**Project for Devops Certification: -**

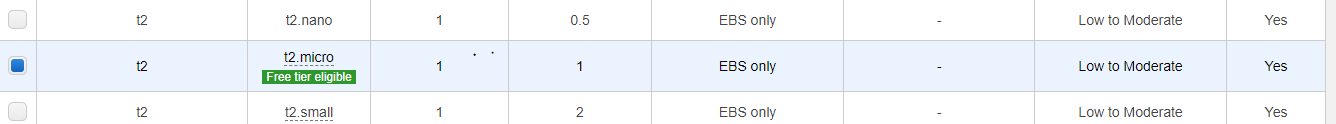
**Subject: - configure and manage slave node using automated build pipeline**

* **Create AWS EC2 instance (Master and Test Server)**

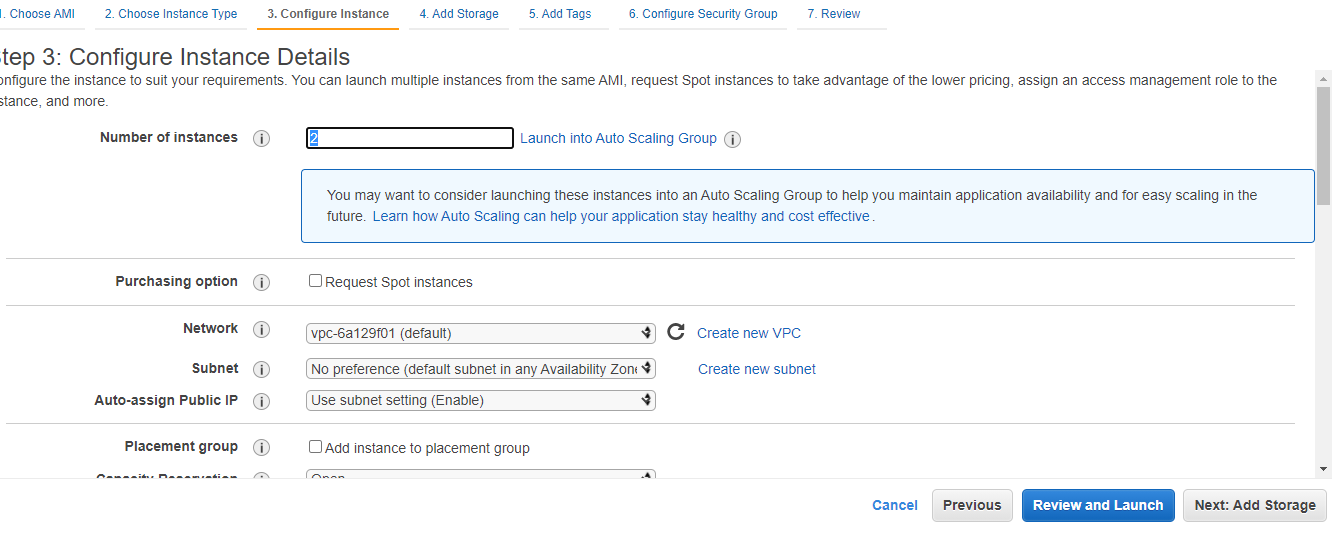
1. Log in to AWS console.
2. Launch a Virtual Machine with EC2 instance.
3. Click on the next. Select Ubuntu server 18.04 LTS(HVM)



1. Click on the next - For Instance, type we have selected free tier eligible t2. micro.

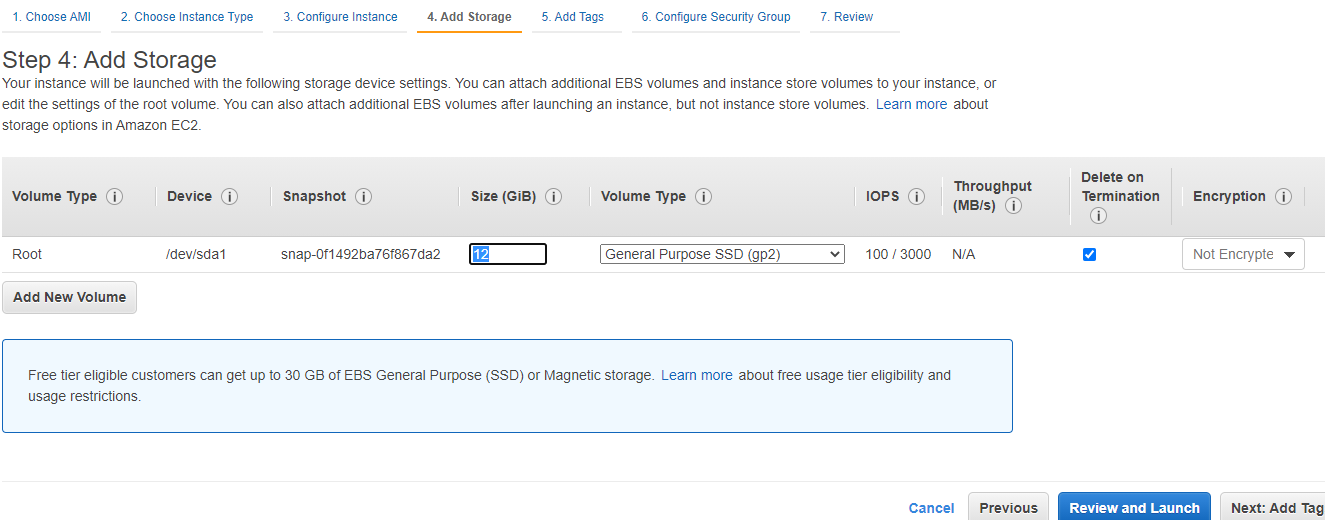


1. Click on the next - Number instance we making 2.

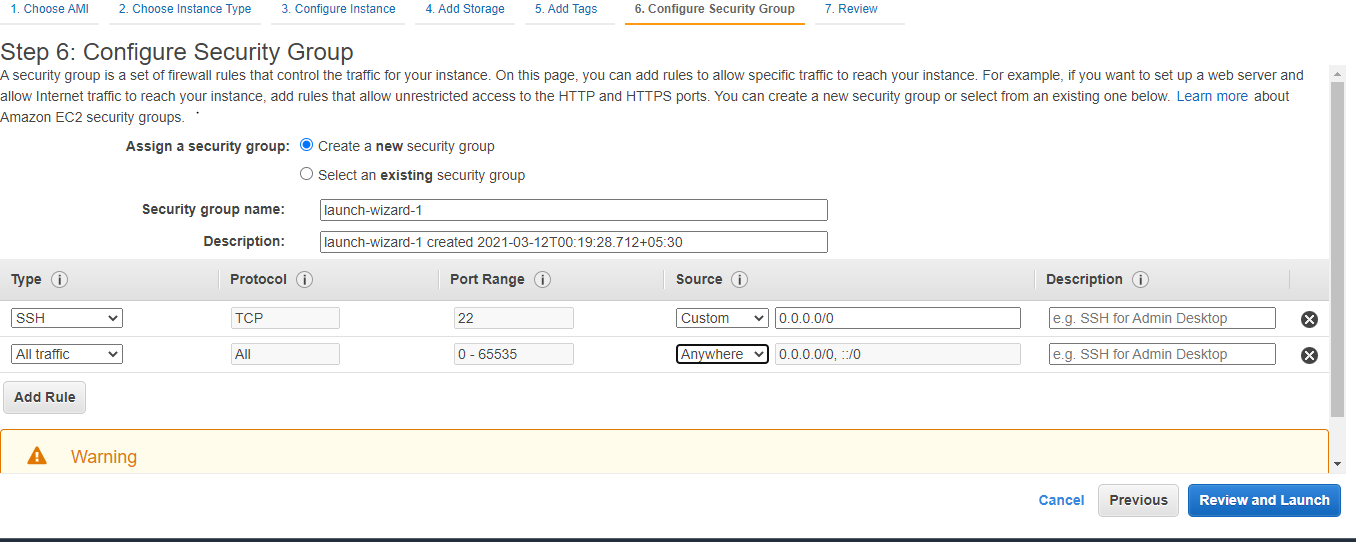


* 1 instance for the master
* 2 instances for web deployment server

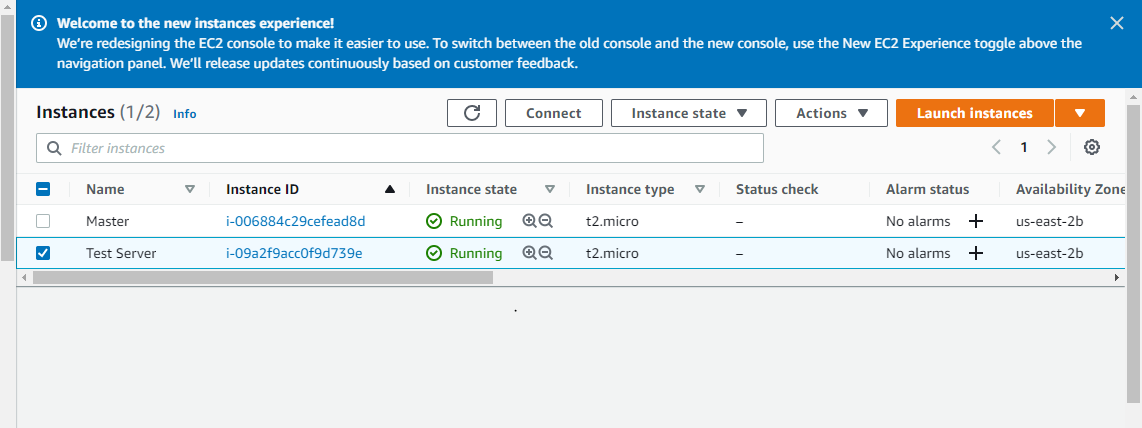
1. Click on the next, add storage and extend upto 12 GB.



1. Click on the next, add tags.
2. Click on the next, configure Security Group.
3. Now we will require to open all the port.
4. Click on add rule for all traffic.



1. Then, review and launch.
2. Create a new keypair. Download the keypair. key name is “**project**”
3. Instance has been created.



1. We have given the name to instance **“Master and Test Server”.**

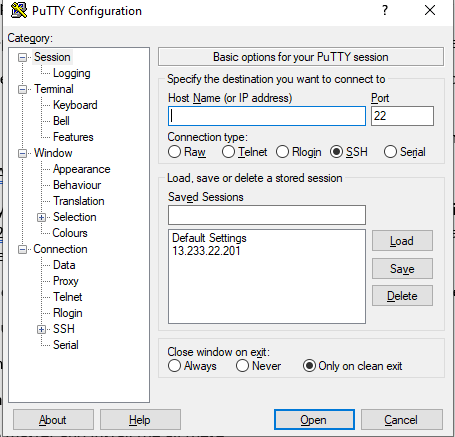
**Subject - Deployment of the project is PHP application we will handling Test Server machine to completely from the Master.**

* **Configuring Test Server with Master and deployment to Test Server machine.**

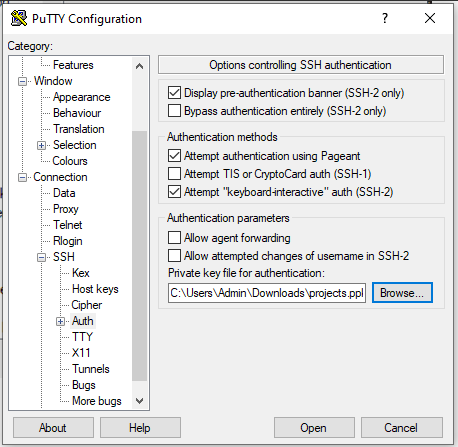
**Overview: -**

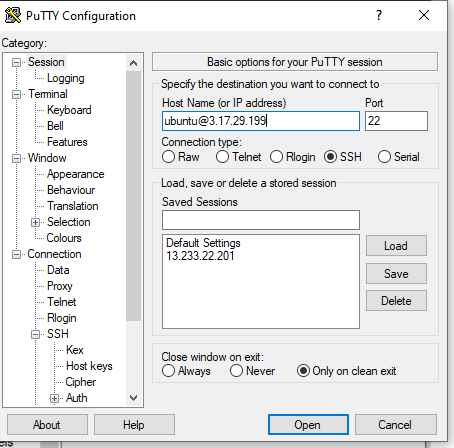
* In Master machine we require to install Git, JAVA, Jenkins and ansible.
* In Test Server machine for deployment of PHP application firstly, we require to dockerizing PHP application. Like we are putting PHP application as a docker image then we running docker image and containerization in Test Server Machine for deployment.
* On Test Server machine we need to docker because docker image is run in Test Server machine.
* After deployment we will do selenium testing. So for this we need to chromium browsers, chrome driver etc.
* Our Test Server machine as Jenkins Slave. On slave machine Java Should be install.
* Git should be install in slave machine.
* These are all dependencies we should install in slave machine.
* We have configuration management tool we have ansible here.
* Ansible should be install in test server machine.
* Task: -
* Install Java, Jenkins, Ansible on the machine. And add that Test Server Machine Ansible Hosts.
* Also add Jenkins in our Test Server.
* Then we will deploy our application using the ansible first configure Test Server machine.
* Install all the dependencies Java, chromium browsers, chrome driver etc. on the Test Server.
* Using Jenkins, we have to deploy our PHP applications. We will dockerize first and then we will deploy it.
* Now 1st we will download the ansible playbook and download the all thing in master instances.
* (Jenkins, Java, Ansible in the master)
* The thing is very important because we are using instance type t2. micro. there is some limitation t2. micro 1 GB of RAM per instance. Limitation of 1 GB RAM is that you can’t install the Jenkins and puppet in same machine.
* Since, puppet and Ansible we are using for same thing like configuration management.
* So, we don’t require both of them.
* We are using only ansible here for our project.
* Now, Master machine we will install Git, Java, Jenkins.
* Let’s connect to master and install the all these.

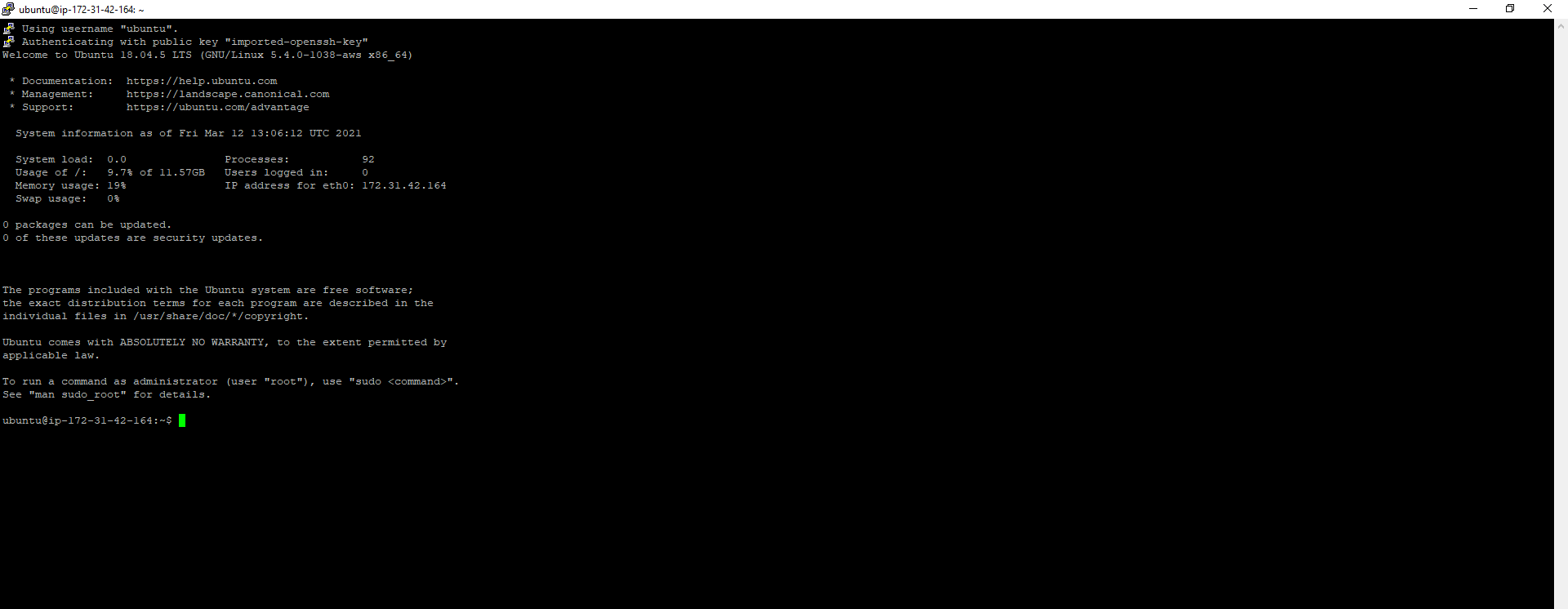
1. Open the Putty for connect to Master.



1. Enter the hostname or IP address. IP address is: - 18.220.79.166.
2. We will set project.pem key pair, so firstly we will create the .ppk file for connect to master.
3. Noe selects the project.ppk file in SSH authentications.



