**Jenkins tool setup and execution**

**# yum install wget unzip tree git -y**

**# yum install java -y**

To install jenkins, we have add the package to repo and then install it.

**#** **wget -O /etc/yum.repos.d/jenkins.repo** [**https://pkg.jenkins.io/redhat/jenkins.repo**](https://pkg.jenkins.io/redhat/jenkins.repo)

**# rpm --import** [**https://pkg.jenkins.io/redhat/jenkins.io.key**](https://pkg.jenkins.io/redhat/jenkins.io.key)

If you've previously imported the key from Jenkins,

the "rpm --import" will fail because you already have a key. Please ignore that and move on.

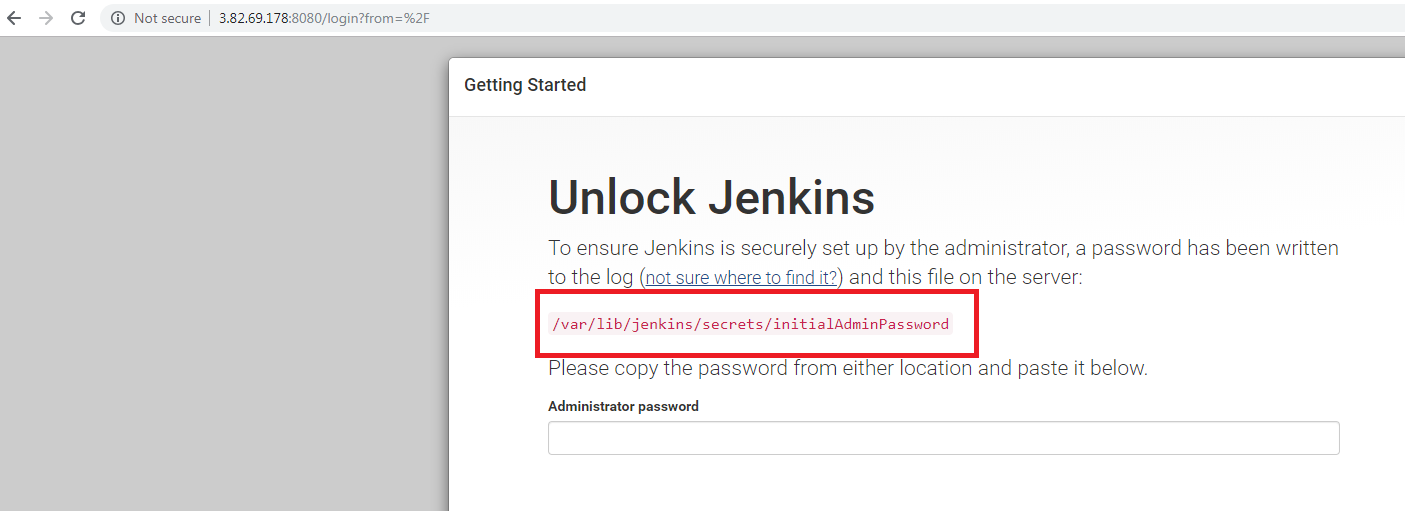
**# yum install jenkins -y**

**# service jenkins start**

* Browse URL with public IP with port number (8080)

**Ex:** <http://54.243.5.107:8080/>

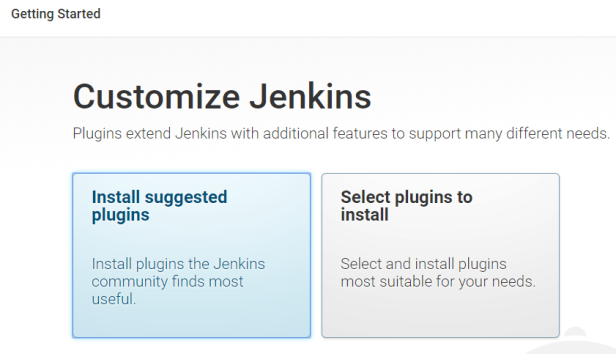
You can see the Jenkins page first time login as below



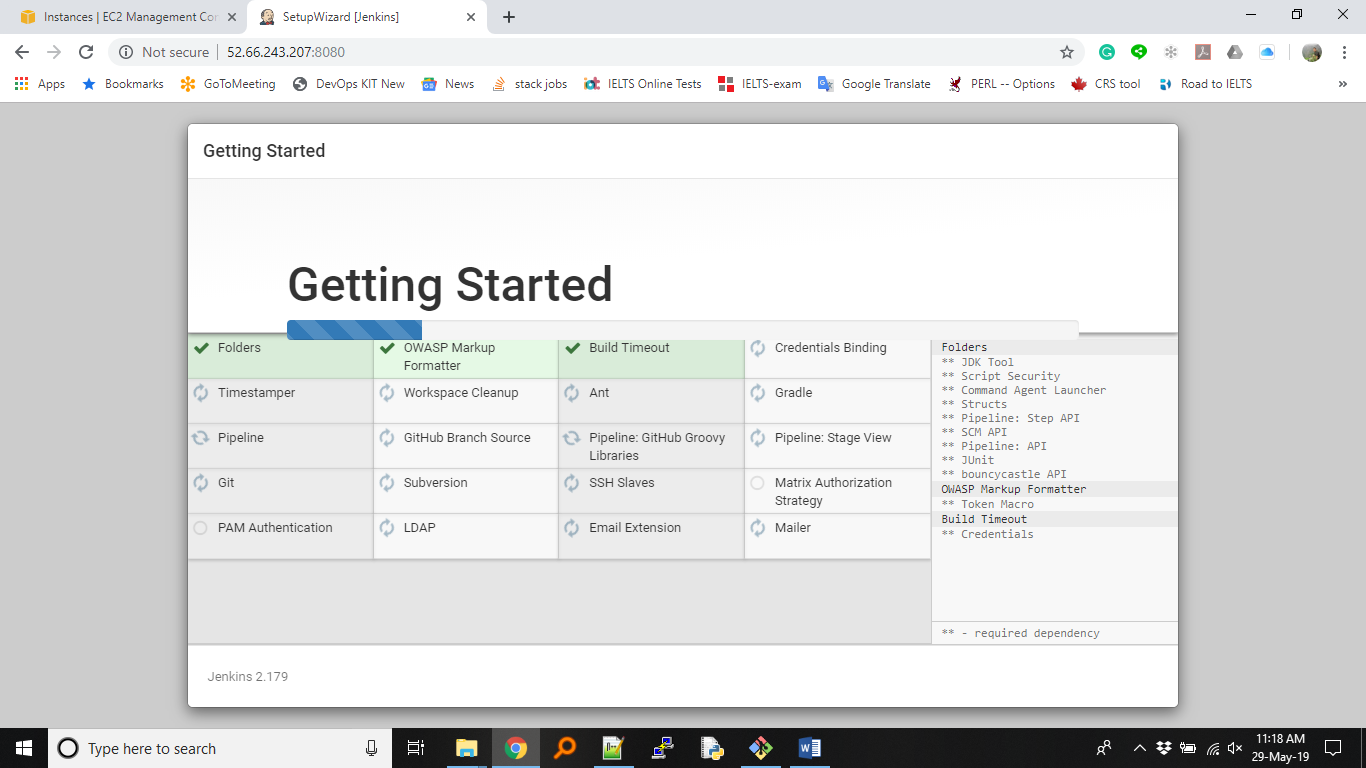
Copy the initial admin password from linux machine from this file (**initialAdminPassword**) and paste on the UI at prompting place.

**# cat /var/lib/jenkins/secrets/initialAdminPassword**

f735142601ba4d71bd32692c4d08946c



Click on install suggested plugins



So now, you are able to see the getting started page.

