

## **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 be built every time there is a push to Git-Hub. Use the

following pre-built container for your application:

hshar/webapp

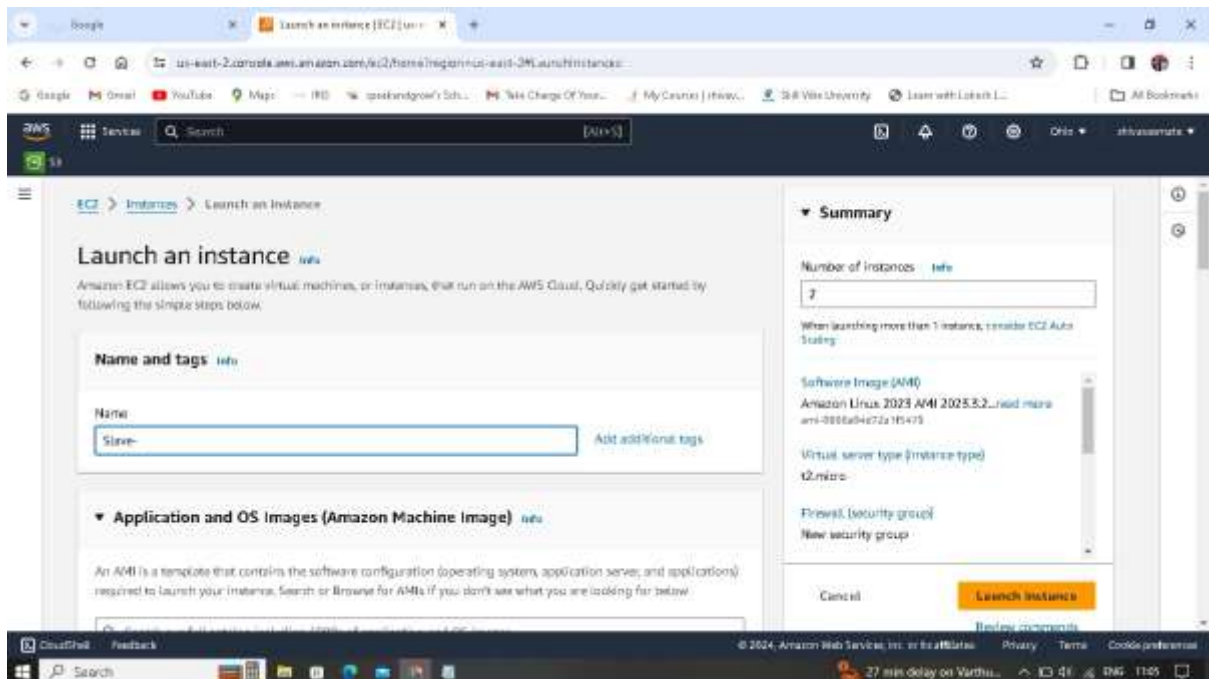
The code should reside in '/var/www/html'

5. The above tasks should be defined in a Jenkins Pipeline, with the following jobs: Job1: build , Job2: test, Job3: prod.

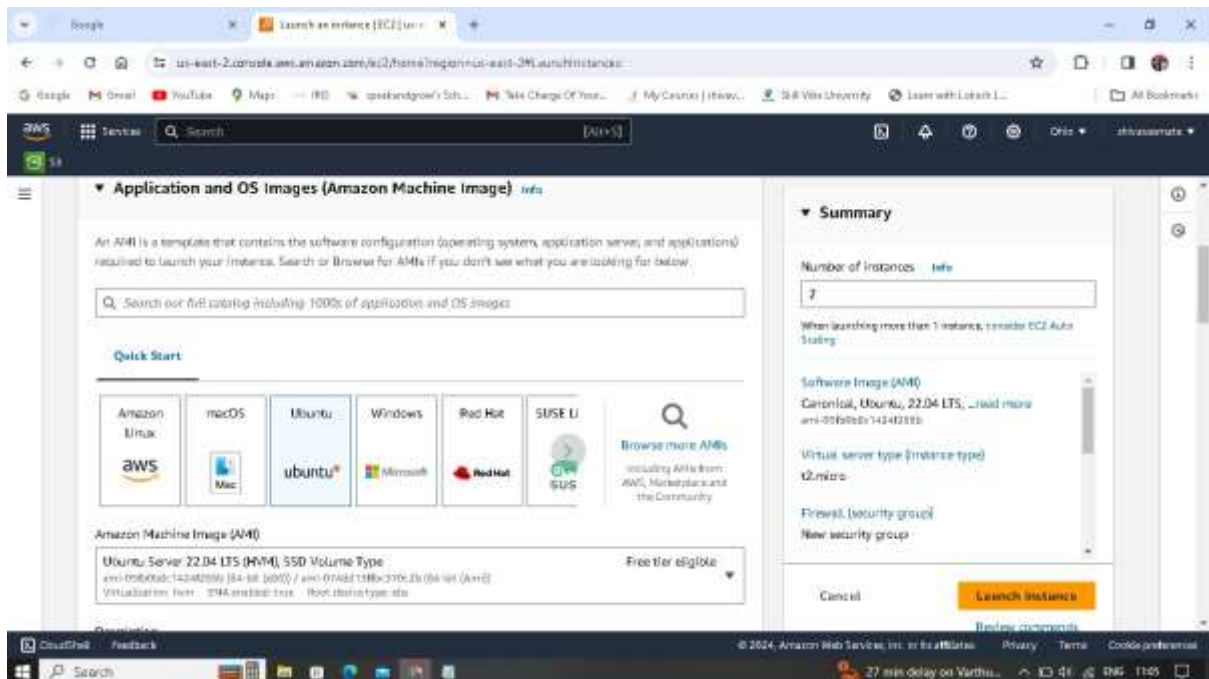
Ans:

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-

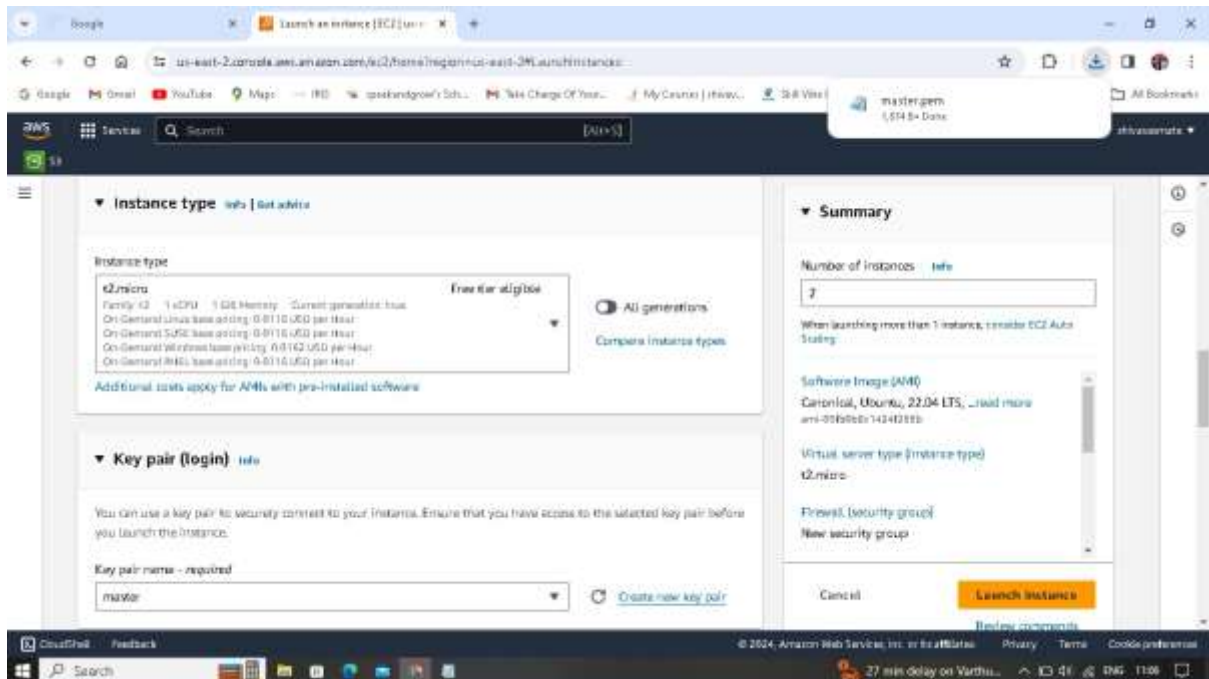


b. AMI: QuickStart >> Ubuntu [Any version which is free tier eligible]

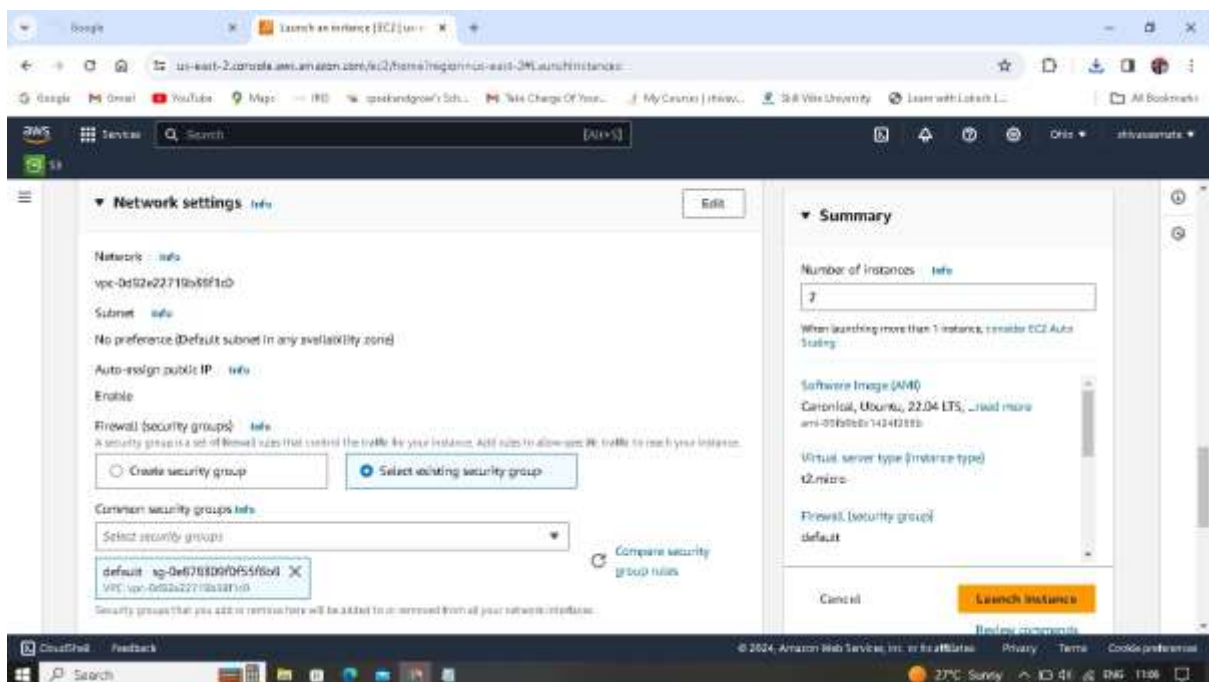


c. Instance type: t2. micro [free tier eligible]

