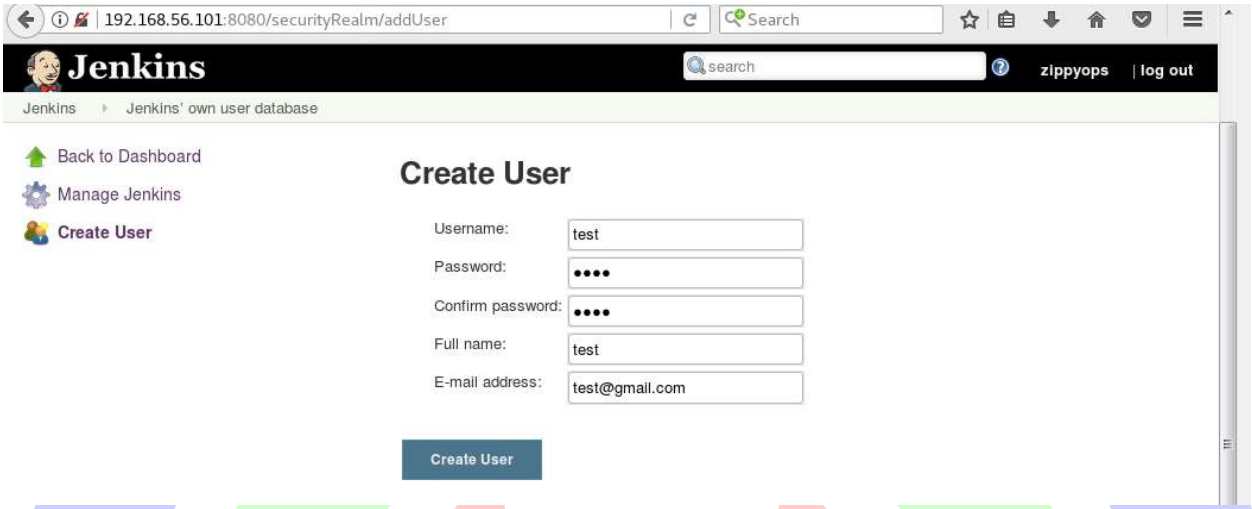
By default you would want a central administrator to define users in the system, hence ensure the 'Allow users to sign up' option is unselected. You can leave the rest as it is for now and click the Save button.



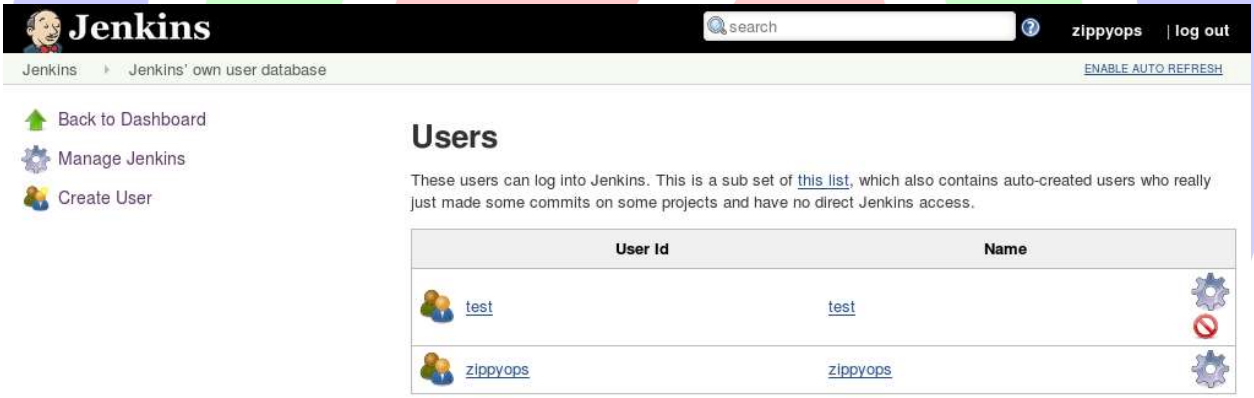
First you can click manage Jenkins, then entered to manager user.



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



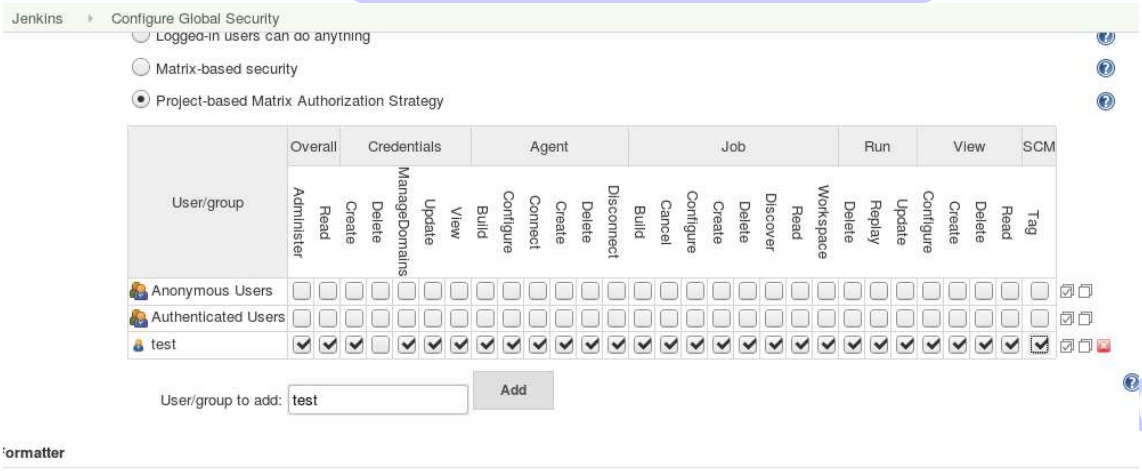
Start creating other users for the system. As an example, we are just creating another user called 'test'.



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'

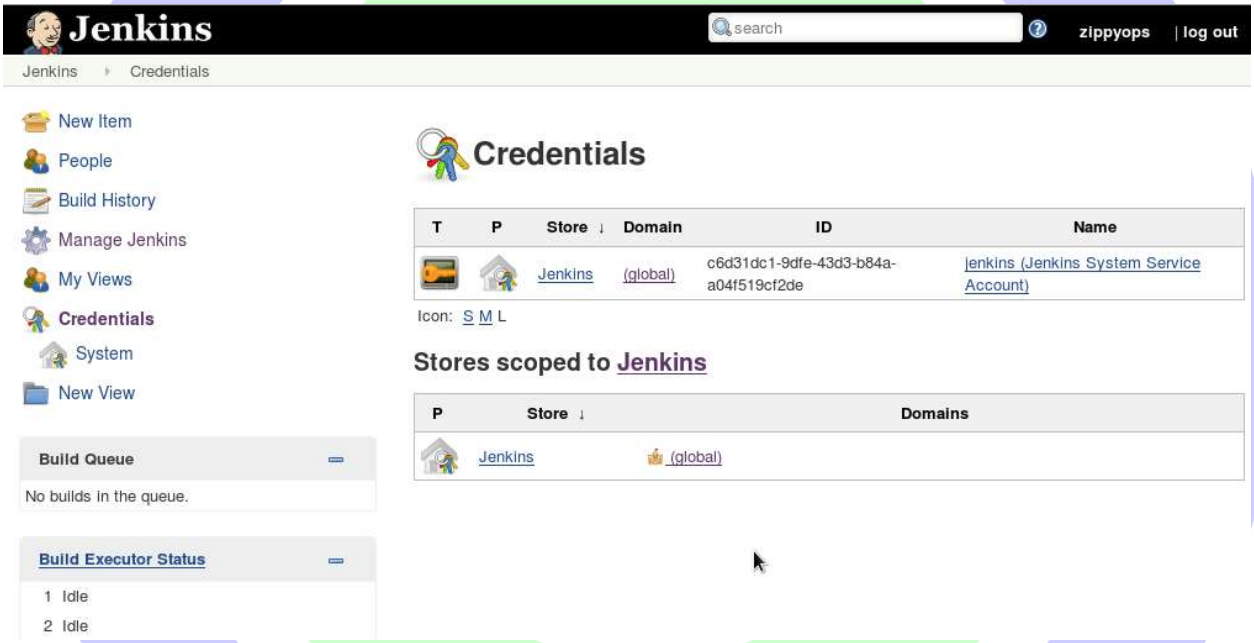


If you don't see the user in the user group list, enter the user name and add it to the list. Then give the appropriate permissions to the user.

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

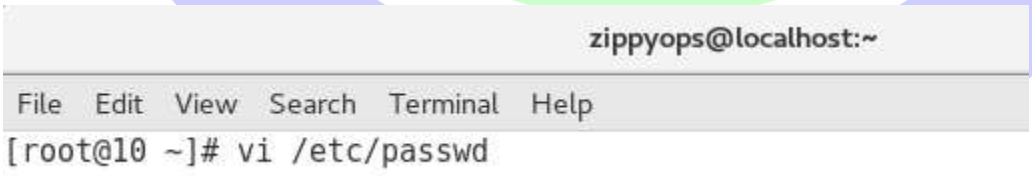
Your Jenkins security is now setup.