To check the Jenkins server running or not

**# service jenkins status**

**or**

**# ps -ef | grep jenkins**

[root@jenkinserver ~]# ps -ef |grep jenkins

jenkins 4119 1 10 01:55 ? 00:00:36 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemonized -Djava.awt.headless=true -DJENKINS\_HOME=/var/lib/jenk

ins -jar /usr/lib/jenkins/jenkins.war --logfile=/var/log/jenkins/jenkins.log --webroot=/var/cache/jenkins/war --daemon --httpPort=8080 --debug=5 --handlerCoun

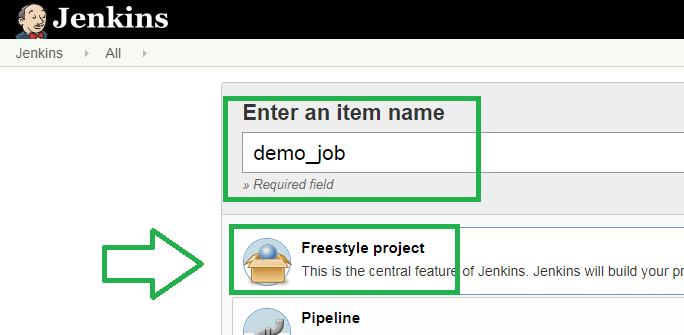
tMax=100 --handlerCountMaxIdle=20

root 4256 4221 0 02:01 pts/0 00:00:00 grep --color=auto jenkins

[root@jenkinserver ~]#

**Exercise 1**

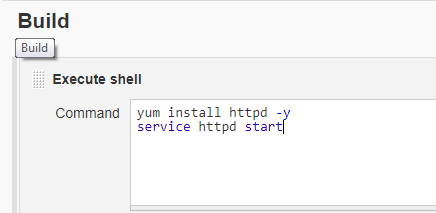
Please create new jobs to get started as demo\_job1

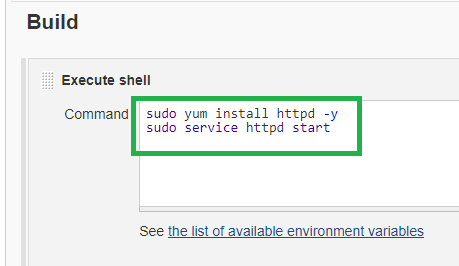


Click on Build tab

Select the option “Execute shell”

Write the below content in the box and save





It will fail even we use sudo command, because, all process running by “jenkins” user.

So, add the “jenkins” user in “sudoers” file at the bottom line to execute commands as a root user.

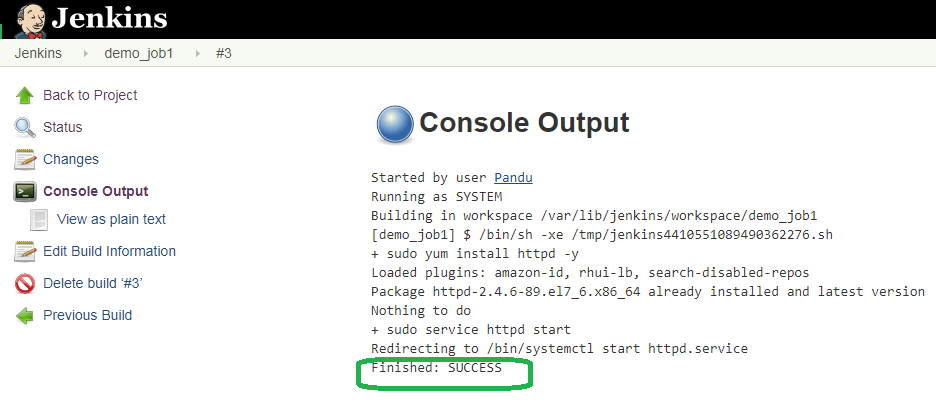
**# vi /etc/sudoers**

#includedir /etc/sudoers.d

ec2-user ALL=(ALL) NOPASSWD: ALL

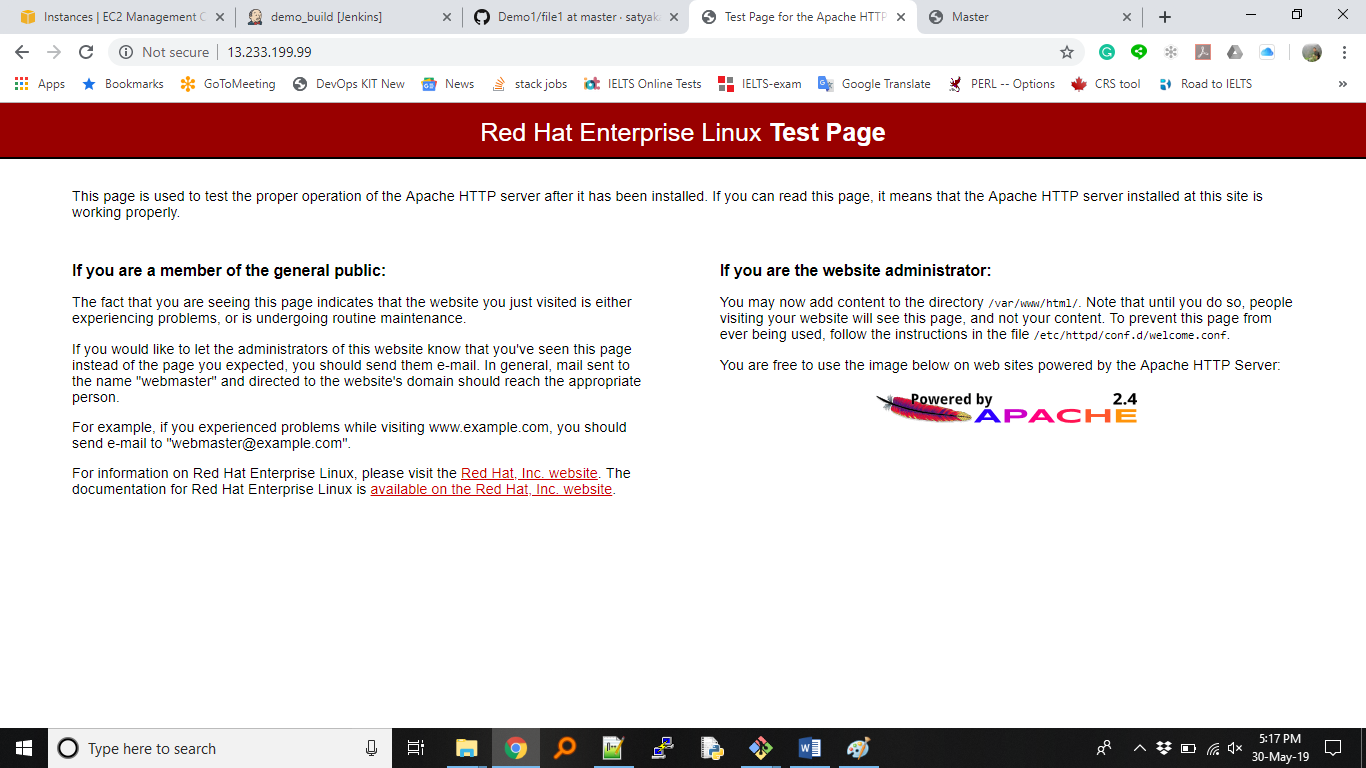
jenkins ALL=(ALL) NOPASSWD: ALL

Then build the job now, it will be successful



Browse the URL with public IP and will get http Apache test page

**Ex:** <http://13.233.199.99>



**Exercise 2**

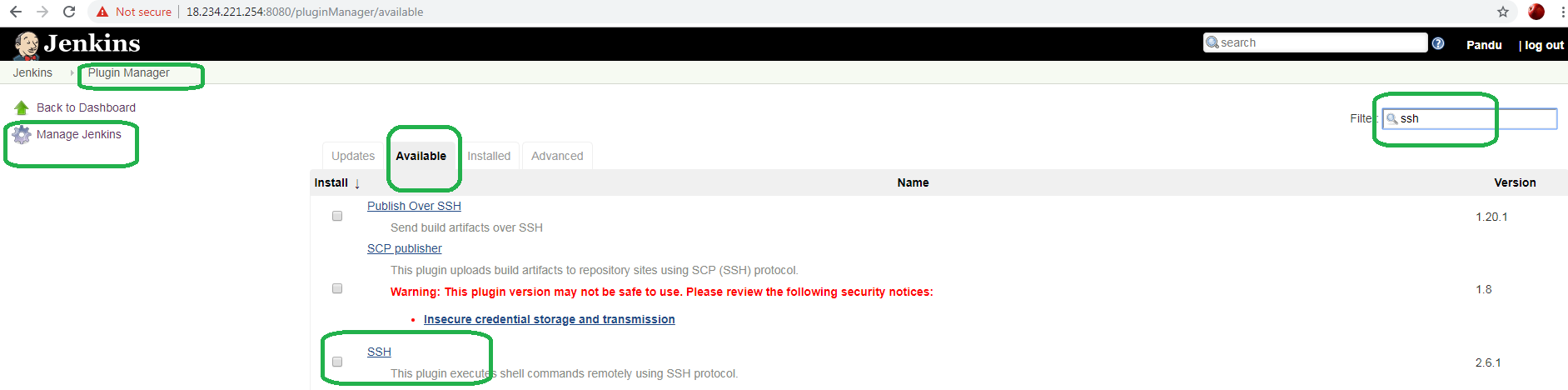
Now, we will try to do activity remote machine using ssh or executing commands on remote machine.

