Capstone Projects - 1

You have been Hired Sr. DevOps Engineer in Abode Software. They want to implement DevOpsLifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company, their product is available on this GitHub link

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

Following are the specifications of the lifecycle:

1. Install the necessary software on the machines using a configuration management tool.

2. Git Workflow has to be implemented

3.Code Build should automatically be triggered once commit is made to

master branch or develop branch.

If commit is made to master branch, test and push to prod

If commit is made to develop branch, just test the product, do not push to prod

4.The Code should be containerized with the help of a Dockerfile. The Dockerfile should bebuilt every time there is a push to Git-Hub. Use the

following pre-built container for your application:

hshar/webapp

The code should reside in '/var/www/html'

5.The above tasks should be defined in a Jenkins Pipeline, with the

following jobs: Job1: build , Job2: test, Job3: prod.

Ans:

1. Open up the AWS Management Console

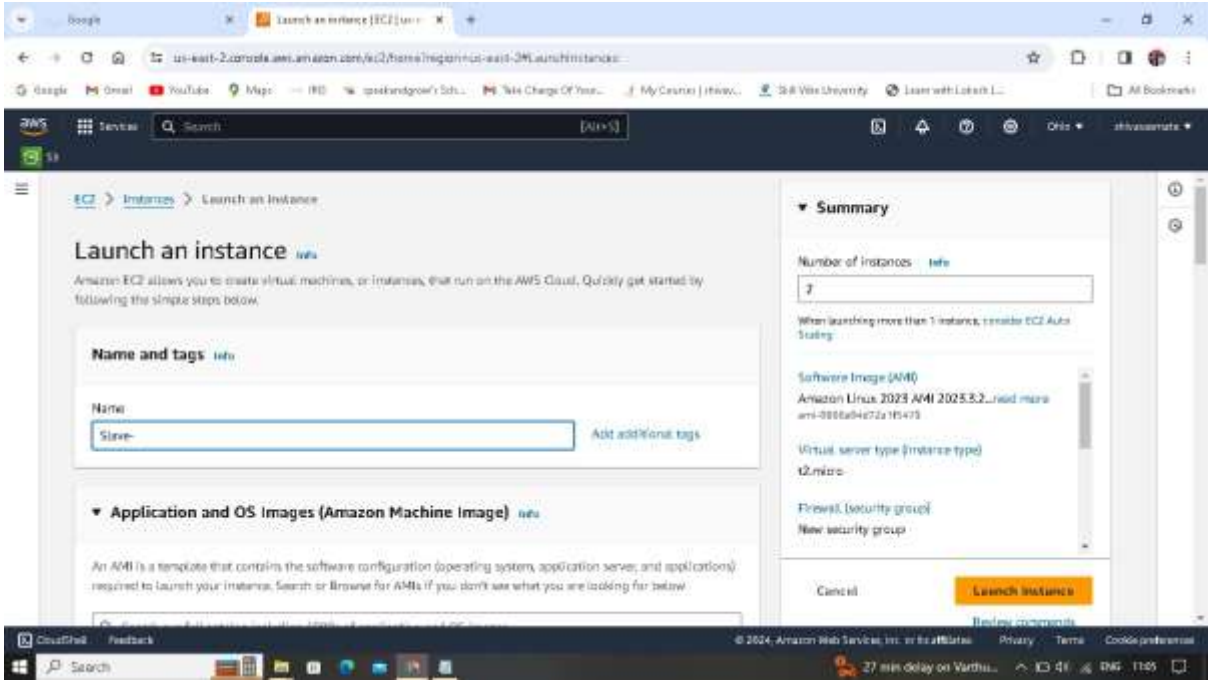
2. Check for the region [us-east-2 (Ohio)]

3. Search for EC2 in the search box

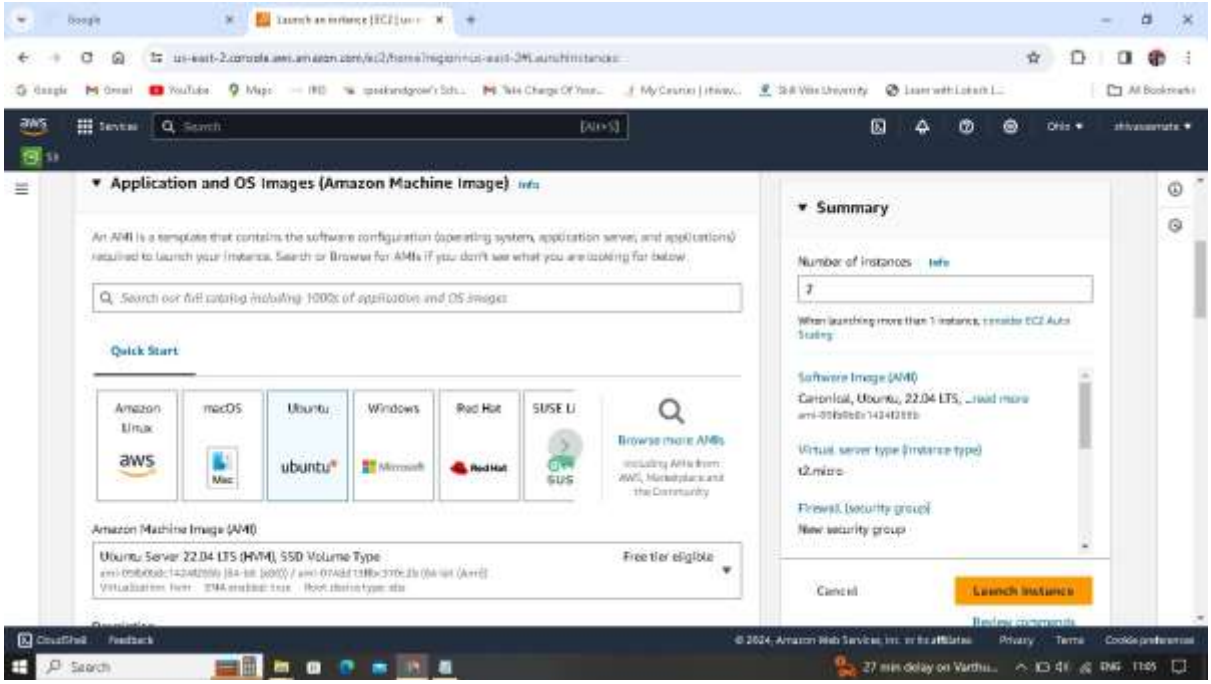
4. Click on instances to go to the EC2 console

5. Click on Launch Two Instances and setup the instance for UbuntuOS:

a. Name: Slave-

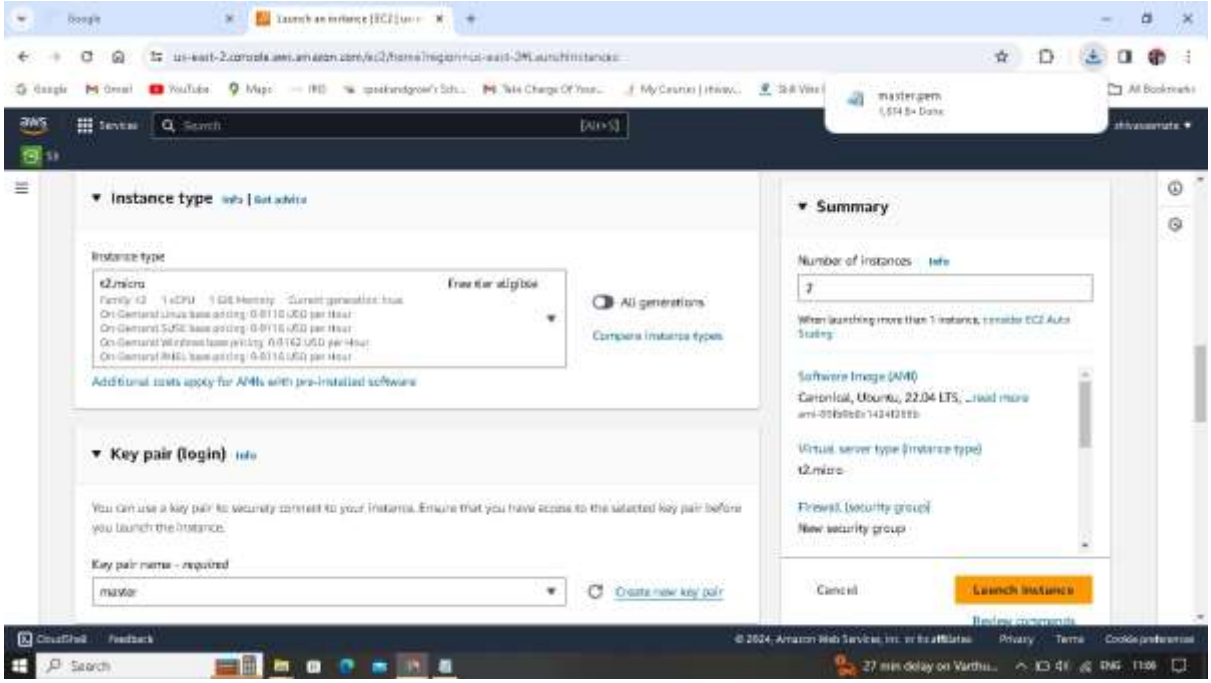


b. AMI: QuickStart >> Ubuntu [Any version which is free tier eligible]

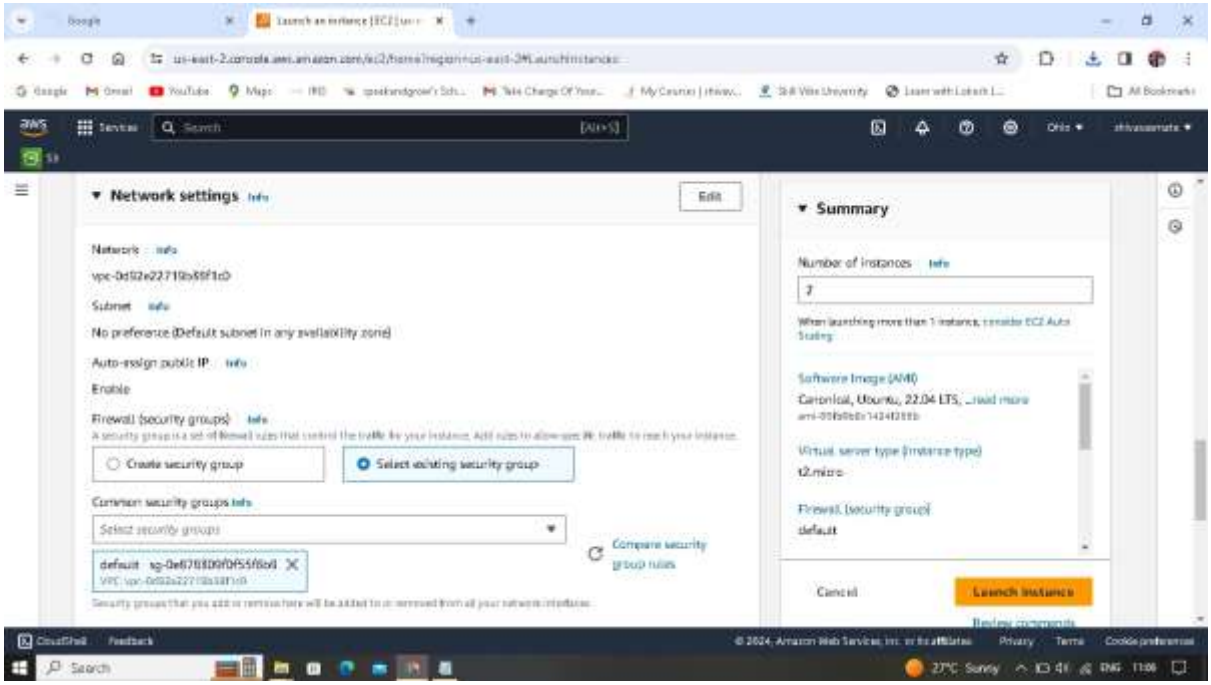


c. Instance type: t2. micro [free tier eligible]

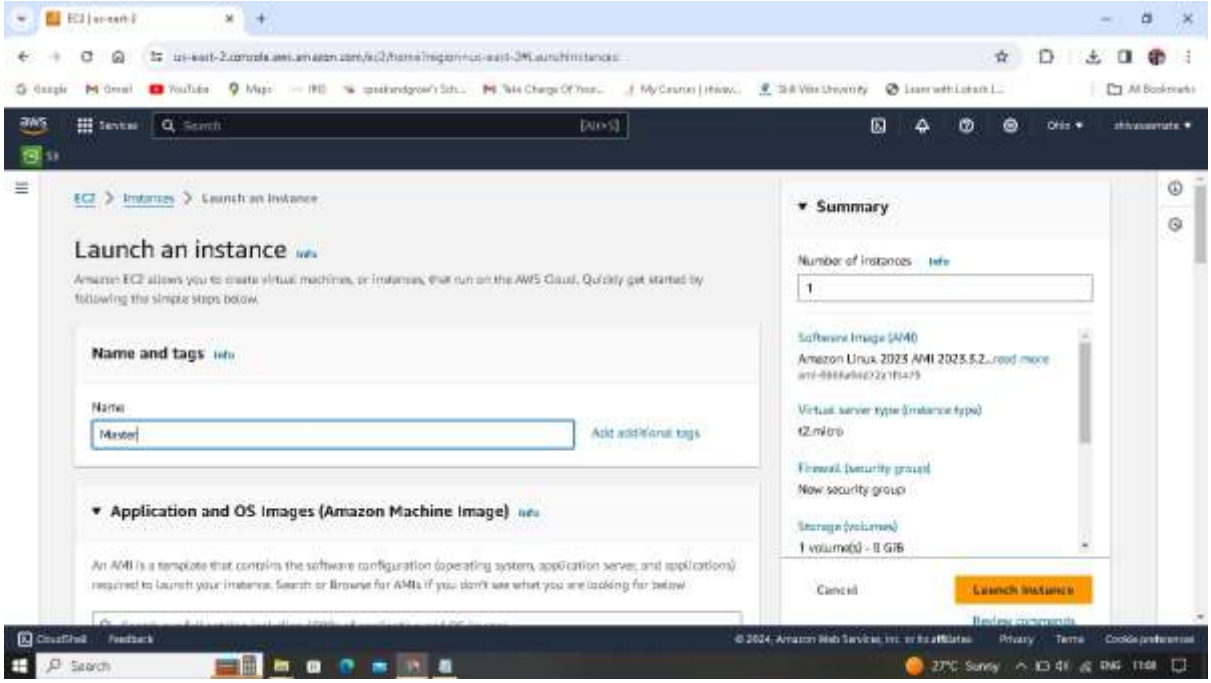
d. Key-pair: Create a key pair [rsa and .pem] with a name master



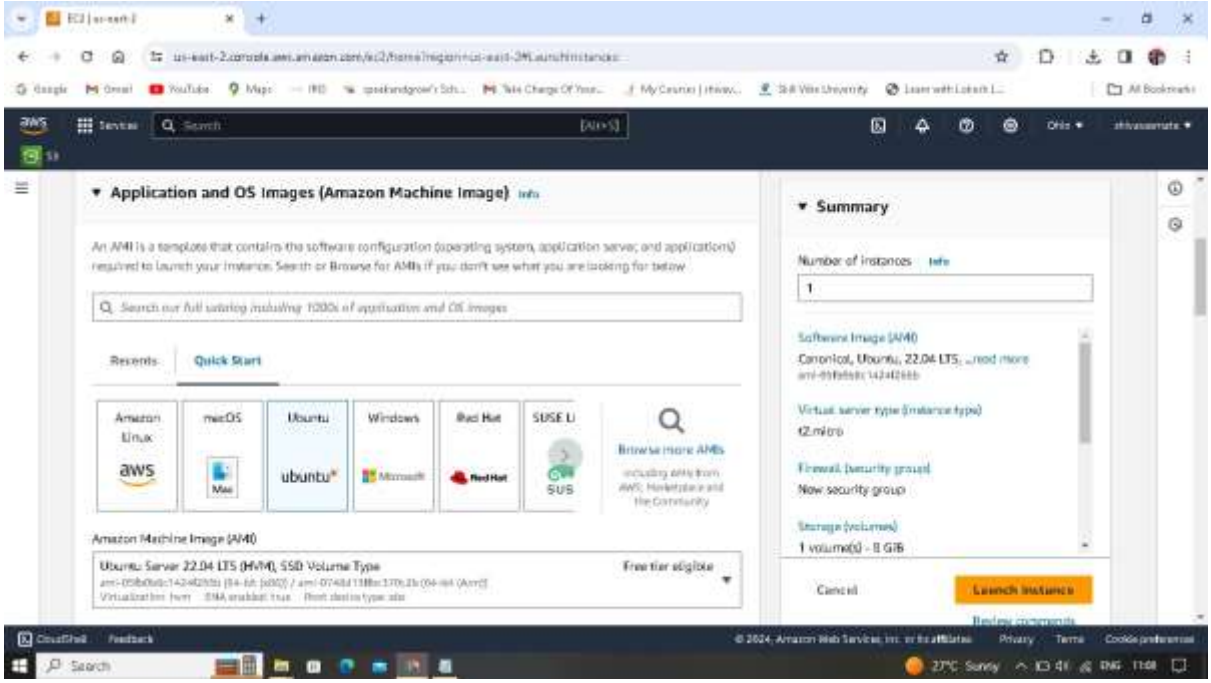
e. VPC and Security group as default with ssh and HTTP protocols click on launch instance.



6. a. One Master Instances and setup the instance for UbuntuOS

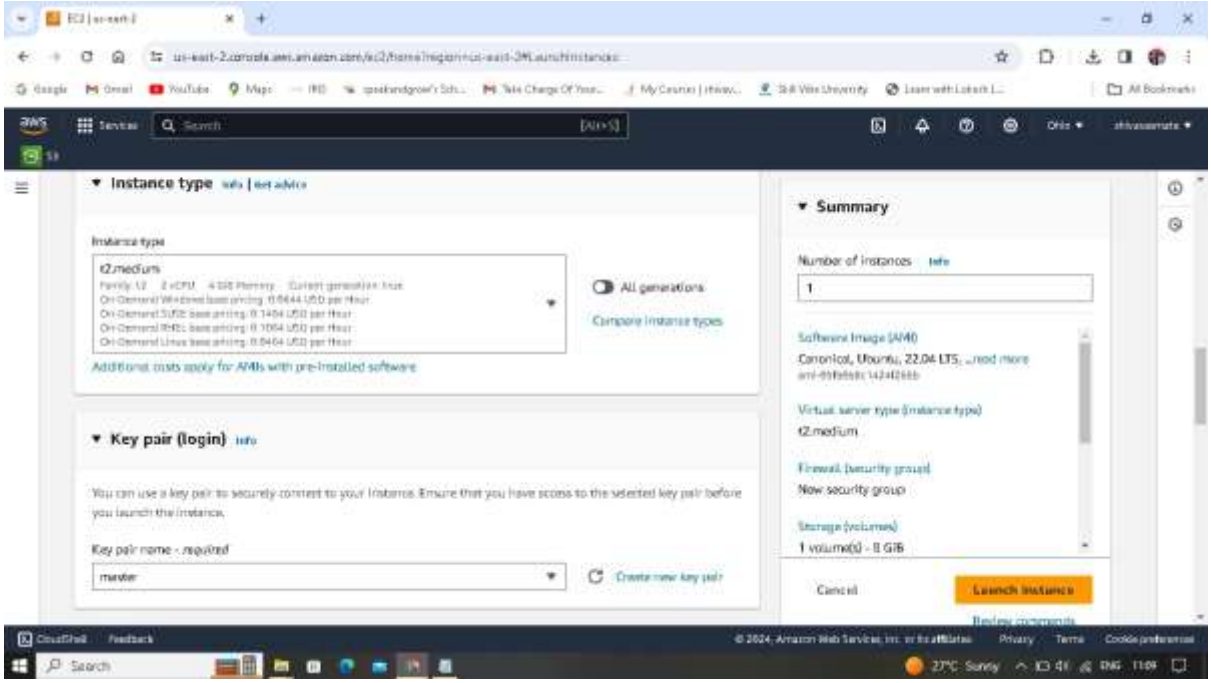


b. AMI: QuickStart >> Ubuntu [Any version which is free tier eligible]

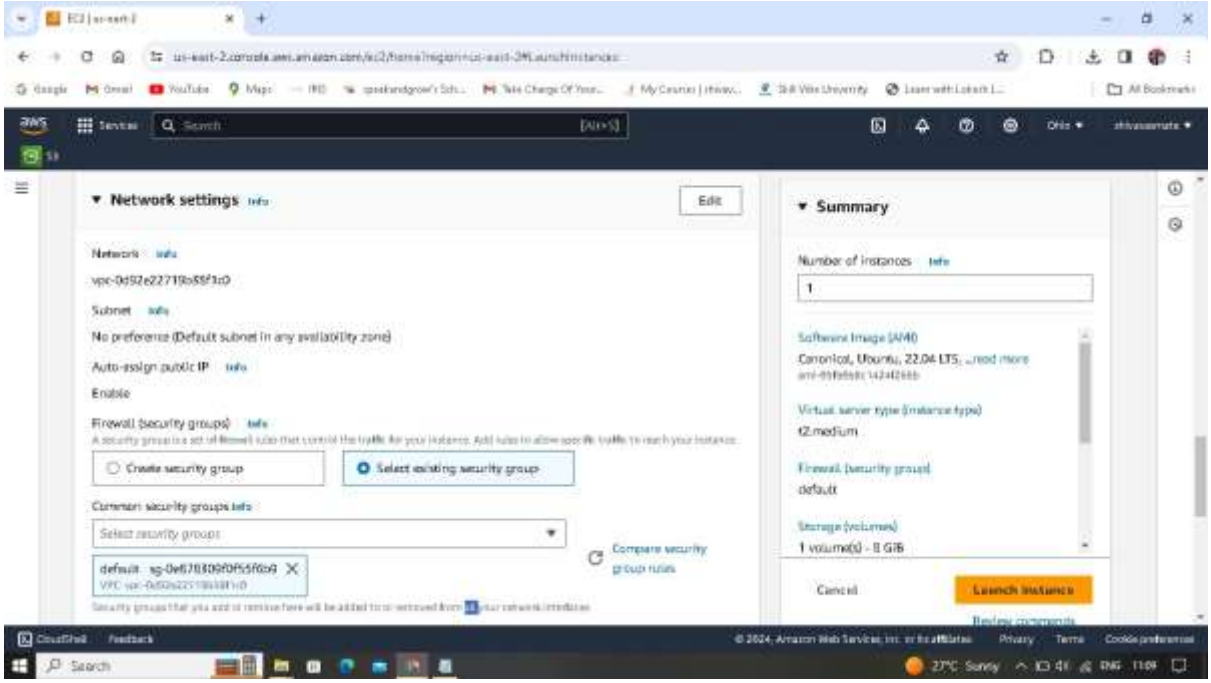


c. Instance type: t2. medium

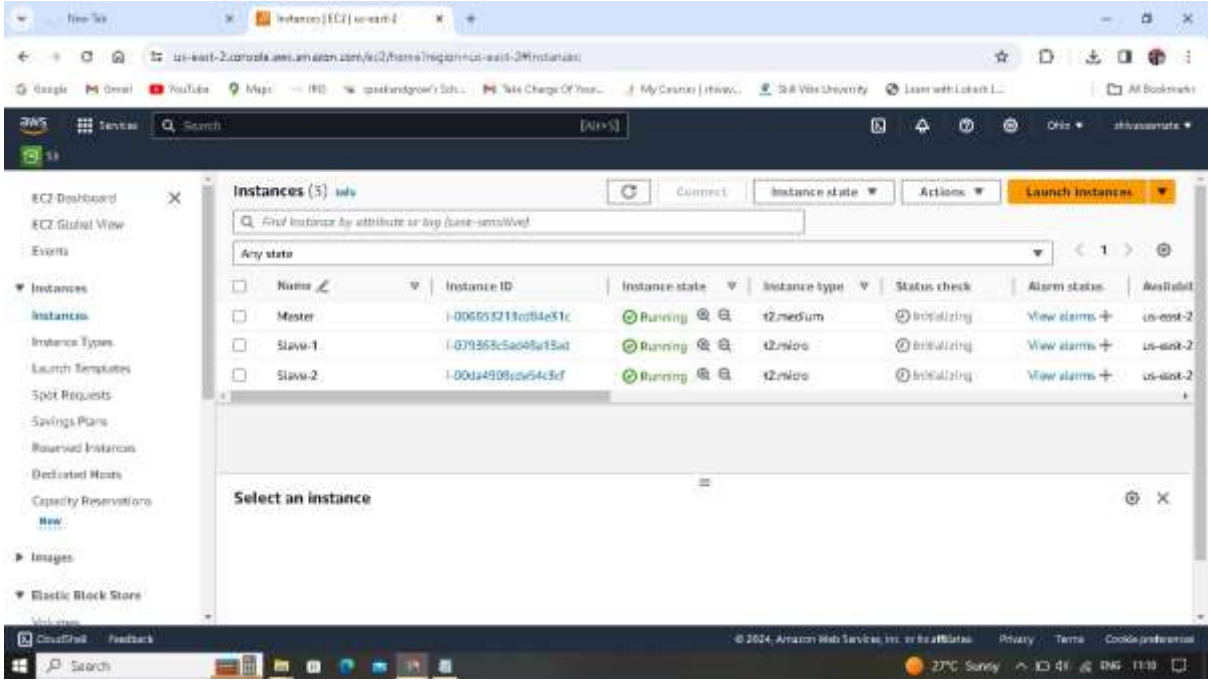
d. Key-pair: Create a key pair [rsa and .pem] with a name master.



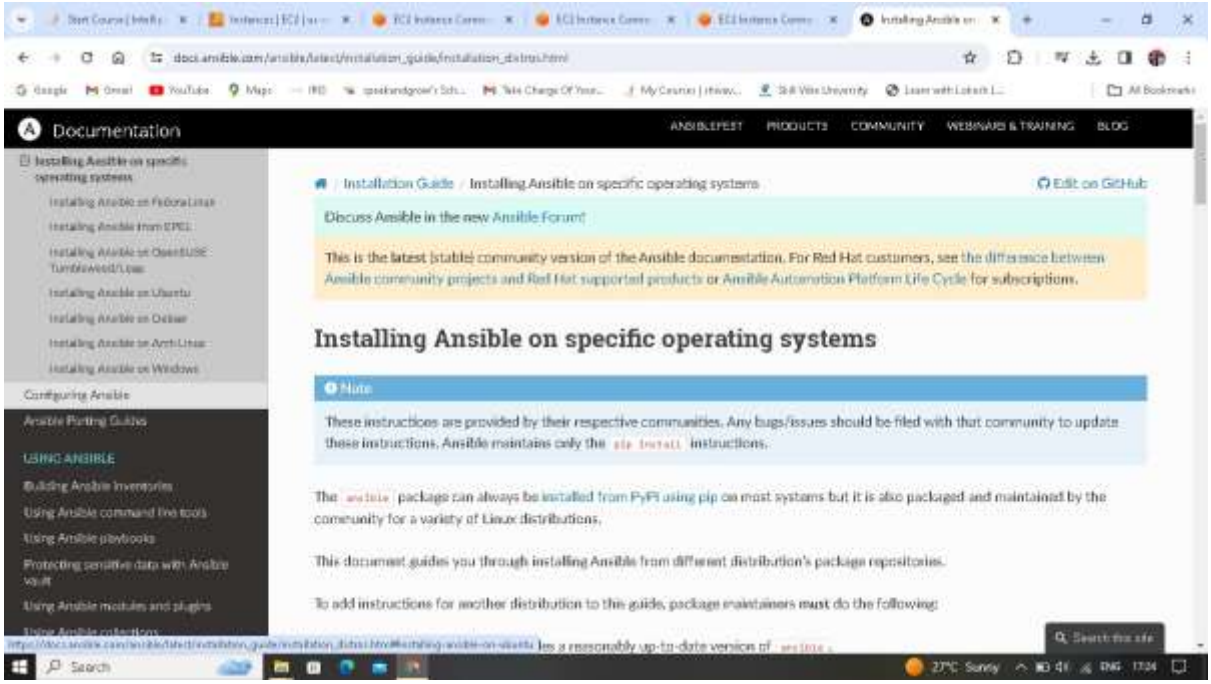
e. VPC and Security group as default with ssh and HTTP protocols click on launch instance.



7. All three instances are in running state. Now edit name as Master, Slave1 and Slave2 .



8.Now install ansible for that go to another browser search for ansible documentation to install.



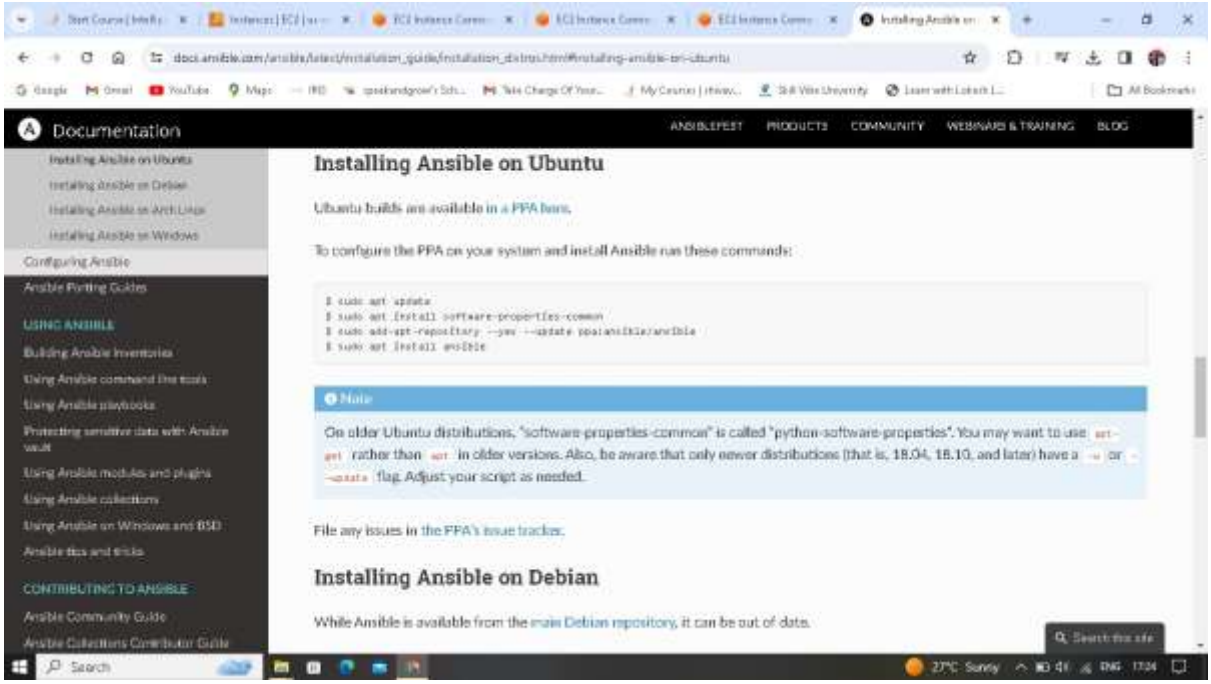
9. On left side click on the installing Ansible on Ubuntu.

$ sudo apt update

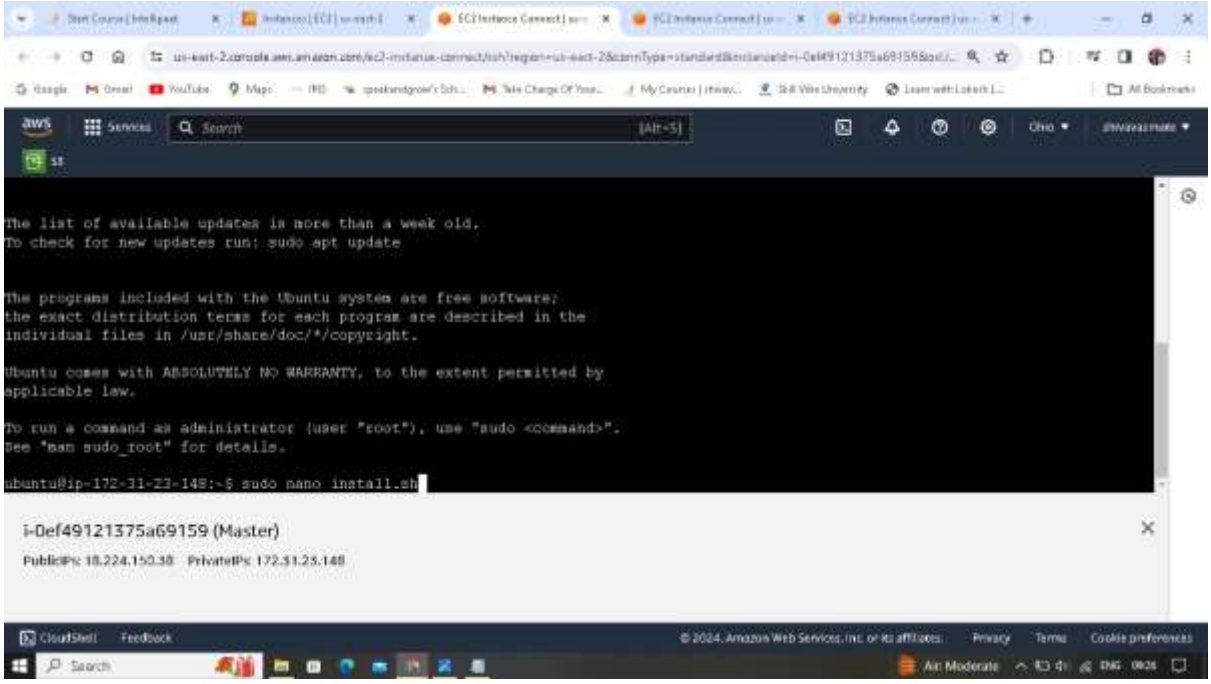
$ sudo apt install software-properties-common

$ sudo add-apt-repository --yes --update ppa:ansible/ansible

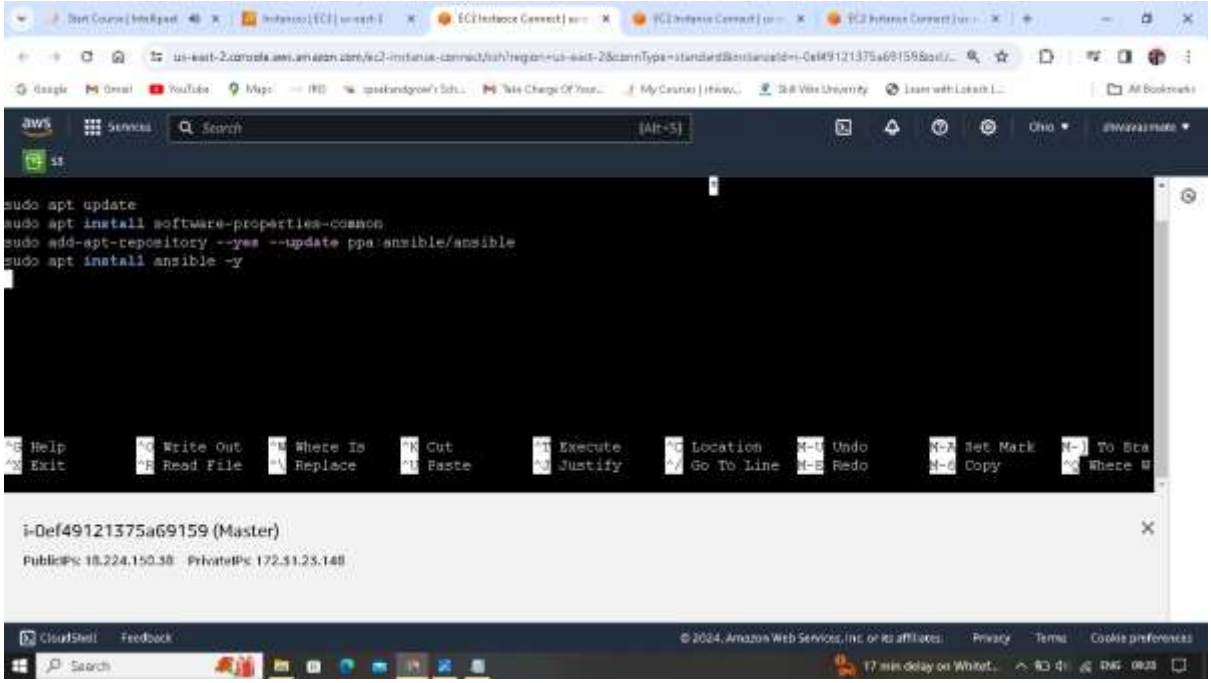
$ sudo apt install ansible



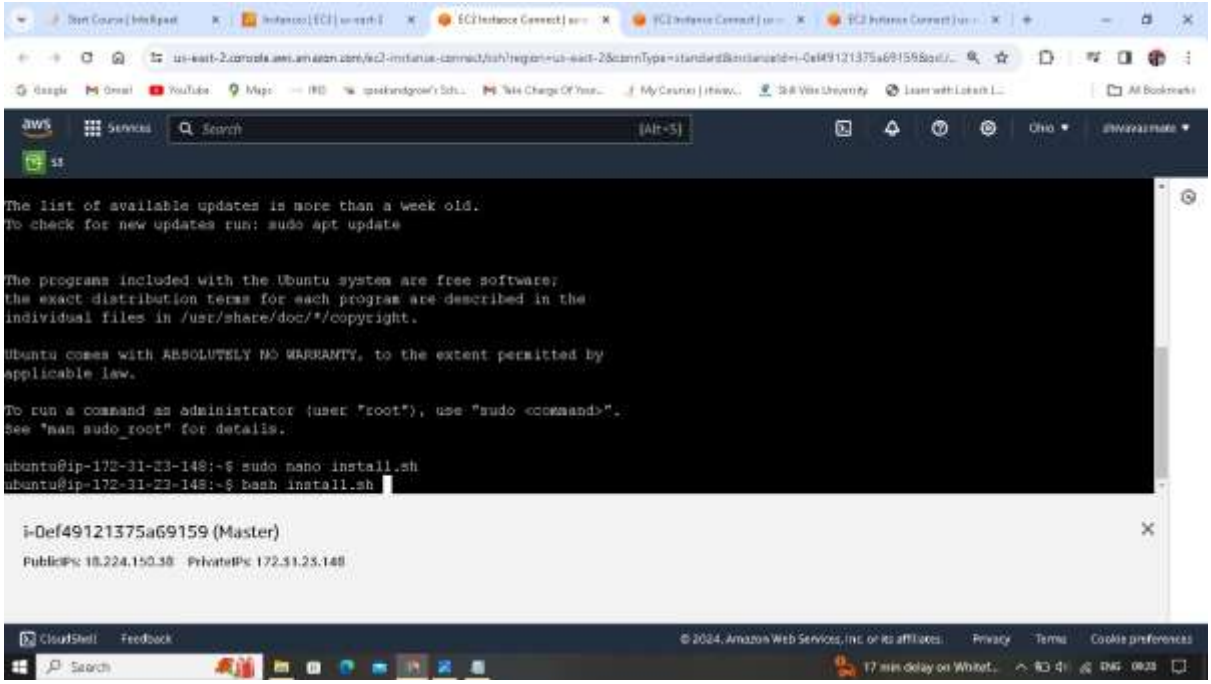
10. Open a text editor by using command sudo nano install.sh



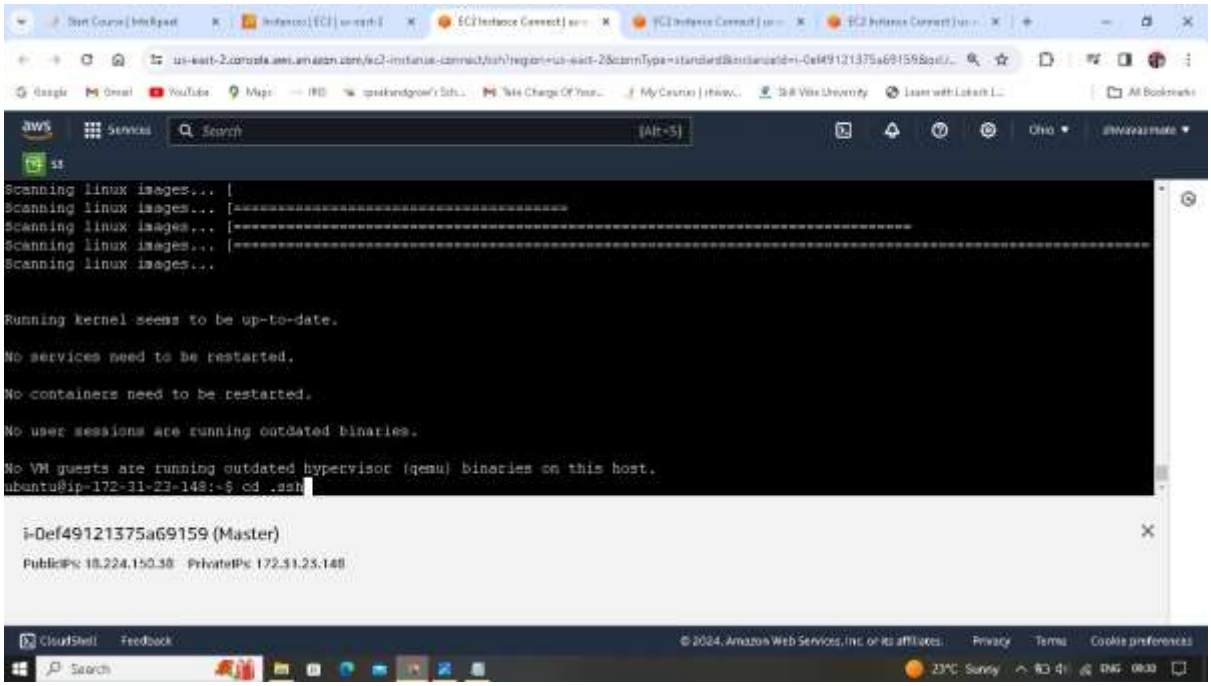
11.Copy and paste all ansible install commands in install.sh. ctrl + s to save and ctrl + x to exit.