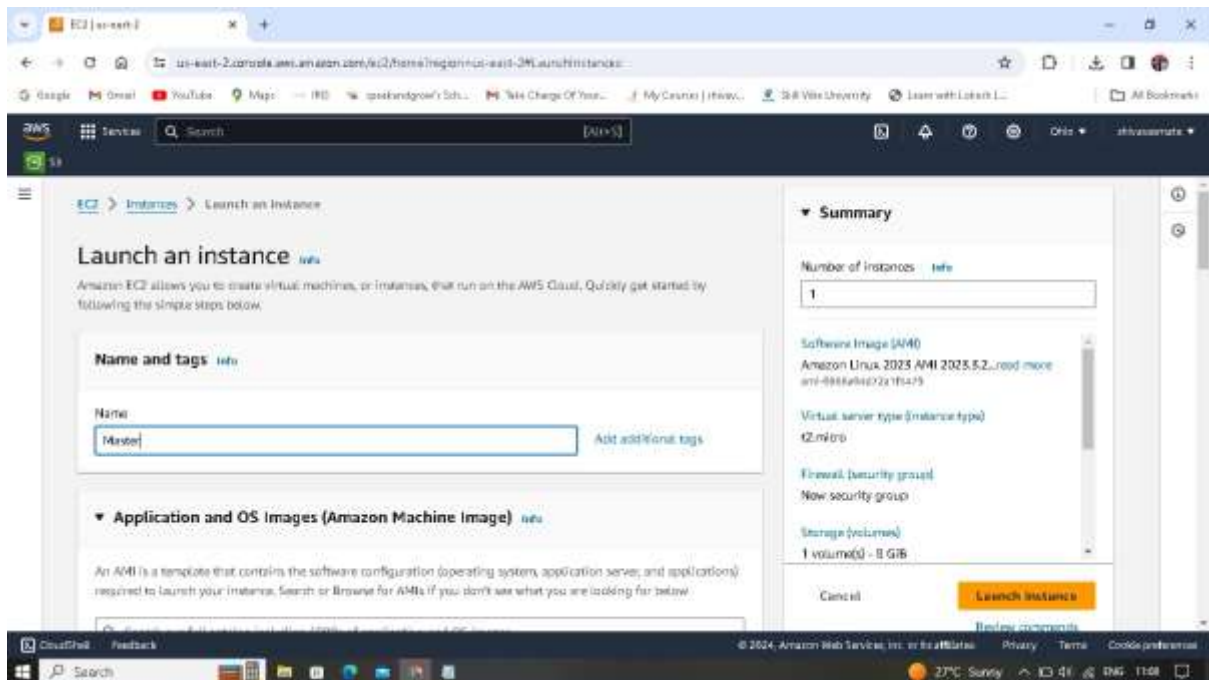
d. Key-pair: Create a key pair [rsa and .pem] with a name master



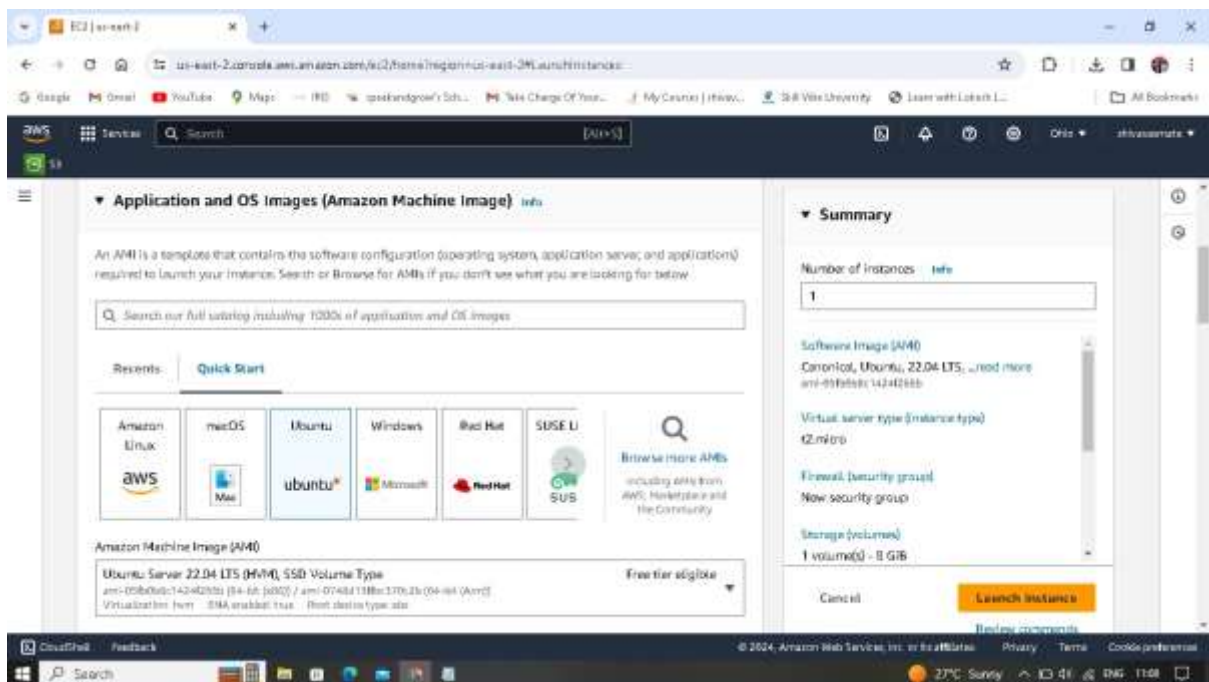
e. VPC and Security group as default with ssh and HTTP protocols click on launch instance.



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

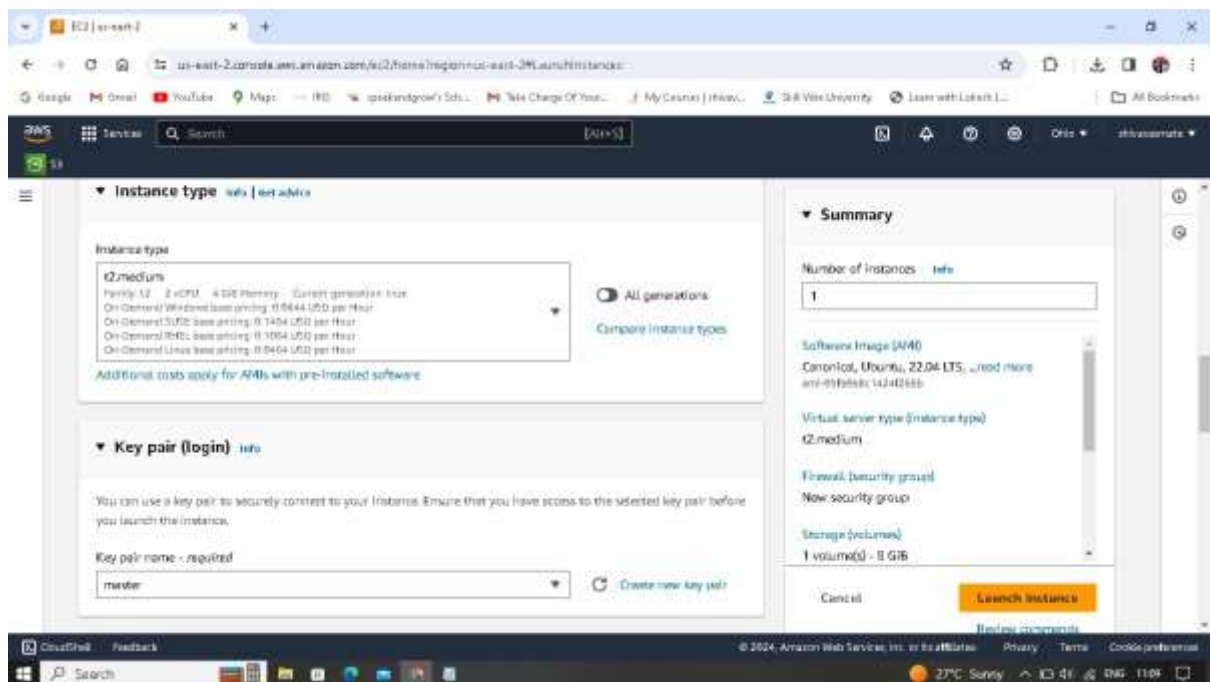


## b. AMI: QuickStart >> Ubuntu [Any version which is free tier eligible]

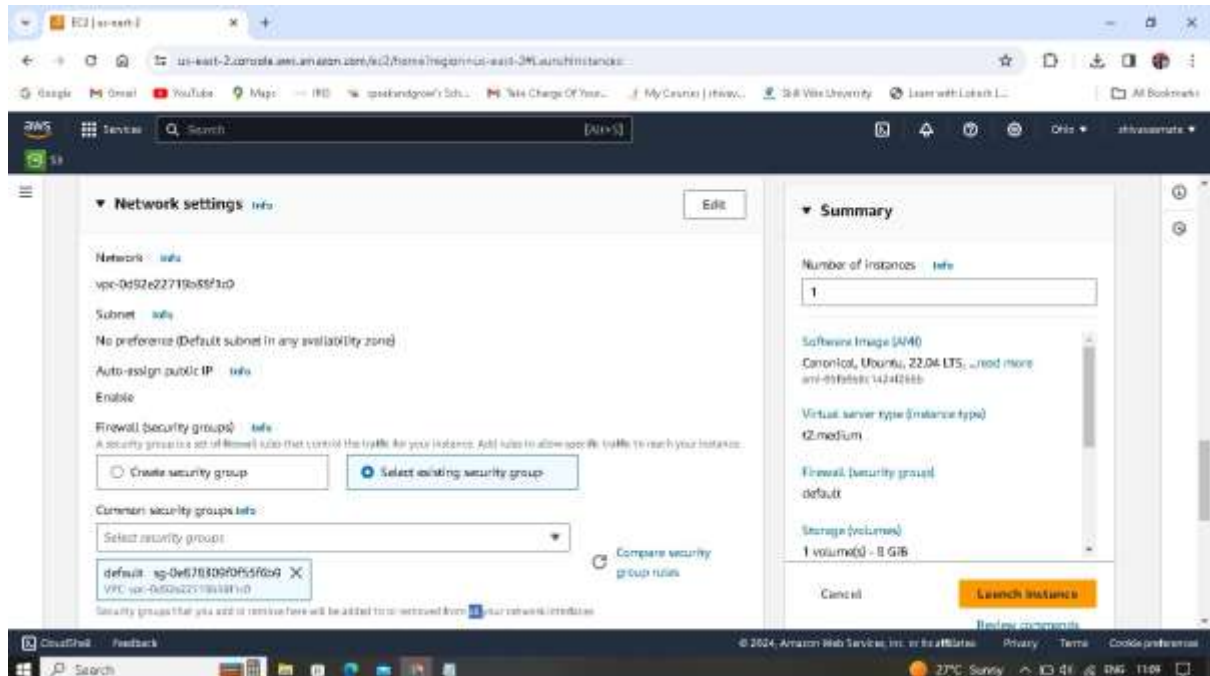


c. Instance type: t2. medium

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

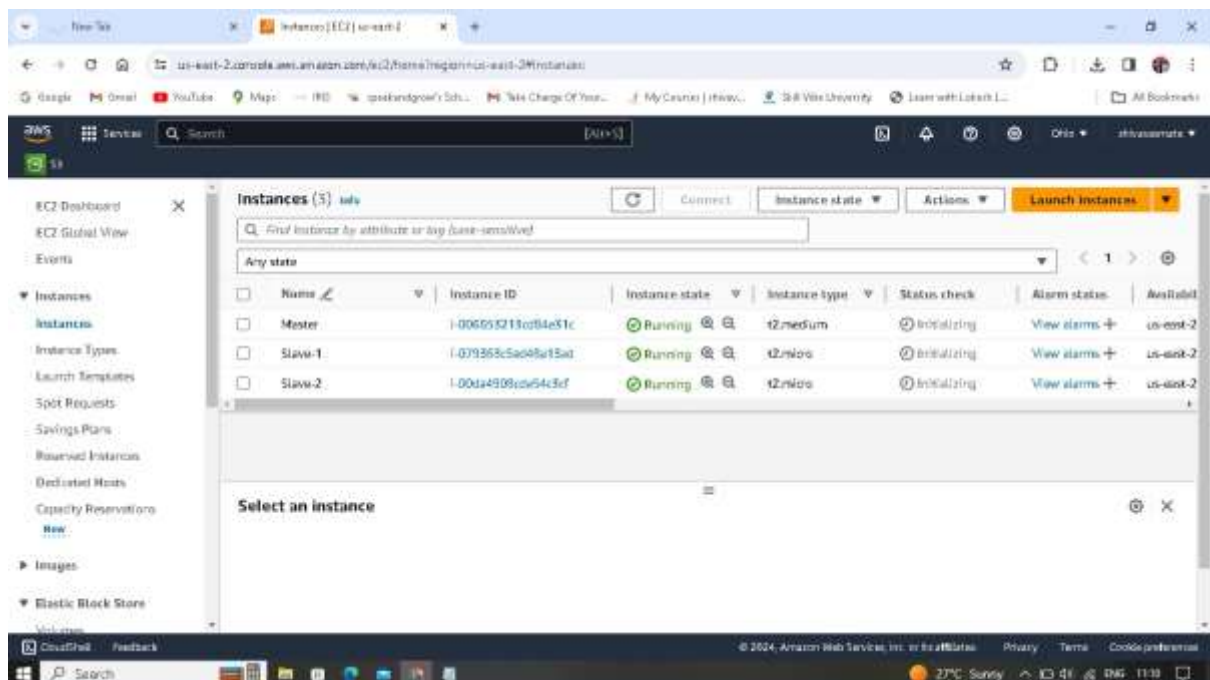


e. VPC and Security group as default with ssh and HTTP protocols click on launch instance.