**Step1:** Create another RHEL machine

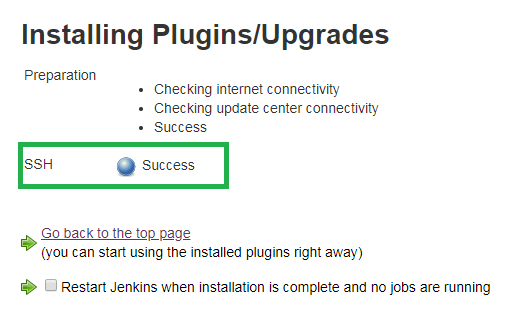
**Step2:** Install SSH plugins in Jenkins server, show that it can talk with remote machine

Jenkins 🡪 Manage Jenkins 🡪 Manage Plugins 🡪 Click on "Available” tab

In filter engine search respective plugin example: SSH

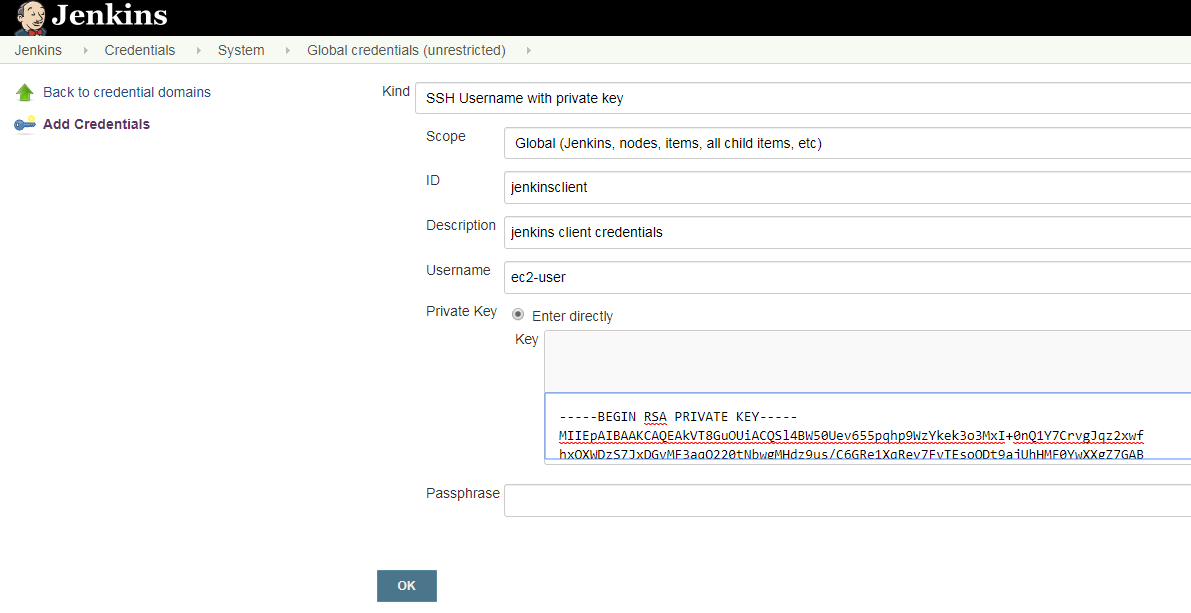


Select the “SSH” plugin -> Click on “Install without restart”



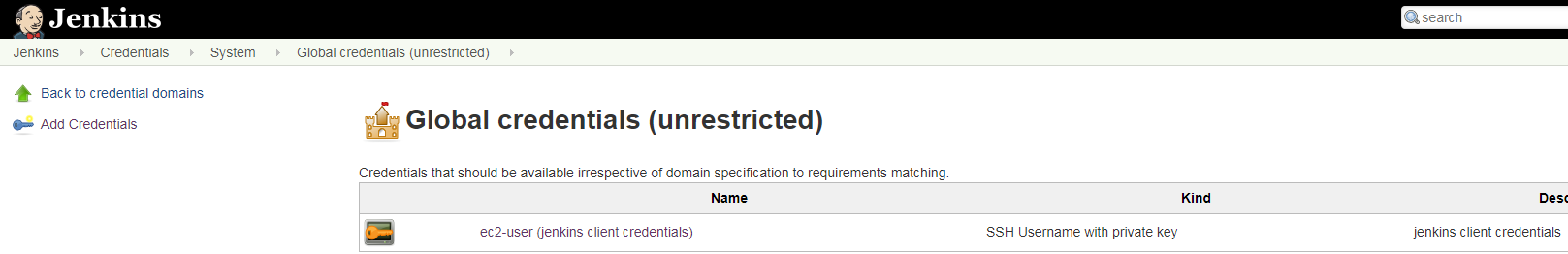
**Step3:** To access remote machine and execute commands on remote machine, need to be setting up credentials in Jenkins server.

Jenkins 🡪 Credentials 🡪 global 🡪 Add Credentials



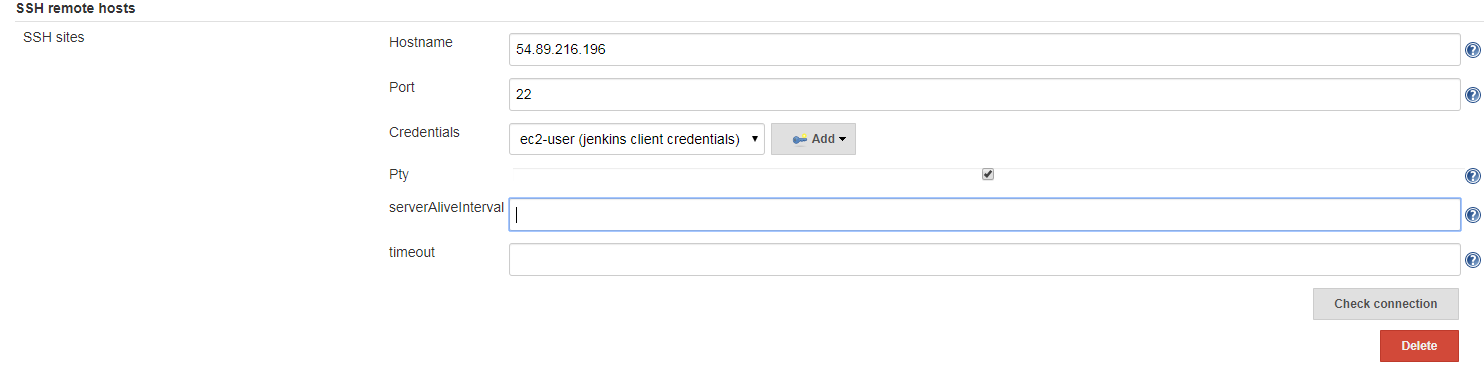
In the Private key box space, paste the pem file contents (Ex: Mumbai23May.pem).

Open the pem file in a notepad, select all the lines (Ctrl+a), copy it then paste it in that Private key box space.



**Step4:** Adding SSH sites,

Jenkins 🡪 Manage Jenkins 🡪 Configure System

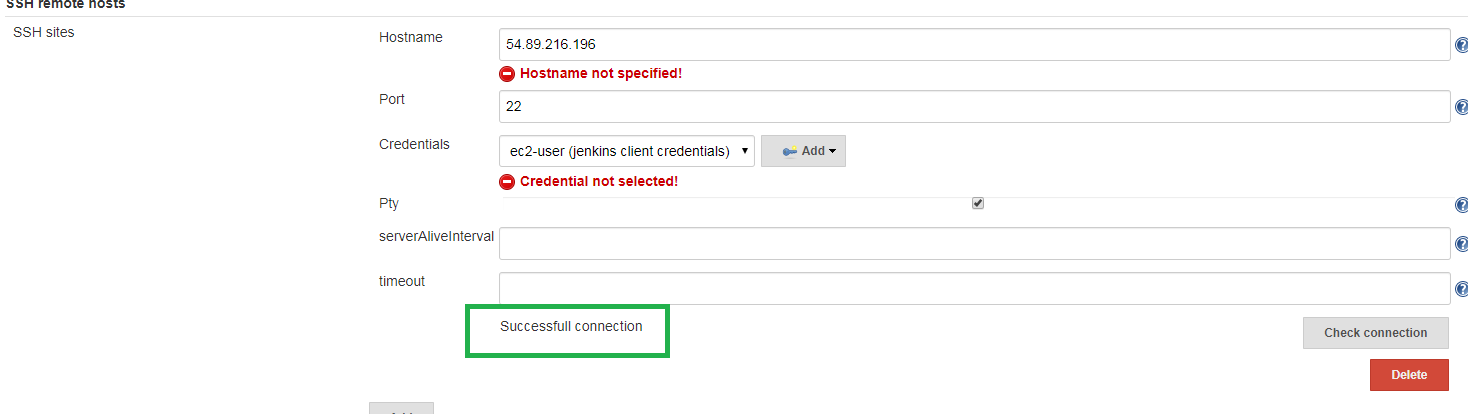


Hostname: 54.89.216.196 (Private IP of newly created remote machine)

Port: 22

Credentials: ec2-user (Jenkins client credentials)

Click on Check connection 🡪 you will see Successful connection

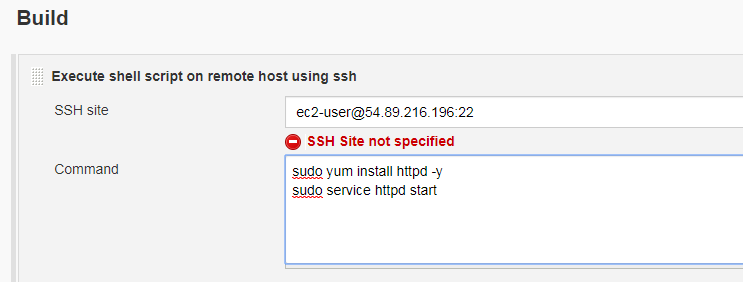


Click on the "save" button at the bottom.

