JENKINS HANDS-ON

DevOps Certification Training

support@intellipaat.com
+91-7022374614
US: 1-800-216-8930(Toll Free)

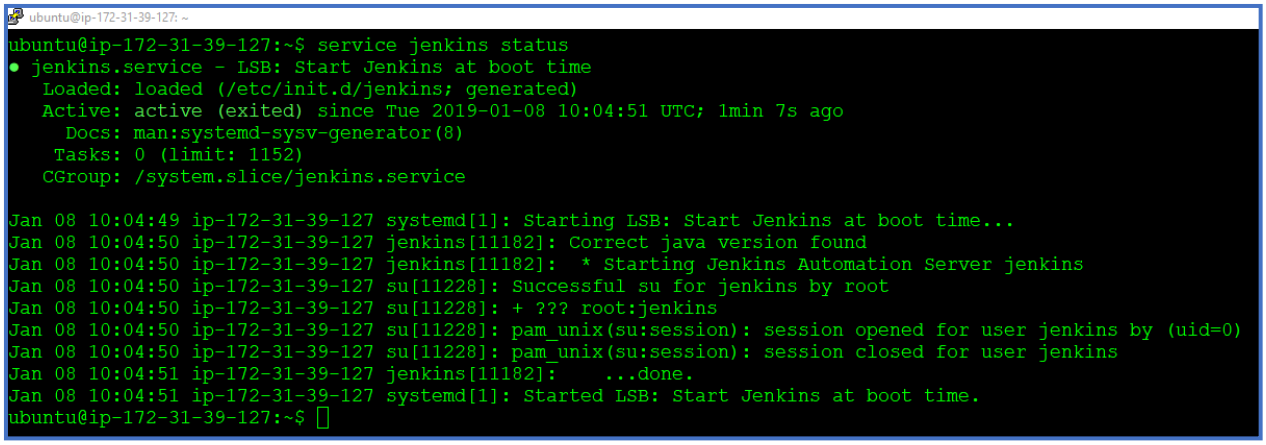
JENKINS HANDS-ON

1. **Create 3 instances (Master, Slave-1, Slave-2) on EC2 server.**
2. **Install Jenkins on Master. (Refer to the Jenkins installation documentation)**
3. **Set up a Jenkins Master-Slave Cluster on AWS**
4. **Create a CI CD pipeline triggered by Git Webhook.**

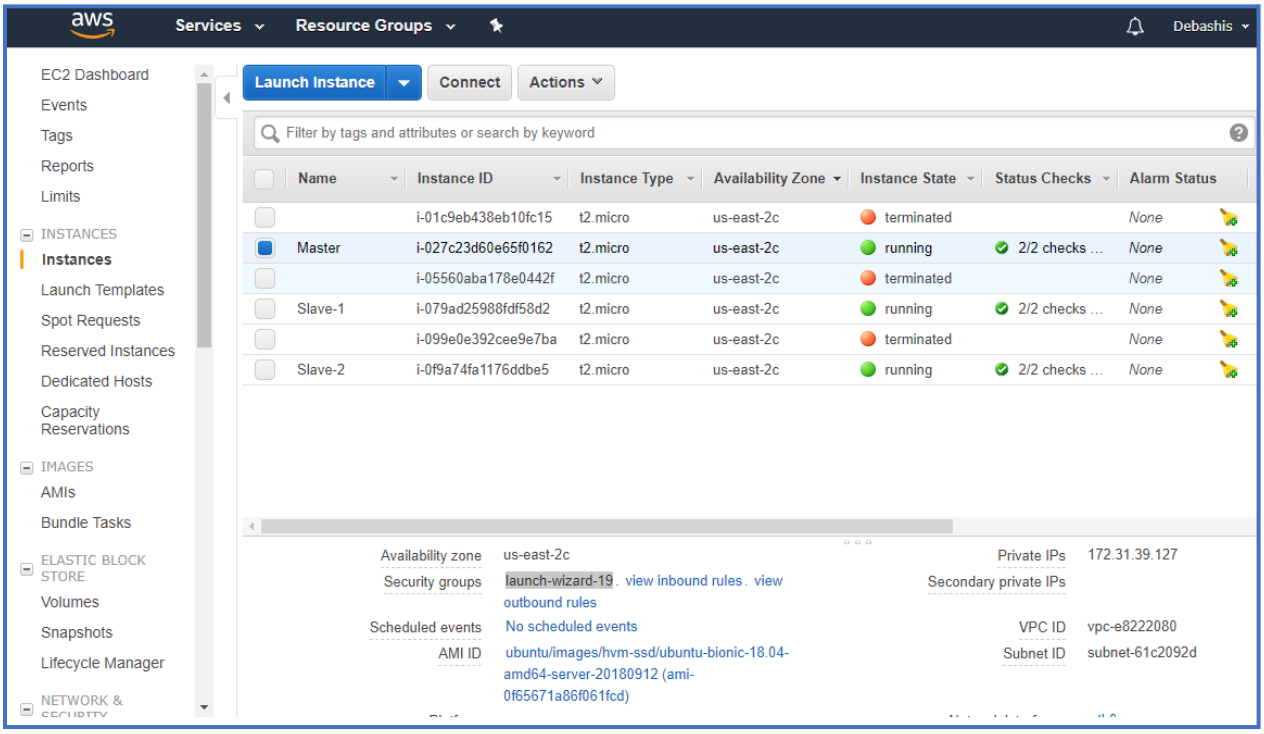
First, we have created 3 instances Master (Green terminal), Slave-1(Orange Terminal) and Slave-2(Blue Terminal) on EC2 Server. And then we have installed the Jenkins on Master Machine. Now Let us set up the Jenkins Master-Slave Cluster.

Step 1: Check the status of the Jenkins first.

```
$ service jenkins status
```



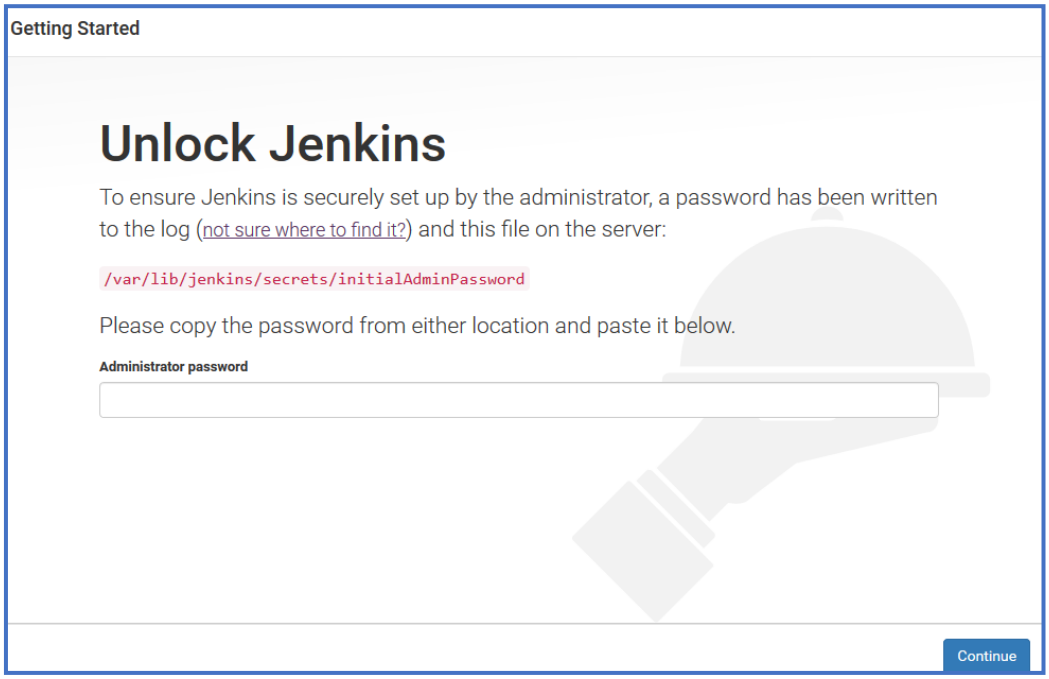
Step 2: Got to EC2 server. Select Master click on ***launch-wizard-xx.***



Step 3: Go to inbound connections. Click on edit. Edit the inbound rules as shown below. Then save the changes.

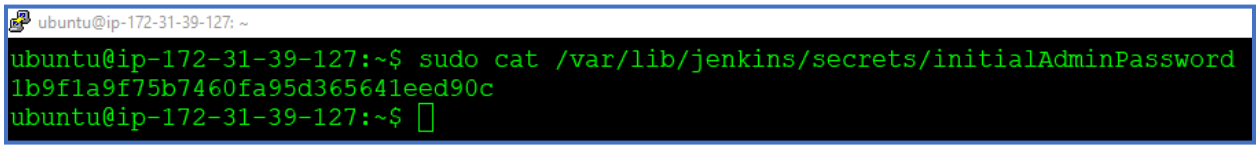
Step 4: Now open browser and enter ***masterIP:8080***

You should land on a page like this:



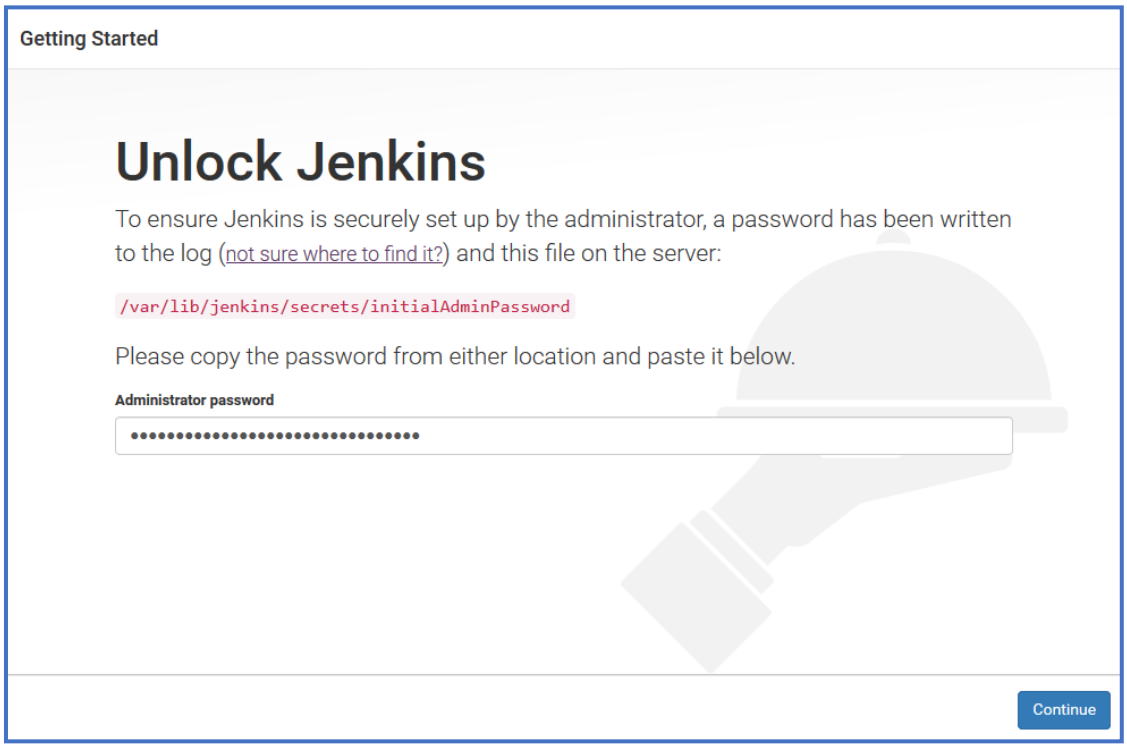
Step 5: Copy the path mentioned in the page and perform cat operation in master terminal.

```
$ sudo cat <path>
```



This will give us the password which we will use to unlock our Jenkins.

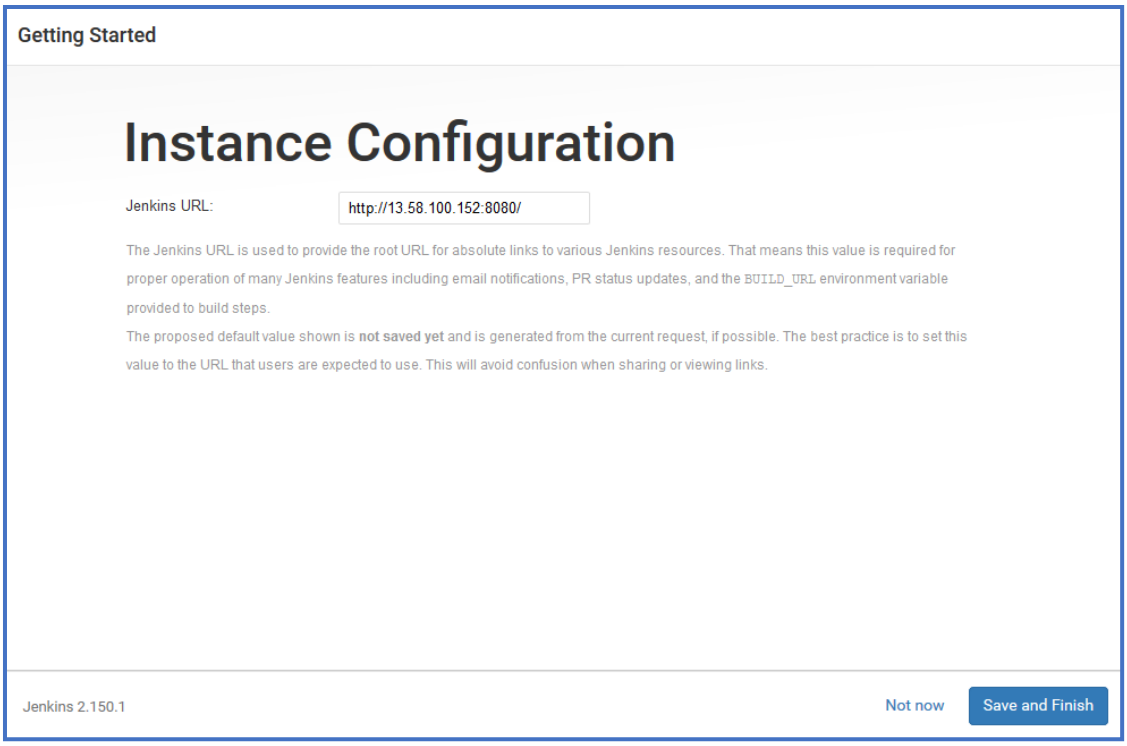
Copy the password from there and paste it on the Jenkins Server page.



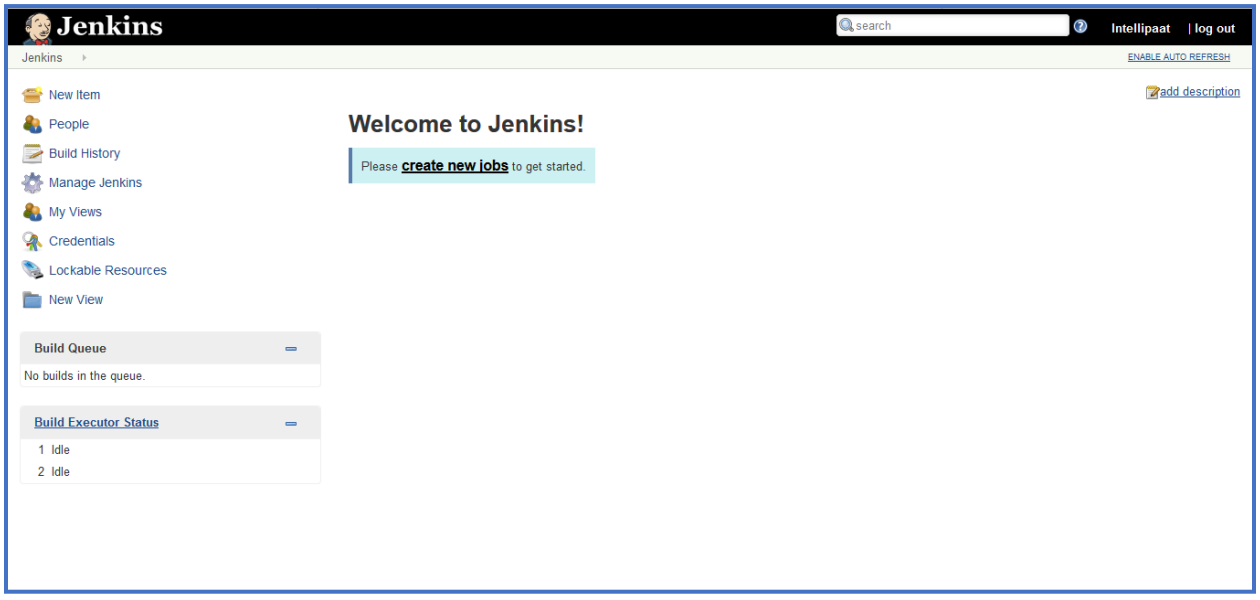
Now click on continue. Then click on Install Suggested Plugins.