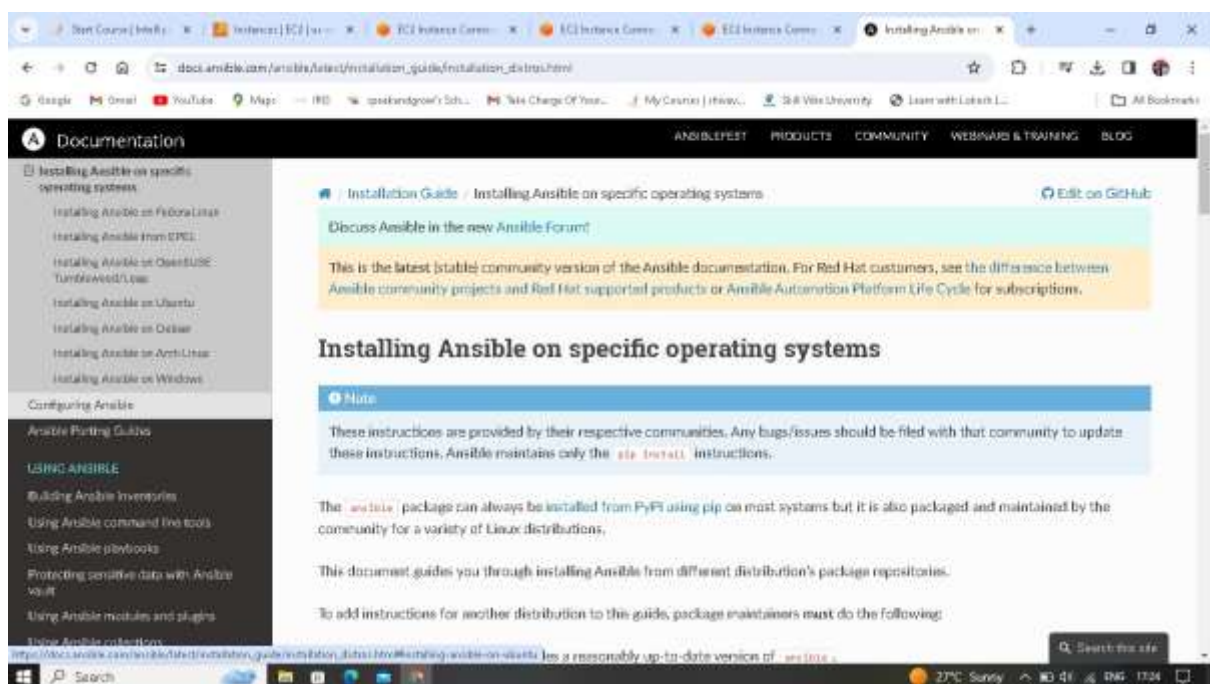




7. All three instances are in running state. Now edit name as Master, Slave1 and Slave2 .



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
```

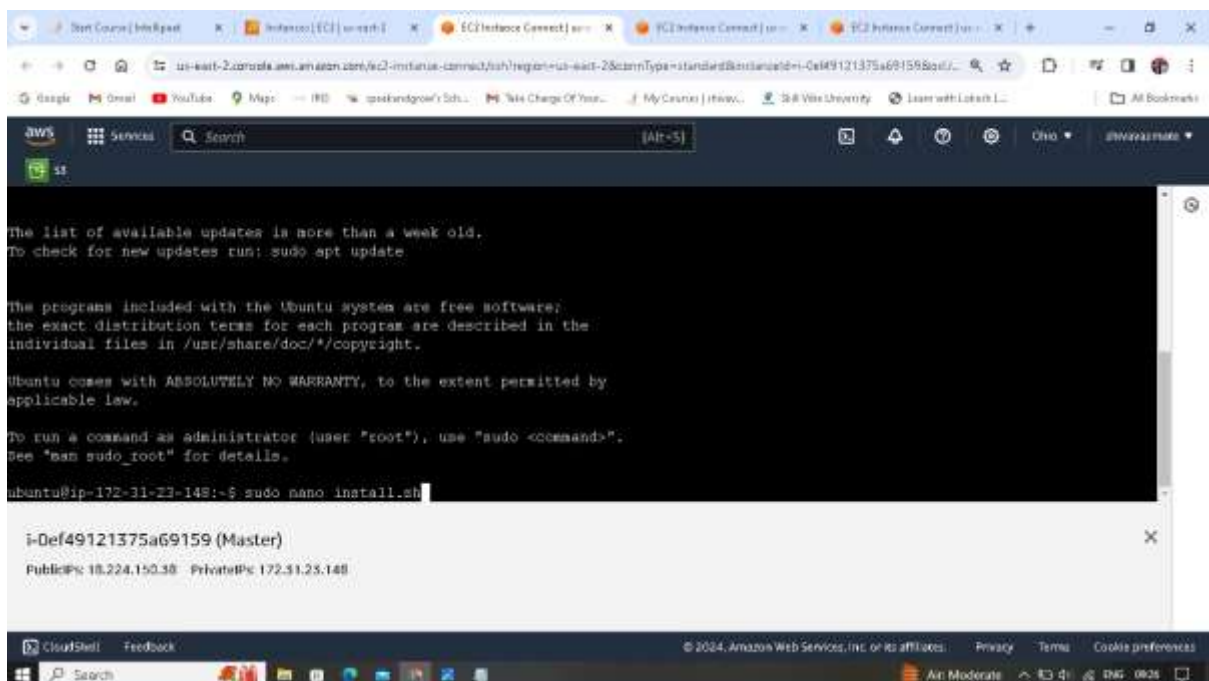
```
$ sudo apt install software-properties-common
```

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

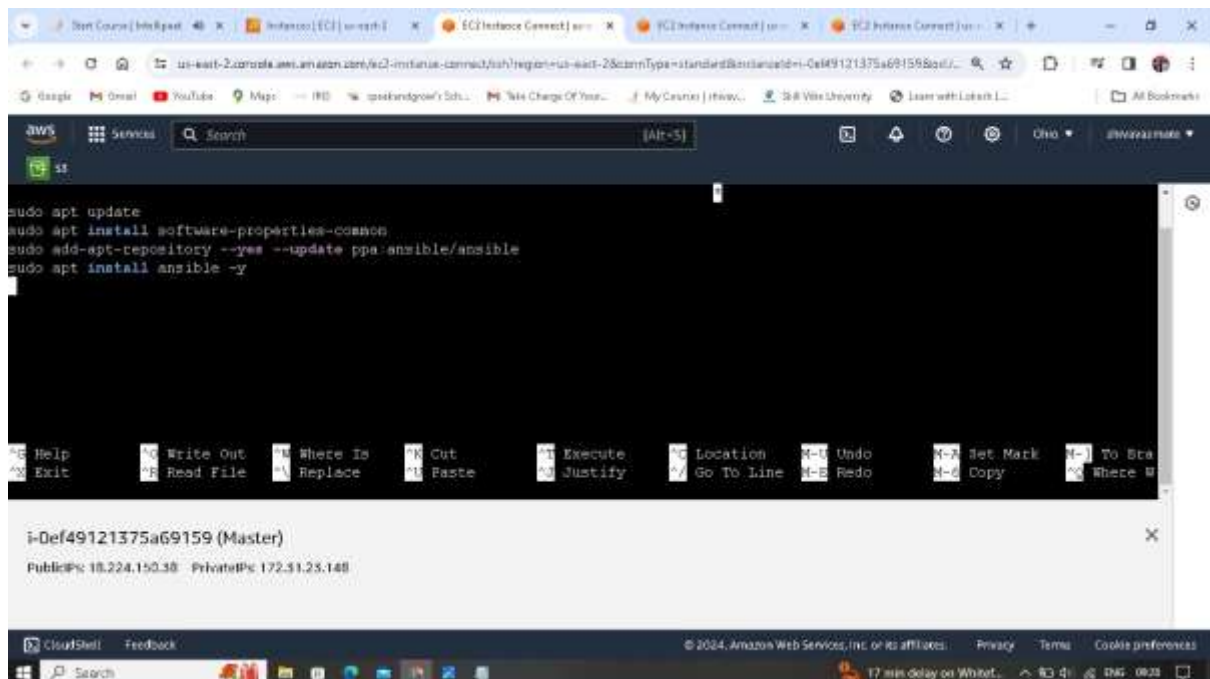
\$ sudo apt install ansible



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.

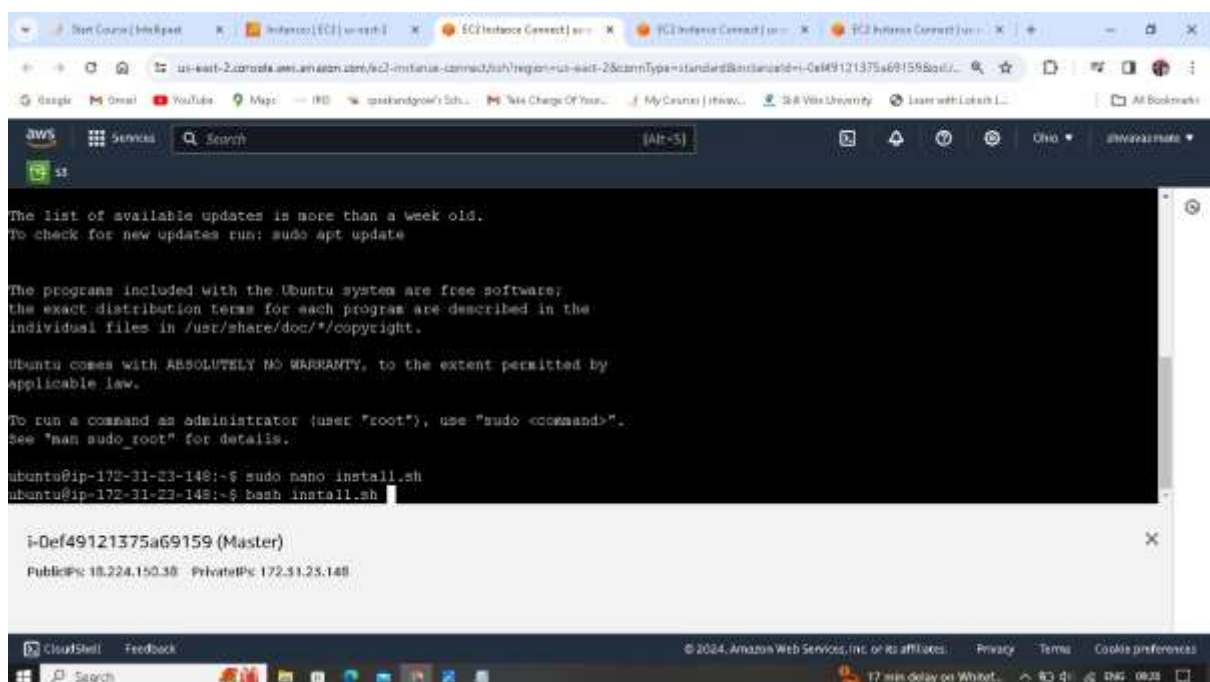


The screenshot shows an AWS CloudShell terminal window. The terminal output is as follows:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible -y
```

Below the terminal output, the instance details are shown: i-Def49121375a69159 (Master), Public IP: 18.224.150.38, Private IP: 172.31.23.148.