MANAGE CREDENTIALS:



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

So we have edit passwd file to bash ,



```
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 tcsd 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
-- INSERT --
```

```
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 tcsd 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
```


To change false to bash ,

After that we can edit /etc/sudoers file ,

```
File  Edit  View  Search  Terminal  Help
[root@10 ~]# vi /etc/sudoers
```

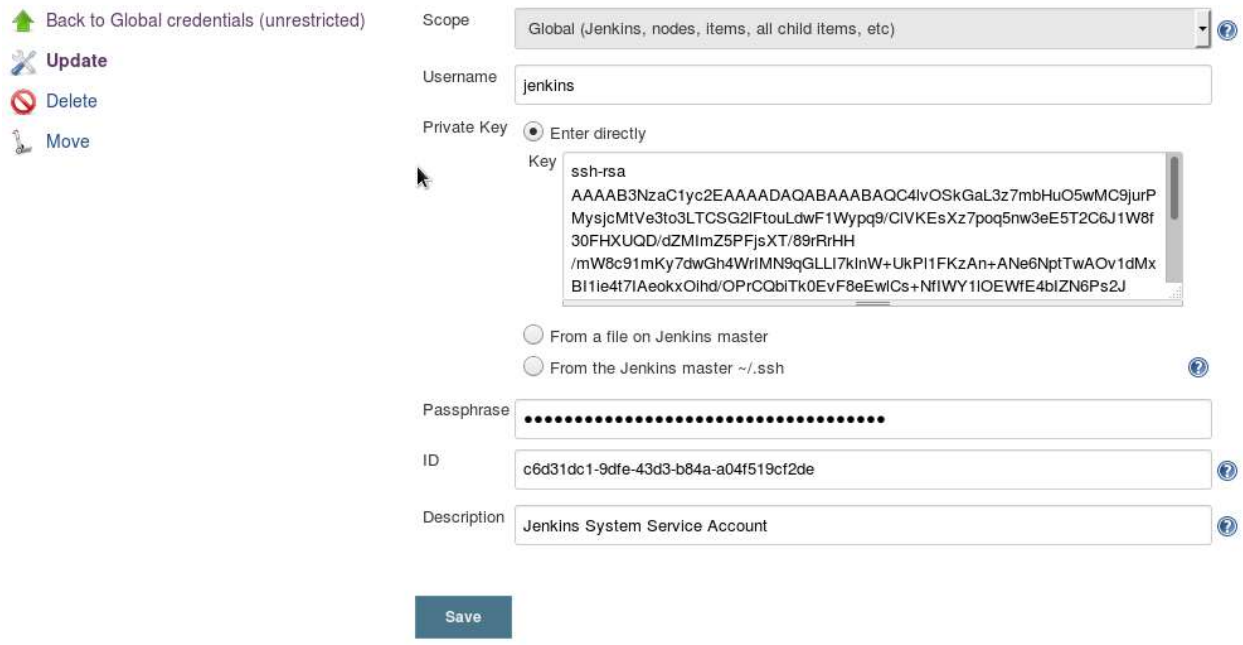
```
File  Edit  View  Search  Terminal  Help

#
# 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@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:KHyOiXWpvCG2bhM/USfizSJ0JJxLKF2BITGEXm3B48g root@10.0.2.15
The key's randomart image is:
+---[RSA 2048]----+
|**.==o.          |
|+oB..=           |
|o.o++ .          |
| .oEo.oo.        |
| . o+==oS        |
|  o++0o          |
|  ++*o.          |
| .oooo           |
| oo...           |
+----[SHA256]-----+
```

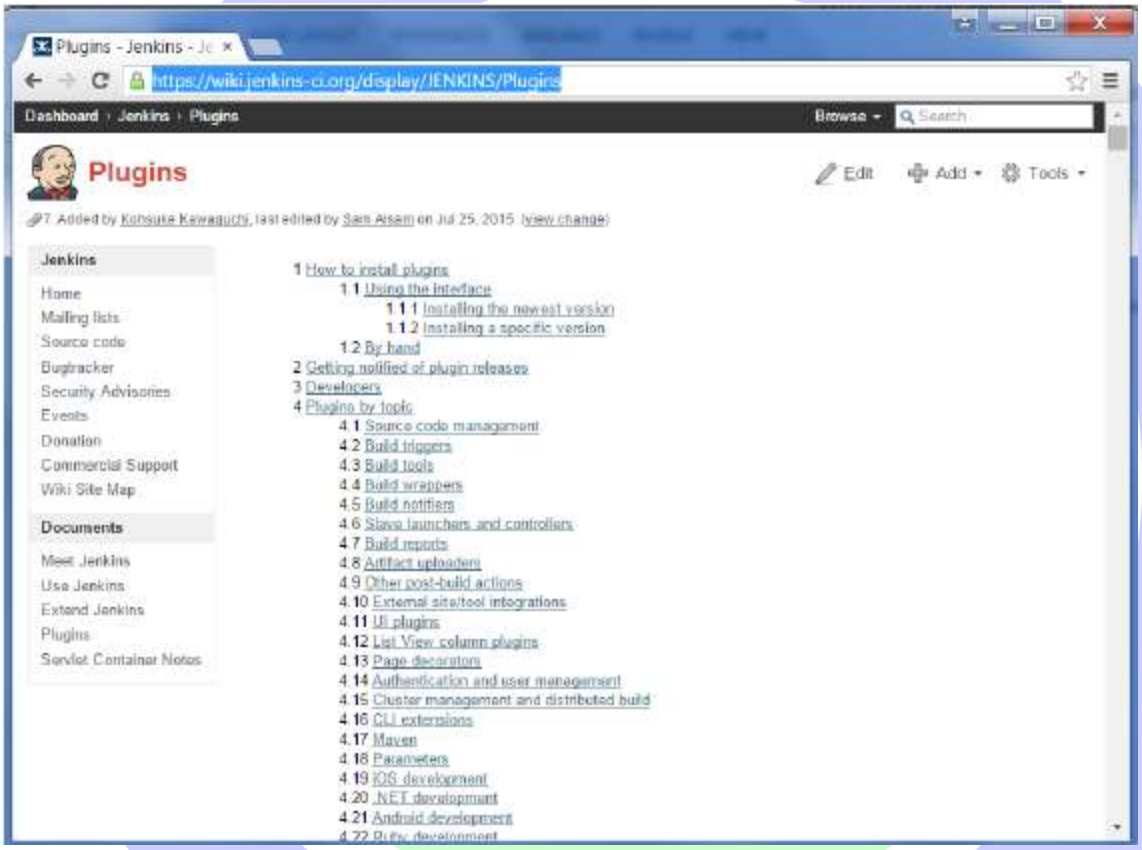
After that go to localhost:8080 ,



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

PLUGIN MANAGEMENT:

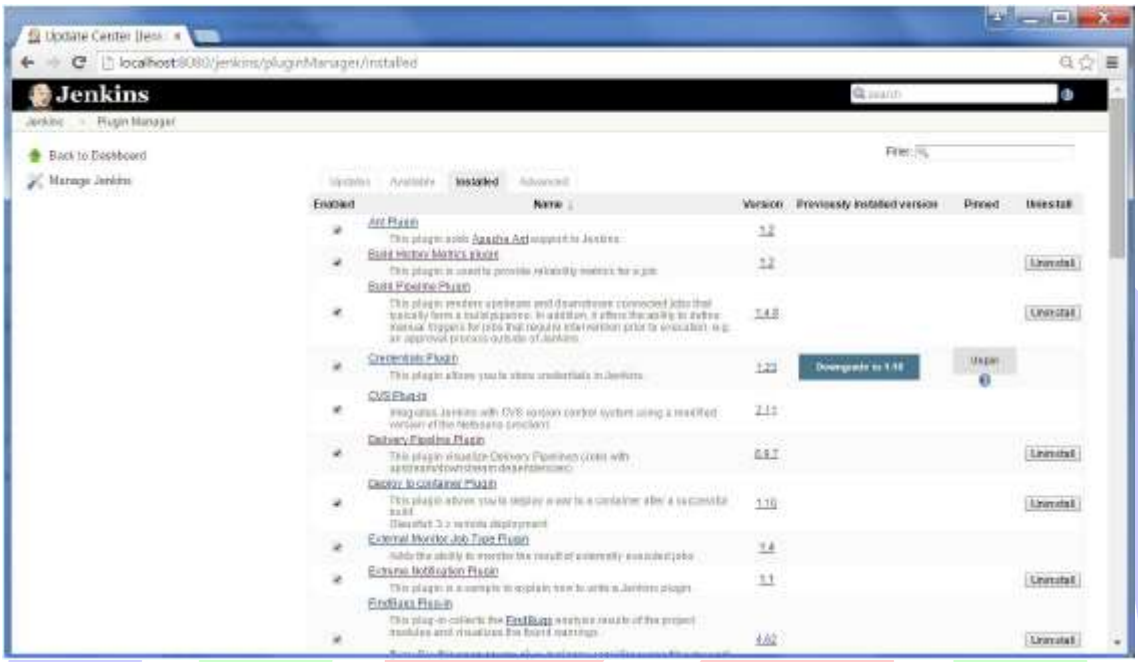
To get the list of all plugins available within Jenkins, one can visit the link –
https://wiki.jenkins-ci.org/display/JENKINS/Plugins



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

UNINSTALLING PLUGINS:

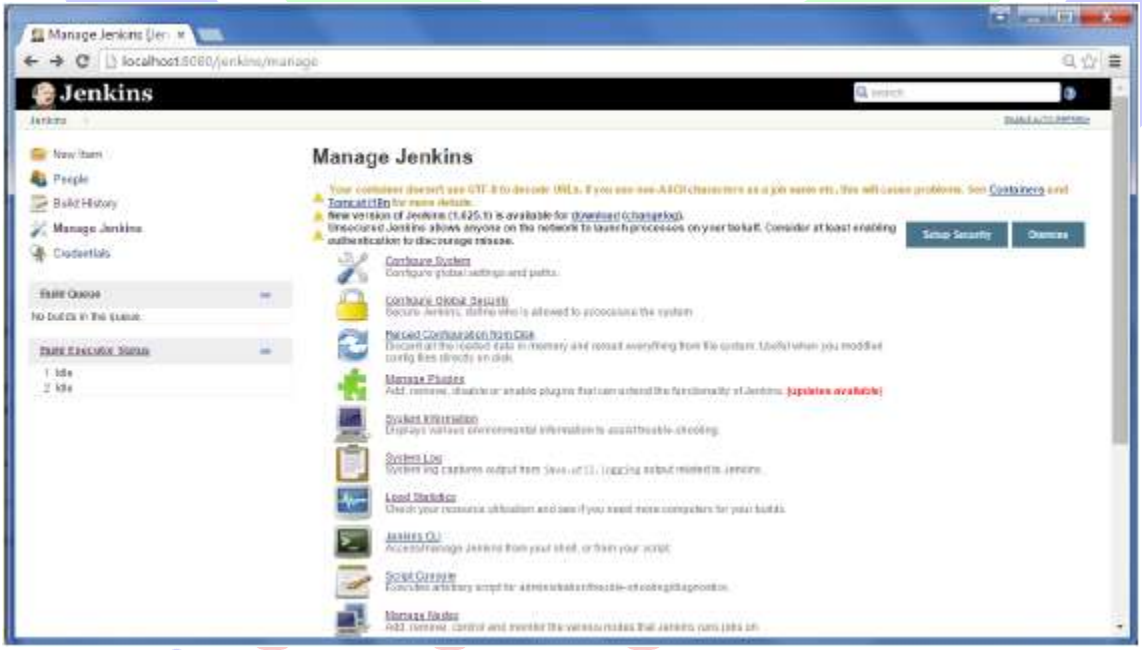
To uninstall a plugin, go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.