Step 6: Once done, enter the Admin User details.

Then click on **Save and Continue**.



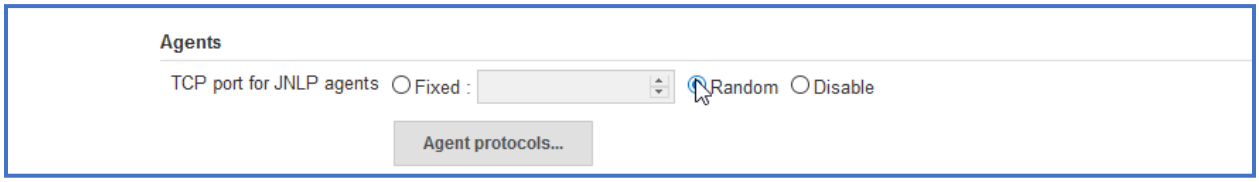
Again, click **Save and Finish**. Click on **Install Suggested Plugins.** Once it's done we will land on a page as shown below.



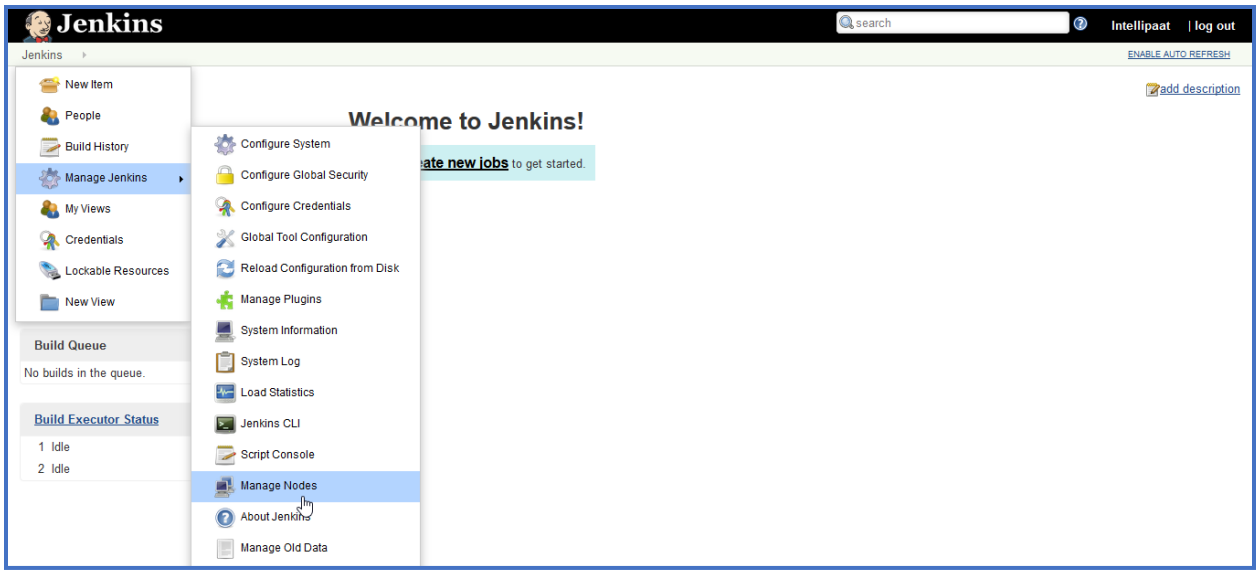
This is our **Jenkins Dashboard**.

Step 7: Go to **Manage Jenkins**. Click on **Configure Global Security.**

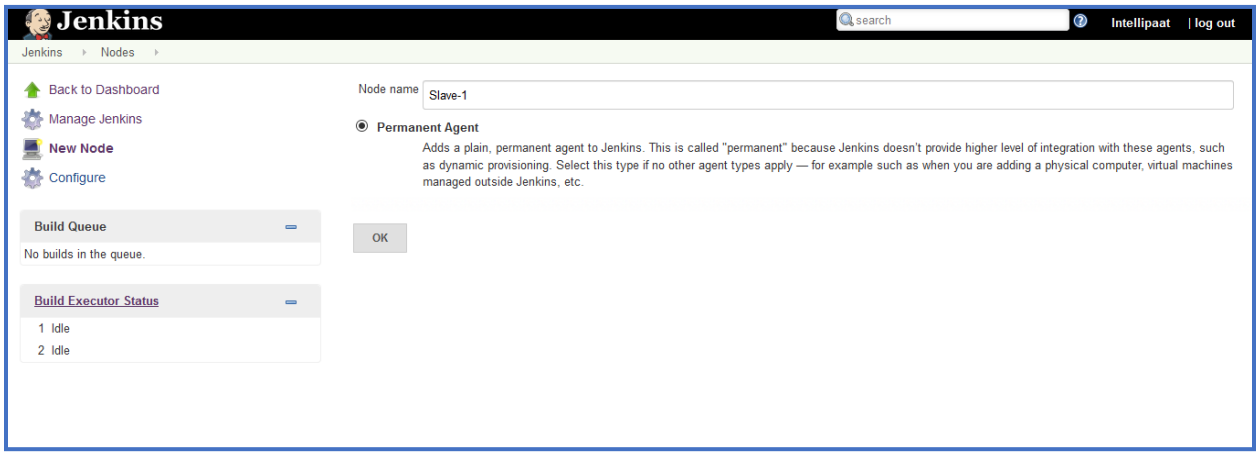
Step 8: Change the **Agents** to **Random**. Then click on **Save**.



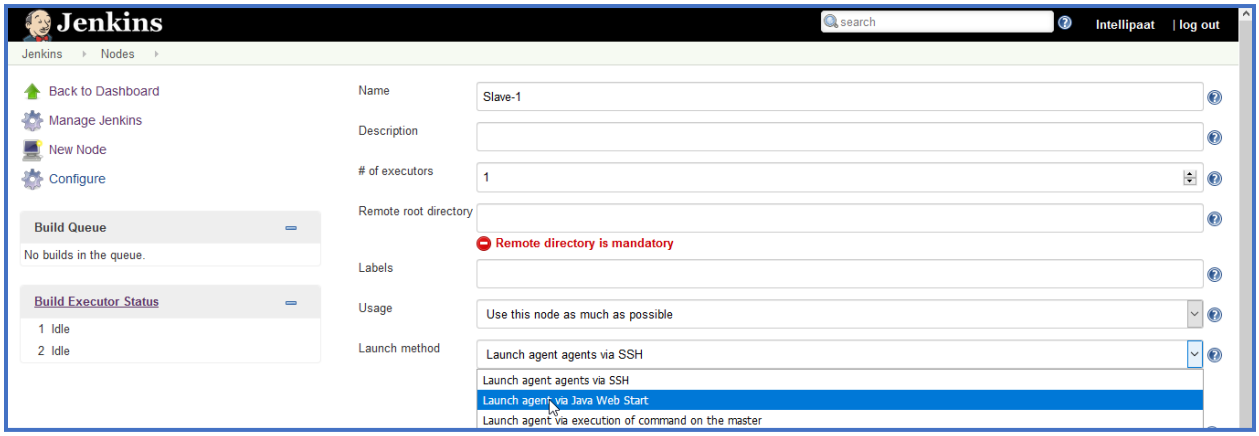
Step 9: Now go to **Manage Nodes**.



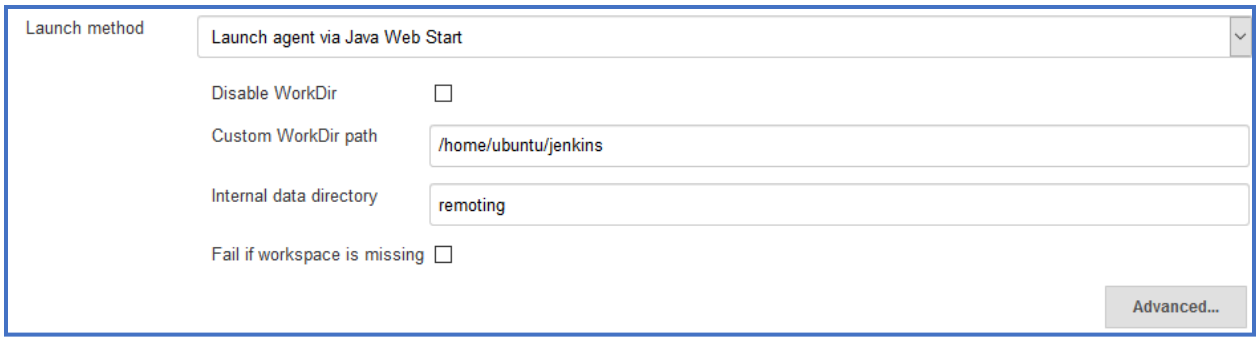
Step 10: Click on **New Node.** Add **Slave-1** as new node and make **Permanent Agent**. Click on **ok**.



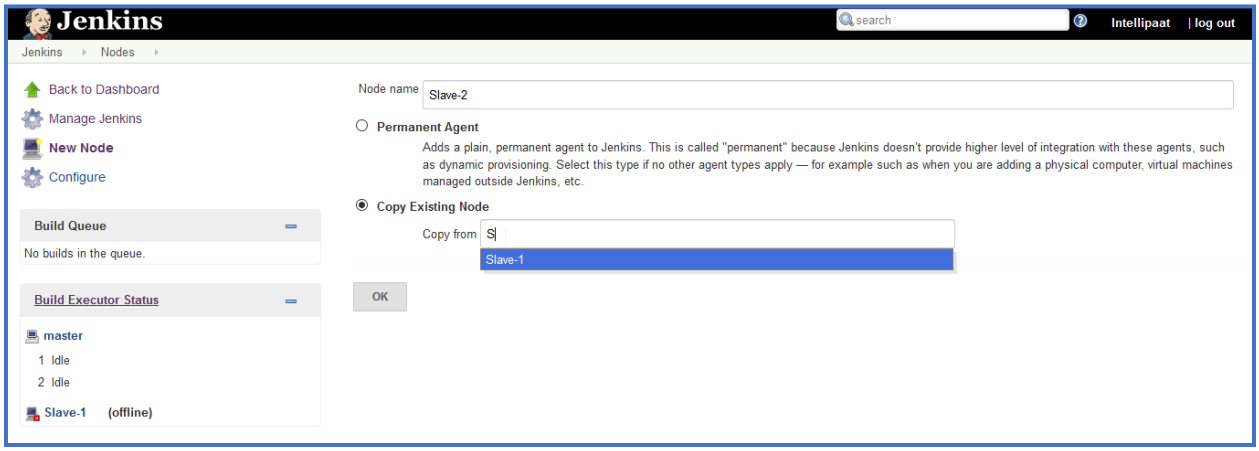
Step 11: Go to **Launch method** change it to **Launch agent via Java Web Start.**



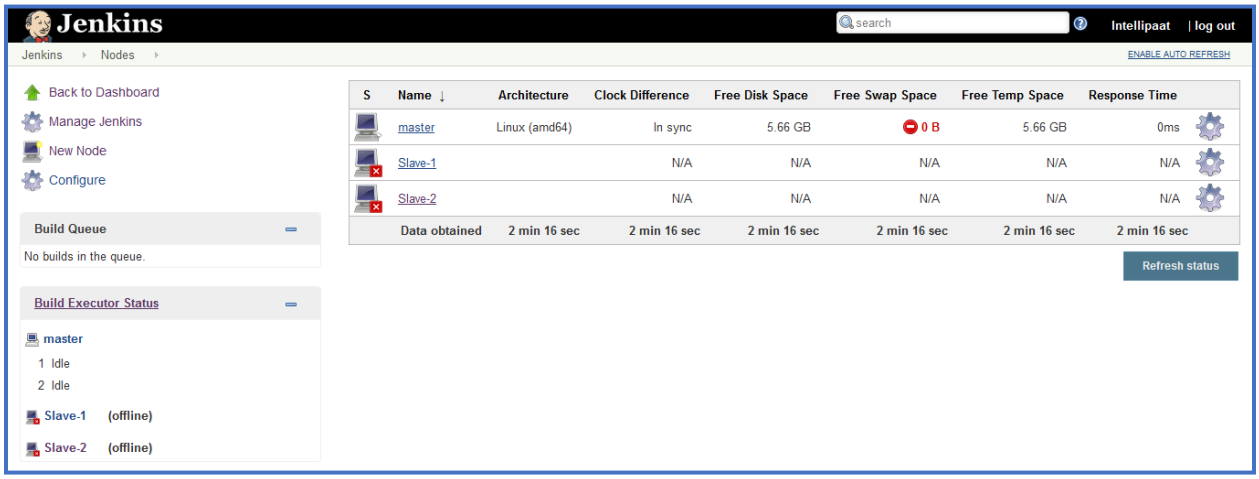
Step 12: Then add the current working directory path to **/home/ubuntu/jenkins.** Then click on **Save.**



Step 13: Make another node **Slave-2** and copy from **Slave-1 as** shown below:

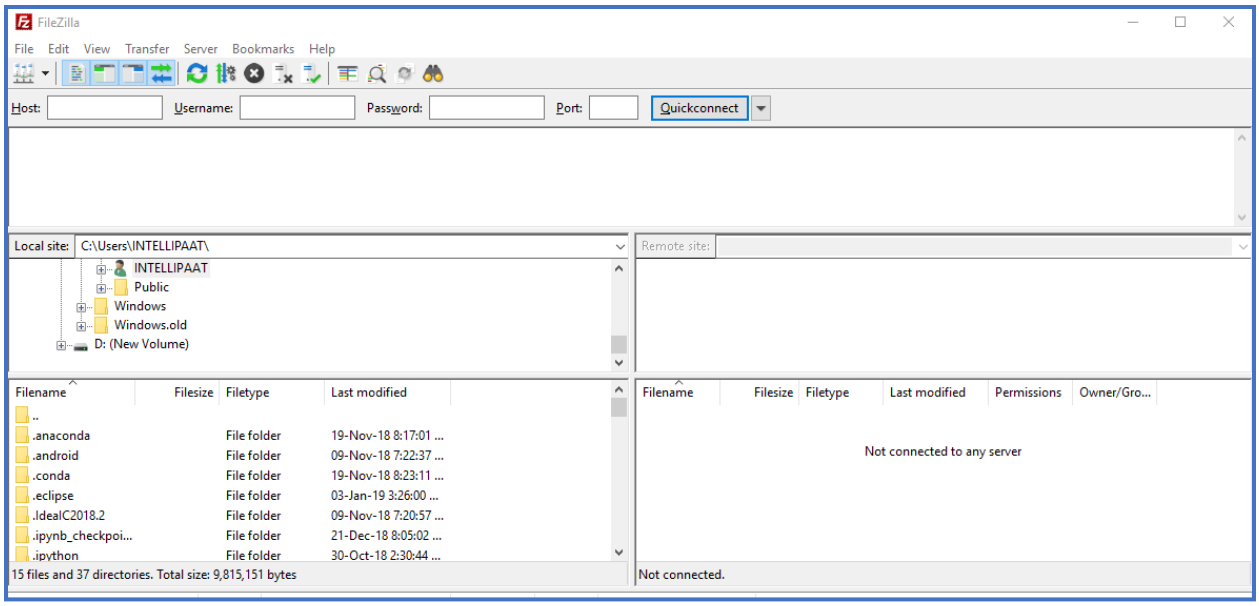


Step 14: Then click ok. You can see the list of nodes that we have on the Jenkins Dashboard.