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



The screenshot shows the same AWS CloudShell terminal window. The terminal output is as follows:

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

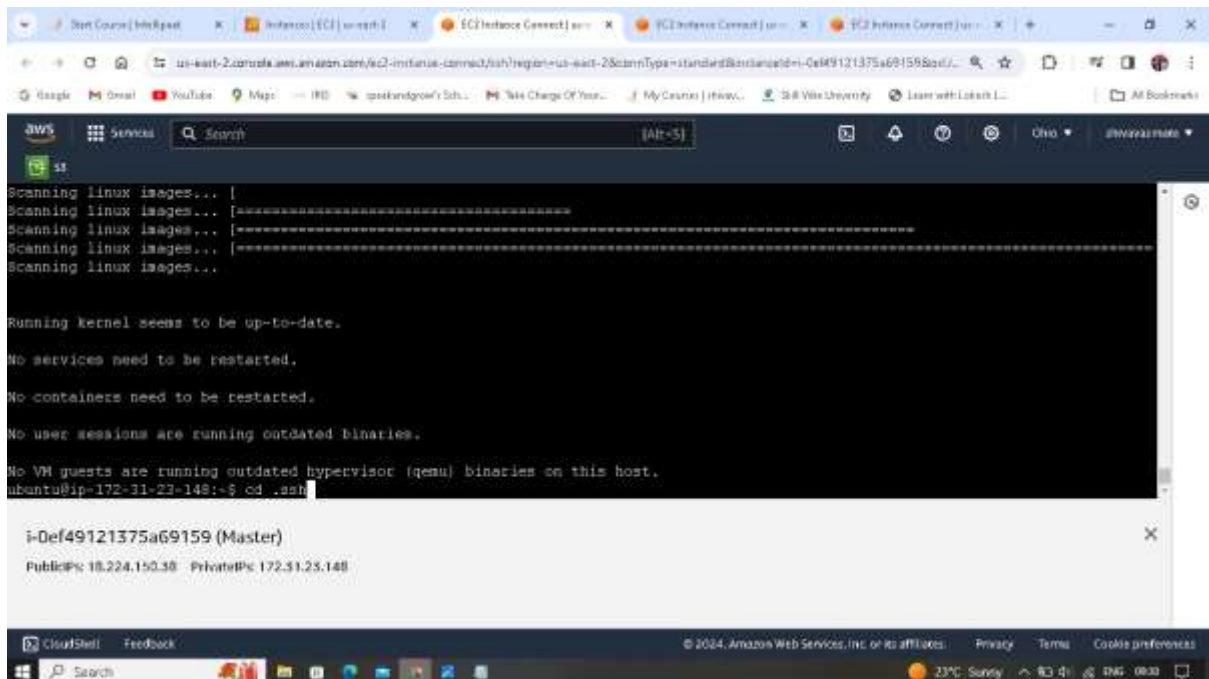
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-23-148:~$ sudo nano install.sh
ubuntu@ip-172-31-23-148:~$ bash install.sh
```

Below the terminal output, the instance details are shown: i-Def49121375a69159 (Master), Public IP: 18.224.150.38, Private IP: 172.31.23.148.



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



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output indicates that the system is up-to-date, with no services, containers, or user sessions requiring restarts. The user has entered the command `cd .ssh` to navigate to the SSH directory. A metadata box at the bottom of the terminal shows the instance ID `i-Def49121375a69159` (Master) and its public/private IP addresses: `PublicIP: 18.224.150.38` and `PrivateIP: 172.31.23.148`.

```
Scanning linux images... |
Scanning linux images... [=====]
Scanning linux images... [=====]
Scanning linux images... [=====]
Scanning linux images... [=====]

Running kernel seems to be up-to-date.

No services need to be restarted.

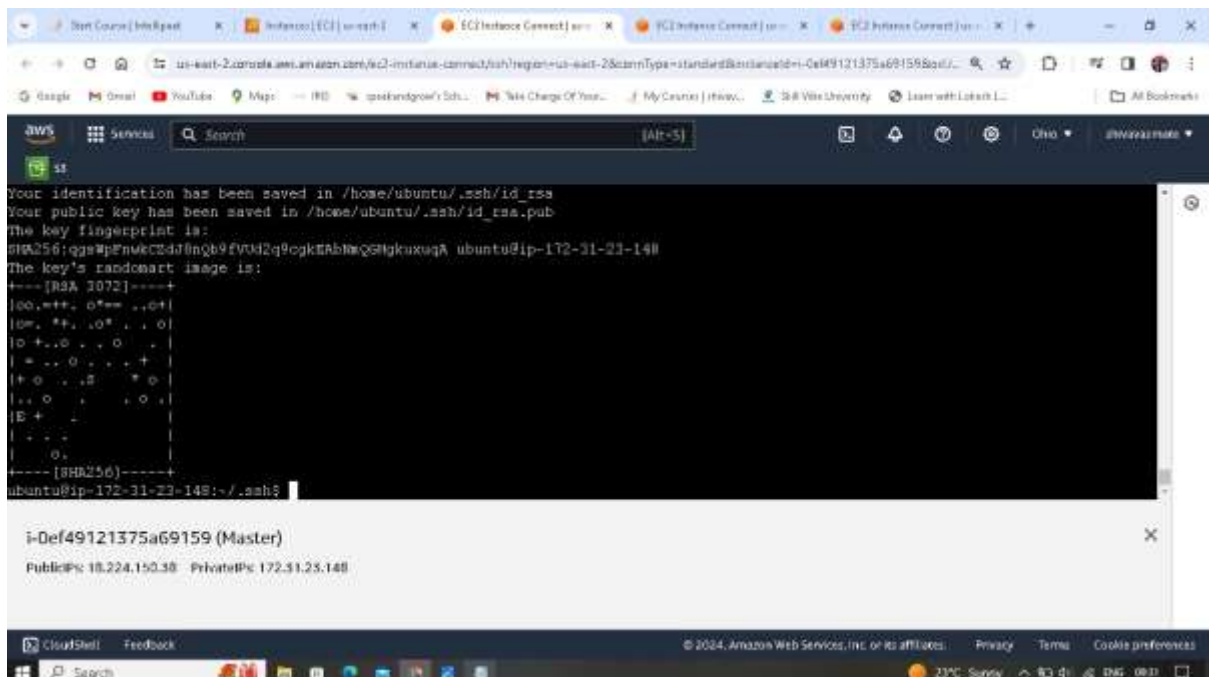
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-23-148:~$ cd .ssh
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148

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

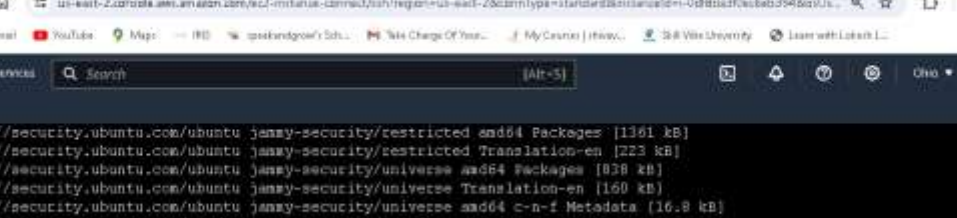


The screenshot shows the AWS CloudShell terminal where the `ssh-keygen` command has been executed. The output confirms that the keys have been saved in `/home/ubuntu/.ssh/` with filenames `id_rsa` and `id_rsa.pub`. The terminal also displays the key fingerprint and a visual representation of the key's randomart image. The user is currently in the `.ssh` directory. The same metadata box from the previous screenshot is visible at the bottom.

```
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qgsWpFmwECZddfnqb9fVvD2q9ogkERbhmQSHgkuxuQA ubuntu@ip-172-31-23-148
The key's randomart image is:
+--[RSA 3072]-----+
|oo.+++ .o*me .o+|
|om. *+ .o* . .o|
|o+.o . .o . .|
|+..o . .+ .+|
|+o . .s *o|
|..o . .o .o|
|E+ .|
| . .|
| . .|
+---[SHA256]-----+
ubuntu@ip-172-31-23-148:~/.ssh$
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148

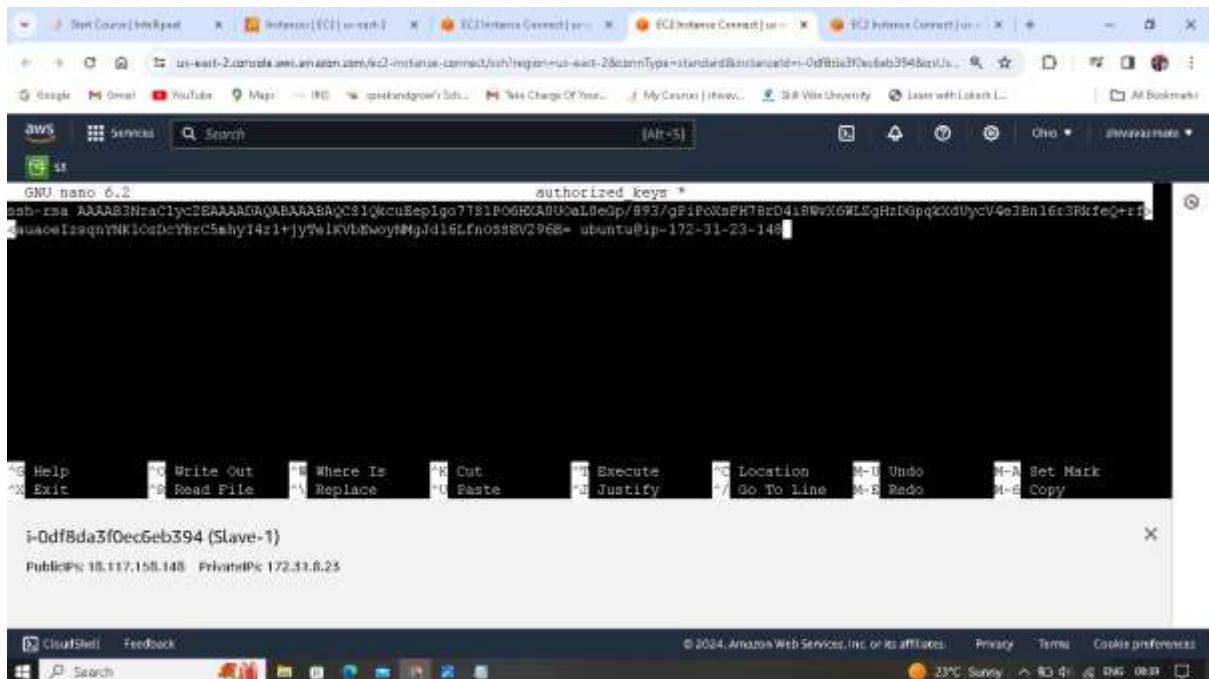
[illegible]



The screenshot shows a CloudShell terminal window with a dark background. The terminal output displays the installation of jannyssec on an Ubuntu instance. It lists the packages to be installed, their sizes, and the progress of the installation. The user is prompted to run 'apt list --upgradable' to see the status of the installed packages. The terminal also shows the user's current directory and the command to edit the authorized\_keys file.