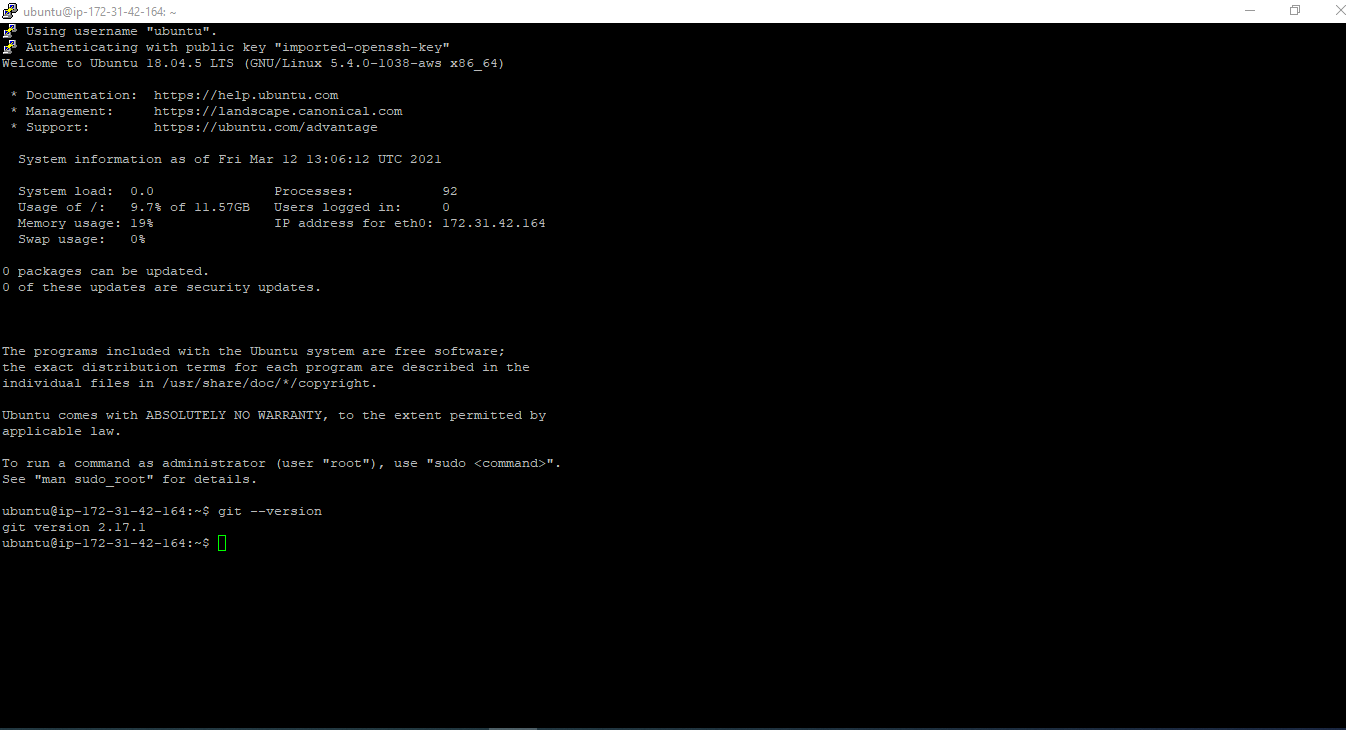




1. Now EC 2 instance Master has been connected through putty.
2. Now, we are installing all the require tools.
3. For Git: - EC 2 instance Master has already installed Git by default.

To check the git version.

**Command: - git --version**



1. For JAVA: - installation

Firstly, Update the repositories:

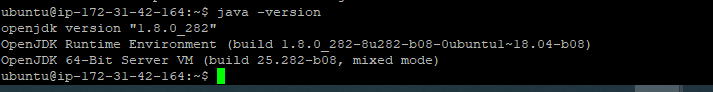
**Command: - sudo apt-get update**

Install Open JDK

**Command: - sudo apt-get install openjdk-8-jdk**

Verify the version of the JDK:

**Command: - java -version**



1. For Jenkins: - installation

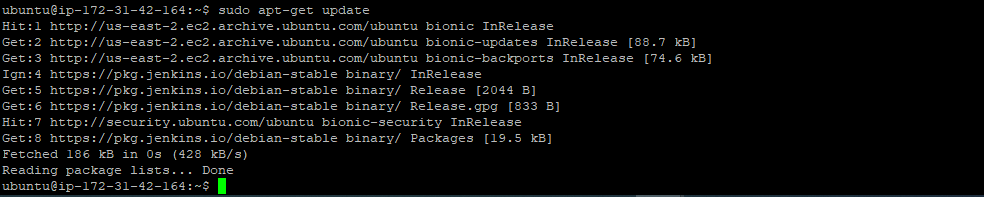
**Run the command for Jenkins’s installation: -**

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add –



sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > \

/etc/apt/sources.list.d/jenkins.list'

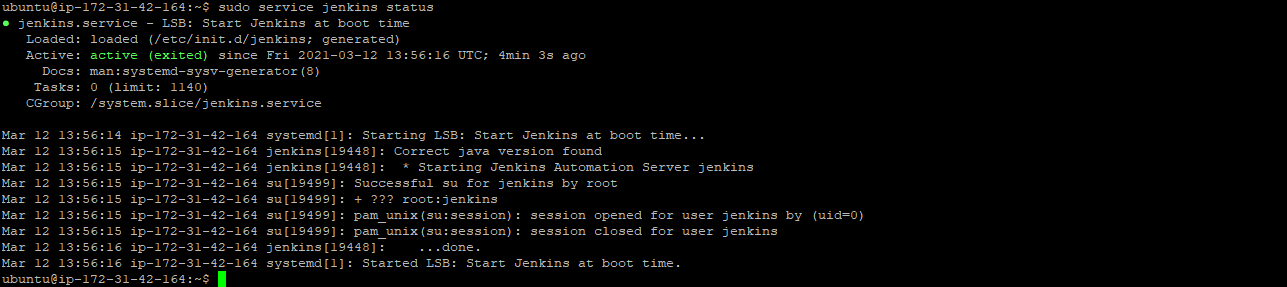


sudo apt-get update

sudo apt-get install jenkins

1. Our jenkins should be installed.

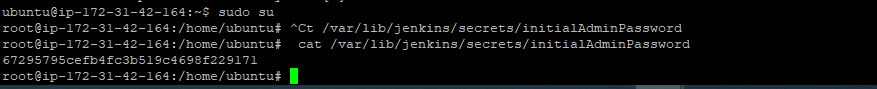
We know the service status command: - **sudo service jenkins status**



1. Now jenkins actively running in master.
2. We will connect to Jenkins in the browser.

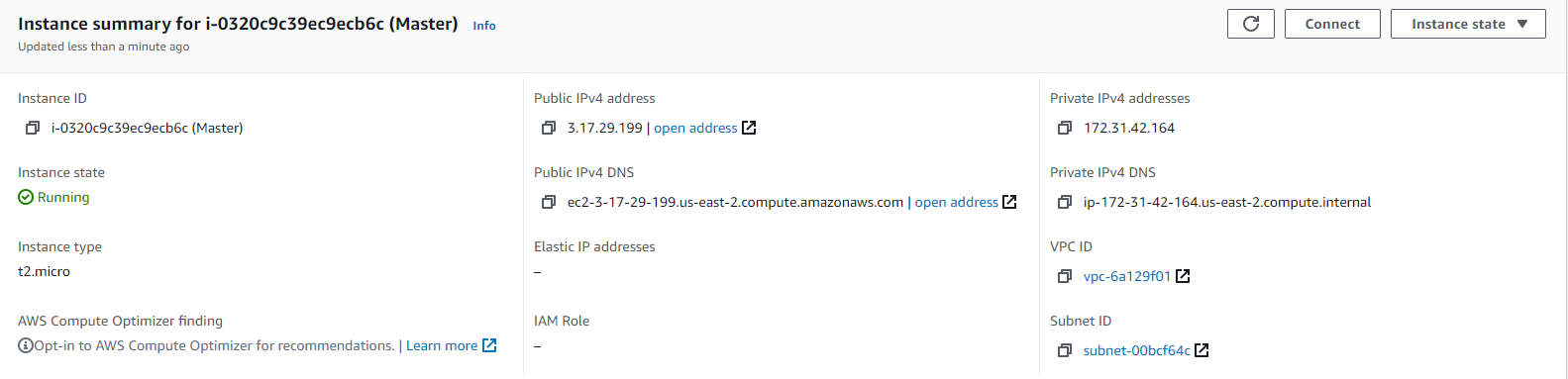
Command: -

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



**initialAdminPassword: - 67295795cefb4fc3b519c4698f229171**

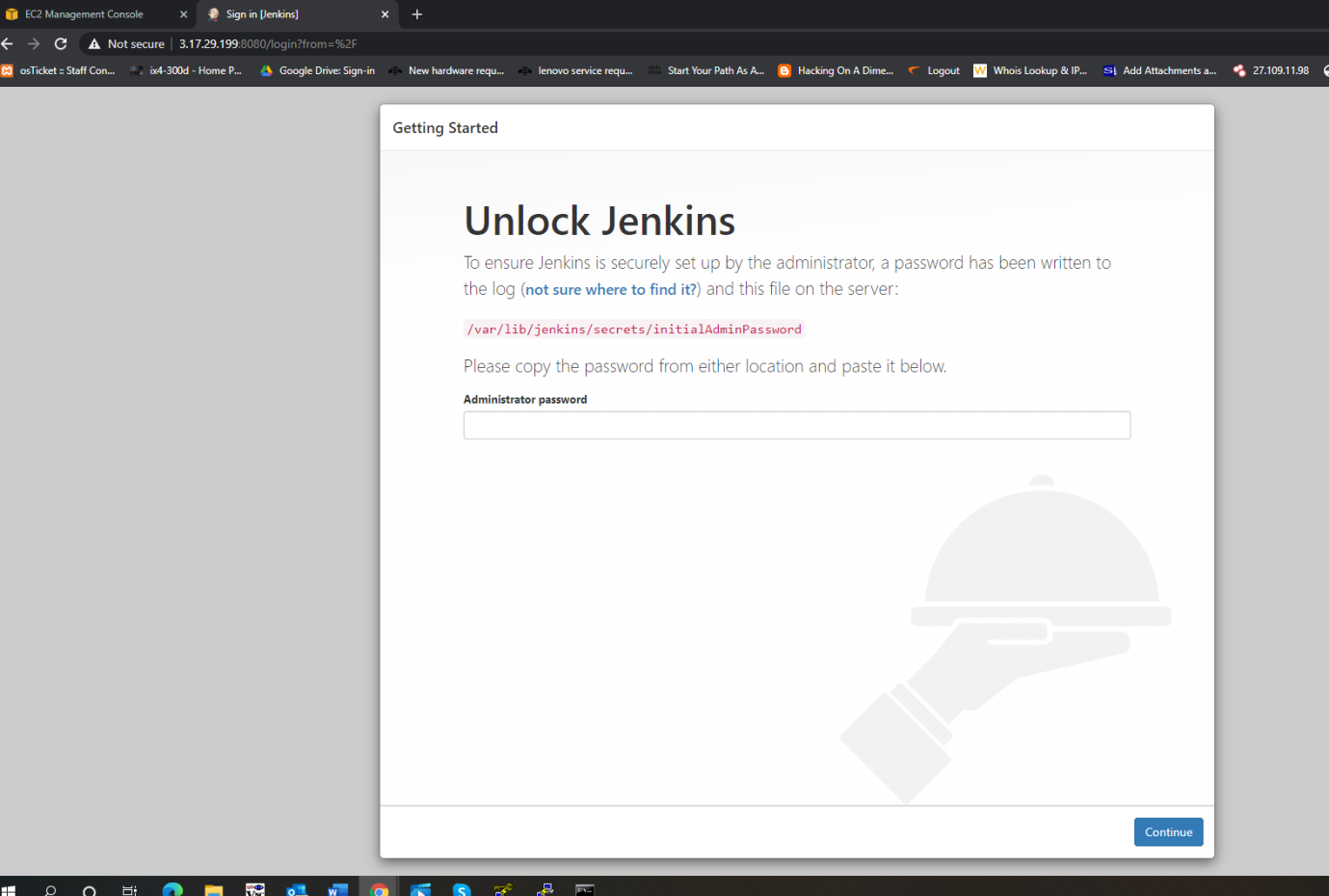
1. For Open Jenkins in browser: - we have used public IP of Master EC 2 instances.



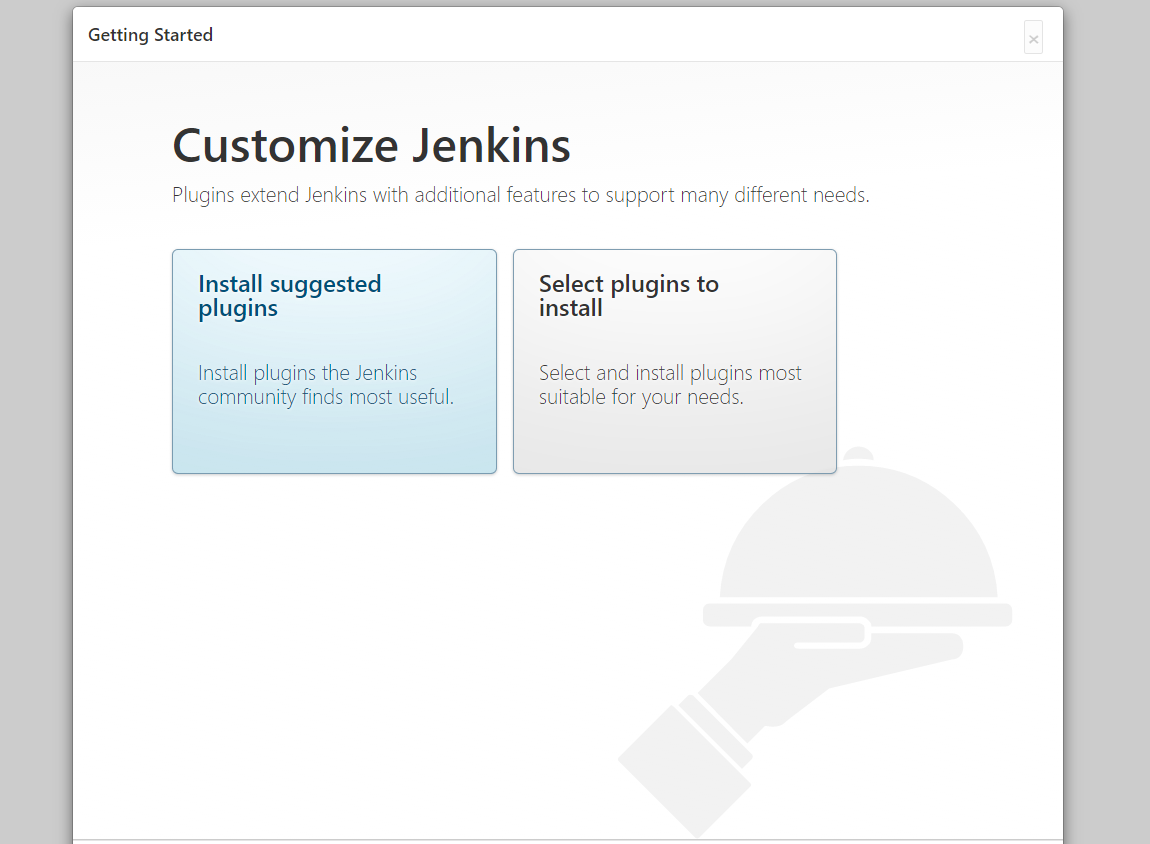
Public IP: - 3.17.29.199

1. Use public IP in browser.
2. Jenkins Run as port 8080.

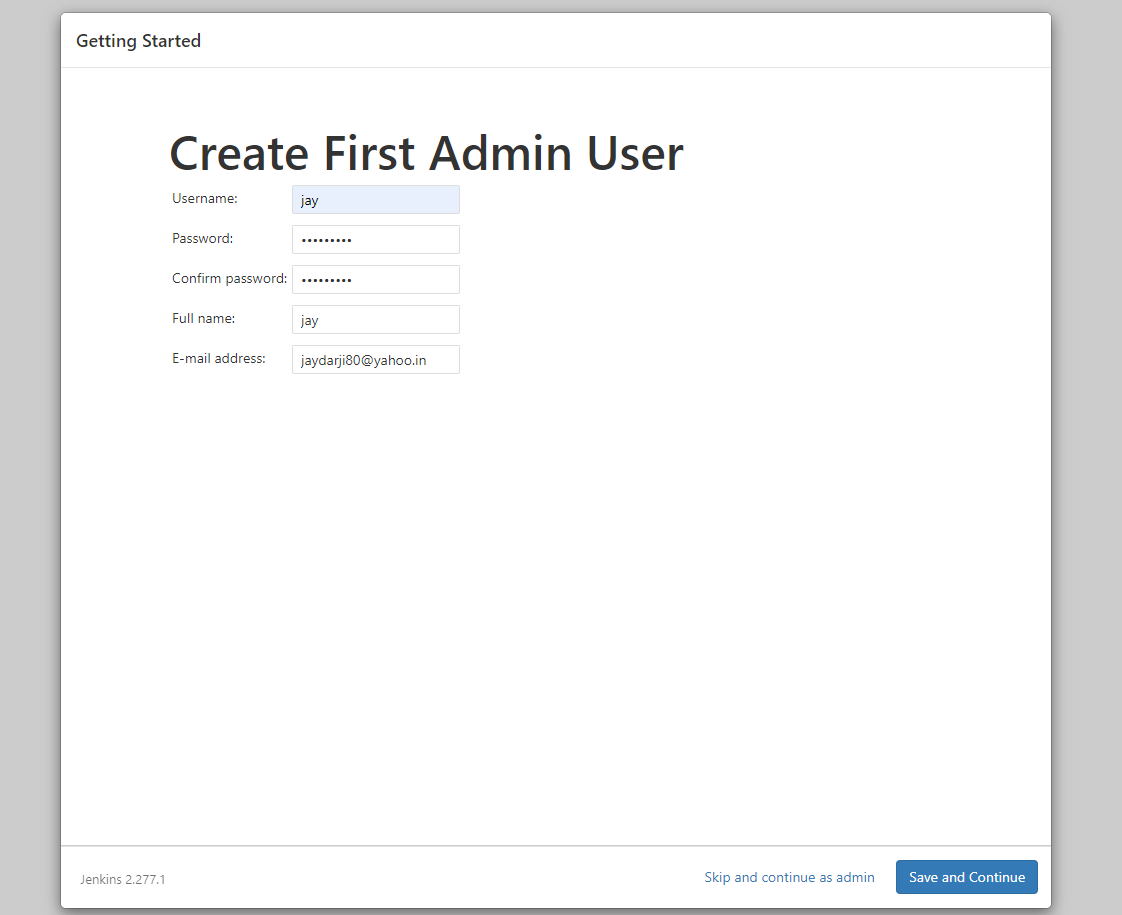
URL: - <http://3.17.29.199:8080/login?from=%2F>