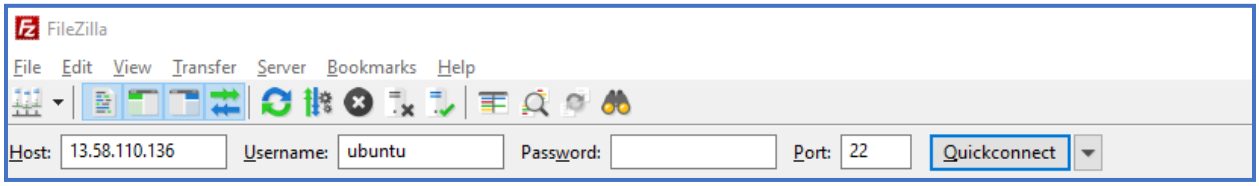


Step 15: Before moving ahead, download **FileZilla**.

Step 16: Once you install **FileZilla** the home page looks like this:

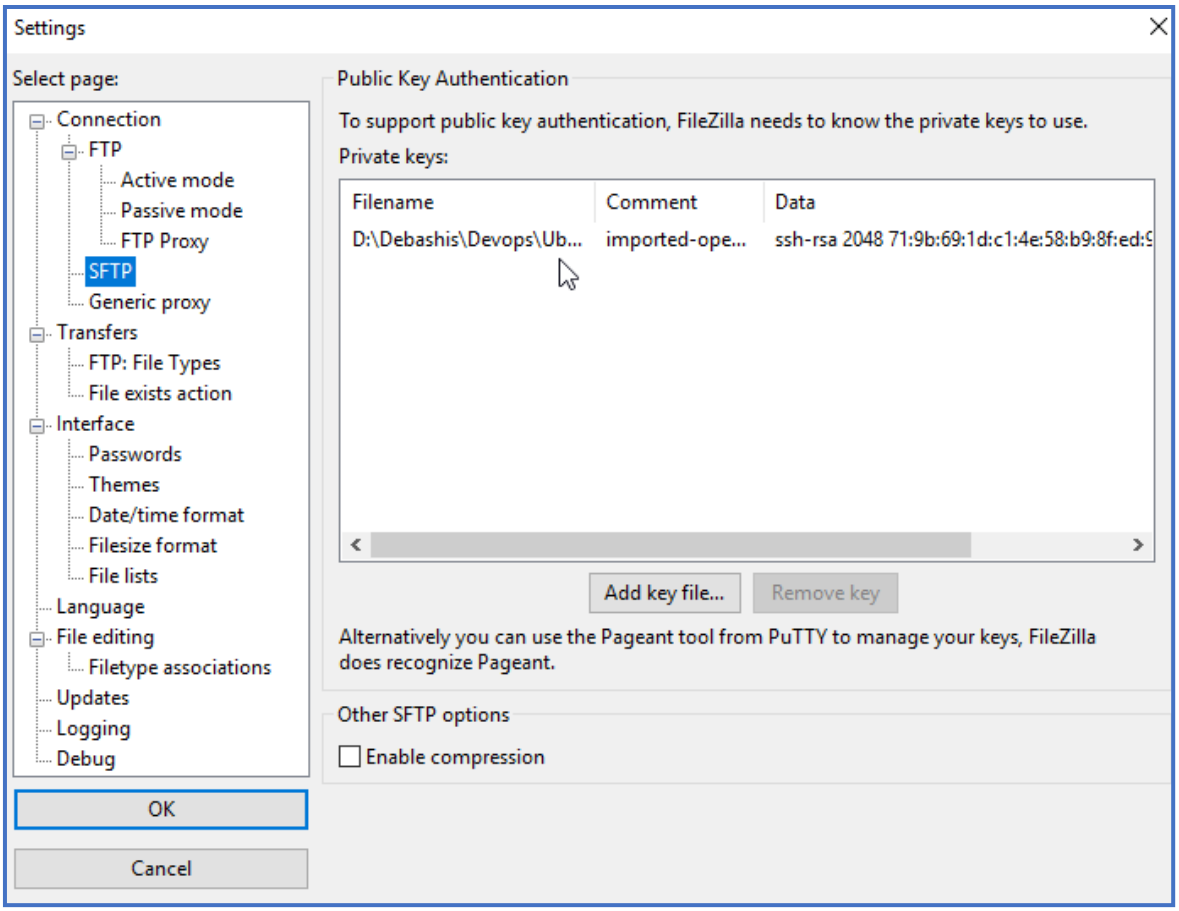


Step 17: Now copy the **Slave-1 IP address.** And add it as **Host**. Add **ubuntu as username**. Leave the password field empty. Add **Port as 22.**

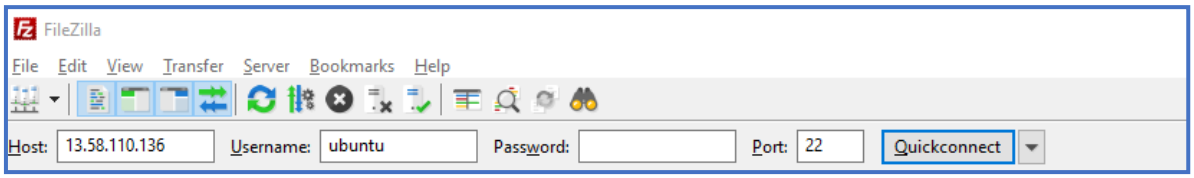


Don't start the connection yet.

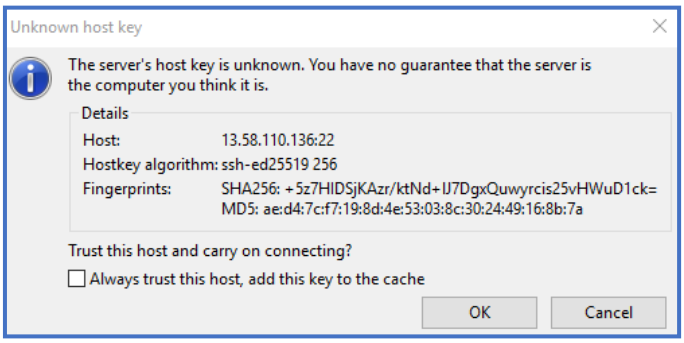
Step 18: Before we start the connection, we need to add the Private key (PPK file). Go to **Edit**, click on **Settings**. Click on **SFTP**. Add the PPK file there and click **ok.**



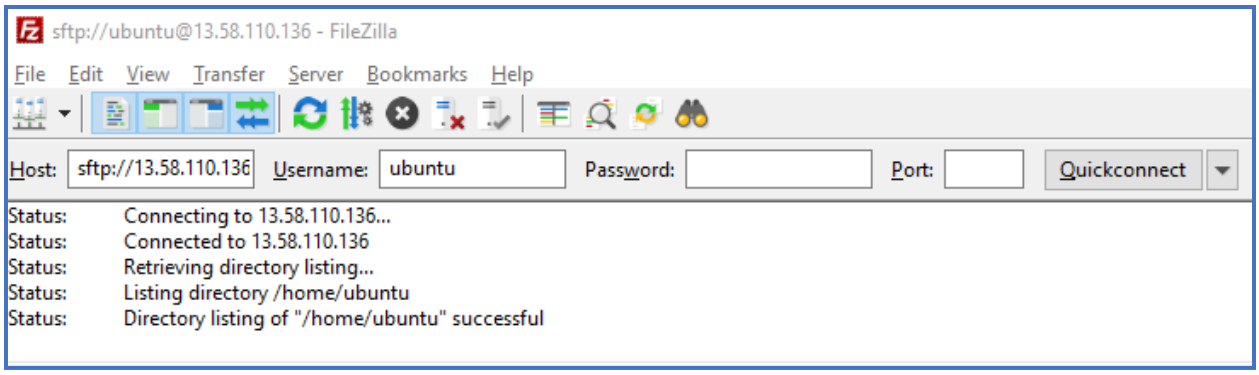
Step 19: Click on **Quickconnect**.



Then click on **ok.**

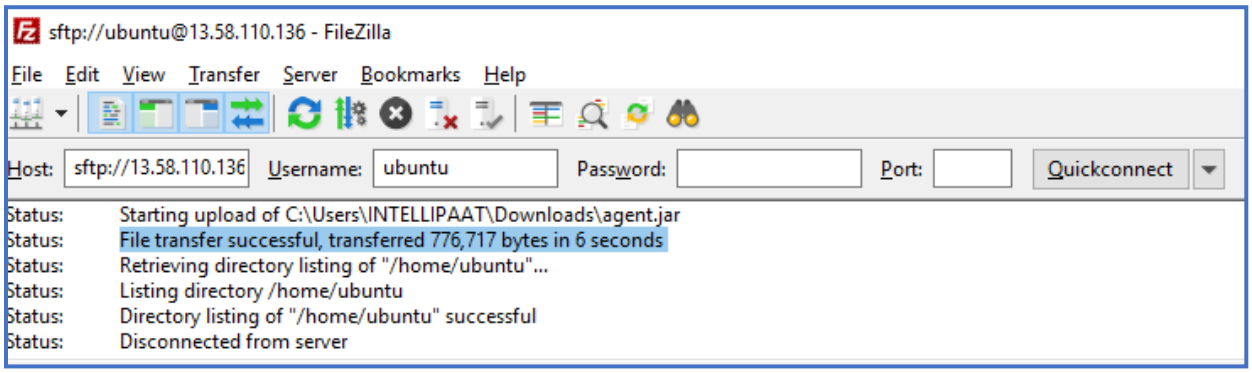


As you can see our connection is successful.

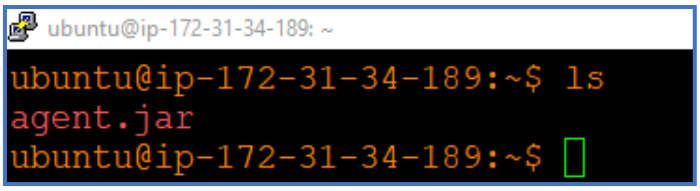


Step 20: Go to the Jenkins Dashboard, Click on **Slave-1**. Download the **Agent.jar** file by clicking on it.

Step 21: Now **drag and drop** the **agent.jar** file on the ubuntu folder in FileZilla.

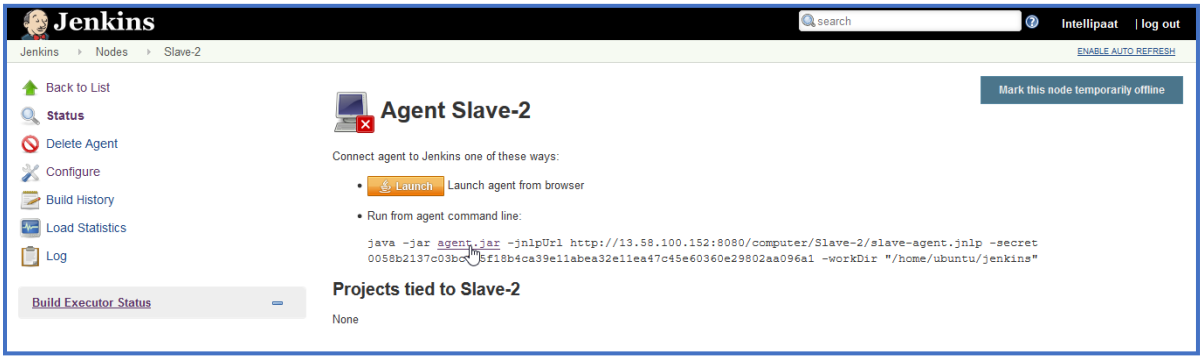


Step 22: Let us verify if the file has been transferred to **Slave-1** or not. Open a new session on putty. Connect to slave-1. Run **ls command.**

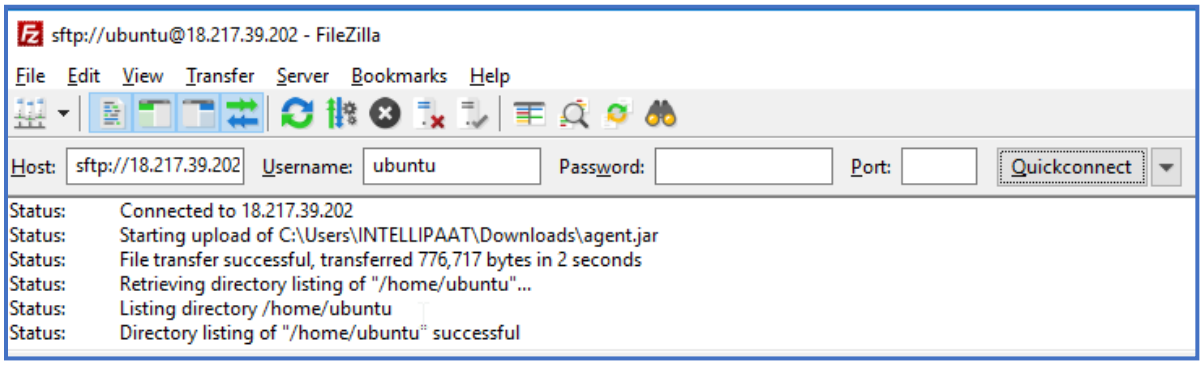


As you can see the agent.jar file appears there, which means our file has been successfully transferred to **Slave-1.**

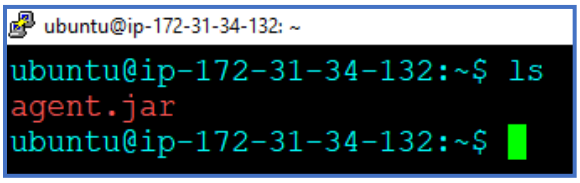
Step 23: Perform the steps 19 to 22 for Slave-2 as well. (Tip: Rename the agent.jar file of **Slave-2.** Before performing transfer operation in FileZilla)



File transferring is successful for Slave-2 agent.jar file as well.



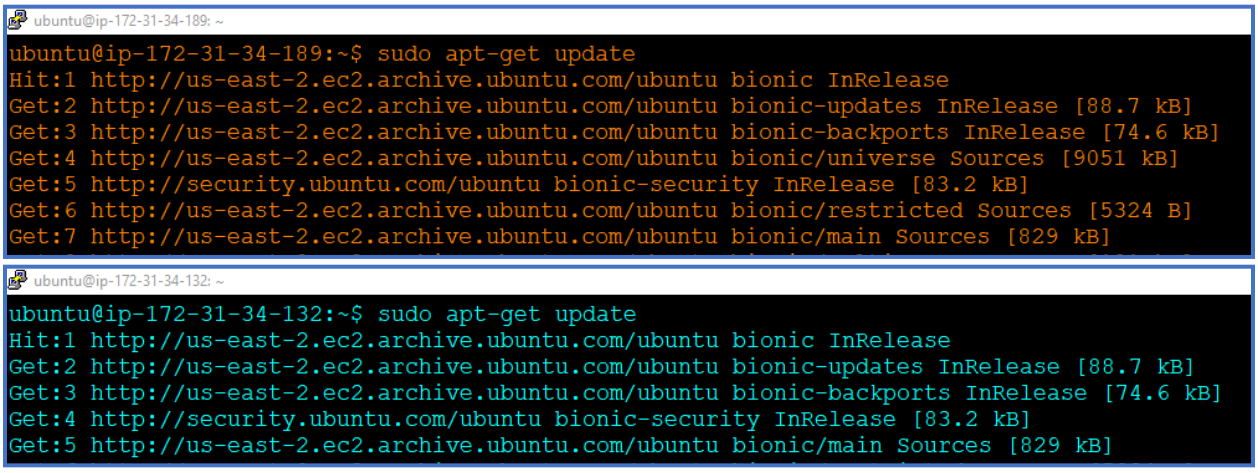
Step 24: Again, verify by opening a new putty session for Slave-2.



Looks file!