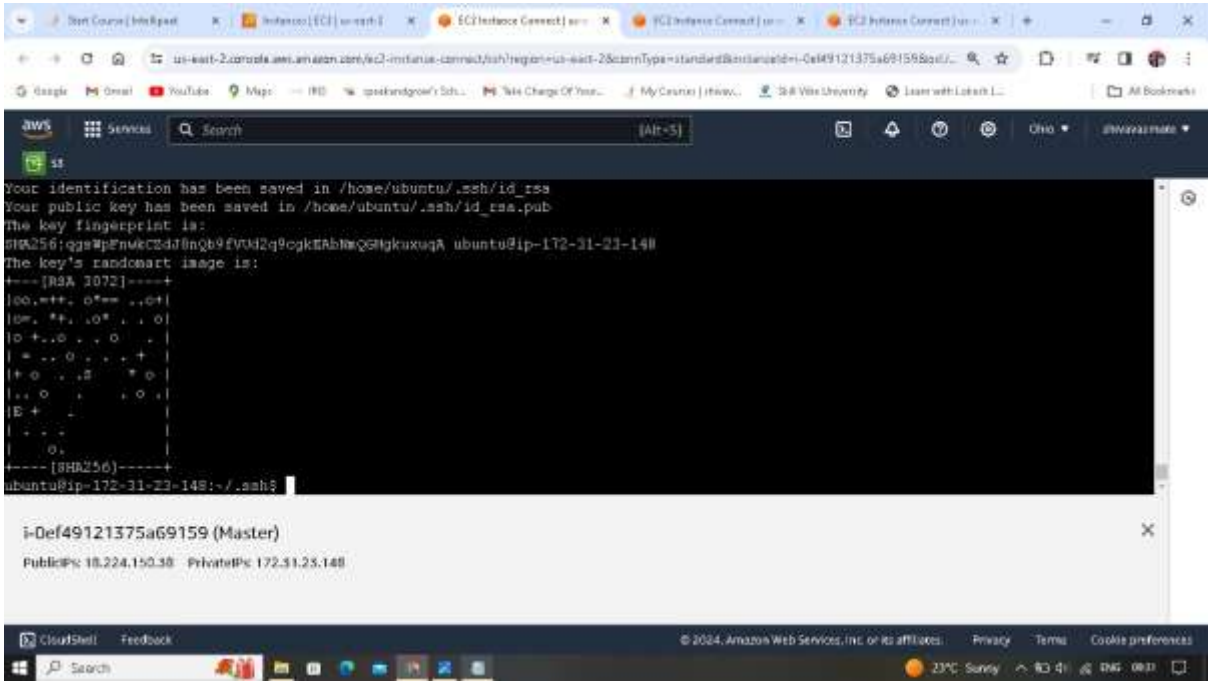
12. Now run the install.sh file with the command bash install.sh



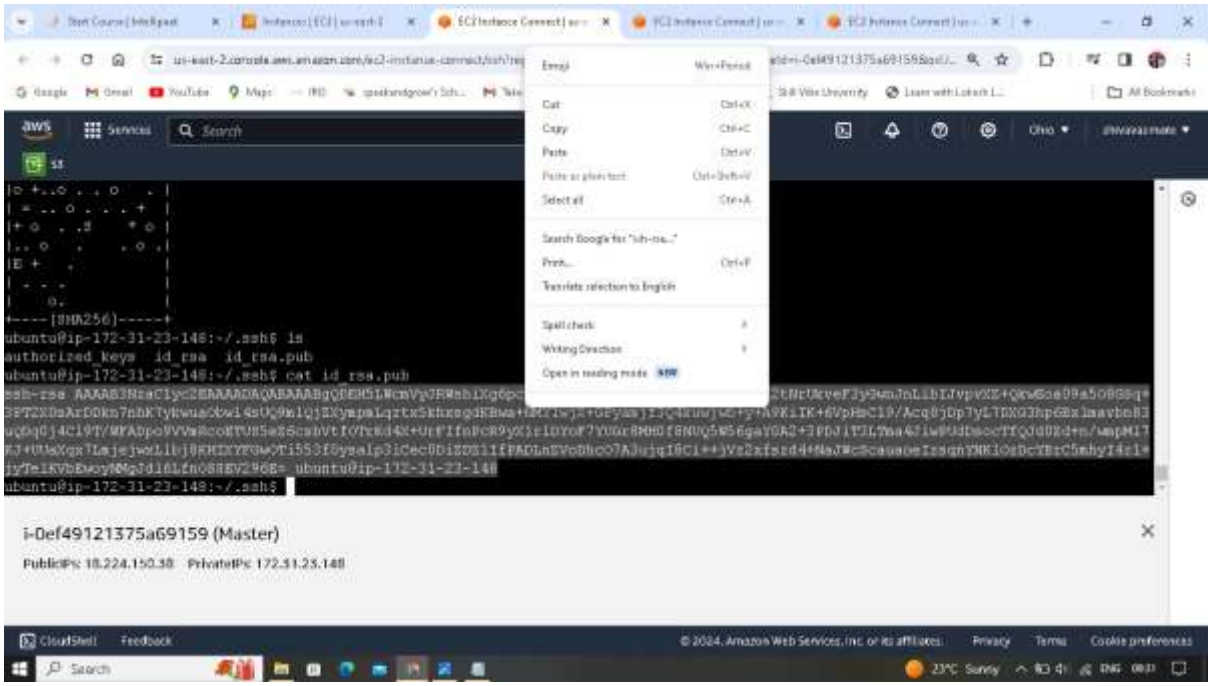
13.Go to .ssh folder by cd .ssh command



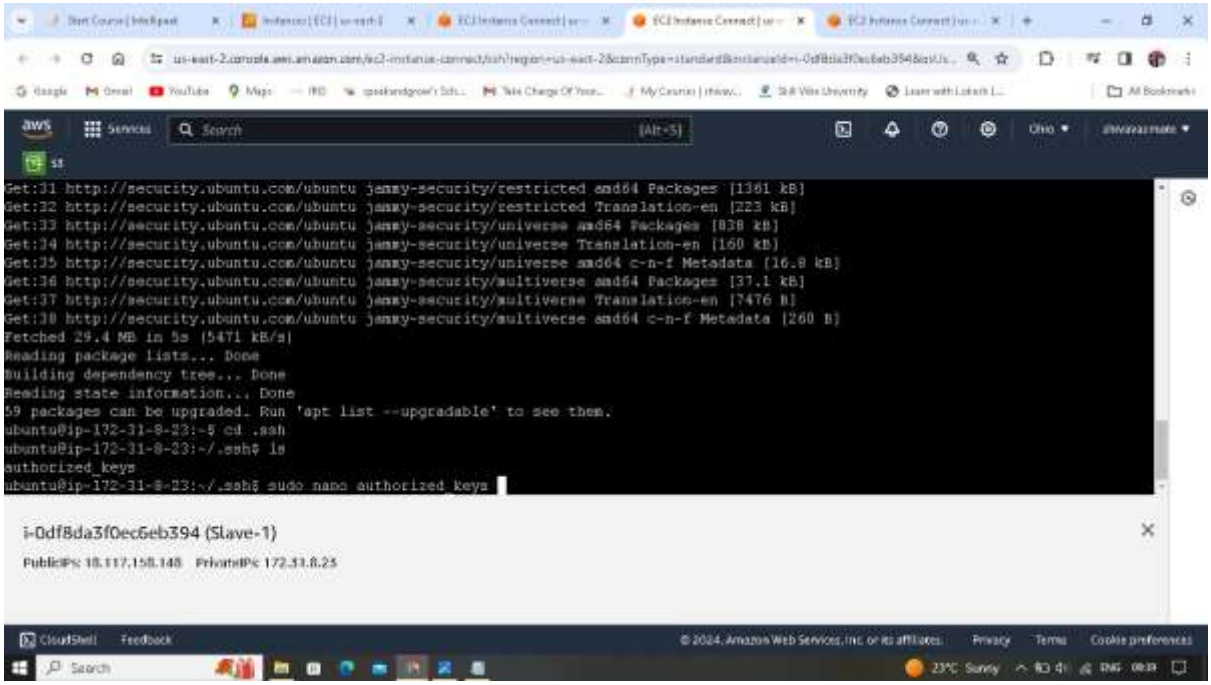
14. Generate private and public key by ssh-keygen command.



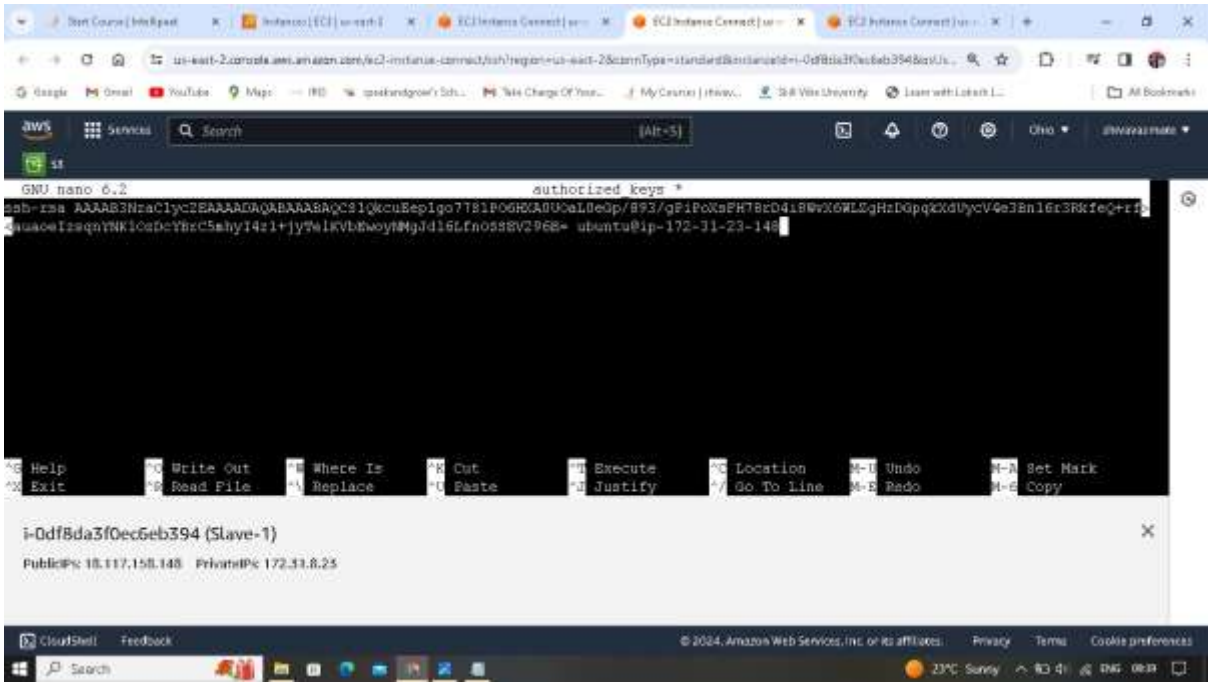
15. See the content of id_rsa.pub by cat command and copy it.



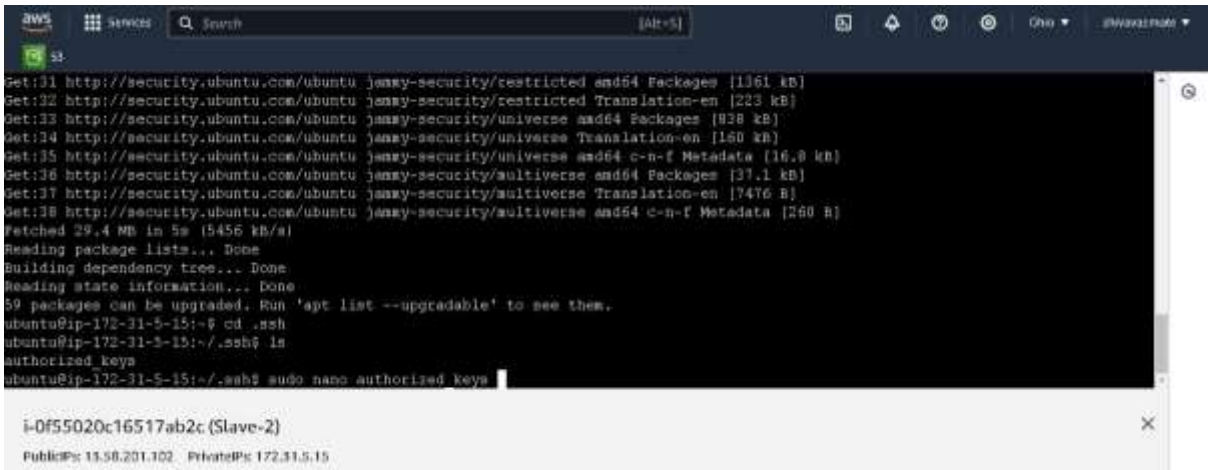
16.Go to .ssh folder in Slave1. do ls and open text editor by sudo nano authorized_keys



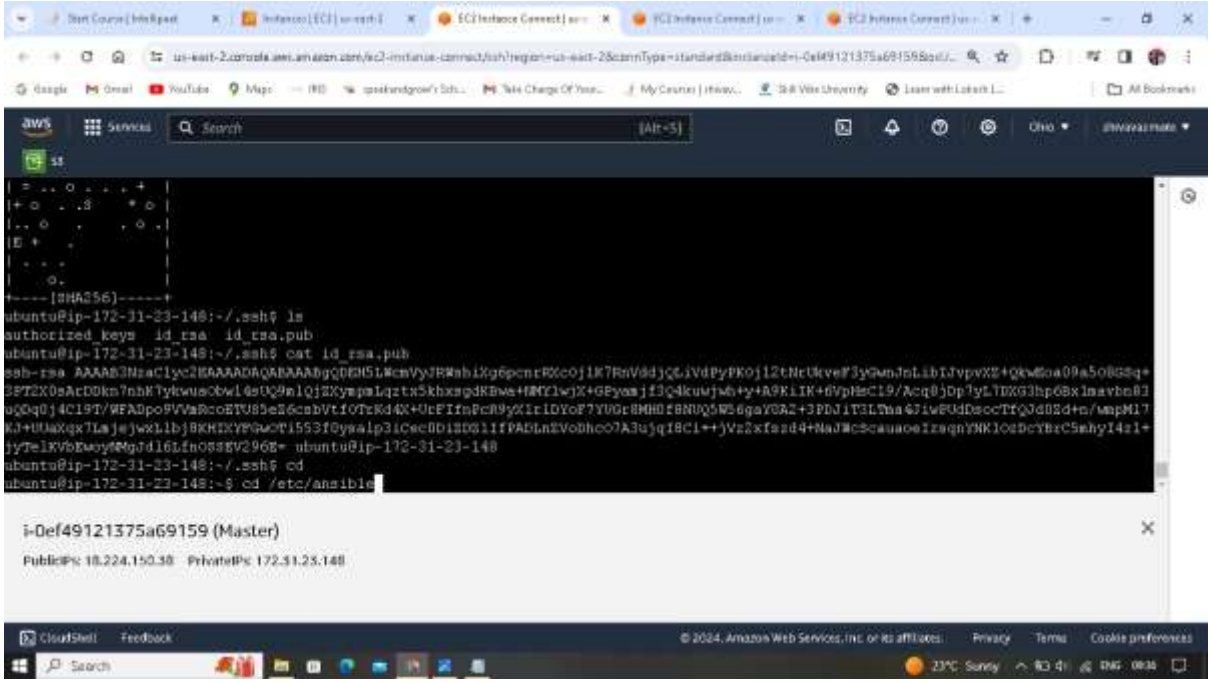
17.Paste the content of the id_rsa.pub. on the second line of the authorized_keys save and exit.



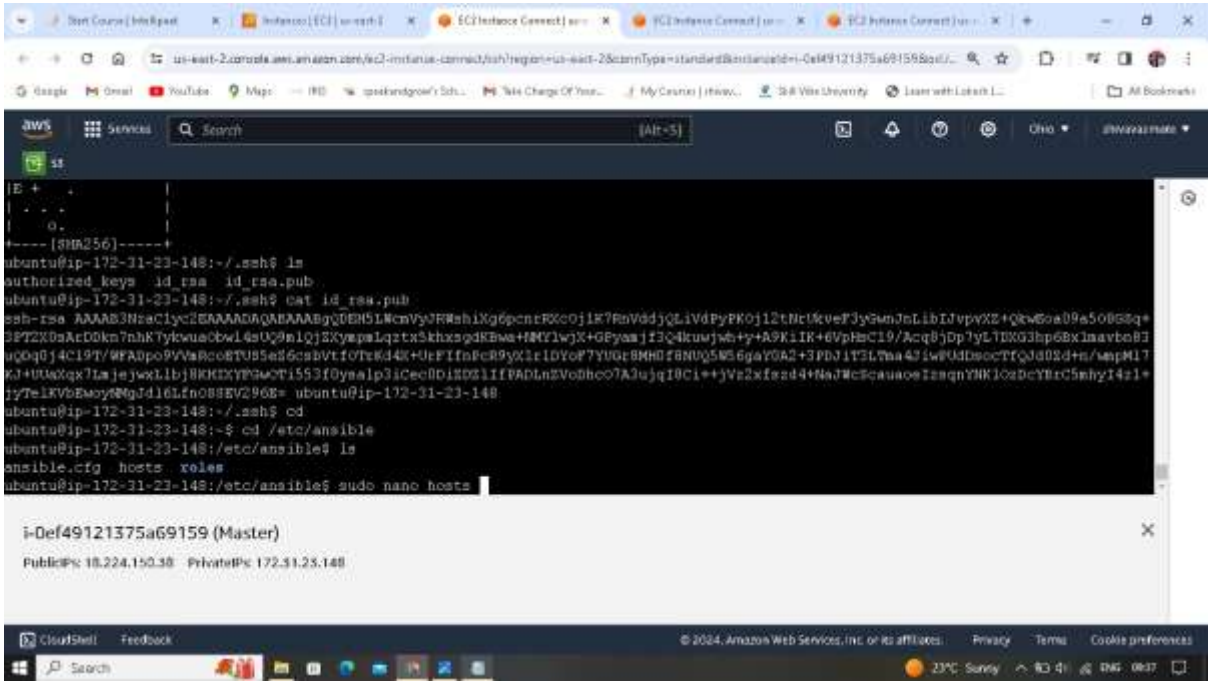
18.Similarly for the slave2 also follow same steps.



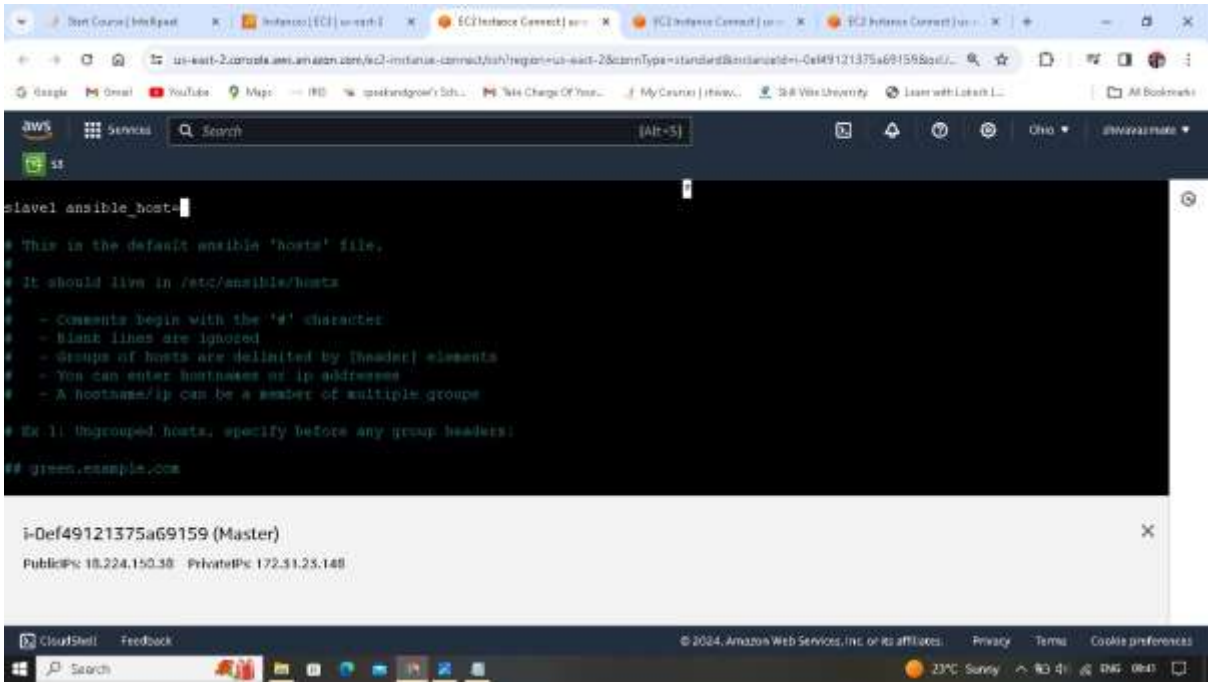
19.Go out of .ssh by cd command



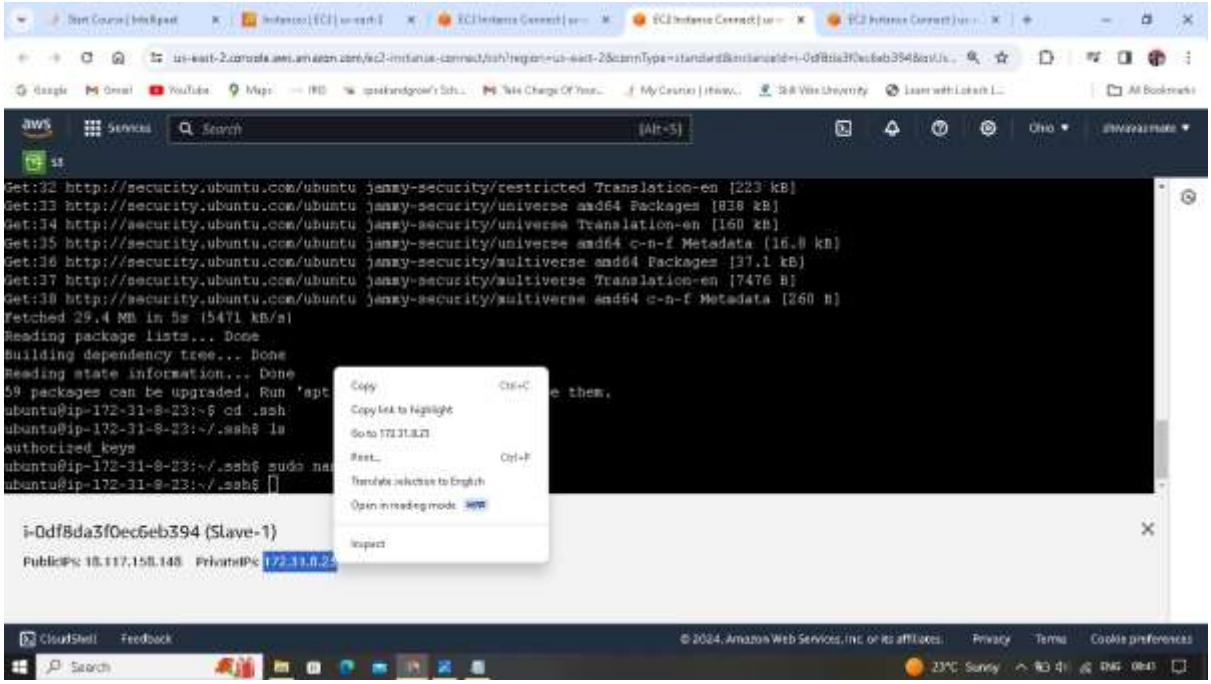
20. Go host folder inside /etc/ansible and open text editor by sudo nano hosts command



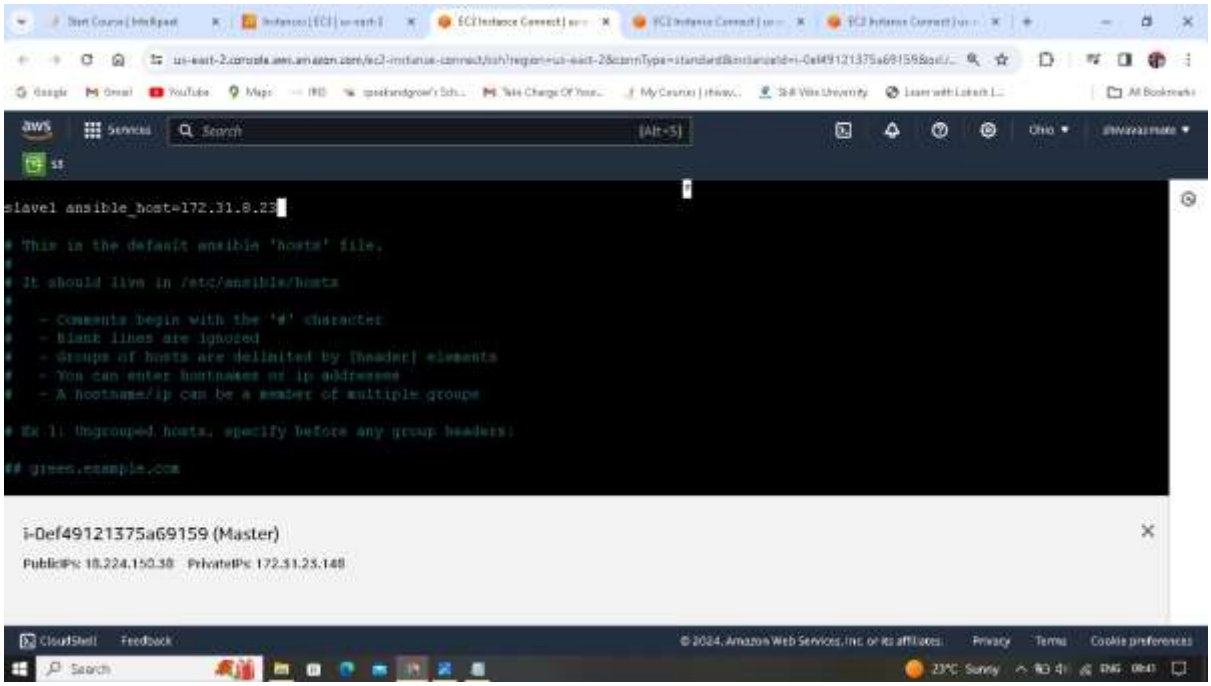
21. Inside it type slave1 ansible_host= private IP of salve1 machine



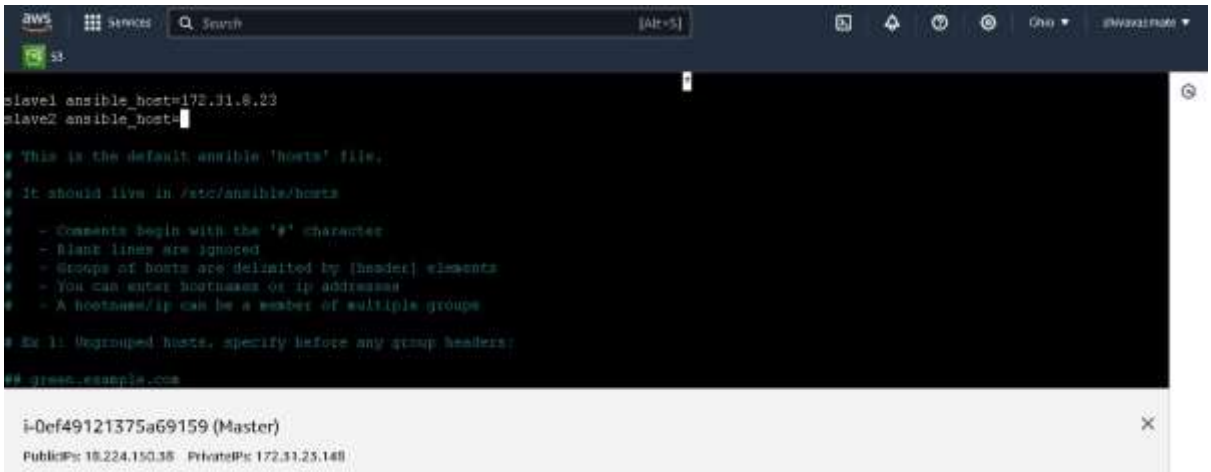
22.Go to Slave1 machine copy the private IP

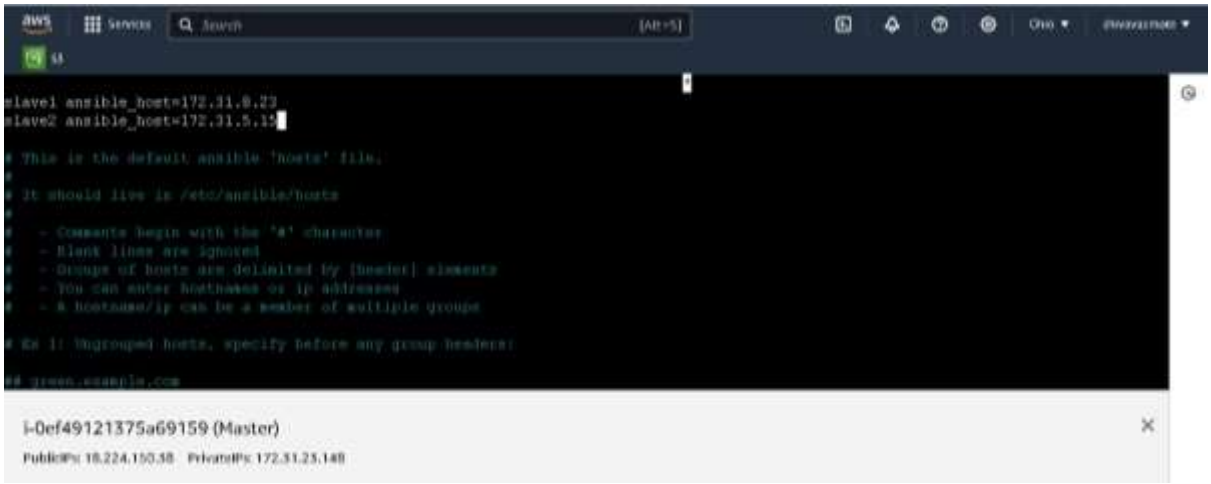


23.Paste it.

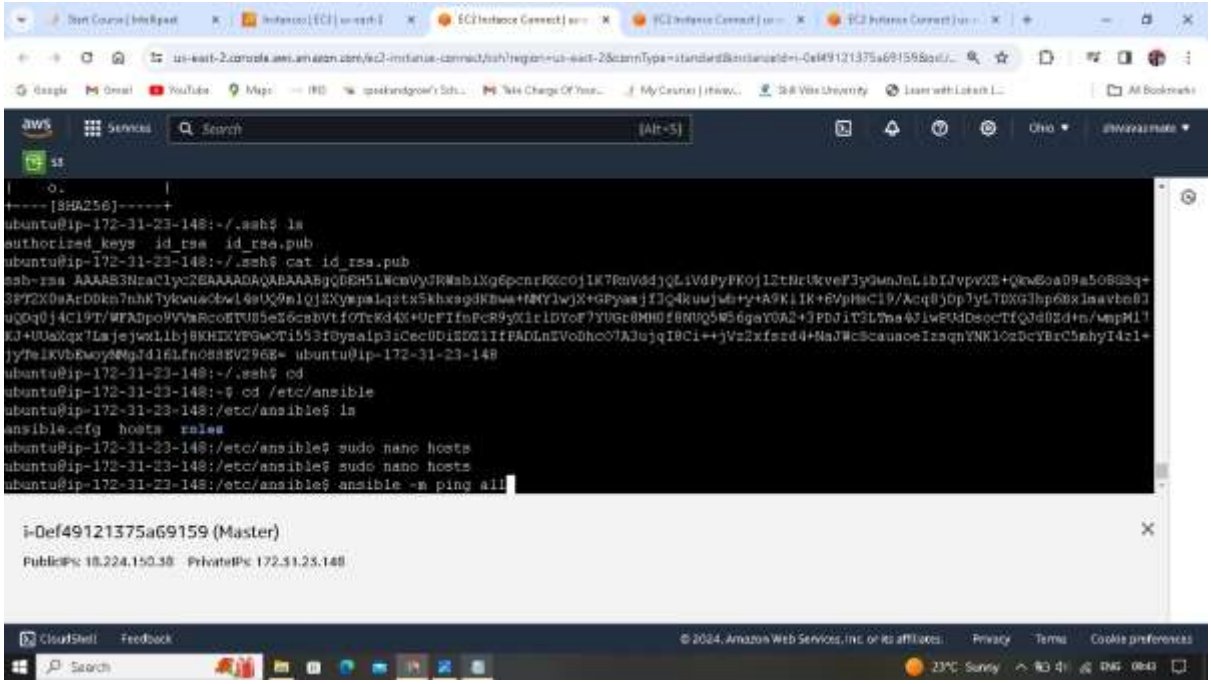


24.Similarly for Slave2 machine

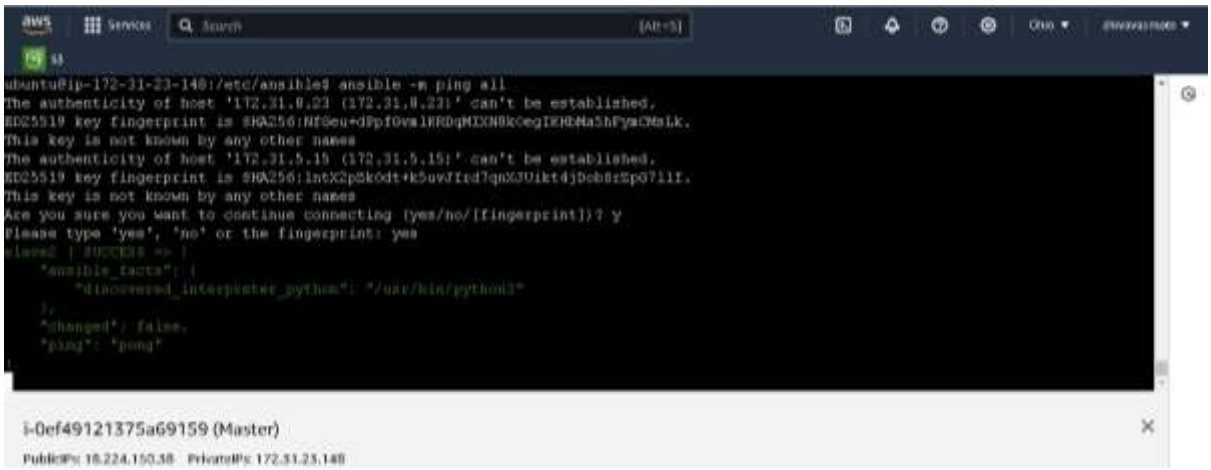




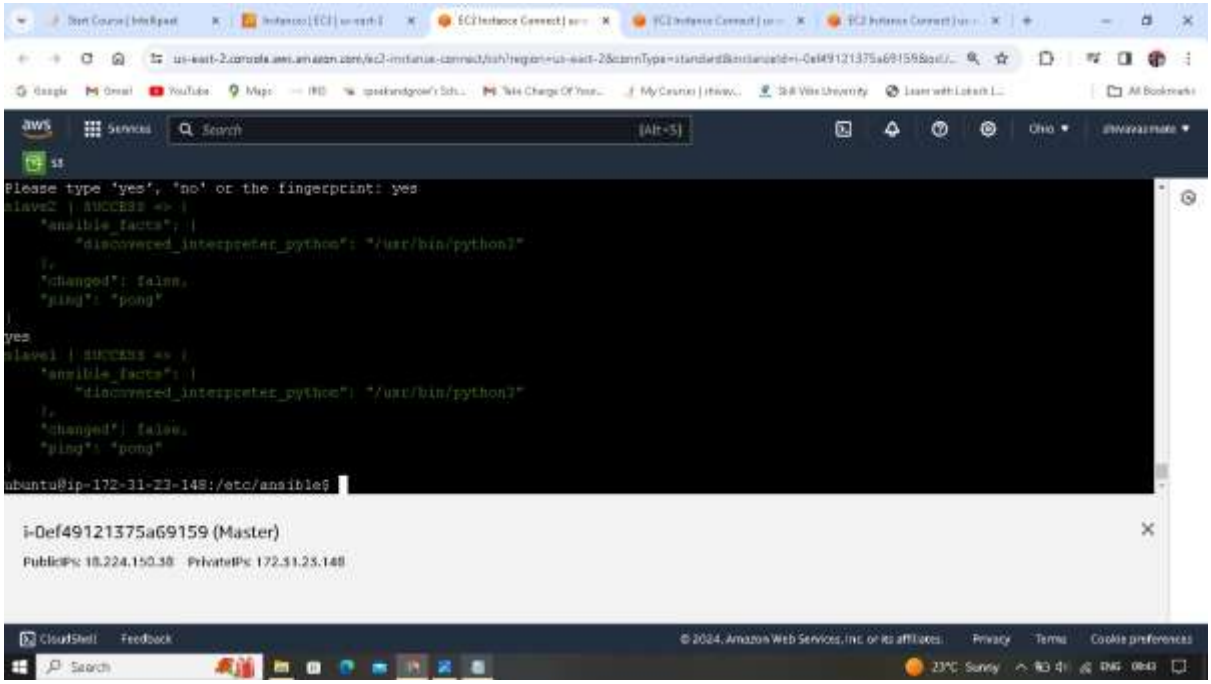
25.Check of machines by ansible -m ping all command



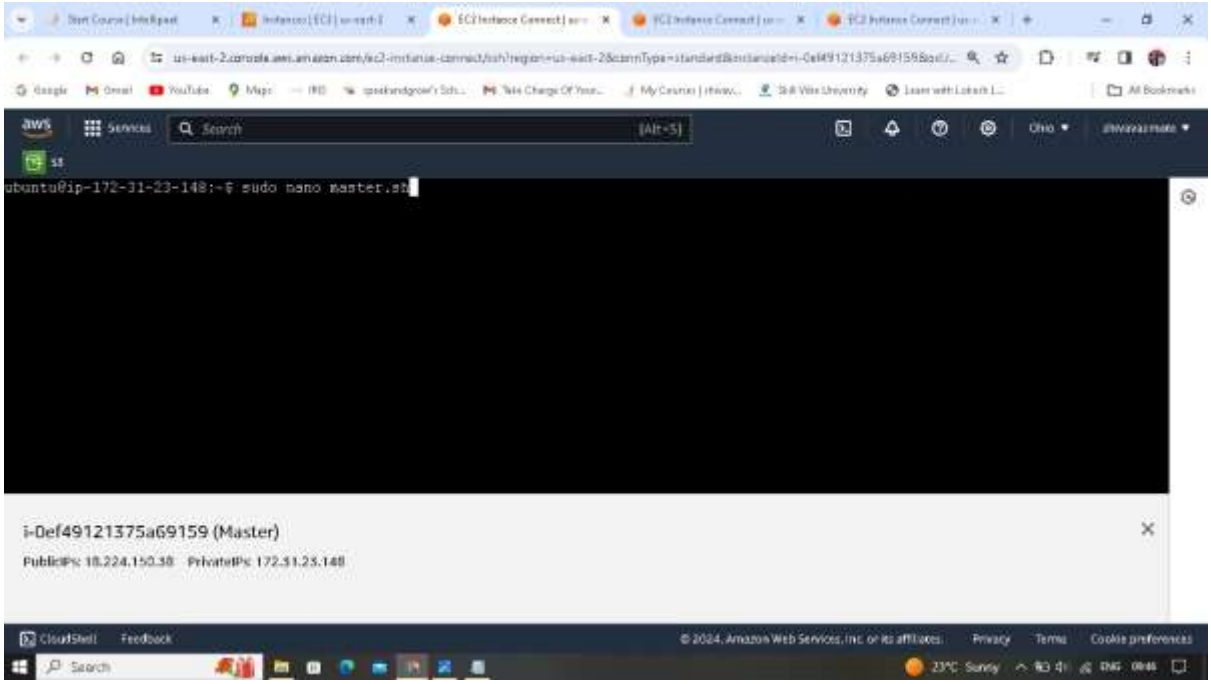
26. Type yes for slave2 machine



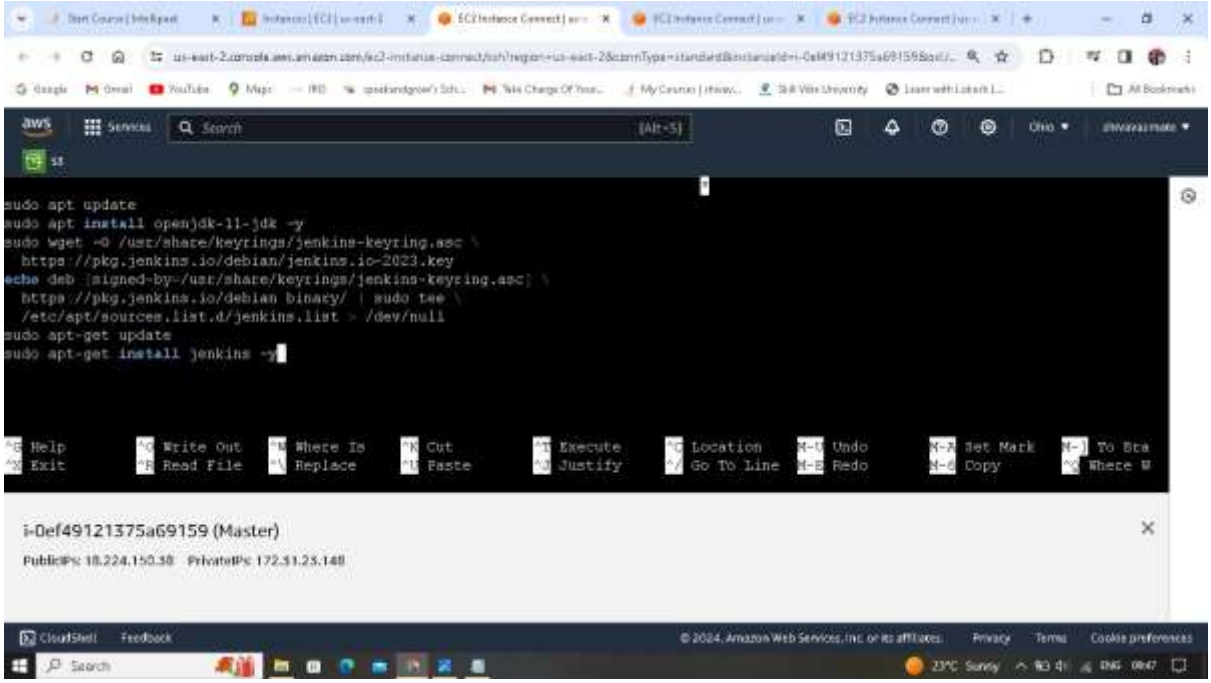
27.Again type yes for slave1 machine. Now both slaves machine are connect with master machine.