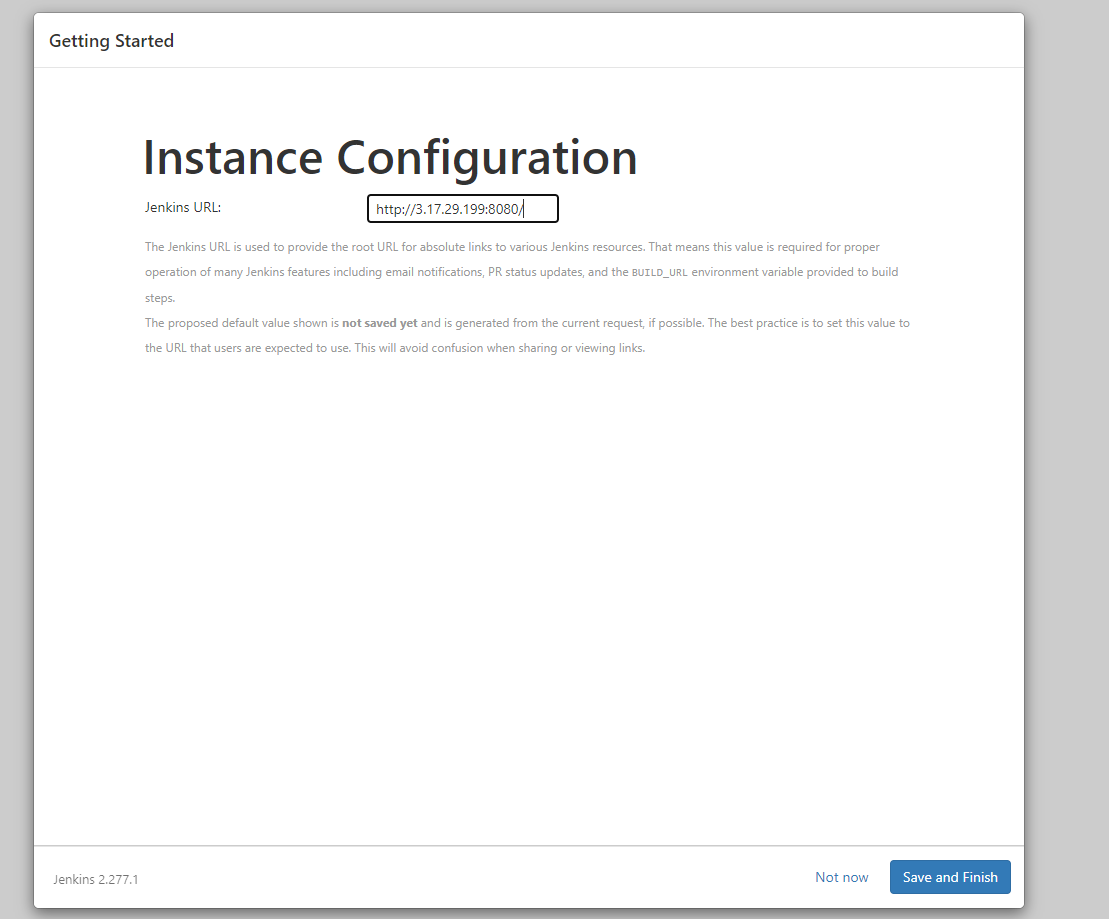


1. Add the administrator password: - **67295795cefb4fc3b519c4698f229171**
2. Add Password and continue.
3. Then Select or click on: - **Install suggested plugins.**

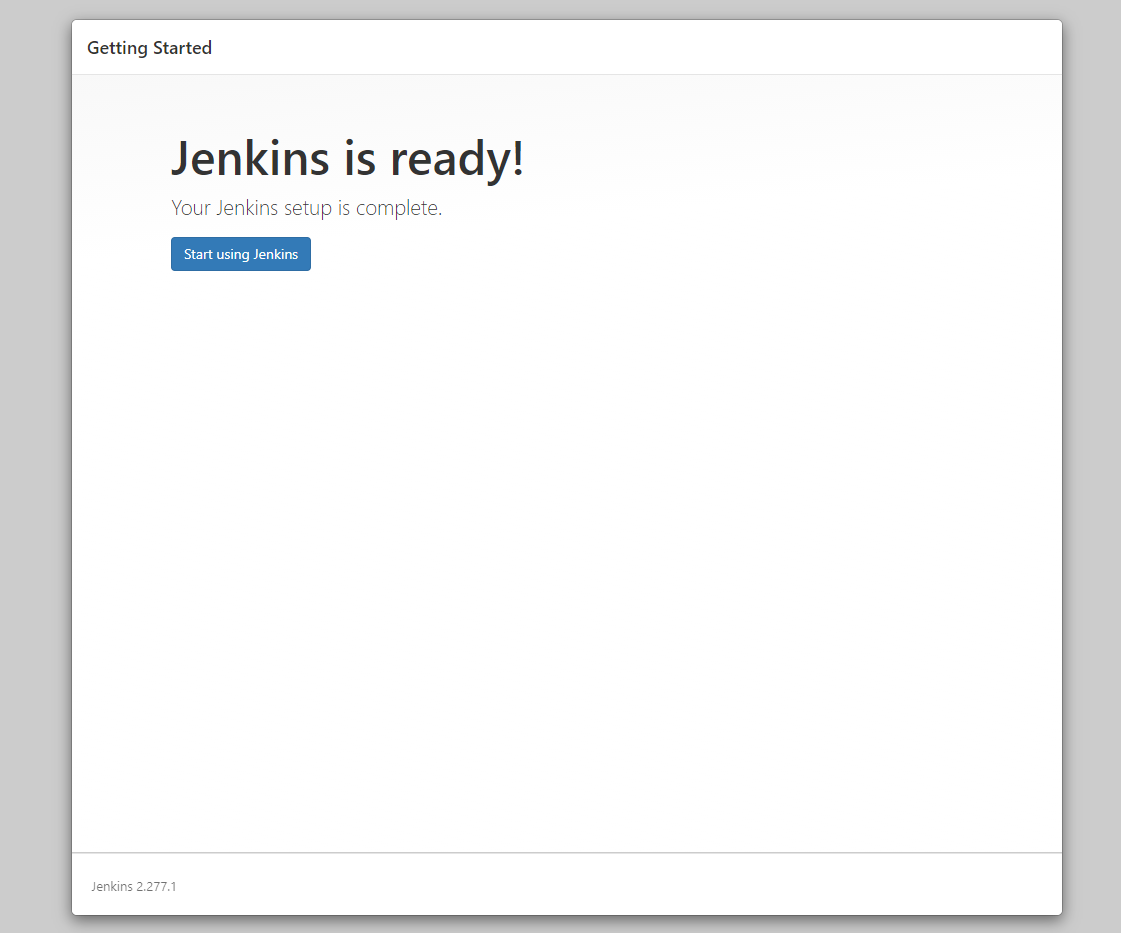


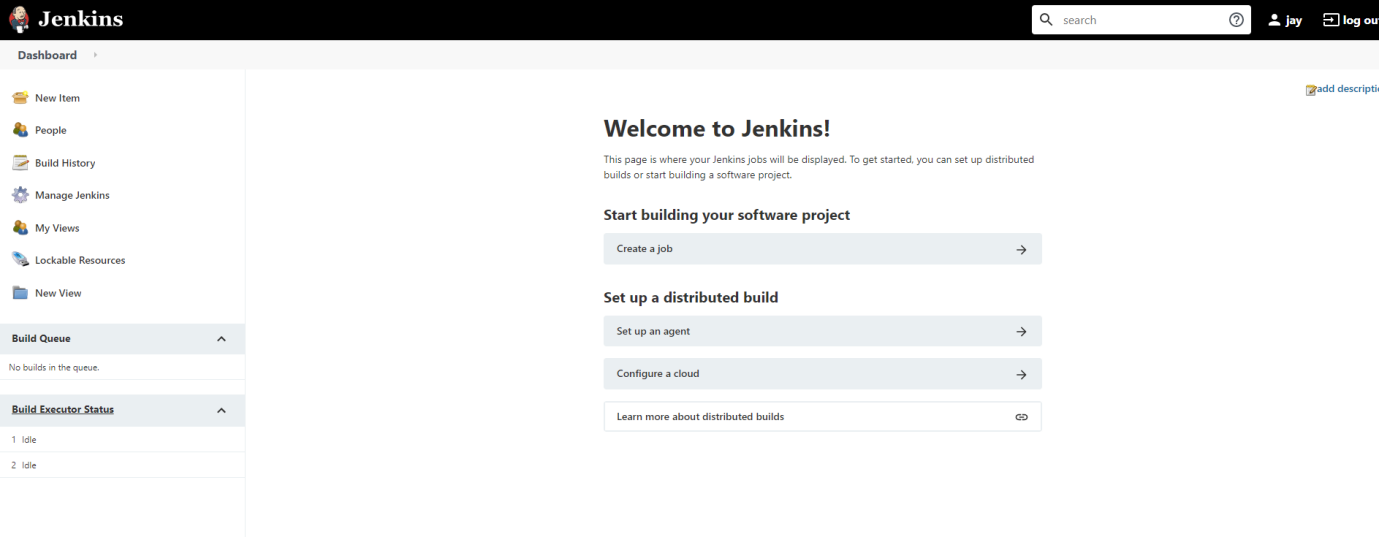
1. **Create First Admin User**





1. Start Using Jenkins.



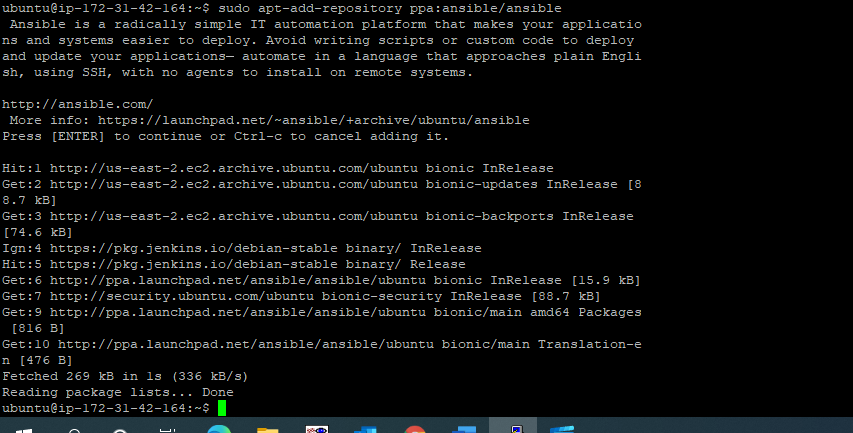


1. Now, we will install the ansible in Master EC2 instance.

To using Ansible as a means of managing our server infrastructure, we need to install the Ansible software on the machine that will serve as the Ansible control node.

From our control node, run the following command to include the official project’s PPA (personal package archive) in your system’s list of sources:

**sudo apt-add-repository ppa: ansible/ansible**



22. Next, refresh your system’s package index so that it is aware of the packages available in the newly included PPA:

**sudo apt update**

1. Following this update, we can install the Ansible software with:

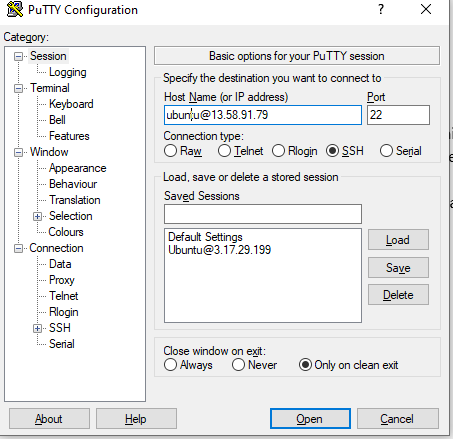
**sudo apt install ansible**

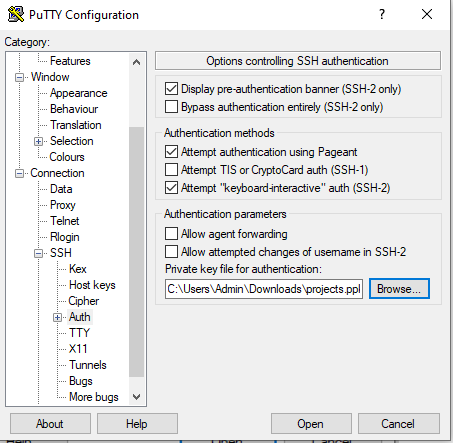
1. Our Ansible control node now has all of the software required to administer hosts. Next, we will go over how to add our hosts to the control node’s inventory file so that it can control them.
2. After installation of Ansible in Master machine. Then we will install the ansible in Slave machine (Test Server).

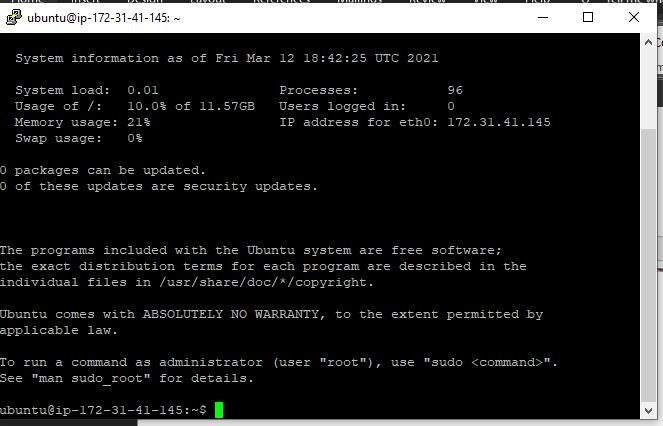
Now we will connect the Test Server instance EC2 via Putty.

Test Server Public IP: - 13.58.91.79

Key pair: - Projects

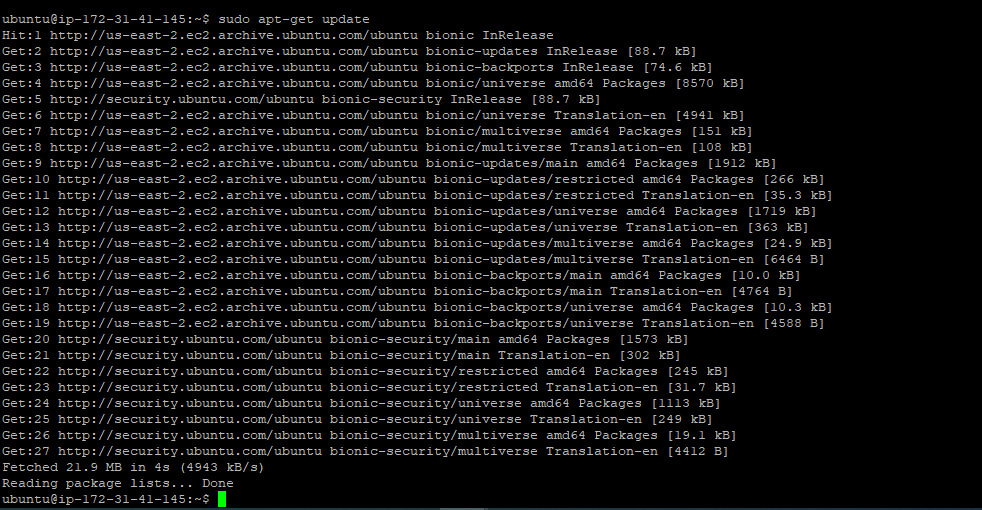






Slave Test server EC2 instance.

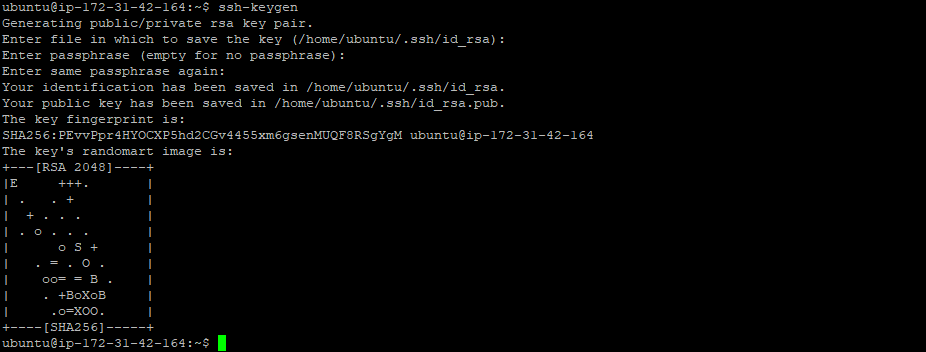
1. First thing does update the packaging.



1. We add the machine as ansible host and SSH connection will be establishing in both machines.
2. We have to established SSH connection. Master to the Test Server Machine.
3. We have to generate SSH key.

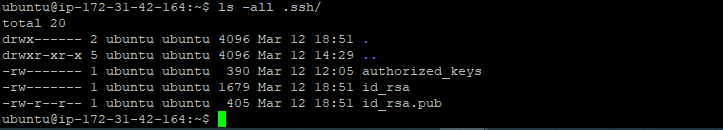
First SSH key generate in Master machine.