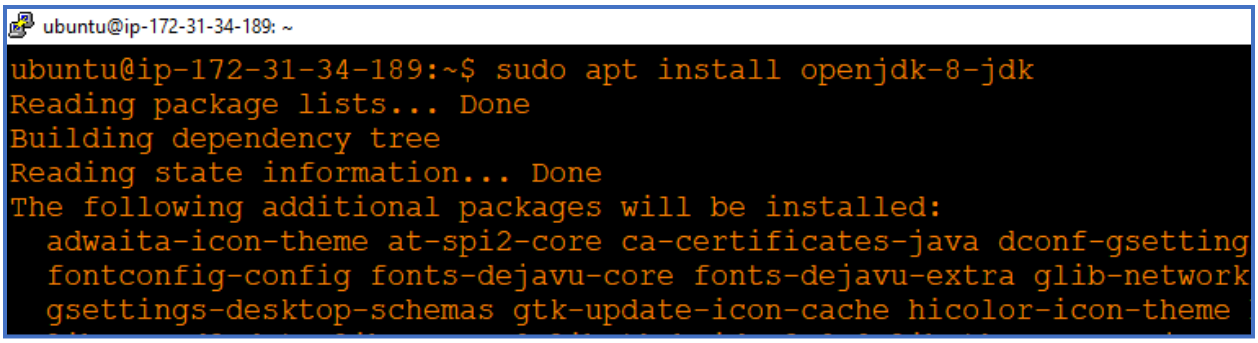
Step 25: Now before moving ahead **install open jdk on both Slave-1 and Slave-2**.

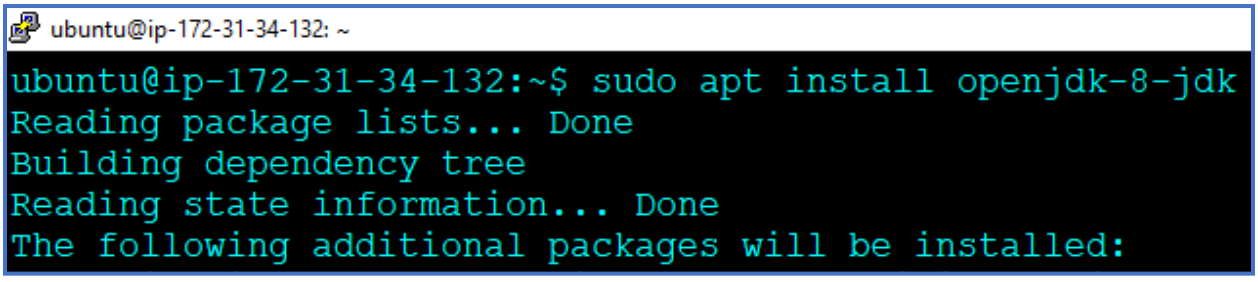
```
$ sudo apt-get update
```



Step 26: Now install run the following installation command on both terminal.

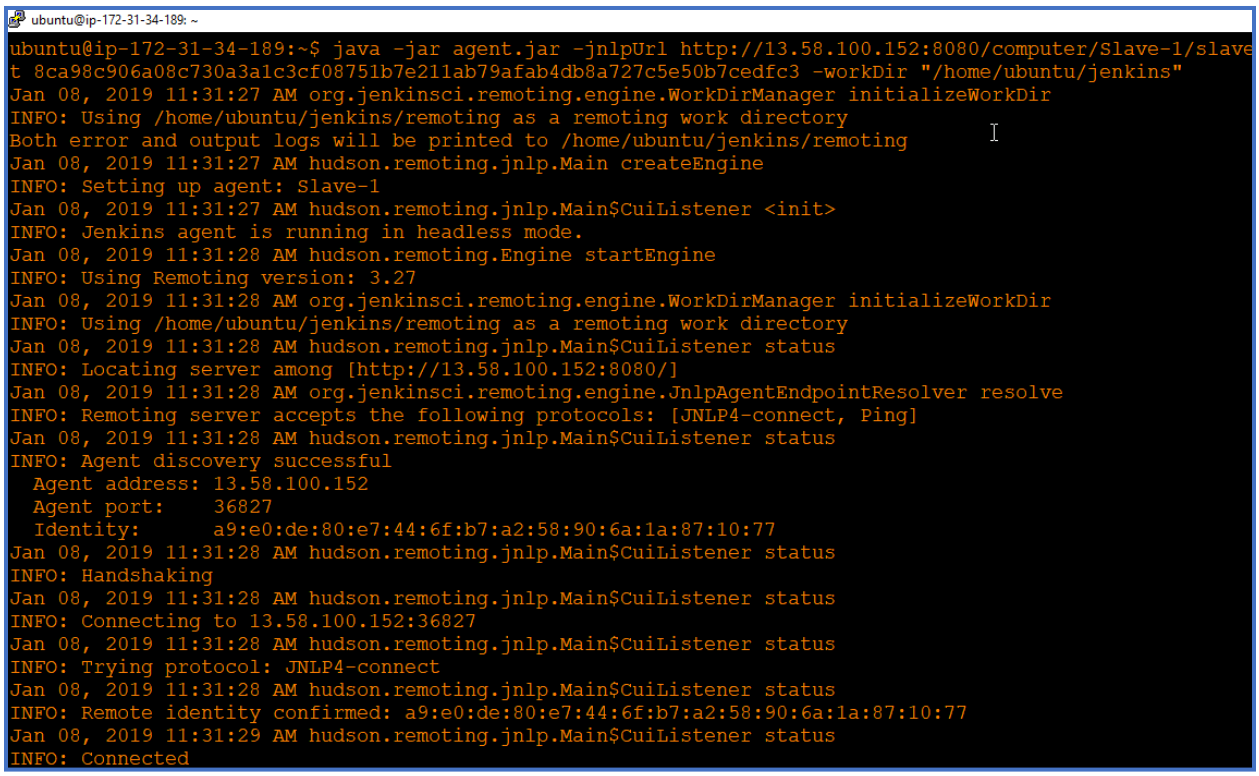
```
$ sudo apt install open-9-jdk
```





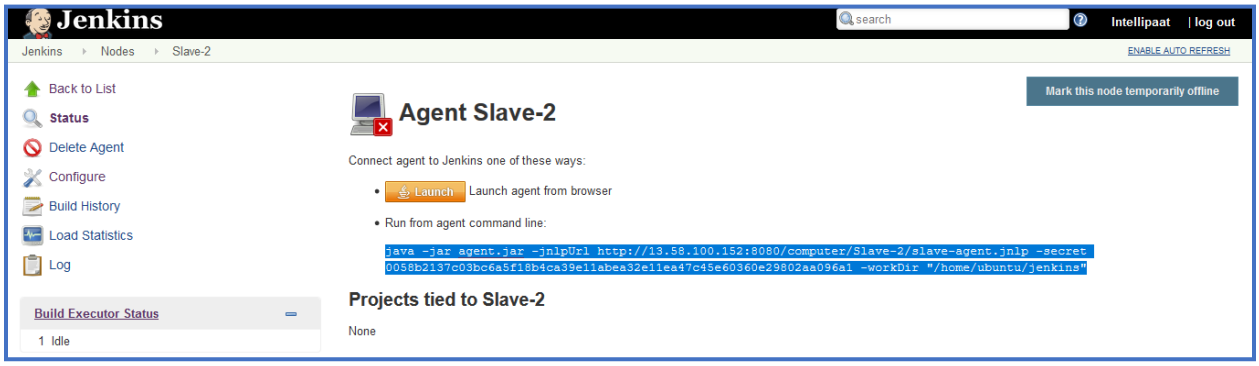
Step 27: Now we will connect Slave-1 and Slave-2 to the AWS Jenkins Server. Go to the Jenkins Dashboard, Click on Slave-1, **Copy the command line** as shown.

Run the command line from Slave-1 as shown below.

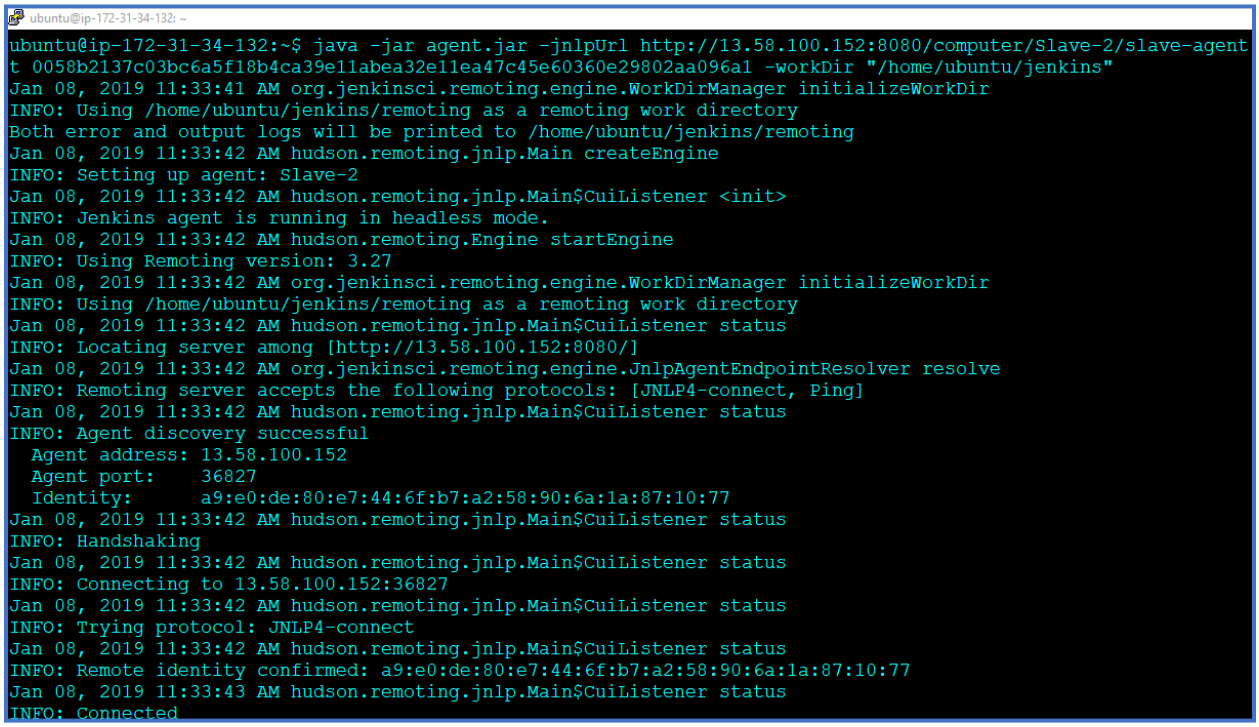


It shows "**Connected**".

Step 28: Perform the **Step-27** for **Slave-2** as well.

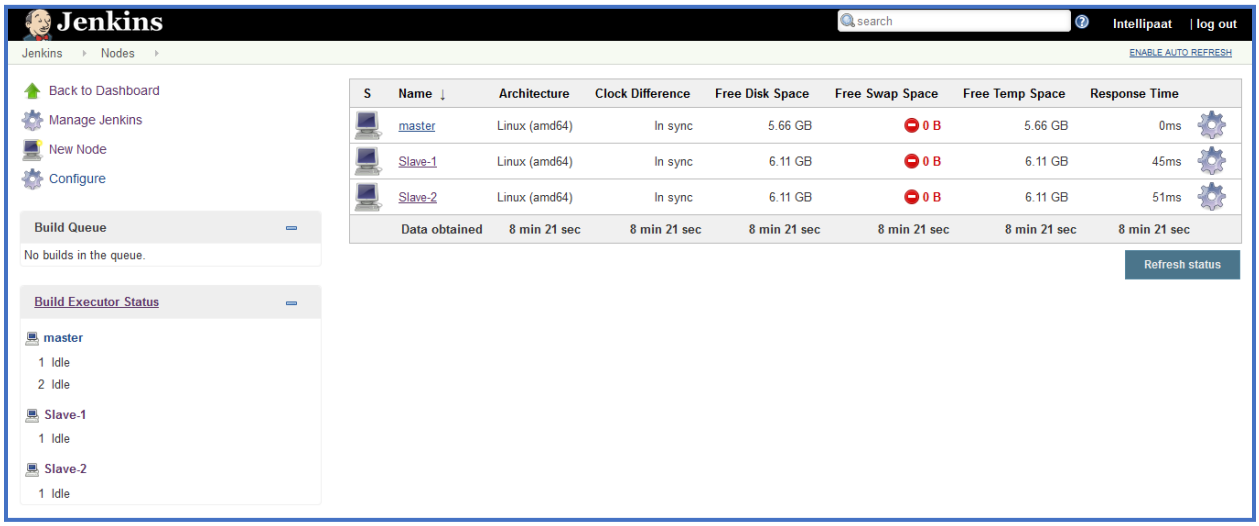


Paste the command line in the Slave-2 Terminal.



Important Note: *Don't end the Sessions that we just Connected. To perform further operations on Slave-1 and Slave-2 duplicate the sessions.*

So now that our Slave-1 and Slave-2 has been connected to Jenkins Server, it looks like this.



After we have successfully created the Master Slave Cluster on AWS Jenkins. We will now create a CI CD pipeline triggered by Git Webhook.

Hands-on: Create a CI CD pipeline triggered by Git Webhook.

Step 1: Before that open your GitHub account and import the below given repository.

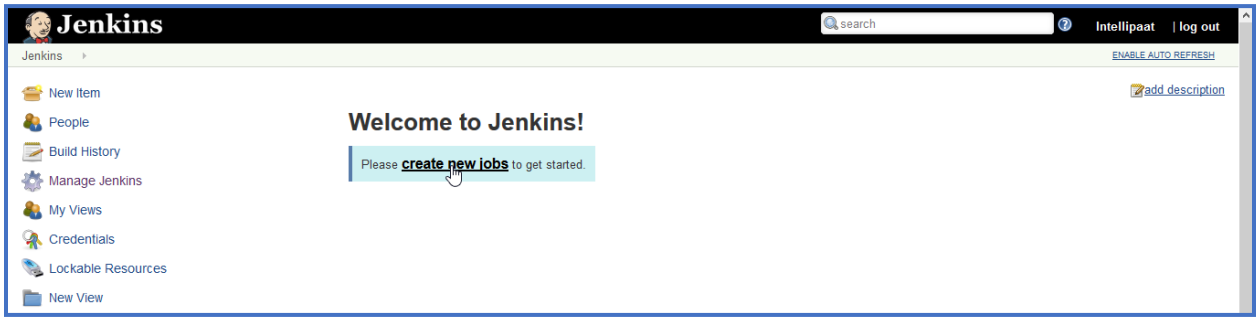
https://github.com/hshar/devopsIQ.git

Step 2: Install docker on both **Slave-1** and **Slave-2**.

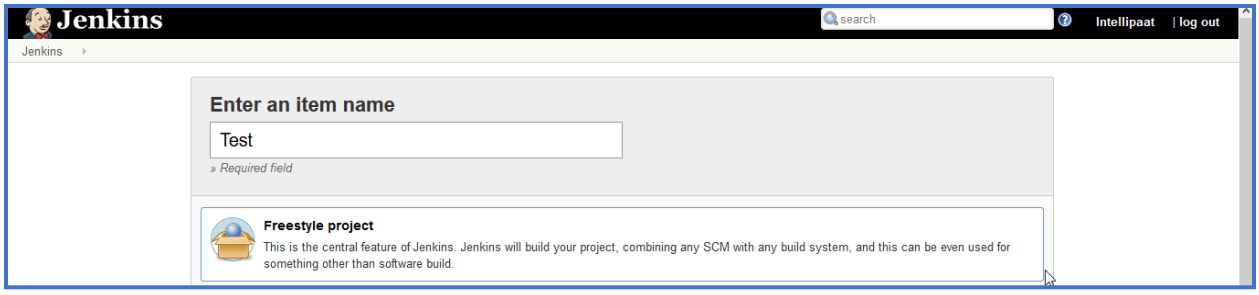
```
ubuntu@ip-172-31-34-189: ~
ubuntu@ip-172-31-34-189:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount libltdl7 pigz ubuntu-fan
Suggested packages:
  ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount docker.io libltdl7 pigz ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 121 not upgraded.
Need to get 40.3 MB of archives.
After this operation, 198 MB of additional disk space will be used.
```

```
ubuntu@ip-172-31-34-132: ~
ubuntu@ip-172-31-34-132:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount libltdl7 pigz ubuntu-fan
Suggested packages:
  ifupdown aufs-tools debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount docker.io libltdl7 pigz ubuntu-fan
0 upgraded, 6 newly installed, 0 to remove and 121 not upgraded.
Need to get 40.3 MB of archives.
After this operation, 198 MB of additional disk space will be used.
```

Step 3: Open **Jenkins Dashboard**. **Create a new job** (Freestyle Project) for Slave-1.