**Step5:** Create a job to execute on remote machine.

Jenkins 🡪 New item 🡪 RemoteJob1 🡪 Freestyle project

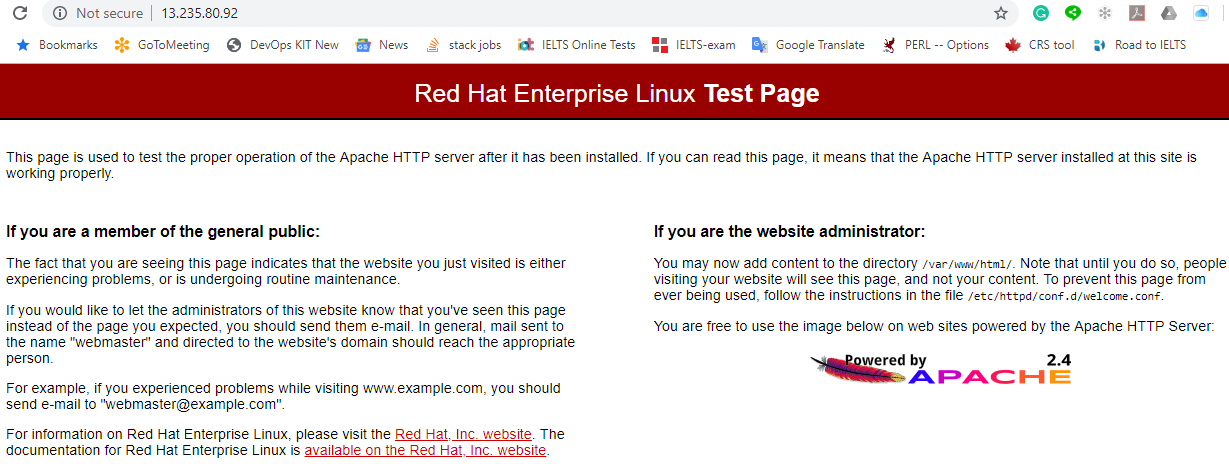
Create new job as remote\_job1



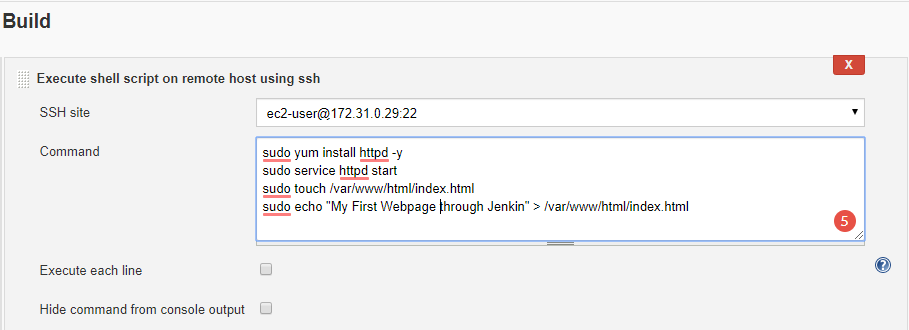
**Save => Build Now.**

It should be successful.

To verify, go to the browser, put the public IP of remote machine in the URL section. It will display http apache page like below.

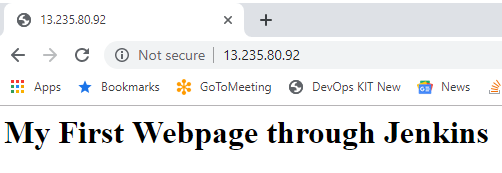


**Step6:** Now, try to create index file in remote machine location with some content.



* Save and Build it again. The latest job should be successful (blue)

Refresh the URL and check it now.



**Exercise 3**

Here we will see a simple war file configuration in Jenkins.

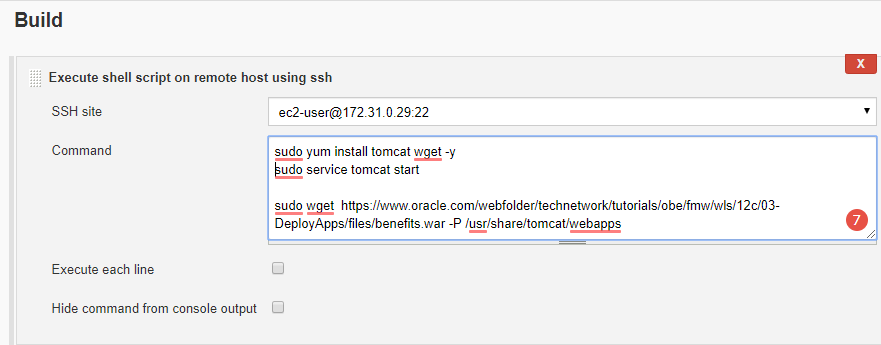
Now download the “benefits.war” from the oracle website through command and access it in webpage.

Command:

**# sudo yum install tomcat wget -y**

**# sudo service tomcat start**

**# sudo wget https://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/03-DeployApps/files/benefits.war -P /usr/share/tomcat/webapp**



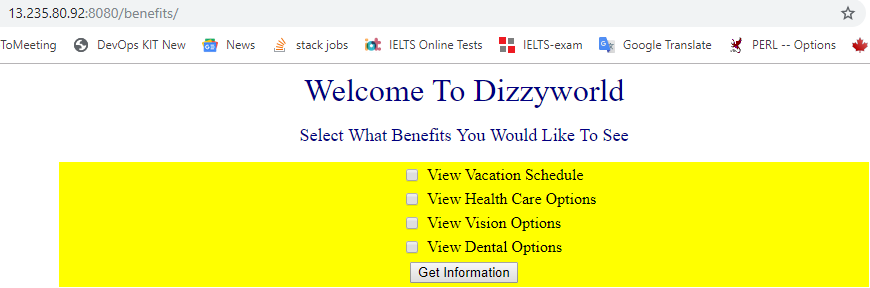
Save -> Build now -> You can monitor the console output -> The build might be successful or failed.

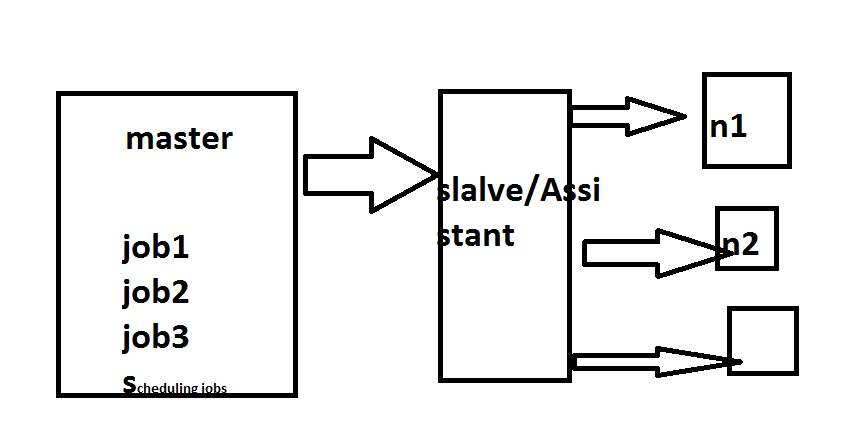
If failed check the log. And debug it.

FYI, Jenkins and Tomcat using same port no 8080. So, if you are using jenkins and tomcat in same server, please change the port no in “**server.xml”** file under /etc/tomcat/ folder.

Now open the browser and check the benefits war file.