Command: - ssh**-keyen**



1. Now we will check our SSH key.

**Command: - ls -all .ssh/**



Id\_rsa – Private key

Id\_rsa.pub- public key

1. Copy to the public key of the Master machine and authorized\_keys of the Master machine.

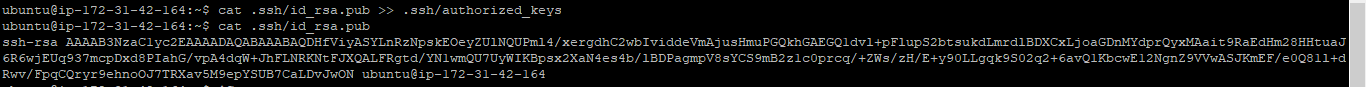
Command: - **Cat .ssh/id\_rsa.pub >> .ssh/authorized\_keys**

1. **While making the SSH connection they checking the Key. Key is valid or key is matching then connection would be established.**

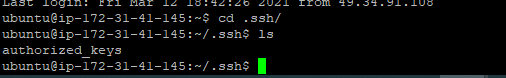
**Command: - Cat .ssh/id\_rsa.pub**

**Open the public key and copy the key.**

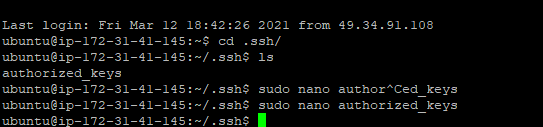
**ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDHfViyASYLnRzNpskEOeyZUlNQUPml4/xergdhC2wbIviddeVmAjusHmuPGQkhGAEGQ1dvl+pFlupS2btsukdLmrdlBDXCxLjoaGDnMYdprQyxMAait9RaEdHm28HHtuaJ6R6wjEUq937mcpDxd8PIahG/vpA4dqW+JhFLNRKNtFJXQALFRgtd/YN1wmQU7UyWIKBpsx2XaN4es4b/lBDPagmpV8sYCS9mB2z1c0prcq/+ZWs/zH/E+y90LLgqk9S02q2+6avQ1KbcwE12NgnZ9VVwASJKmEF/e0Q81l+dRwv/FpqCQryr9ehnoOJ7TRXav5M9epYSUB7CaLDvJwON ubuntu@ip-172-31-42-164**



1. **Now go to the slave , SSH directory**



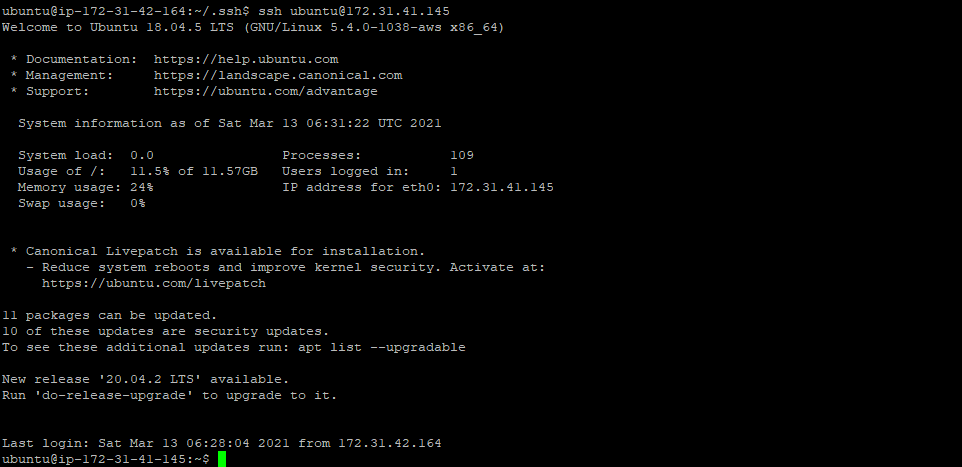
1. **Copy the public key master machine in the slave machine of authorized file.**
2. **Command: - sudo nano authorized\_keys in slave machine**
3. **Open the command line.**
4. **Paste master machine public in second line.**
5. **We will save the files.**



1. **Now we have made the SSH connection between master machine and Slave /Test server machine.**
2. **For SSH.**

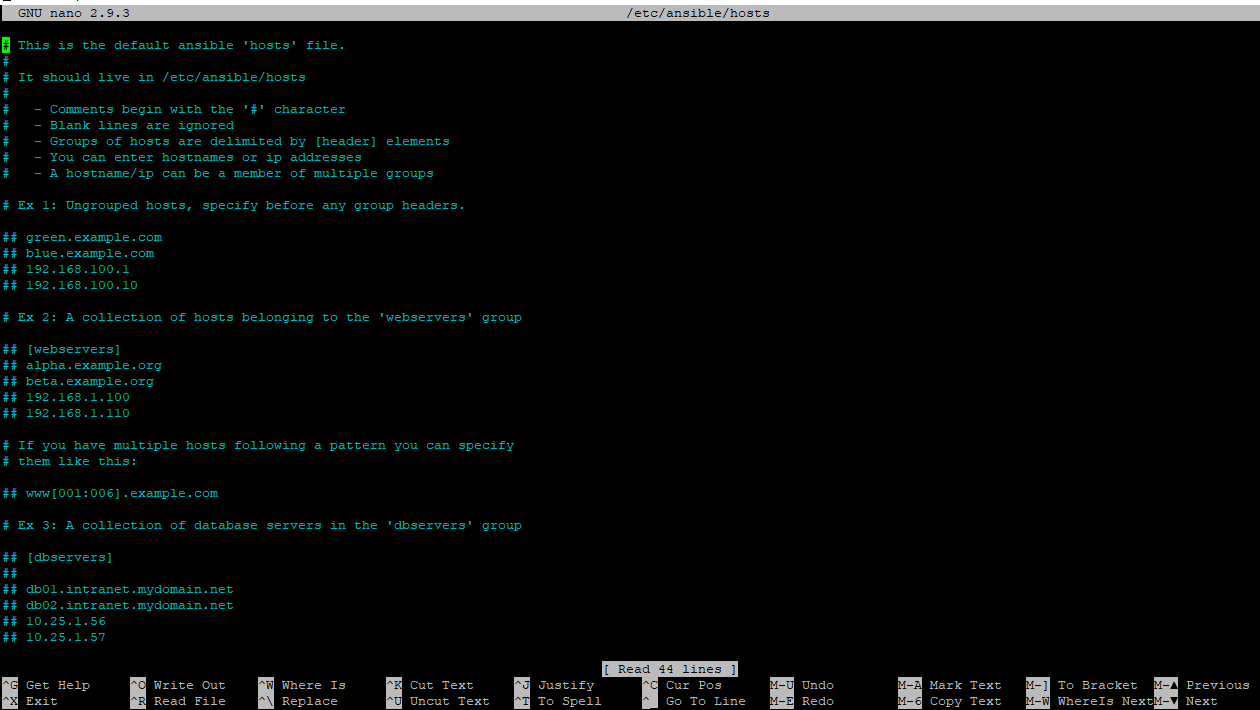
**Command: - ssh slave machine user name@private IP to Test Server**

**ssh root@** **172.31.41.145**

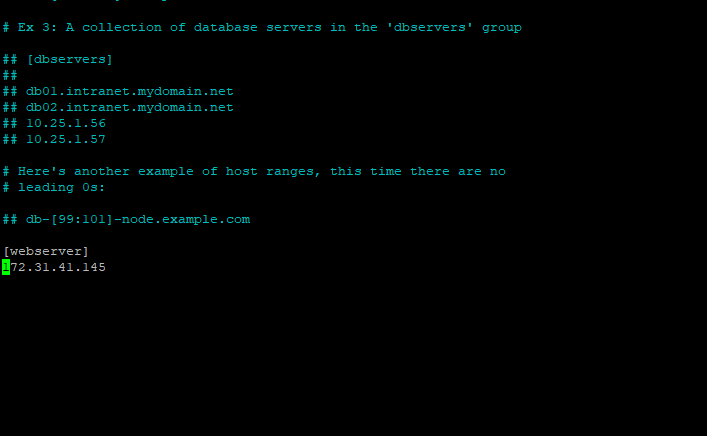


**It’s connected to the slave through ssh.**

1. **Now we have added the second machine in the inventory file of ansible we have all those machines.**
2. **In master machine edit this file sudo nano /etc/ansible/hosts (this is the inventory file)**



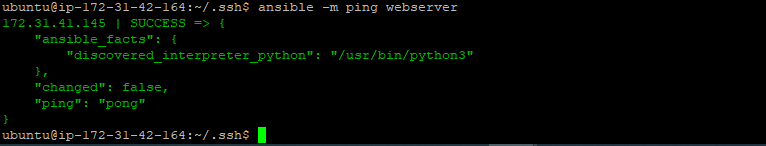
1. **Creating the new group [webserver]**



**For new we are using private ip of Test Server machine.**

1. **Now we have edited the Test Server machine inventory file of ansible.**
2. **We try to ping that machine which the ansible reach that machine or not.**
3. **To do that, command ansible -m ping groupname.**

**Command: - ansible -m ping webserver**



1. **We able to ping the second machine using ansible command. Once ansible connection is establish.**
2. **Now, in master machine we have to installed git, Java, Jenkins, Ansible.**
3. **Now, Ansible we have to configure in Test Server machine. We have installing require tools like git, java , docker, chrome driver, chromium browser on to the Test server machine.**
4. **For install the chrome driver - firstly the chrome driver installs to the master machine.**
5. **Then directly copy the chrome driver Master to Test Server.**
6. **Run the command directly using ansible to Test Server machine. There are 5-6 command. It will increase the playbook size. Rather we do first install the chrome driver on to the master and then directly copy to the Test server machine using ansible playbook.**
7. **For chrome driver installation command is below: -**

**sudo apt-get update**

**sudo apt-get install -y unzip xvfb libxi6 libgconf-2-4**

**sudo curl -sS -o - https://dl-ssl.google.com/linux/linux\_signing\_key.pub | apt-key add**

**sudo echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google-chrome.list**

**sudo apt-get -y update**

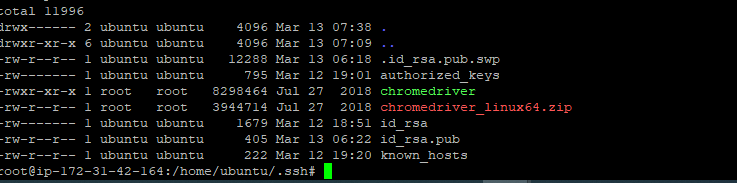
**sudo apt-get -y install google-chrome-stable**

**wget https://chromedriver.storage.googleapis.com/2.41/chromedriver\_linux64.zip**

**unzip chromedriver\_linux64.zip**

1. **Chrome driver has been downloading**

**(Root user - /home/ubuntu/.ssh)**



1. **Now, directly copy the chromedriver to our test server machine.**
2. **Write the ansible playbook to install the dependencies of the Test Server machine.**
3. **Creating first playbook now,**
4. **Any playbook name (command: - sudo nano project.yml)**
5. **Write the playbook.**

|  |
| --- |
|  |
| --- |
|  | - hosts: all |
|  | become: true |
|  | tasks: |
|  | - name: Install Git |
|  | package: |
|  | name: git |
|  | state: present |
|  |  |
|  | - name: Run update |
|  | apt: |
|  | update\_cache: true |
|  |  |
|  | - name: Install jdk |
|  | package: |
|  | name: openjdk-8-jdk |
|  | state: present |
|  |  |
|  | - name: Download Chrome browser |
|  | command: wget https://dl.google.com/linux/direct/google-chrome-stable\_current\_amd64.deb |
|  |  |
|  | - name: Install Chrome Browser |
|  | command: apt install ./google-chrome-stable\_current\_amd64.deb -y |
|  |  |
|  | - name: Run update |
|  | apt: |
|  | update\_cache: yes |
|  |  |
|  | - name: download chromedriver |
|  | command: wget https://chromedriver.storage.googleapis.com/87.0.4280.88/chromedriver\_linux64.zip |
|  |  |
|  | - name: install unzip |
|  | command: apt install unzip |
|  |  |
|  | - name: unzip chromedriver |
|  | command: unzip chromedriver\_linux64.zip |
|  |  |
|  | - name: move chromedriver |
|  | command: mv chromedriver /home/ubuntu/ |
|  |  |

1. **save name with project.yml**
2. **now we check the syntax of the playbook.**

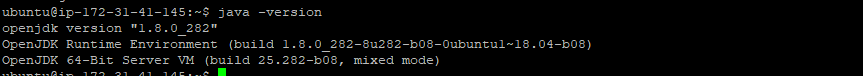
**Command: - ansible-playbook project.yml**

**Ansible-playbook project.yml --check**



1. **Go to the Test Server machine check java.**

**Java -version**



1. **In Test server chromedriver is there.**



1. **Then, Run the command**

**Sudo chmod -R 777 chromedriver**

* **Now we require to Docker on the slave (Test Server) machine.**

1. **Create the file docker.yml.**

**Sudo nano docker.yml**

**Script: -**

---

- hosts: all

  become: yes

  become\_user: root

  tasks:

  - name: Add Docker GPG key

    apt\_key: url=<https://download.docker.com/linux/ubuntu/gpg>

  - name: Add Docker APT repository

    apt\_repository:

      repo: deb [arch=amd64] <https://download.docker.com/linux/ubuntu> {{ansible\_distribution\_release}} stable

  - name: Install list of packages

    apt:

      name: "{{ item }}"

      state: present

      update\_cache: yes

    with\_items:

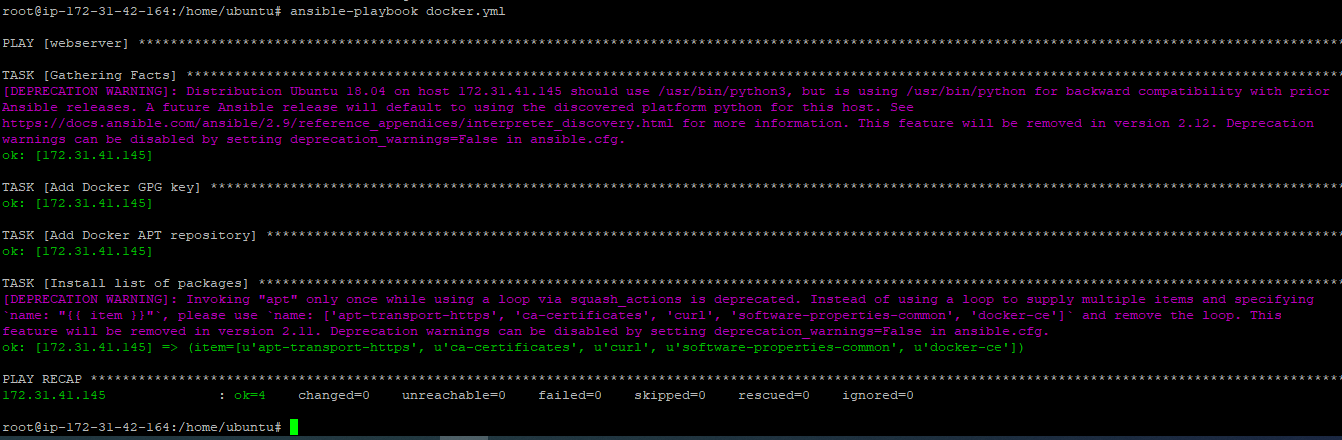
      - apt-transport-https

      - ca-certificates

      - curl

      - software-properties-common

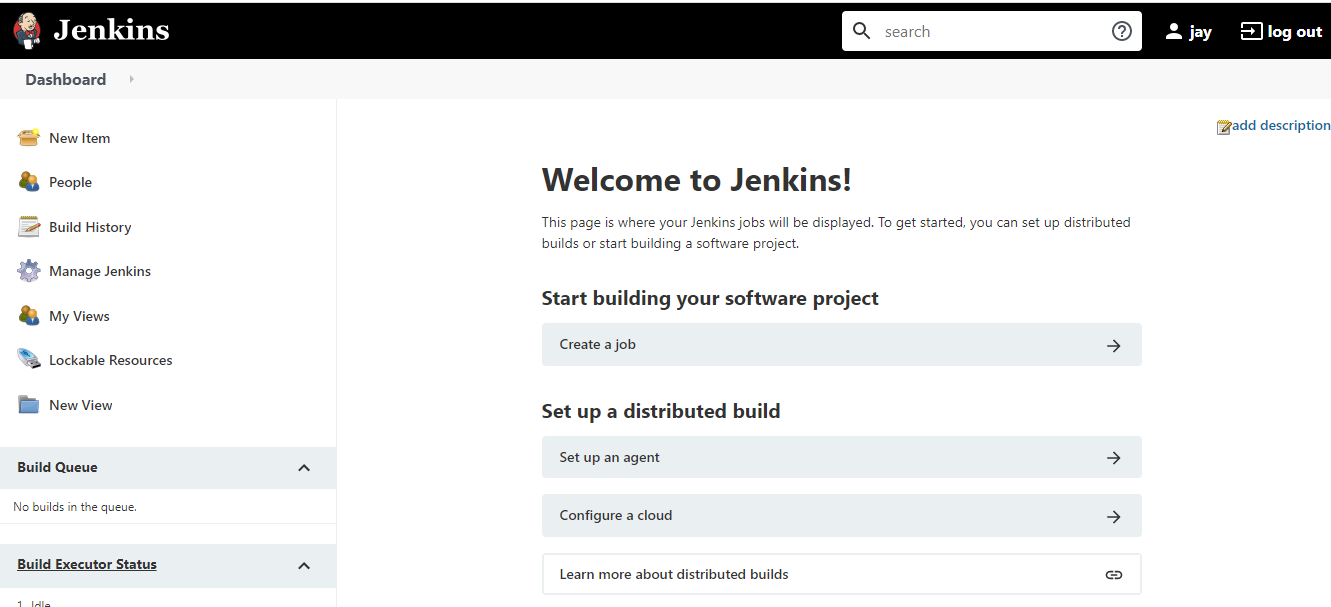
      - docker-ce



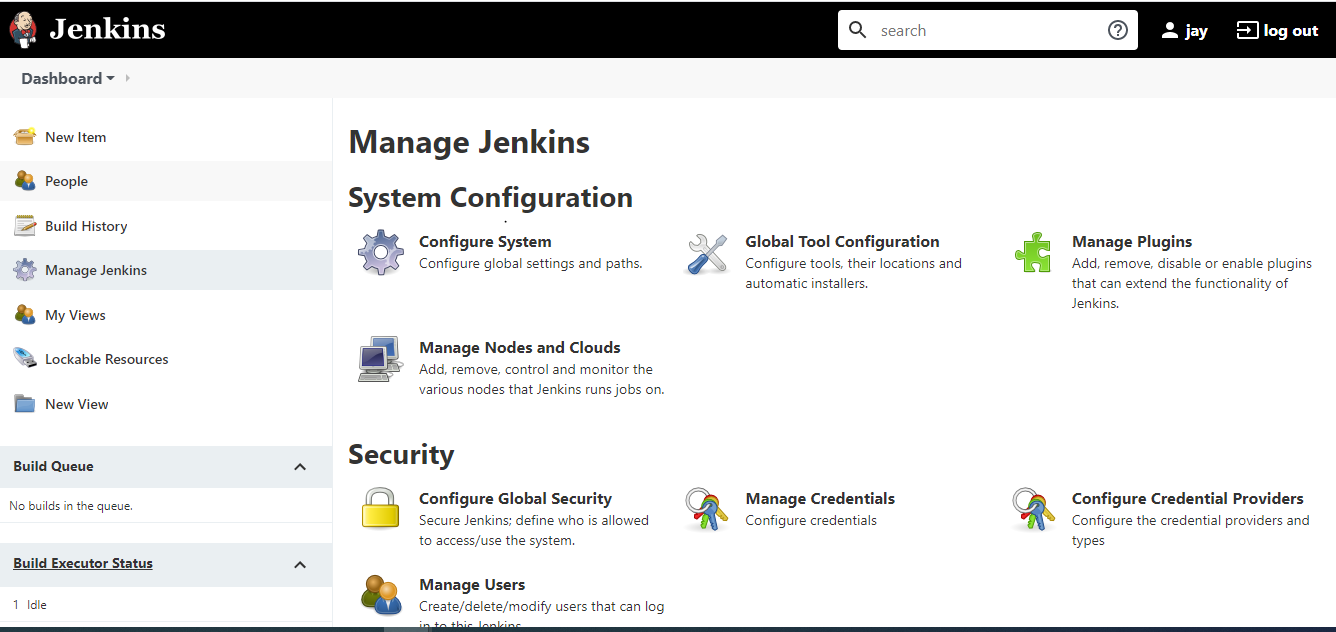
1. **Docker has been installed now**
2. **All the dependencies installed in Test Server. (Git, java, Chromedriver, chromium browser, Docker)**
3. **Ansible part has been done now.**