28.Open an editor by sudo nano master.sh



29.Write a script for installing java, docker and Jenkins on master machine.



sudo apt update

sudo apt install docker.io -y

sudo apt install openjdk-11-jdk -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

  https://pkg.jenkins.io/debian/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

  https://pkg.jenkins.io/debian binary/ | sudo tee \

  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

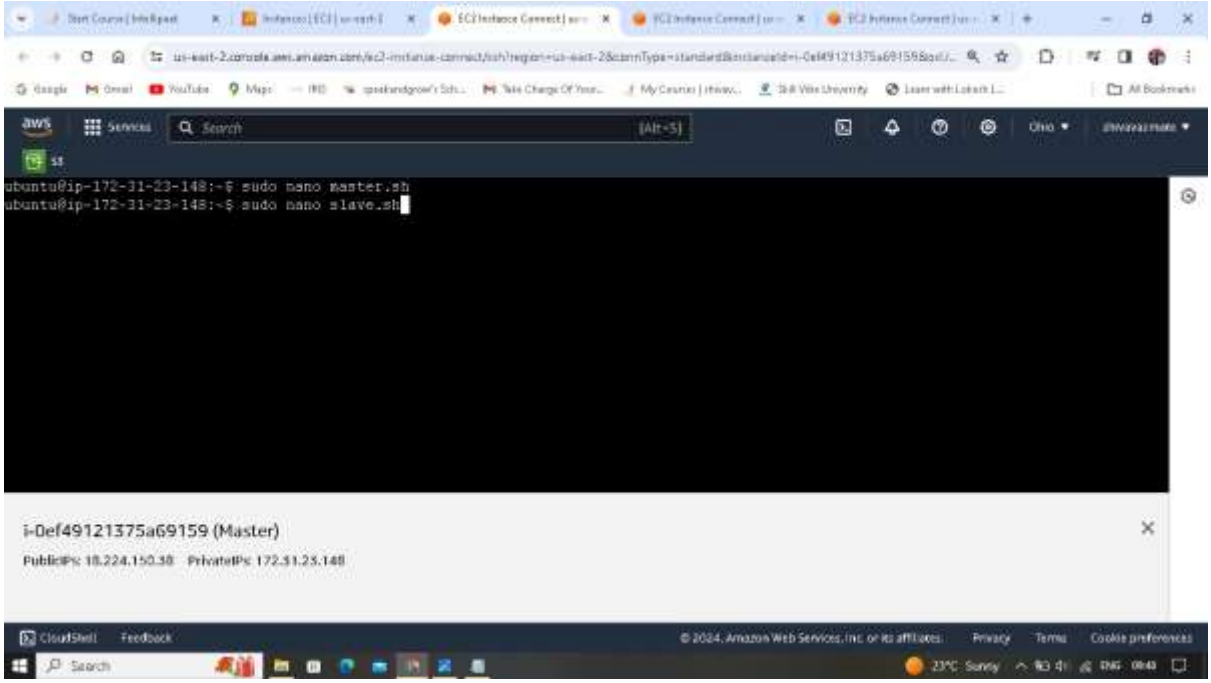
sudo apt-get install jenkins -y

ctrl + s to save

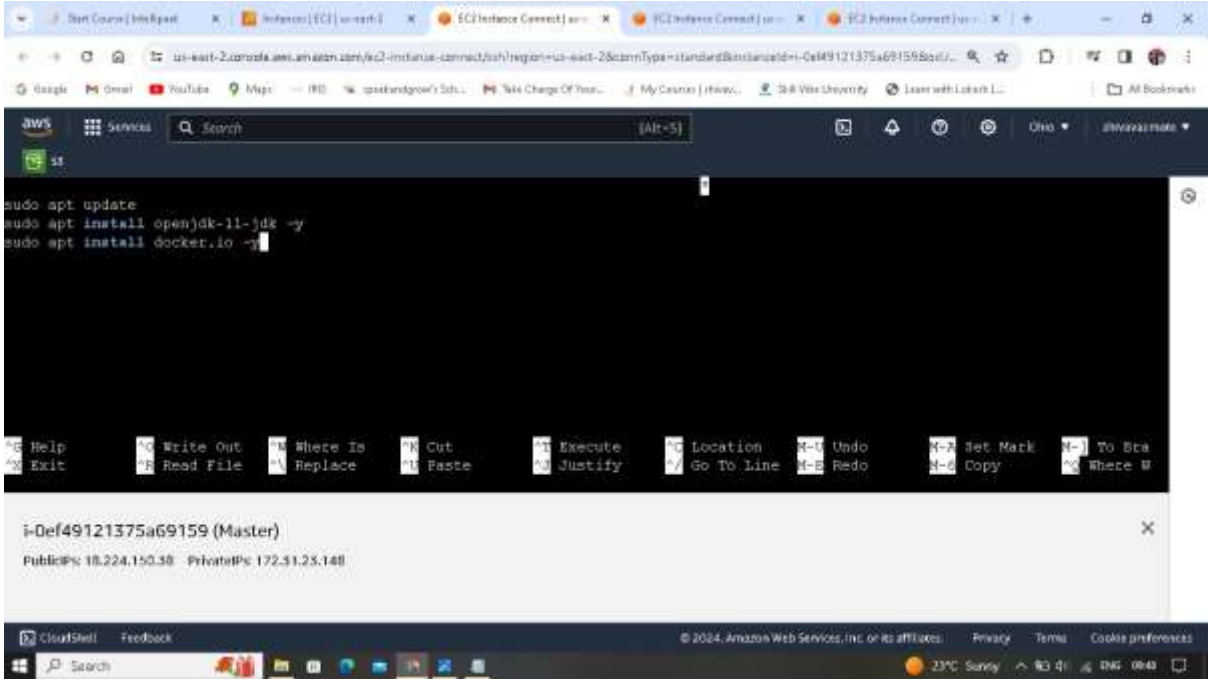
and

ctrl + x to exit.

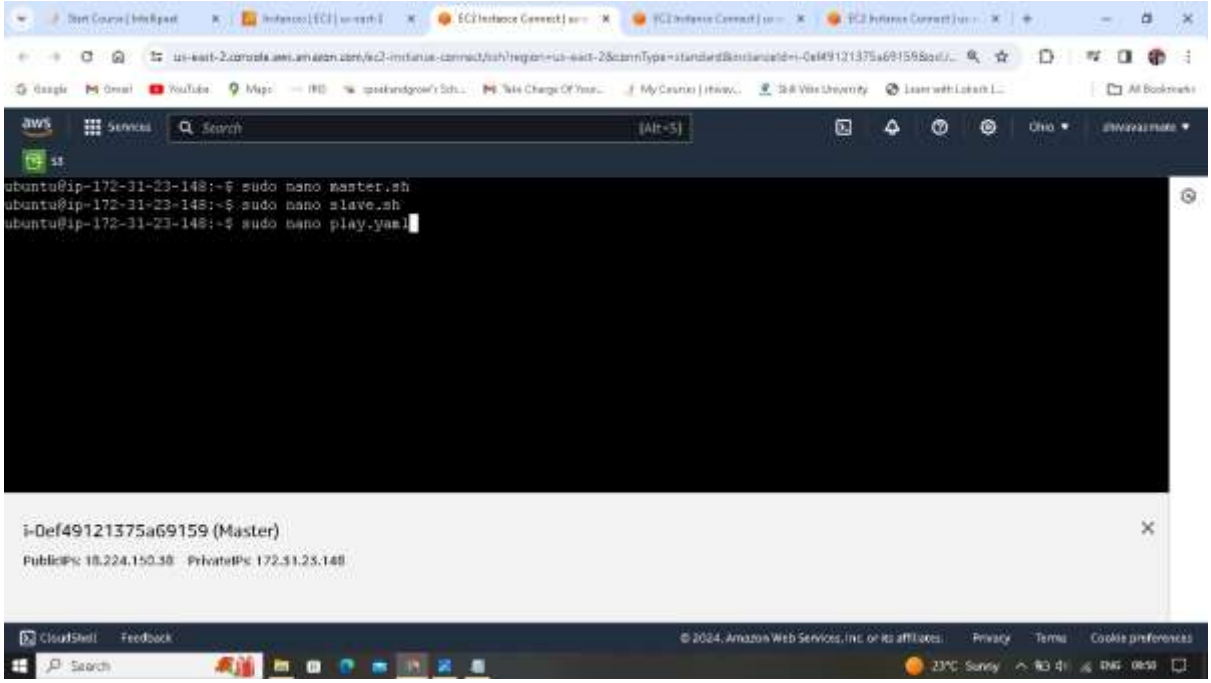
30. Open an editor by sudo nano slave.sh



31.Write a script for installing java and docker on slave machines save and exit.

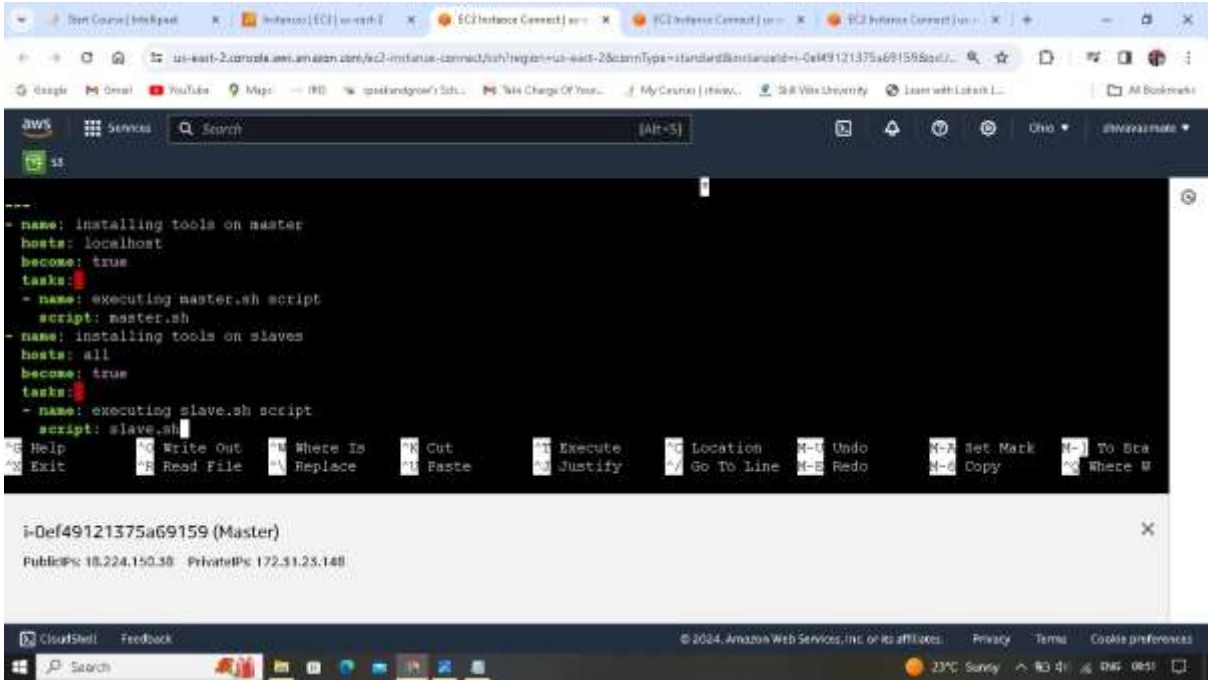


32. Open an yaml file with sudo nano play.yaml command



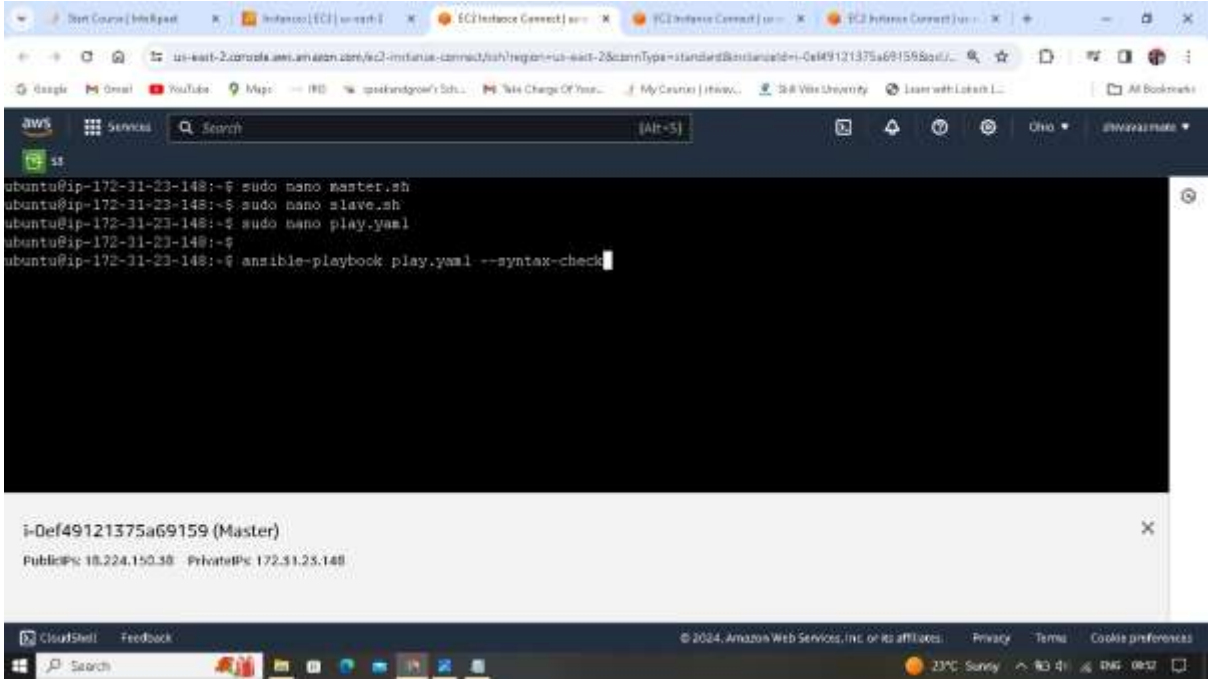
33. Write a ymal file as follows

```
---

- name: installing tools on master

  hosts: localhost

  become: true

  tasks:

  - name: executing master.sh script

    script: master.sh

- name: installing tools on slaves

  hosts: all

  become: true

  tasks:

  - name: executing slave.sh script

    script: slave.sh
```



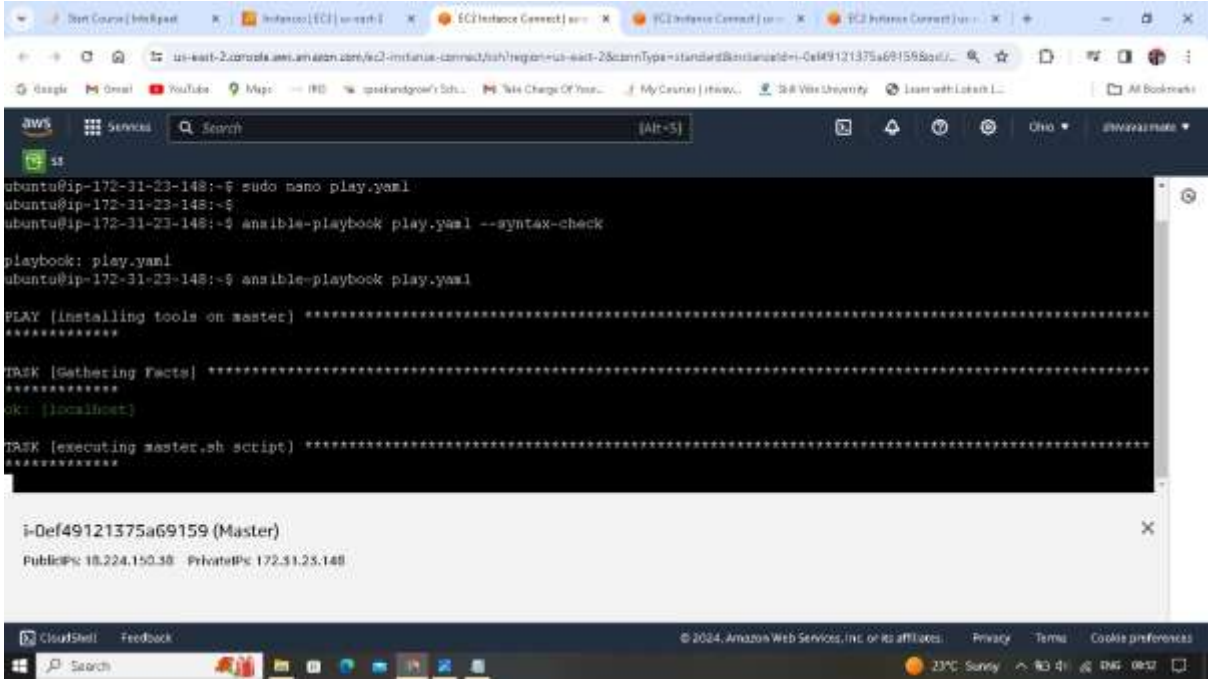
ctrl + s to save and ctrl + x to exit.

34.Check for any errors in the yaml file by ansible-playbook play.yaml –syntax-check

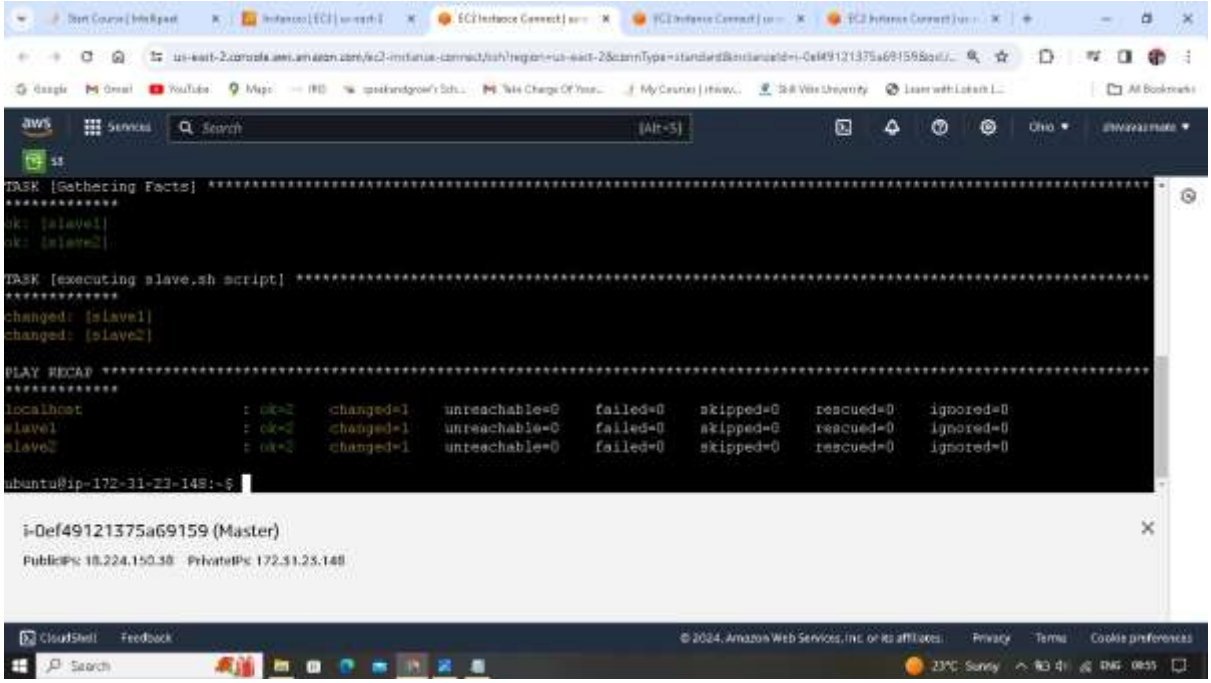


35.No errors and run ansible-playbook play.yaml command

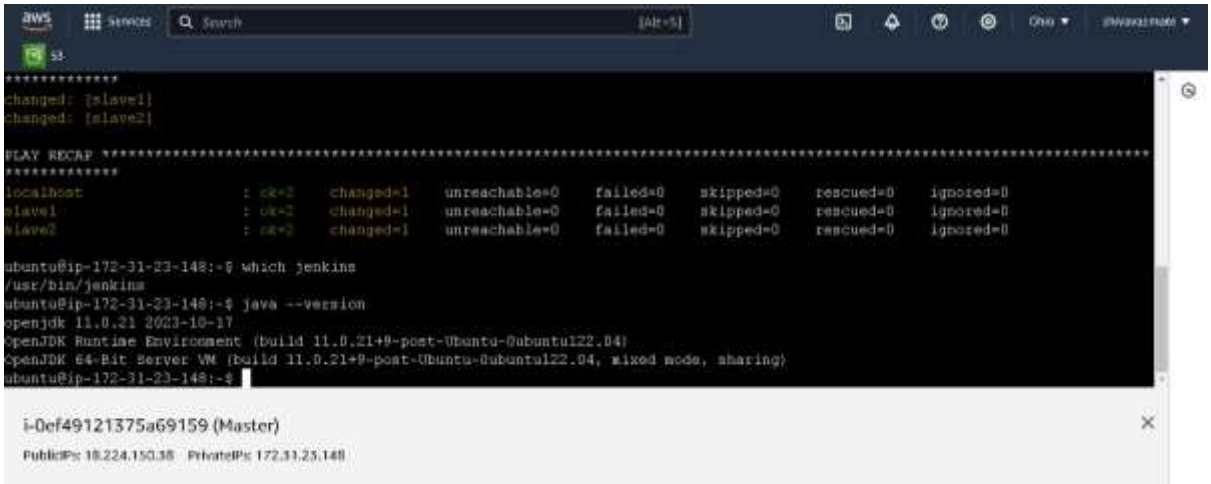
36.Frist on master.sh script is running to install



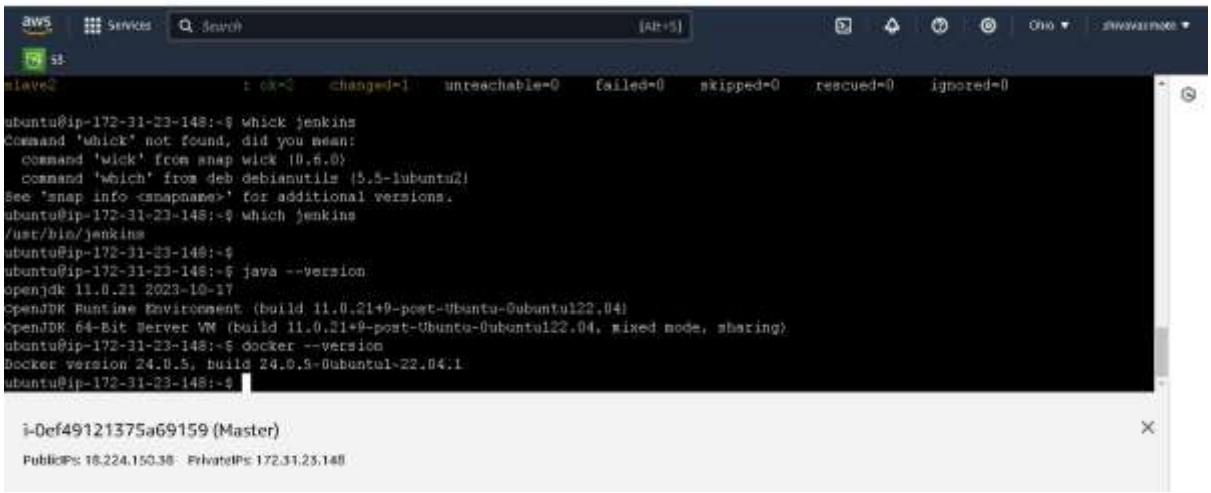
37.Secondly on slave machines are running to install and finally all tasks are successful.



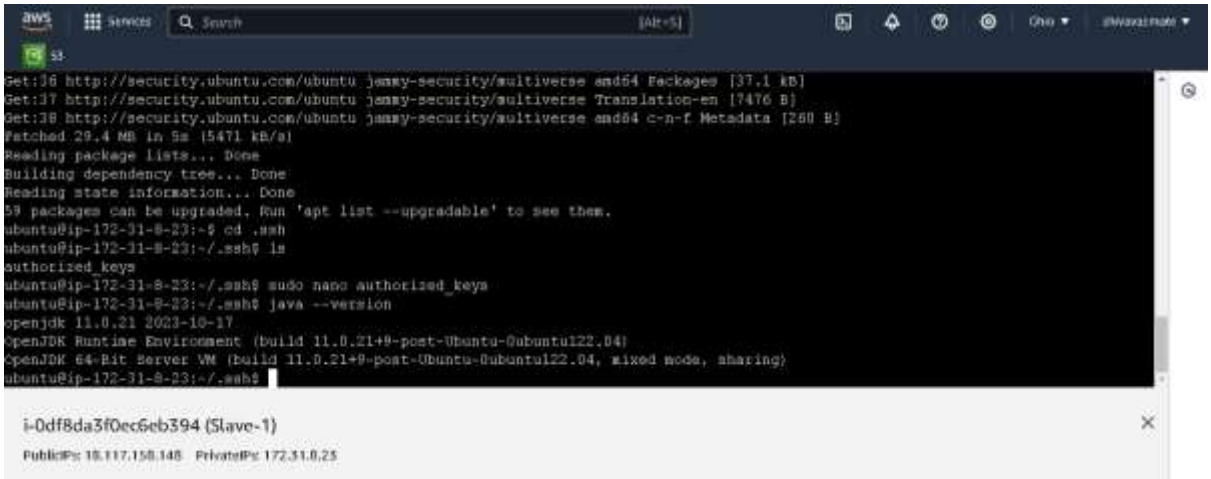
38.Check on master machine Jenkins by which jenkins command, java –version for java



And for docker by docker --version

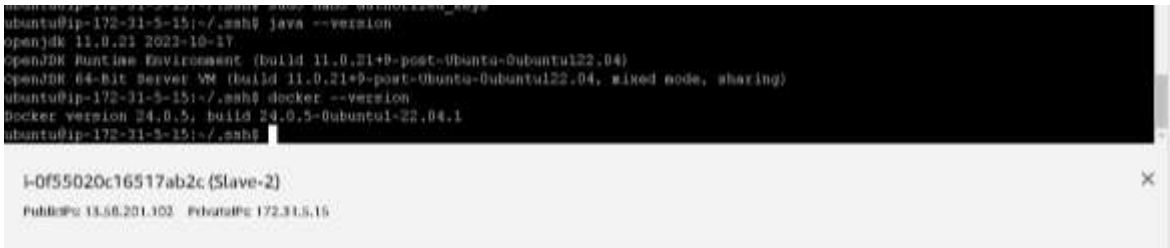


39. Go to slave1 check for java by java –version command

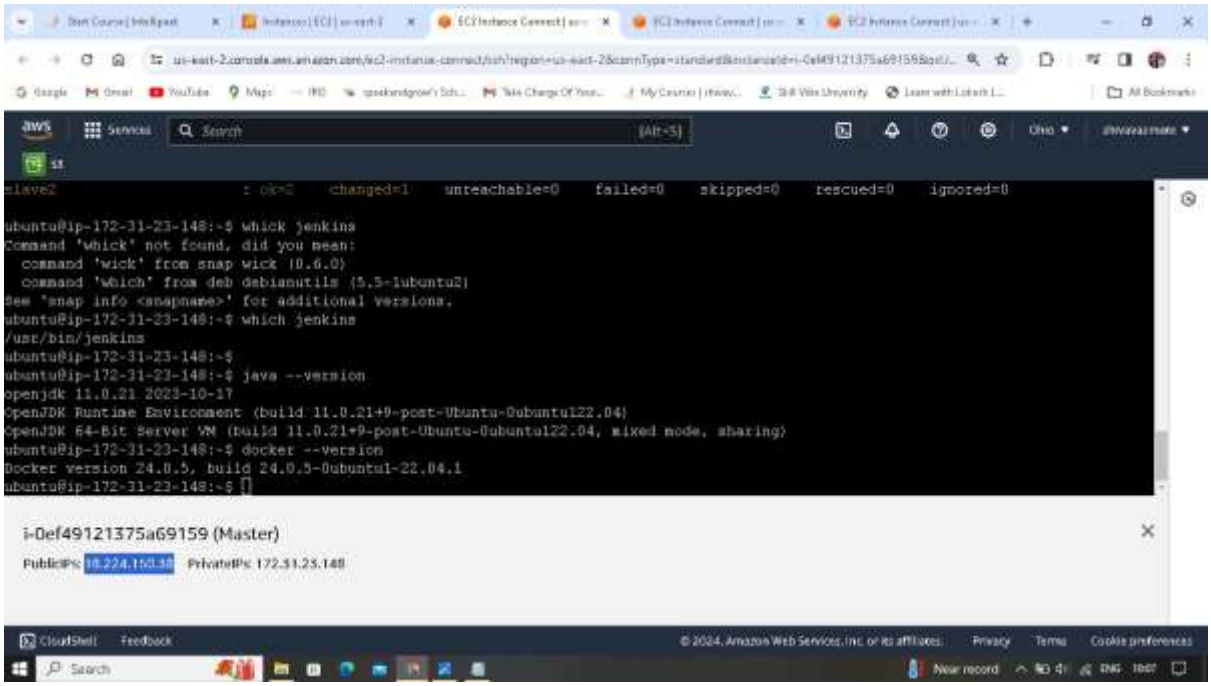


And for docker by docker –version

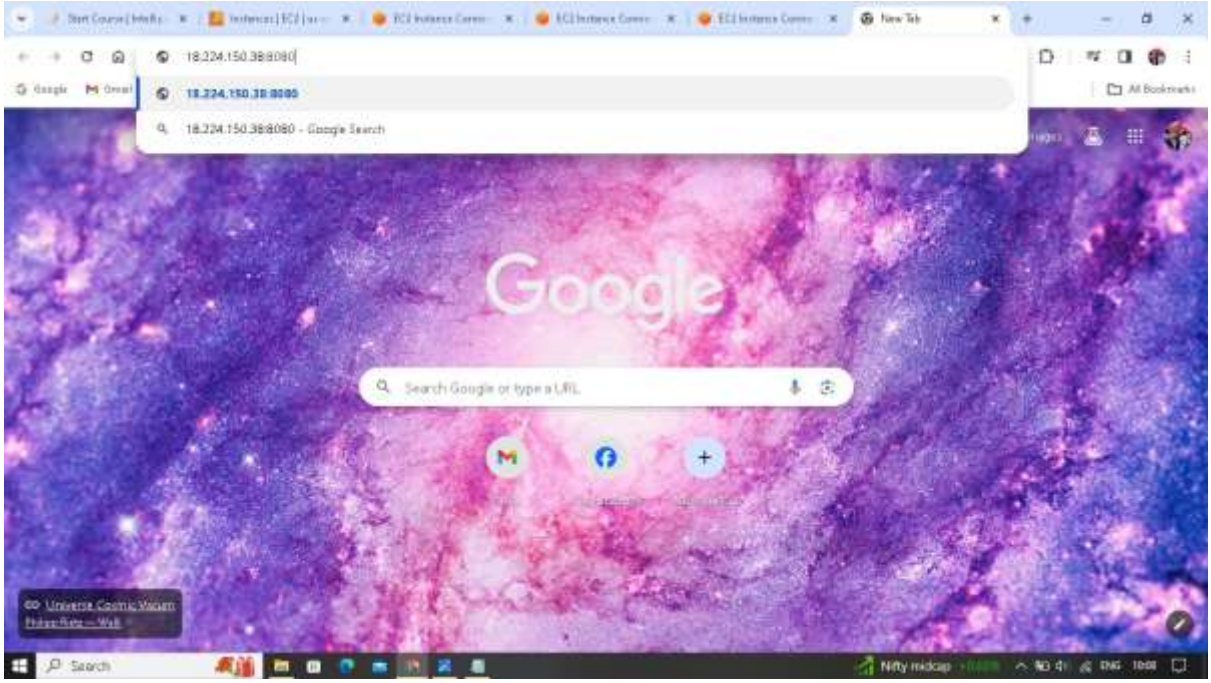
40. Go to slave2 check for java by java –version command and for docker by docker –version



41.Go to master machine copy the public IP

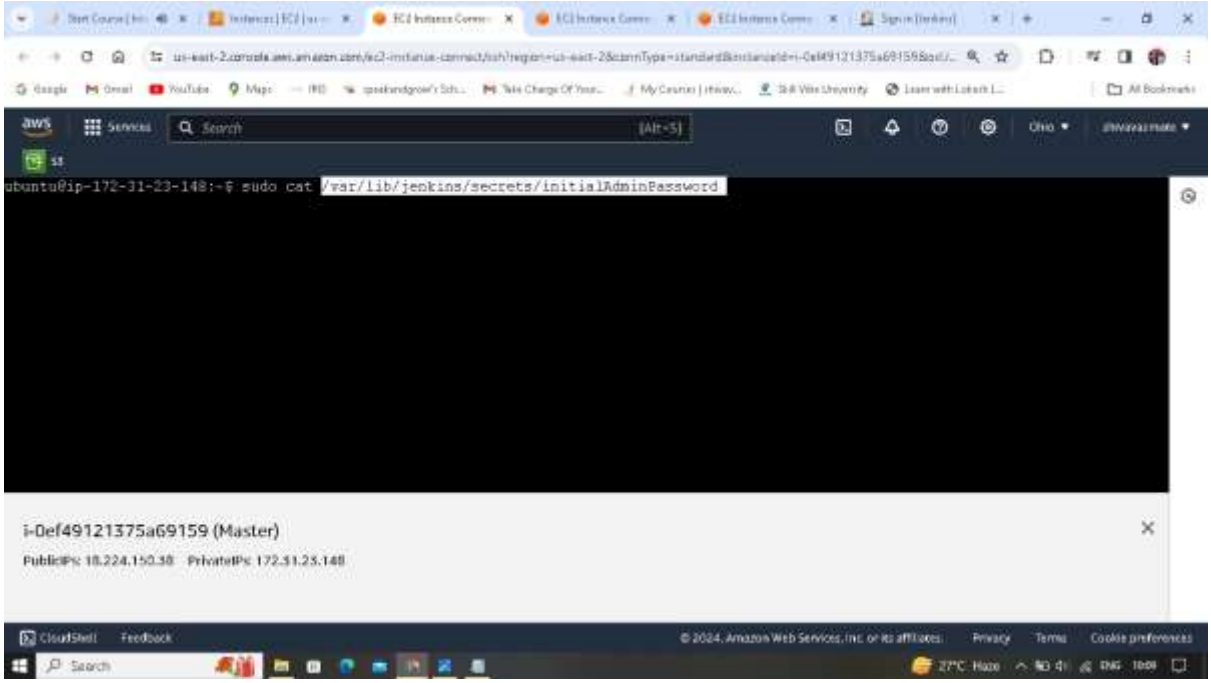


42.Paste it with port number:8080 in another browser.

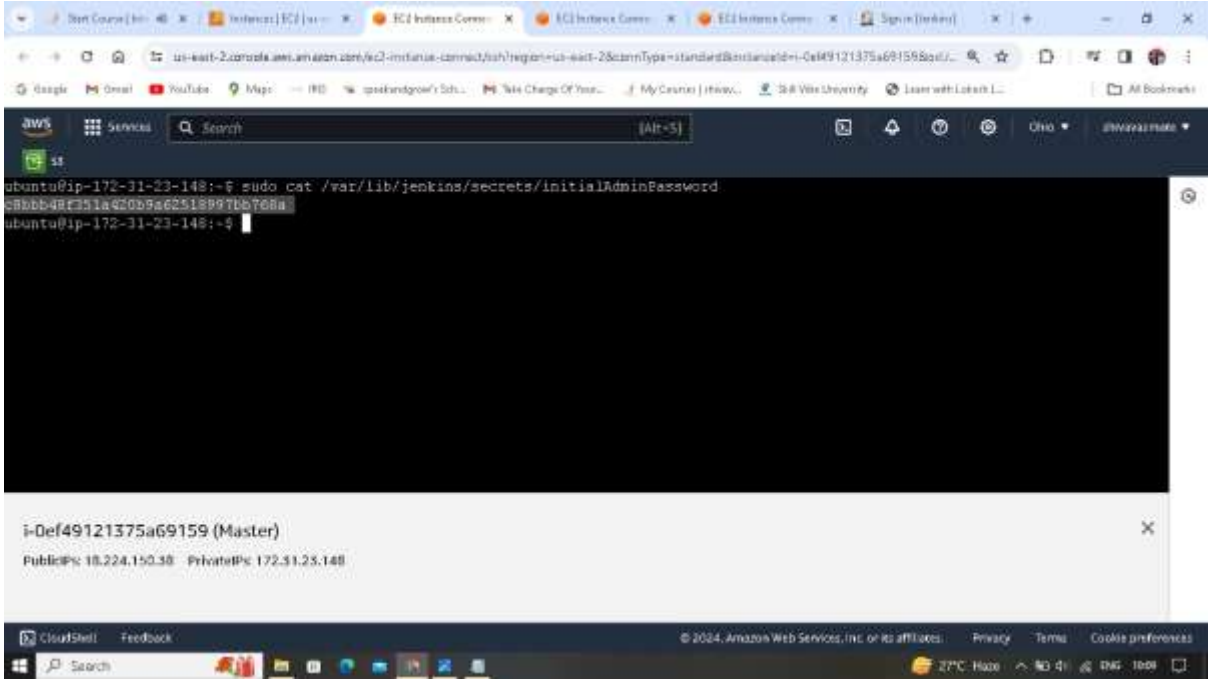


43.Copy the /var/lib/jenikins/secrets/initialadminPassword.

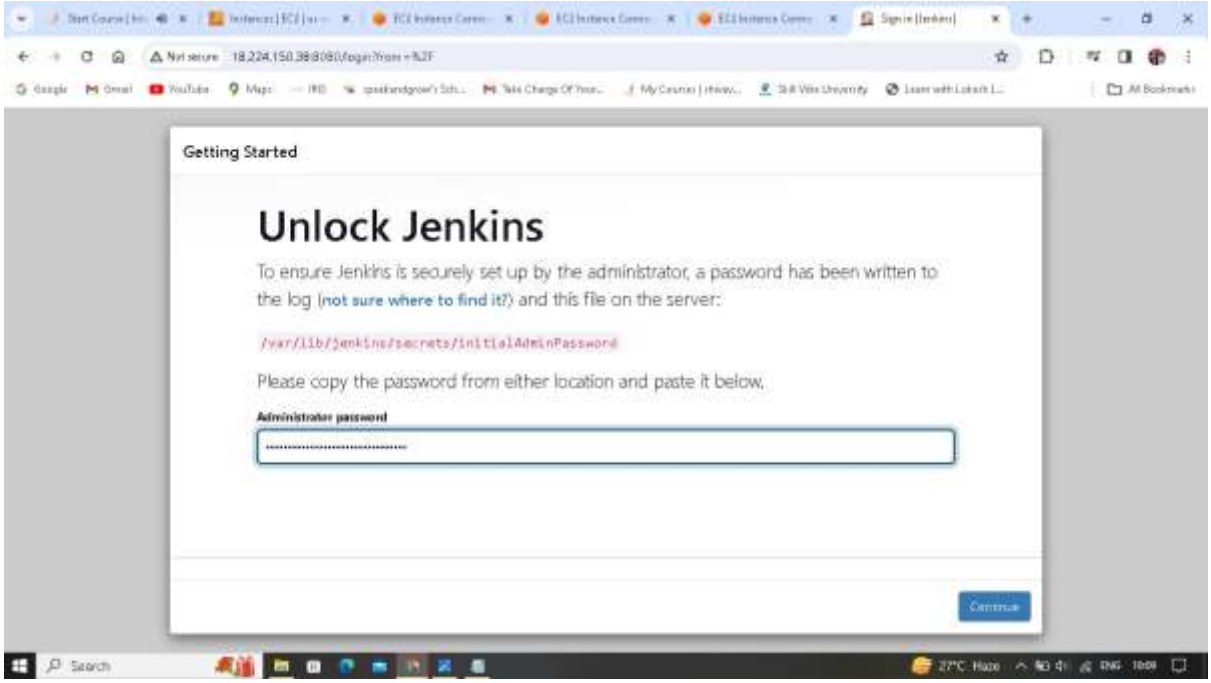
44.Open the content of it by sudo cat command paste it.



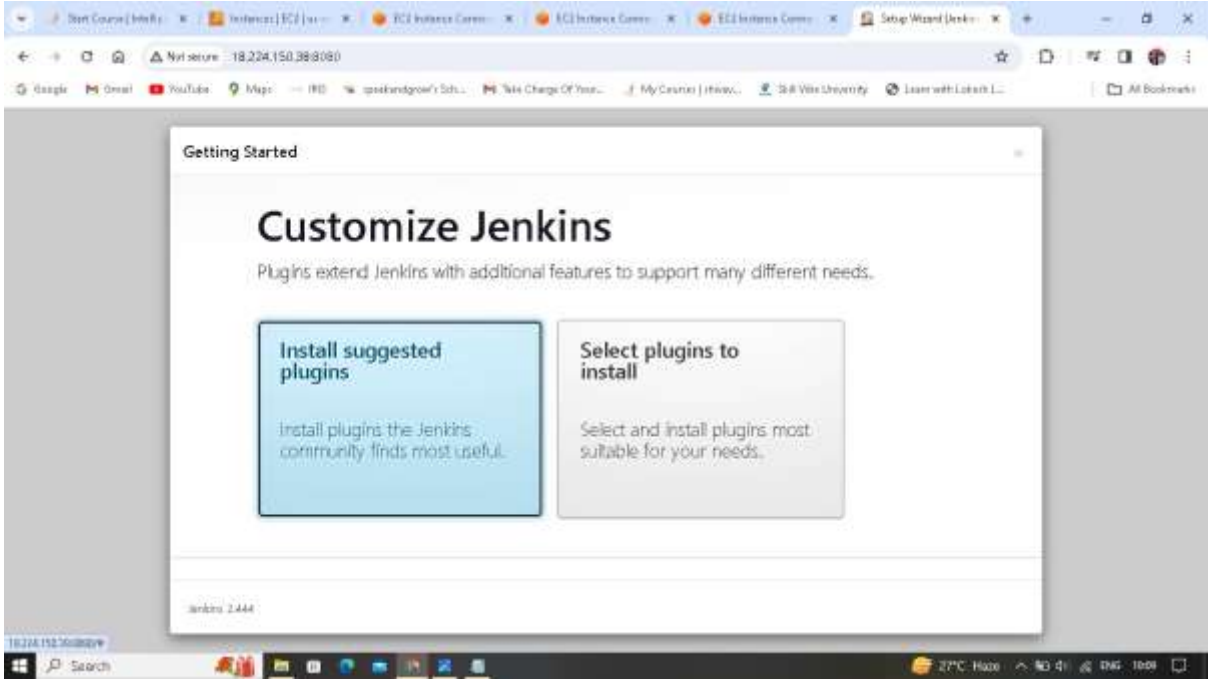
45.Copy the password



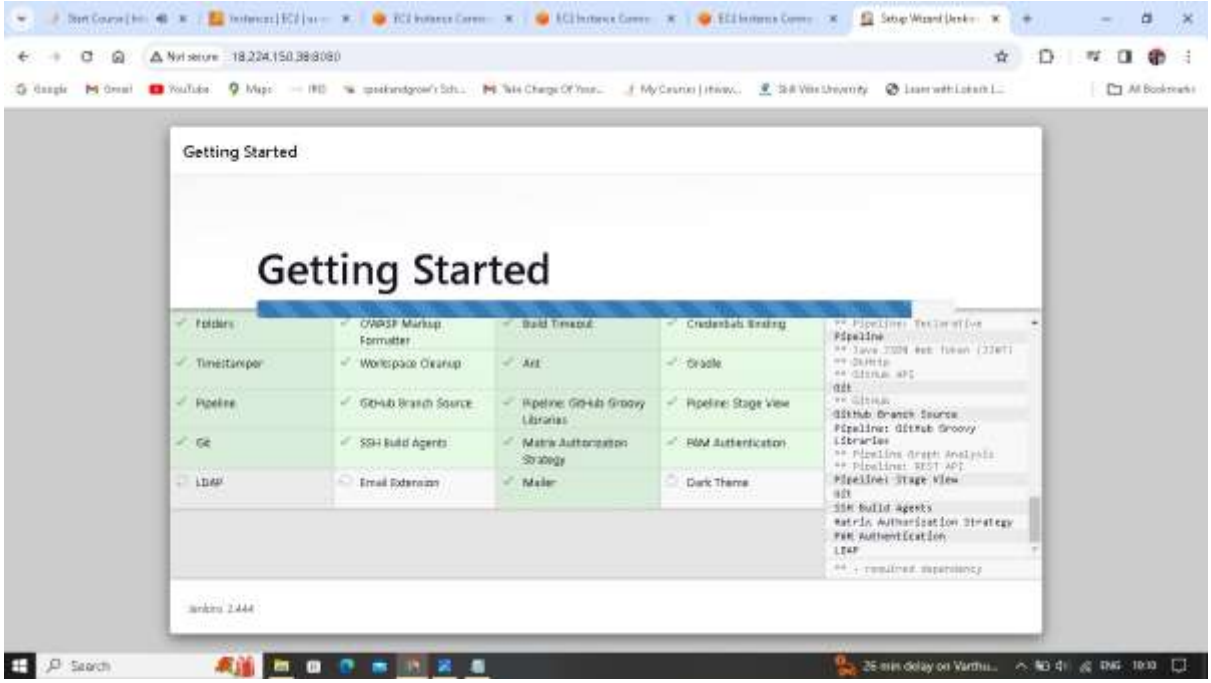
46.Paste it and click on continue.