**Ex:** [**http://13.235.80.92:8080/benefits/**](http://13.235.80.92:8080/benefits/)



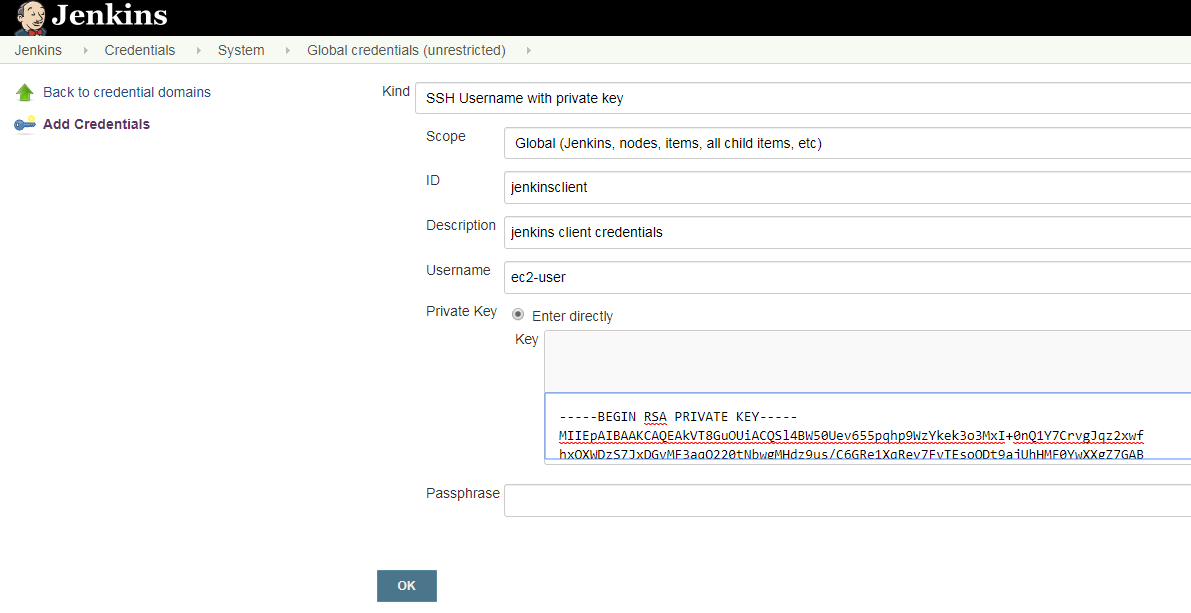
**===========\*\*\*\*\*\*\*\*\* Master & Slave setup \*\*\*\*\*\*\*\*\*=============** 

Create another RHEL machine for slave setup, and named it as “Slave1”.

Create a new credentials using the same pem file for Slave1 machine. Steps are mentioned above in “**Exercise2, step3**”.

Follow that.

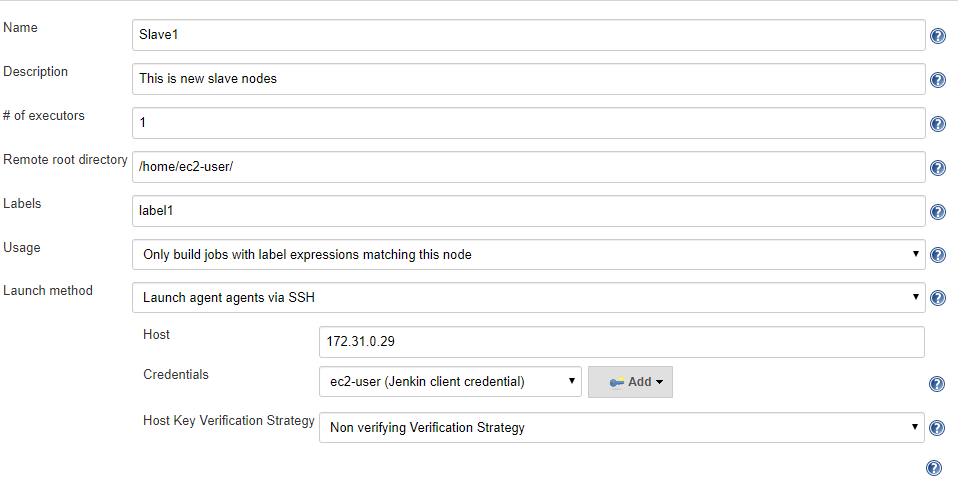
Hints: Jenkins 🡪 Credentials 🡪 global 🡪 Add Credentials

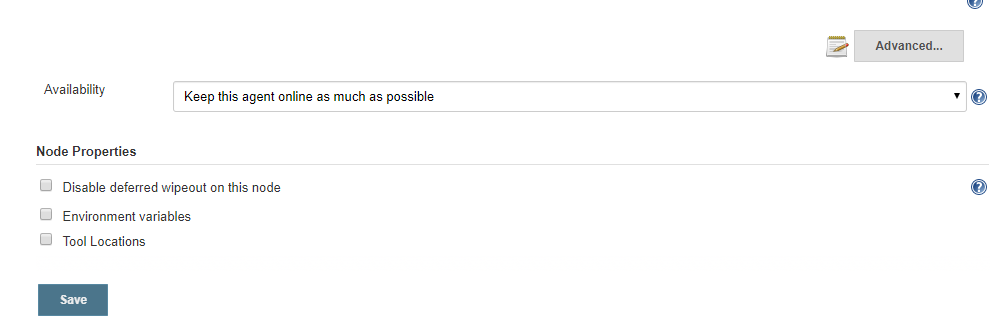


Jenkins --> Manage Jenkins --> Manage Nodes

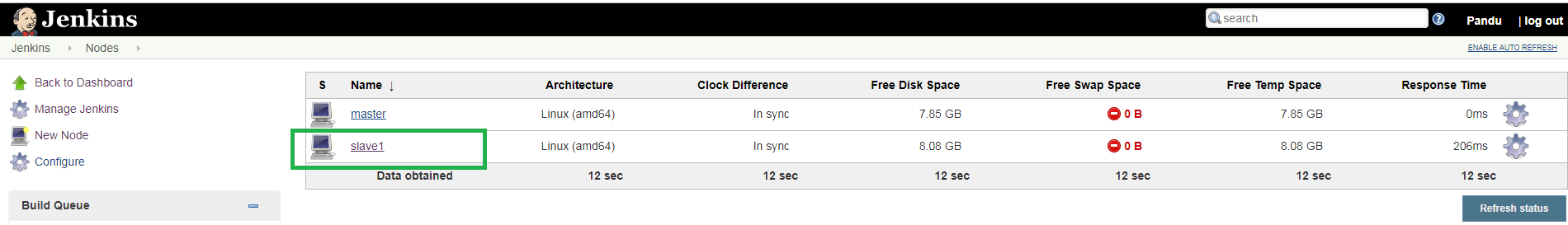
Click on New node --> Provide Node name (Slave1) --> Select Permanent Agent option.



****

****

Provide the details properly and Save.



If the Slave1 is marked with cross symbol, just refresh it. It will show with details. Still if it’s not coming up, then again configure it and read the logs.

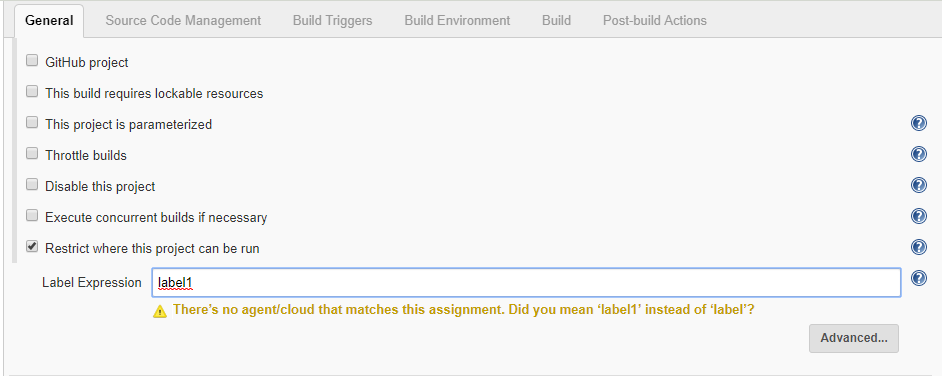
In my case, it says java isn’t installed, so I installed jdk and set the ‘JAVA\_HOME’ path variable in .bash\_profile file and reconfigure it. Now check it might be come up.

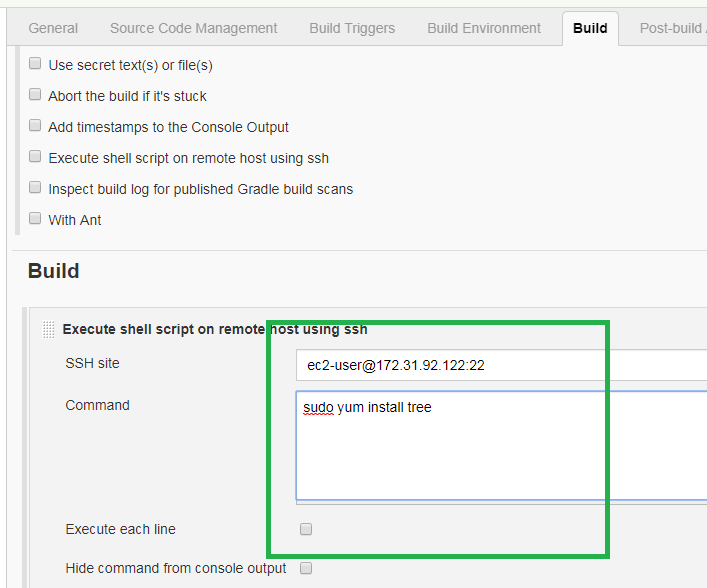
Now, create a job for slave machine.