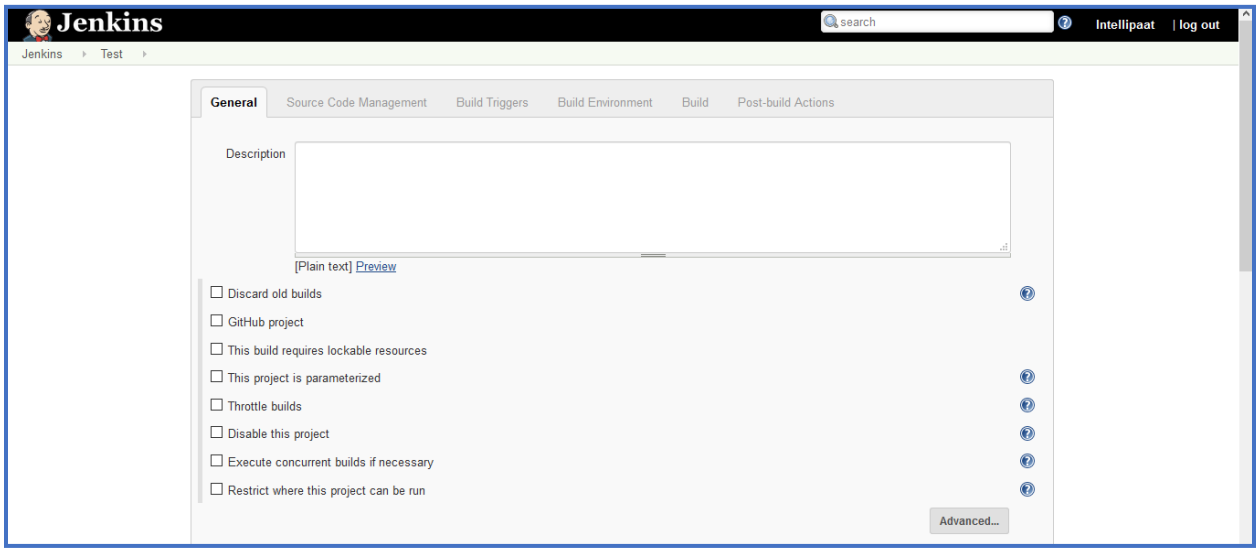


Name the Project as **Test**, Select **Freestyle Project** option.

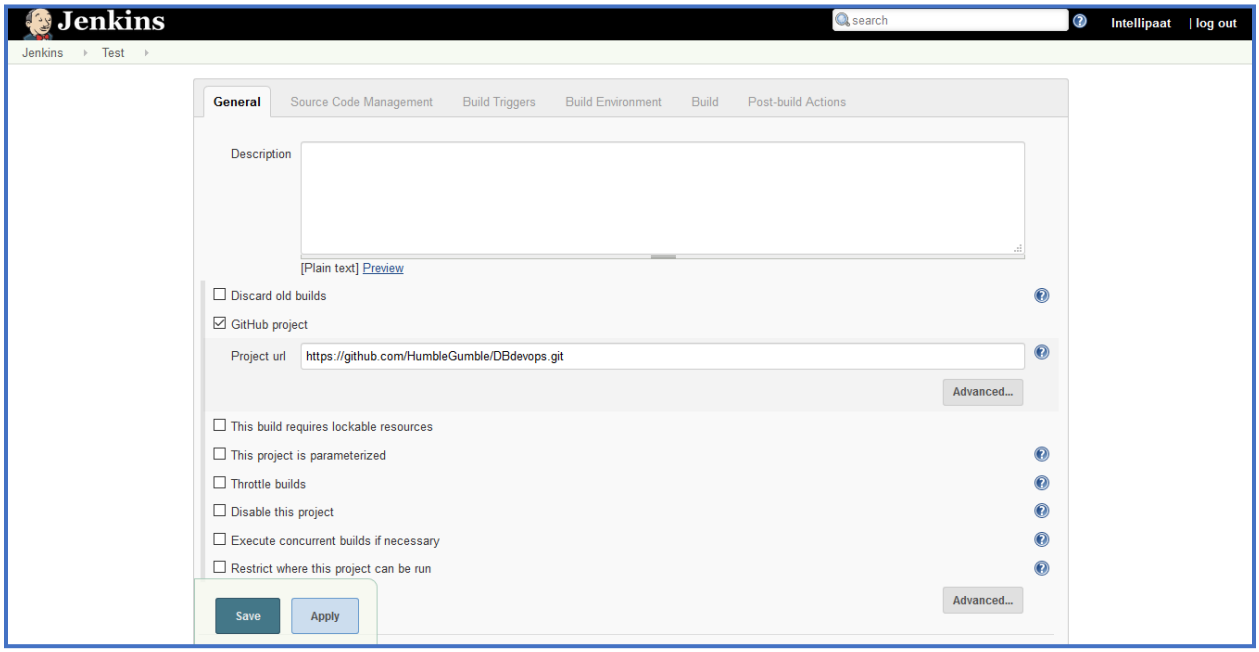


Then click on **Ok**.

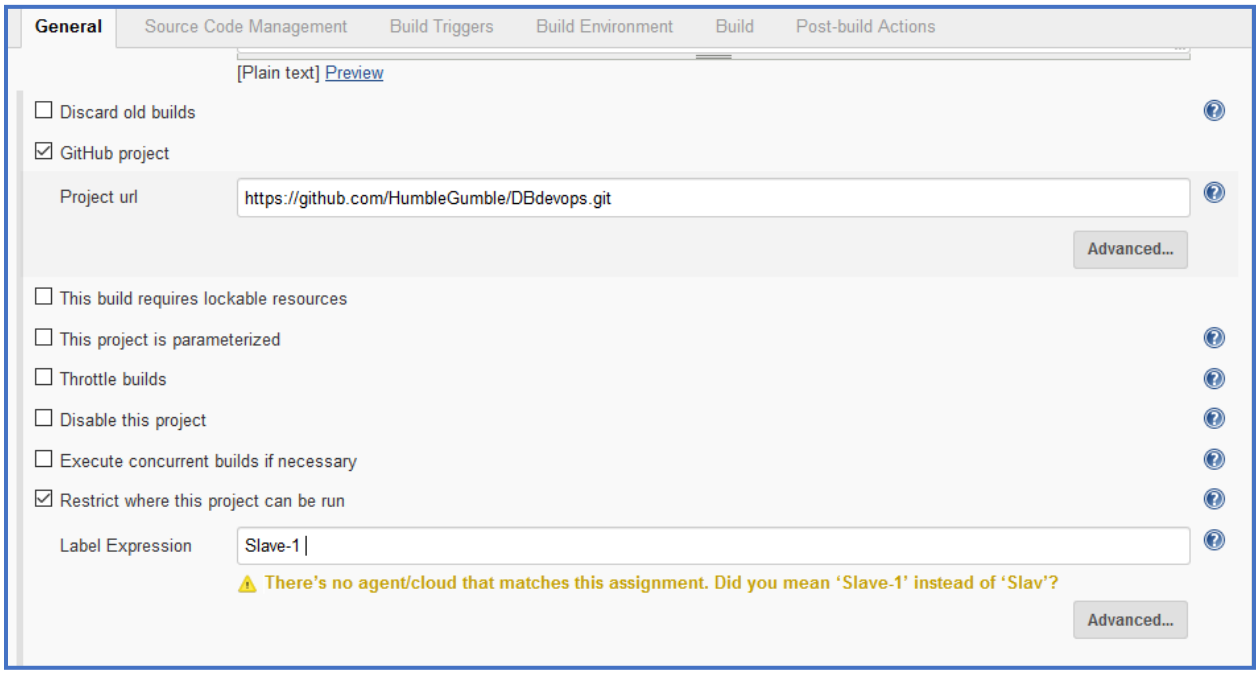
You should land on a page like this.



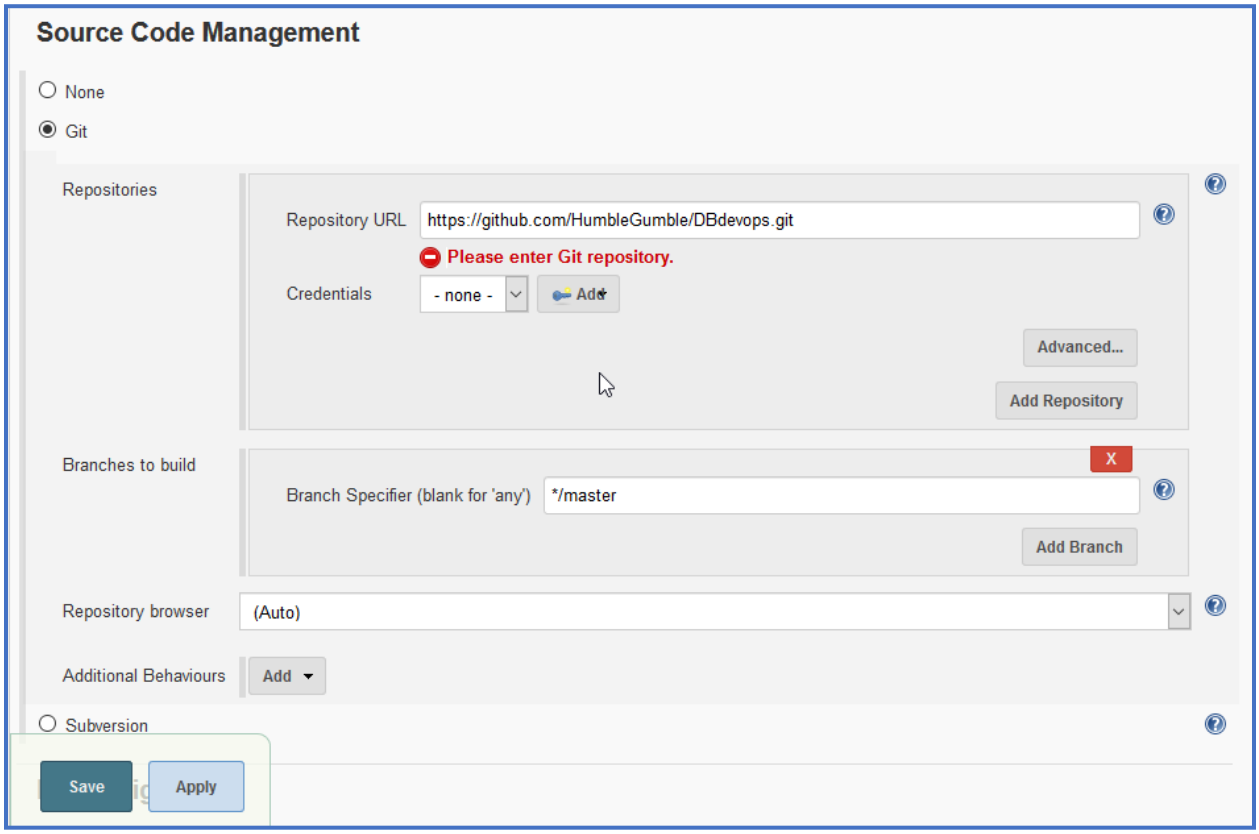
Step 4: Place your git repository link as shown below.



Click on **Restrict where this project can be run**. Add **Slave-1** there.

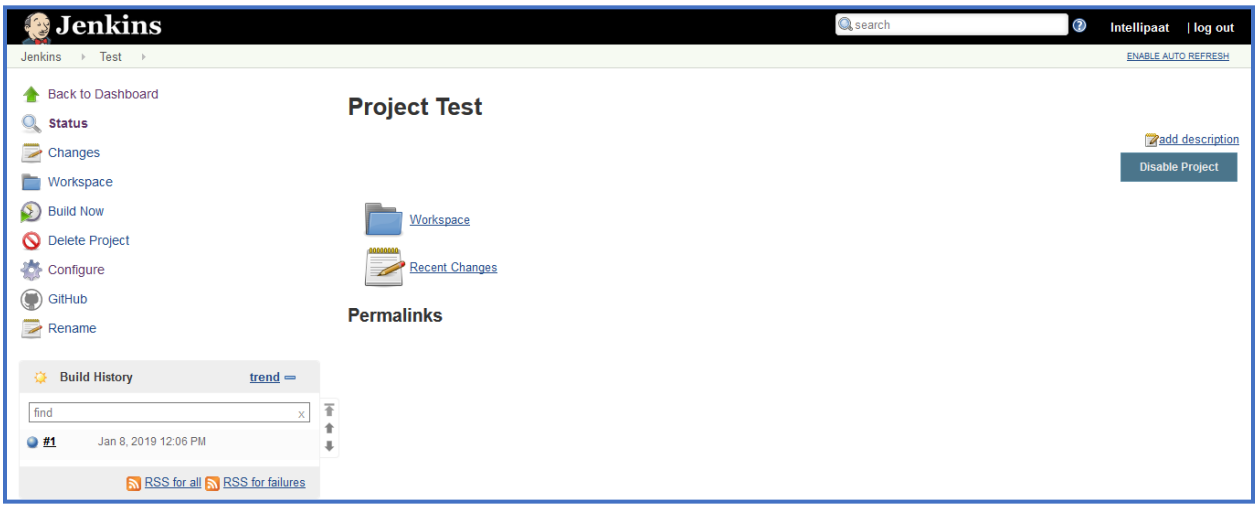


Go to **Source Code Management**, click on **git**, add the **git repository link** there as well.

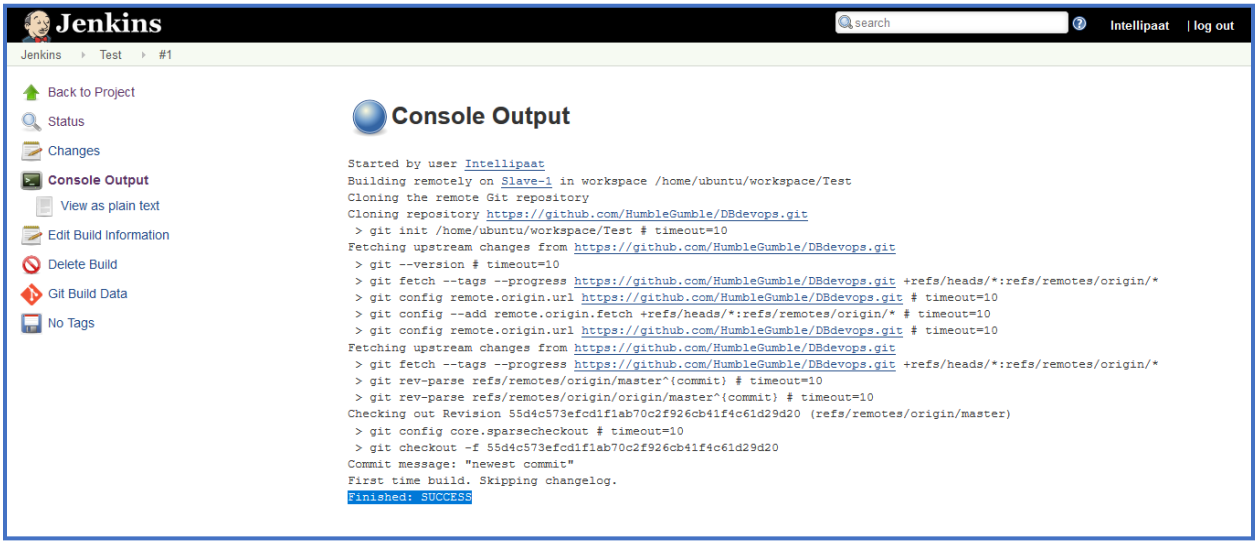


Click on **Save.**

Step 5: Click on **Build Now,** if the building is done without any error there will be **blue circle** in the building history.