```
Get:31 http://security.ubuntu.com/ubuntu jannys-security/restricted amd64 Packages [1361 kB]
Get:32 http://security.ubuntu.com/ubuntu jannys-security/restricted Translation-en [223 kB]
Get:33 http://security.ubuntu.com/ubuntu jannys-security/universe amd64 Packages [838 kB]
Get:34 http://security.ubuntu.com/ubuntu jannys-security/universe Translation-en [160 kB]
Get:35 http://security.ubuntu.com/ubuntu jannys-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jannys-security/multiverse amd64 Packages [37.1 kB]
Get:37 http://security.ubuntu.com/ubuntu jannys-security/multiverse Translation-en [7476 B]
Get:38 http://security.ubuntu.com/ubuntu jannys-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 29.4 MB in 5s (5471 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
59 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-8-23:~$ cd .ssh/
ubuntu@ip-172-31-8-23:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-8-23:~/.ssh$ sudo nano authorized_keys
```

17. Paste the content of the id\_rsa.pub. on the second line of the authorized\_keys save and exit.

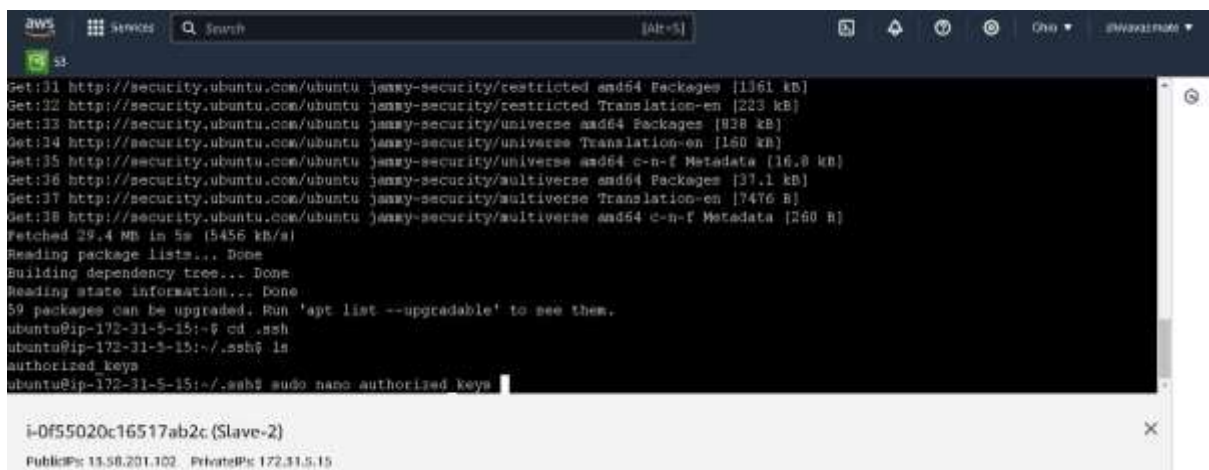


The screenshot shows an AWS CloudShell interface with a terminal window. The terminal is running the nano text editor, editing the file `authorized_keys`. The first line of the file is a public key for `ssh-rsa`. The second line is being edited, and the content `ubuntu@ip-172-31-23-148` is being pasted. The terminal output shows the file's permissions and the user's location. The AWS CloudShell interface includes a search bar, a list of services, and a navigation pane on the left.

```
GNU nano 6.2 authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA1QkcuBep1go7781P06H0ABUpaL0e0p/893/gF1PoXaPH78cD41B9wX0WLGHzDQpqcXddUycV4eJEn16cSRKfcQ+rF
4uacelzsqnYK1k0cDcY8zC5ahyI4z1+jy7MkVbEwoyMgJd16LfnoSS8V2968= ubuntu@ip-172-31-23-148
^O Help      ^O Write Out  ^O Where Is   ^O Cut        ^O Execute   ^O Location  ^O Undo      ^O Set Mark
^O Exit      ^O Read File  ^O Replace    ^O Paste      ^O Justify   ^O Go To Line^O Redo      ^O Copy
```

i-OdfBda3f0ec5eb394 (Slave-1)  
PublicIP: 18.117.158.148 PrivateIP: 172.31.8.25

18. Similarly for the slave2 also follow same steps.



The screenshot shows an AWS CloudShell interface with a terminal window. The terminal is running the `apt` command to update the package list and install packages. The output shows the progress of the update and the installation of the `ssh` package. The terminal output shows the file's permissions and the user's location. The AWS CloudShell interface includes a search bar, a list of services, and a navigation pane on the left.

```
Get:11 http://security.ubuntu.com/ubuntu jessy-security/restricted amd64 Packages [1361 kB]
Get:12 http://security.ubuntu.com/ubuntu jessy-security/restricted Translation-en [223 kB]
Get:13 http://security.ubuntu.com/ubuntu jessy-security/universe amd64 Packages [838 kB]
Get:14 http://security.ubuntu.com/ubuntu jessy-security/universe Translation-en [160 kB]
Get:15 http://security.ubuntu.com/ubuntu jessy-security/universe amd64 c-n-f Metadata [16.0 kB]
Get:16 http://security.ubuntu.com/ubuntu jessy-security/multiverse amd64 Packages [37.1 kB]
Get:17 http://security.ubuntu.com/ubuntu jessy-security/multiverse Translation-en [7476 B]
Get:18 http://security.ubuntu.com/ubuntu jessy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 29.4 MB in 5s (5456 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
59 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-5-15:~$ cd .ssh
ubuntu@ip-172-31-5-15:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-5-15:~/.ssh$ sudo nano authorized_keys
```

i-Of55020c16517ab2c (Slave-2)  
PublicIP: 13.58.201.102 PrivateIP: 172.31.5.15

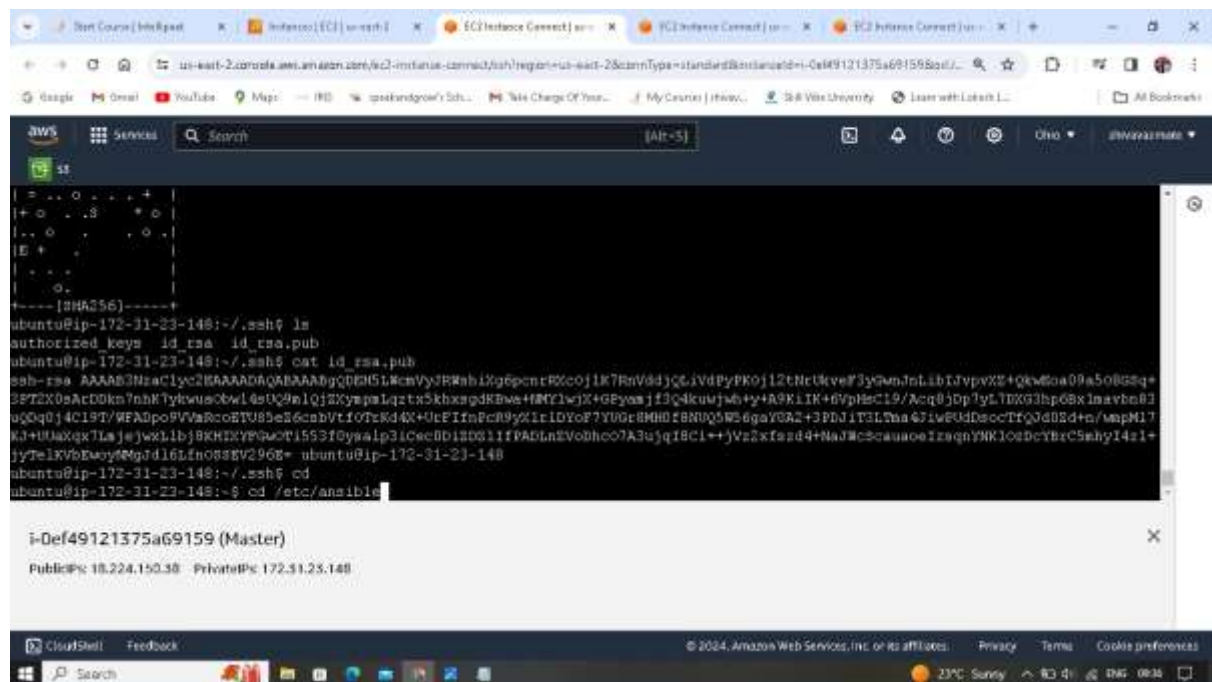


The screenshot shows an AWS CloudShell interface with a terminal window. The terminal is running the nano text editor, editing the file `authorized_keys`. The first line of the file is a public key for `ssh-rsa`. The second line is being edited, and the content `ubuntu@ip-172-31-23-148` is being pasted. The terminal output shows the file's permissions and the user's location. The AWS CloudShell interface includes a search bar, a list of services, and a navigation pane on the left.

```
GNU nano 6.2 authorized_keys *
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA1QkcuBep1go7781P06H0ABUpaL0e0p/893/gF1PoXaPH78cD41B9wX0WLGHzDQpqcXddUycV4eJEn16cSRKfcQ+rF
4uacelzsqnYK1k0cDcY8zC5ahyI4z1+jy7MkVbEwoyMgJd16LfnoSS8V2968= ubuntu@ip-172-31-23-148
^O Help      ^O Write Out  ^O Where Is   ^O Cut        ^O Execute   ^O Location  ^O Undo      ^O Set Mark
^O Exit      ^O Read File  ^O Replace    ^O Paste      ^O Justify   ^O Go To Line^O Redo      ^O Copy
```

i-Of55020c16517ab2c (Slave-2)  
PublicIP: 13.58.201.102 PrivateIP: 172.31.5.15

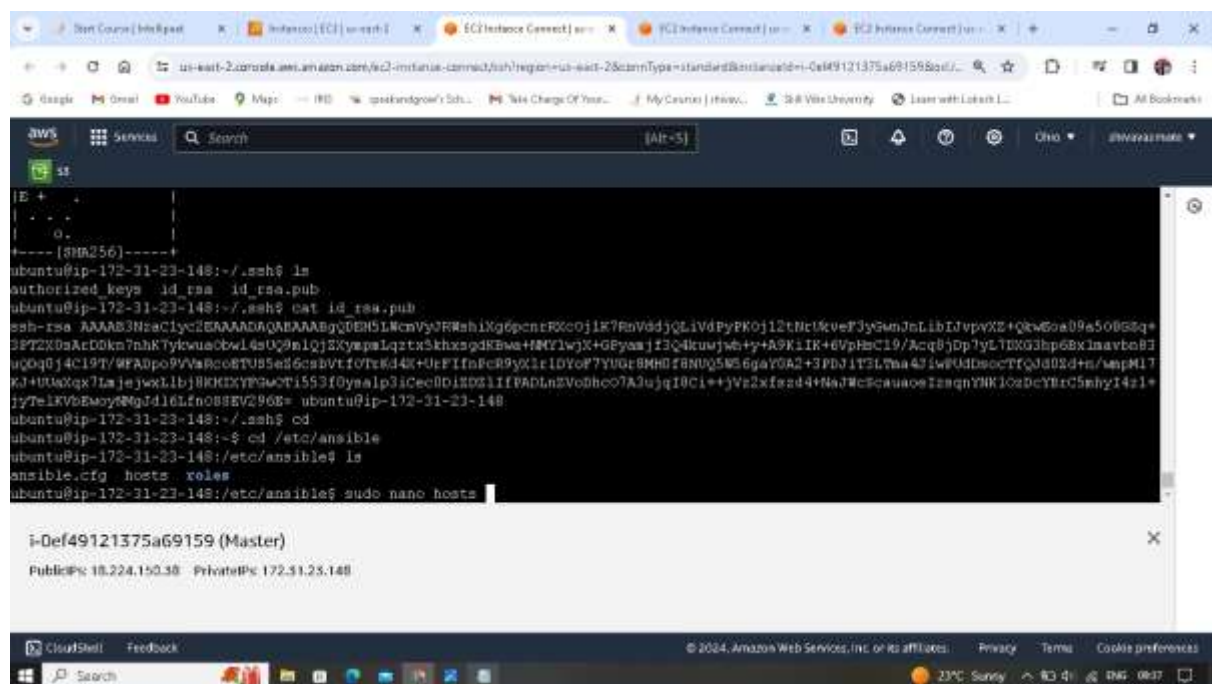
## 19. Go out of .ssh by cd command