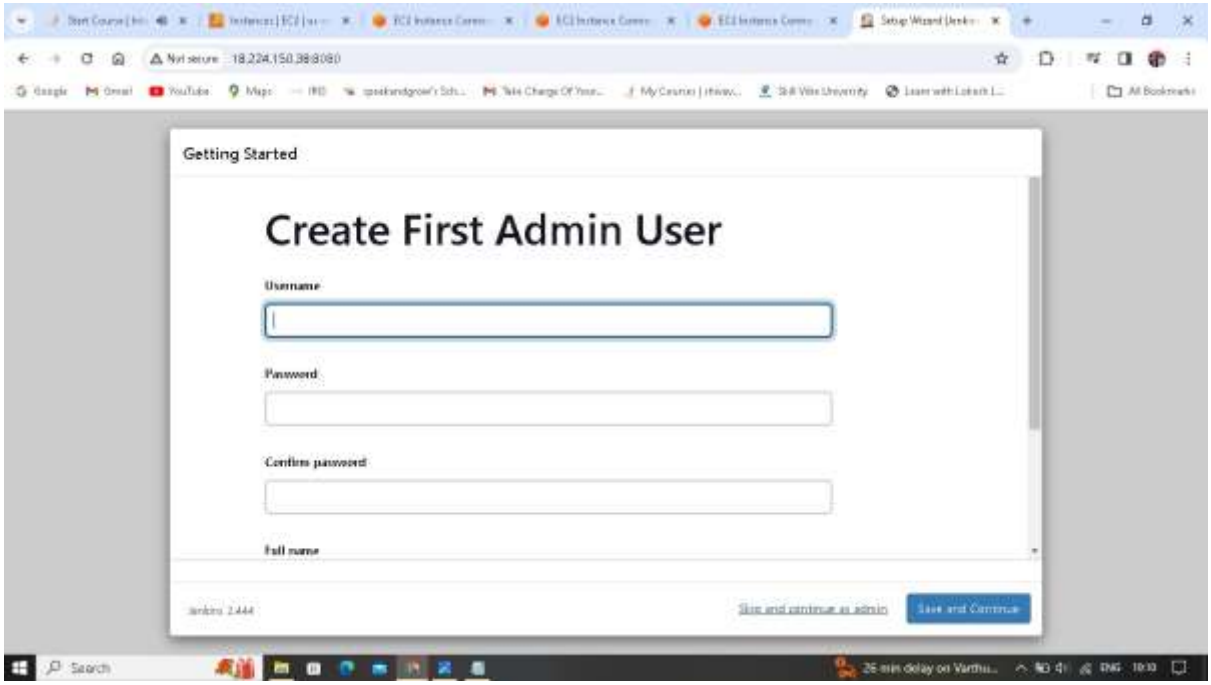
47.Click on Install suggested plugins



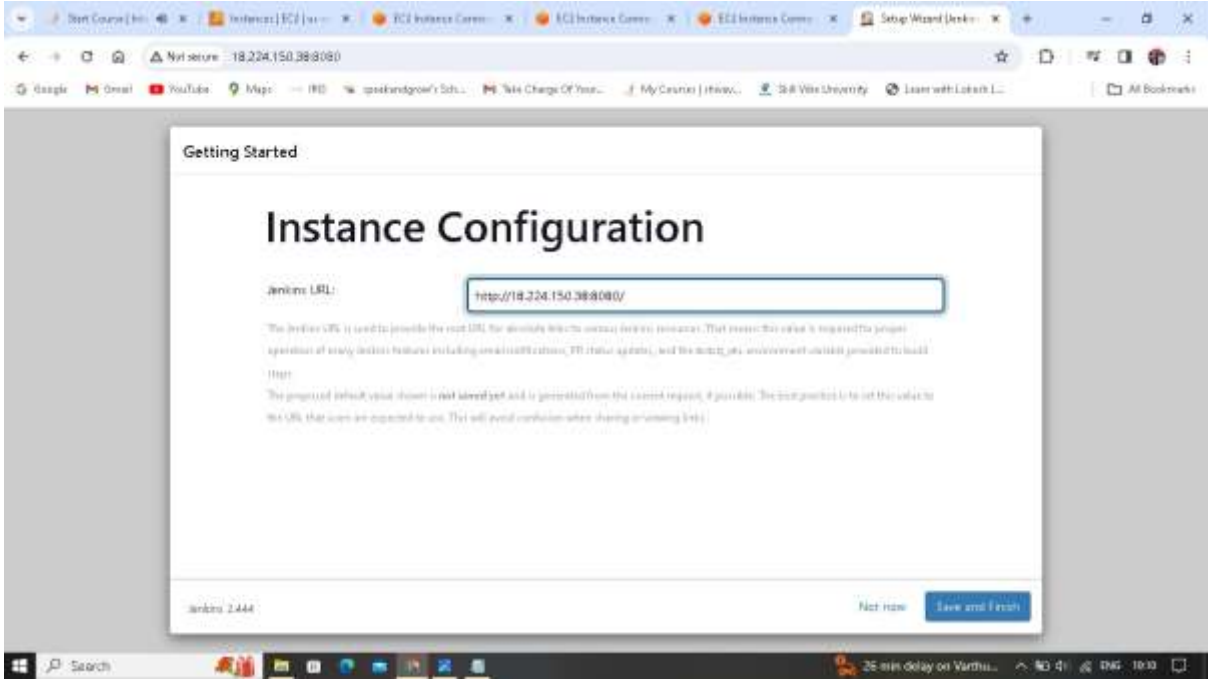
48.Let it install all plugins



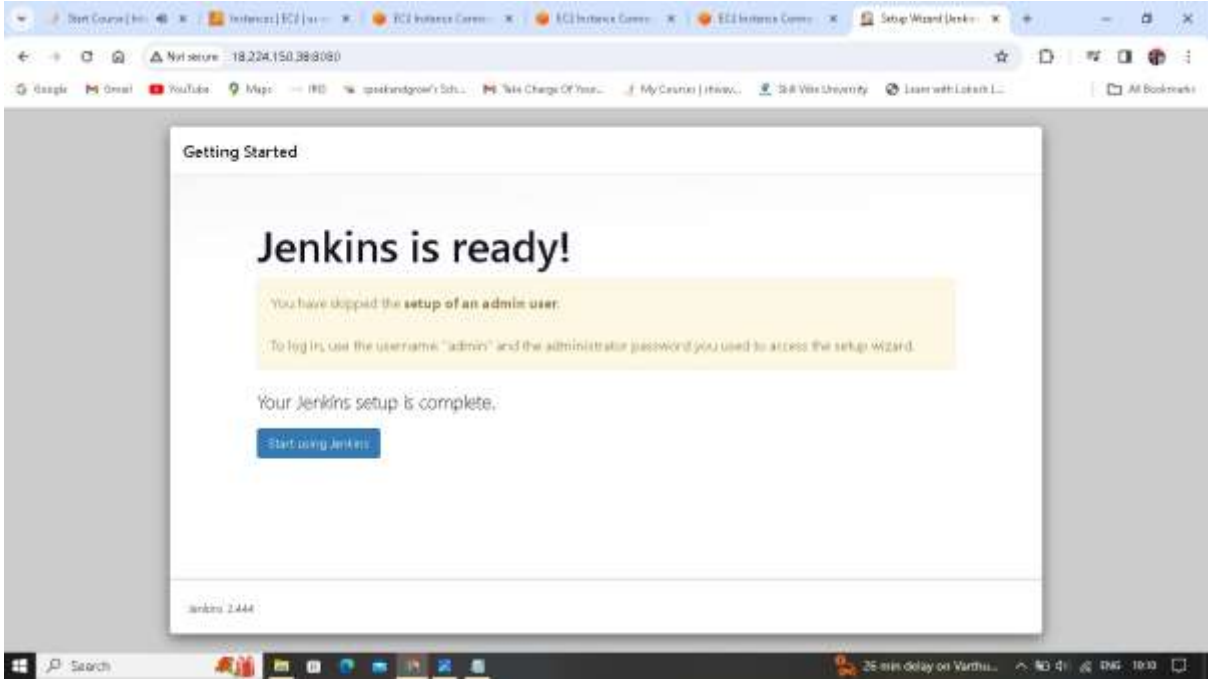
49.Click on skip and continue as admin.



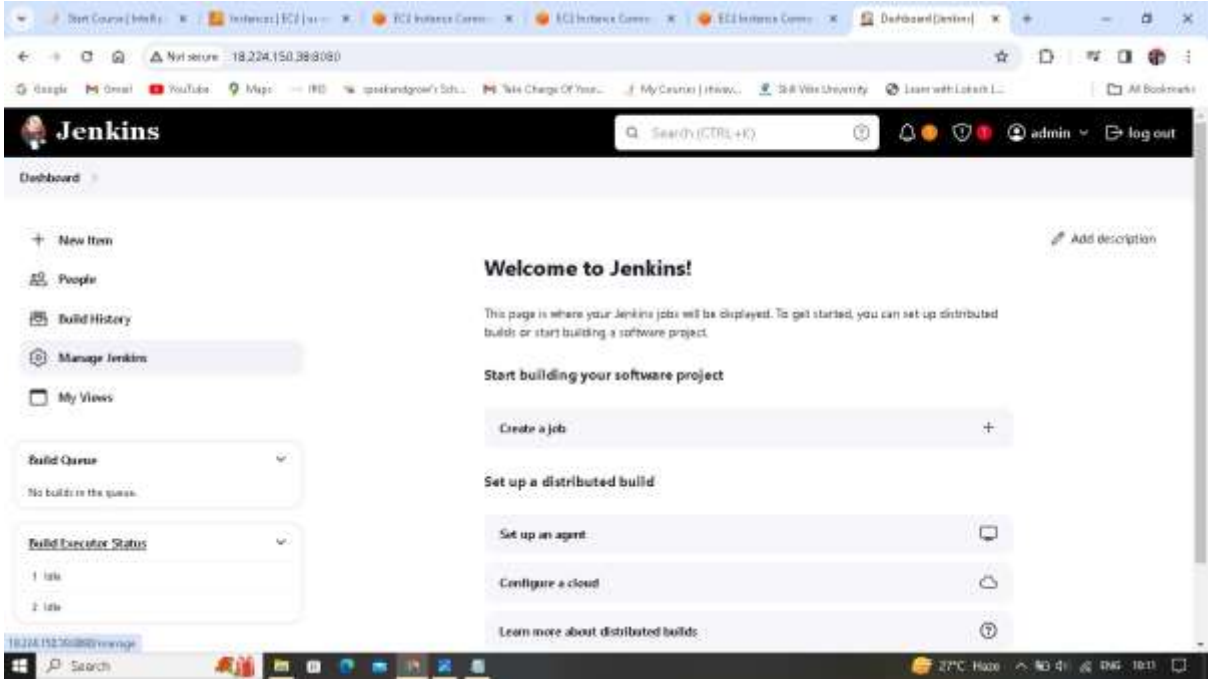
50.Click on save and finish



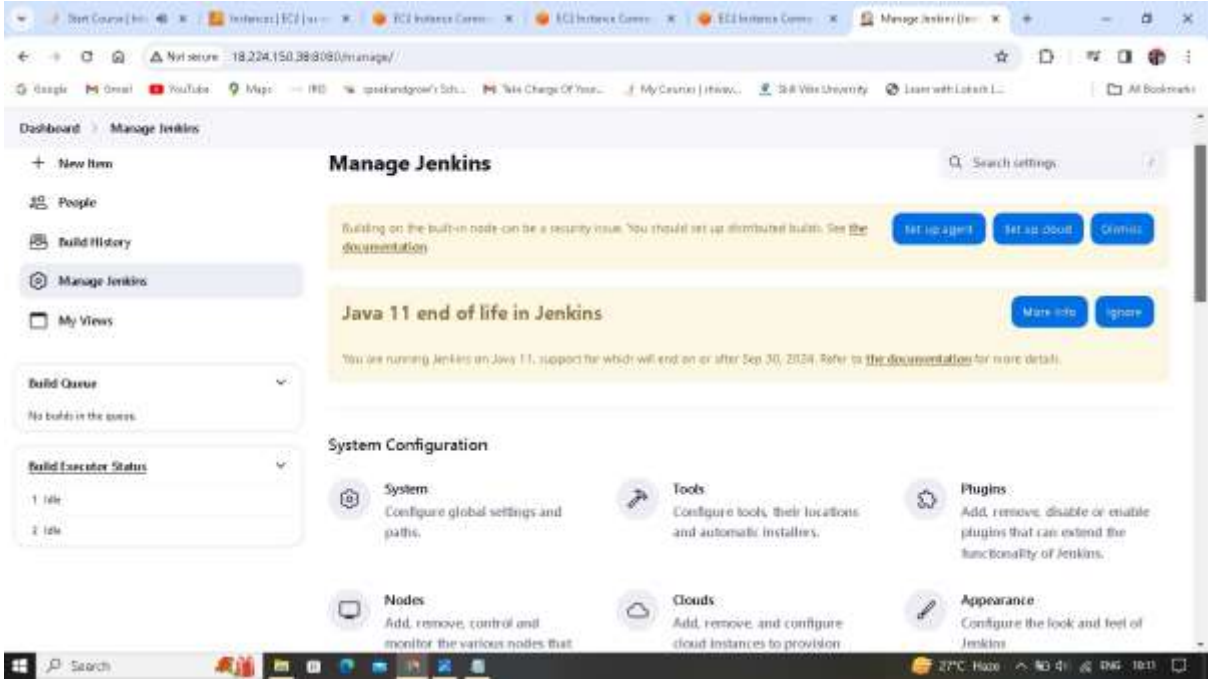
51 Jenkins is ready. Click on Start using Jenkins



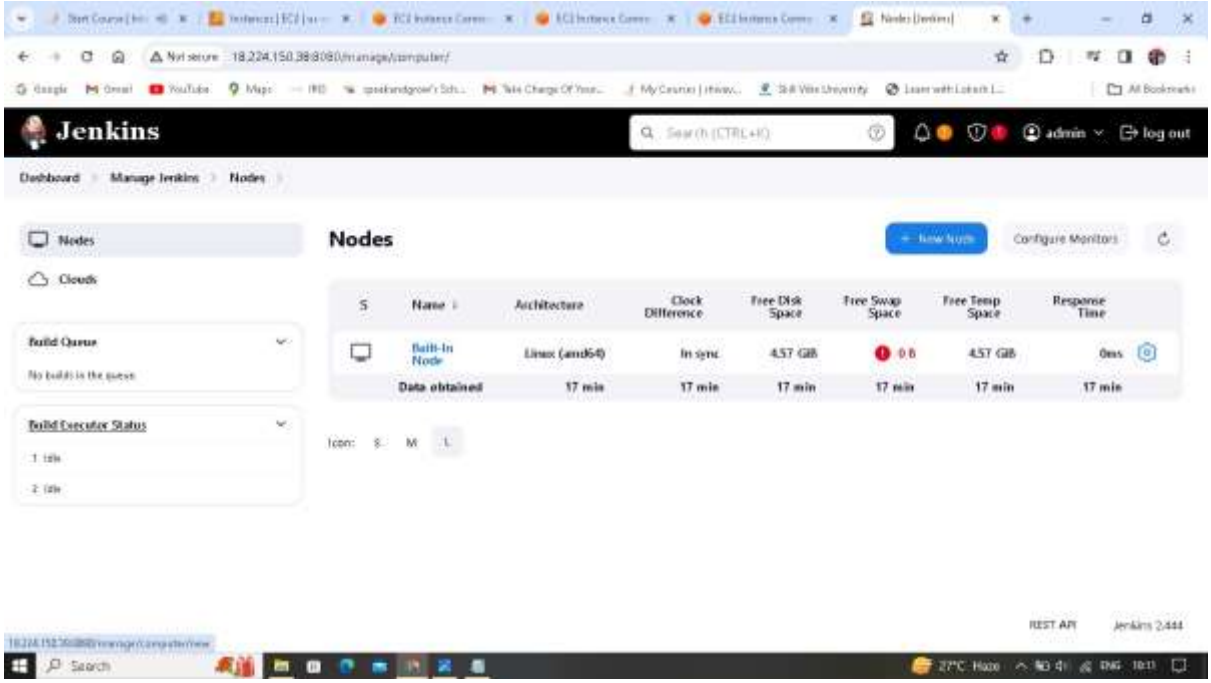
52.Click on Manage jenkins



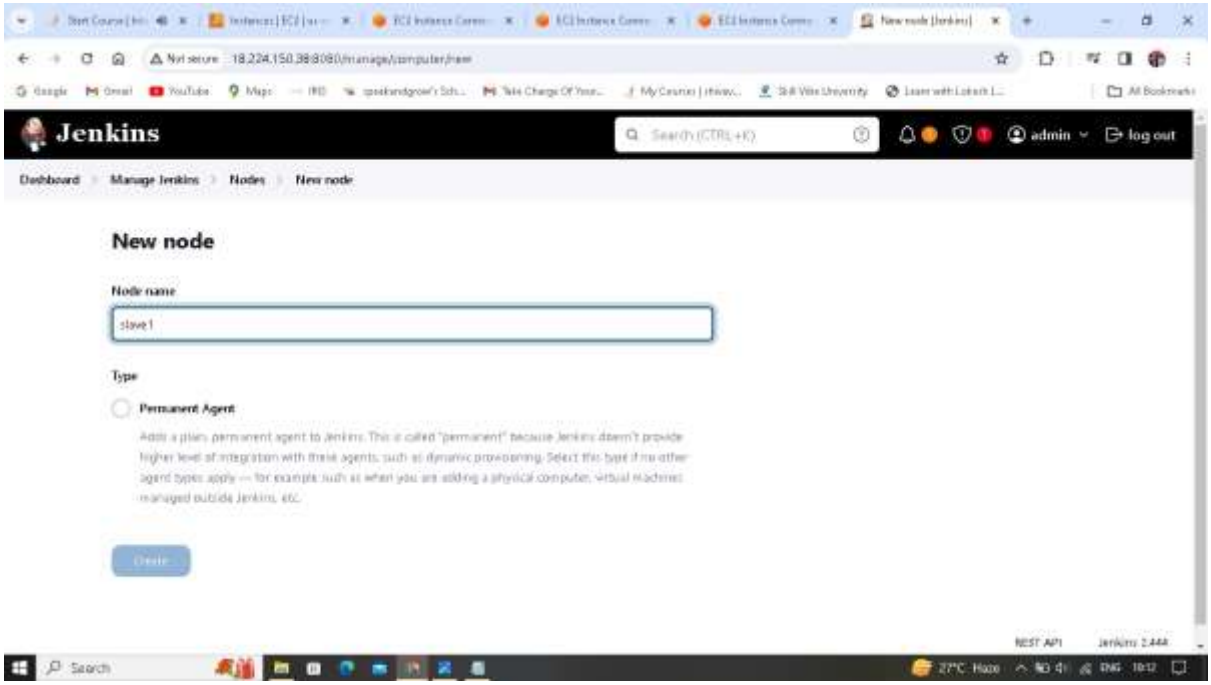
53.Click on Nodes



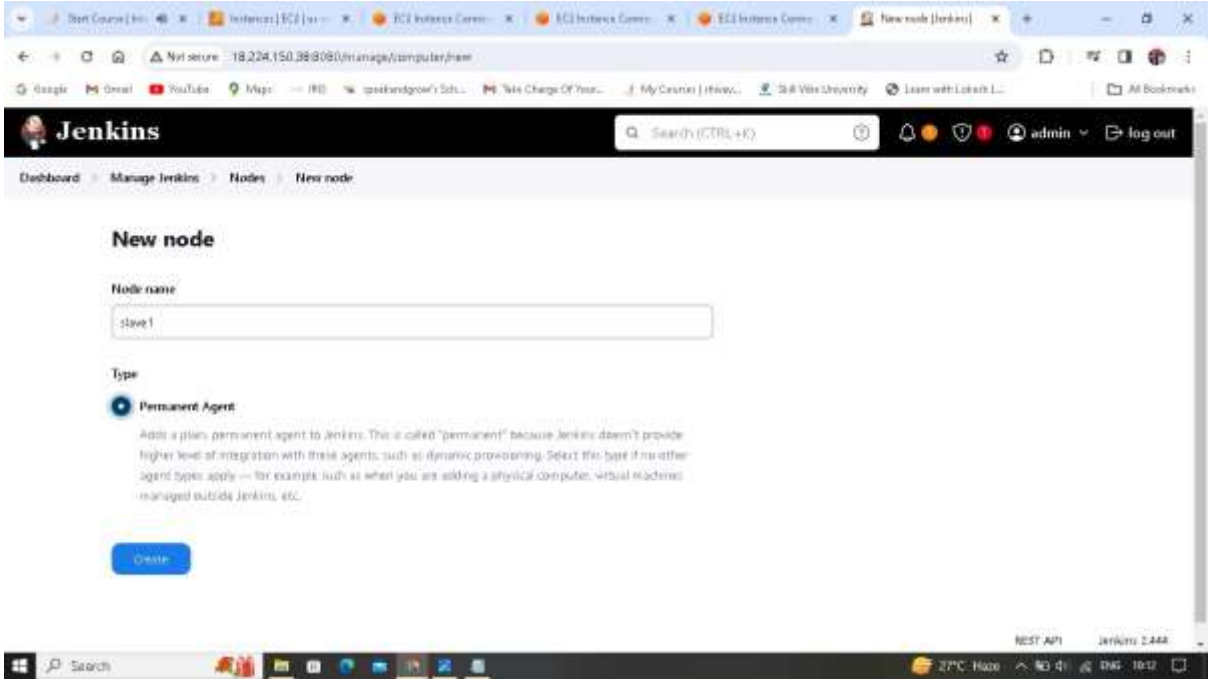
54.Click on +New Node



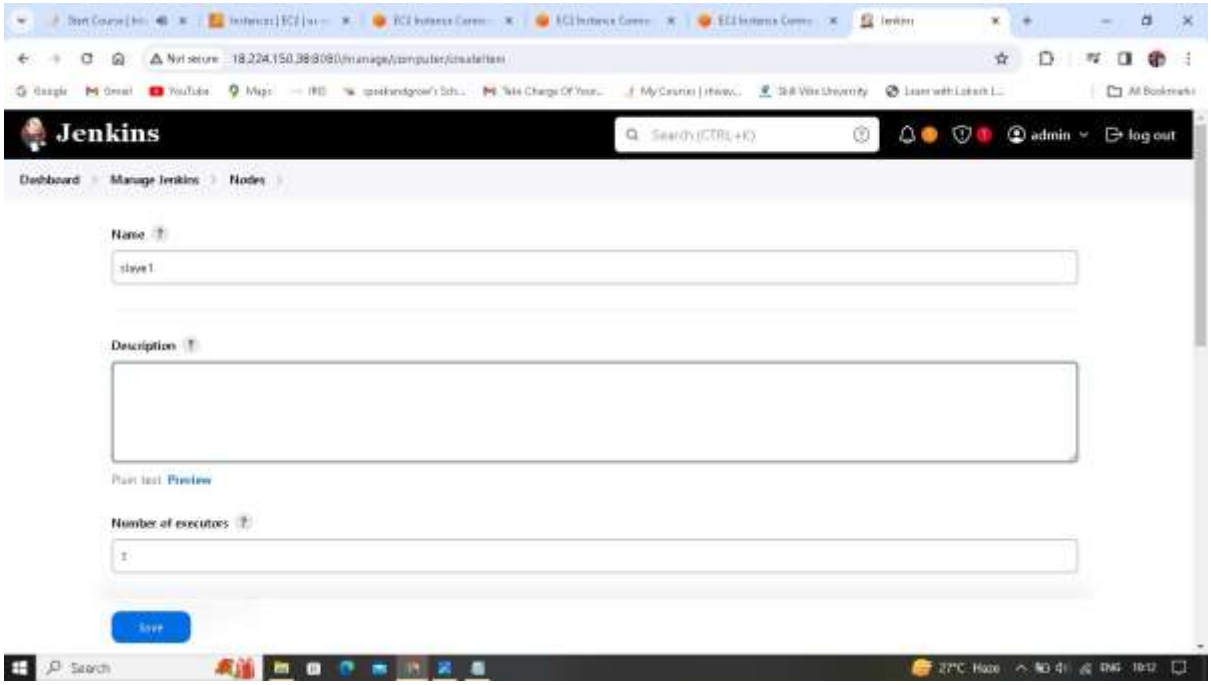
55. Give Node name as slave1



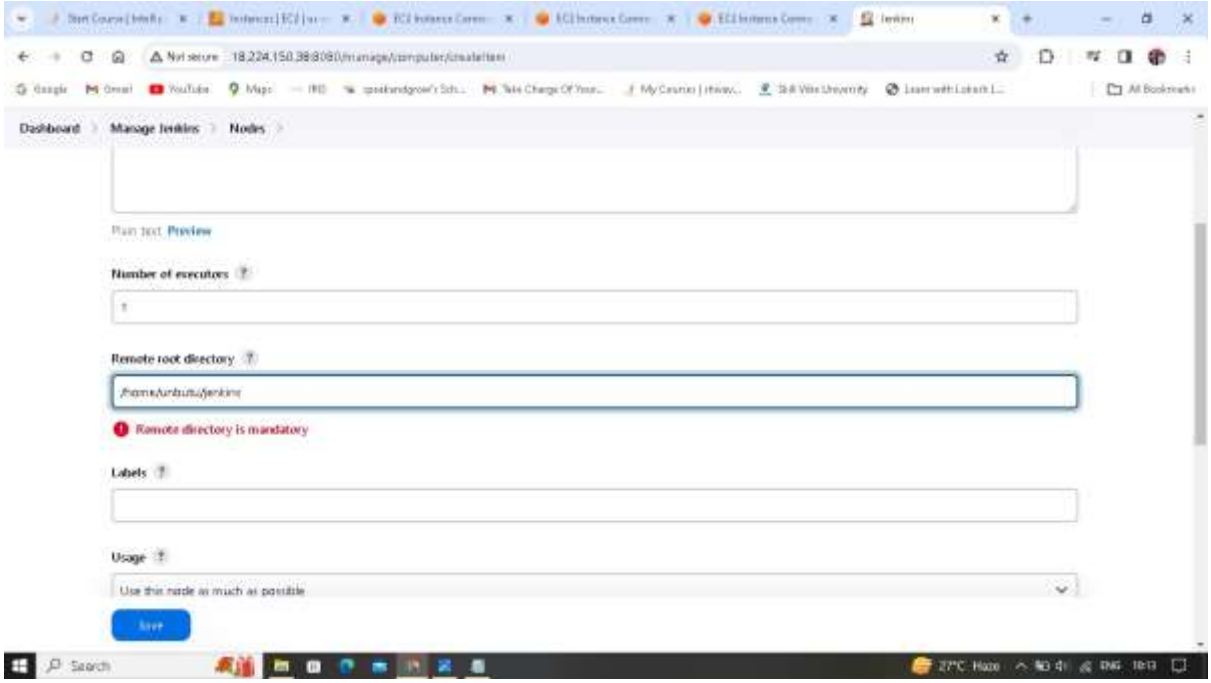
56.Check in Permanent Agent and click on create



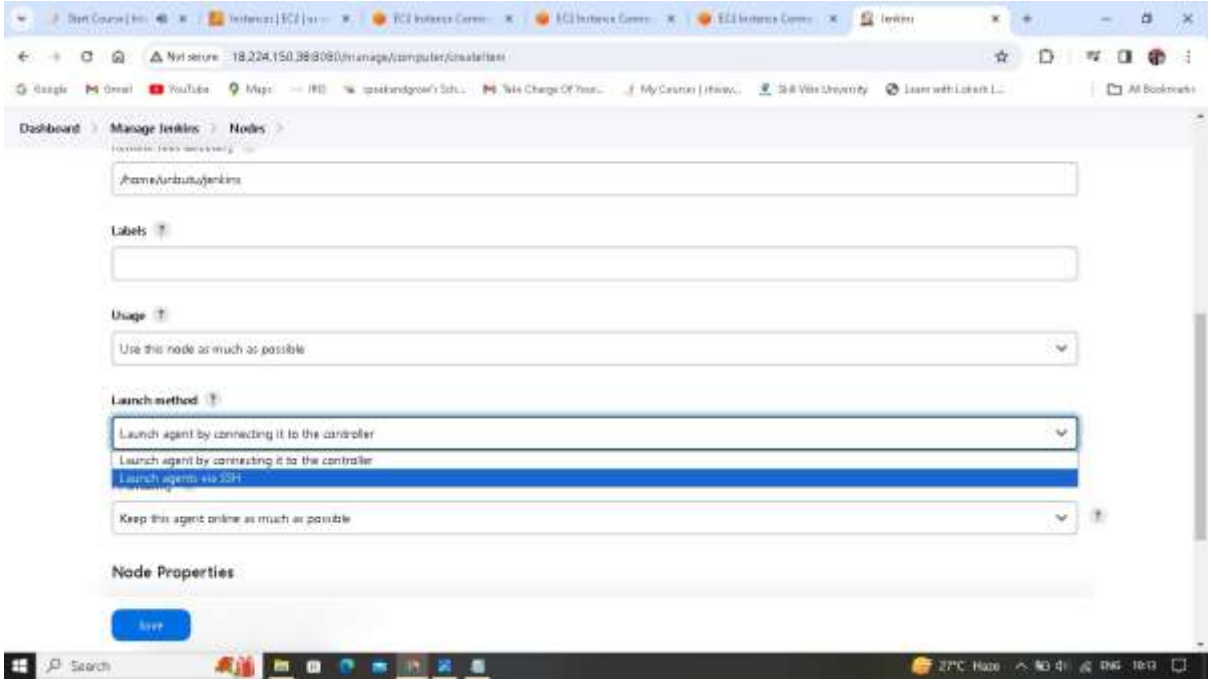
57.Fill one by one



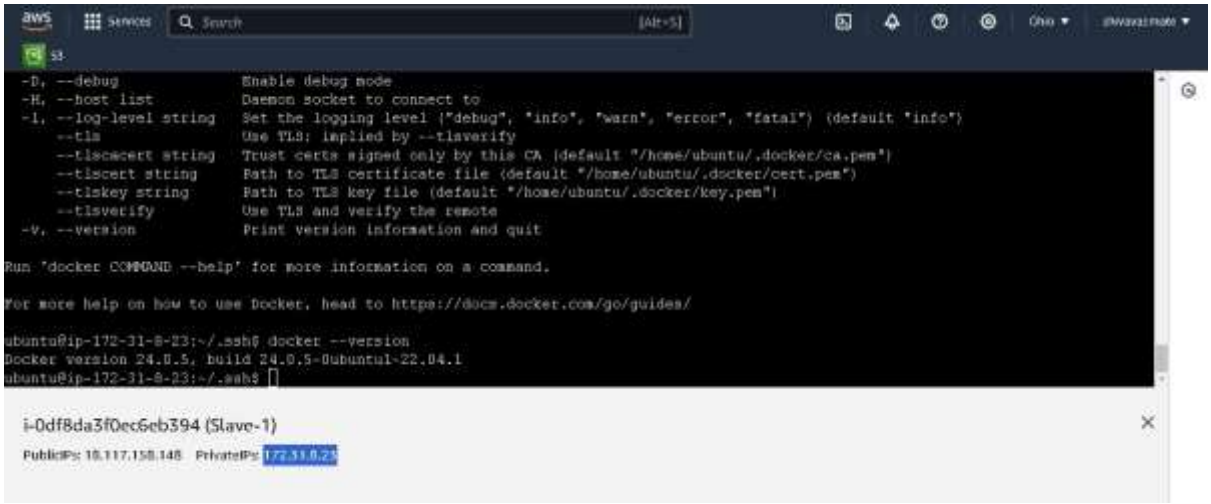
58.In the Remote root directory give /home/ubuntu/jenkins



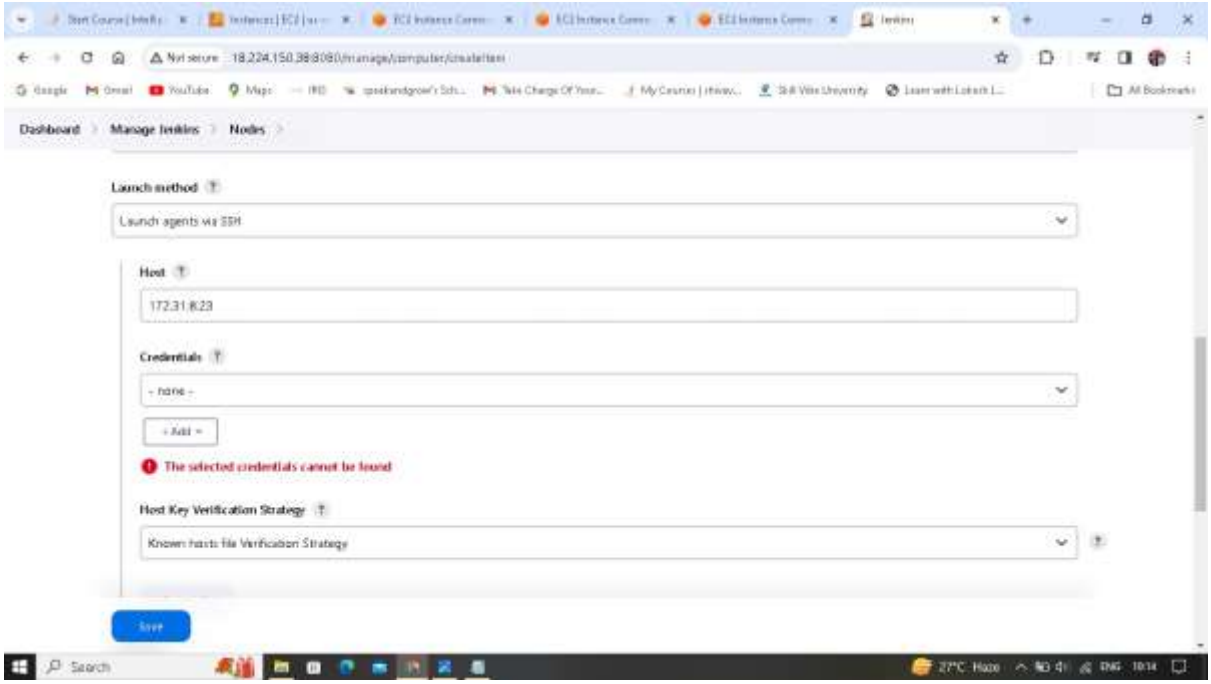
59.In Launch method choose Launch agents via SSH



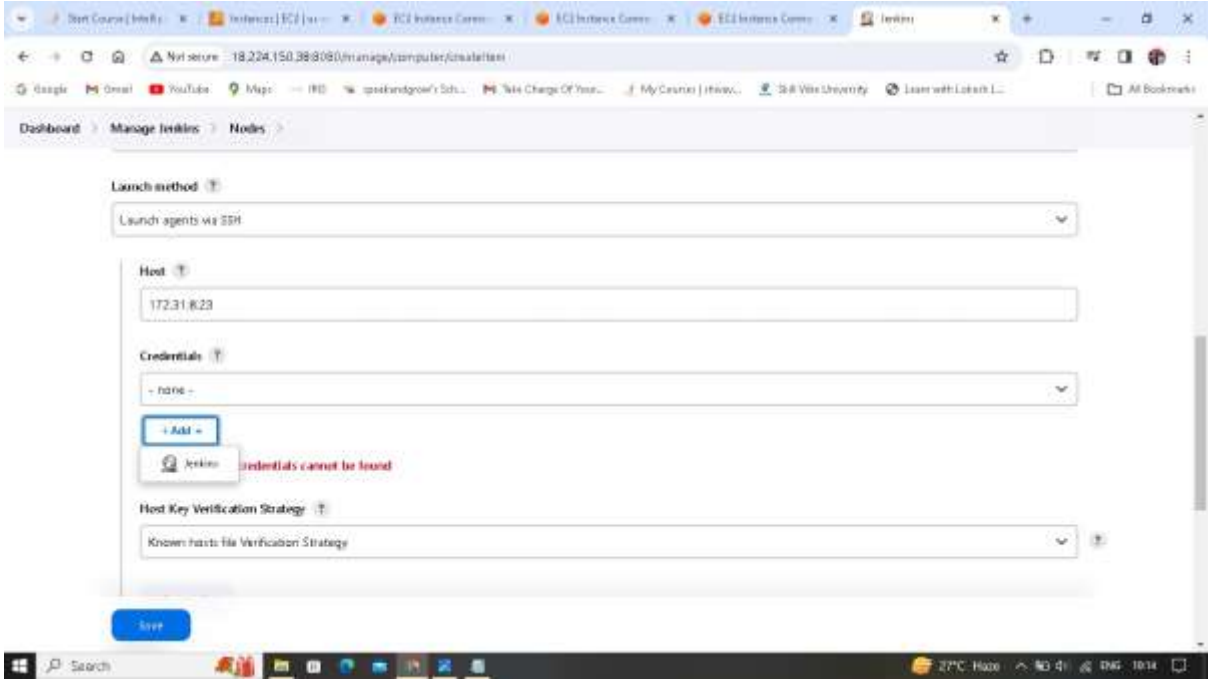
60.Go to Slave1 copy the private IP



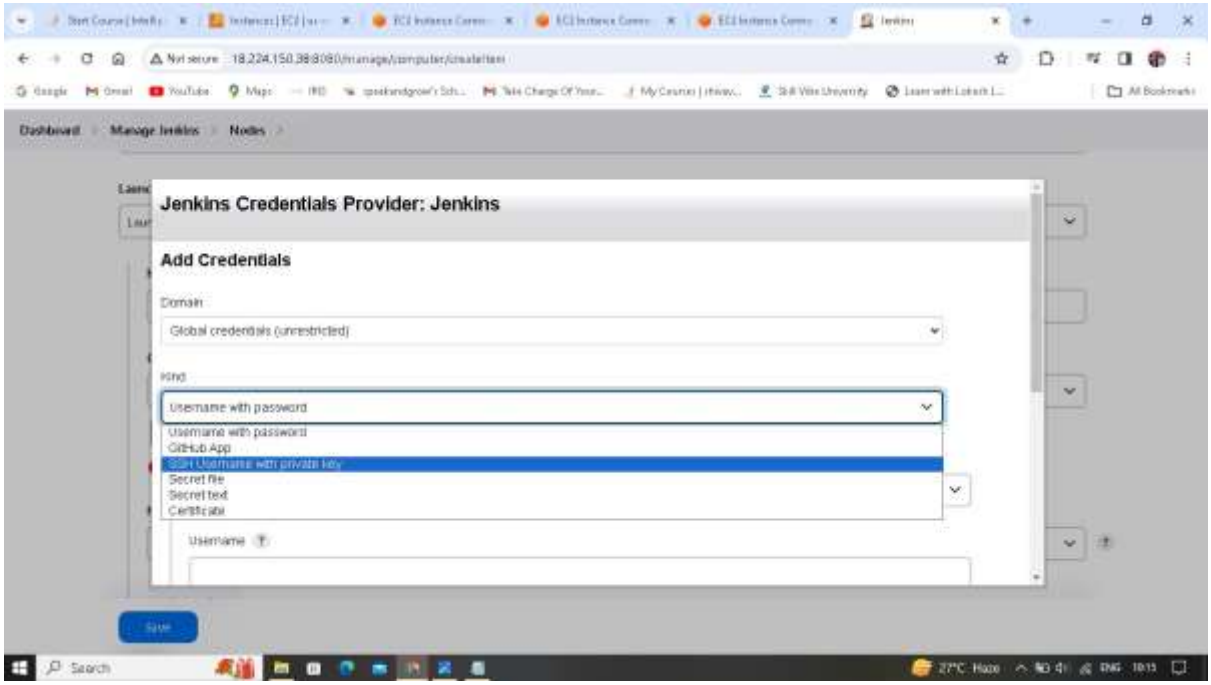
61.Under Host paste it.