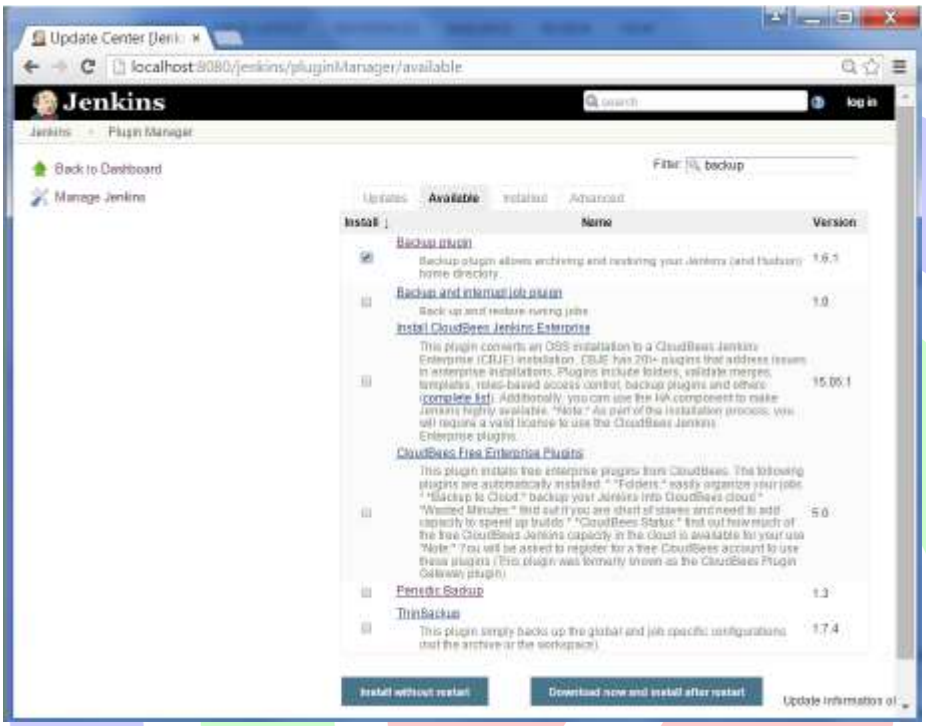
BACKUP PLUGIN:

Jenkins has a backup plugin which can 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.

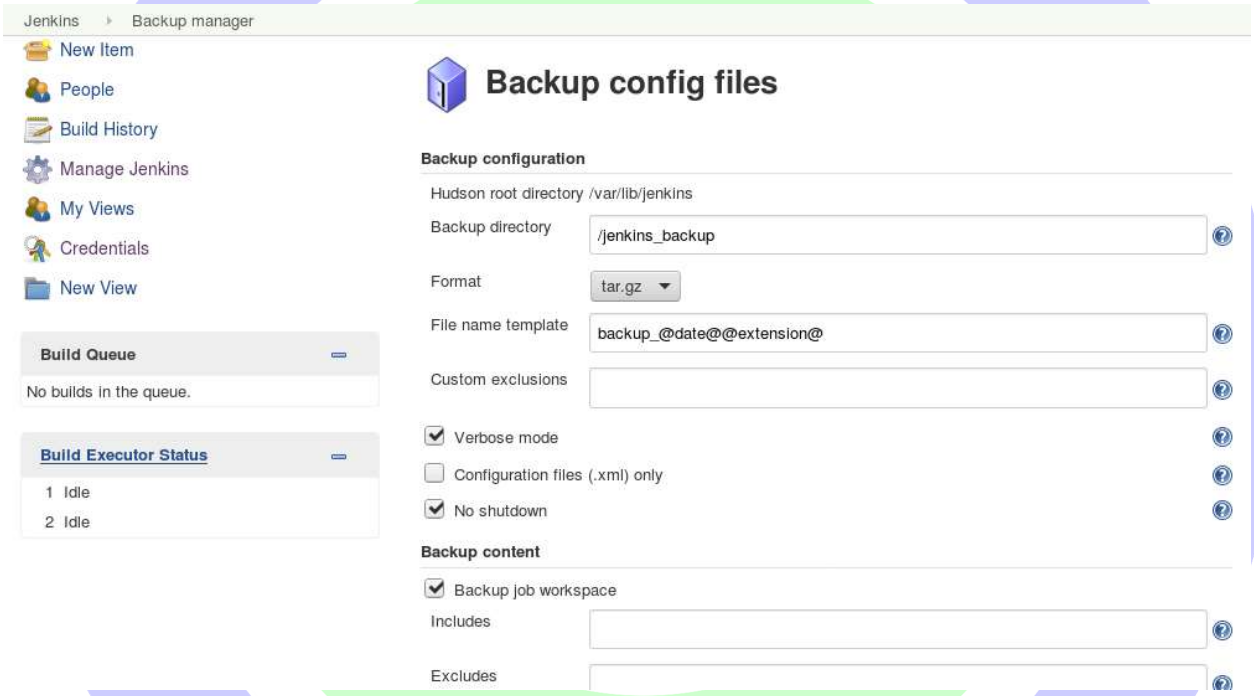
Click on Setup.

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.

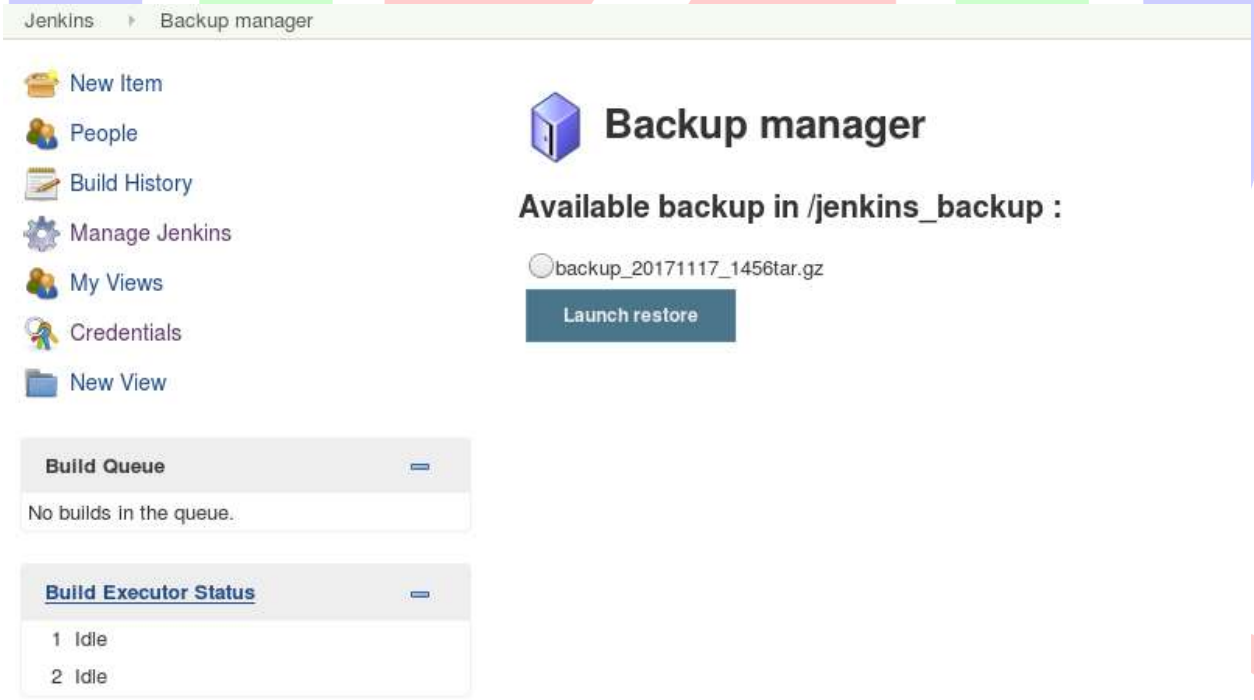


Click on the 'Backup Hudson configuration' from the Backup manager screen to initiate the backup.

The next screen will show the status of the backup

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

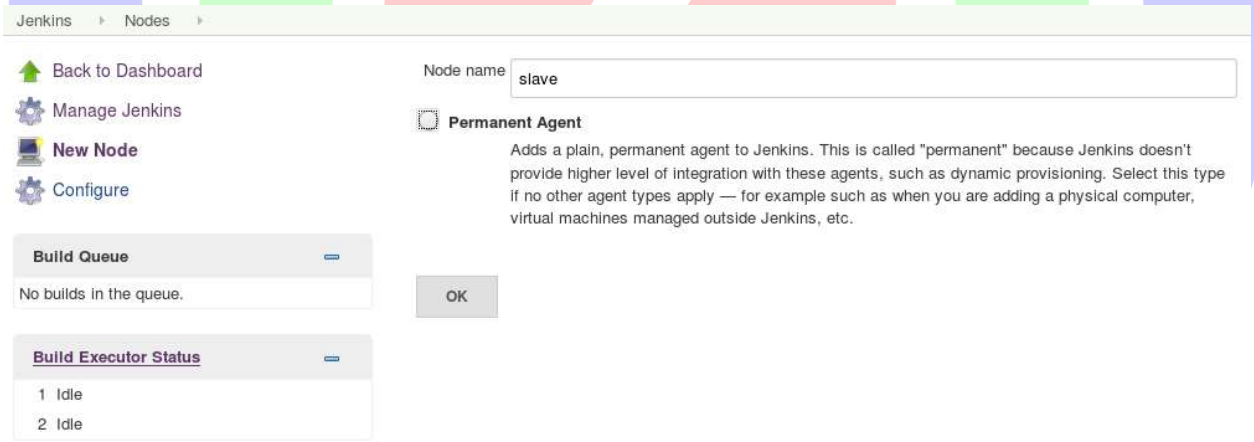
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



CONFIGURE JENKINS SLAVE AND CONNECT TO MASTER:

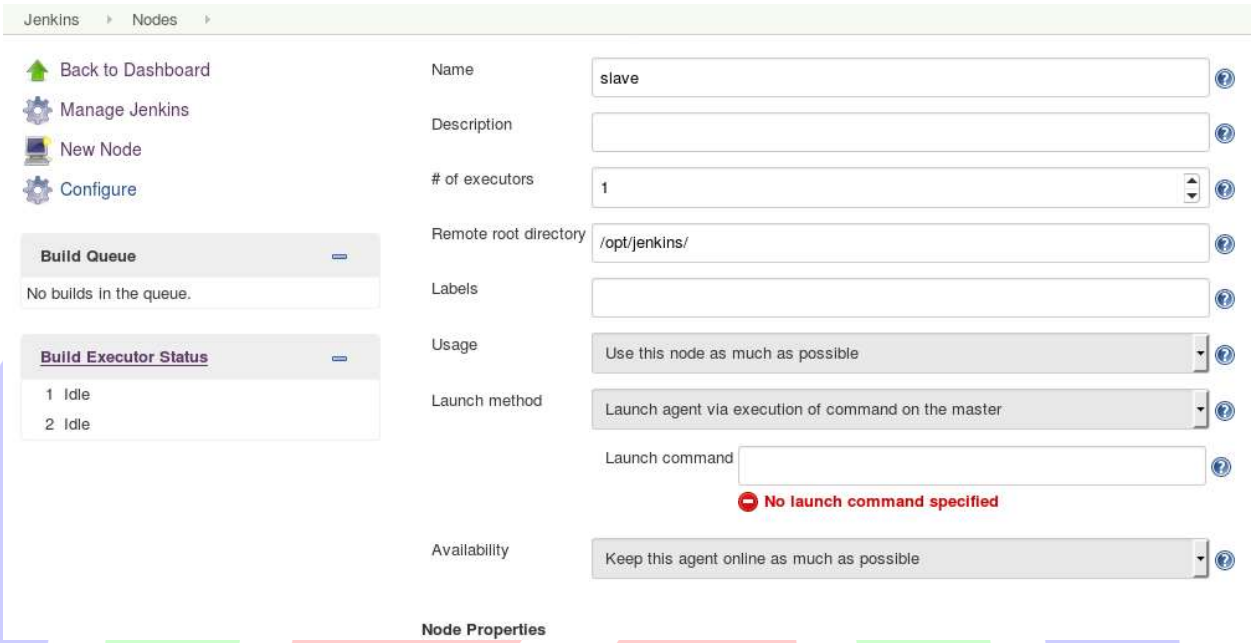
On Jenkins master go to Manage Jenkins > Manage Nodes

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



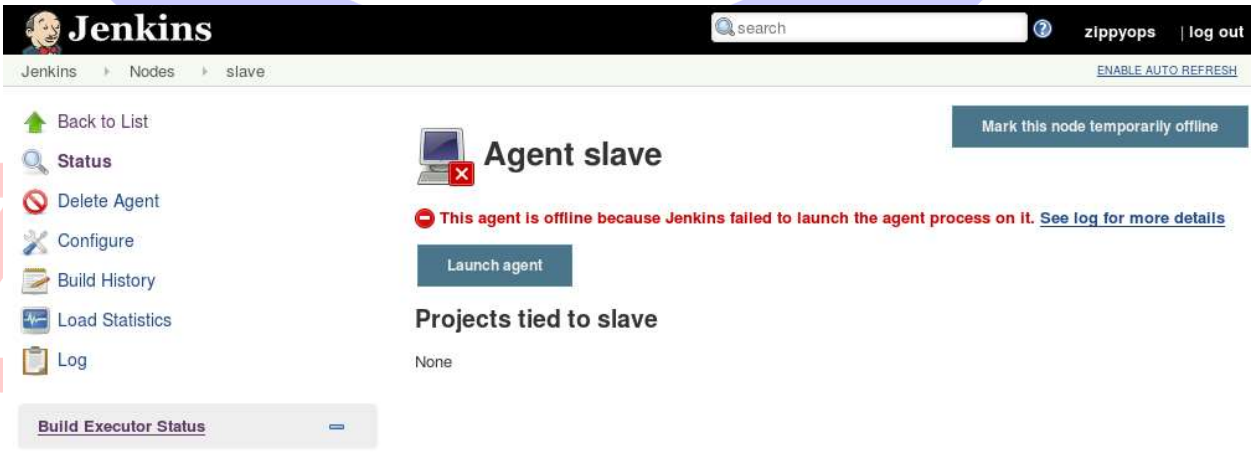
On the next page, fill in the following fields:

- 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.



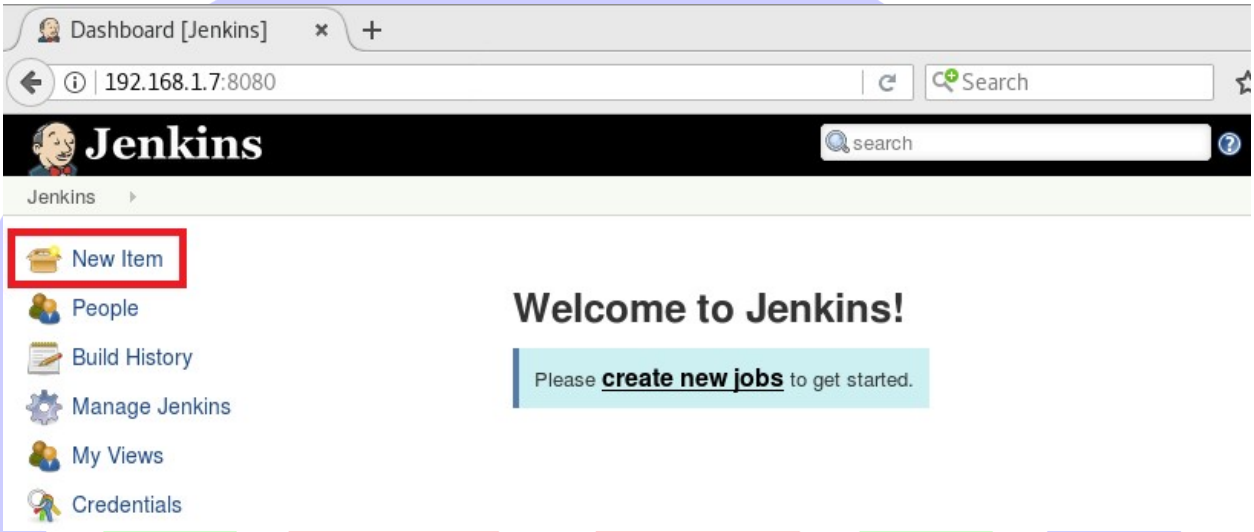
Now for connecting to slave to the master by

- 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.

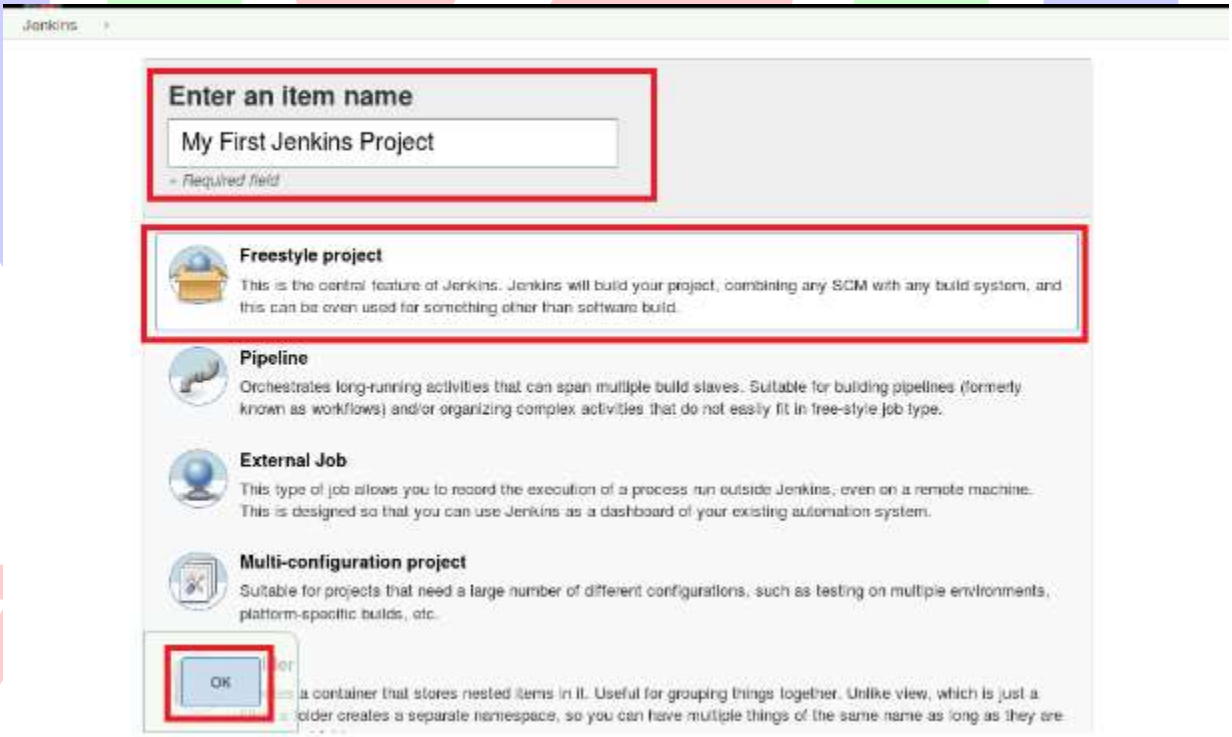


Create your first Jenkins project:

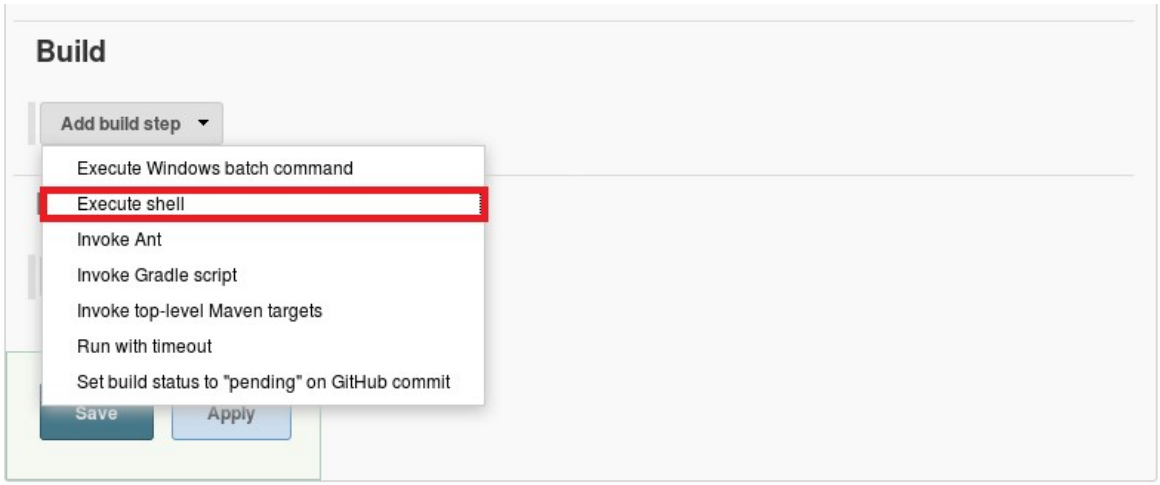
Let's create a sample build using shell command. To do that, go to Jenkins Web Portal –> New Item or create new jobs.