```
ubuntu@ip-172-31-23-148:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub
ubuntu@ip-172-31-23-148:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQB0H5LWcnVYjRRshIXg6pcnrR0coj1K7RnVddjQCLiVdPyPK0j12tNcUkeFJyGwnJmLibJvvpvX2+QkwG0a09a50B8sq+
3PT2X0sAcD0km7nhKTykvwObwL4sUQ9e1QjEXyapeLqztX5kxspdkEwa+HMYLwJX+GPYaaJf3Q4kuwJw+y+A9K1IK+6VpHeC19/Acq8jDp1yLT0XG3hp68Xlnavbn83
uQ0q0j4C19T/WFA0p0VvVaRcoETU85eS6sbVtfoTeKd4X+UeFIfnF0c89yX1rIDYoF7YUGr8MH0f8NUQ5W56gaY0A2+3PDJ1T3LTma43iWpUdDeocTrQd8Sd+n/vapM17
KJ+UuaXqx7IaJeJwXl1bJ8KHIXYF0w0T1553f0ymalp3iCec0DiSD011fPADLnZVobhco7A3uJqI0C1++jVz2xfsd4+NajWcscuawoeIzegnYMKJ0aDcYRc5mhyI4z1+
jyTelKVbEwoy8Mgd16lfn088EV2968= ubuntu@ip-172-31-23-148
ubuntu@ip-172-31-23-148:~/.ssh$ cd
ubuntu@ip-172-31-23-148:~$ cd /etc/ansible
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148

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

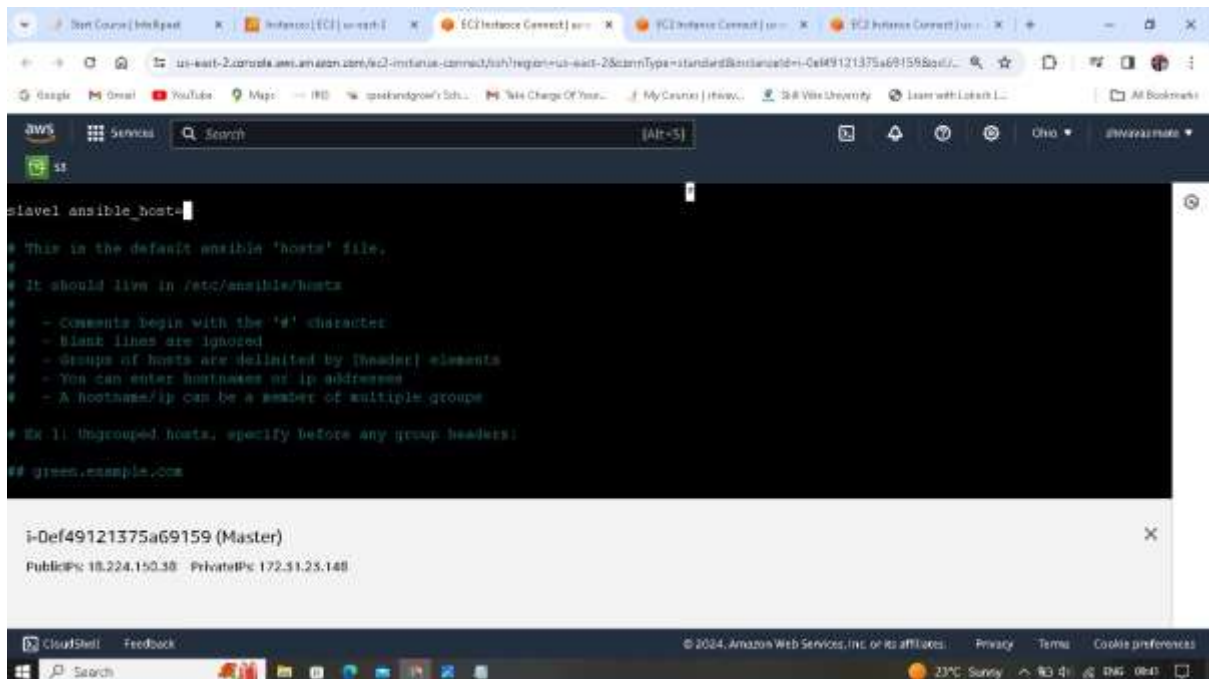


```
ubuntu@ip-172-31-23-148:~$ cd /etc/ansible
ubuntu@ip-172-31-23-148:/etc/ansible$ ls
ansible.cfg  hosts  roles
ubuntu@ip-172-31-23-148:/etc/ansible$ sudo nano hosts
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148



21. Inside it type slave1 ansible\_host= private IP of slave1 machine



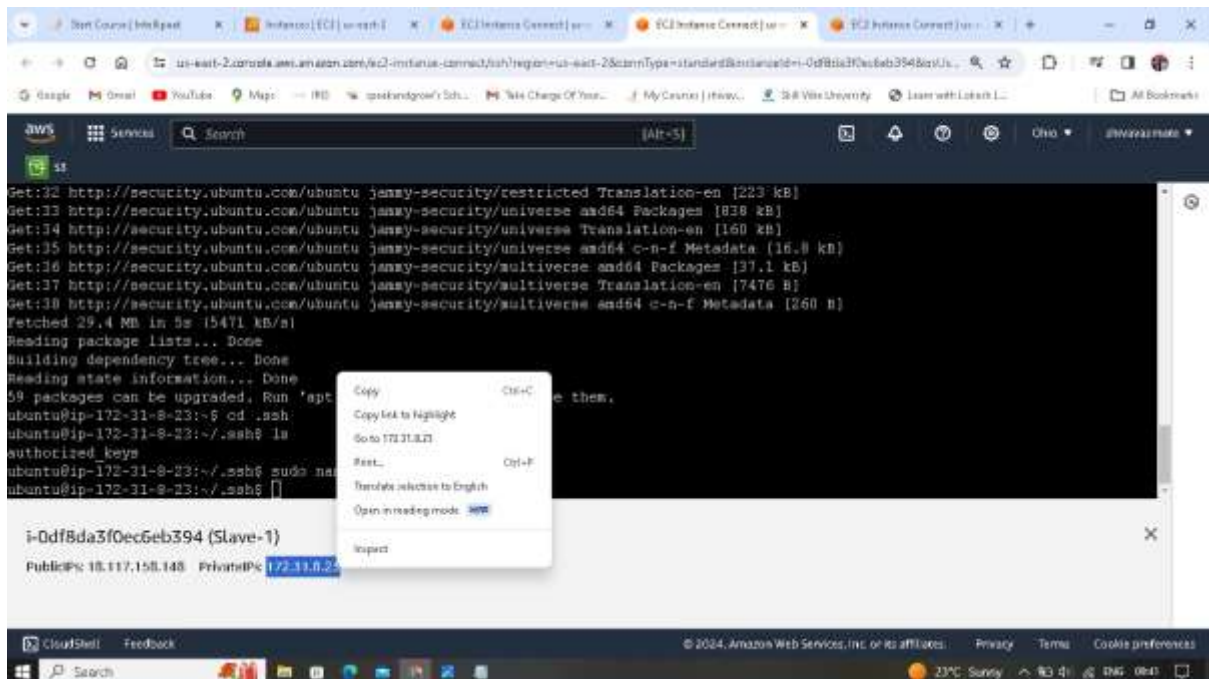
The screenshot shows the AWS CloudShell interface with a terminal window. The terminal displays the content of the `ansible.cfg` file, which is a configuration file for Ansible. The file includes comments explaining its structure and usage. The terminal output shows the file content as follows:

```
slave1 ansible_host=

# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers:
## green.example.com
```

Below the terminal window, a metadata box for the master node is visible, showing the instance ID `i-Def49121375a69159` and its public IP address `18.224.150.38`.

22. Go to Slave1 machine copy the private IP



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal displays the output of the `apt-get install ansible` command, which installs the Ansible package on the slave1 machine. The output shows the package being downloaded and installed, along with its dependencies. The terminal output is as follows:

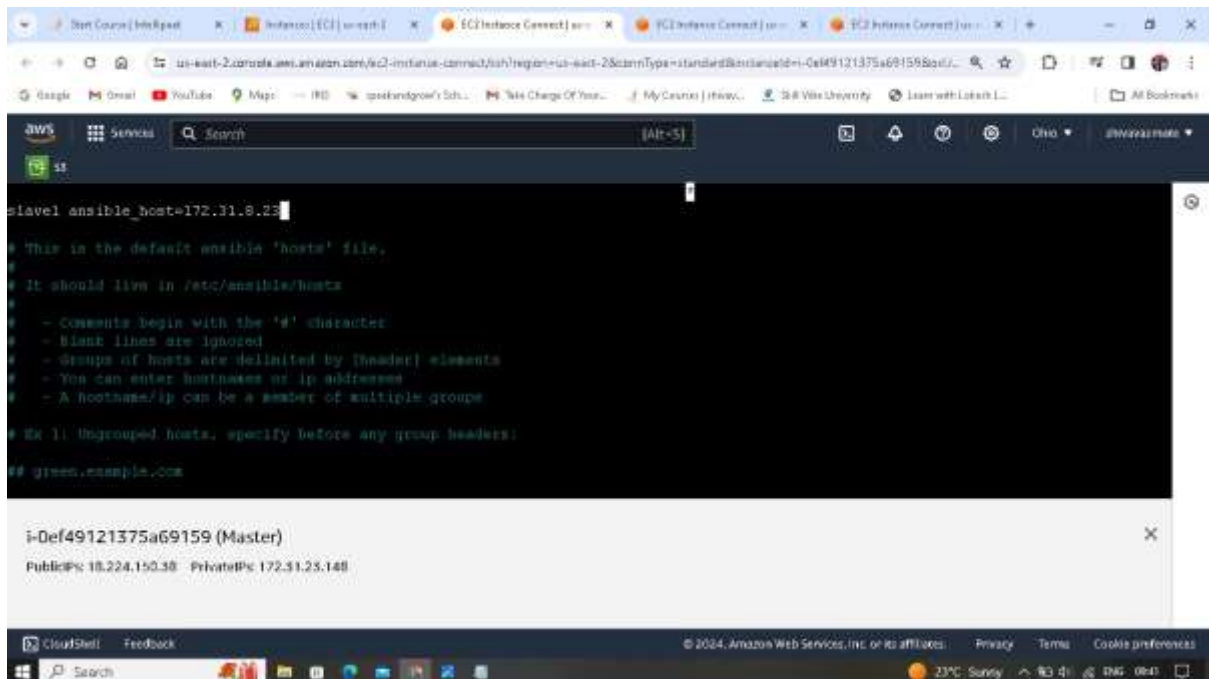
```
Get:12 http://security.ubuntu.com/ubuntu jenny-security/restricted Translation-en [223 kB]
Get:13 http://security.ubuntu.com/ubuntu jenny-security/universe amd64 Packages [838 kB]
Get:14 http://security.ubuntu.com/ubuntu jenny-security/universe Translation-en [160 kB]
Get:15 http://security.ubuntu.com/ubuntu jenny-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:16 http://security.ubuntu.com/ubuntu jenny-security/multiverse amd64 Packages [37.1 kB]
Get:17 http://security.ubuntu.com/ubuntu jenny-security/multiverse Translation-en [7476 B]
Get:18 http://security.ubuntu.com/ubuntu jenny-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 29.4 MB in 5s (5471 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
59 packages can be upgraded. Run 'apt update' to see them.
ubuntu@ip-172-31-8-23:~$ cd .ssh
ubuntu@ip-172-31-8-23:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-8-23:~/.ssh$ sudo apt install ansible
ubuntu@ip-172-31-8-23:~/.ssh$
```

A context menu is open over the terminal output, showing options like "Copy", "Copy link to highlight", "Go to 172.31.8.23", "Paste...", "Translate selection to English", "Open in reading mode", and "Inspect".

Below the terminal window, a metadata box for the slave1 node is visible, showing the instance ID `i-0df8da3f0ec6eb394` and its private IP address `172.31.8.23`.

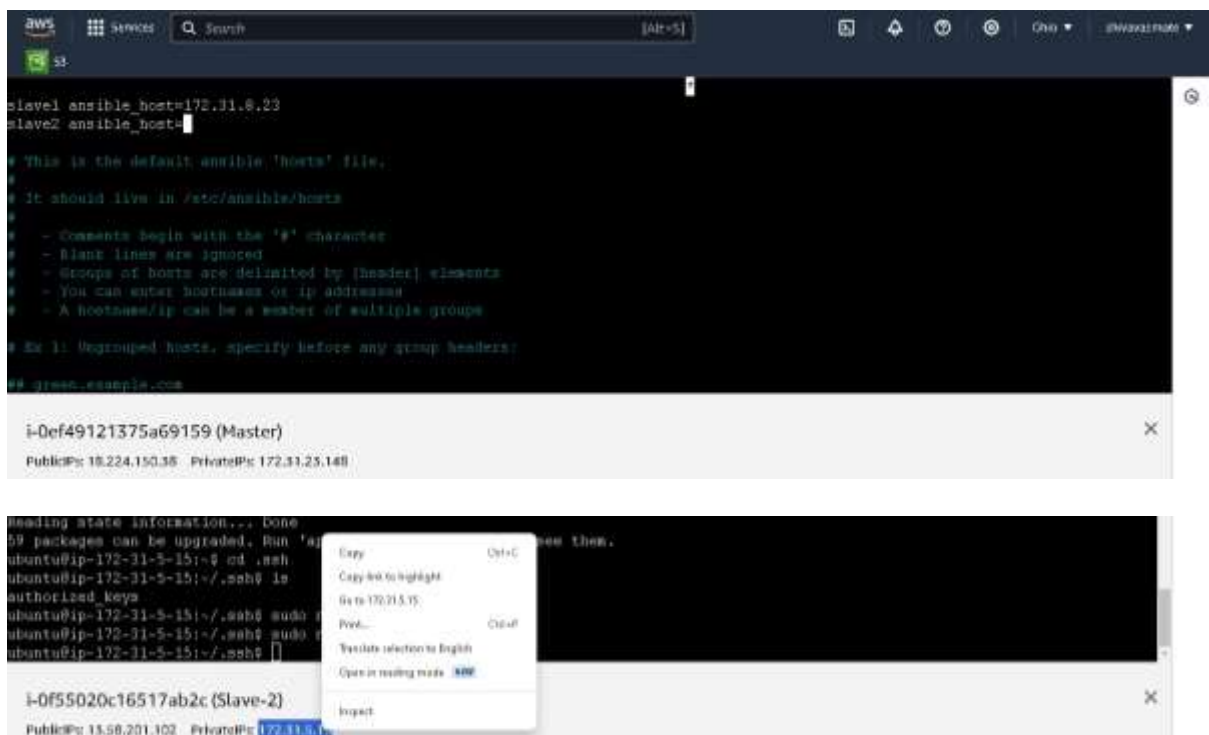


23. Paste it.



The screenshot shows the AWS CloudShell interface. The terminal window displays the command `slave1 ansible_host=172.31.8.23` and the contents of the `/etc/ansible/hosts` file. The file contains comments explaining the format and an example entry `## green.example.com`. Below the terminal, the instance details for `i-Def49121375a69159 (Master)` are shown, with Public IP `18.224.150.38` and Private IP `172.31.25.148`. The bottom of the screen shows the Windows taskbar with a search bar and system tray icons.

24. Similarly for Slave2 machine



The screenshot shows the AWS CloudShell interface. The terminal window displays the command `slave2 ansible_host=` followed by a paste operation. The terminal also shows the contents of the `/etc/ansible/hosts` file. Below the terminal, the instance details for `i-0f55020c16517ab2c (Slave-2)` are shown, with Public IP `15.58.201.102` and Private IP `172.31.8.15`. A context menu is visible over the terminal, showing options like 'Copy', 'Copy to highlight', 'Paste', and 'Inspect'. The bottom of the screen shows the Windows taskbar with a search bar and system tray icons.