* **Add Test Server machine as Jenkins’s slave:**

1. **Open Jenkins.**
2. [**http://3.17.29.199:8080/**](http://3.17.29.199:8080/)
3. **Username: jay**



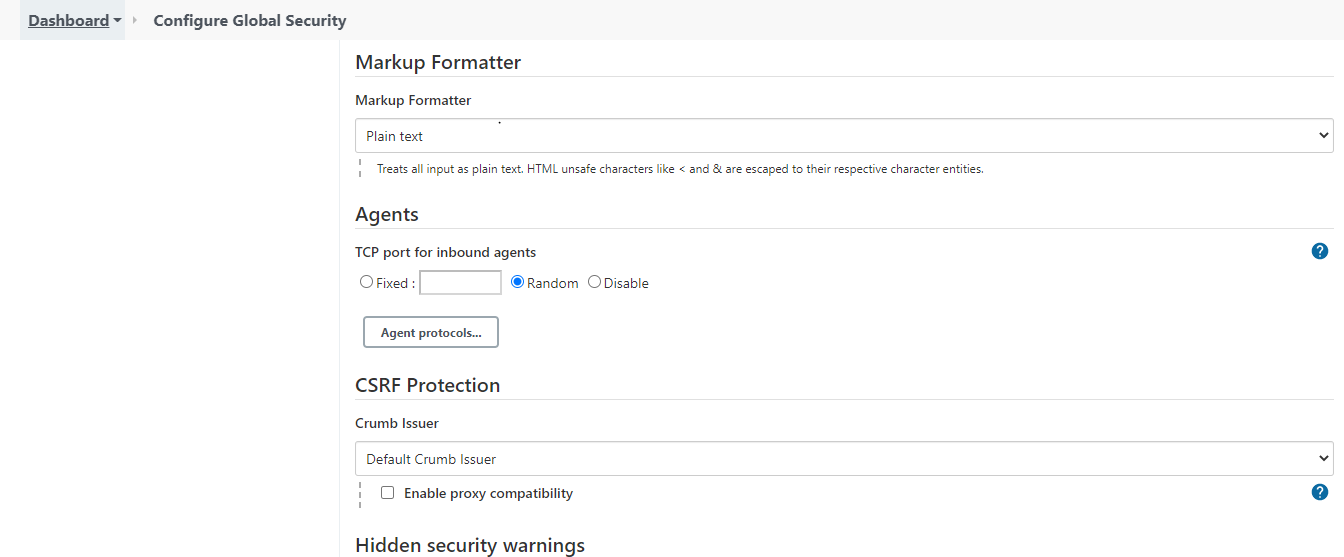
1. **Click on manage Jenkins.**



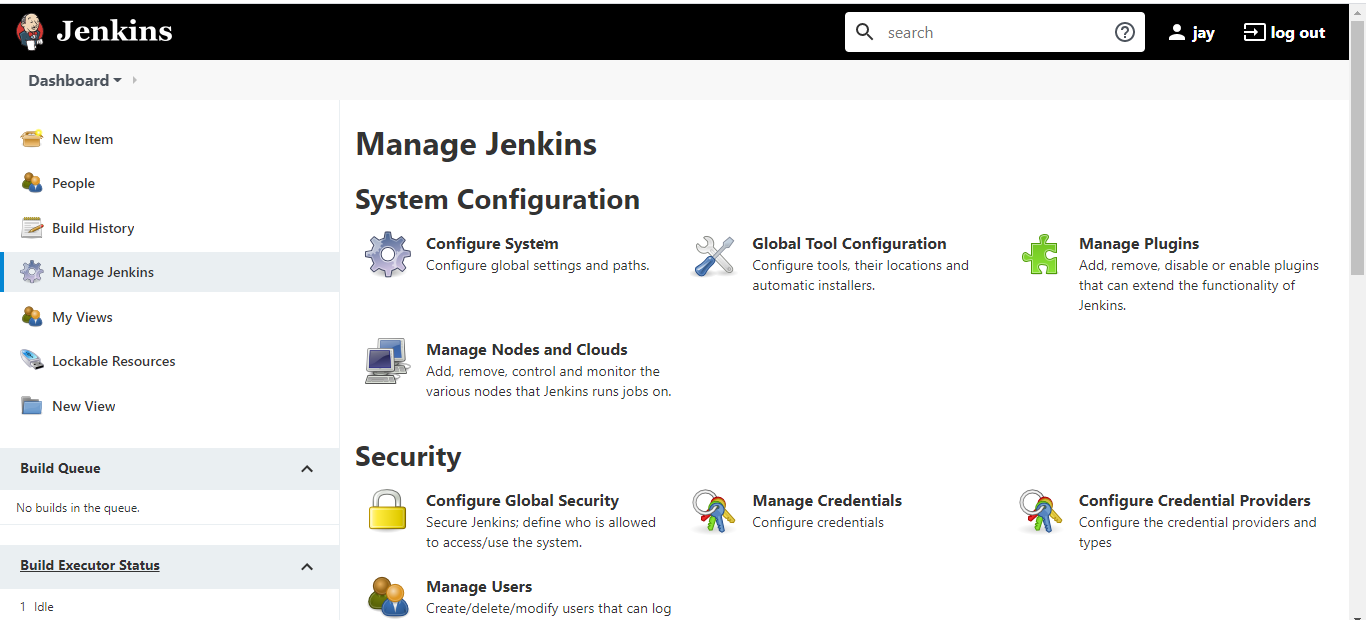
1. **Click on configure Global Security.**
2. **Firstly, we have to select the Agent.**

**TCP port for inbound agents**

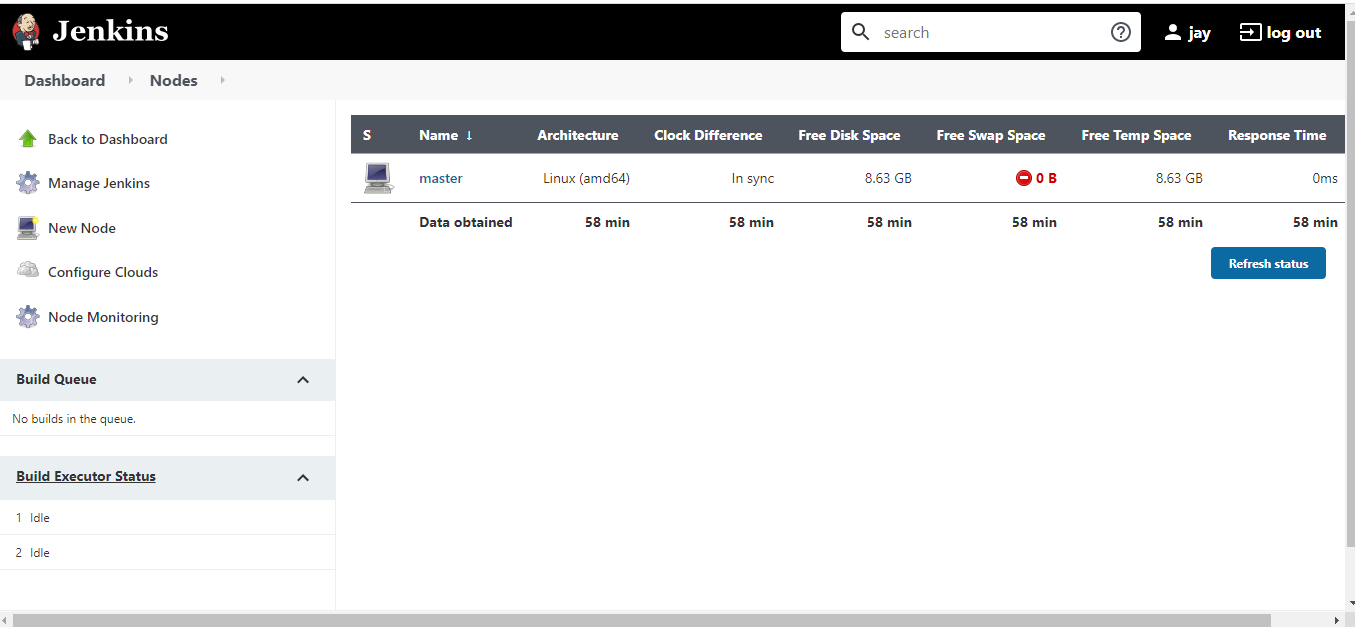
**Select: - Random**



1. **Then save.**
2. **Click on Manage Nodes and Clouds**



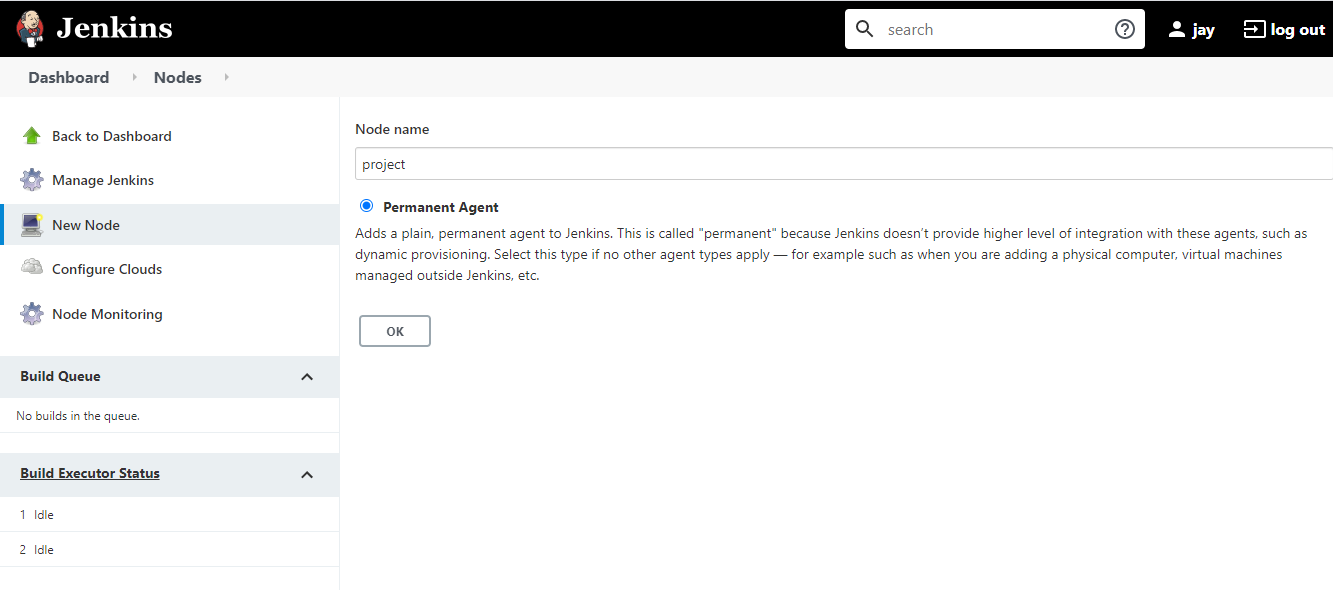
1. **Click New Node.**



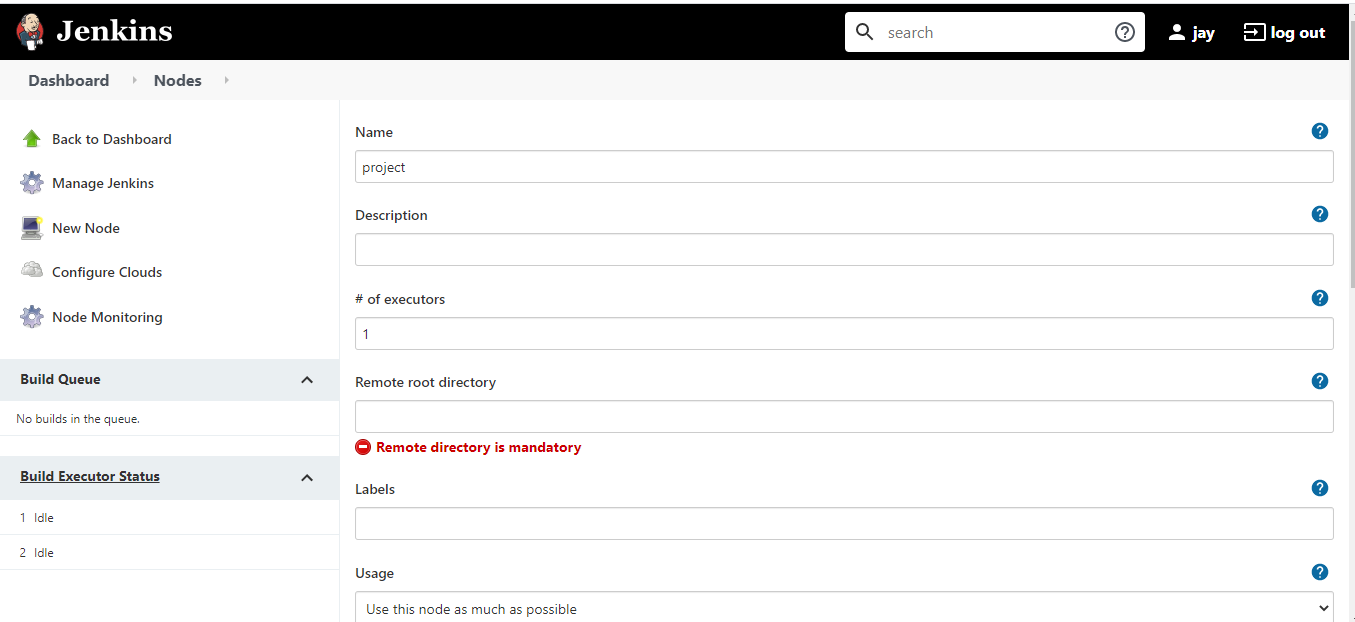
1. **Enter Node name – project**

**Select permanent Agent**

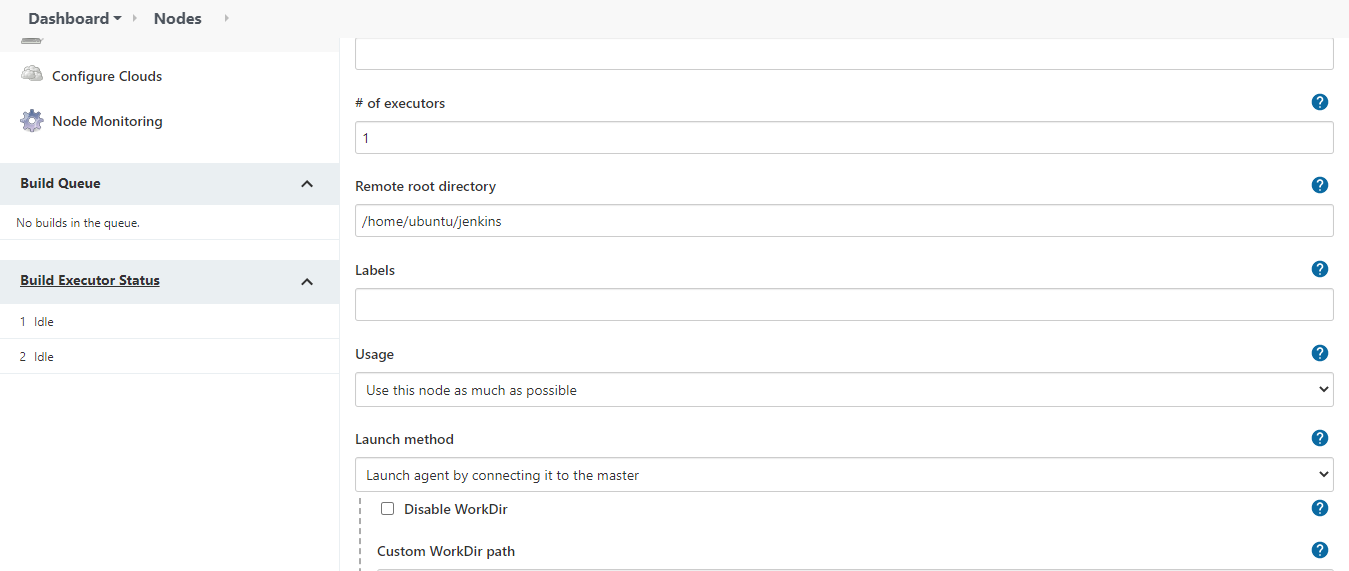
**Then click on OK**



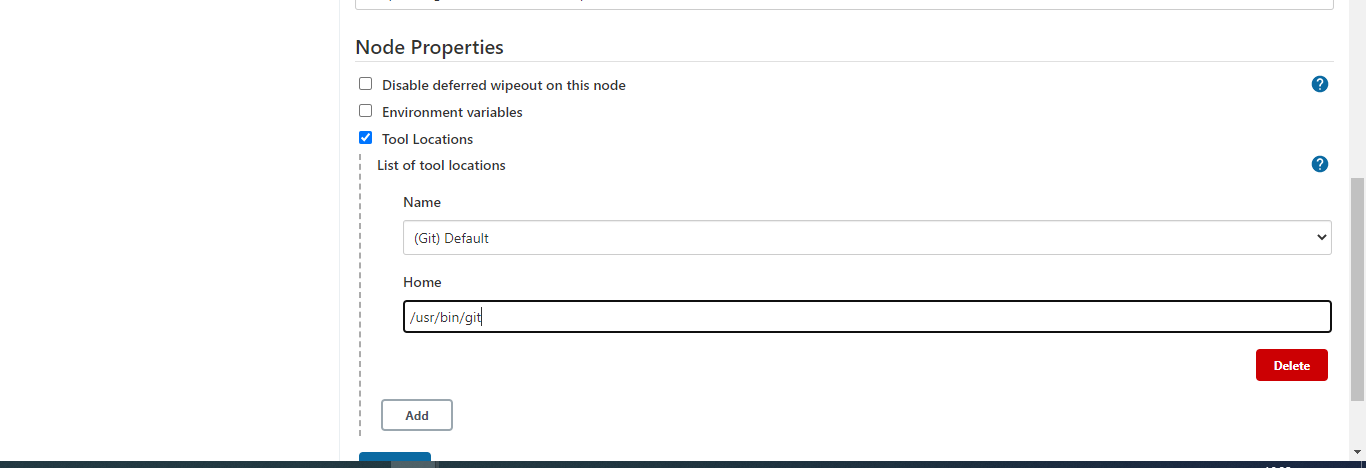
1. **Now , this is going to the configuration,**