Jenkins--> New Item-->

Enter an item name "Slavejob1"

Freestyle Project --> click on ok



\*

Save and Build Now.

So far we have ran job on slave machine.

**========= Parameterized Tasks ==========**

**Tasks1:**

Step1: Create a new job "demo\_job5"

**In General section**, check the option "This project is parameterized"

Select "Add Parameter" option as "String Parameter"

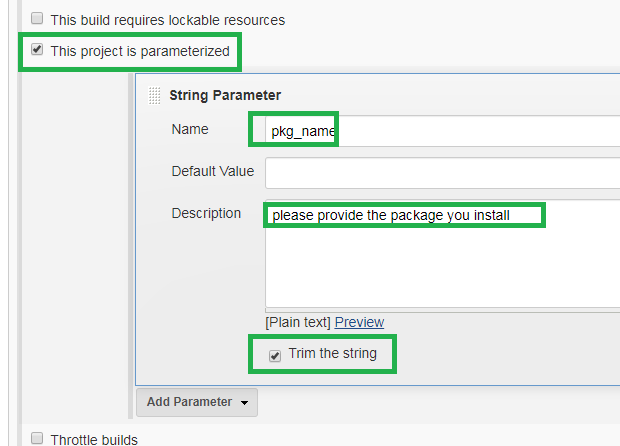
Provide name and description

Example:

Name pkg\_name

Description "please provide the package you install"

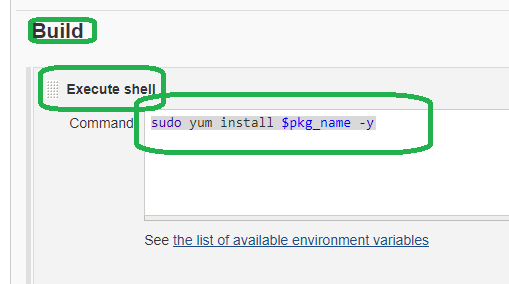
Select "Trim the string"



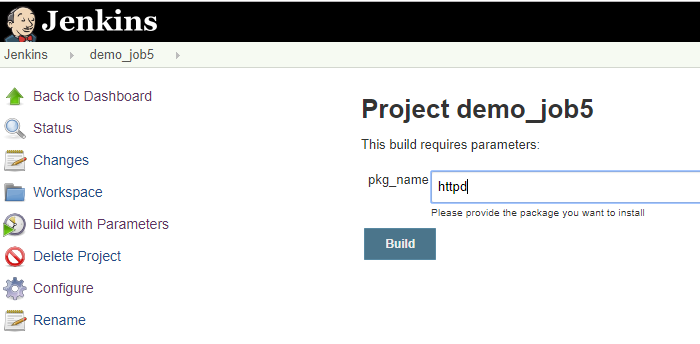
Now, go to “Build” tab and select "Execute Shell"

Provide this command: "sudo yum install $pkg\_name –y”

Save and build.



Click on “Build with Parameters”



* The job will be success.

**Tasks2:**

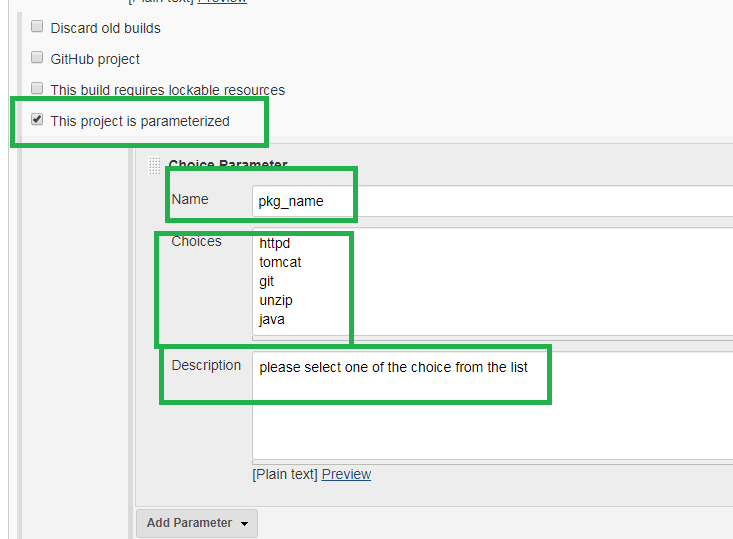
Step1: Create a new job "demo\_job6"

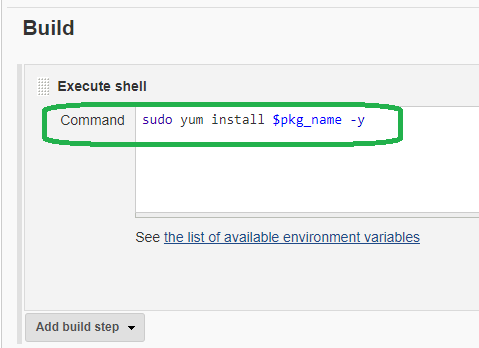
**In General section**, select option "This project is parameterized"

Select "Add Parameter" option as "Choice Parameter"

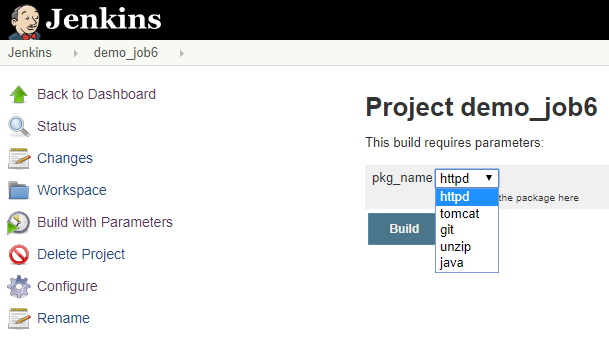
Save and Build.

Provide details as below





Click on “Build with Parameters”



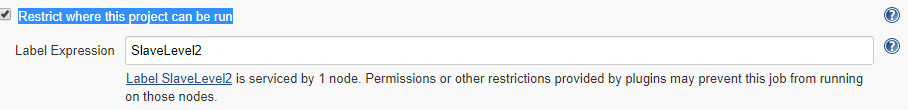
* the job should run successfully

**Few other options under “General” tab.**

General -> Disable this project

General -> Execute concurrent builds if necessary

General -> Restrict where this project can be run



**Home Assignment:**

Q. How to restore jenkins?

1. Take backup of jenkins folder

* # **cp /var/lib/jenkins /tmp/jenkins\_ddmmyy**

1. Stop jenkins

* # **service jenkins stop**

1. Uninstall jenkins

* # **yum remove jenkins –y**

1. Delete jenkins directory

* # **rm –rf /var/lib/jenkins**

1. Install jenkins tool in Remote host

* Commands are available in this note at the starting point.

1. Copy the backup jenkin folder from host to remote host

* # **scp /tmp/jenkins\_ddmmyy ec2\_user@RemoteHost\_publicIP**

1. Open the jenkins page in web browser.

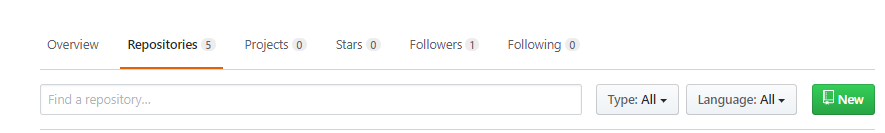
* **publicIP:8080**

**Github webhooking:**

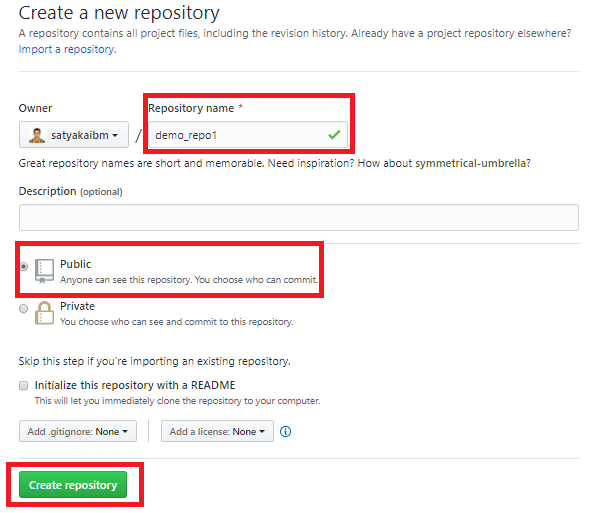
Create an account in github (<https://github.com/>), if you don’t have account there.