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

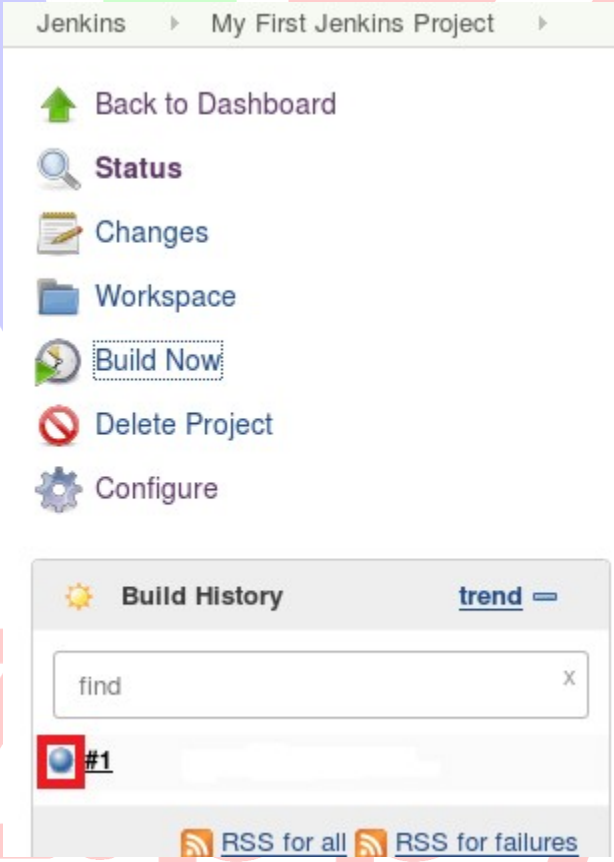


Enter a shell command to execute while building the project.

Click on “SAVE”

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

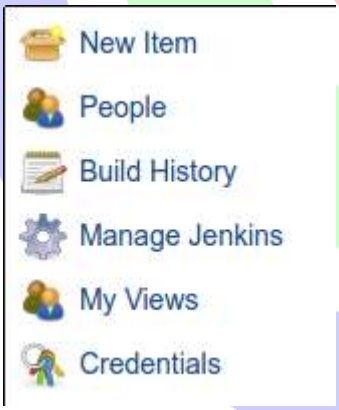
Click on “Console Icon” to see the output of your build.



Console output:

SET UP JENKINS ACCESS TO GITHUB :

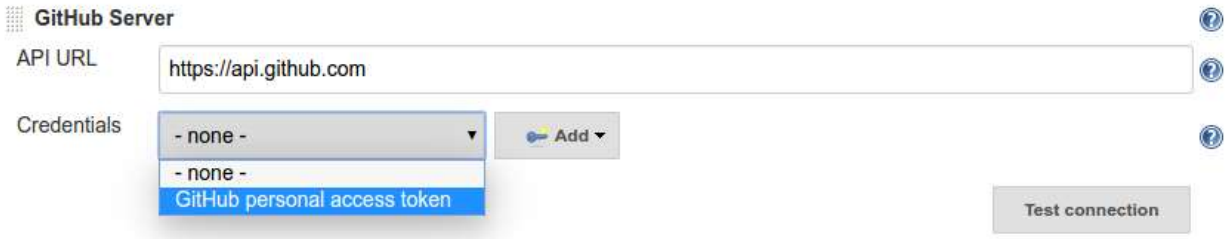
Back in the main Jenkins dashboard, click Manage Jenkins in the left hand menu:



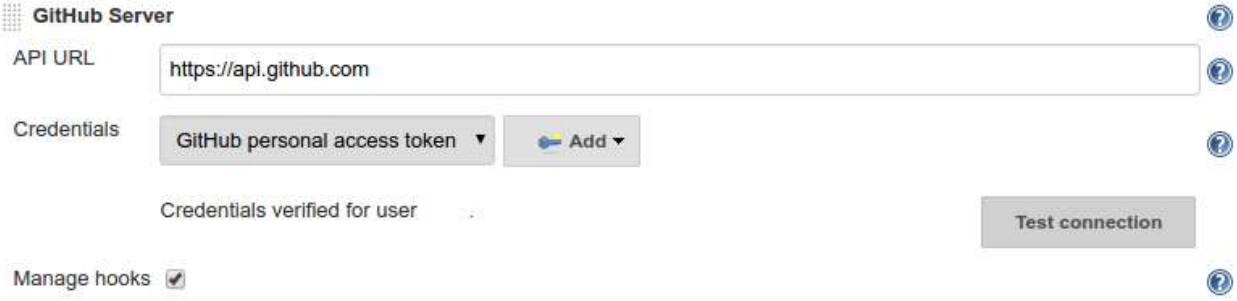
In the list of links on the following page, click Configure System:

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:

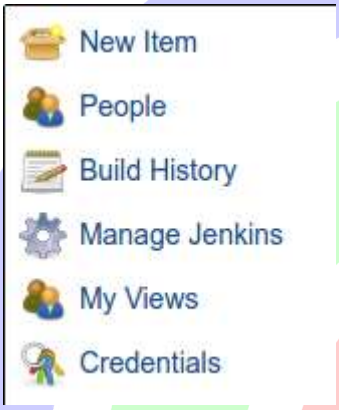


When you are finished, click the Save button to implement your changes.

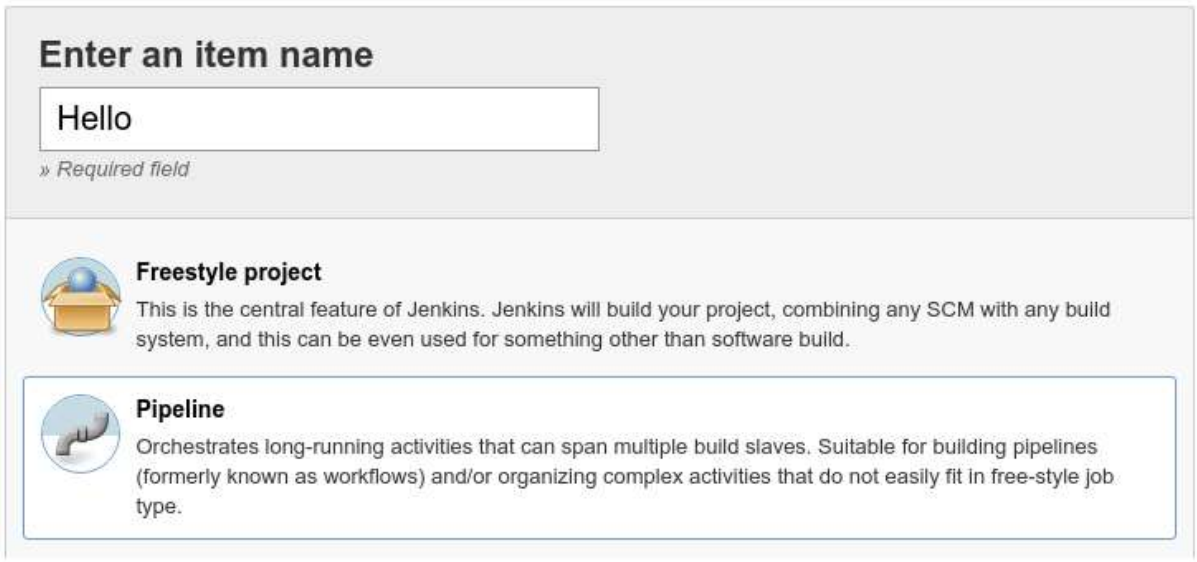
CREATE A NEW PIPELINE IN JENKINS:

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:

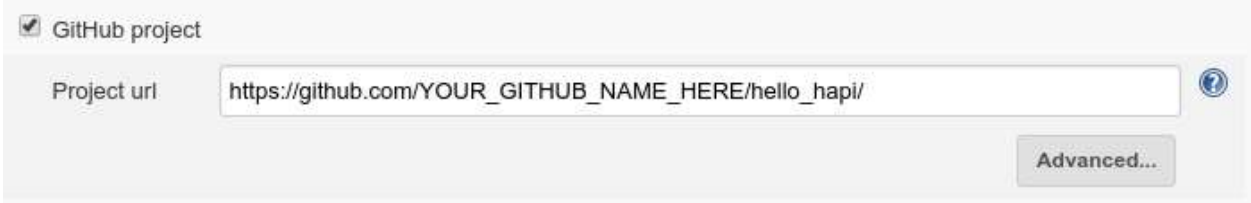


Enter a name for your new pipeline in the Enter an item name field. Afterwards, select Pipeline as the item type:

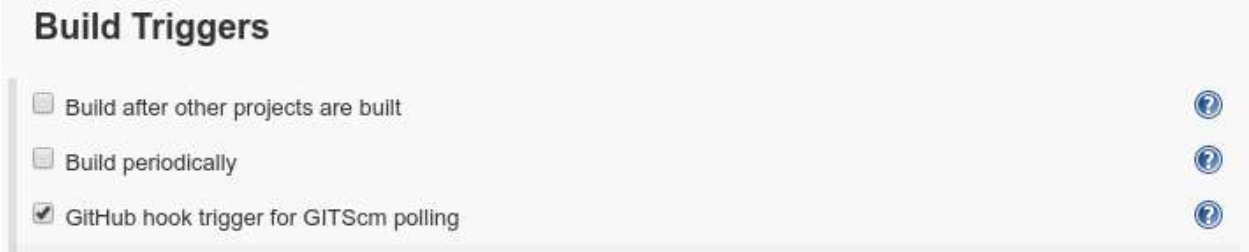


Click the OK button at the bottom to move on.