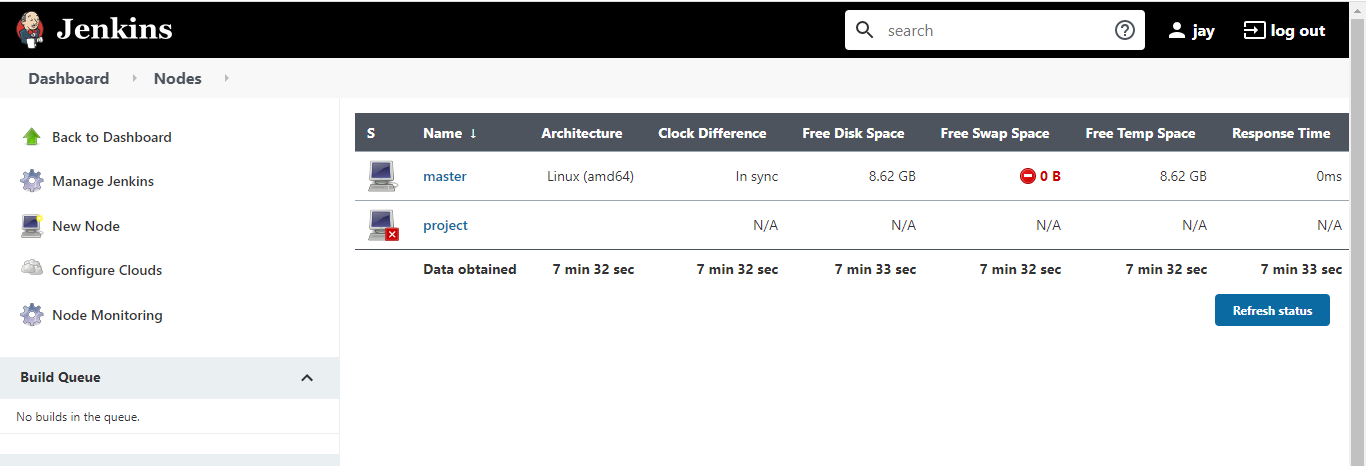
1. **Add Remote root directory.**



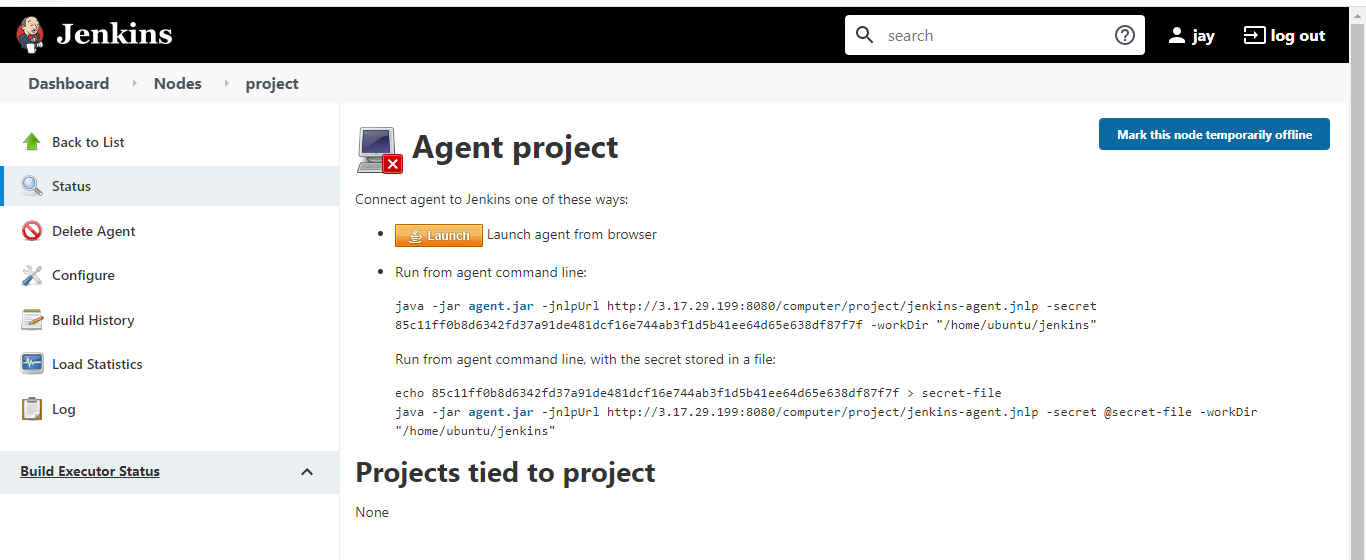
1. **Click on Tool location.**



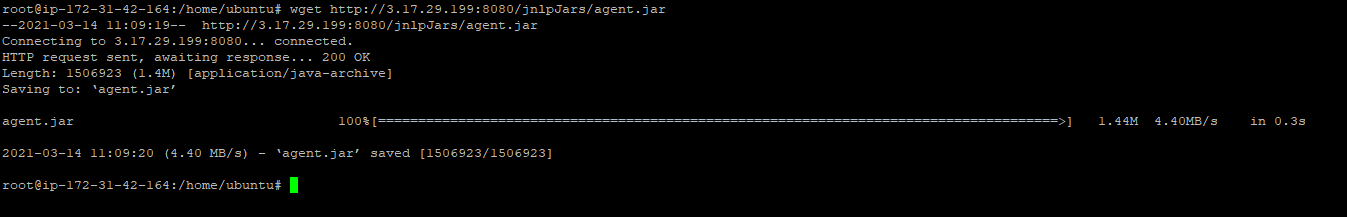
1. **Then Save.**



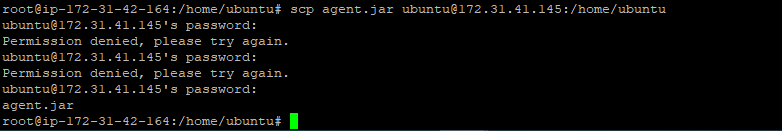
1. **Now click on project.**
2. **We have to do establish connection.**



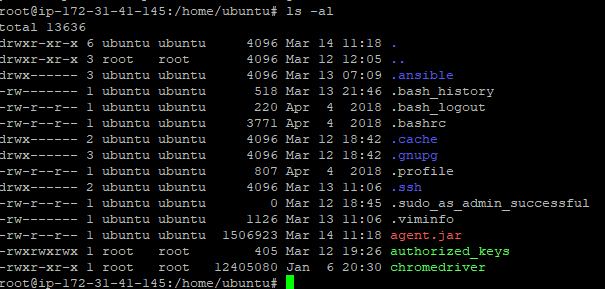
1. **Right click on – agent.jar and copy link address.**
2. **Go to master machine.**
3. **Type the wget** [**http://3.17.29.199:8080/jnlpJars/agent.jar**](http://3.17.29.199:8080/jnlpJars/agent.jar)



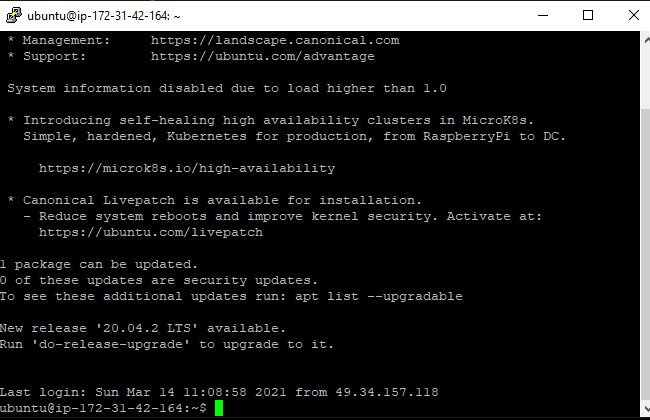
1. **Now copy this file in our slave machine.**
2. **scp agent.jar ubuntu@private address of Test server machine:/home/ubuntu**
3. **now we are using scp agent.jar ubuntu@172.31.41.145:/home/ubuntu**



1. **now check the Test Server.**

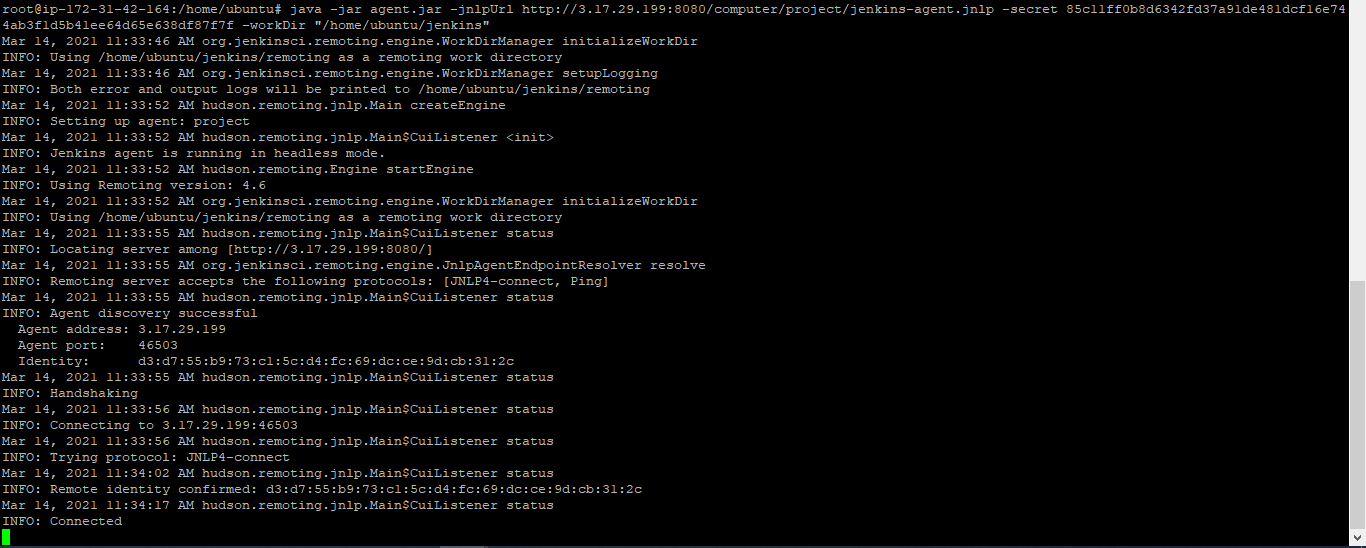


1. **Now, copy java.jar command we have run this command on slave/ Test server machine to establish the connection**
2. **First open new terminal and connect to slave machine.**

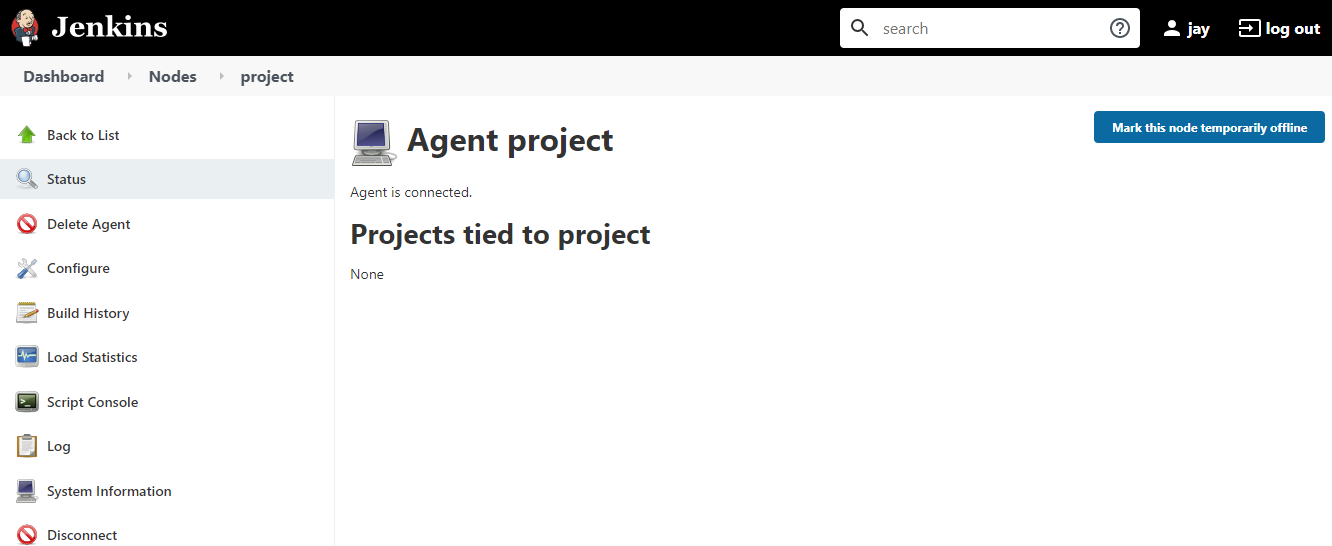


1. **Now copy the java.jar command from jenkins to Master machine**

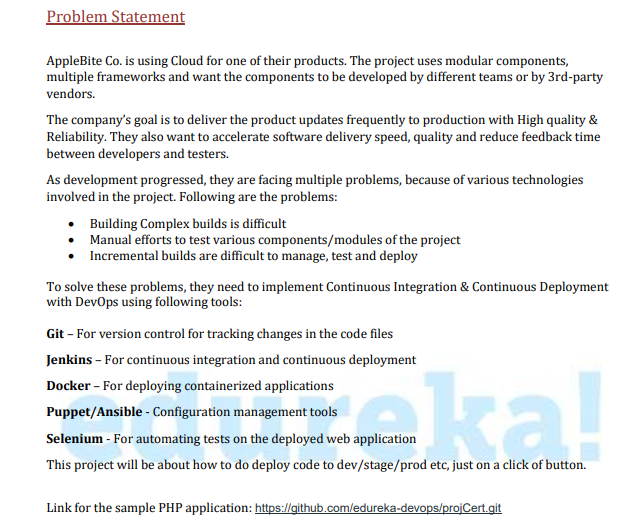
java -jar [agent.jar](http://3.17.29.199:8080/jnlpJars/agent.jar) -jnlpUrl http://3.17.29.199:8080/computer/project/jenkins-agent.jnlp -secret 85c11ff0b8d6342fd37a91de481dcf16e744ab3f1d5b41ee64d65e638df87f7f -workDir "/home/ubuntu/jenkins"



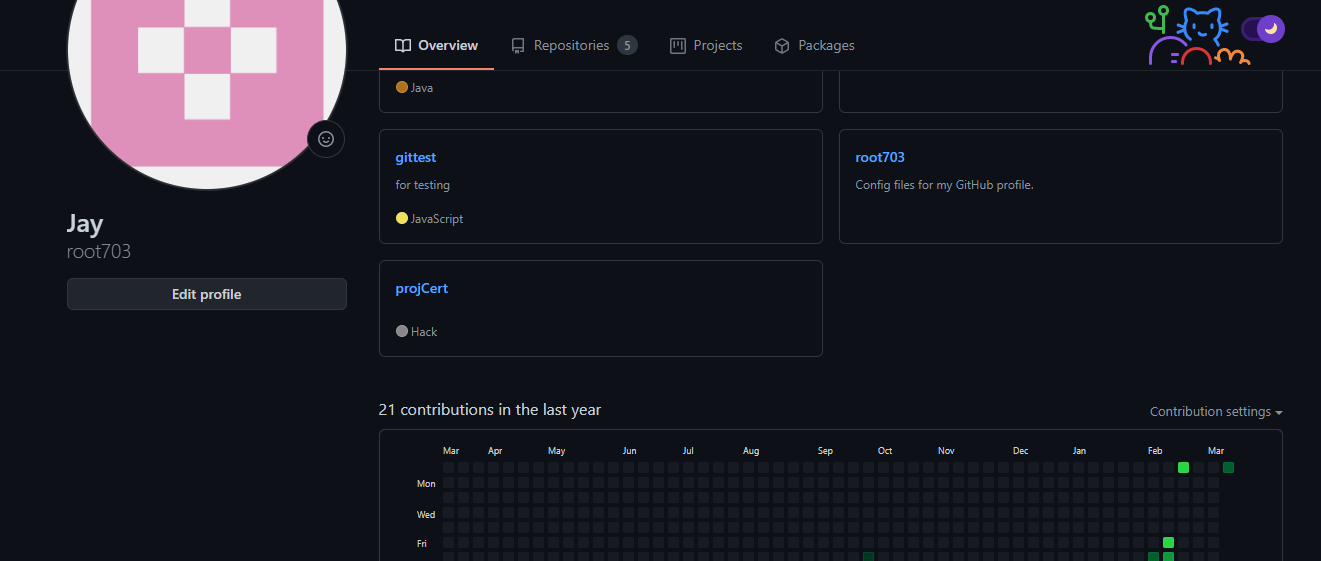
1. **Its shows connected.**
2. **Click on jenkins.**
3. **Refresh the page.**
4. **Agent project is connected.**