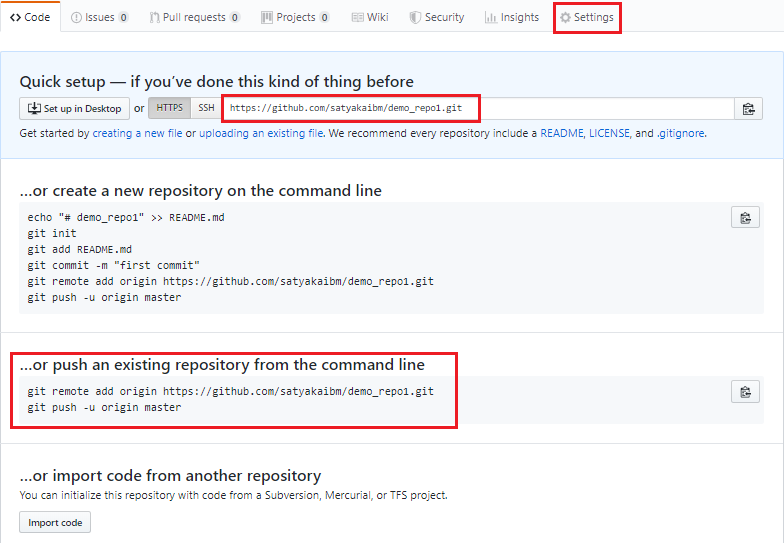
Else, login to github with your credentials.

**Step1:** Create a new repo “demo\_repo1” in github.



Click on New ->





Click on the “Settings” (top right of it)

**Step2:**

Click on Webhooks(left side panel) -> Add webhook ->

**Payload URL\*** “Provide jenkins server url”/github-webhook/

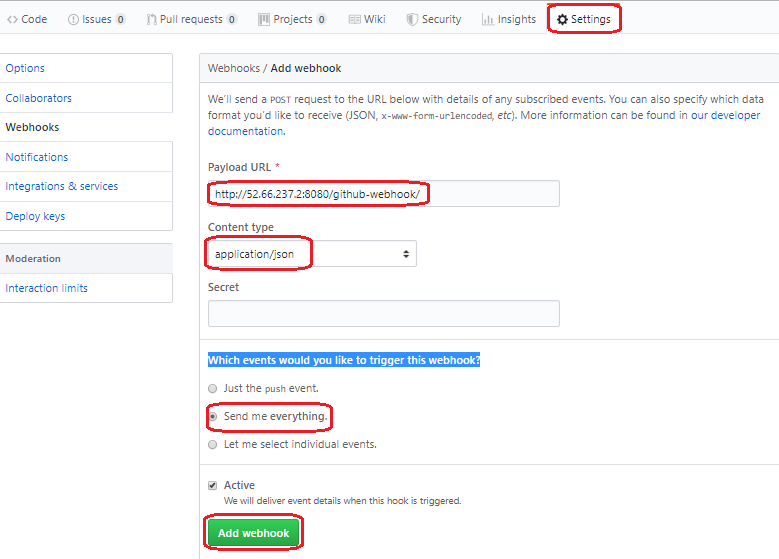
**Content type:** Select “application/json”

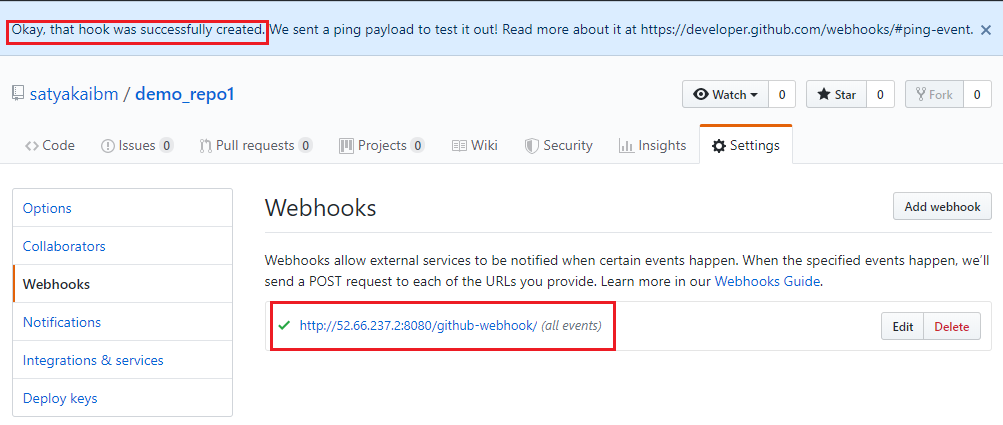
**Which events would you like to trigger this webhook?**

Select “Send me everything” option

**Note:** In Payload URL don’t forget to add “github-webhook/” after the Jenkins URL.

Ex: <http://52.66.237.2:8080/github-webhook/>

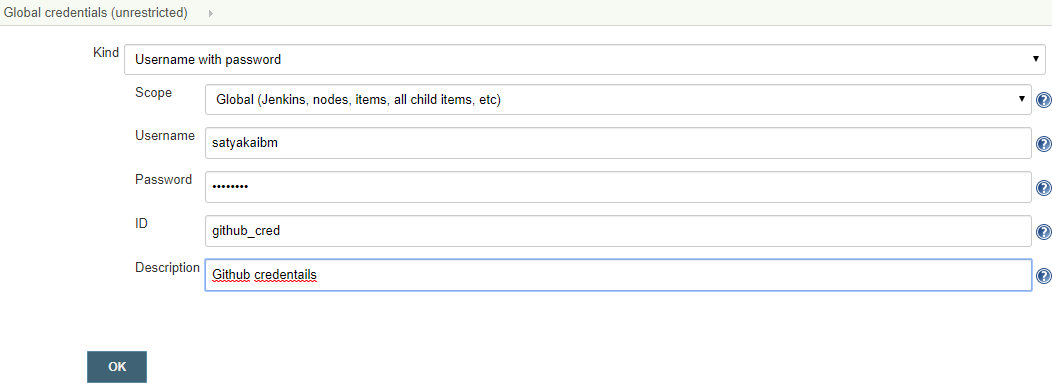




**Step3:**

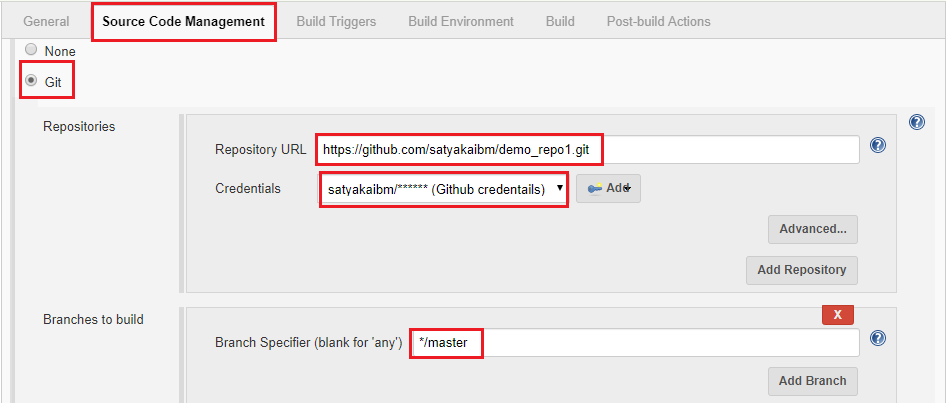
Provide github details URL and credentials in Jenkin.