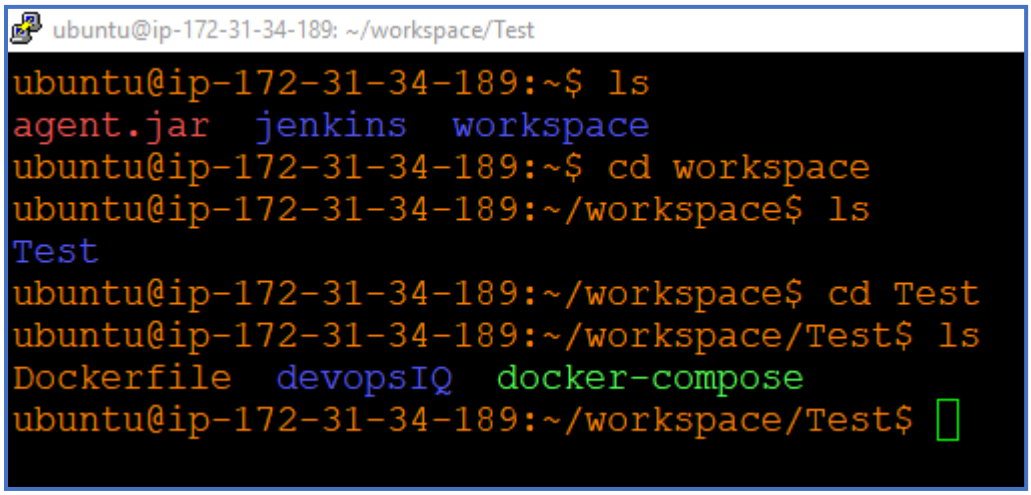
Click on the blue circle of build #1.



You can see it has been built successfully. Let us verify that.

Step 6: Go to slave-1.

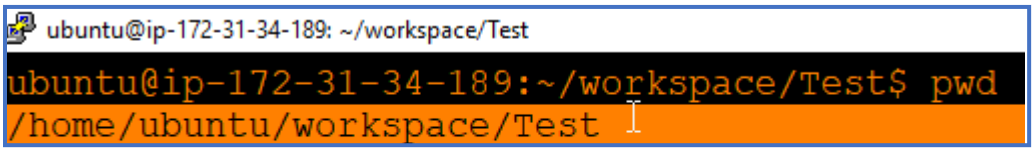
```
$ ls
$ cd workspace
$ ls
$ cd Test
$ ls
```



You can see the repository files there. This means the git repository has been successfully cloned into the Test job.

Now we will deploy the website that we have stored in our repository.

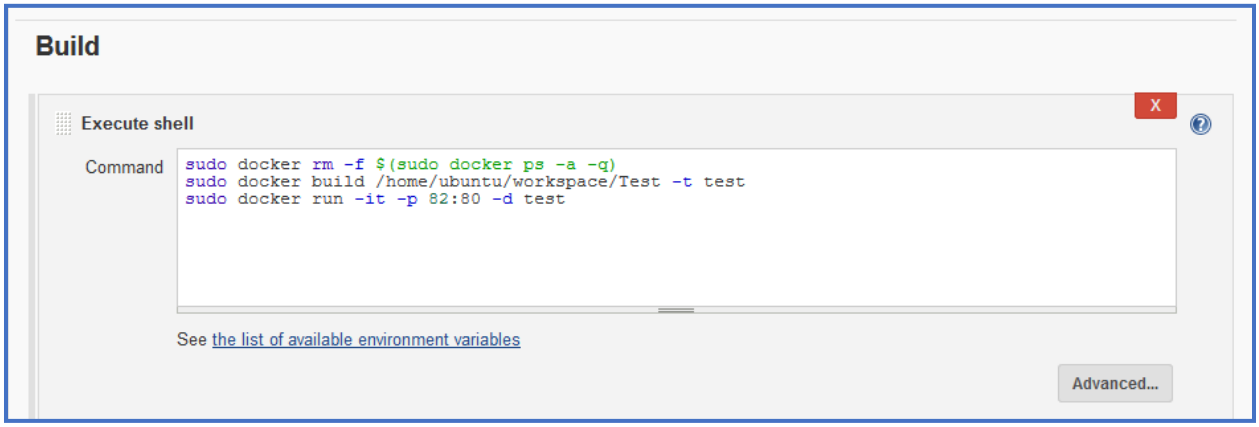
Step 7: To run the **Dockerfile** we have to check the copy the present working directory.



Now go back to configuring the job.

Step 8: Click on **Build**, then go to **Execute shell**

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/workspace/Test -t test
sudo docker run -it -p 82:80 -d test
```



Click on save.

Before building our job again we must add one arbitrary container in slave-1.

Step 9: Add container by performing the following command.

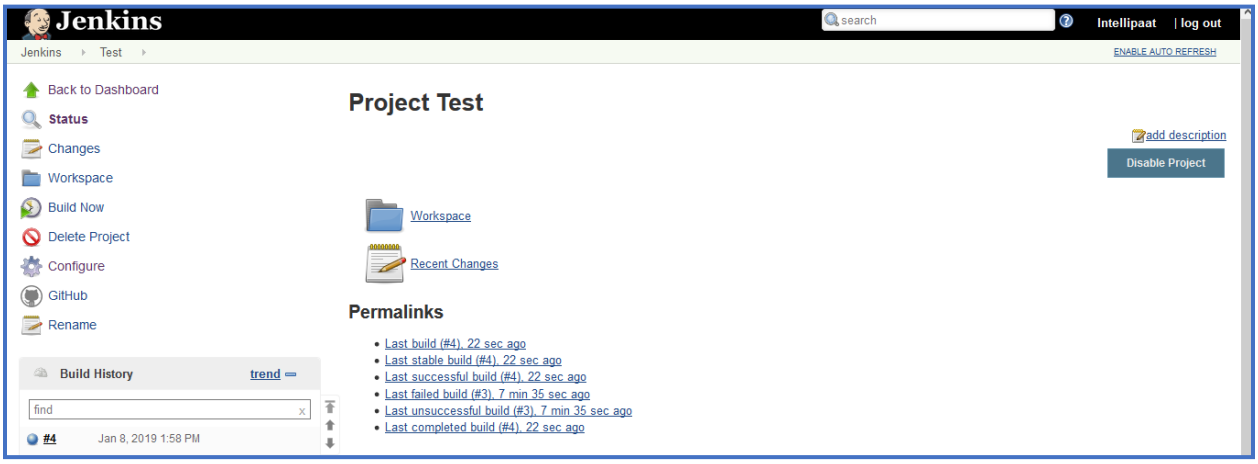
```
$ sudo docker run -it -d ubuntu
```

```
ubuntu@ip-172-31-34-189: ~/workspace/Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ sudo docker run -it -d ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
84ed7d2f608f: Pull complete
be2bf1c4a48d: Pull complete
a5bdc6303093: Pull complete
e9055237d68d: Pull complete
Digest: sha256:868fd30a0e47b8d8ac485df174795b5e2fe8a6c8f056cc707b232d65b8a1ab68
Status: Downloaded newer image for ubuntu:latest
ebe701788bff916b06db2a9bf7ad34b4c7cf3e722e8e789c1ea6deaf5ee2beaf
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

Now we have added in a container.

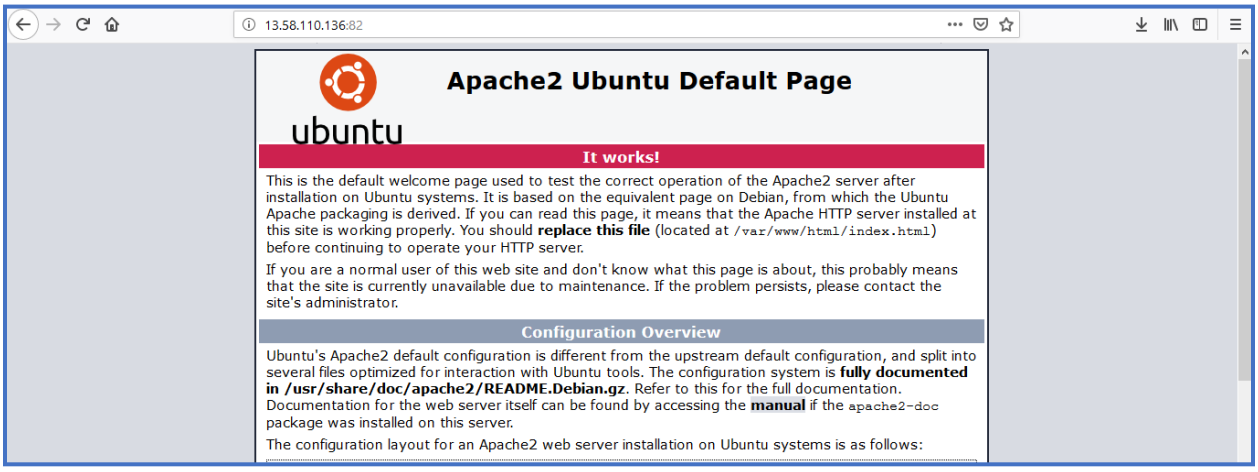
```
ubuntu@ip-172-31-34-189: ~/workspace/Test
ubuntu@ip-172-31-34-189:~/workspace/Test$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
AMES
ebe701788bff        ubuntu              "/bin/bash"         About a minute ago   Up About a minute
elaxed_varahamihira
ubuntu@ip-172-31-34-189:~/workspace/Test$
```

Step 10: Now open Jenkins Dashboard and **build the project**.



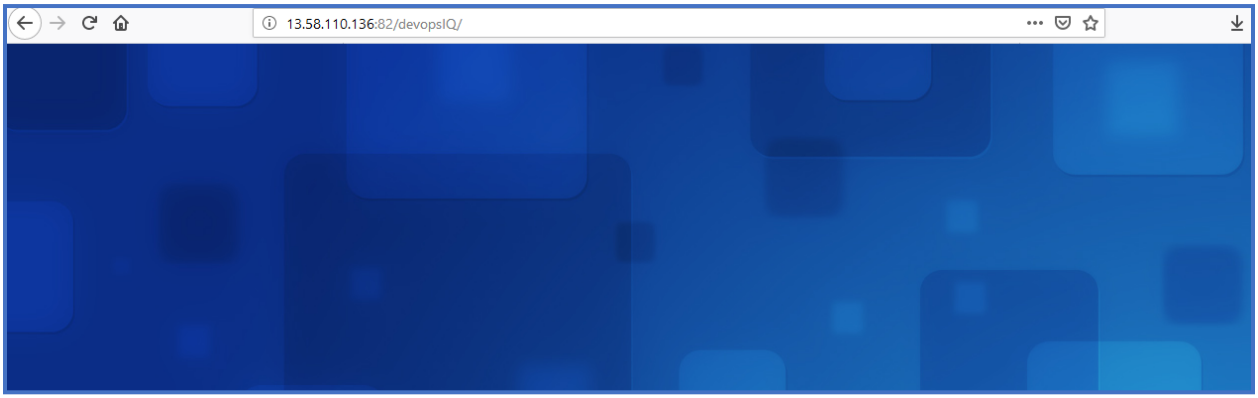
Building was successful.

Step 11: Now open browser and enter **Slave-1 IP:82**



This is the apache page that means our container is working perfectly.

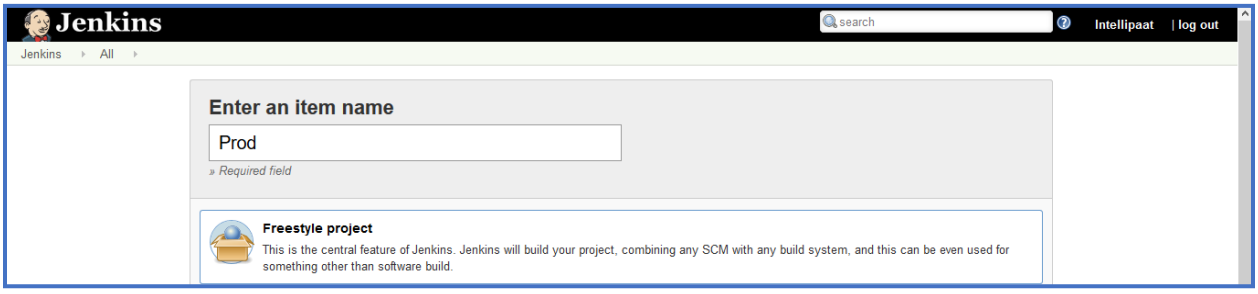
Step 12: Now enter **slave-1 IP:82/devopsIQ/** in the browser.



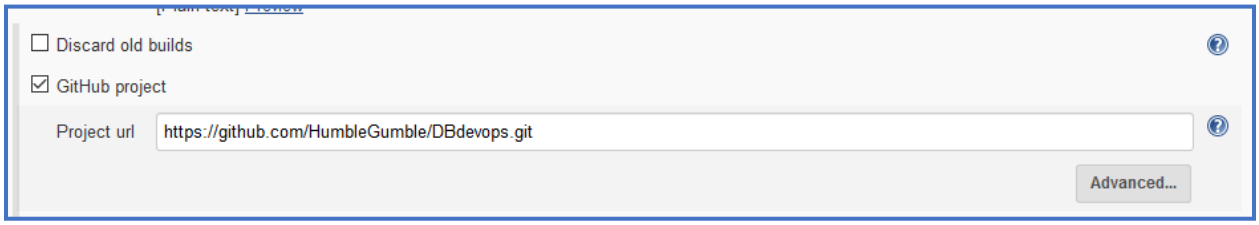
This looks fine.

Now, we will create a new project.

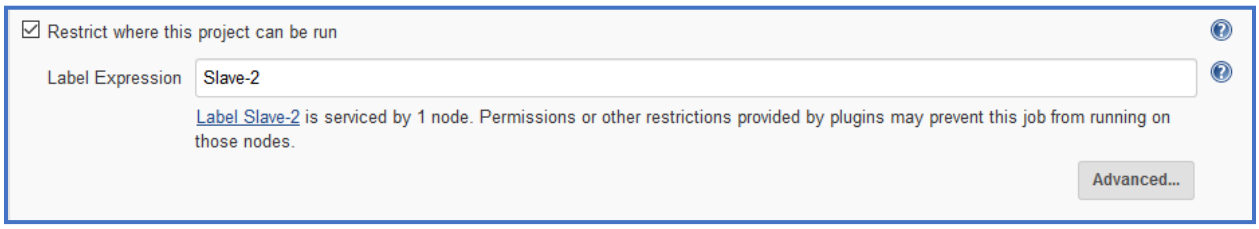
Step 13: Create a new project.



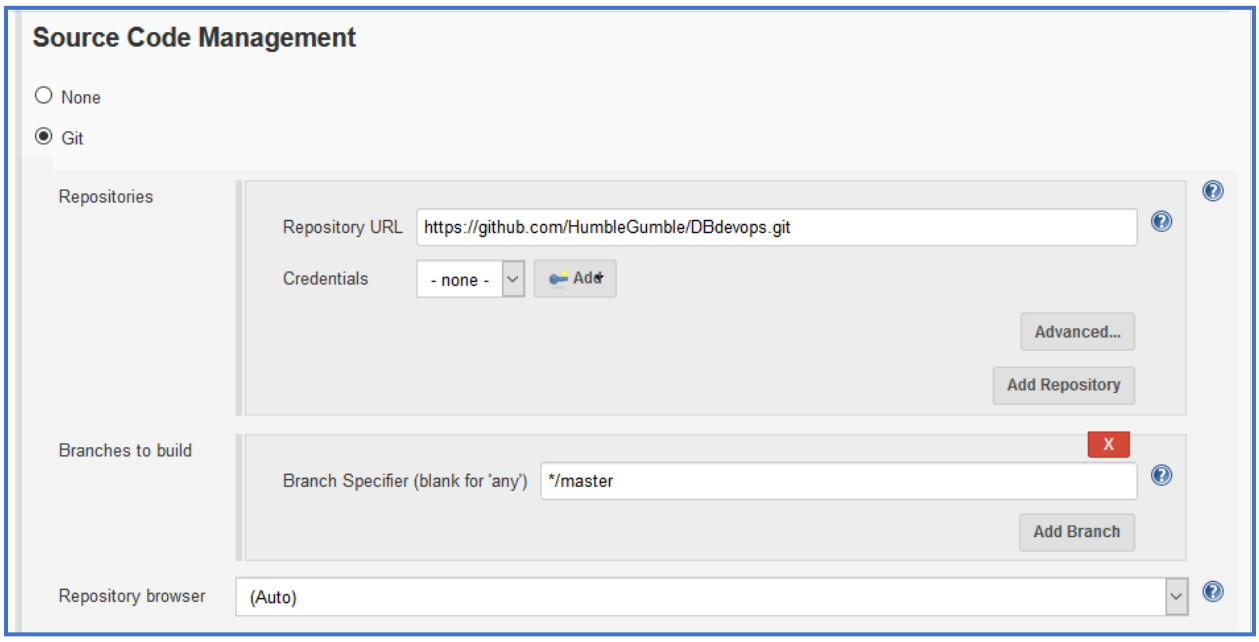
Step 14: Click on git project. Enter the git hub repository URL.