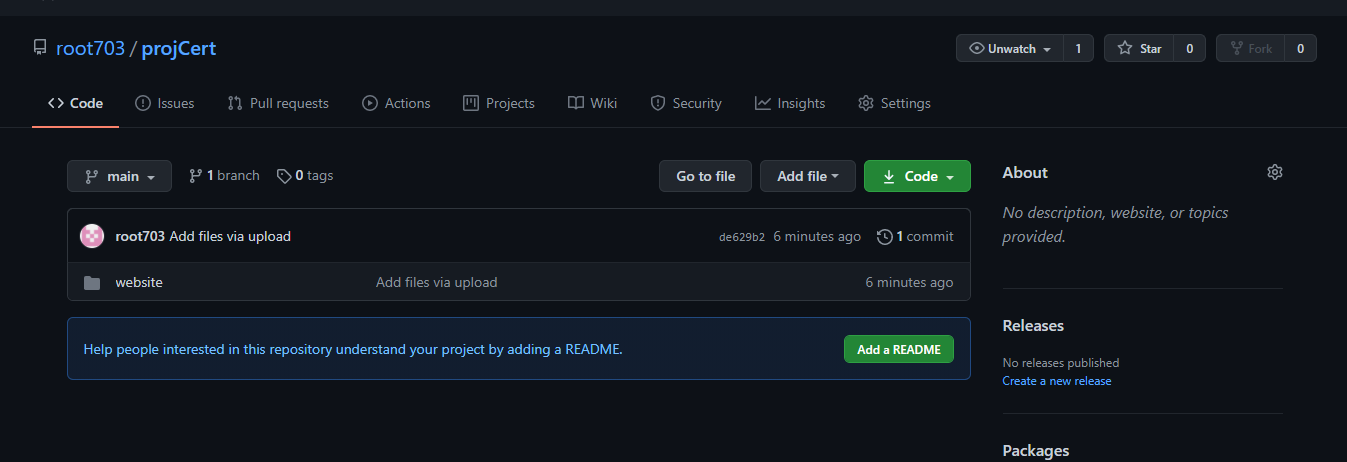


1. **We are created to jenkins slave and connected it.**
2. **Next, we have to dockerize the our php application.**
3. **We have to create docker file. And we should the content of php application on to that image.**
4. **Open the problem statement of project.**



1. **Click on the same PHP application link: -** [https://github.com/root703/projCert](https://github.com/root703/projCert.git)
2. **Repositories has been created under my Git account with same content.**





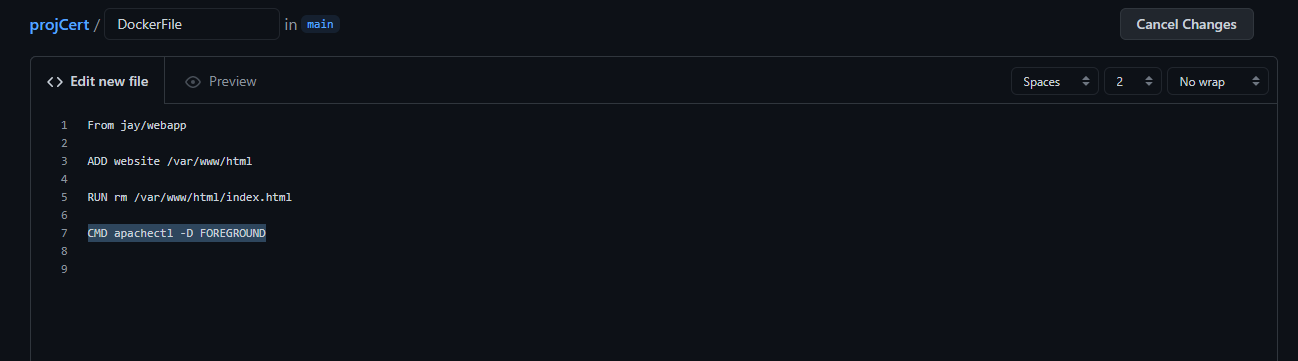
1. **Docker has been created.**

**FROM devopsedu/webapp**

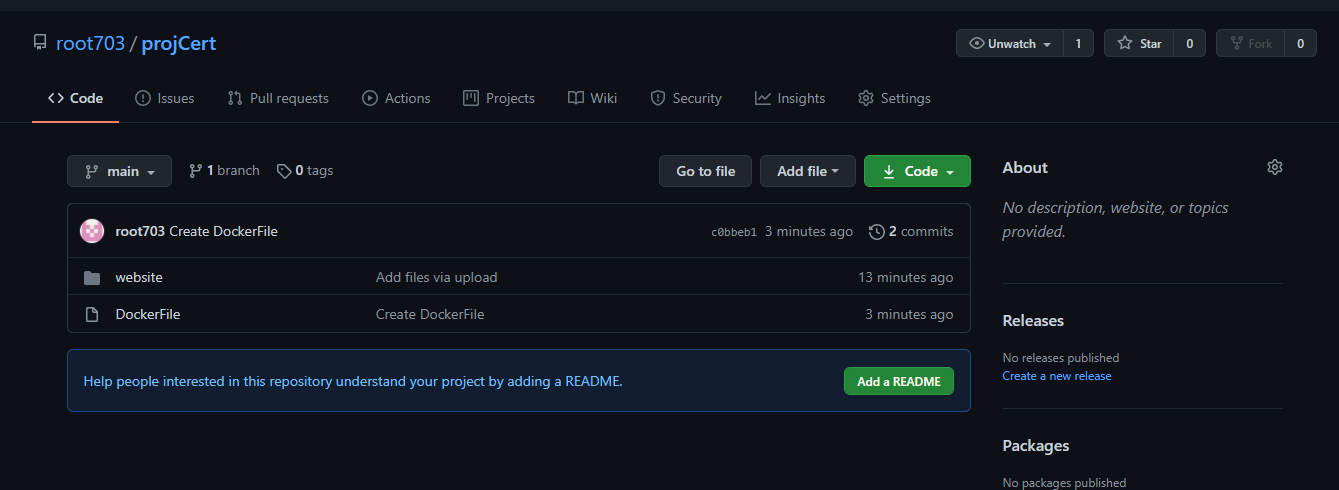
**ADD website /var/www/html**

**RUN rm /var/www/html/index.html**

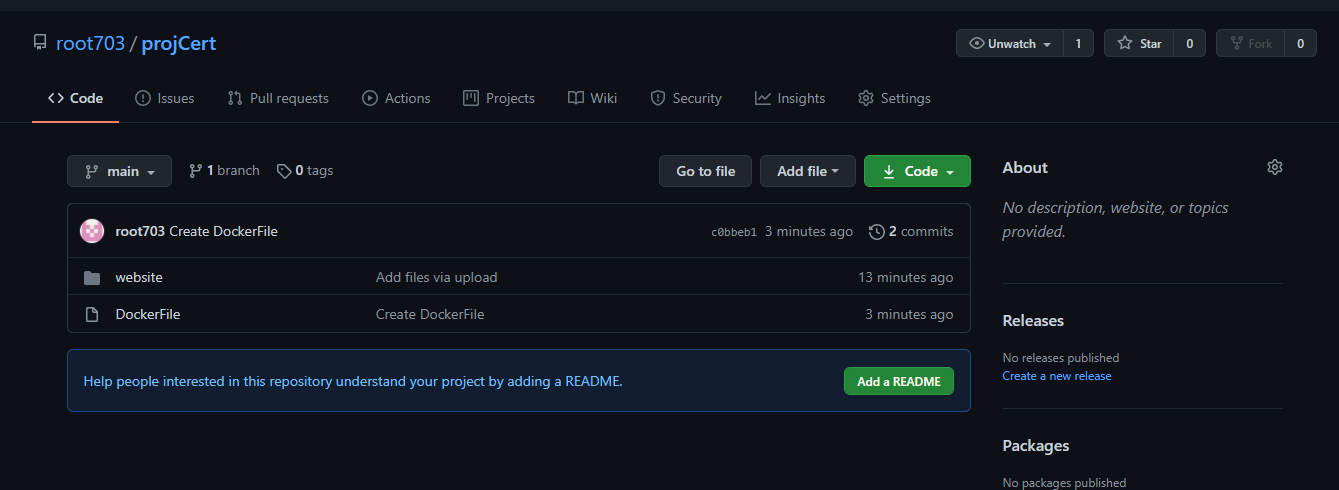
**CMD apachectl -D FOREGROUND**



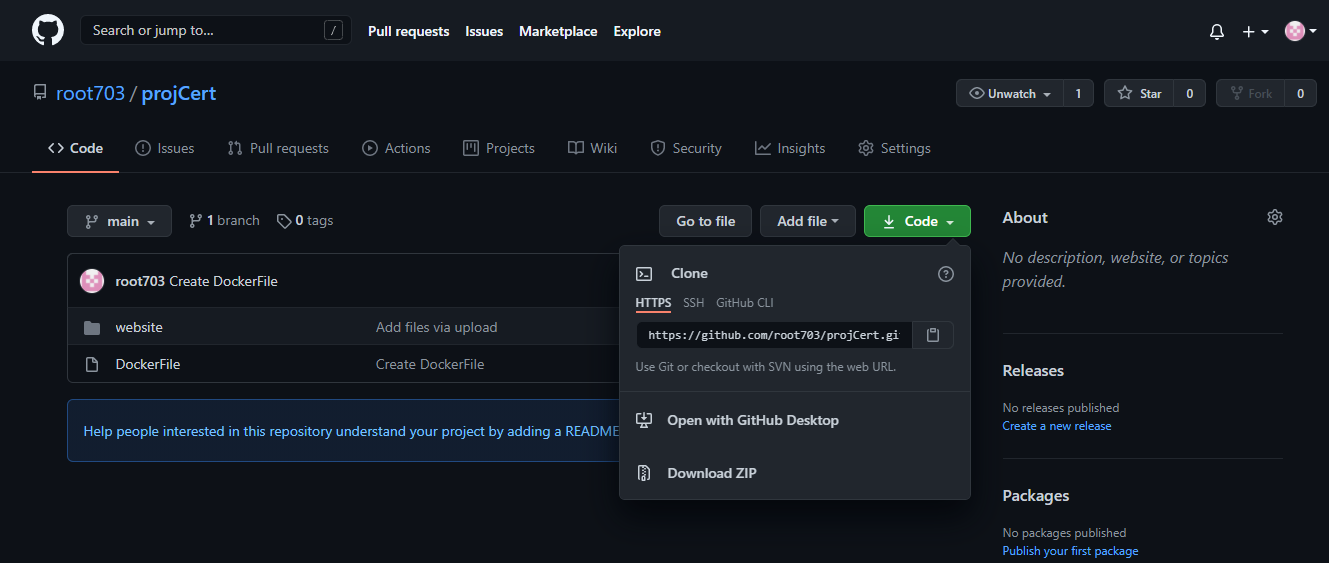
1. **Then commit new file it.**



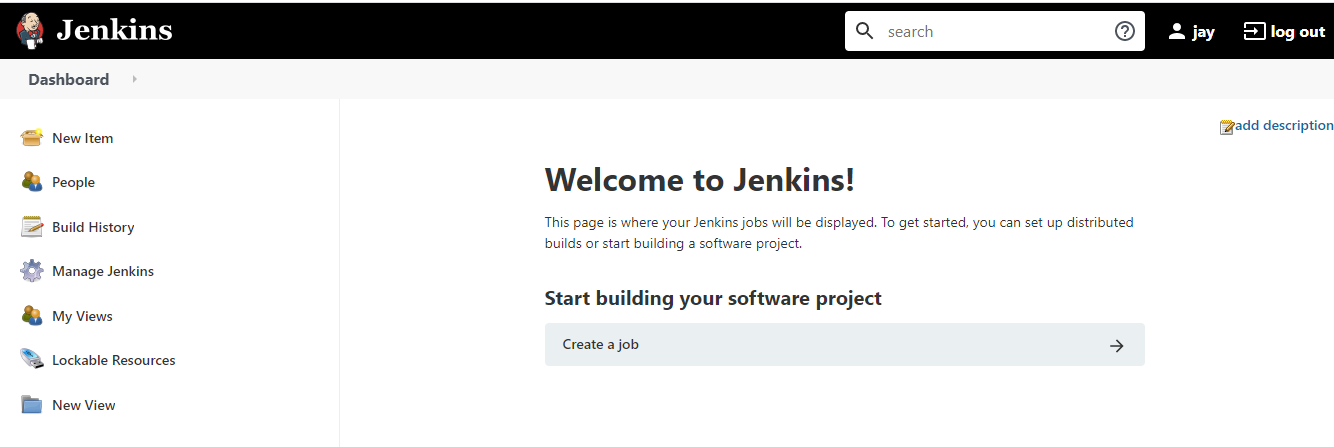
1. **Click on projCert.**



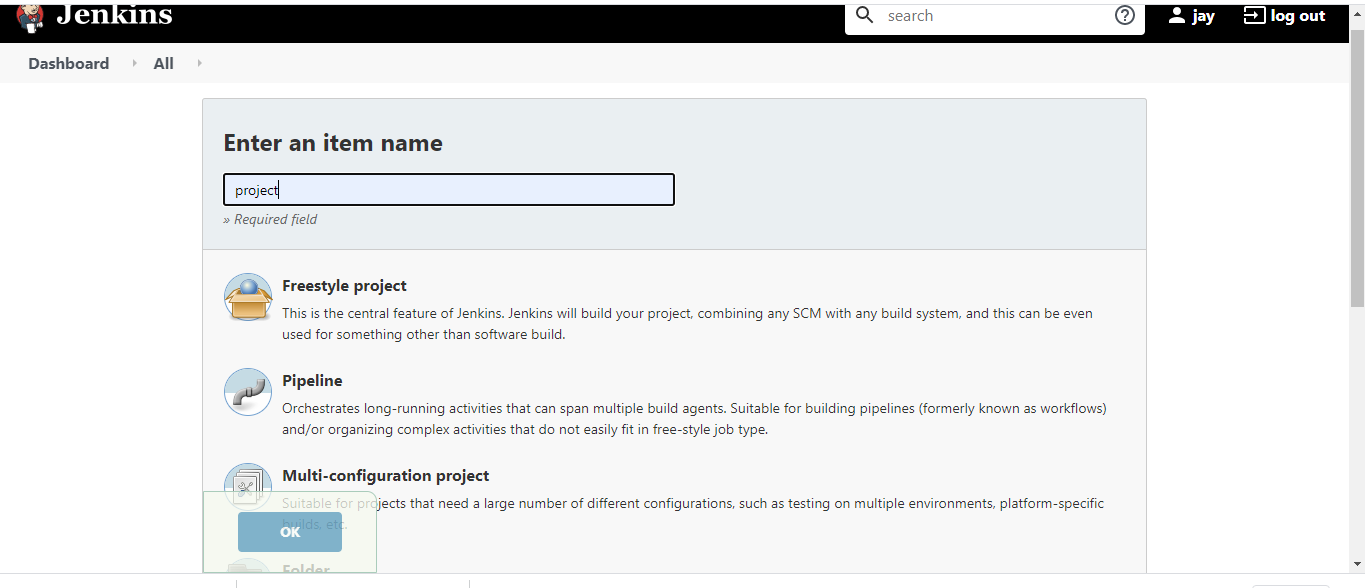
1. **Click on Code or clone / download.**



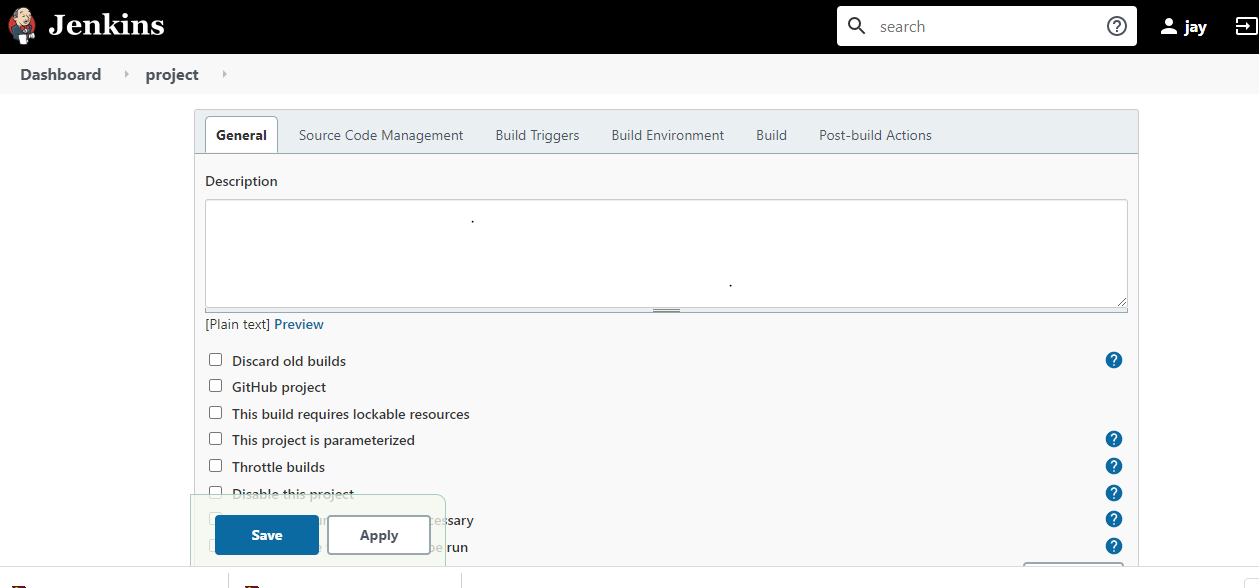
1. **Copy the url-** [**https://github.com/root703/projCert.git**](https://github.com/root703/projCert.git)
2. **Now go to jenkins , we have created jenkins job.**
3. **Job should run on jenkins slave machine**
4. **We have to build this docker file on the slave/Test Server machine then run the docker image which will be created.**
5. **Click on new Item.**