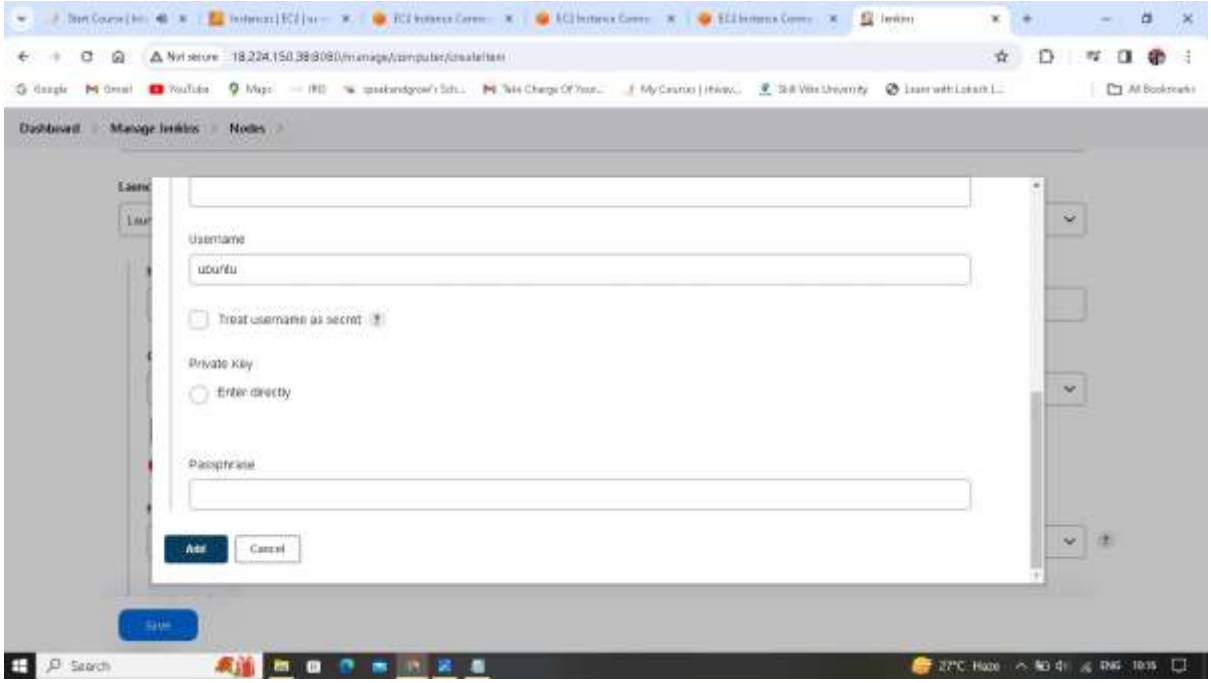
62. Under Credentials click on Add



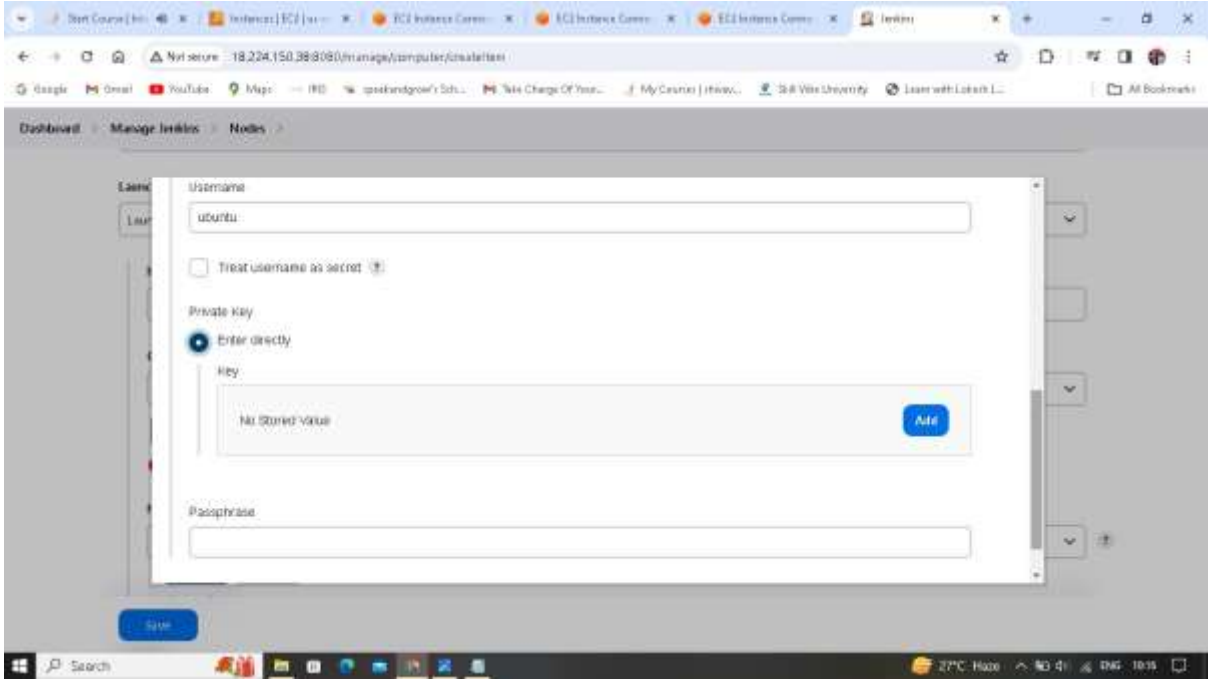
63.Under kind choose SSH Username with private key



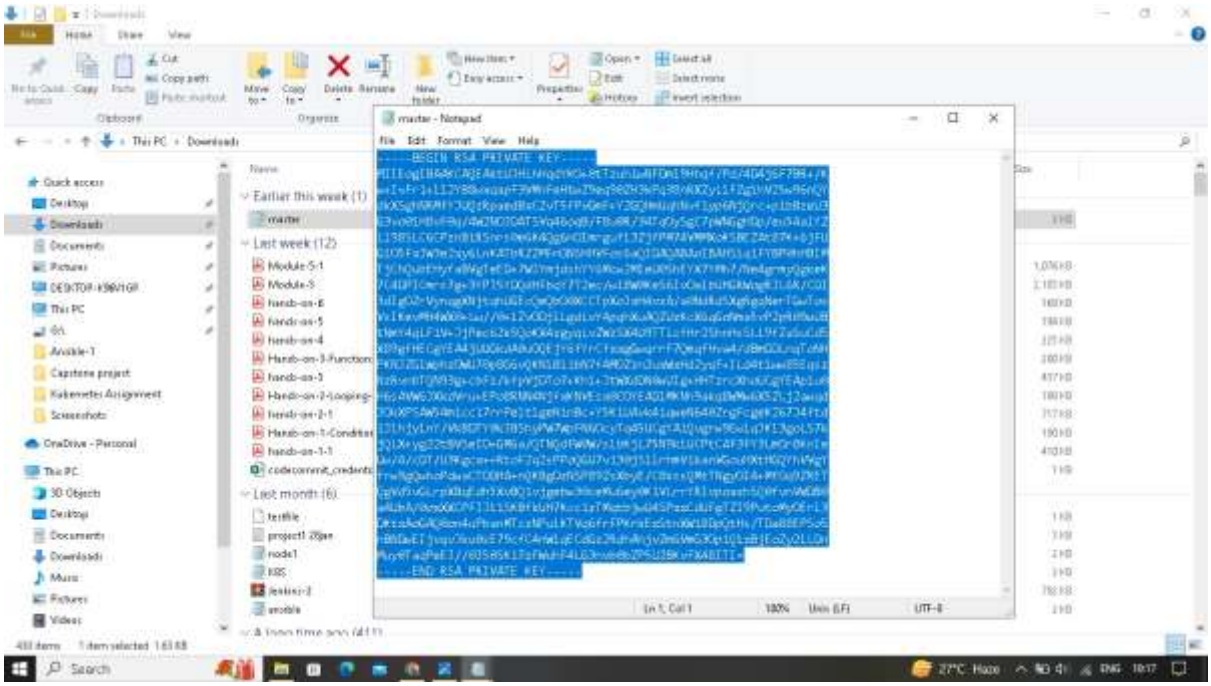
64.Under Username give as ubuntu



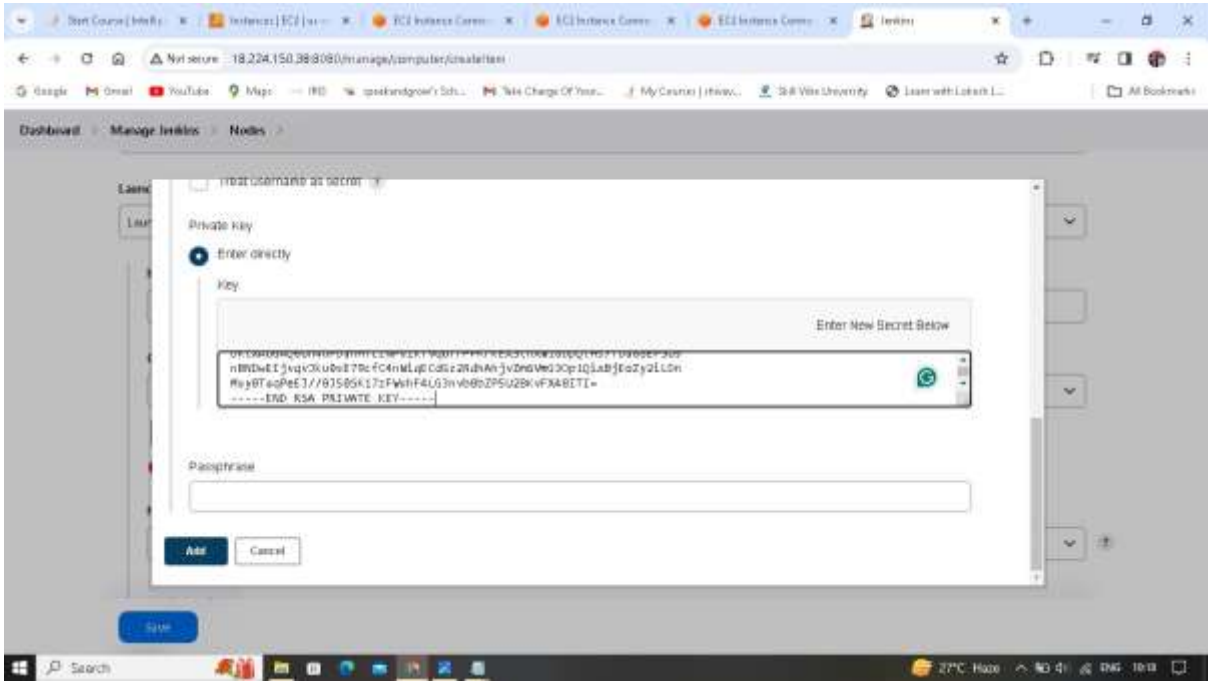
65.Check in Enter directly and click on Add.



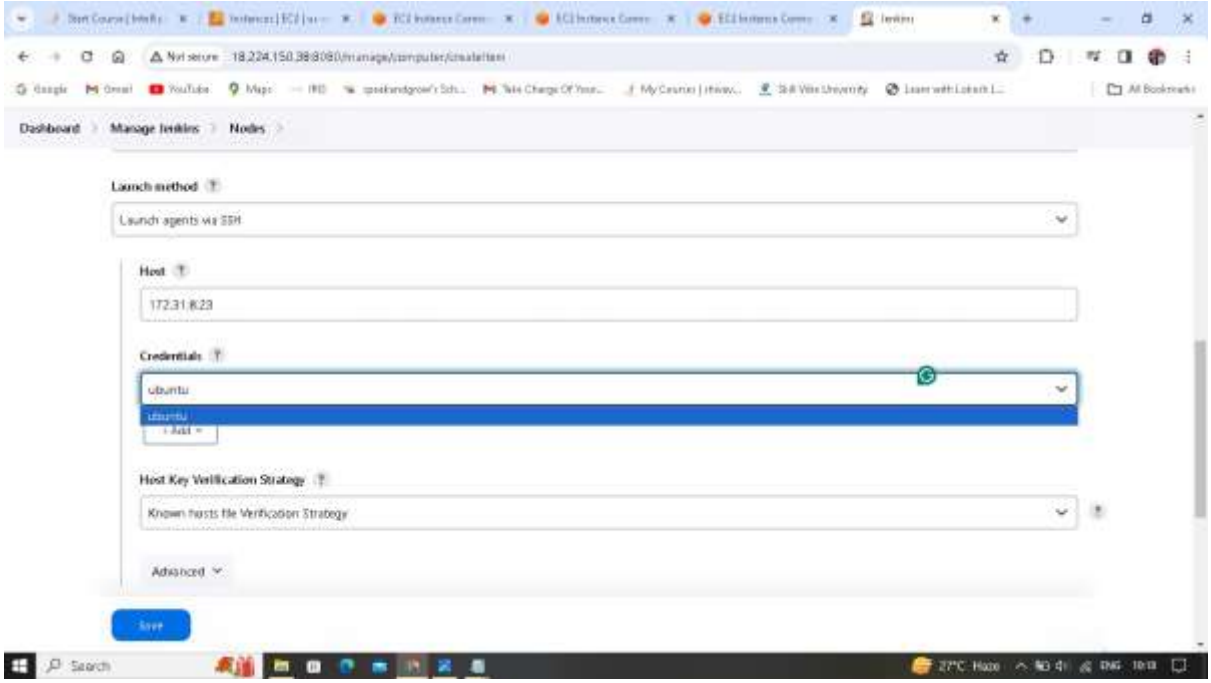
66.Copy the pem file content



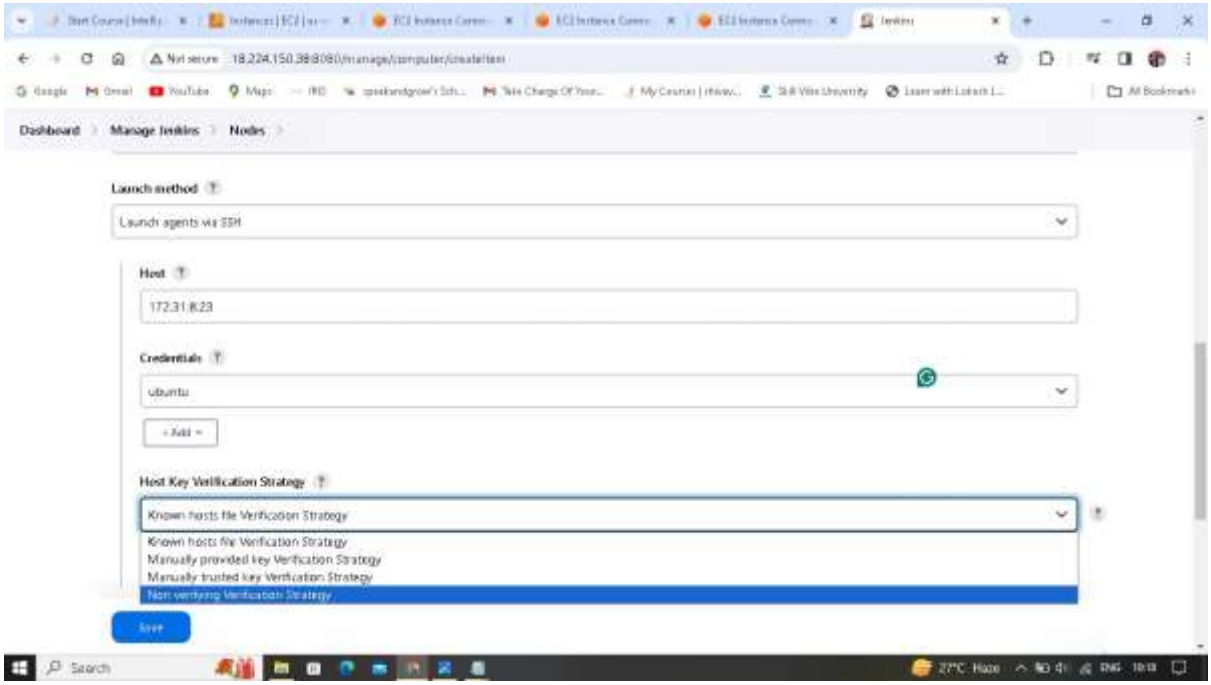
67.Paste it and click on Add.



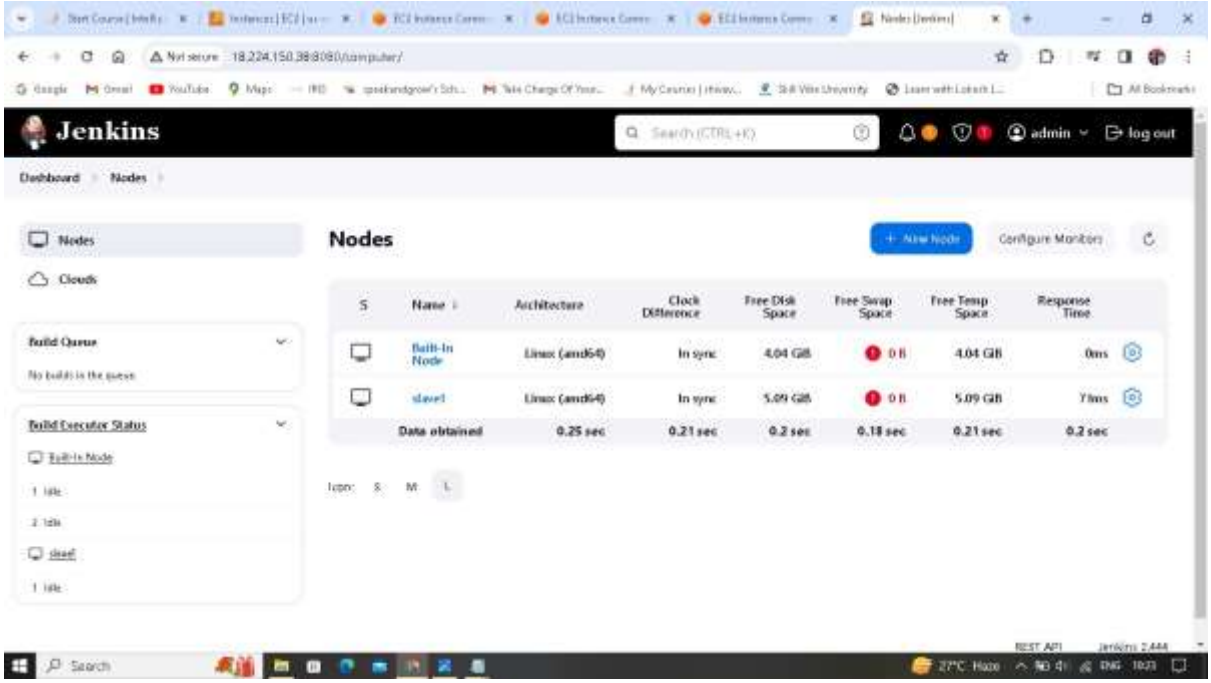
68. Now choose the Credentials as ubuntu



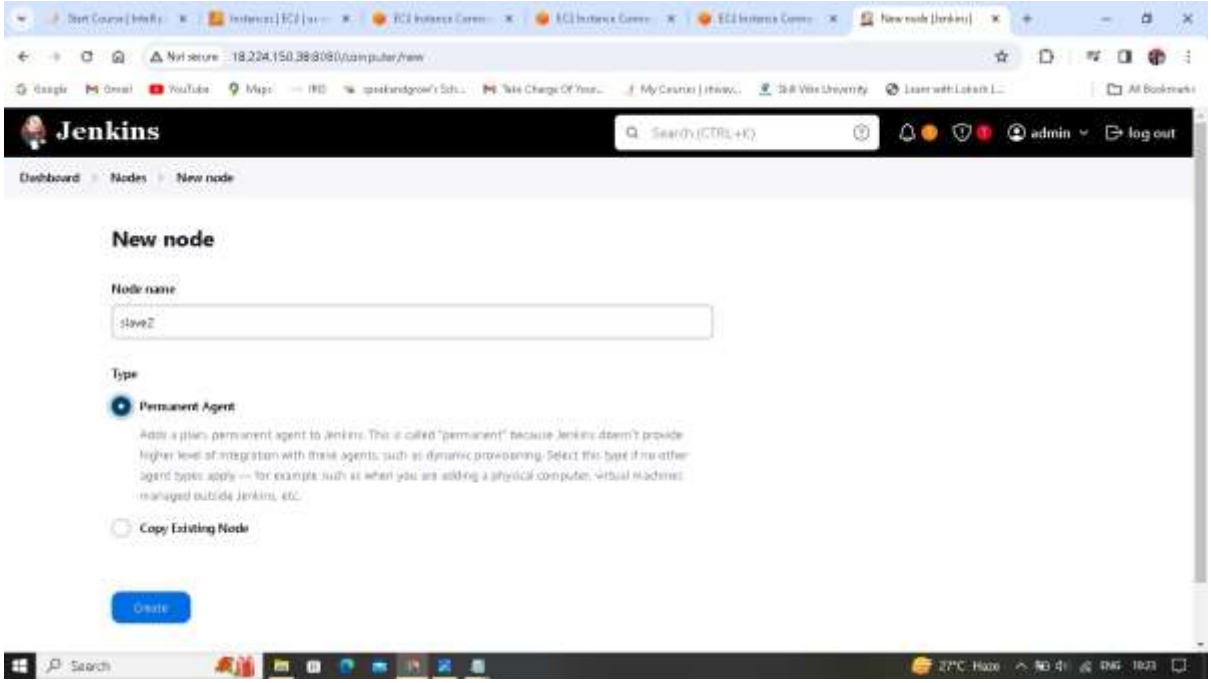
69.Under Host Key Verification Strategy choose Non verifying and click on save



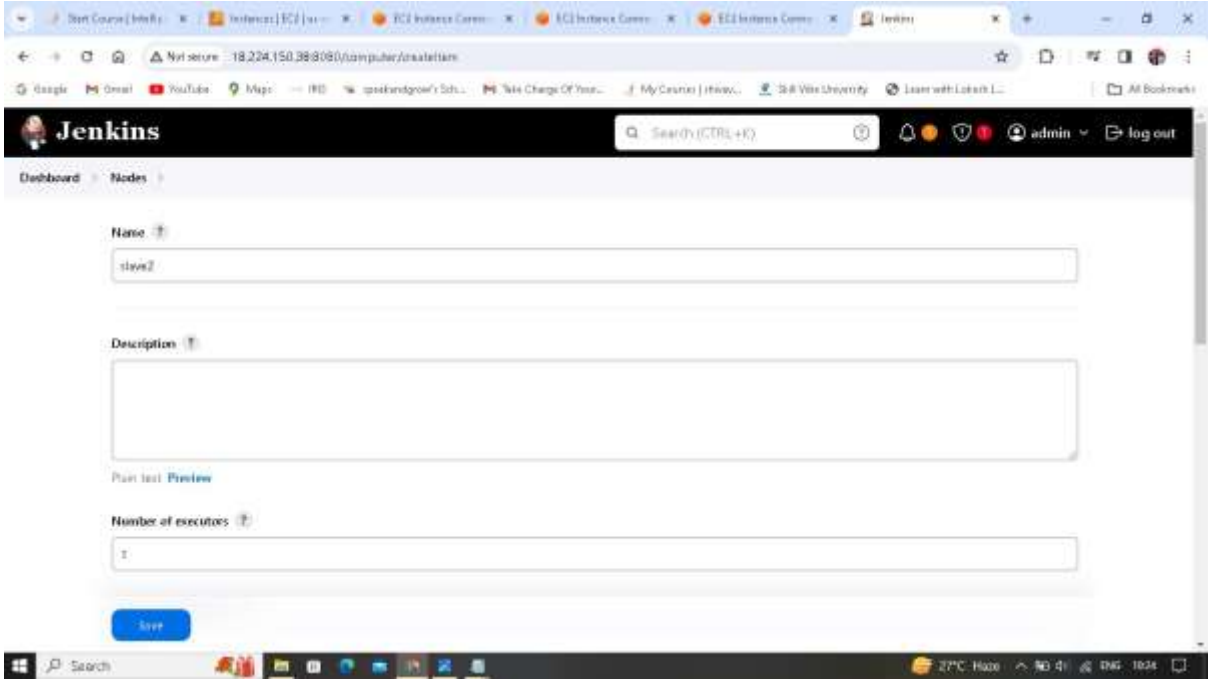
70.The slave1 Node is running and click +New Node



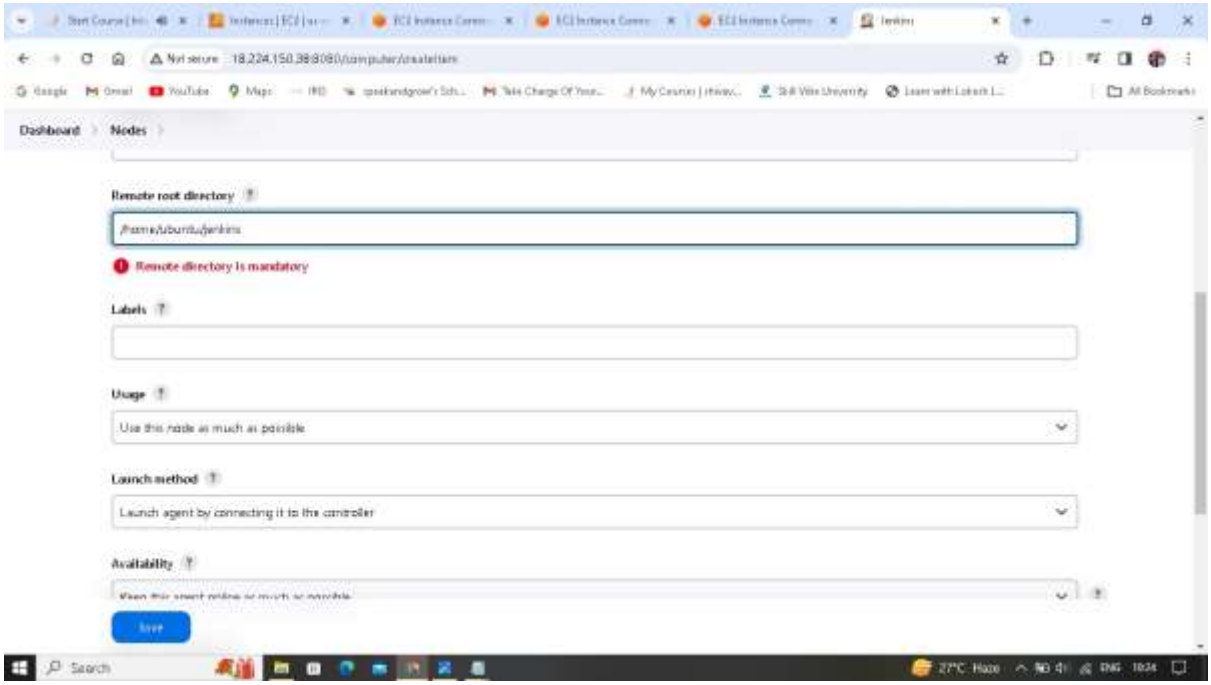
71. Give Node name as slave2, Permanent Agent type and click on create.



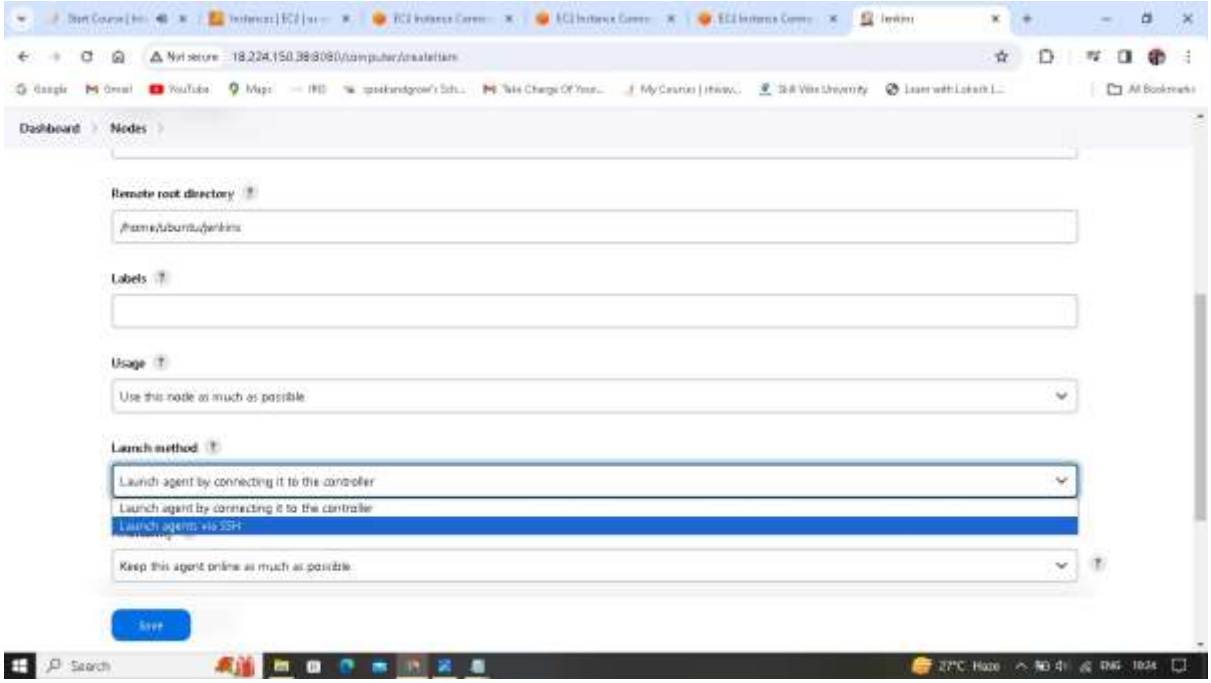
72.Fill it



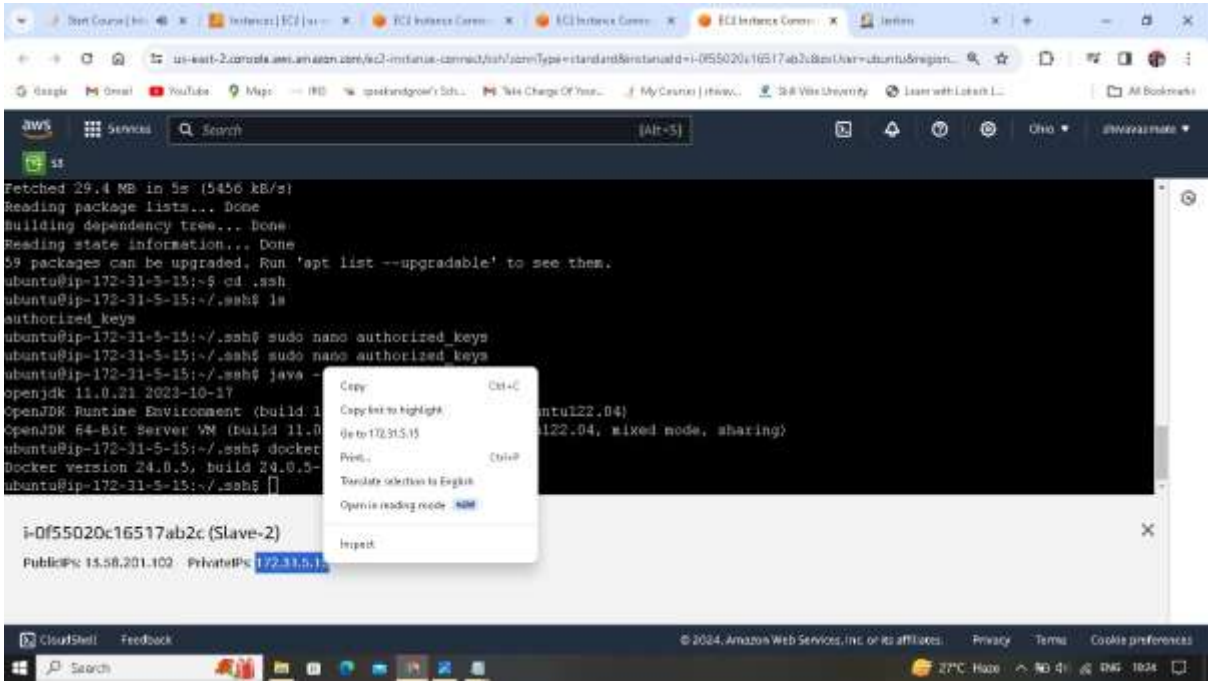
73. In the Remote root directory give /home/ubuntu/jenkins



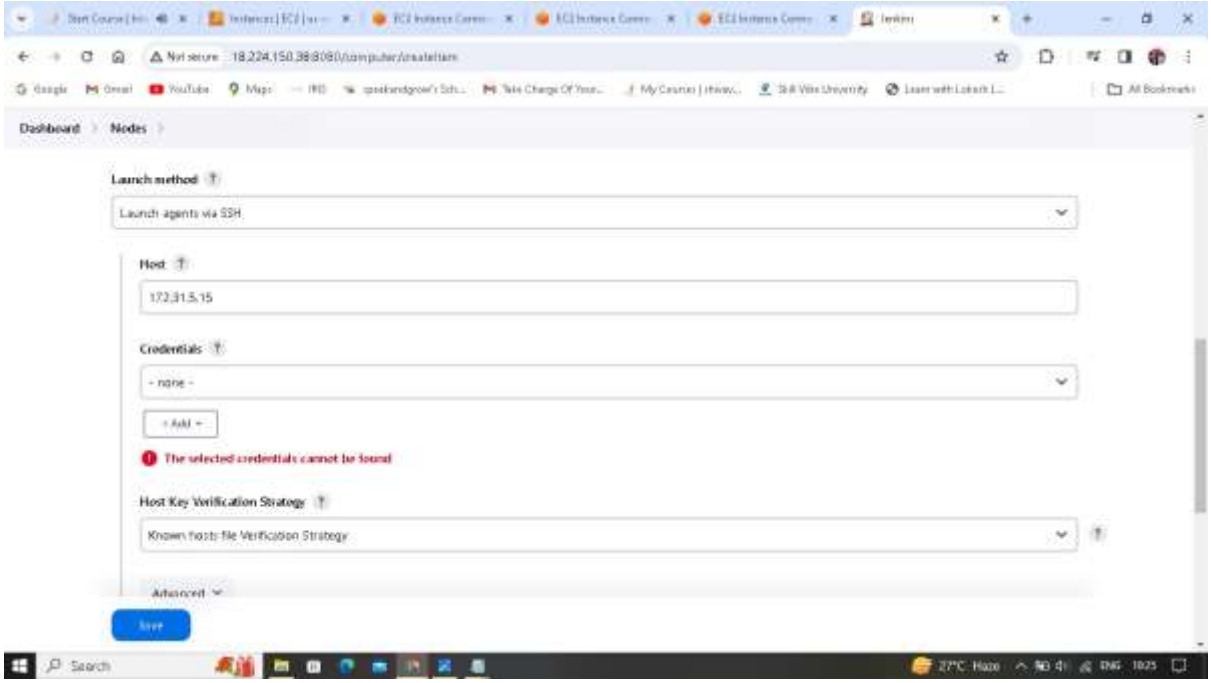
74. In Launch method choose Launch agents via SSH



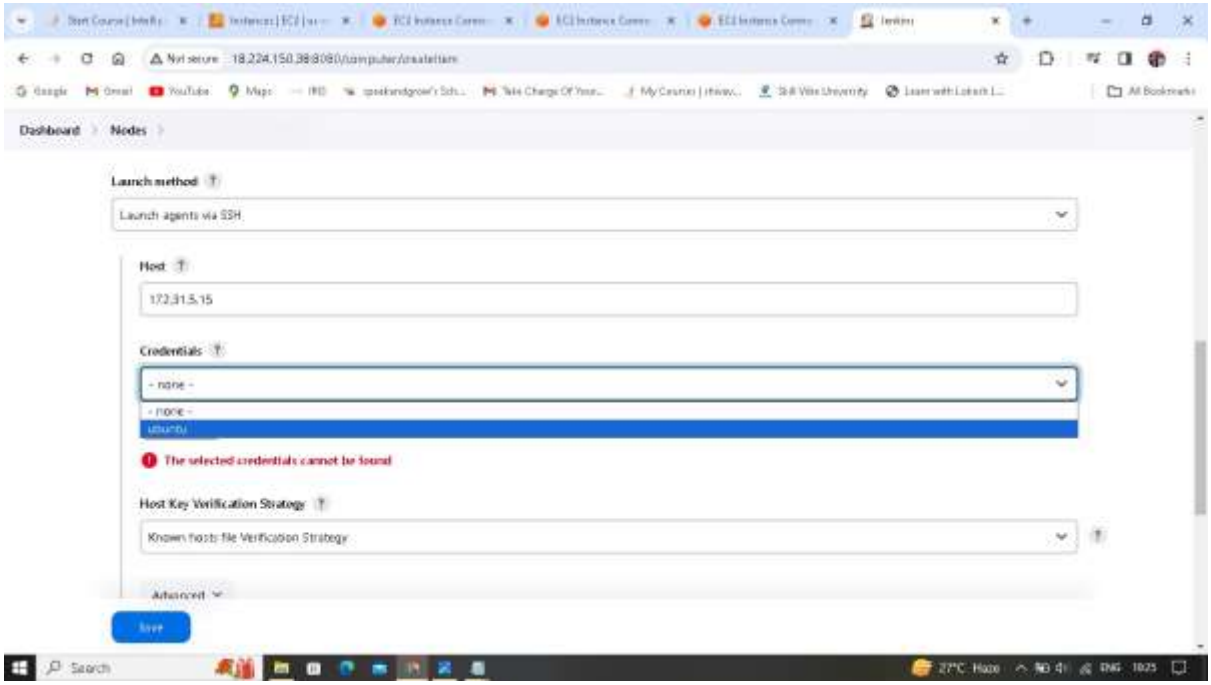
75. Go to Slave2 copy the private IP



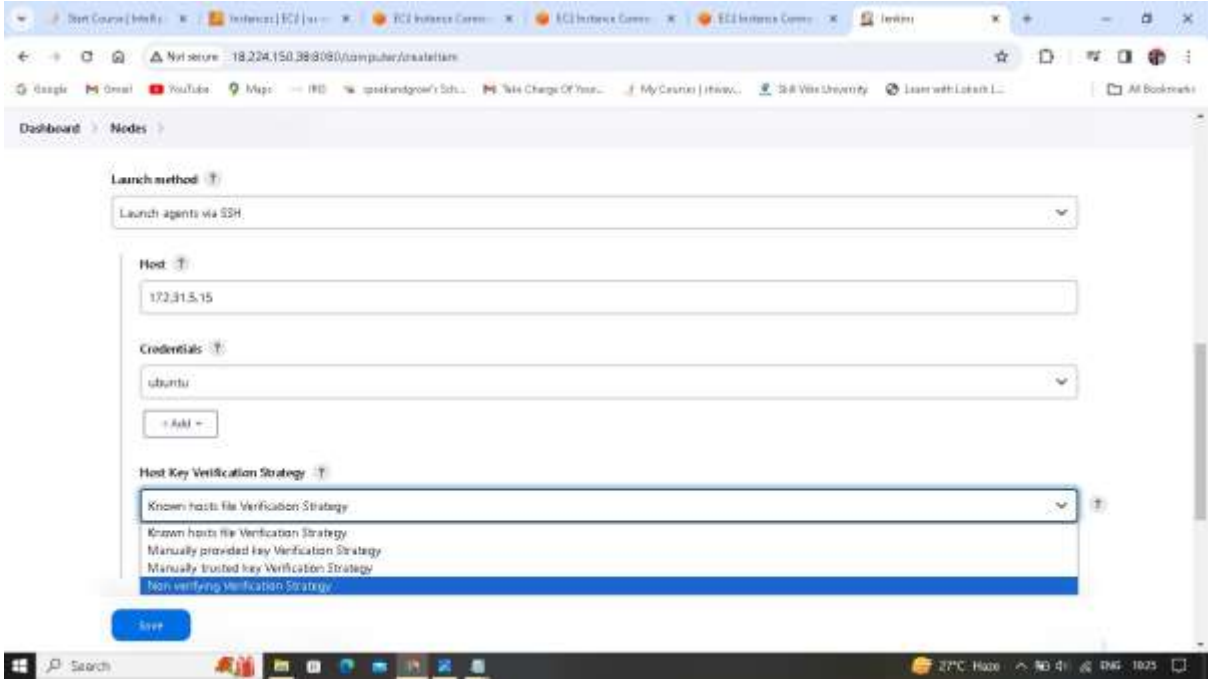
76. Under Host paste it.



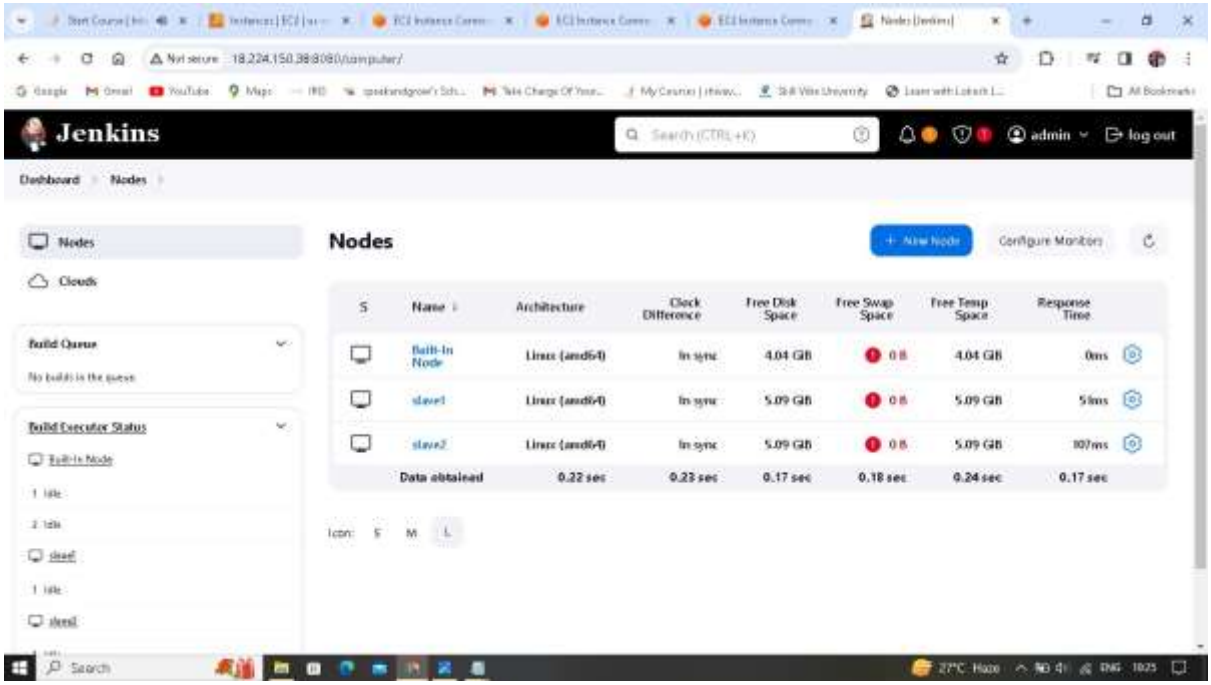
77.This time just choose ubuntu for Credentials.



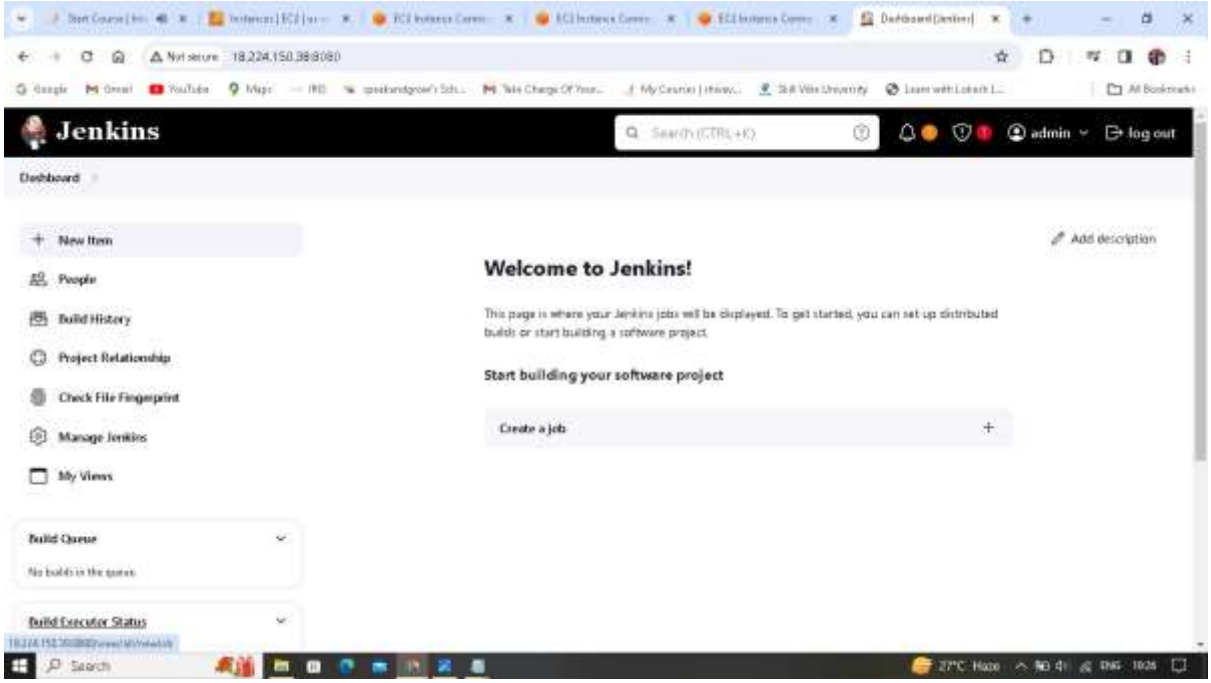
78. Under Host Key Verification Strategy choose Non verifying and click on save



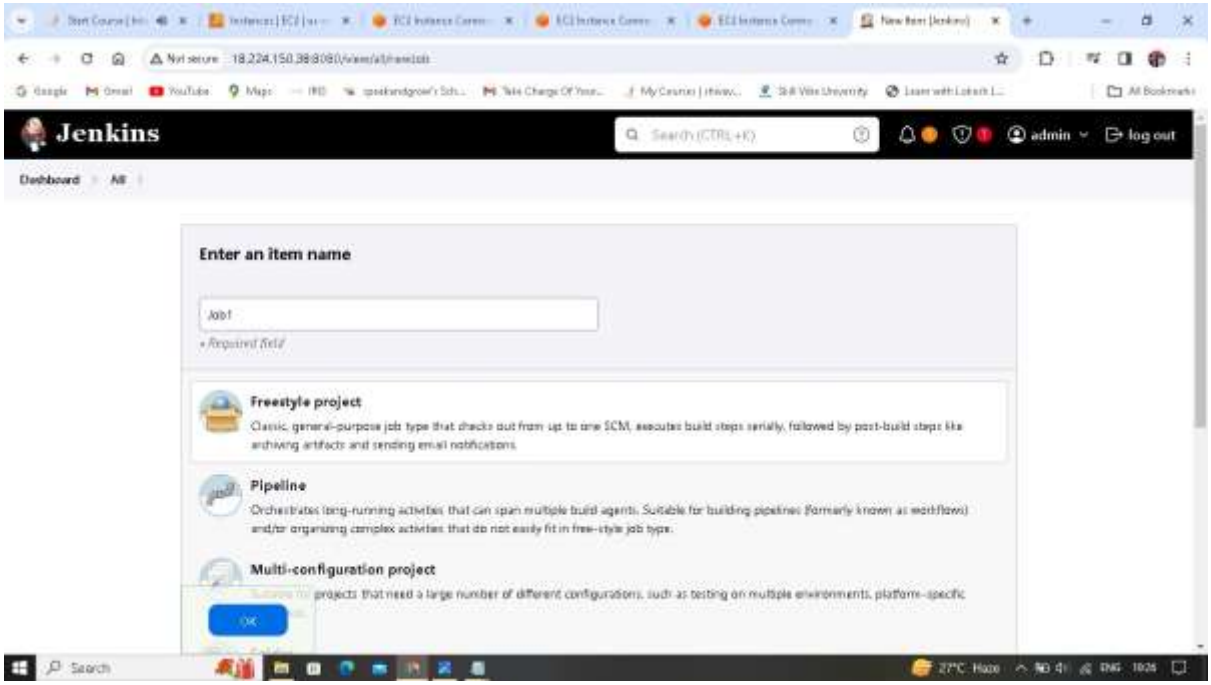
79.Now slave2 also running.