Step 15: Click on **Restrict where this project can be run** enter **Slave-2.**



Step 16: Go to Source code management enter the git repository URL there as well.

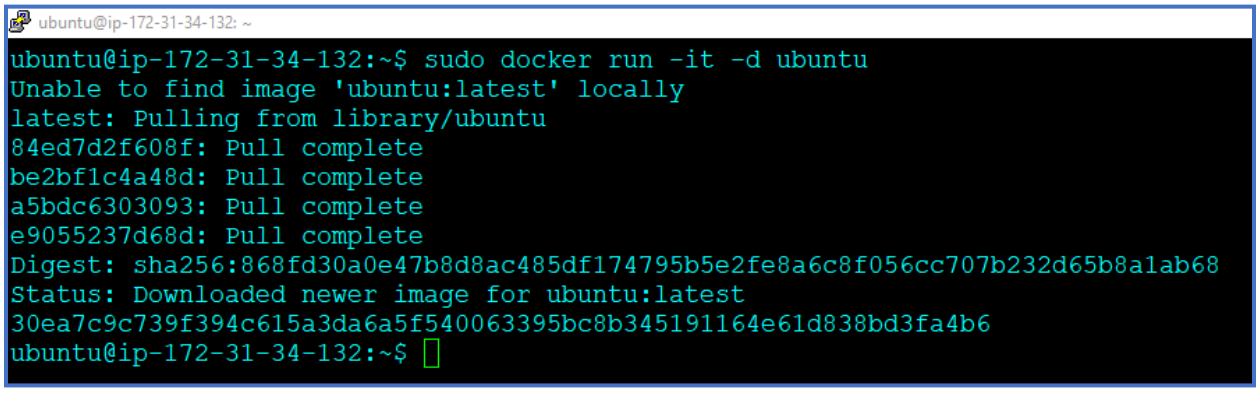


Step 17: Now enter the following command in the **Execution shell**

```
sudo docker rm -f $(sudo docker ps -a -q)
sudo docker build /home/ubuntu/workspace/Prod -t production
sudo docker run -it -p 82:80 -d production
```

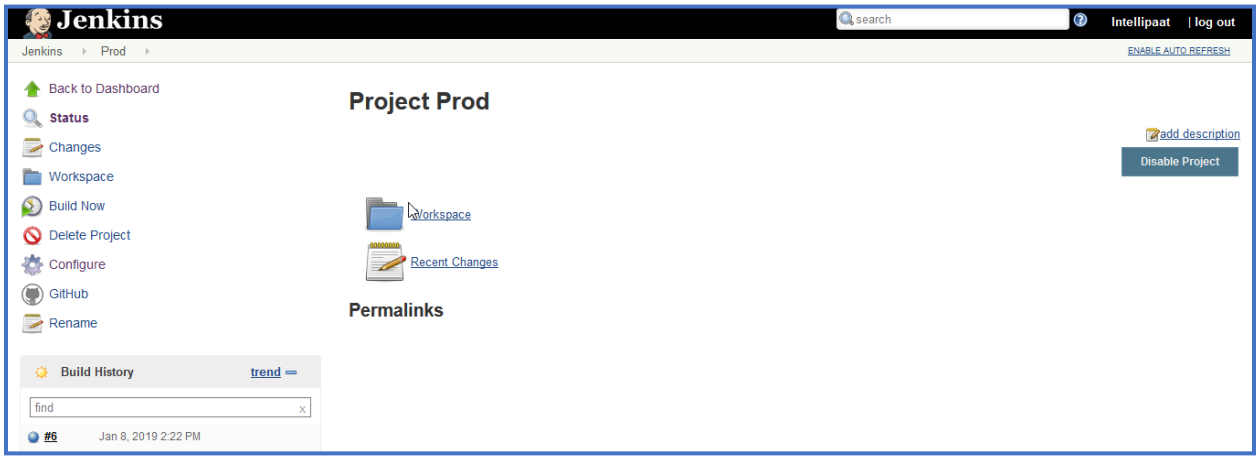
Step 18: Again, add one container to the Slave-2 as shown below.

```
$ sudo docker run -it -d ubuntu
```



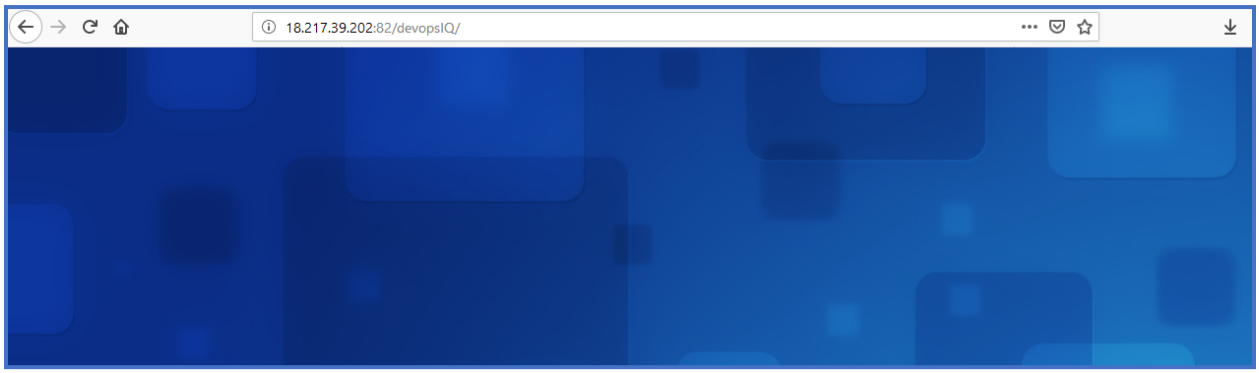
Now that we have added an arbitrary container, go to Jenkins Dashboard and build the project.

Step 19: Build the project **Prod**.



Our Project building was successful.

Step 20: Now go to the browser and enter **Slave-2 IP:82/devopsIQ/**



It's working!

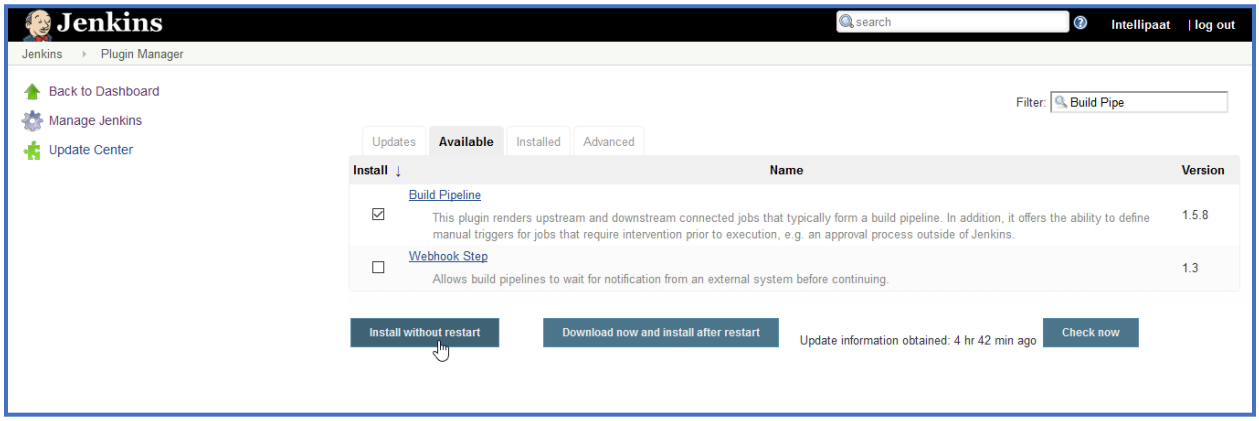
Now we will be triggered **Prod** job only when **Test** job will be completed.

Step 21: Go to the **Test** job, click on **Configure**. Add **Post-Build Actions**. Then go to **Build Other Projects.**

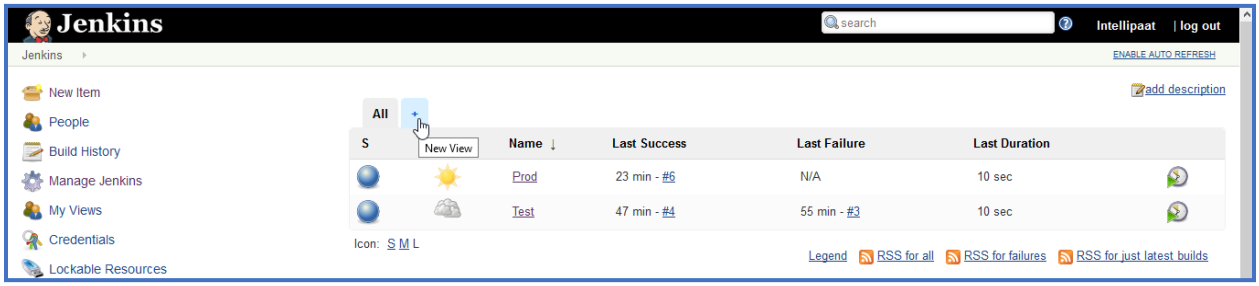
Click on Save.

Now we will run jobs using pipeline.

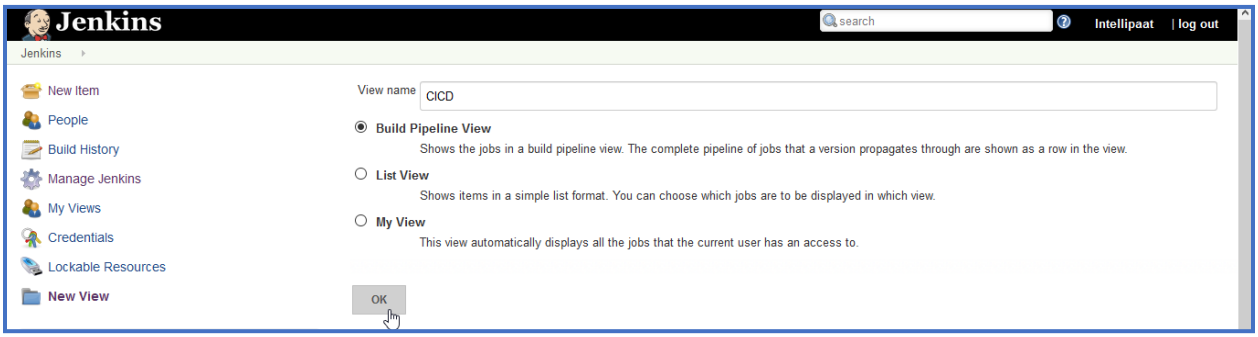
Step 22: Go to the manage jenkins, click on available, search for Build Pipeline. Click on install without restart.



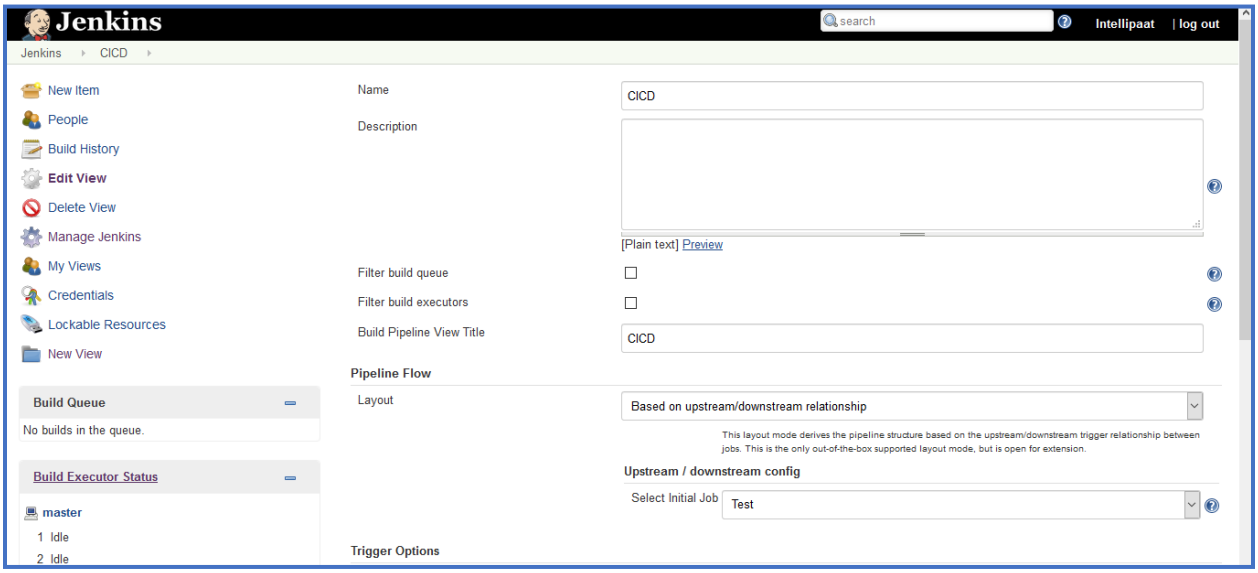
Step 23: Go to the jenkins dashboard. Click on the +.



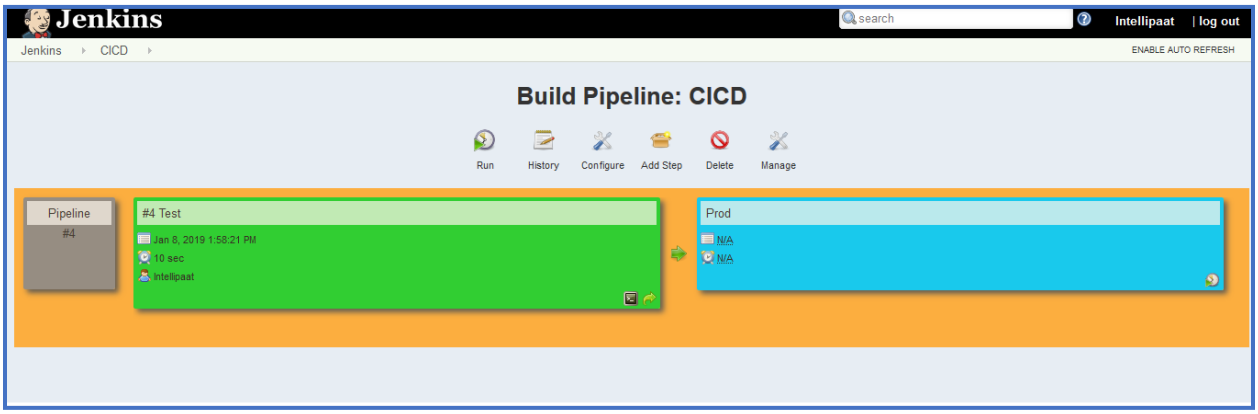
Step 24: Enter view name and click ok.



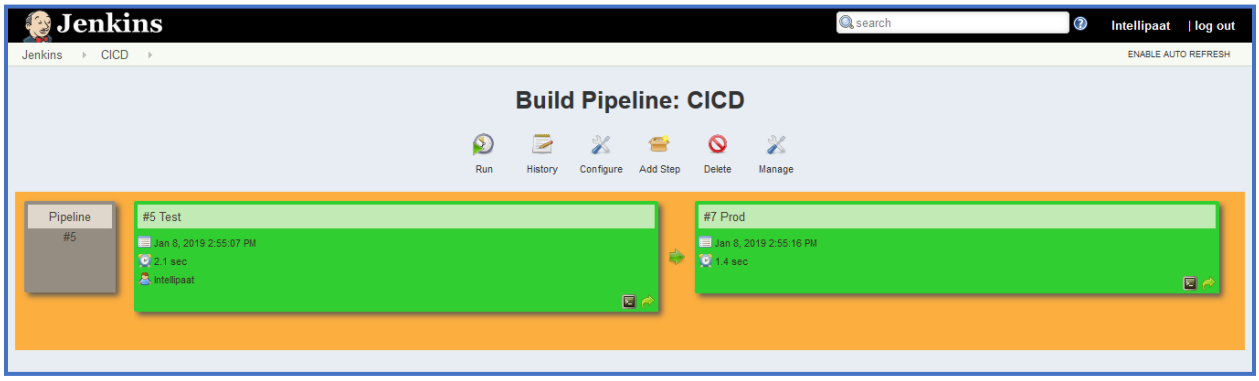
Step 25: There add the Build Pipeline View Title, then Select initial job as Test.