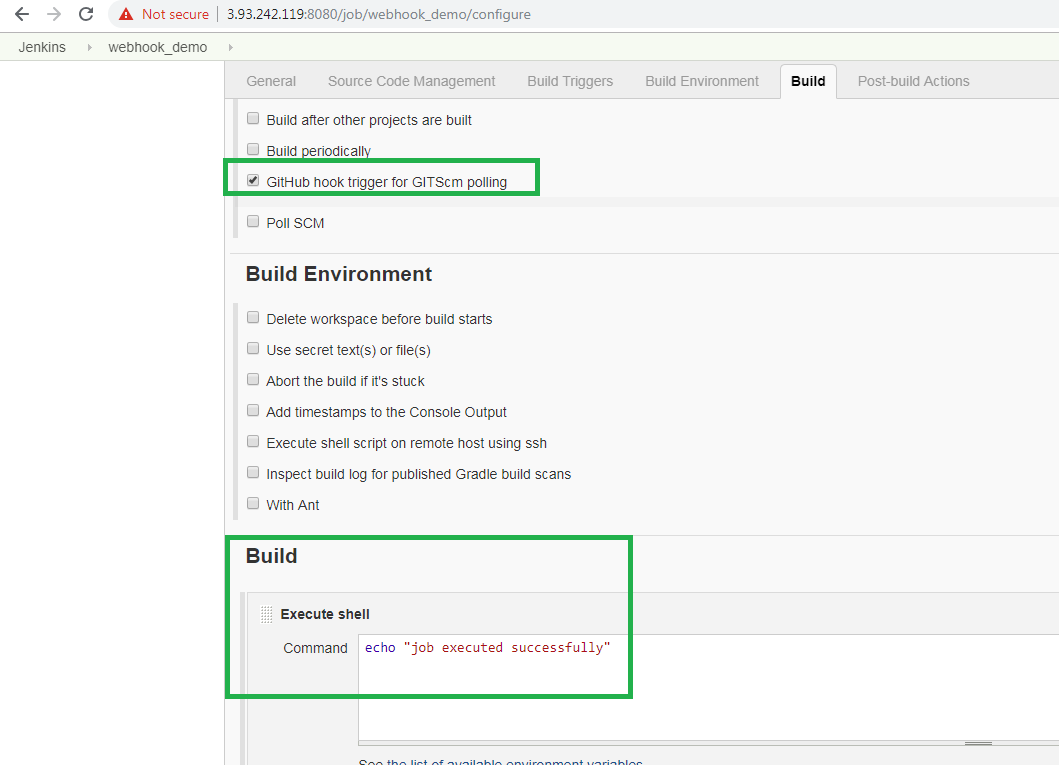
Credentials -> Add Credentials ->



**Step4:** Create a new job to use with github-webhook

Jenkins -> New Item -> (name as: Webhook\_demo)





Save the configuration.

Now, add a new file in the demo\_repo1. You can do either from website or from command prompt.

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject

**$ git clone https://github.com/satyakaibm/demo\_repo1.git**

Cloning into 'demo\_repo1'...

warning: You appear to have cloned an empty repository.

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ git branch**

\* master

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject

**$ cd demo\_repo1/**

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ ls -lart**

total 12

drwxr-xr-x 1 User 197610 0 Jun 10 00:43 ../

drwxr-xr-x 1 User 197610 0 Jun 10 00:47 ./

drwxr-xr-x 1 User 197610 0 Jun 10 00:47 .git/

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ echo "Hello Satya, what can I help for you?" > newFile1.txt**

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ git add newFile1.txt**

warning: LF will be replaced by CRLF in newFile1.txt.

The file will have its original line endings in your working directory

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ git commit -m "Added 1st line" newFile1.txt**

warning: LF will be replaced by CRLF in newFile1.txt.

The file will have its original line endings in your working directory

[master (root-commit) 53ae2fb] Added 1st line

1 file changed, 1 insertion(+)

create mode 100644 newFile1.txt

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ git status**

On branch master

Your branch is based on 'origin/master', but the upstream is gone.

(use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean

User@DESKTOP-QJV3QK7 MINGW64 ~/Documents/GitProject/demo\_repo1 (master)

**$ git push**

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 263 bytes | 52.00 KiB/s, done.

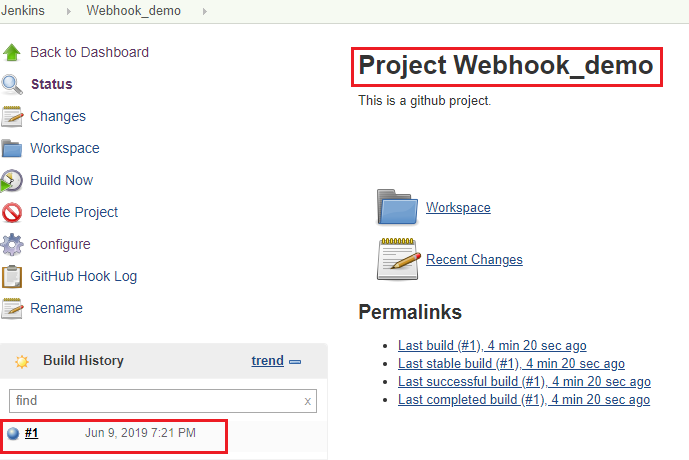
Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/satyakaibm/demo\_repo1.git

\* [new branch] master -> master

Now go to Jenkins and wait for a moment.

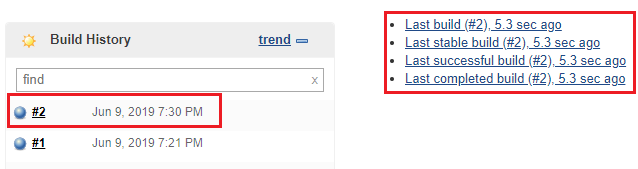
Please don’t click the “Build Now”. The job will be automatically build.



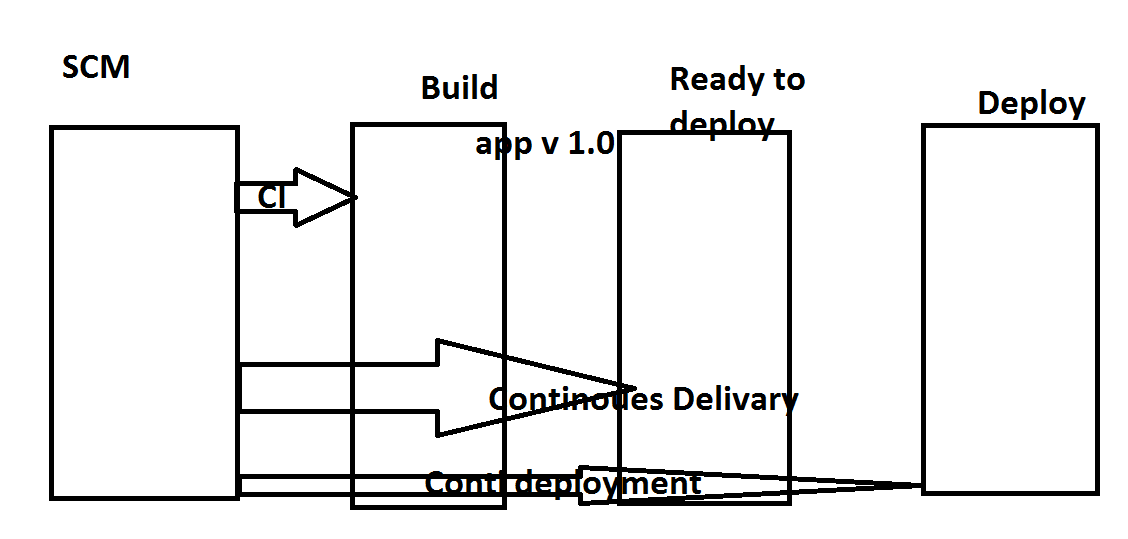
**Step5:**

Again do some changes in the text file and commit it again.

Once you push the file, build will trigger automatically on jenkins. Immediately go to jenkin page and watch.



**===========CI/CD Pipeline=========**



We can create pipeline job in two ways:

* Pipeline code
* Build pipeline.

Pipeline code: It uses Groovy language.

We need to write code in groovy language and in stages mode.

**Exercise1:**

Jenkins --> New item --> Select pipeline and name something as “pipelinejob2”

Build pipeline plugin

**Step1:**

Jenkins-->Manage Jenkins-->Manage Plugins-->

In filter section, search for “build pipeline”

Select “Build Pipeline Plugin”

Install without restart

Create multiple jobs (3 now) like as below

job1: echo "continuous integration getting done"

job2: echo "continuous delivery getting done"

job3: echo "continuous deployment getting done"

go to each job "Configure" --> Build Triggers --> Build after other projects are built -->

Configure each job should trigger one after another job, then run the job.

It will run successfully all jobs.

**Home Exercise:**

Configure build periodically and check once it is running or not periodically.

**Build Triggers**

Using scripts

Build after other projects are build

Job1-> job2 -> job3

If I select this when configure job2 it will going to watch job1

**Note:**

yum install httpd

Don’t give ‘–y’ at the end while we put this cmd in jenkins execute shell, then the cmd will wait for your input for ‘yes or no’ So it won’t be completed. It’s called unstable job.

**Build Periodically**

Like cronjobs.

* \* \* \* \* \*

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

|  |  |
| --- | --- |
| MINUTE | Minutes within the hour (0–59) |
| HOUR | The hour of the day (0–23) |
| DOM | The day of the month (1–31) |
| MONTH | The month (1–12) |
| DOW | The day of the week (0–7) where 0 and 7 are Sunday. |

echo “hello world”

**Poll SCM**

Configure Jenkins to poll changes in SCM.

Note that this is going to be an expensive operation for CVS, as every polling requires Jenkins to scan the entire workspace and verify it with the server. Consider setting up a "push" trigger to avoid this overhead, as described in [this document](https://jenkins.io/redirect/scm-change-trigger).

every min Jenkin will check in SCM, if there is any changes in SCM it will trigger.

**Interview Question:**

Difference between Poll SCM and github web hook?