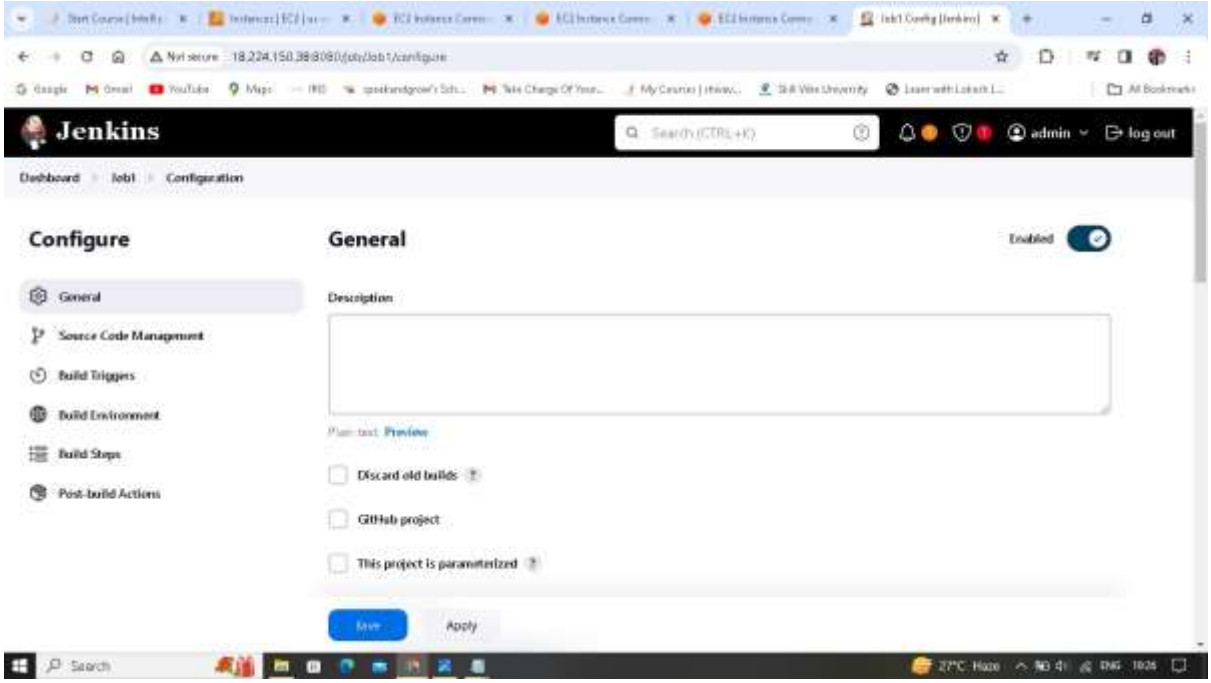
80.Go to Dashboard click on New item to create job.



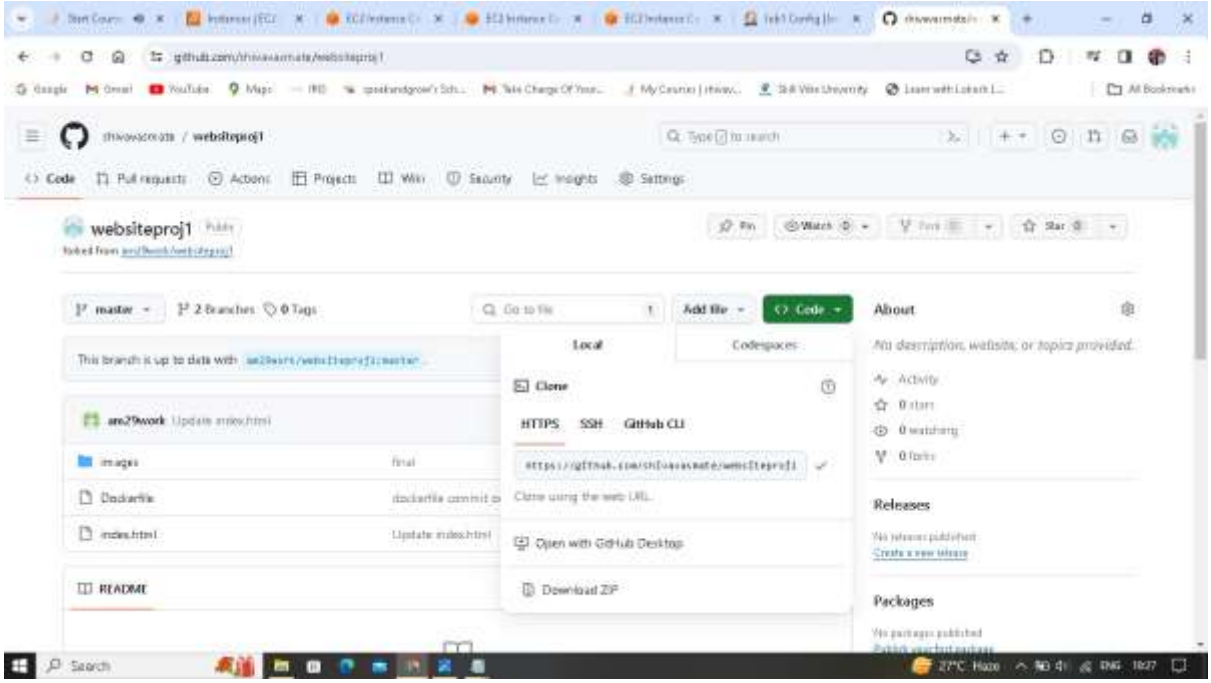
81.Enter an item name as Job1 and choose Freestyle project click on ok.



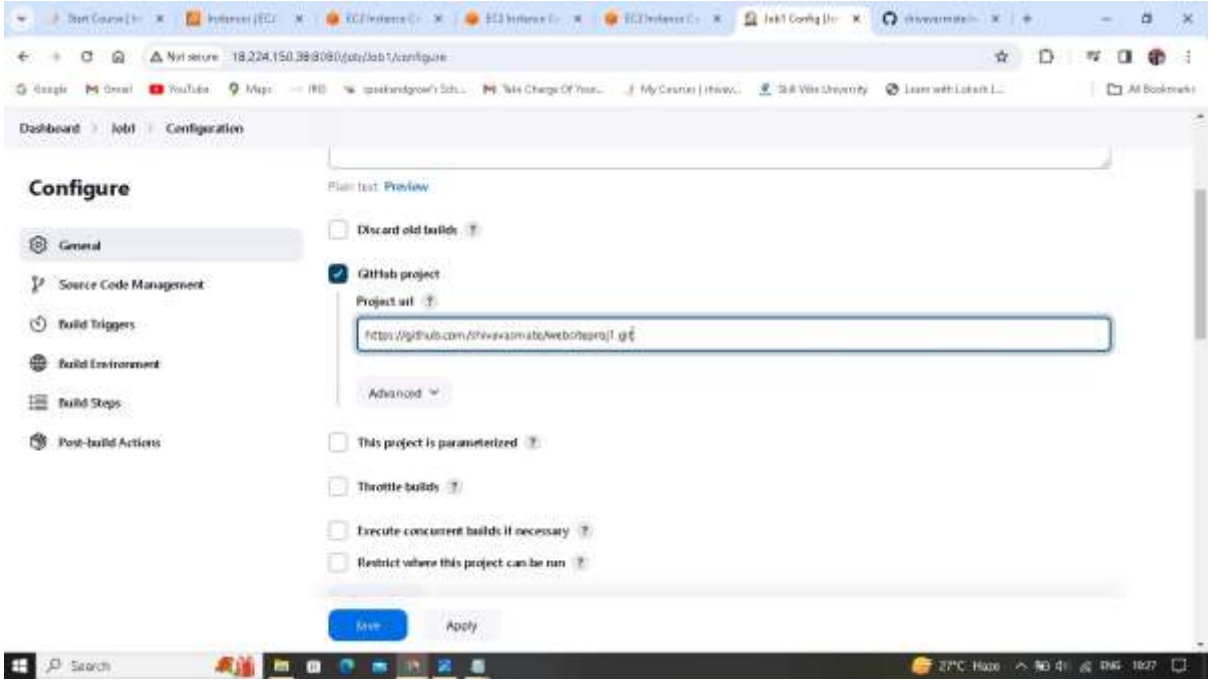
82.Fill one by one. check in GitHub project.



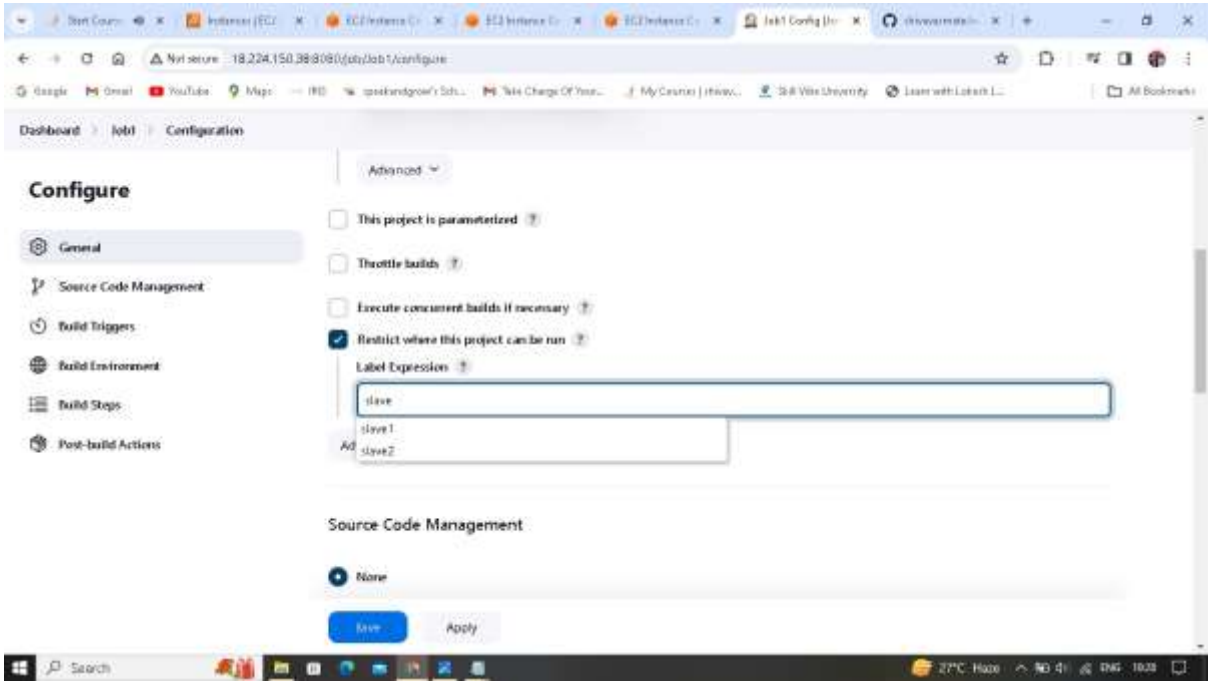
83.Go to github copy the url



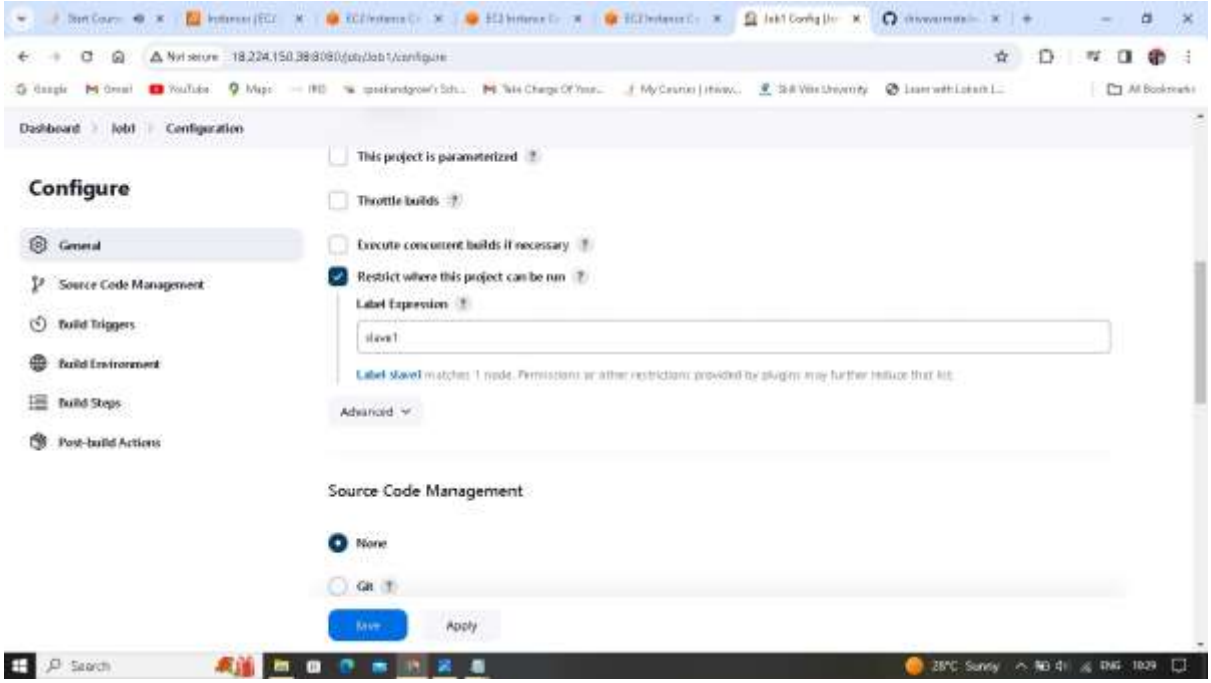
84.Paste it



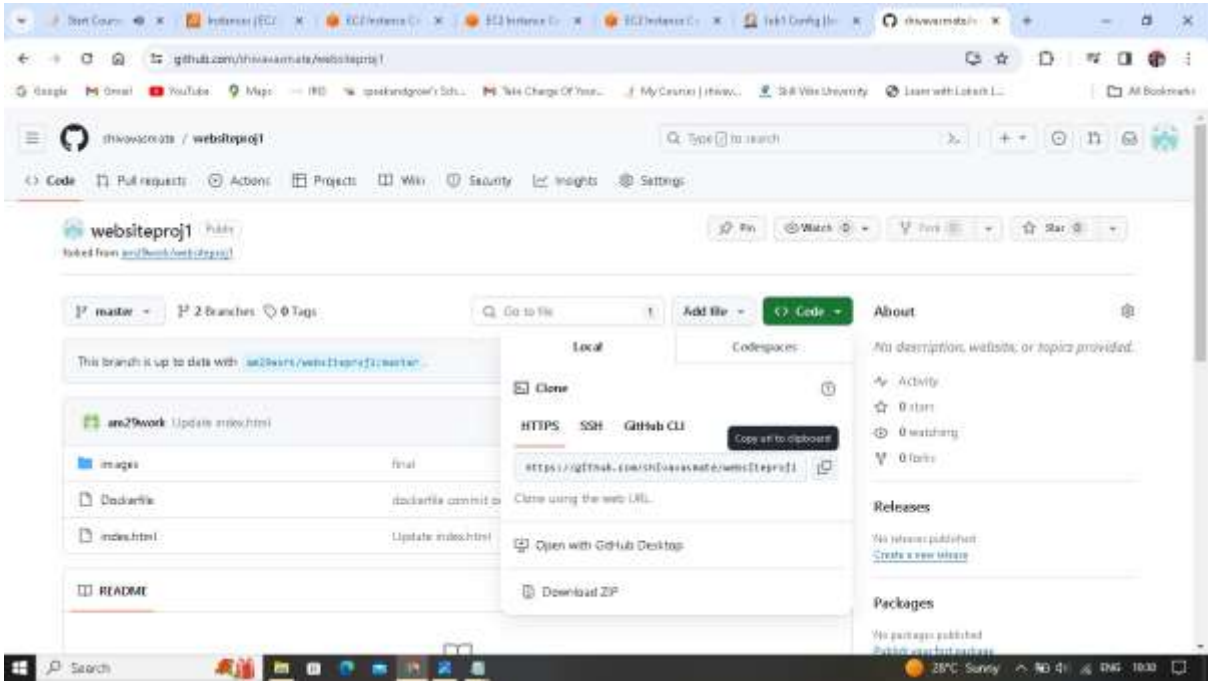
85.Under Restrict where this project can be run type slave1.



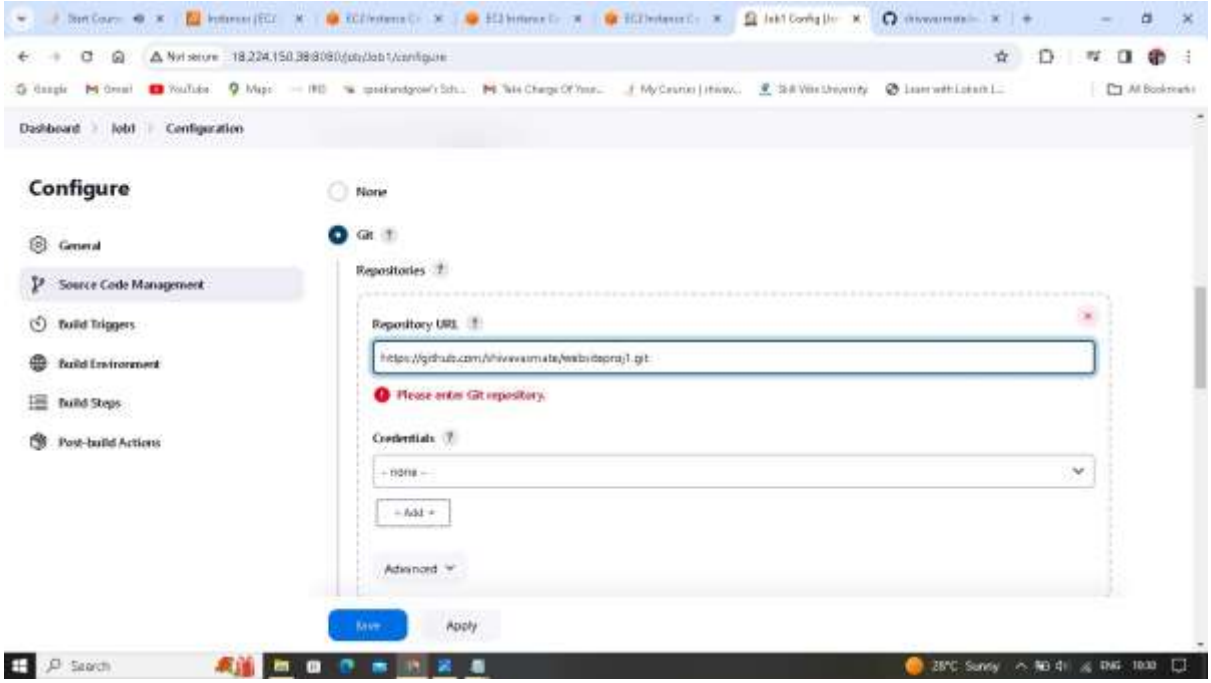
86.Blue colour will appear as Label slave1 and under source code management check in Git.



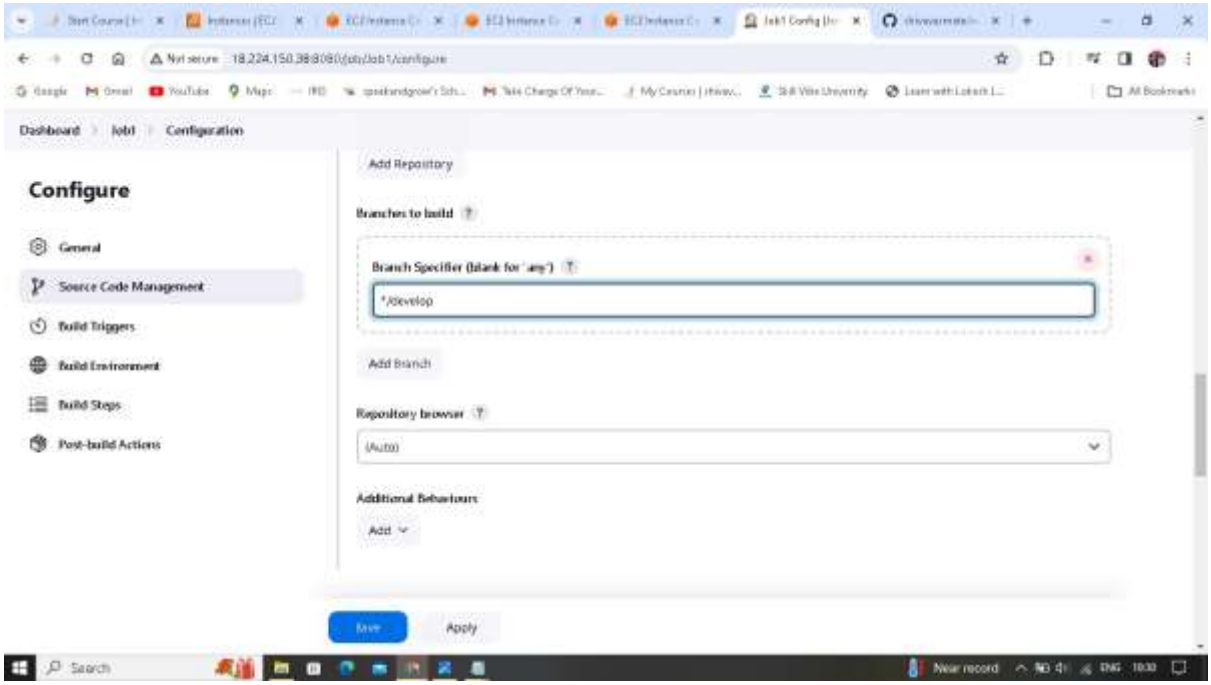
87.Copy the URL for github.



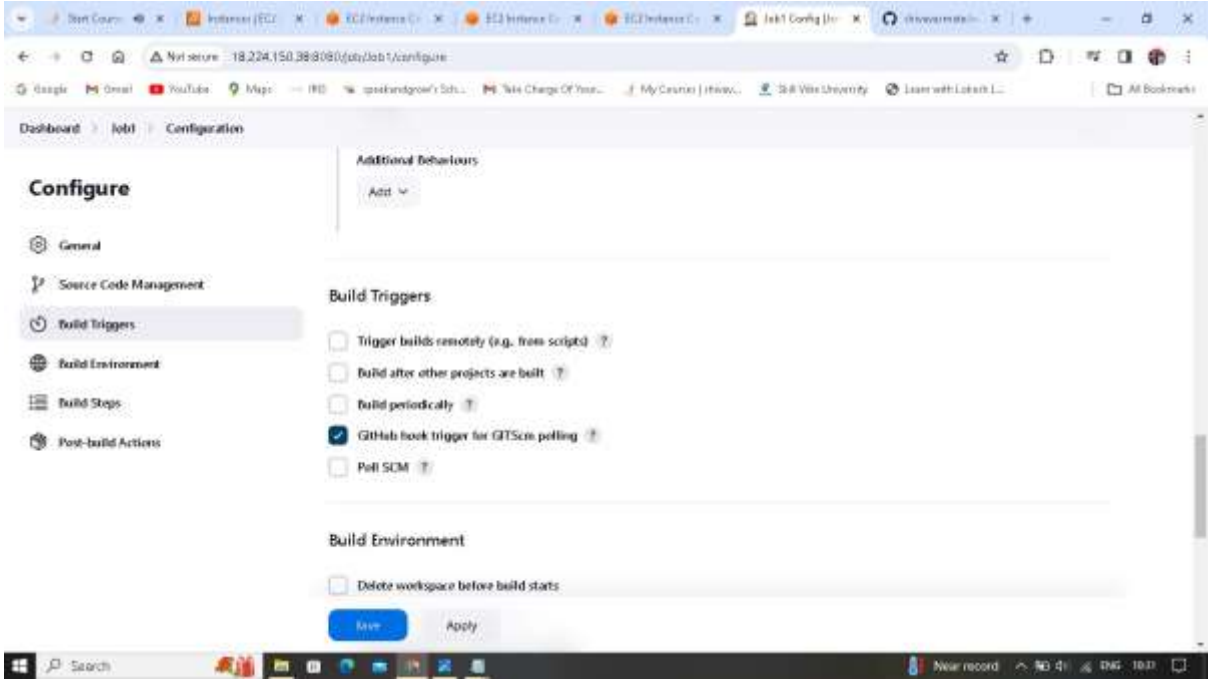
88.Paste it in Repository URL



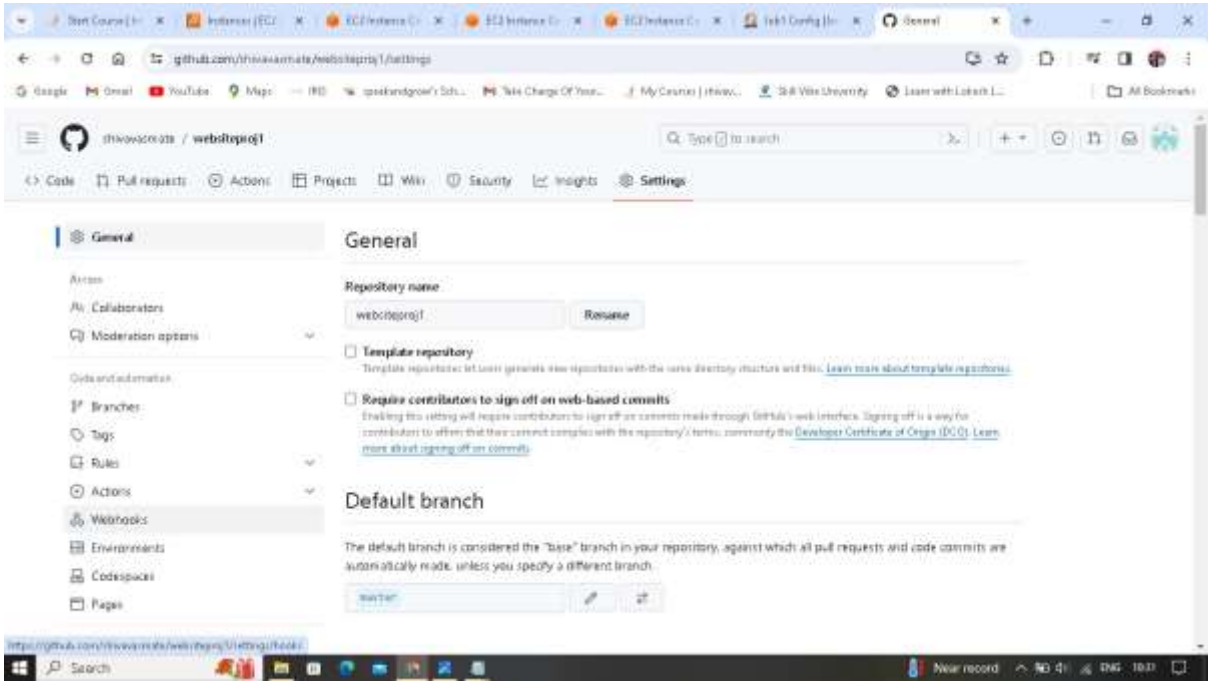
89.Under Branches to build choose */develop as Branch Specifier.



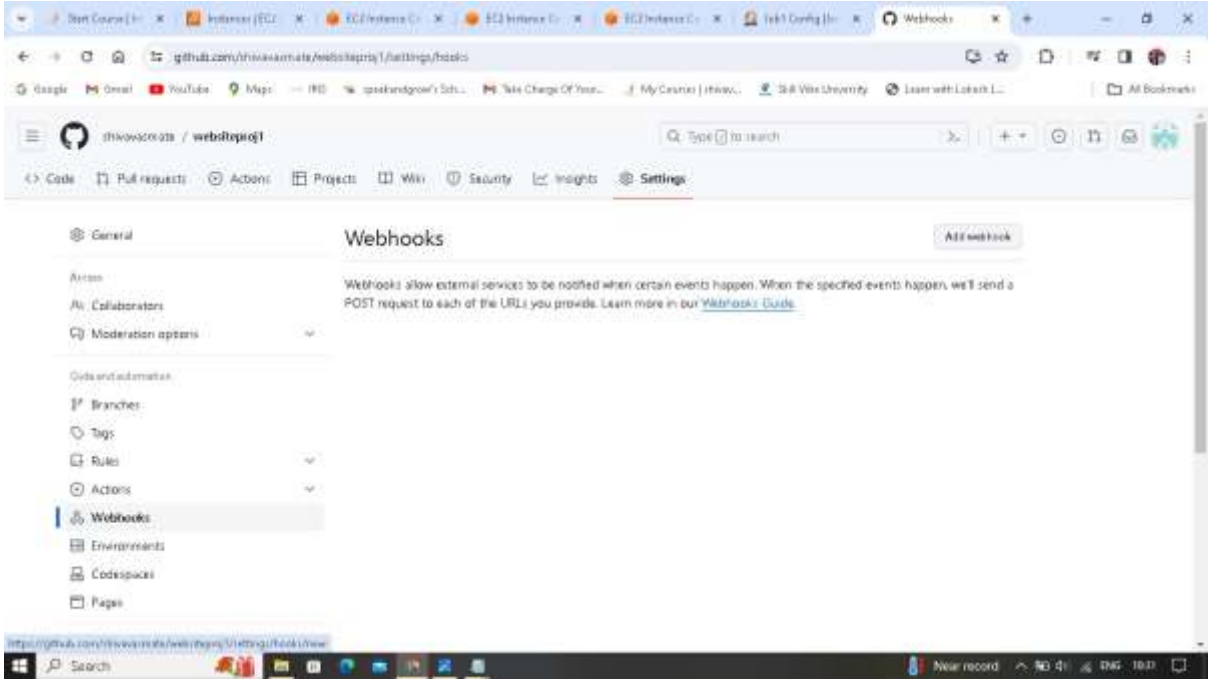
90.Under Build Triggers choose GitHub hook trigger for GIT Scam polling.



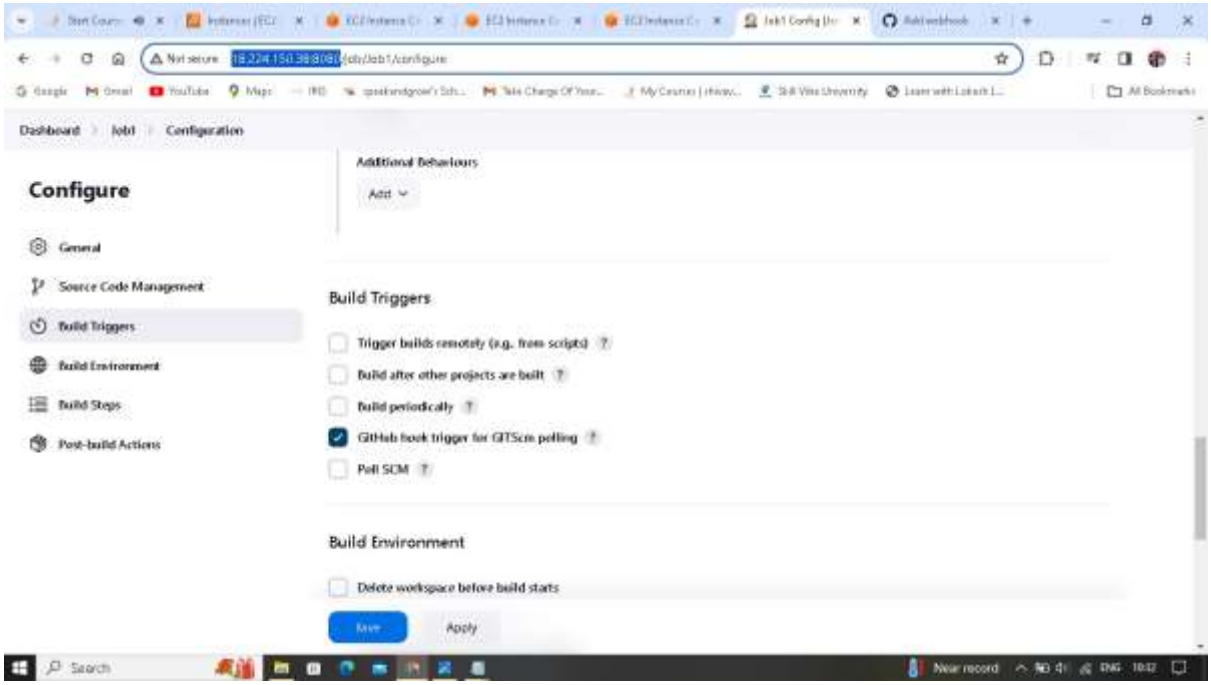
91.For it go to github under repository setting



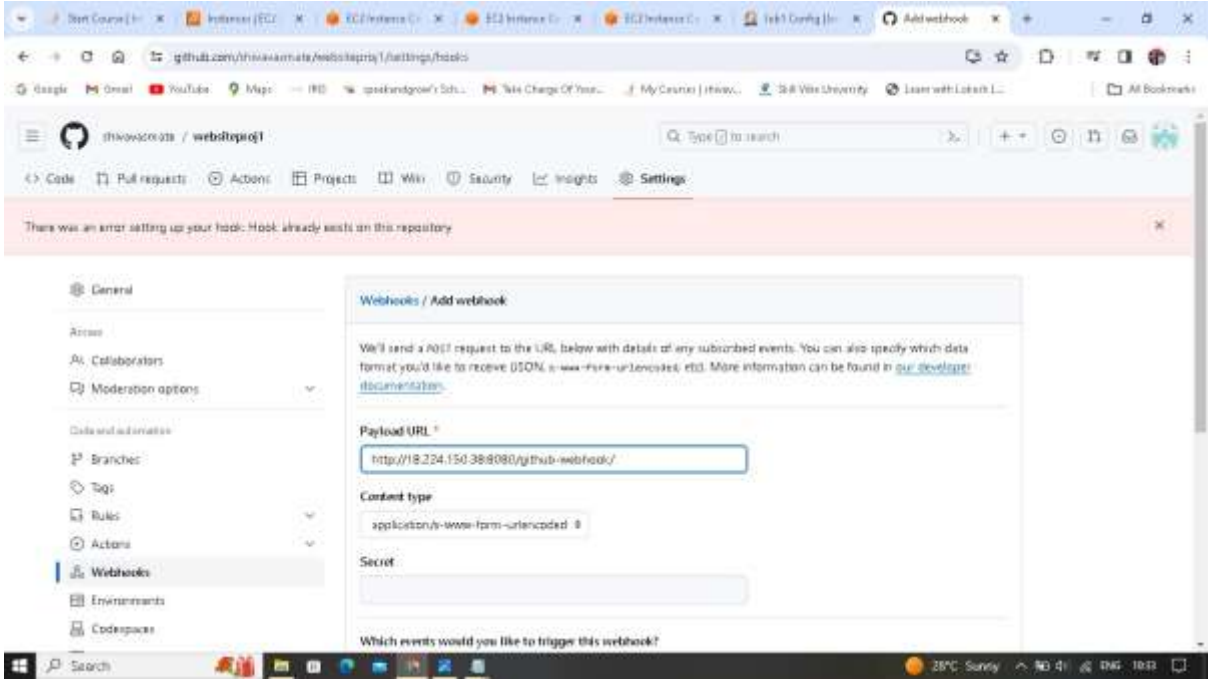
92. On left side click on Webhooks under it click on add webhook.



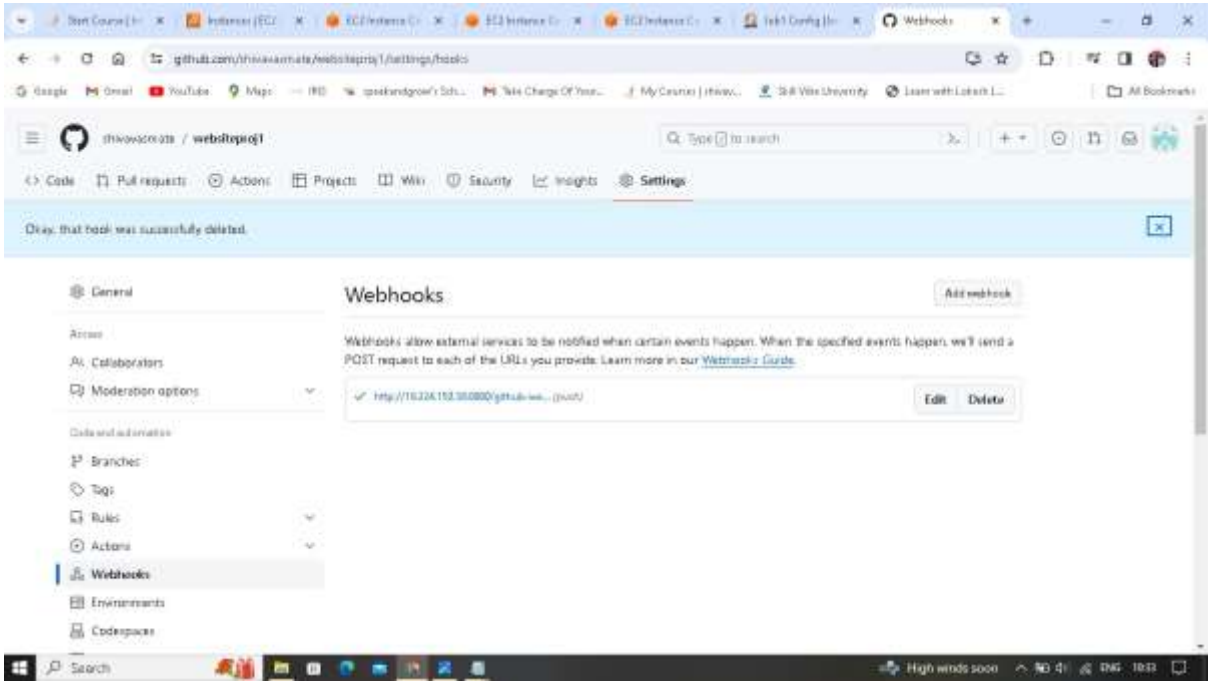
93.Copy the URL of jenkins



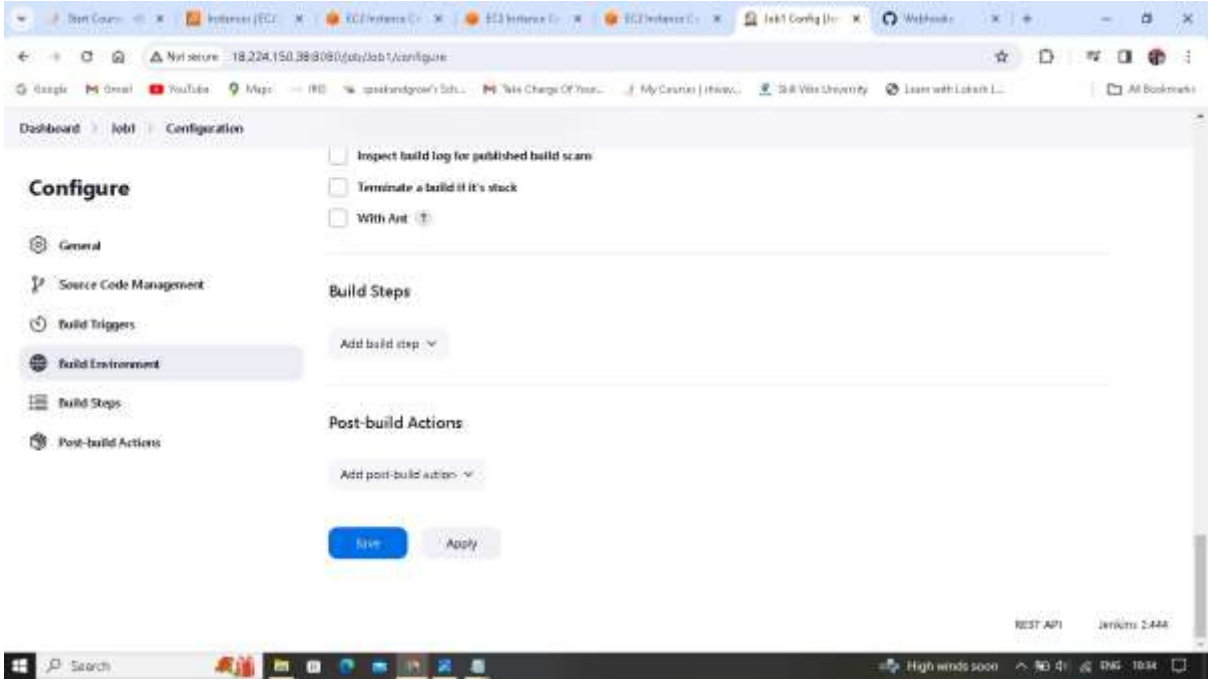
94.Under Payload URL paste it with /github-webhook/ and click active



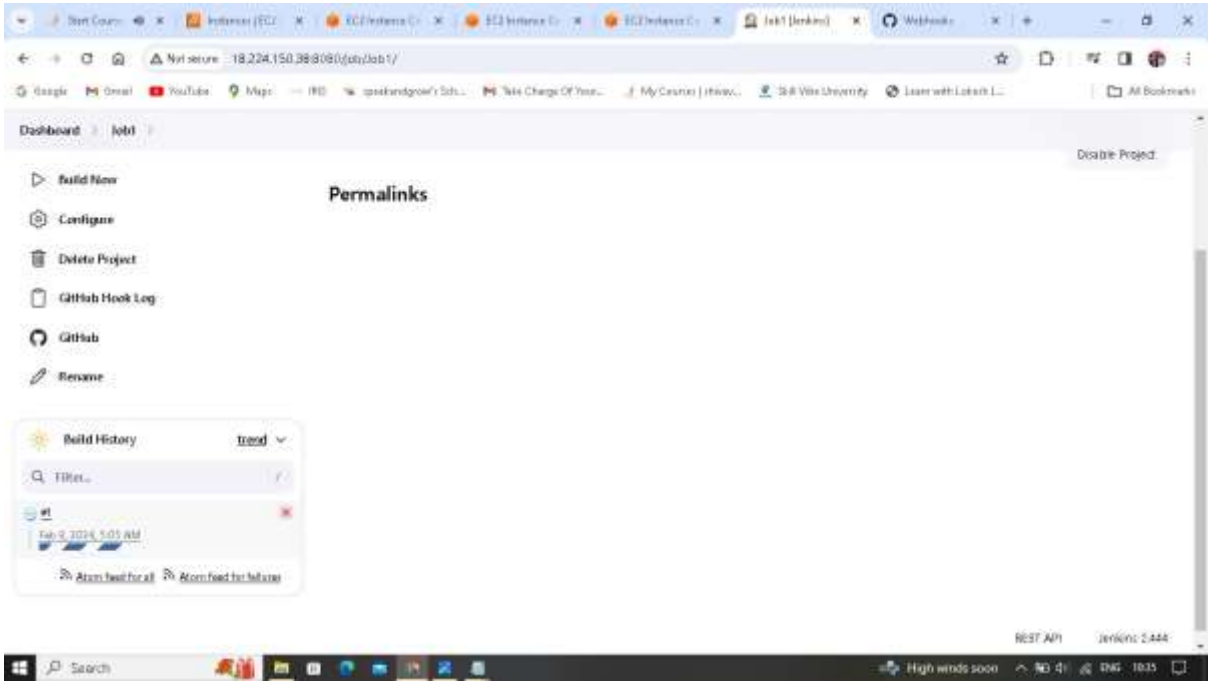
95. Check for tick mark for it.