Click on ok. You should see the Pipeline Page like this.

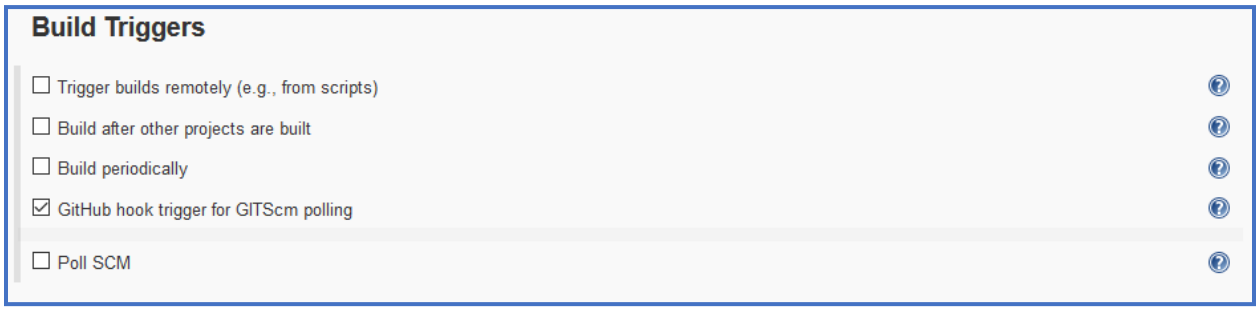


Step 26: Click on Run. Then Refresh the Page once.



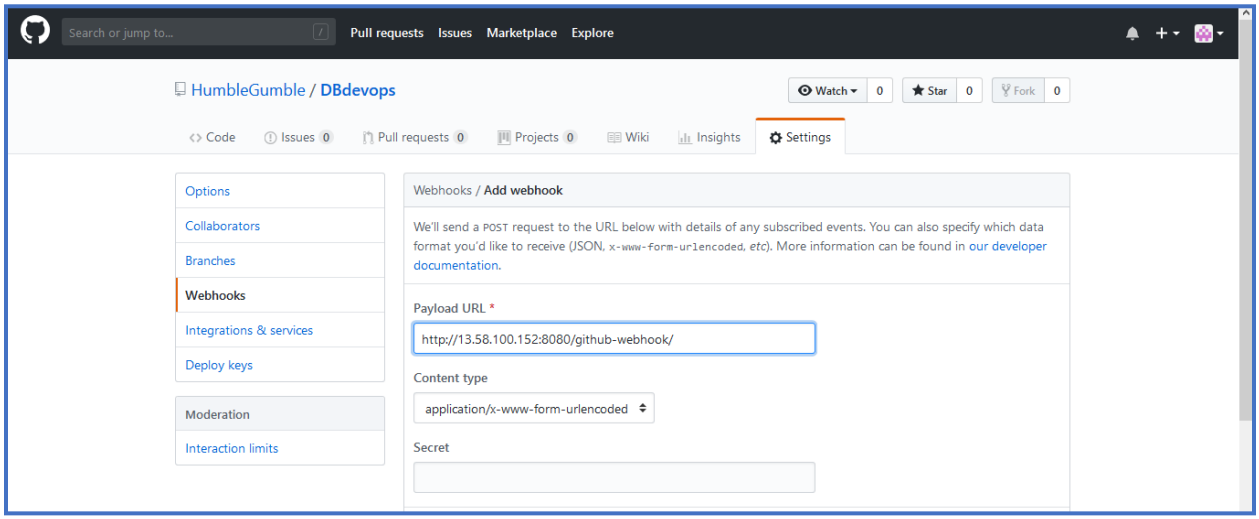
Now we will commit on GitHub, which should trigger our Jenkins Job.

Step 27: Go to the Jenkins Dashboard. Click on Test and then Configure. Check the ***GitHub hook trigger for GITScm polling*** option.

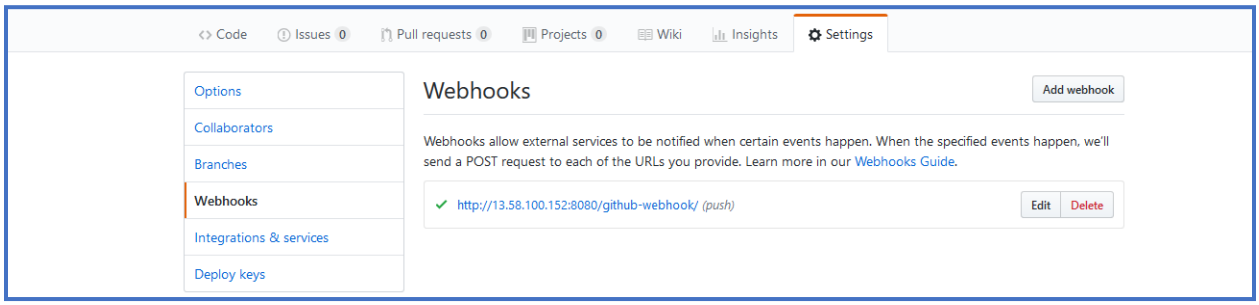


Step 28: Now configure GitHub Webhook. Go to settings, then click on Webhooks, then add webhooks. There insert the Jenkins Server Address as shown.

```
$ JenkinsServer Address/github-webhook/
```

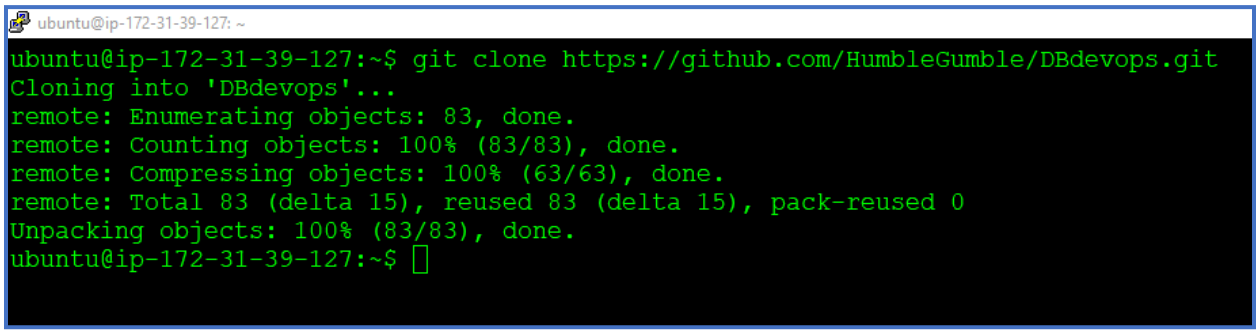


Click on Add webhook. You should see this.



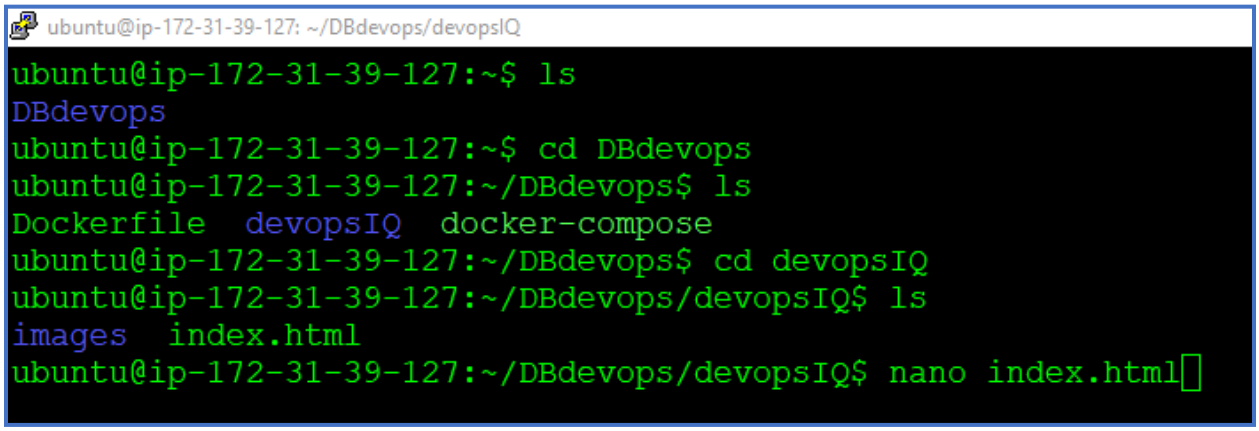
Step 29: Go to the mater terminal to trigger a built.

```
$ git clone <git repository URL>
```

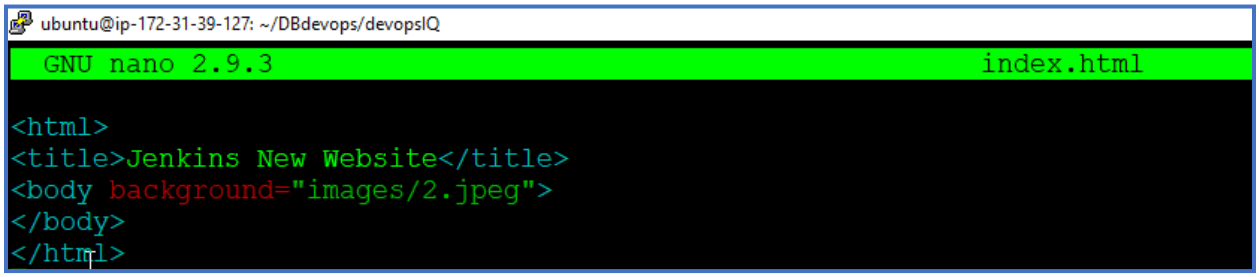


Step 30: Now we will try to modify the website from the master terminal. Go to the master terminal and then go to the devopsIQ directory where you can find index.html file. Open it for modification

```
$ nano index.html
```



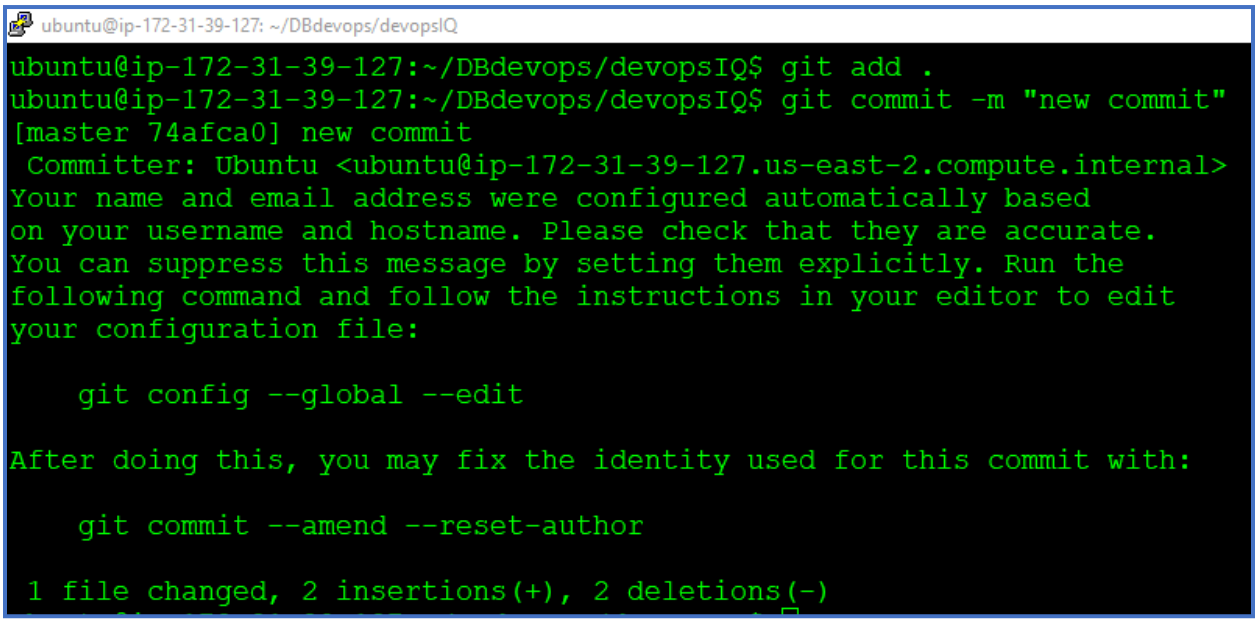
Step 31: Make the modification in the **title** and **body** of that html file as shown below.



Step 32: Finally, perform git add and git commit.

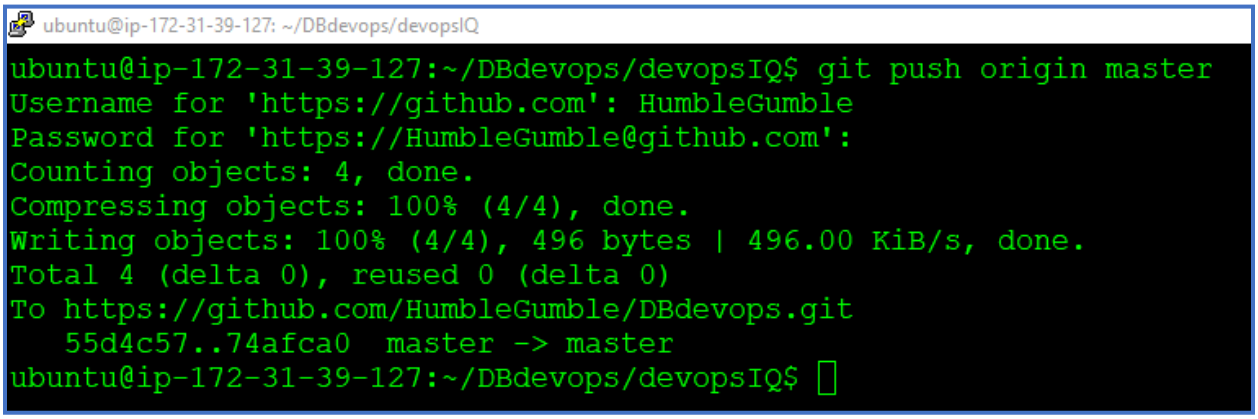
```
$ git add

$ git commit -m “new commit”
```

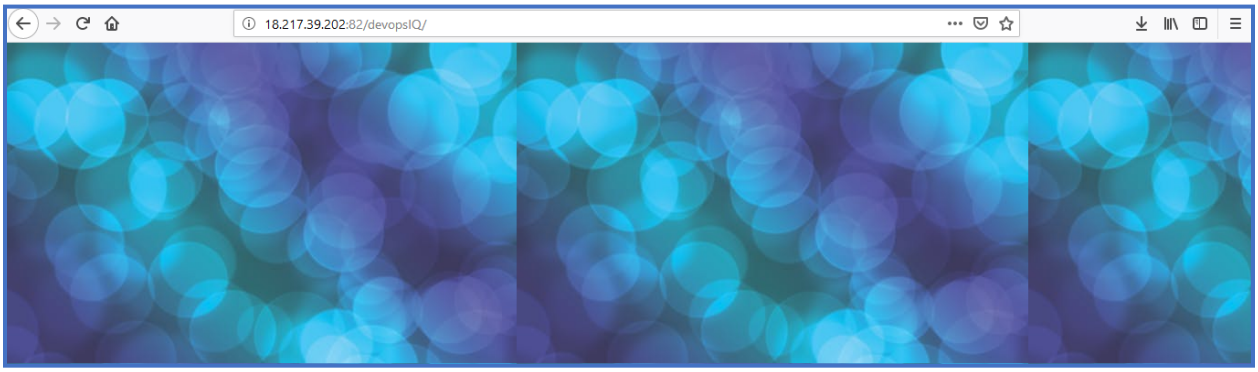


Step 33: Perform git push.

```
$ git push origin master
```



Step 34: Go to the browser. Refresh it. And you can see the background image got changed.



Congratulations! You have successfully completed the hands on.