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

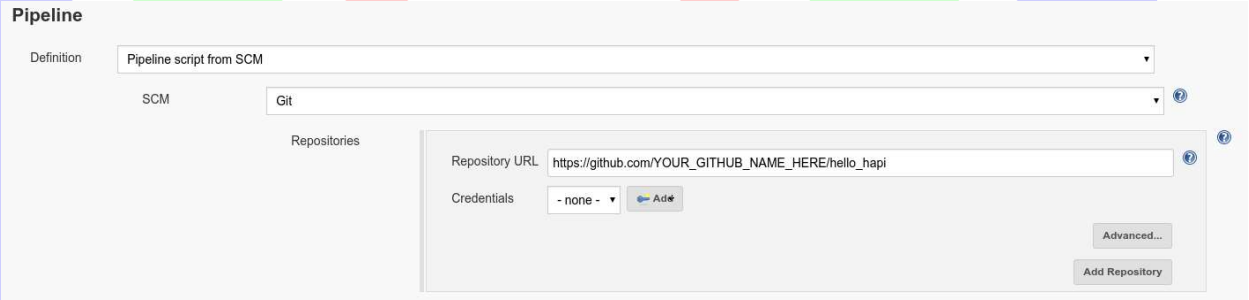


Next, in the Build Triggers section, check the GitHub hook trigger for GITScm polling box:



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:

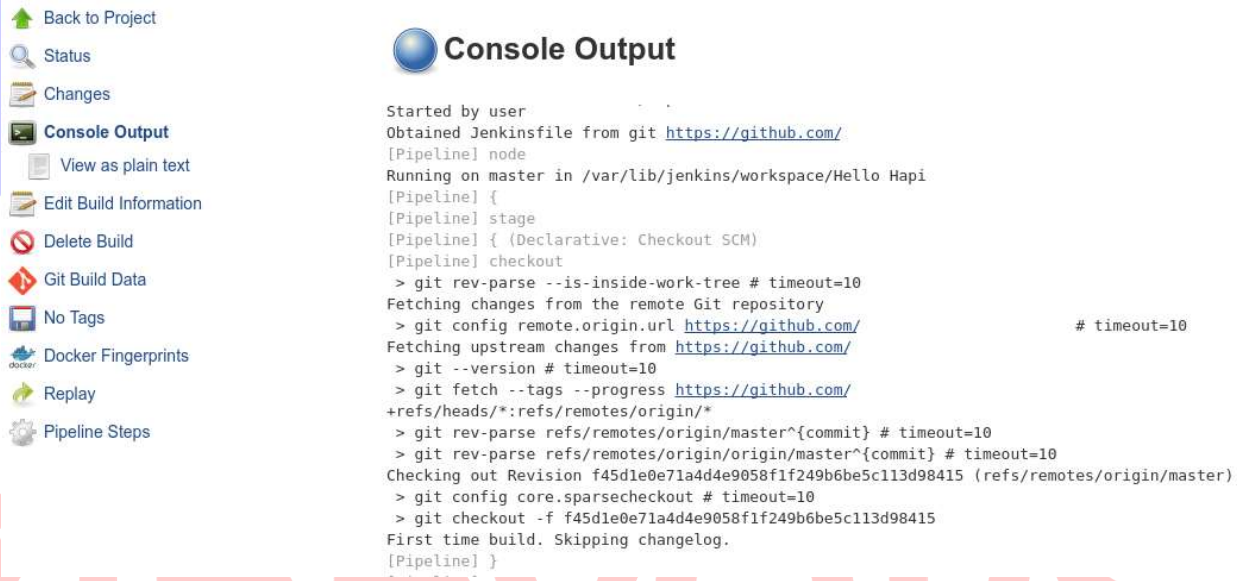


When you are finished, click the Save button at the bottom of the page.

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:

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:

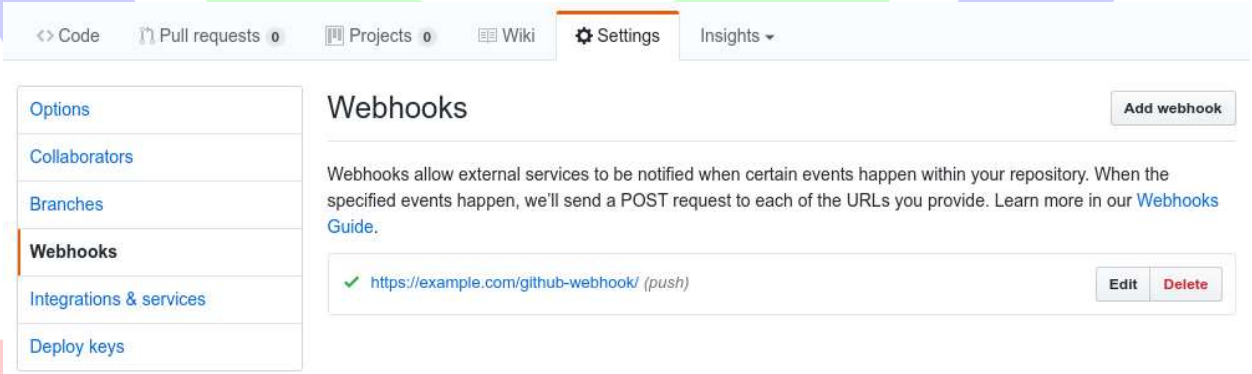


Click the Back to Project item in the left hand menu when you are finished in order to return to the main pipeline view.

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:

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.

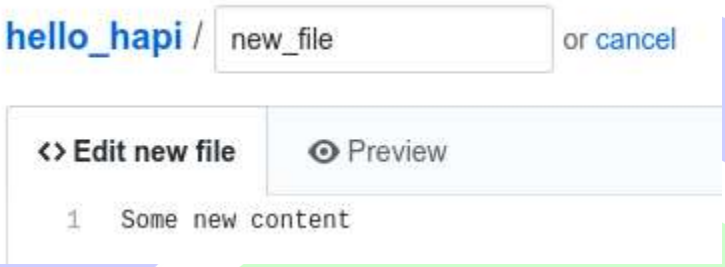
You can verify this by going to your GitHub repository and clicking the Settings button. On the next page, click Webhooks from the side menu. You should see your Jenkins server webhook in the main interface:



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.

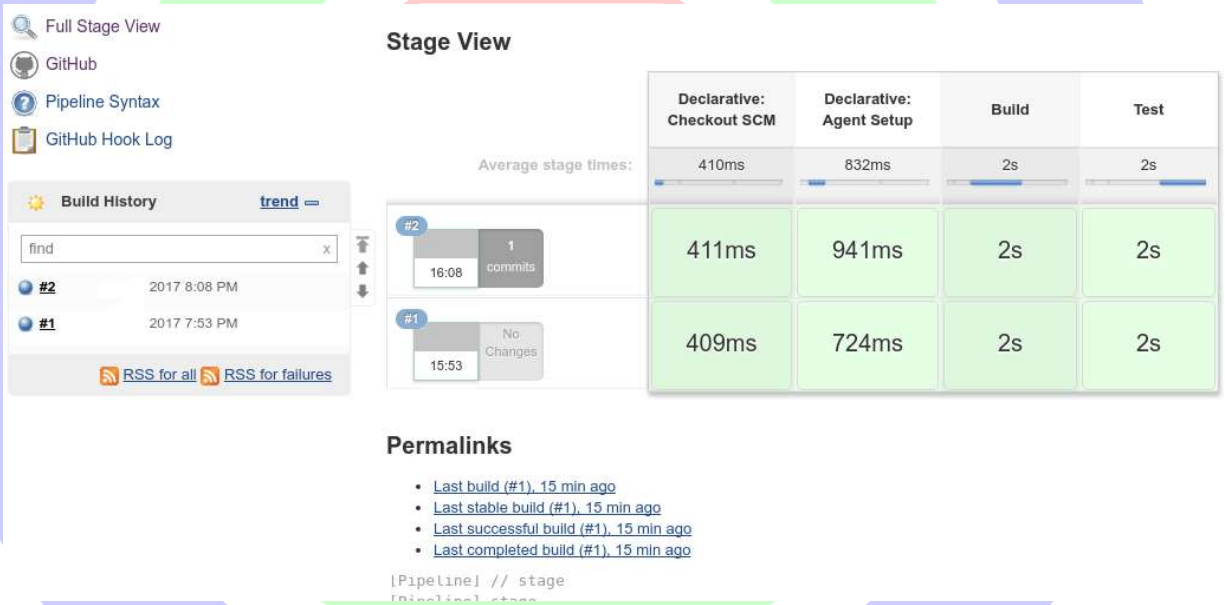
To approximate this, in our repository page on GitHub, you can click the Create new file button to the left of the green Clone or download button:

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:



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
```

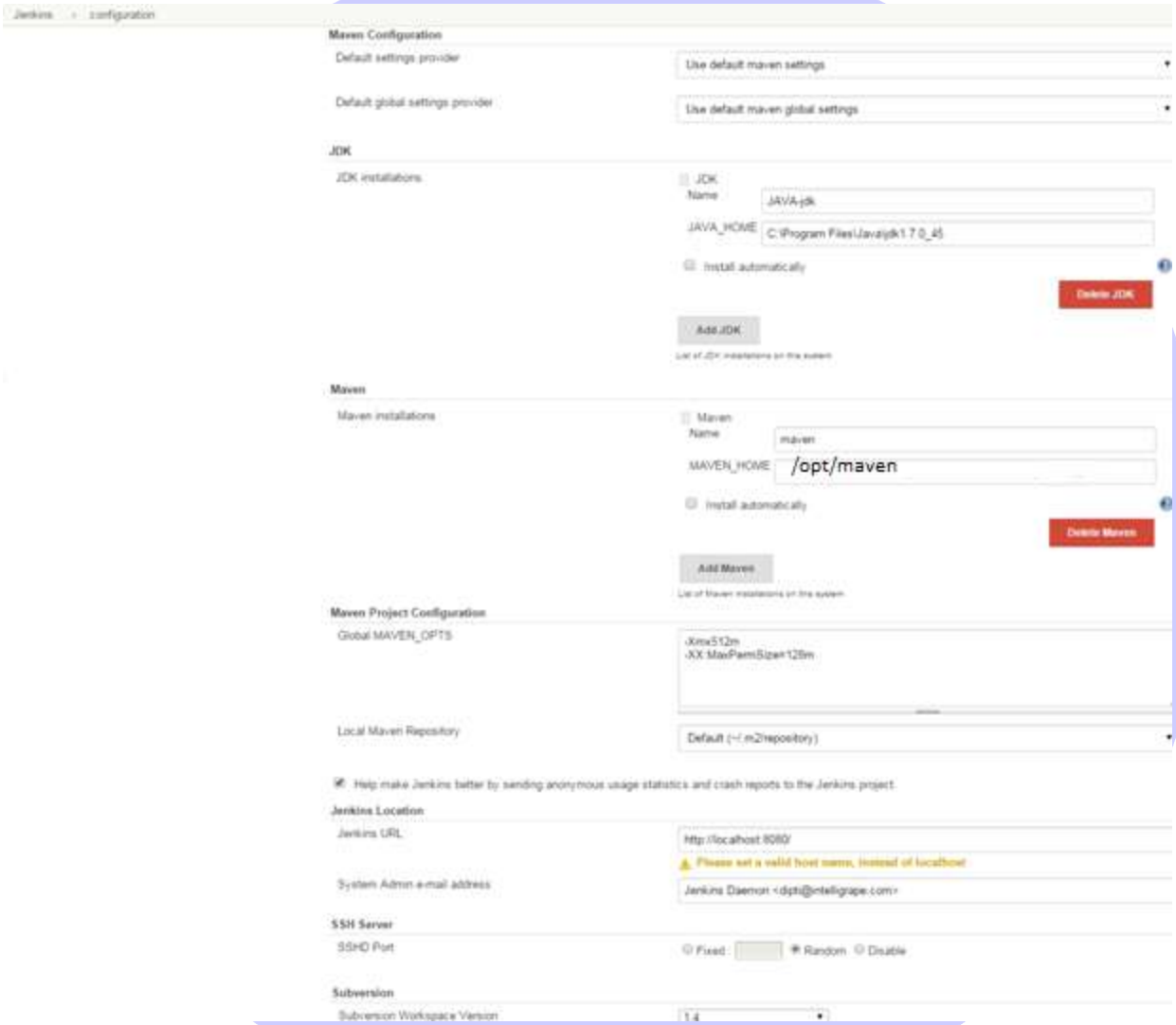
Check the environment variables:

```
# 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.