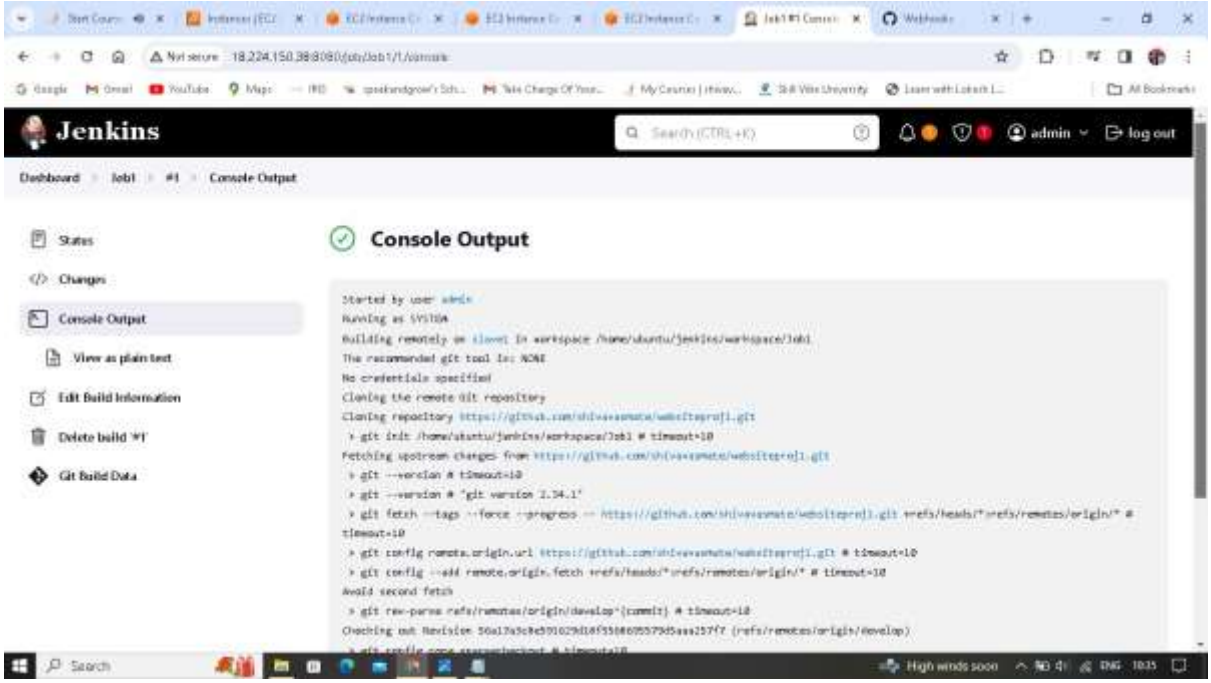
96. Click on both Apply and Save.



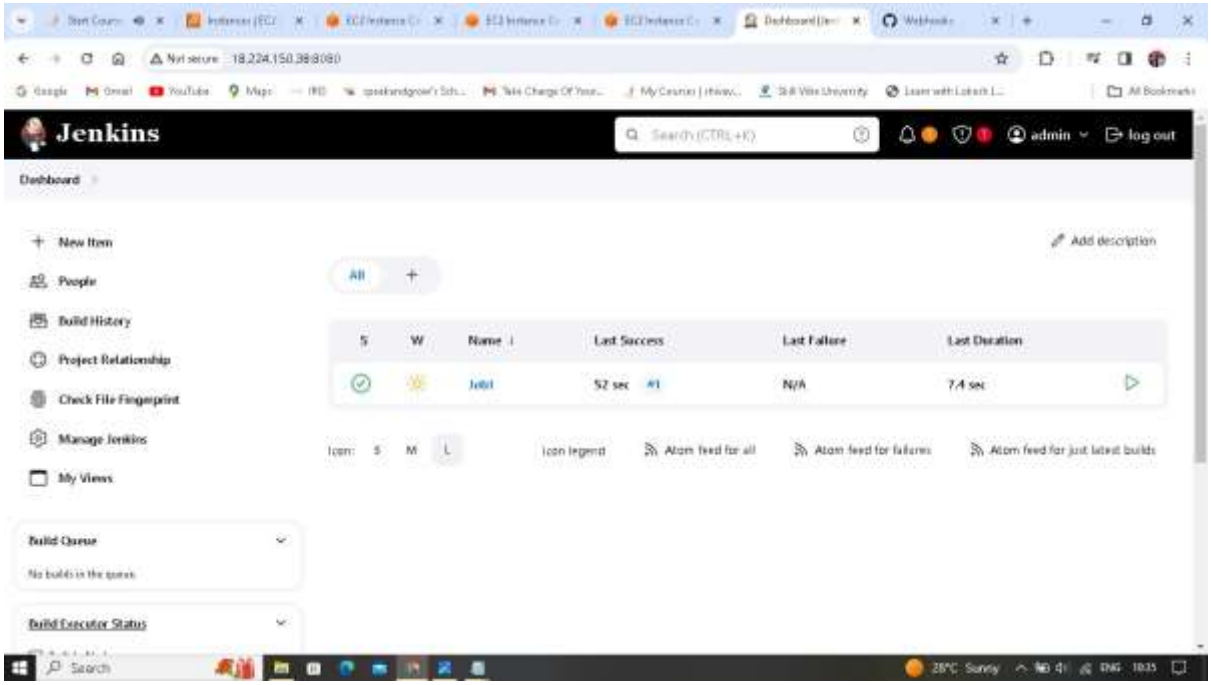
97. Click on Build Now #1 is running



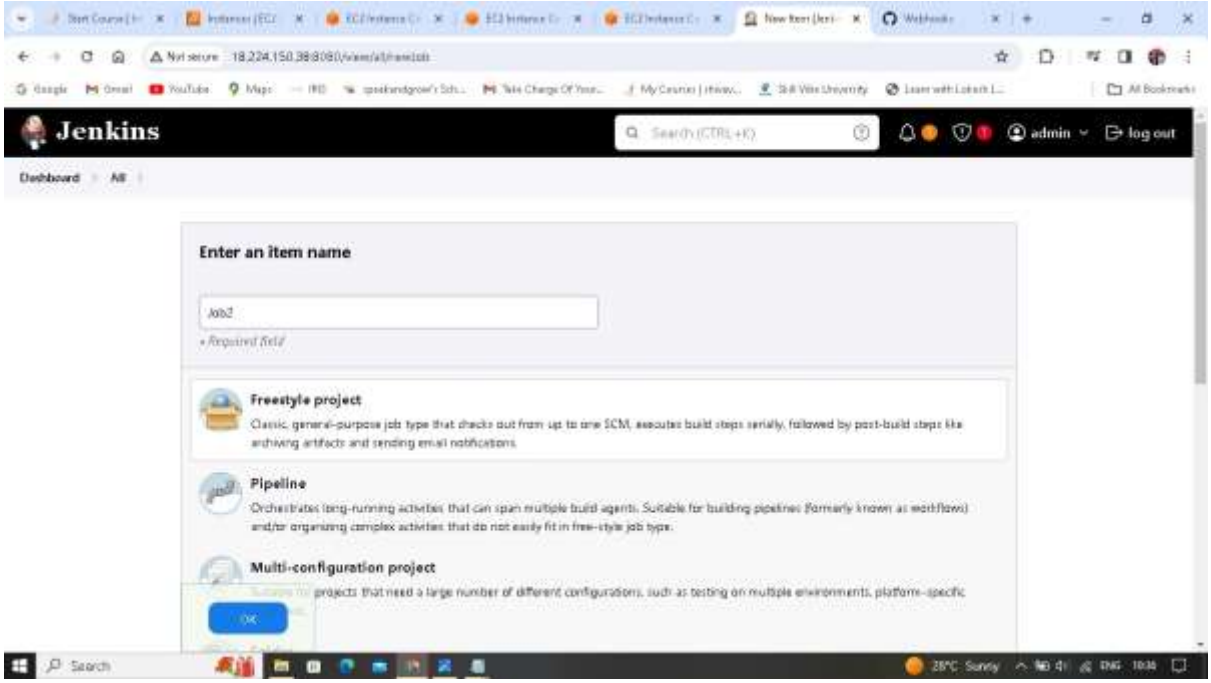
98. Job1 is successful is created.



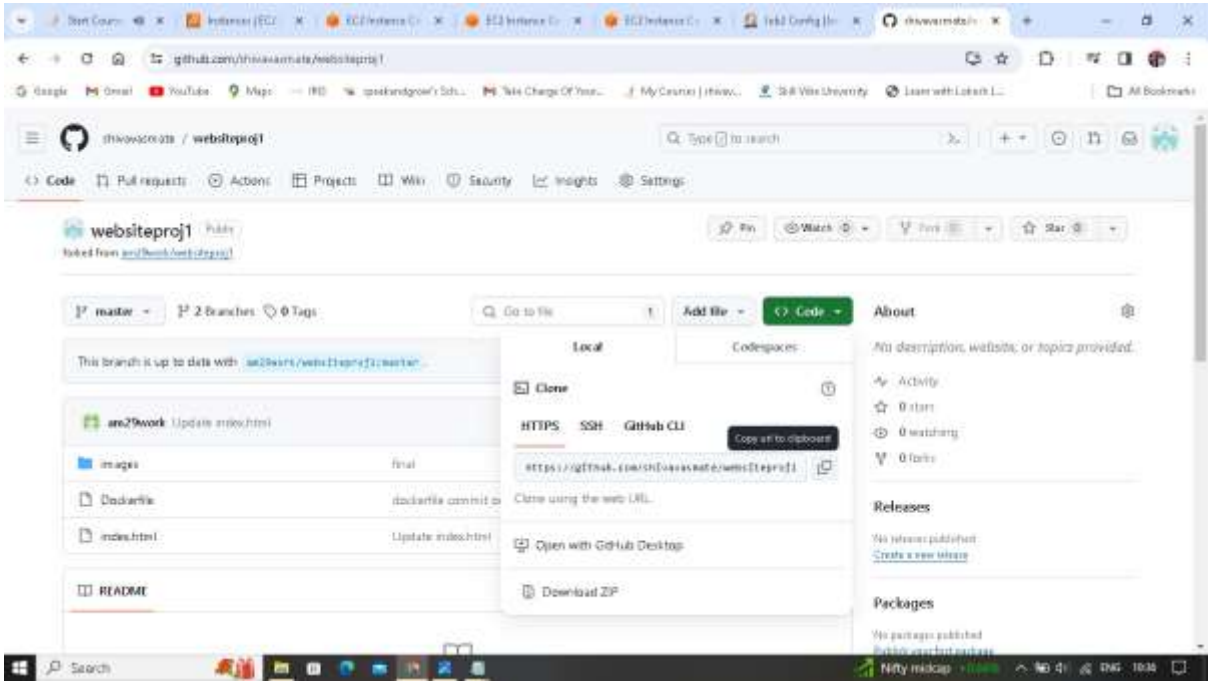
99. Go to Dashboard to new job click on +New Item.



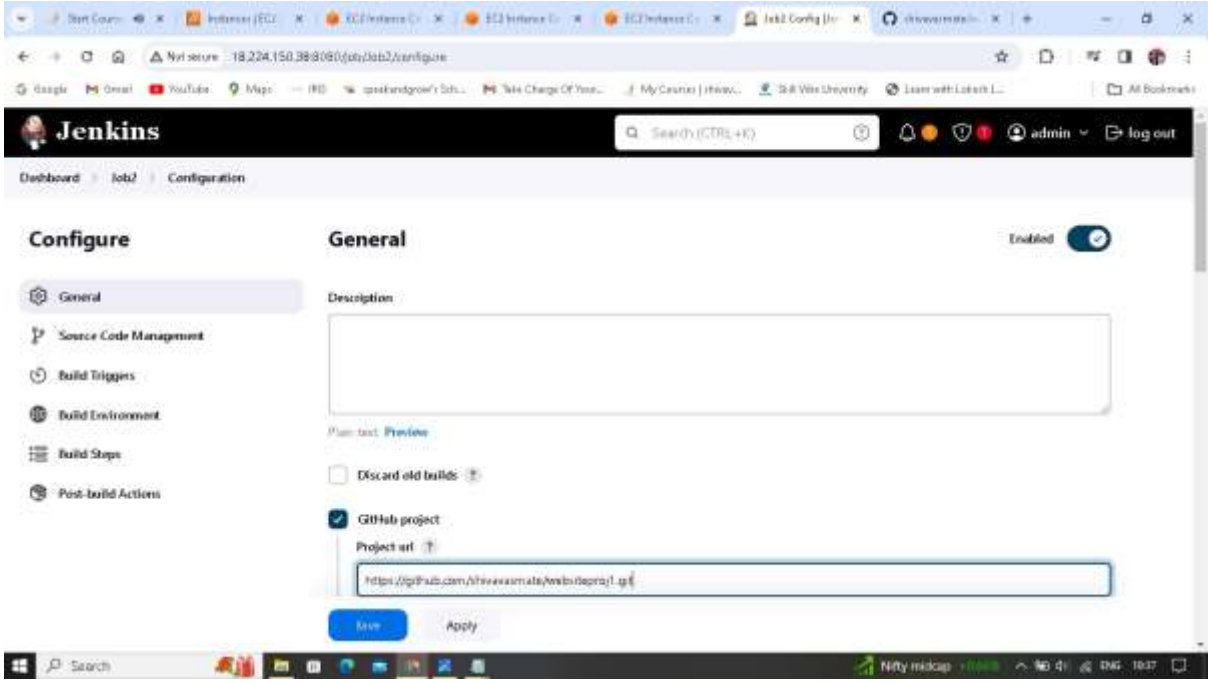
100. Enter an item name as Job2 and choose Freestyle project click on ok.



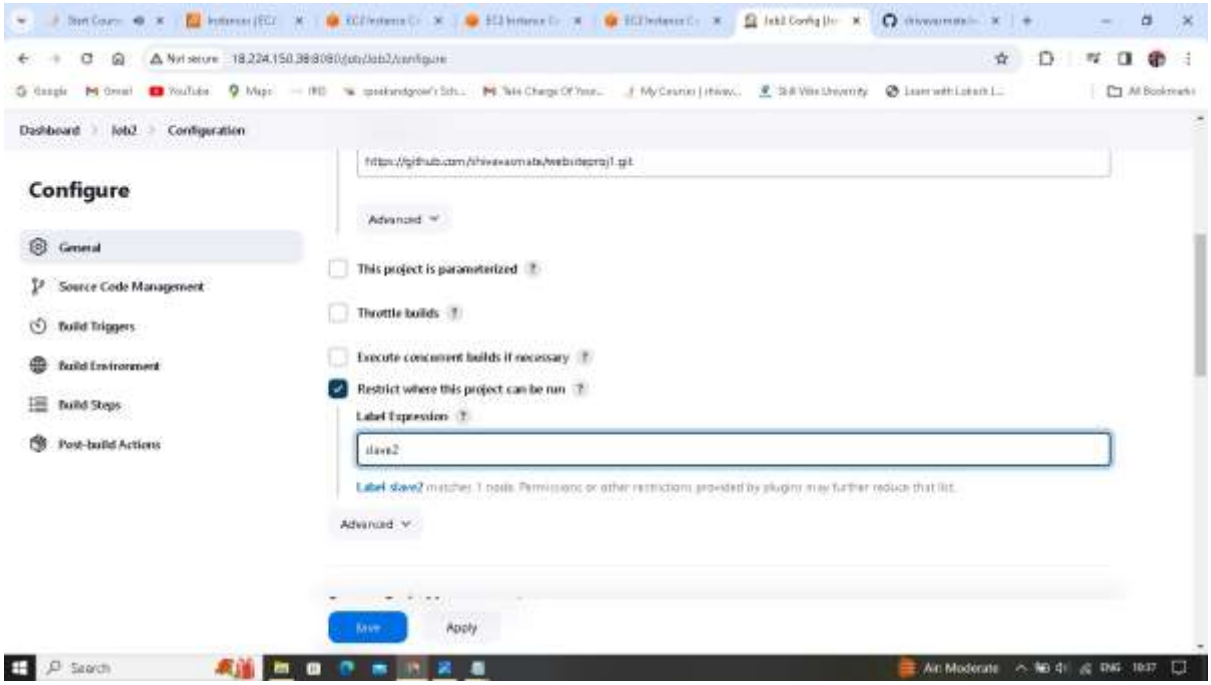
101.Copy the URL from github repo.



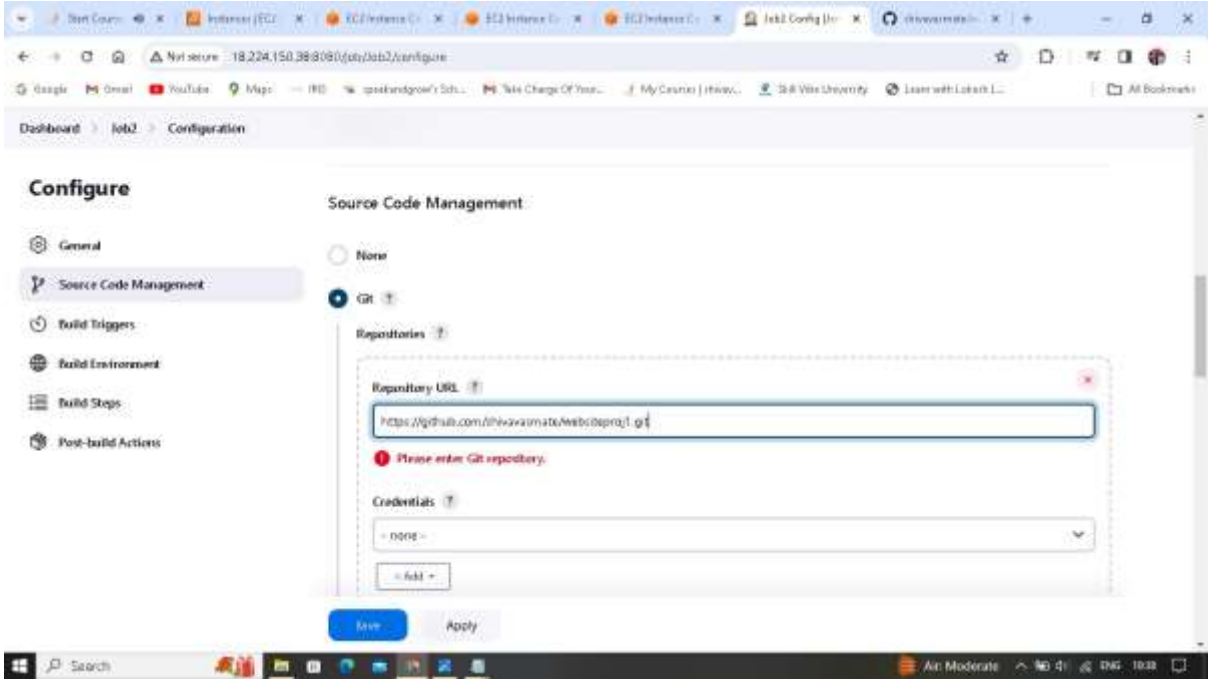
102. Paste it



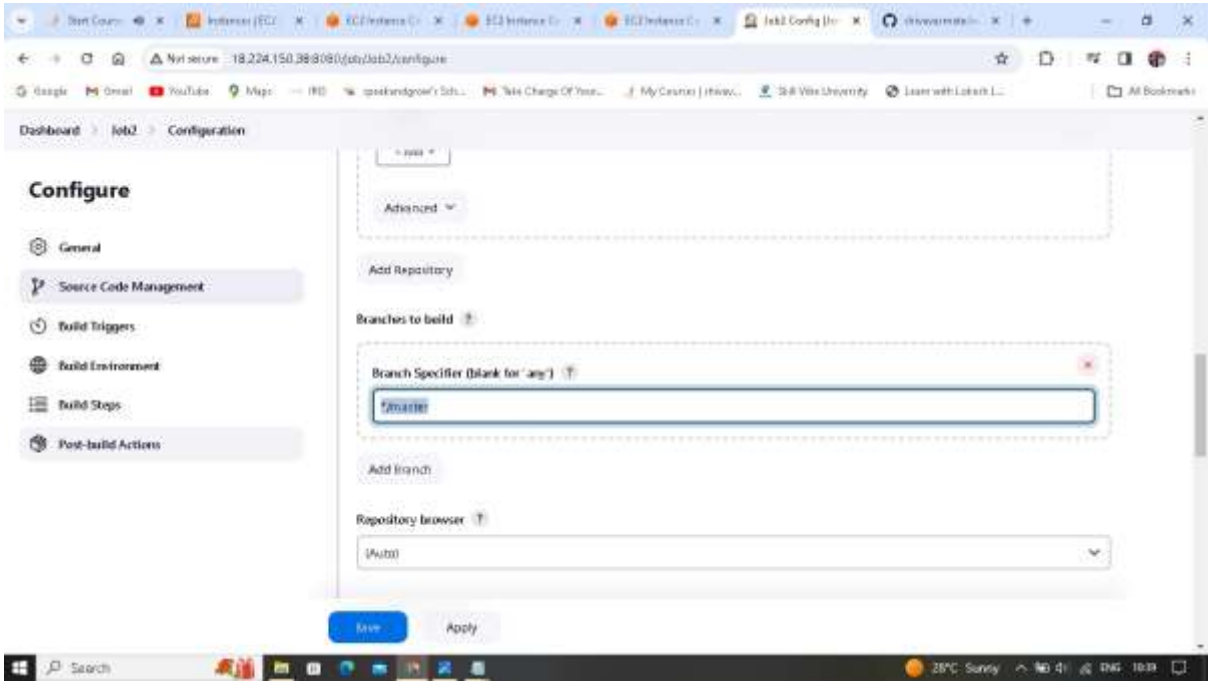
103.This time type slave2.



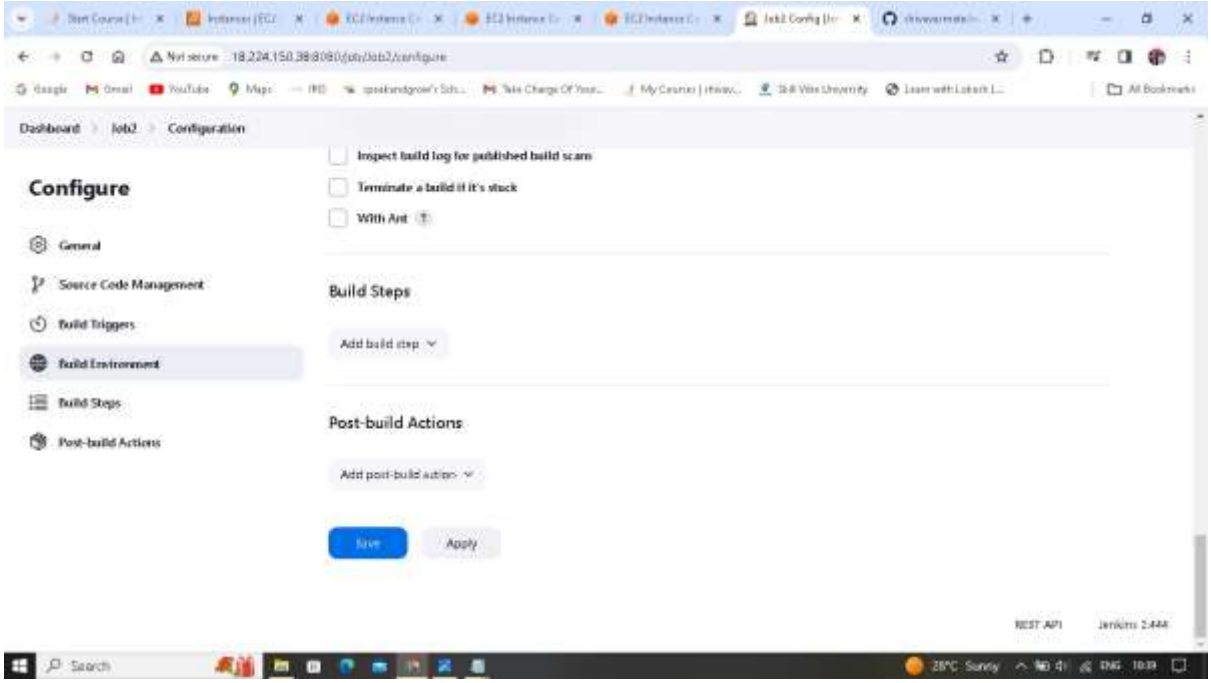
104.Under Source code management paste Repository URL from github repo.



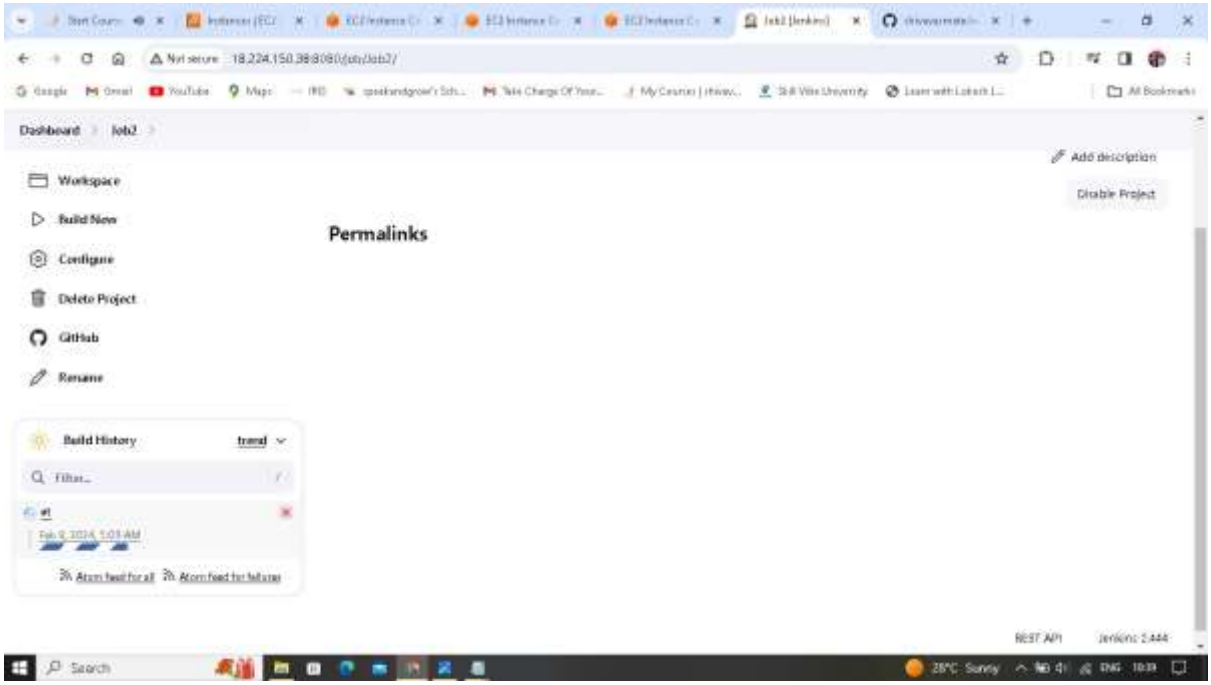
105.This time Branch specifier give */master.



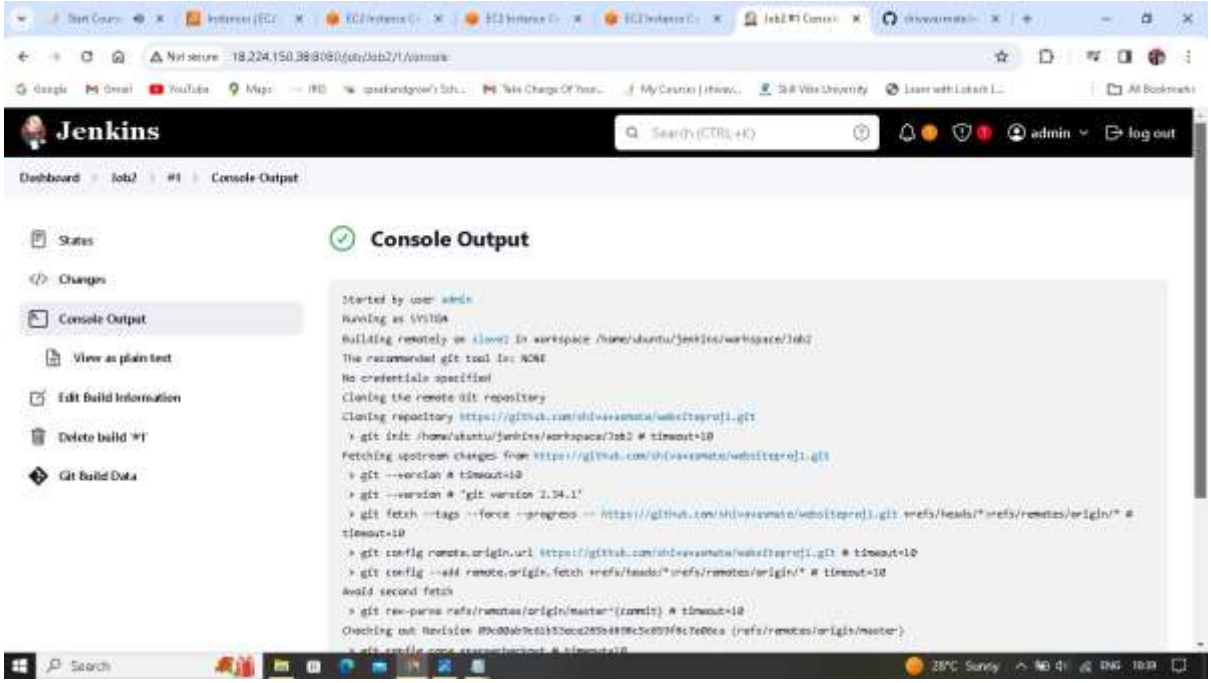
106. Click on both Apply and Save.



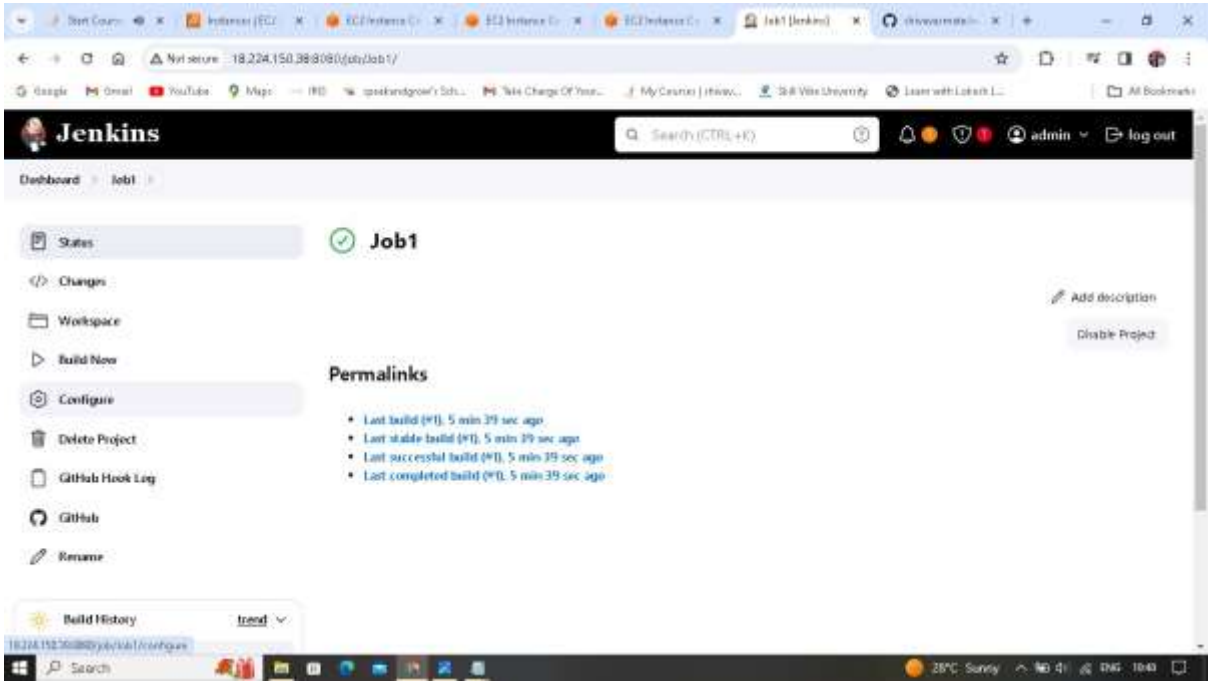
107.Click on Build Now #1 is running



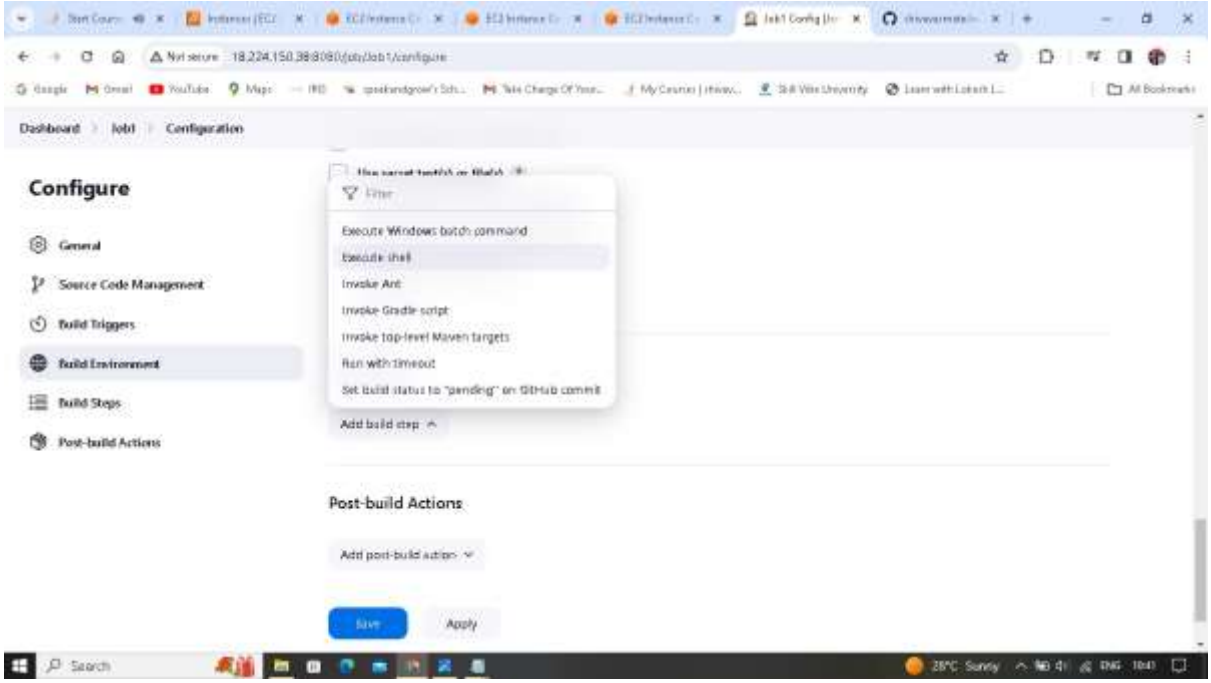
108.Job2 is successfully running.



109. Now create docker files for Job1 and Job2. Go to Job1 click on Configure.



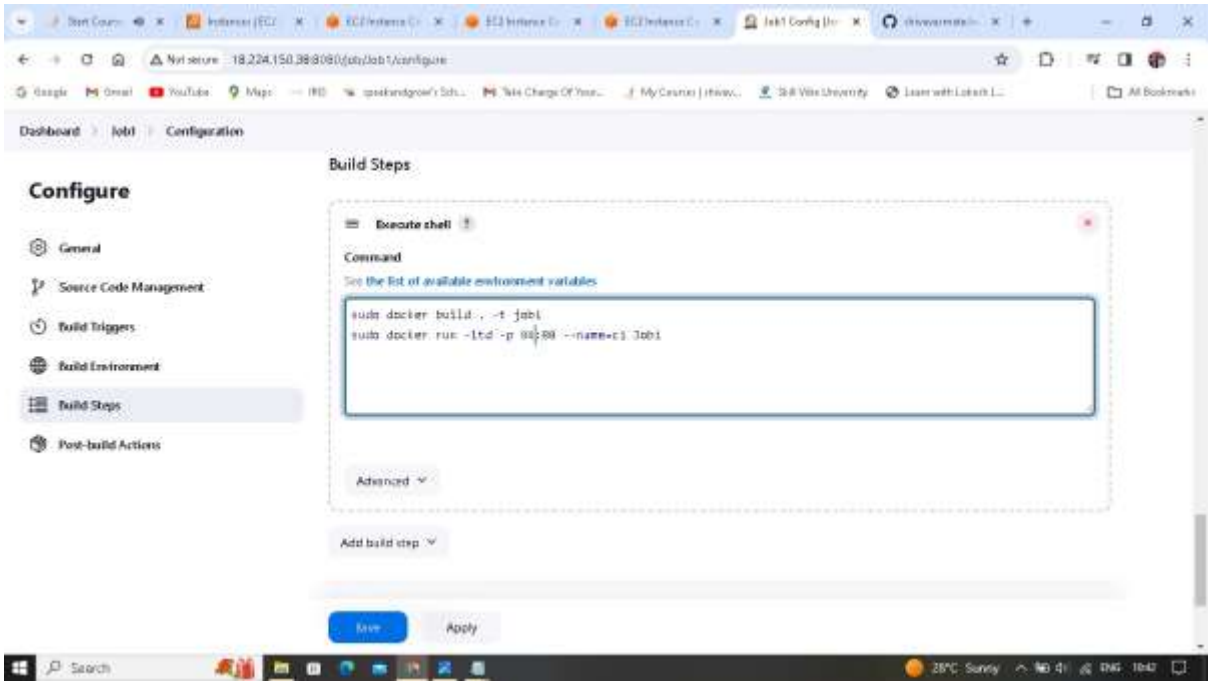
110. Go at the end under Add build step choose Execute shell.



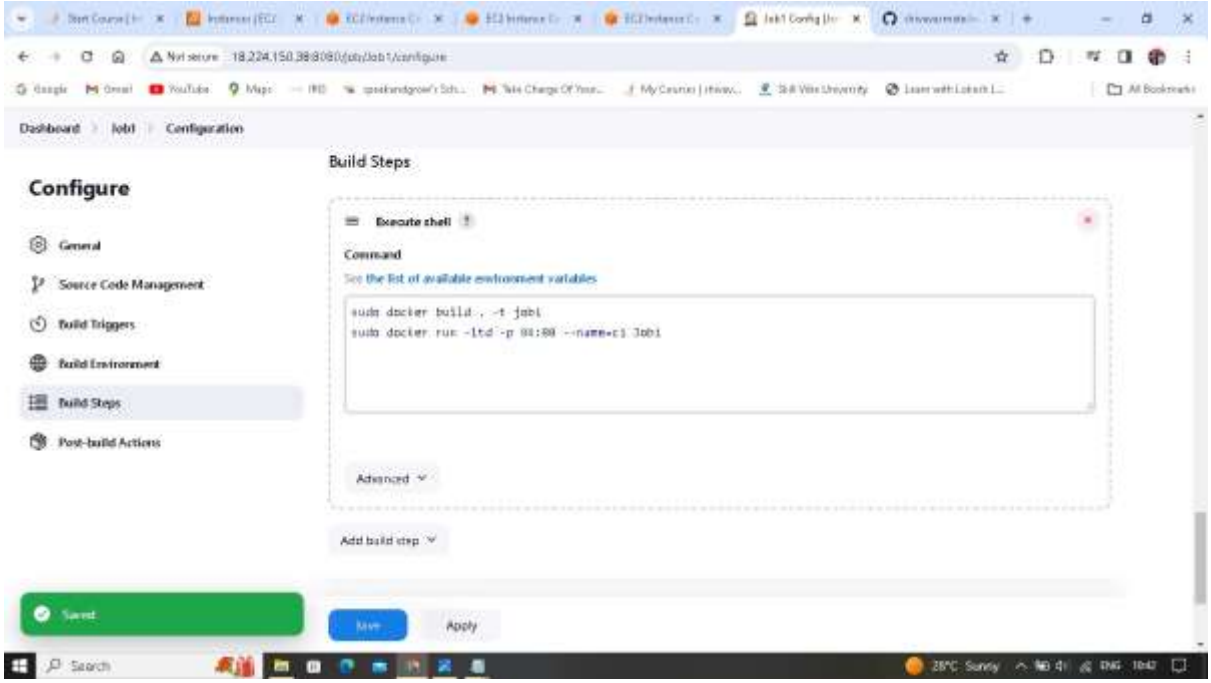
111. Under it type

sudo docker build . -t job1

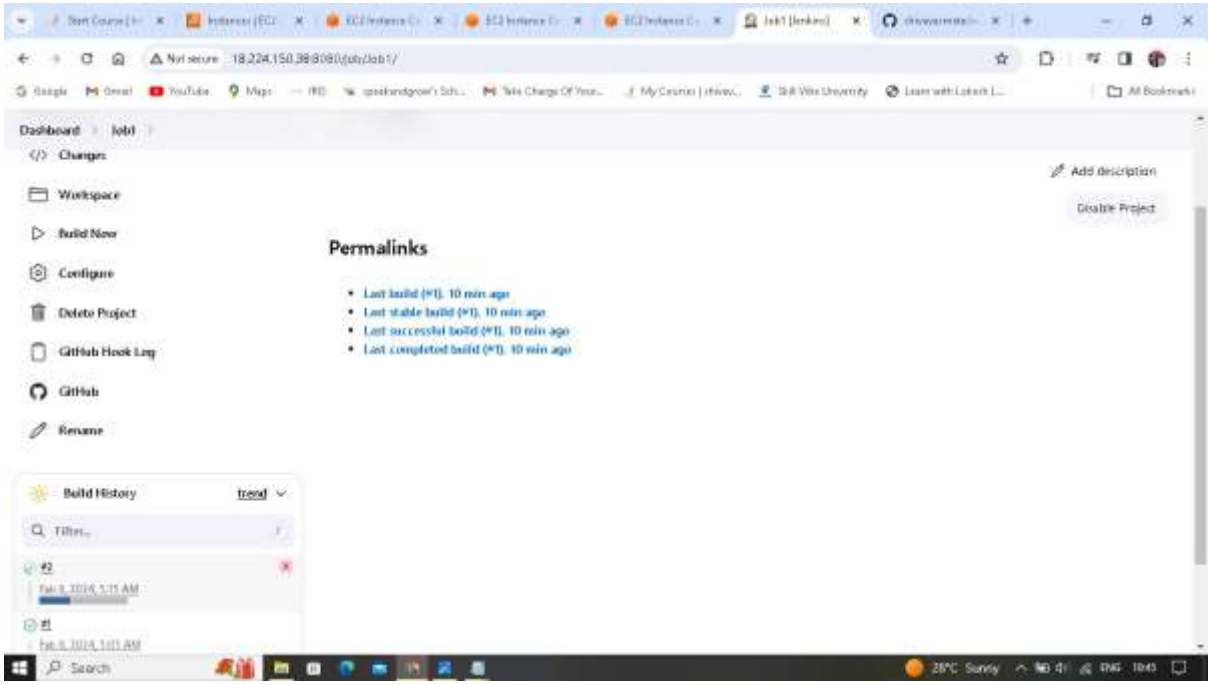
sudo docker run -itd -p 84:80 --name=c1 job1



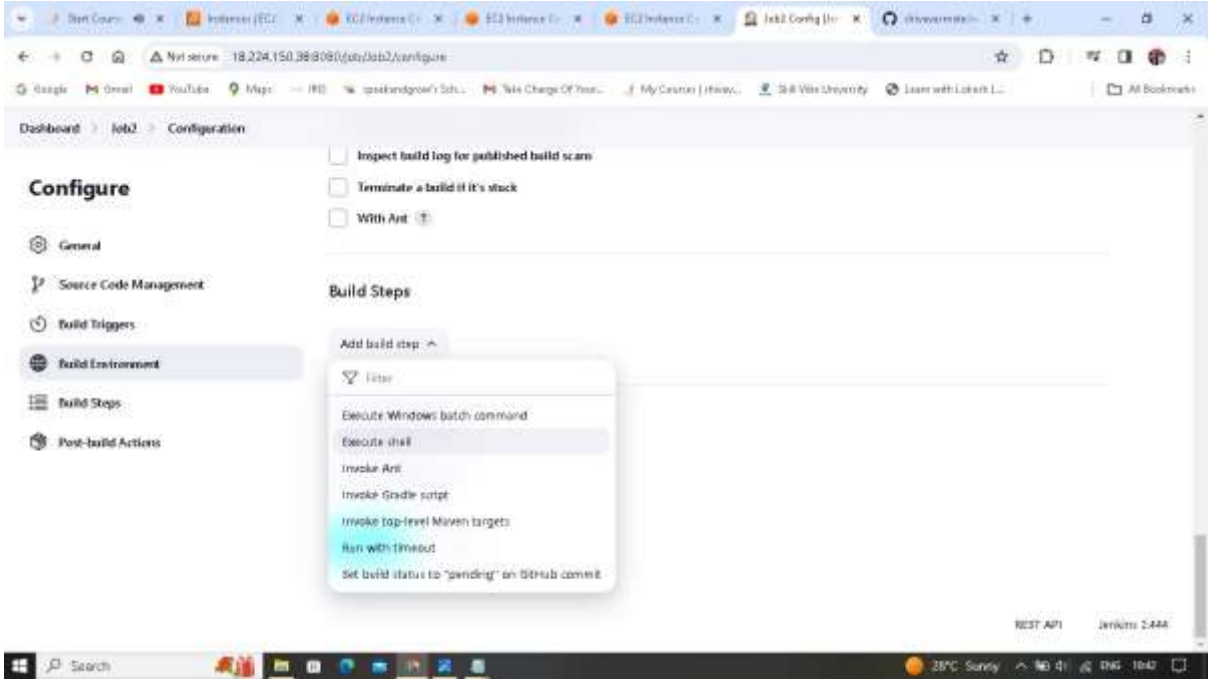
112.Click on both Apply and Save.