1. **Enter name and select the Free style project and click on OK.**



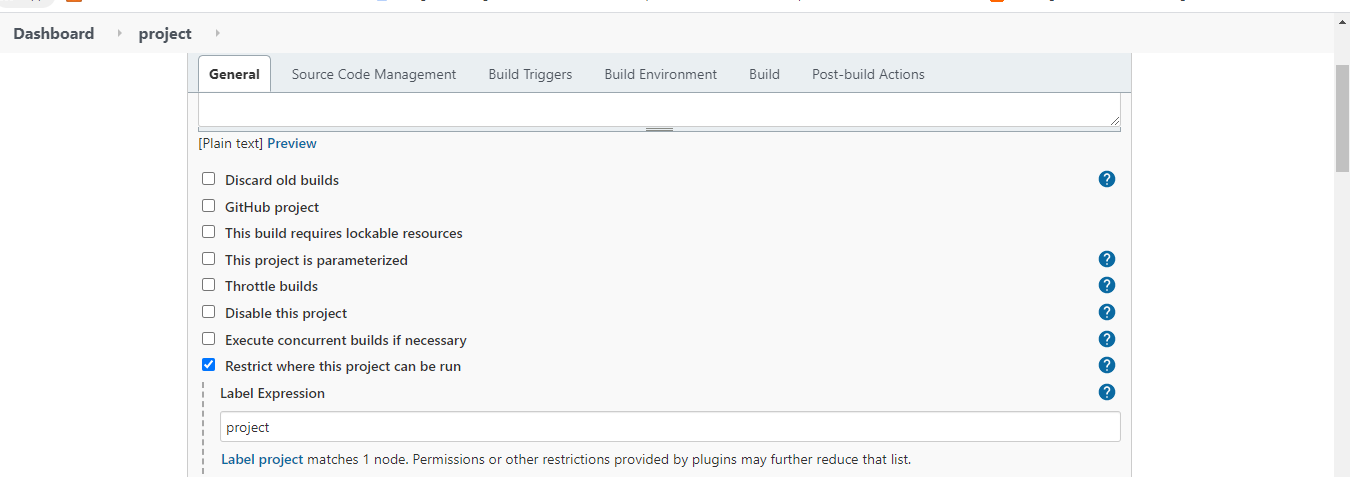
1. **Then we get general setting option**



1. **Select the**

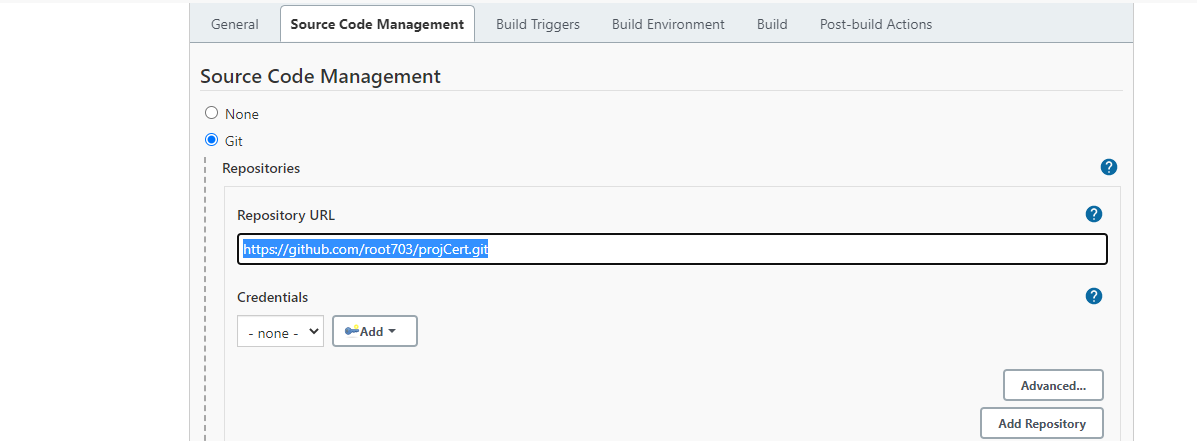
**Restrict where this project can be run**

**Label expression --project**



1. **source code management - select Git and add the Repository URL.**

**https://github.com/root703/projCert.git**



1. **Then Build, add build step select Execute shell.**



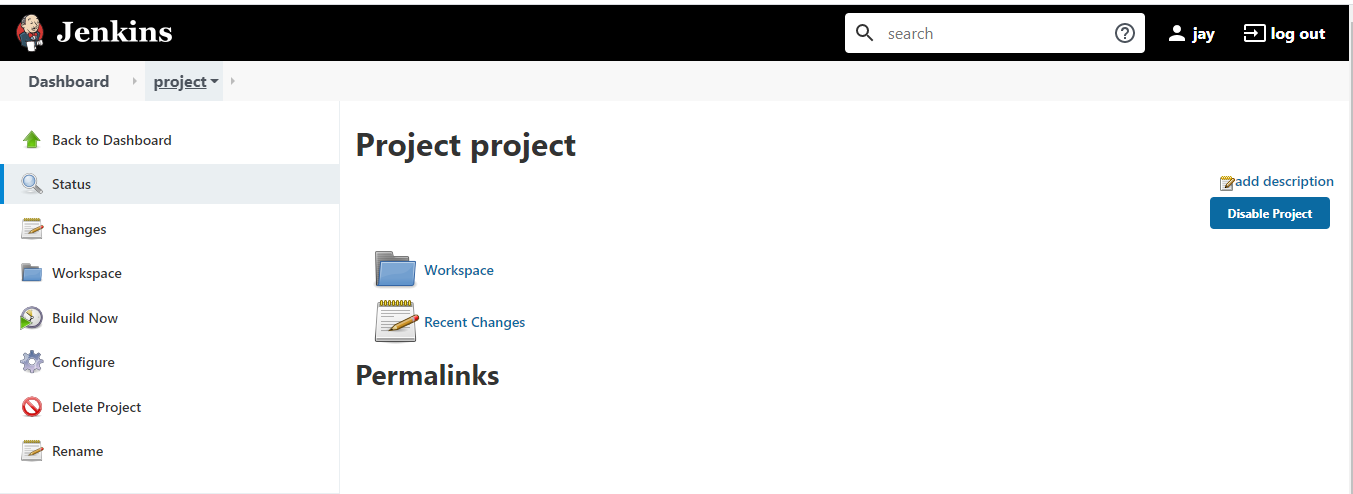
1. **We have run shell command to build the docker file and then to run docker image which will be created.**
2. **Click and type -**

**sudo docker build -t phpapplication .**

**sudo docker run -itd -p 8081:80 phpapplication**

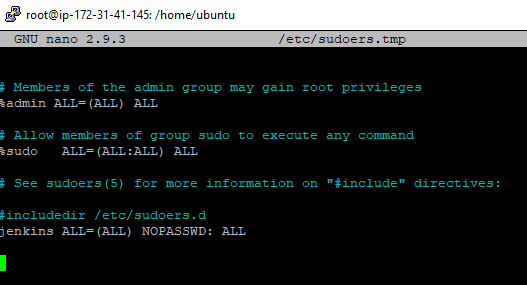
1. **Then apply and save**



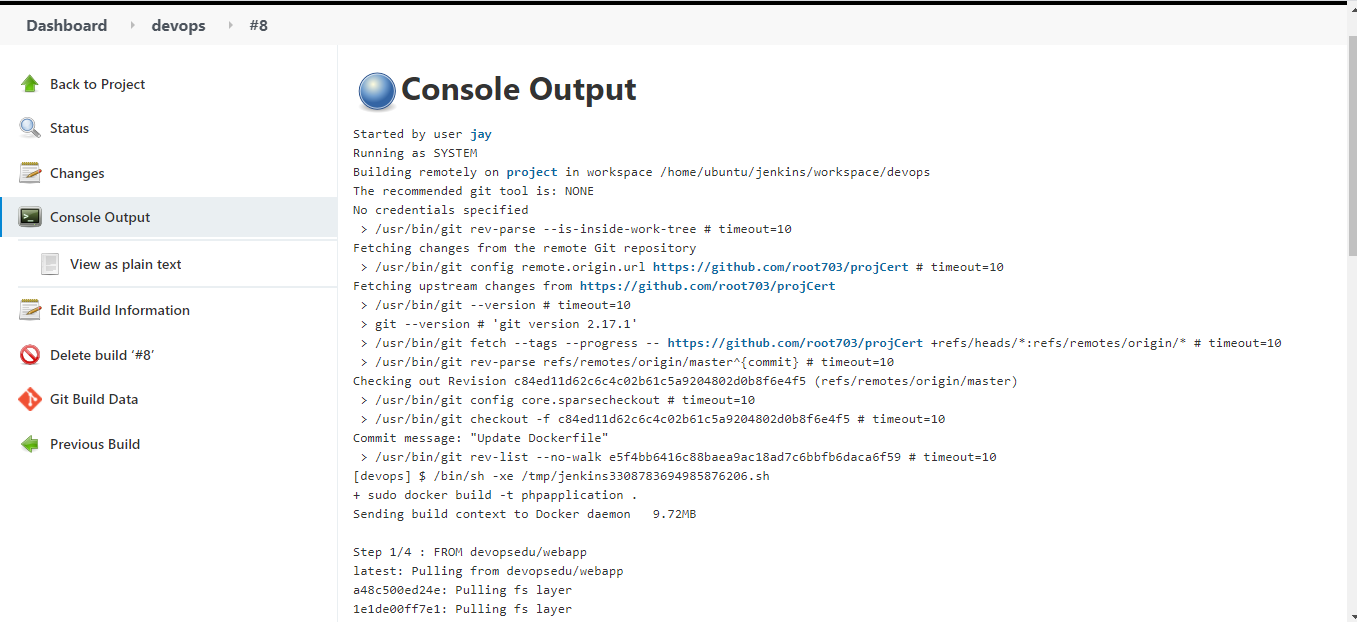


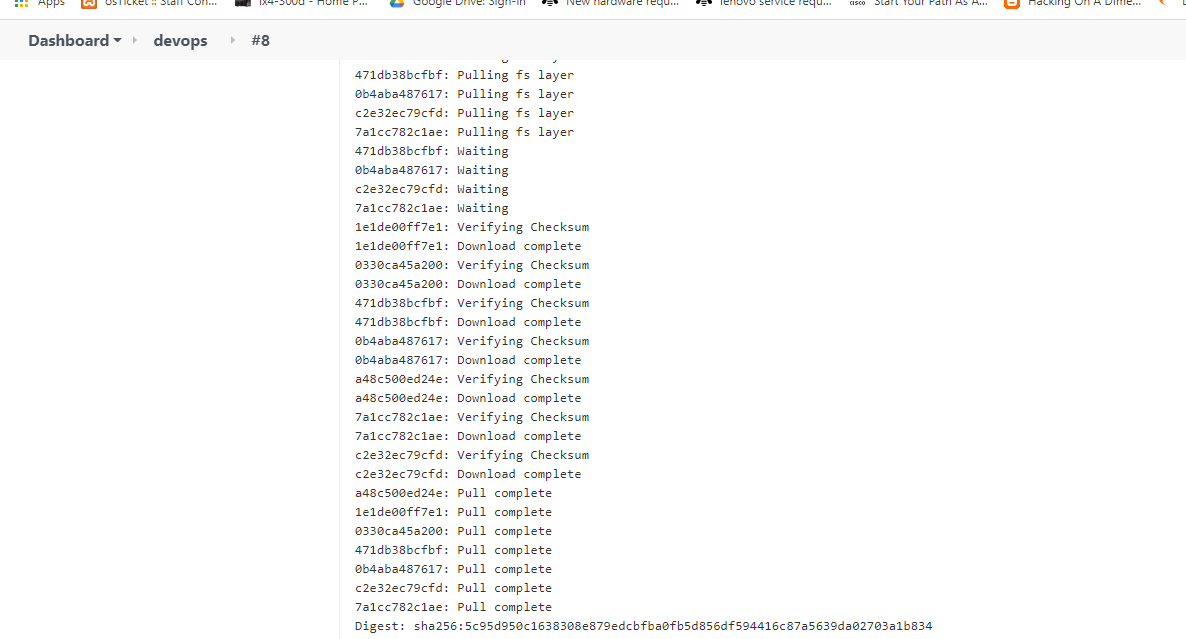
1. **Open the Slave /Test server Machine.**
2. **Type – sudo visudo**

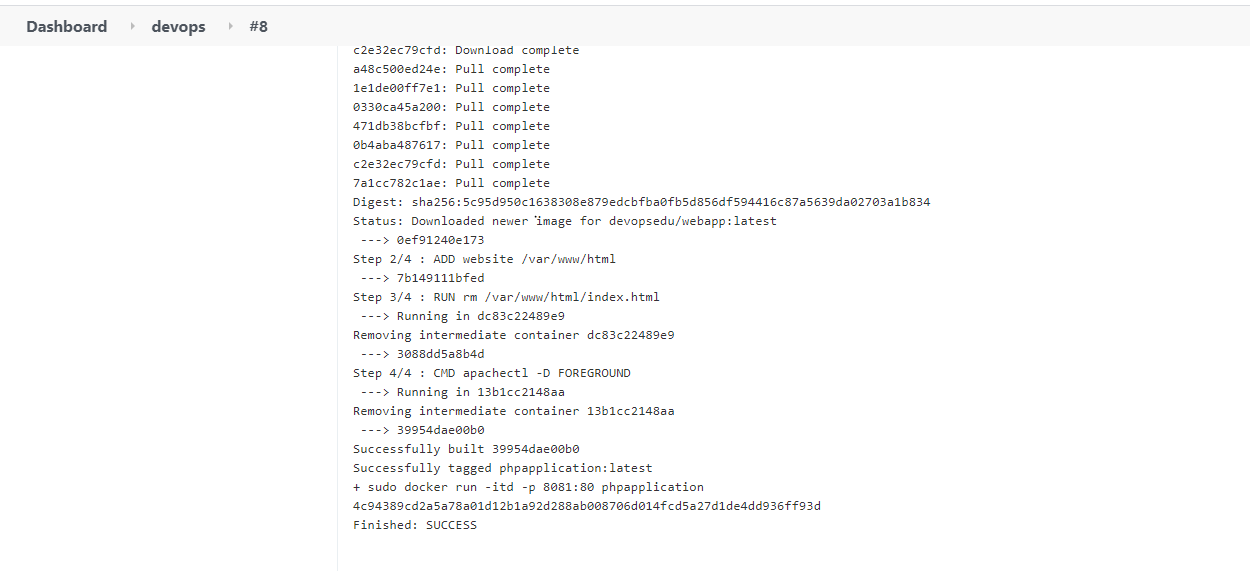
**Go to last and enter the – jenkins ALL=(ALL) NOPASSWD: ALL**



1. **Go to jenkins.**
2. **Then Build now.**

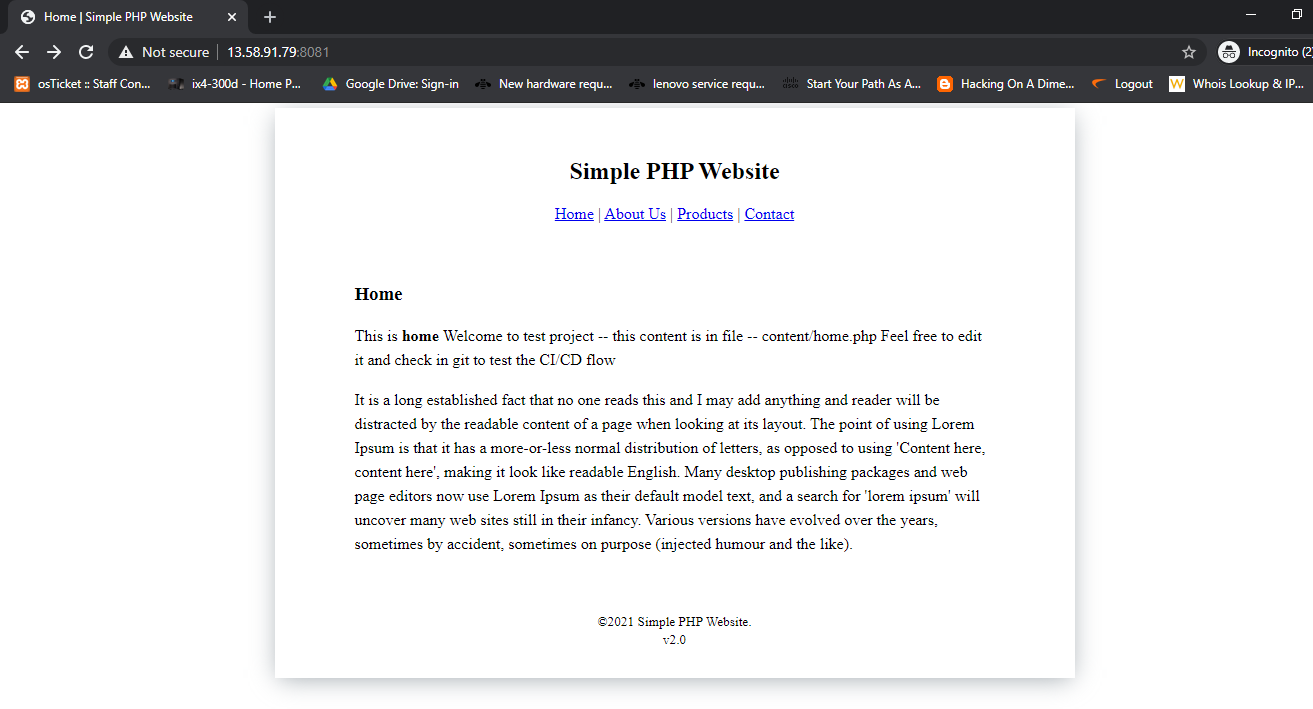






1. **its successful.**
2. **Open the browser and we checked website is live.**

**url: - http://13.58.91.79:8081/**



1. **Now PHP application is live.**
2. **Project deployment successfully complete.**

**Prepare by – Suhail Hameed**