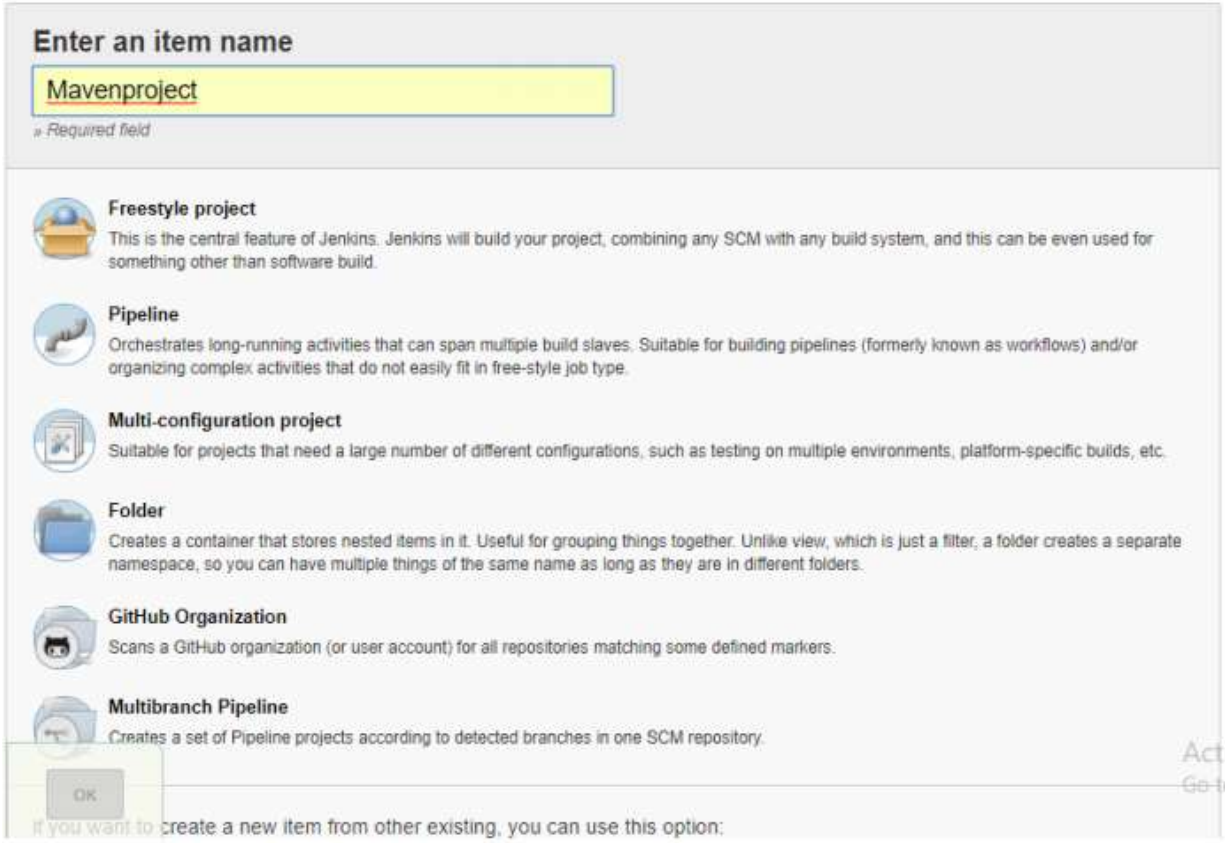


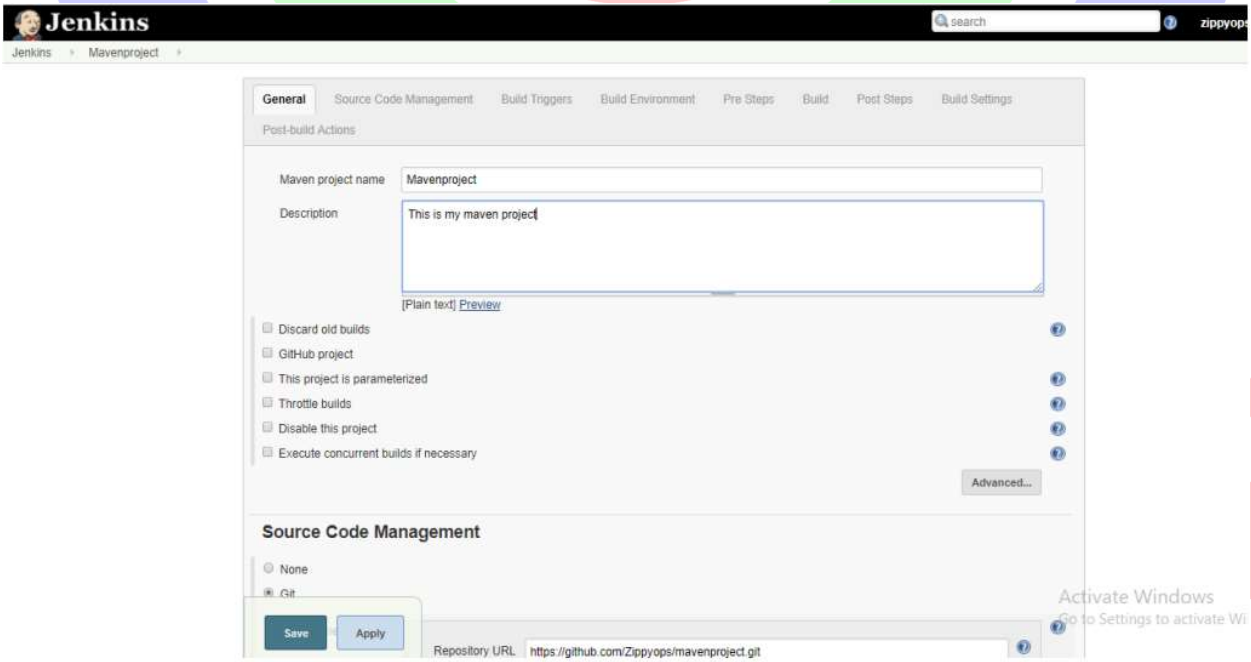
In the global tools configuration, scroll down till you see the Maven section and then click on the ‘Add Maven’ button.

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

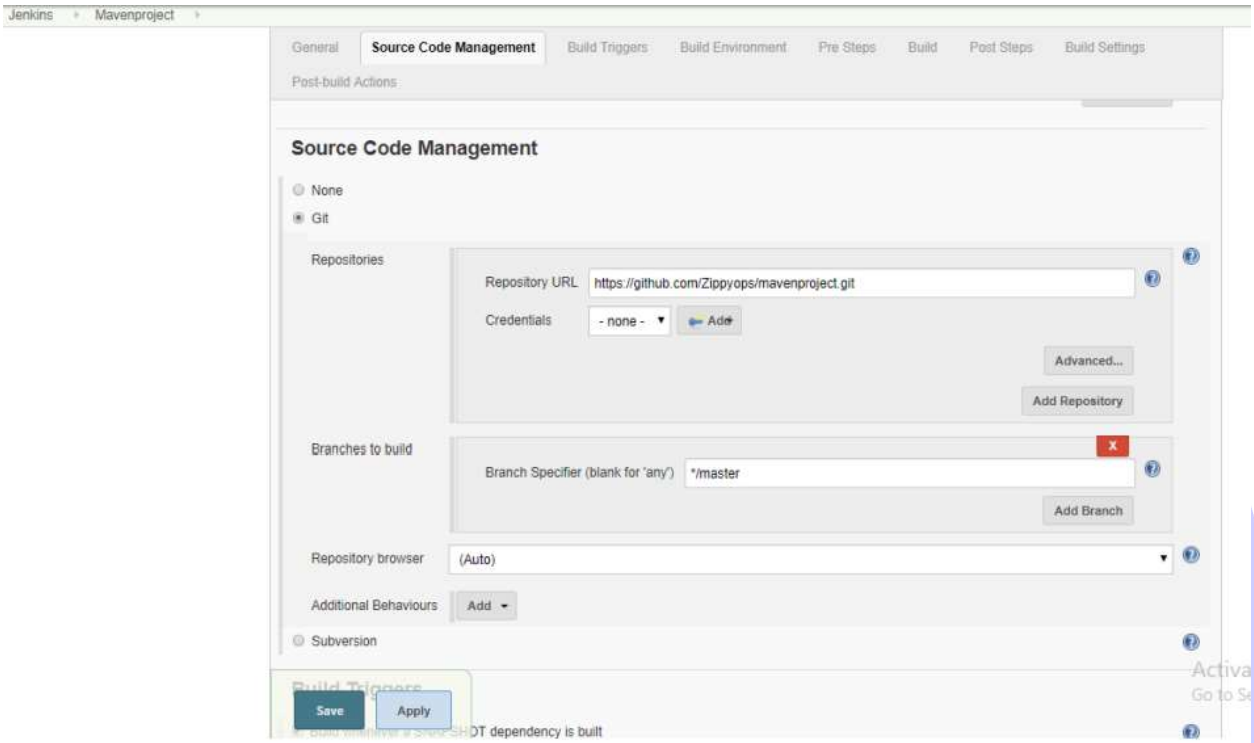
Then, click on the ‘Save’ button at the end of the screen.