113.Click on Build Now



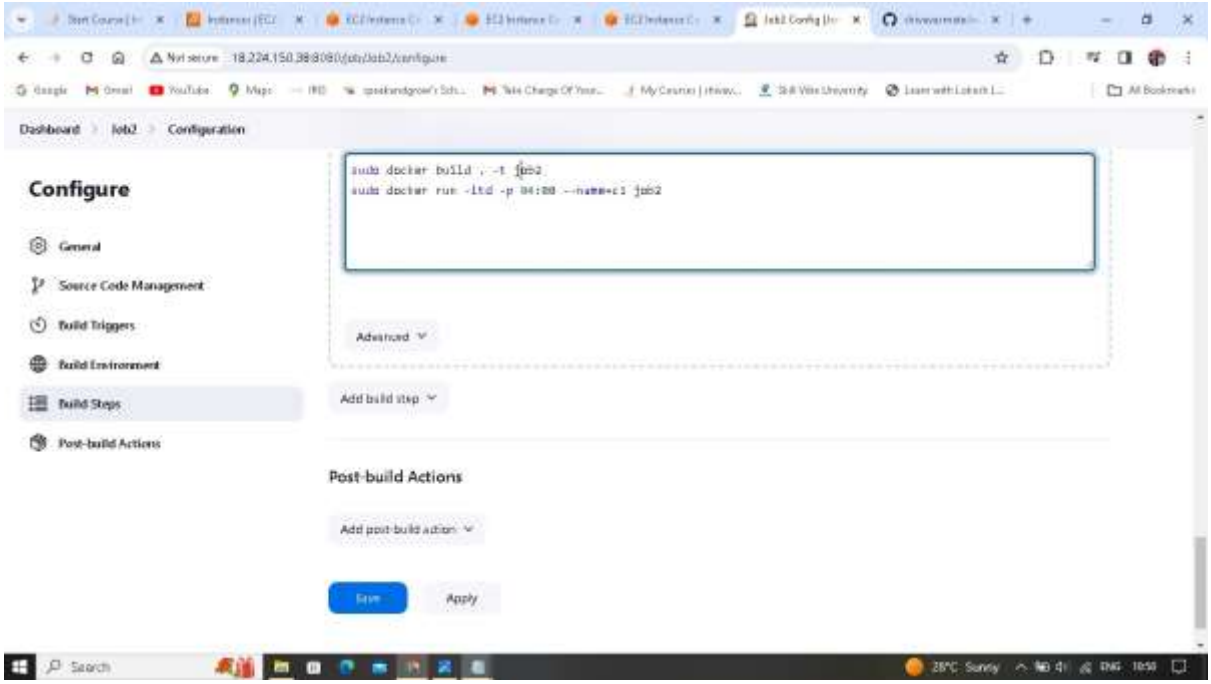
114.Similarly for Job2 go to Build Steps choose Execute shell.



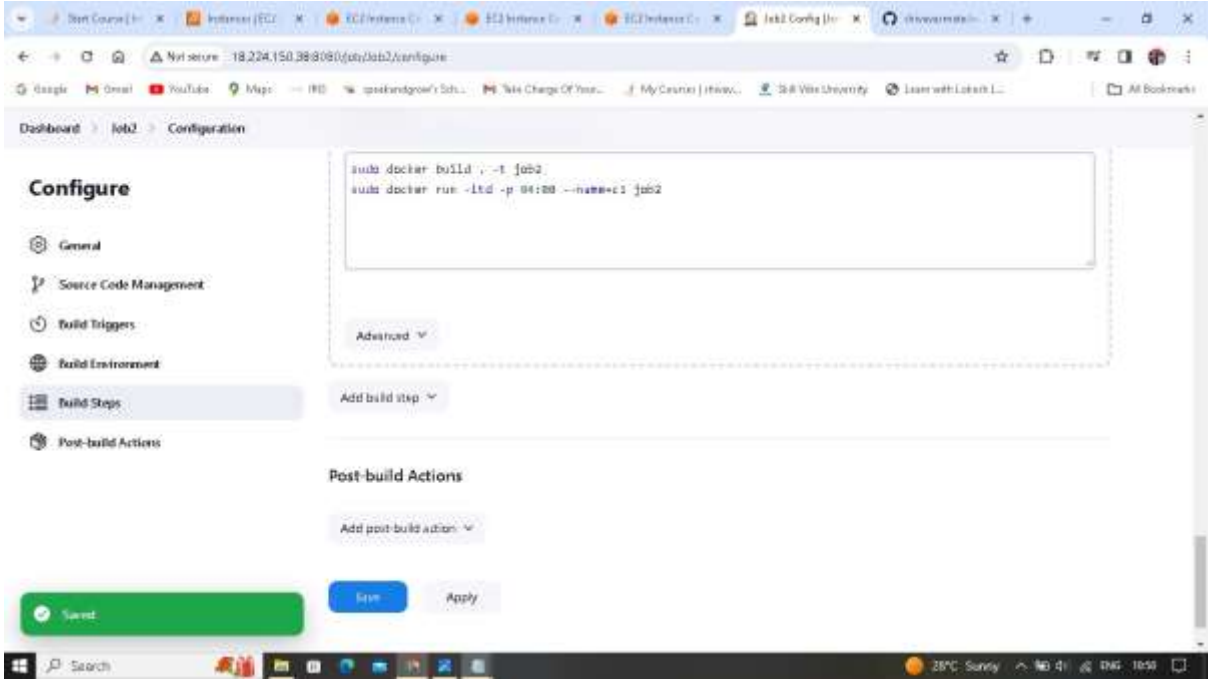
115. Under it type

sudo docker build . -t job2

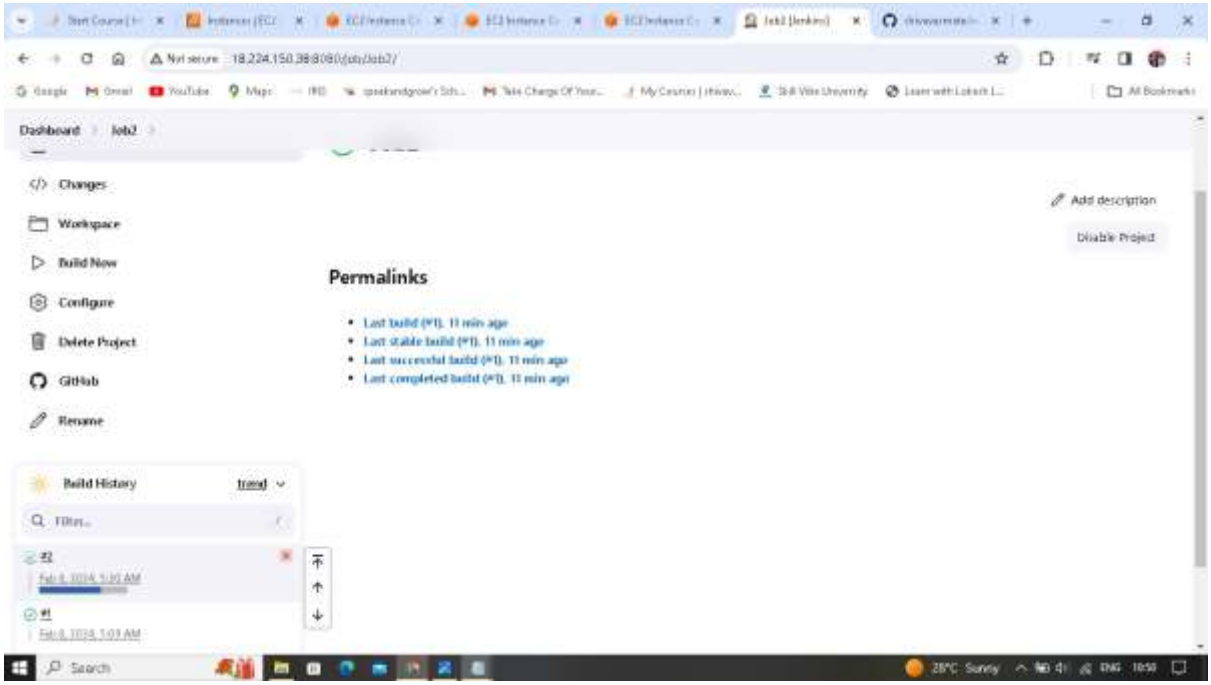
sudo docker run -itd -p 84:80 --name=c1 job2



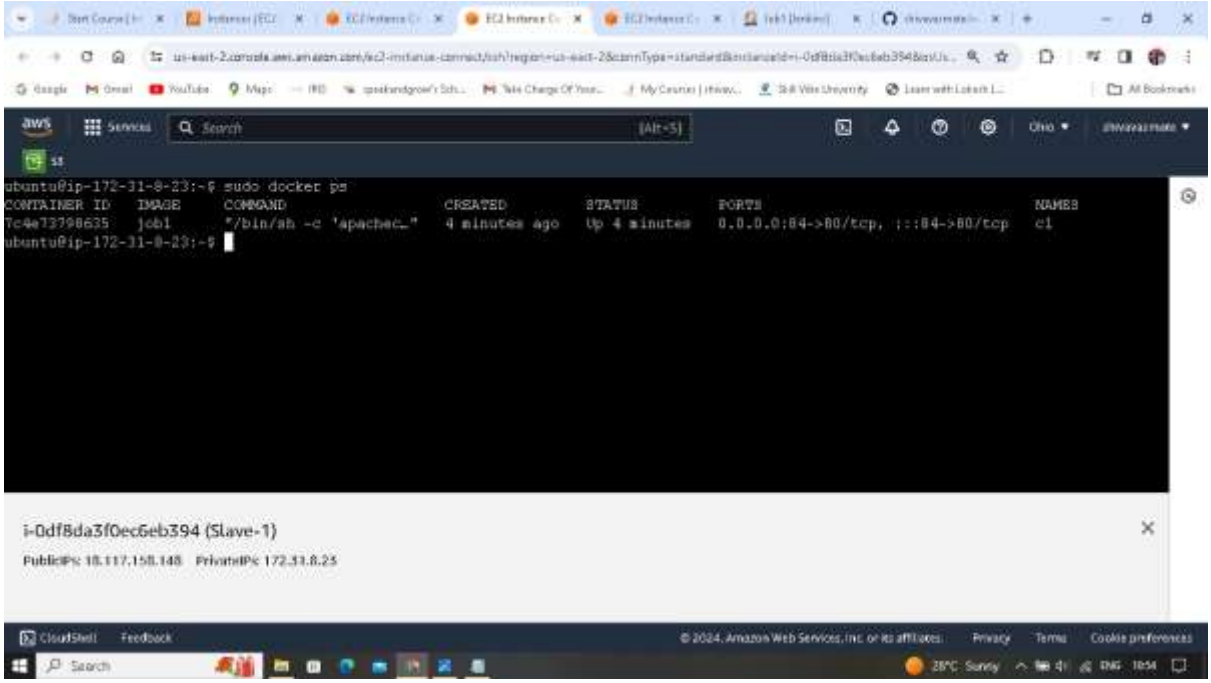
116.Cilck on both Apply and Save.



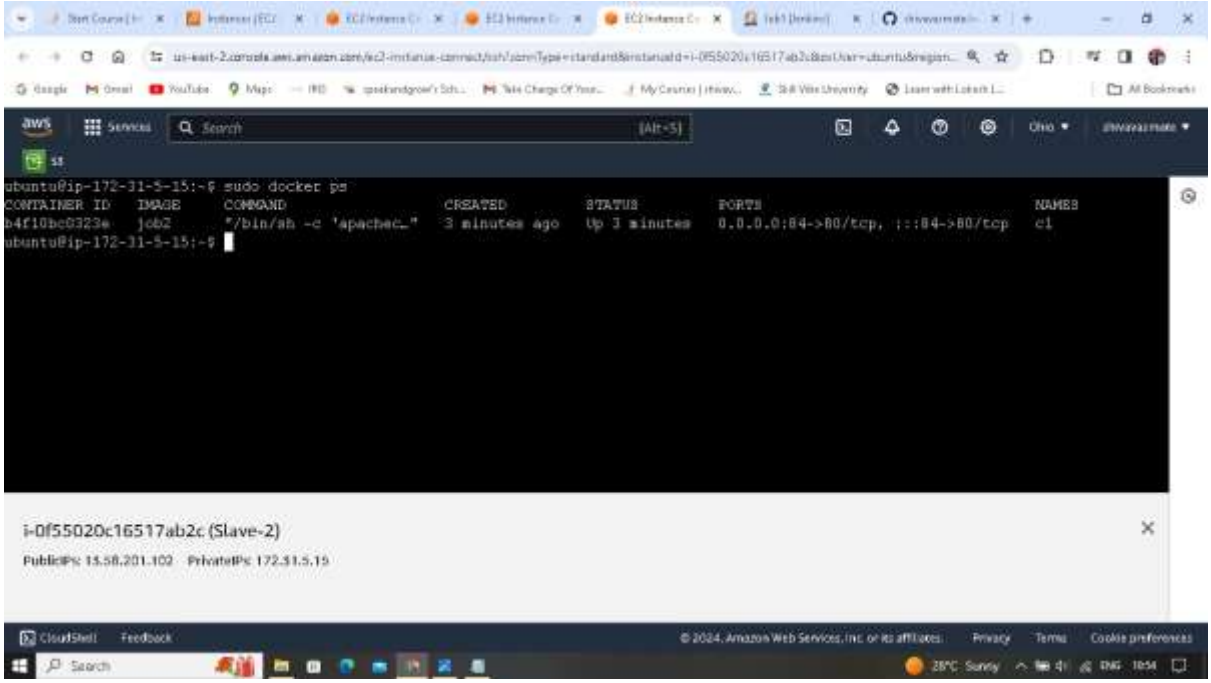
116. Click on Build Now



117. To Verify go to Slave1 machine check for docker by sudo docker ps command.



118.Similarly for Slave2 machine.

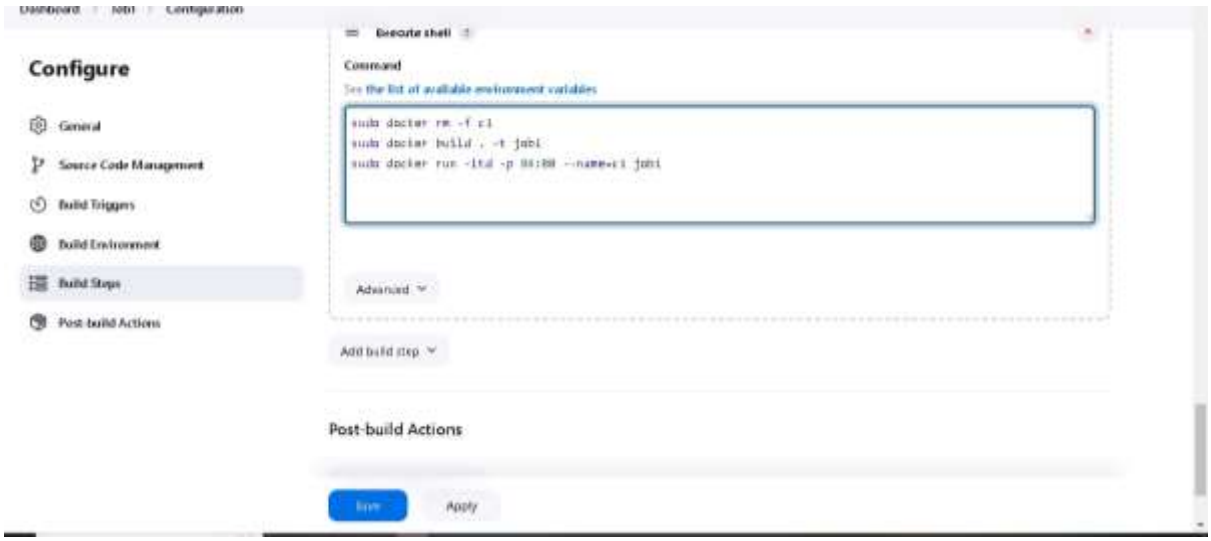


119.For Job1 edit the execute shell by

sudo docker rm -f c1

sudo docker build . -t job1

sudo docker run -itd -p 84:80 --name=c1 job1

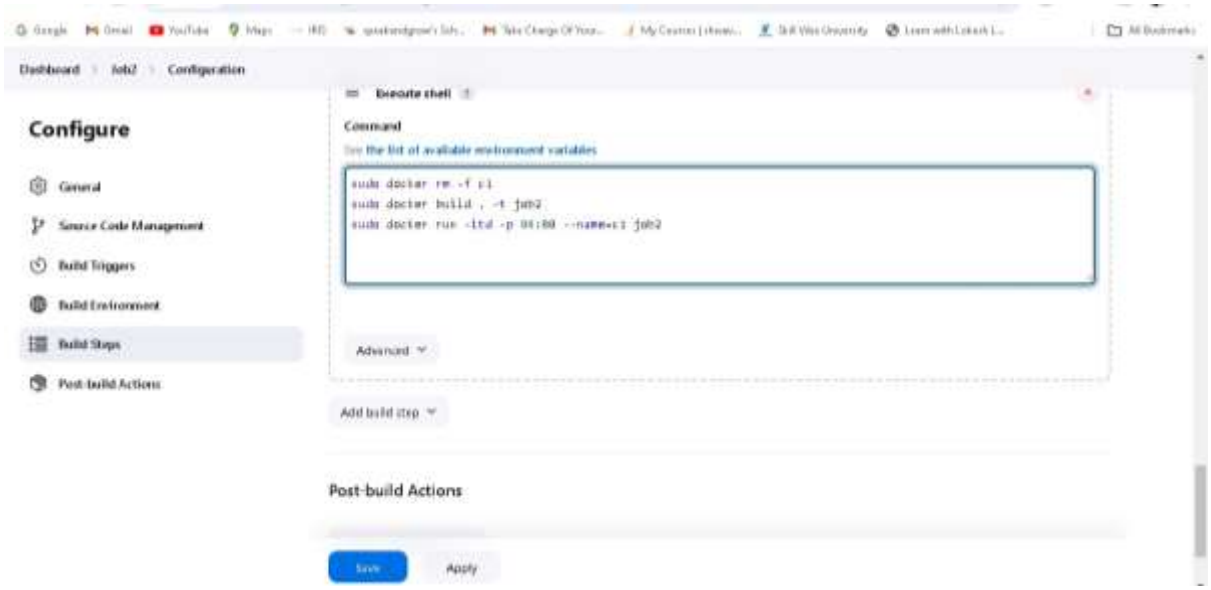


120. For Job2 edit the execute shell by

sudo docker rm -f c1

sudo docker build . -t job2

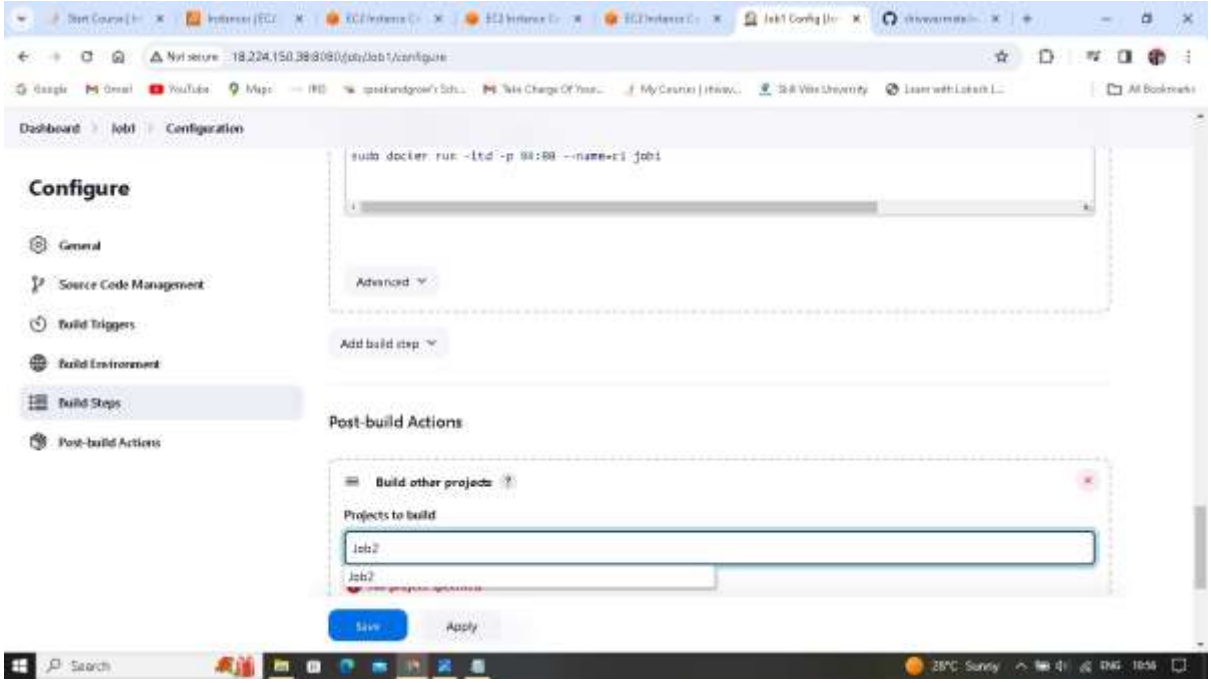
sudo docker run -itd -p 84:80 --name=c1 job2



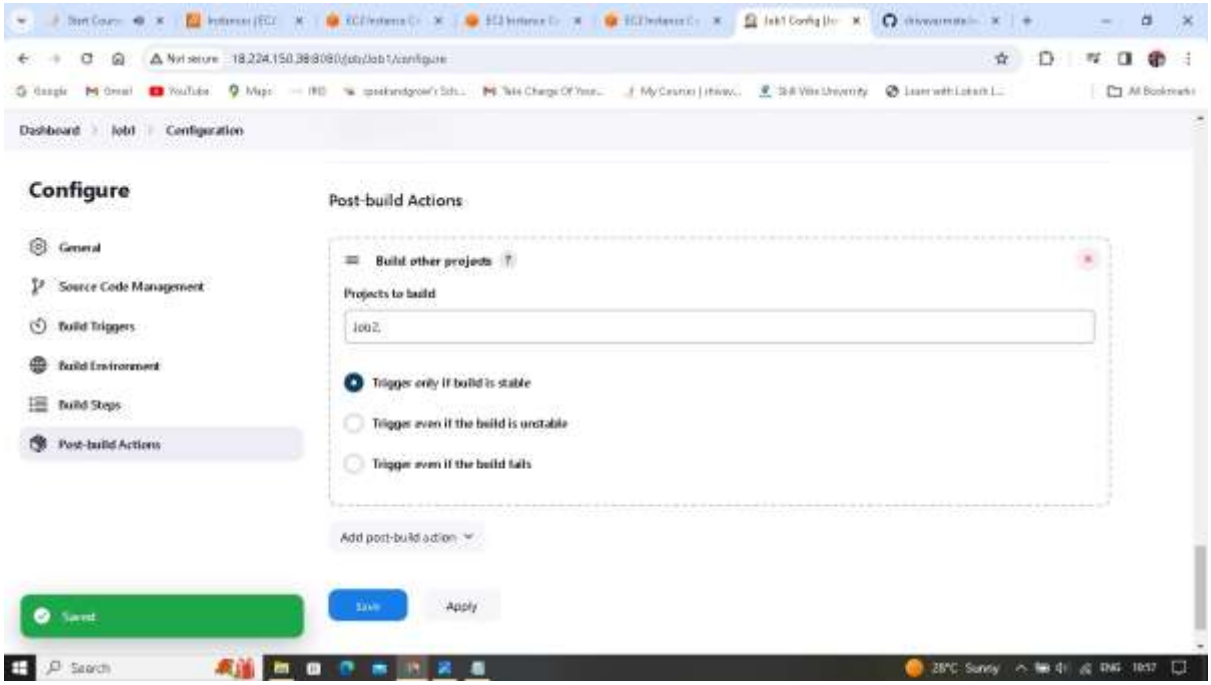
Second time it runs no error will appear. Only one docker container will run only on one port.

121.Go to Job1 configure under Post build Action. Give Job2 as project to build.

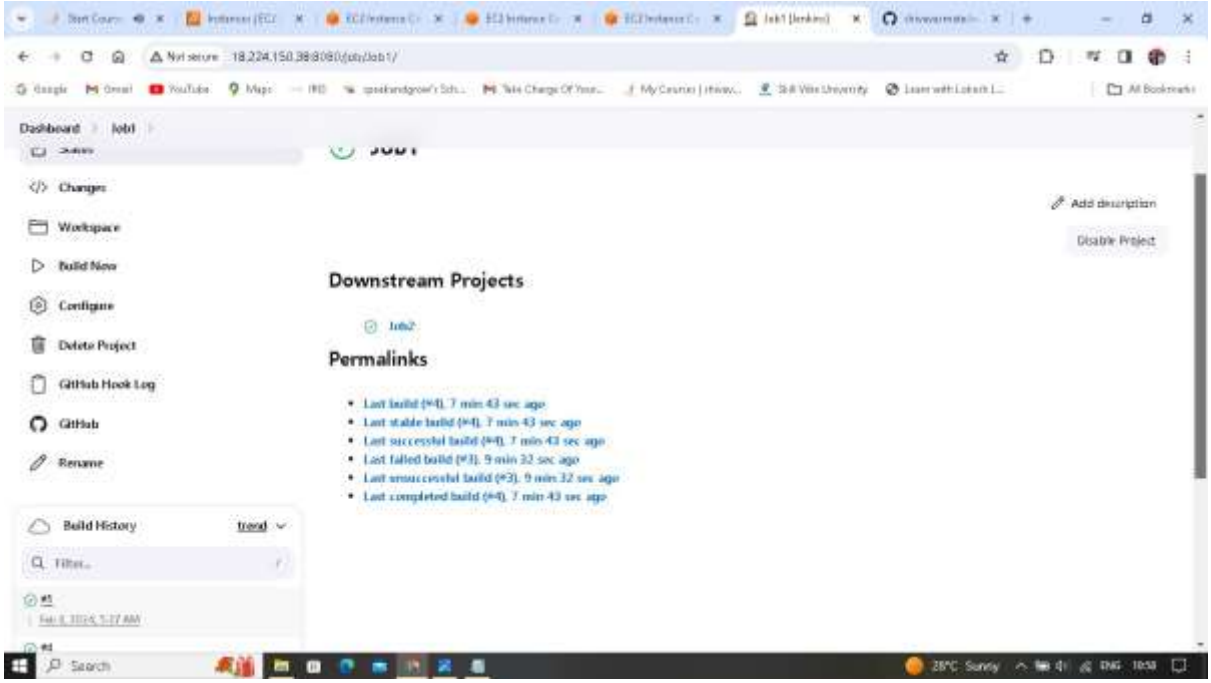
such that after the job1 the job2 trigger automatically.



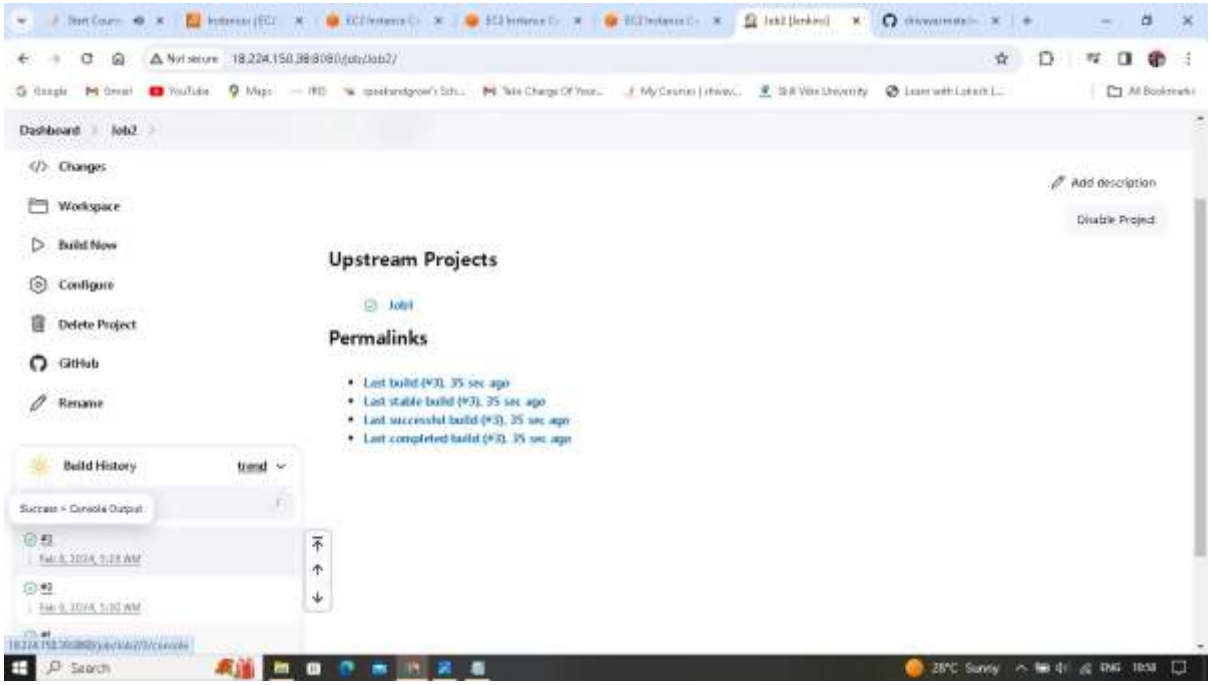
122.Click on both Apply and Save.



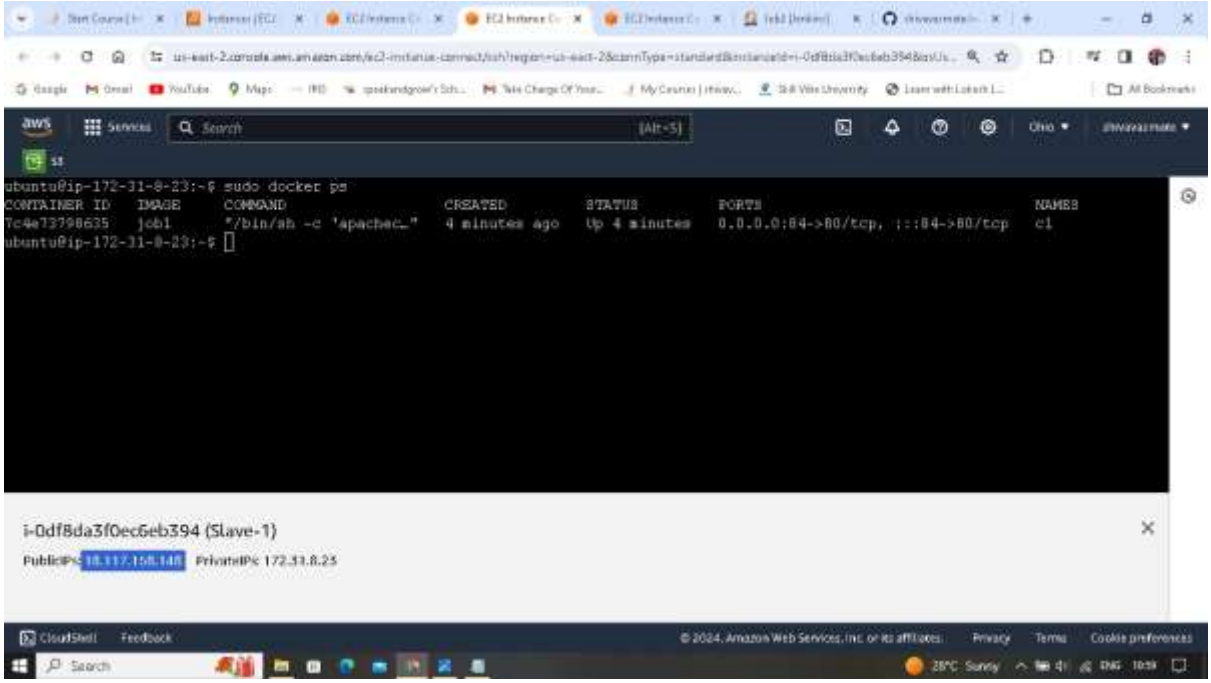
123.Click on Build Now #5 is ran successful.



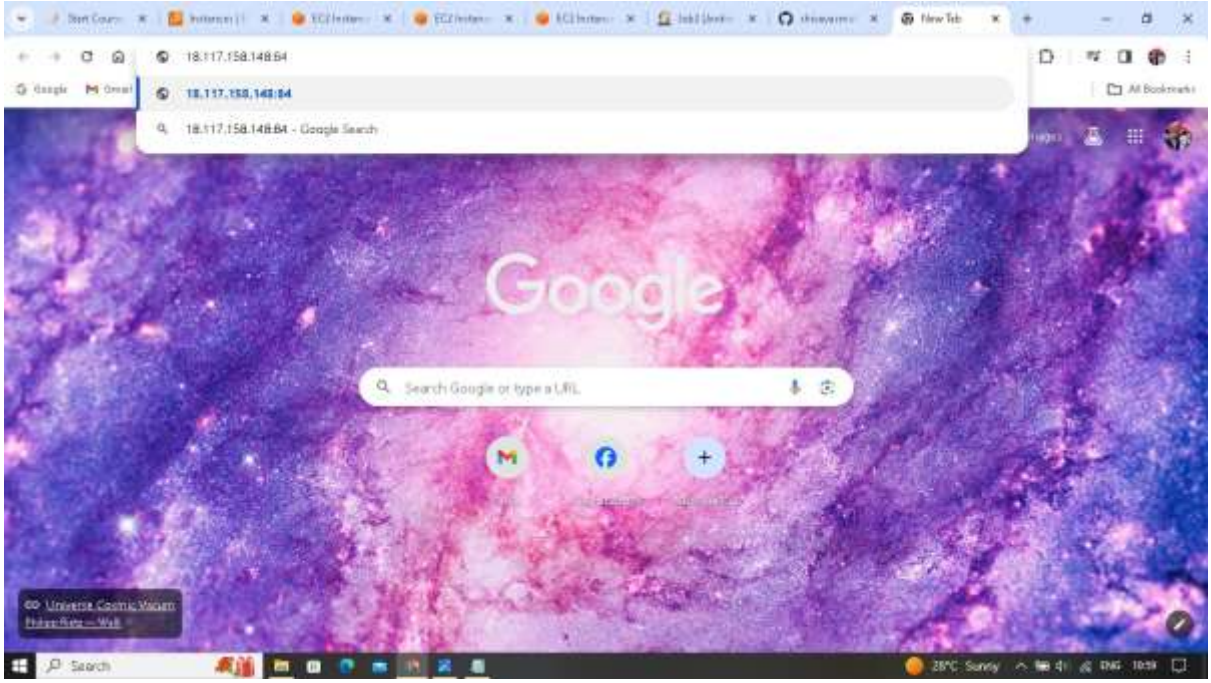
124.Job1 triggers #3 is also successful for the Job2.



125.To Verify copy the public IP of Slave1

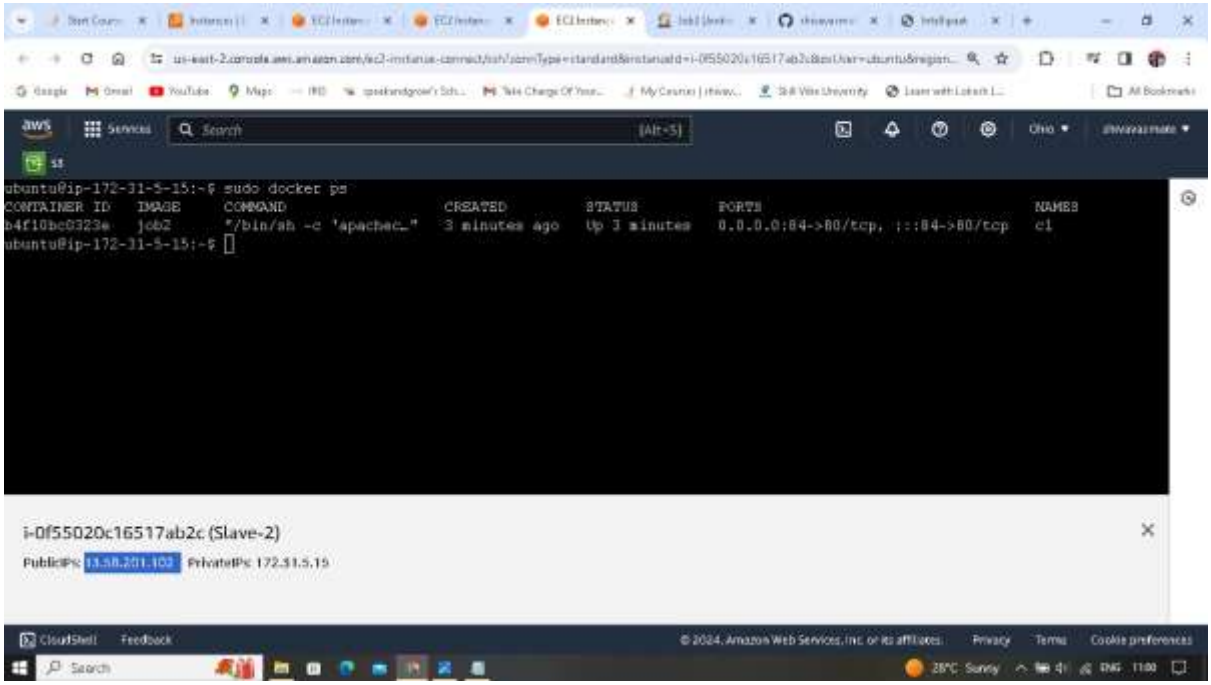


126.Paste it with port :84 in another browser.

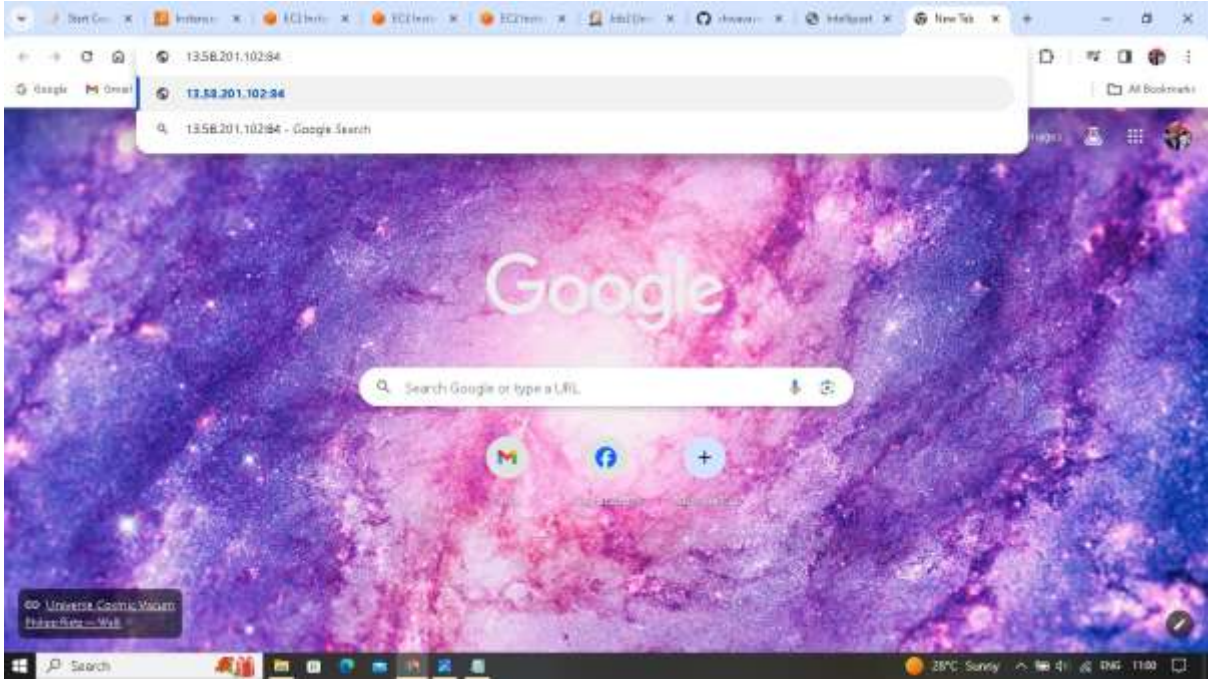


127. Docker image is executed

128.Similarly copy the public IP of Slave2



129. Paste it with port :84 in another browser.



130.Docker image execute successfully.