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

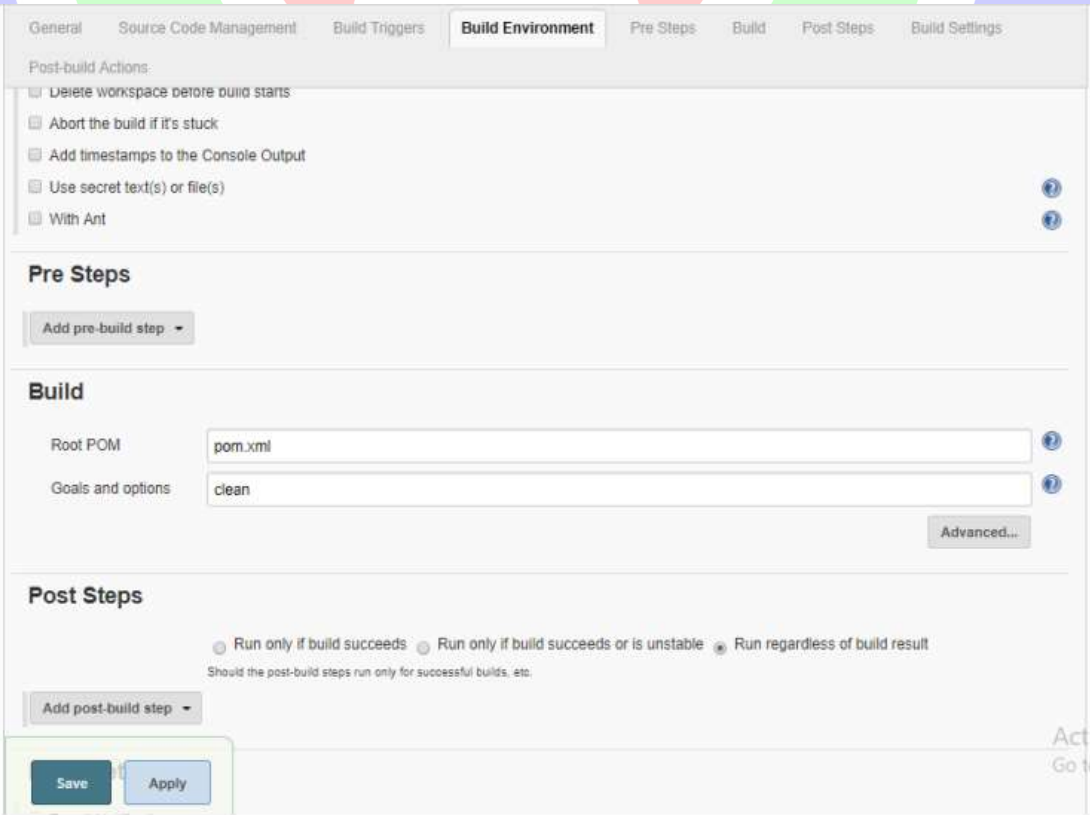




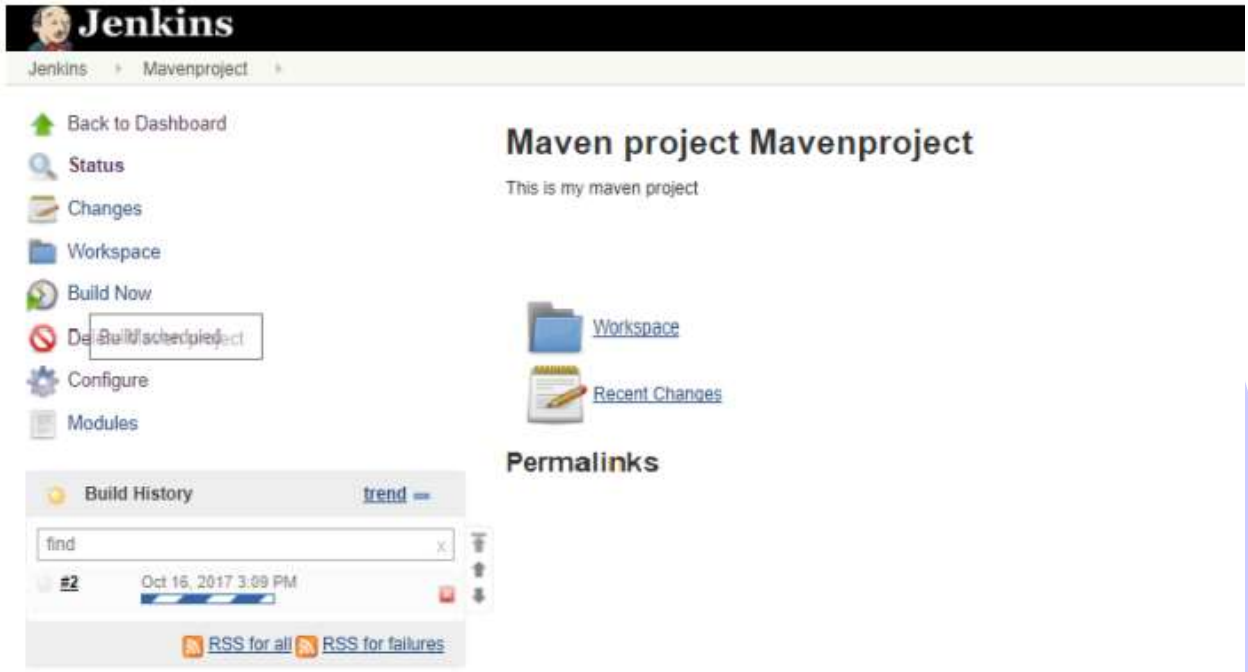
To select source code management, I have selected source code management as git.



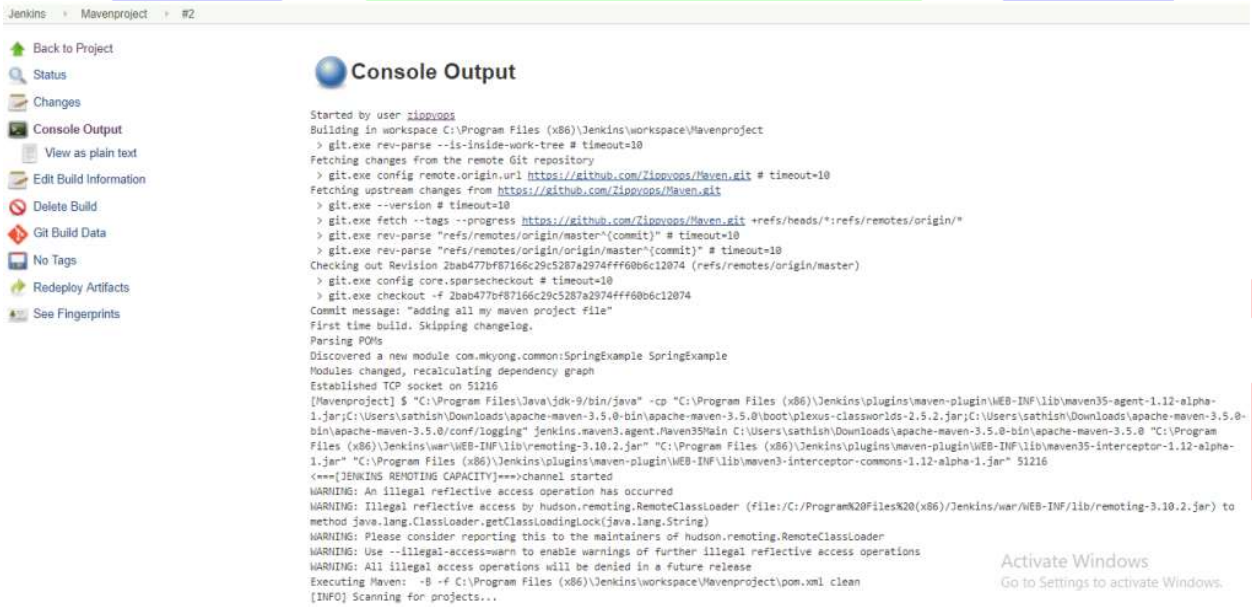
I have added my github repository URL and repository branch as master.



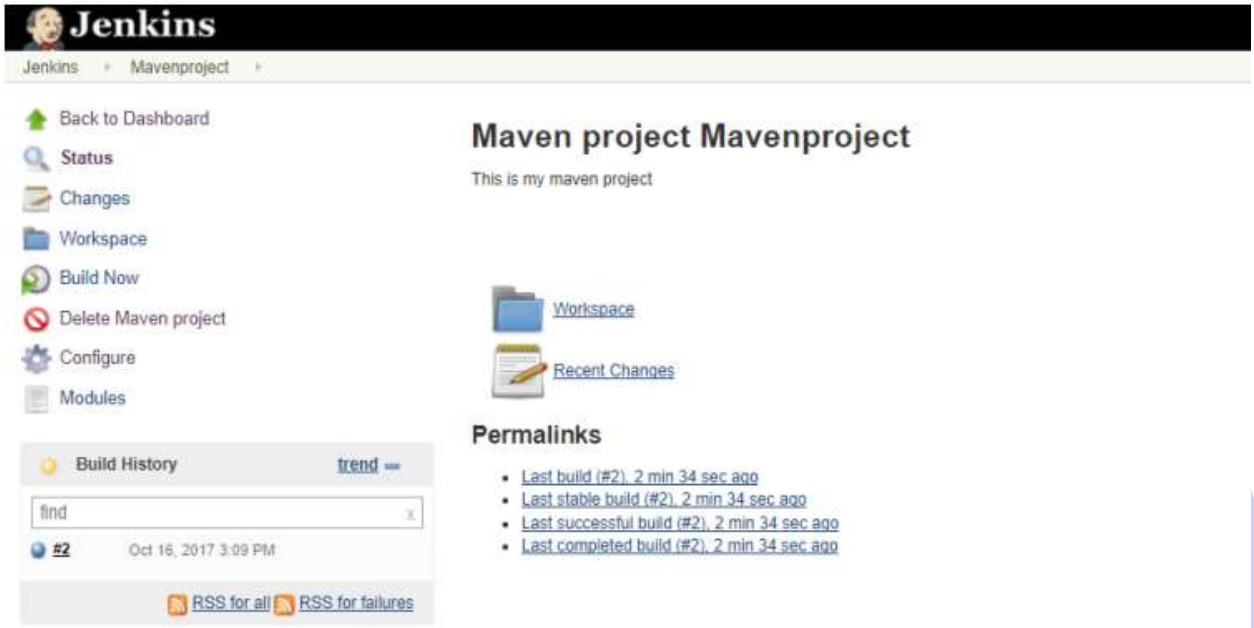
In build Jenkins understands Maven pom files and project structures and added my goal as clean and save the project.



I have clicked build button and build is in process.



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



Output of maven project is success.