```
aws
services
Search
[Alt+S]

slave1 ansible_host=172.31.0.23
slave2 ansible_host=172.31.5.15

# This is the default ansible 'hosts' file:
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers:
#
green.example.com
```

i-0ef49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.25.148

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

```
aws
services
Search
[Alt+S]

+-----[SHA256]-----+
ubuntu@ip-172-31-23-148:~$ ls
authorized_keys id_rsa id_rsa.pub
ubuntu@ip-172-31-23-148:~$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAUQ9M1QjEXYpmlqstx5khxegdkwa+M4YlwjX+GPyasjI3Q4kuujsdy+A9K1K+6VpHwC19/Acq8jOp7yL7DXGJhp68Imavbe83
uQOQ0j4C19T/WRADp09VvARcoTUV85eS6cbvtf0TcKd4X+UcFifnFcr89yXirIDYoF7YUGt8MH0f8NUQ5W56gaY0A2+3PDJIT3L7na67lwFHDDeocTfQd8Zd+n/vmpM11
KJ+UuaXqx7LajejwL1bJ8KHXYFGwOTi553f0ysalp3iCecUDiSD211fFADLnTVOhcc07A3u3jqI8C1++jV2xfzsd4+NaJWcScasuoceIzaqnYMK10zDcyBrC5ahyI4z1+
jy7e1KvBwpy4Mqjd16lfn088W2968= ubuntu@ip-172-31-23-148
ubuntu@ip-172-31-23-148:~$ cd /etc/ansible
ubuntu@ip-172-31-23-148:/etc/ansible$ ls
ansible.cfg hosts roles
ubuntu@ip-172-31-23-148:/etc/ansible$ sudo nano hosts
ubuntu@ip-172-31-23-148:/etc/ansible$ sudo nano hosts
ubuntu@ip-172-31-23-148:/etc/ansible$ ansible -m ping all
```

i-0ef49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.25.148

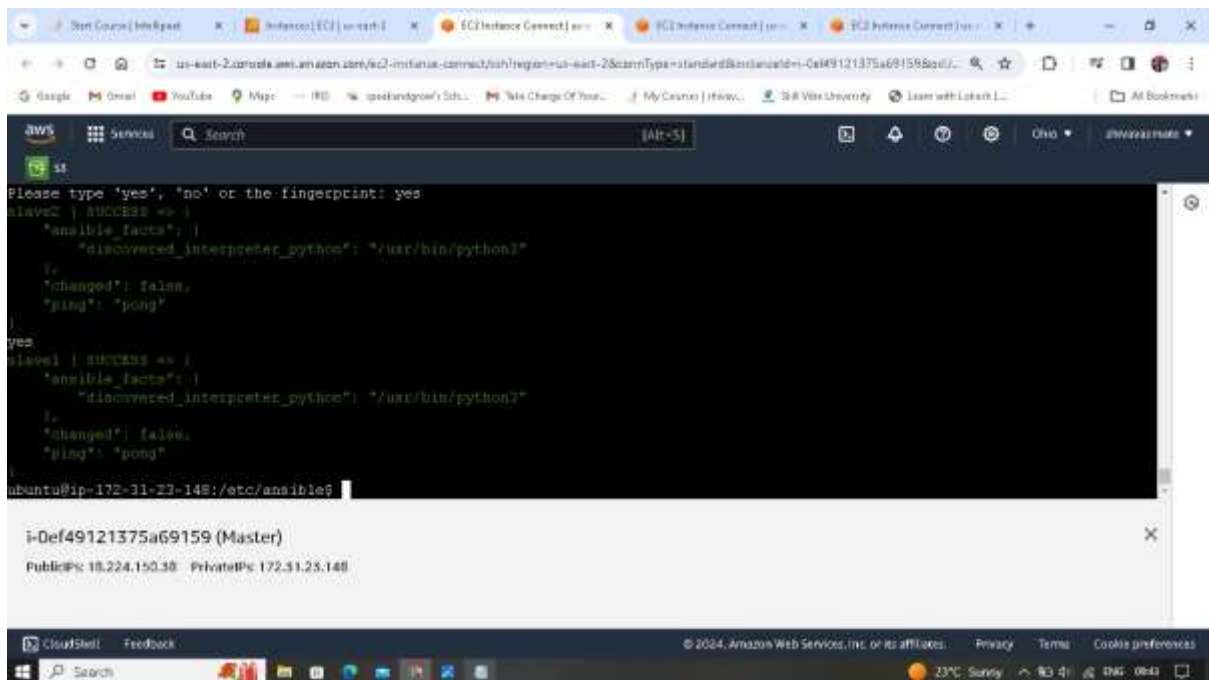
26. Type yes for slave2 machine

```
aws
services
Search
[Alt+S]

ubuntu@ip-172-31-23-148:/etc/ansible$ ansible -m ping all
The authenticity of host '172.31.0.23 (172.31.0.23)' can't be established.
RD25519 key fingerprint is SHA256:NfGou+dPfovalRRDqMXXN8k0egIKR8Na5hfyaChtalk.
This key is not known by any other names
The authenticity of host '172.31.5.15 (172.31.5.15)' can't be established.
RD25519 key fingerprint is SHA256:lnX2pSk0dt+k5uvJfnd7qnXJUikt4jDob8r2p07111.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
slave2 | SUCCESS => |
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
```

i-0ef49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.25.148

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

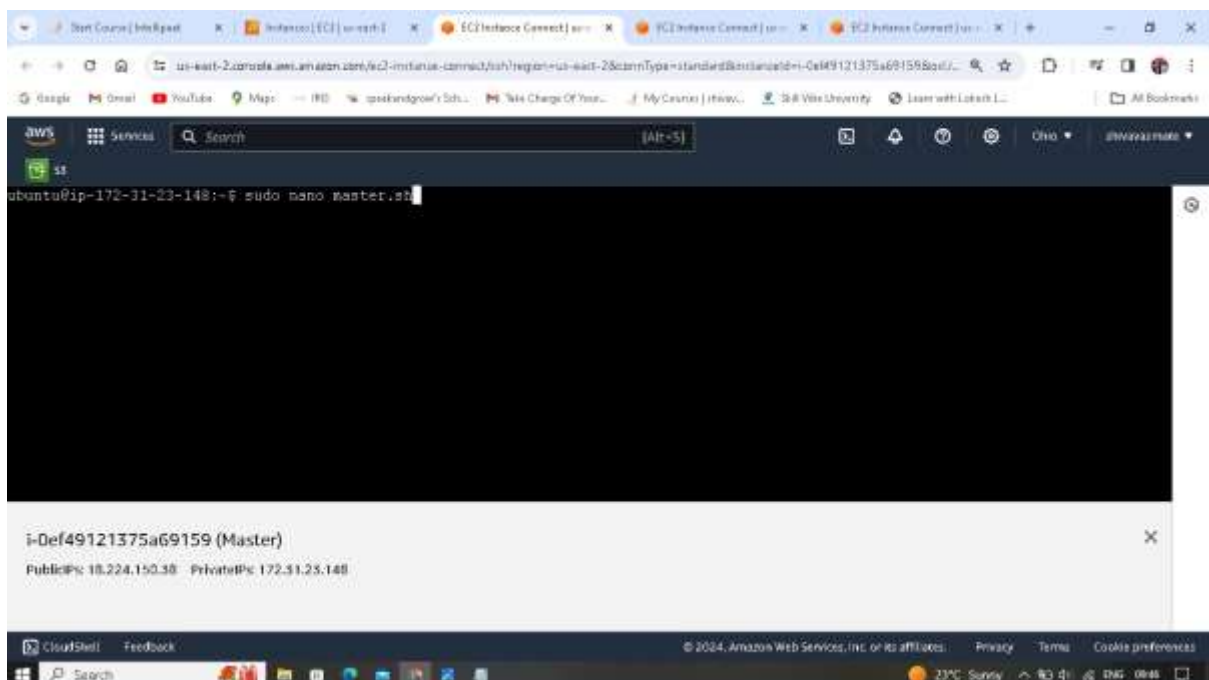


The screenshot shows an AWS CloudShell terminal window. The terminal output displays the execution of an Ansible playbook for two slave machines. The first machine, slave2, is successfully connected and configured. The second machine, slave1, is also successfully connected and configured. The terminal output shows the following commands and results:

```
Please type 'yes', 'no' or the fingerprint: yes
slave2 | SUCCESS => |
  "ansible_facts": |
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
yes
slave1 | SUCCESS => |
  "ansible_facts": |
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
ubuntu@ip-172-31-23-148:/etc/ansible$
```

The terminal window also shows the instance ID i-Def49121375a69159 (Master) and its public and private IP addresses: Public IP: 18.224.150.38, Private IP: 172.31.25.148.

28. Open an editor by sudo nano master.sh

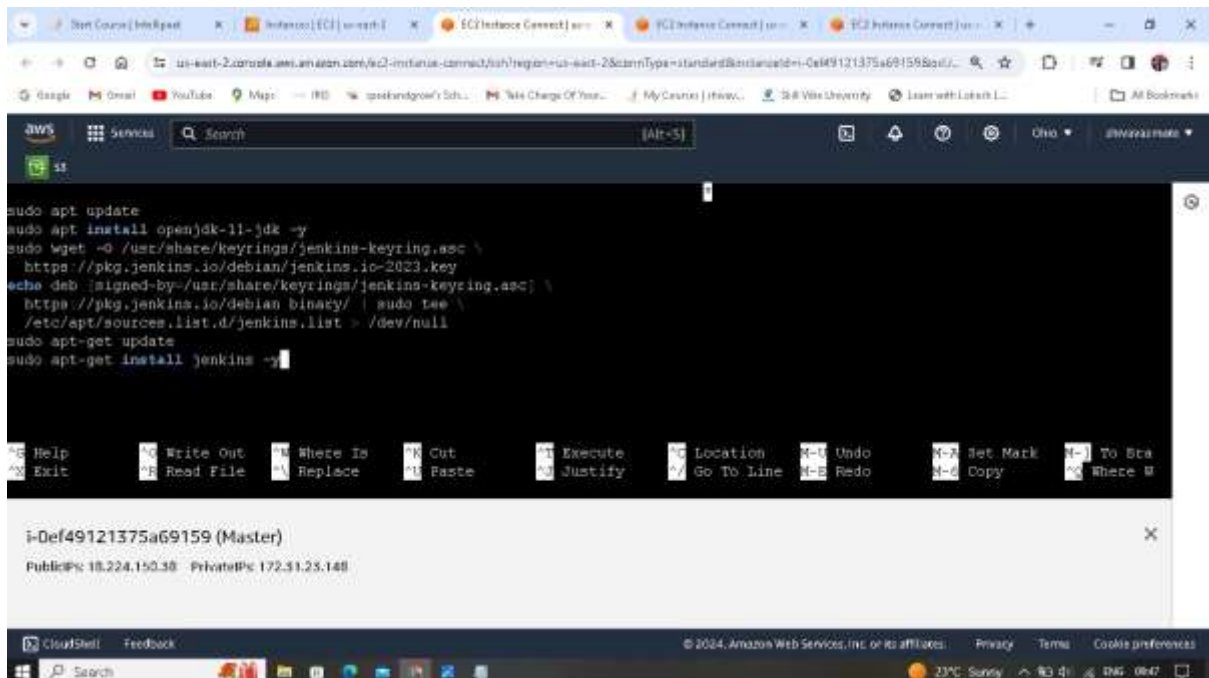


The screenshot shows an AWS CloudShell terminal window. The terminal output displays the command to open a nano editor for master.sh. The terminal output shows the following command and result:

```
ubuntu@ip-172-31-23-148:~$ sudo nano master.sh
```

The terminal window also shows the instance ID i-Def49121375a69159 (Master) and its public and private IP addresses: Public IP: 18.224.150.38, Private IP: 172.31.25.148.

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



```
sudo apt update
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
```

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

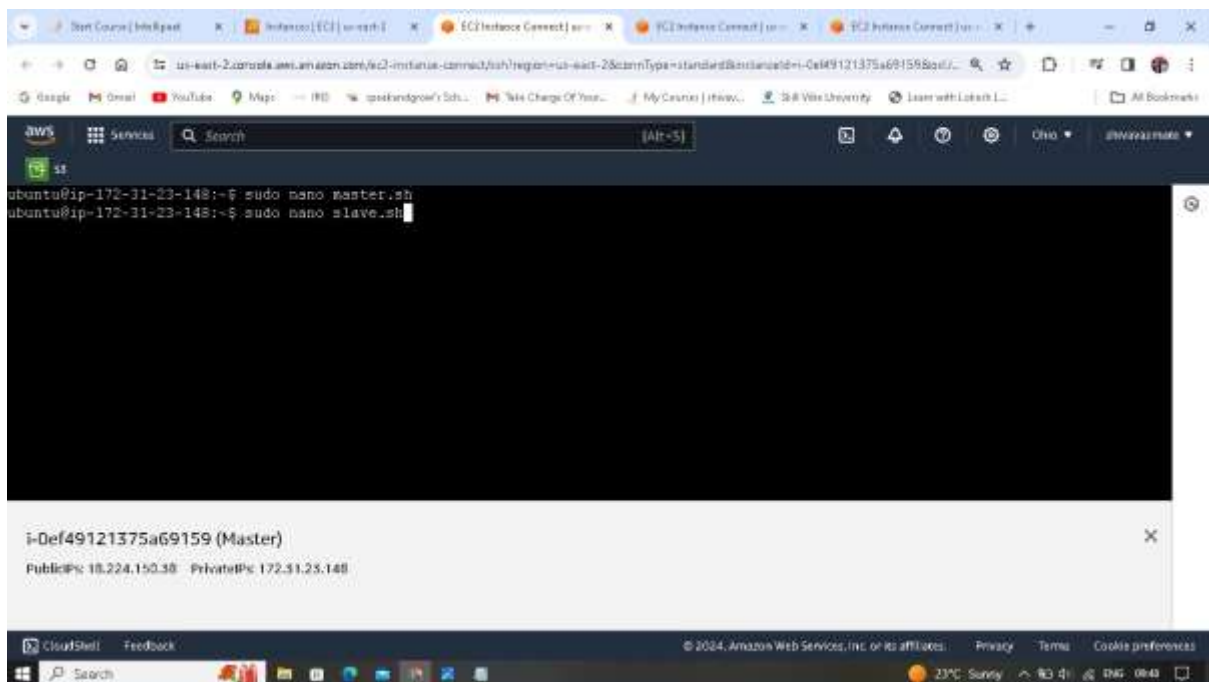
sudo apt-get install jenkins -y

ctrl + s to save

and

ctrl + x to exit.

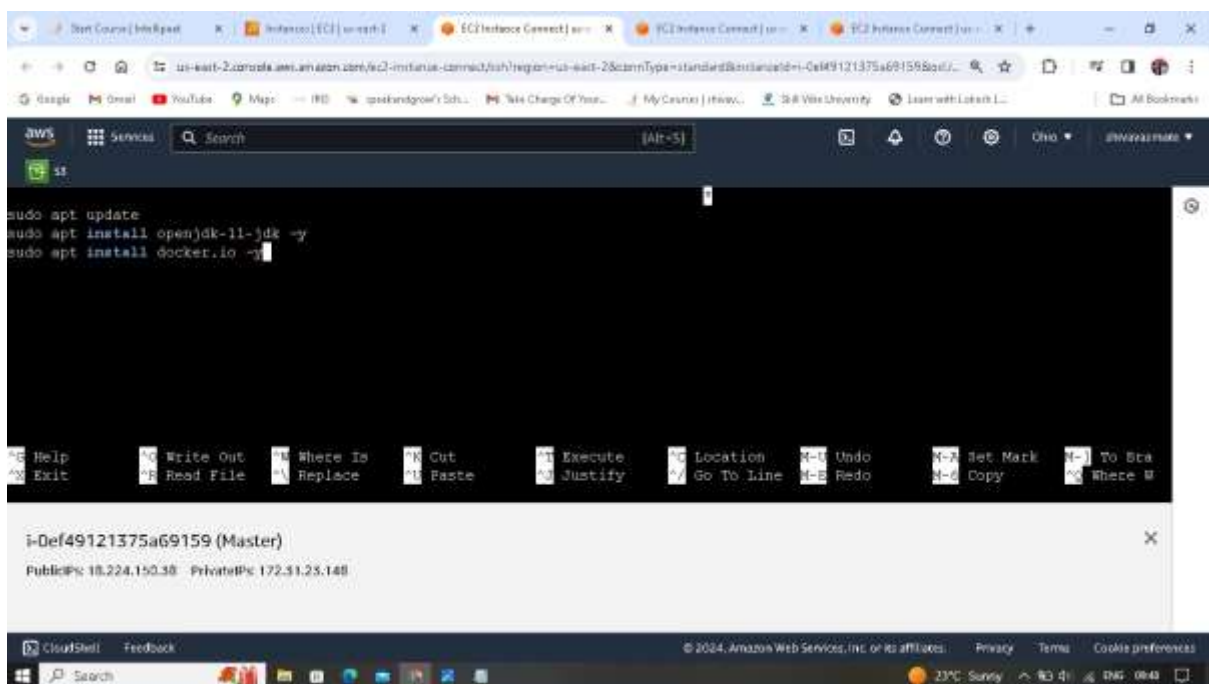
30. Open an editor by sudo nano slave.sh



```
ubuntu@ip-172-31-23-148:~$ sudo nano master.sh
ubuntu@ip-172-31-23-148:~$ sudo nano slave.sh
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148

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

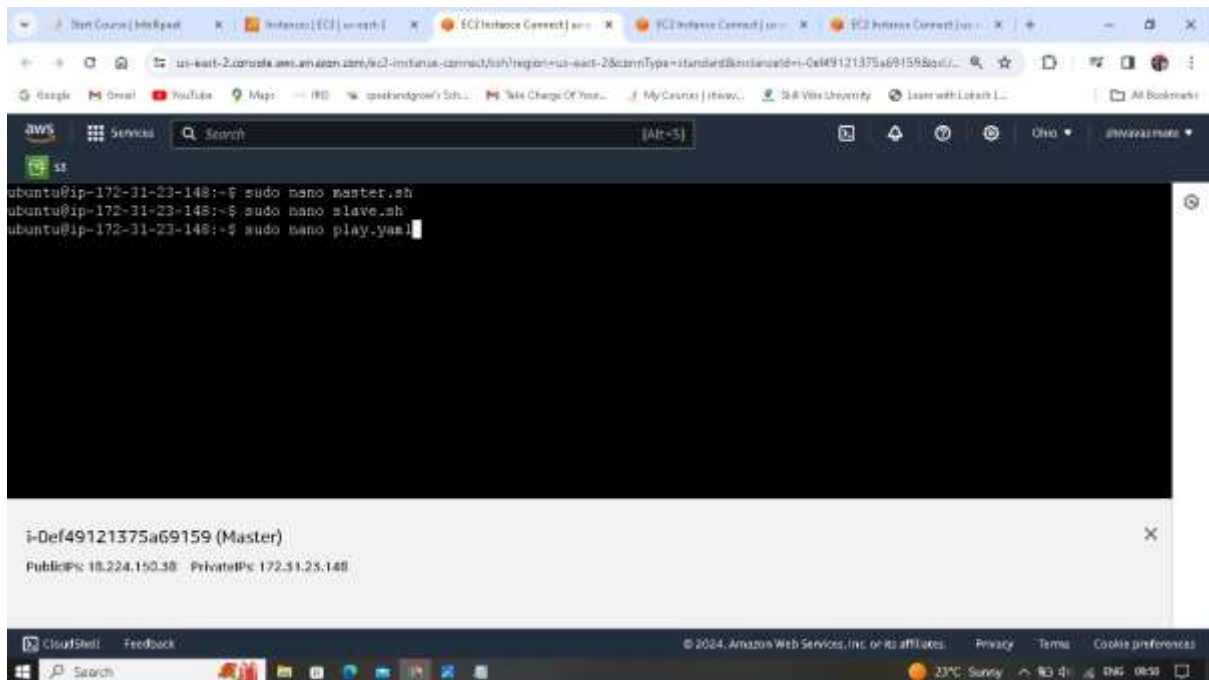


```
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt install docker.io -y
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148



### 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

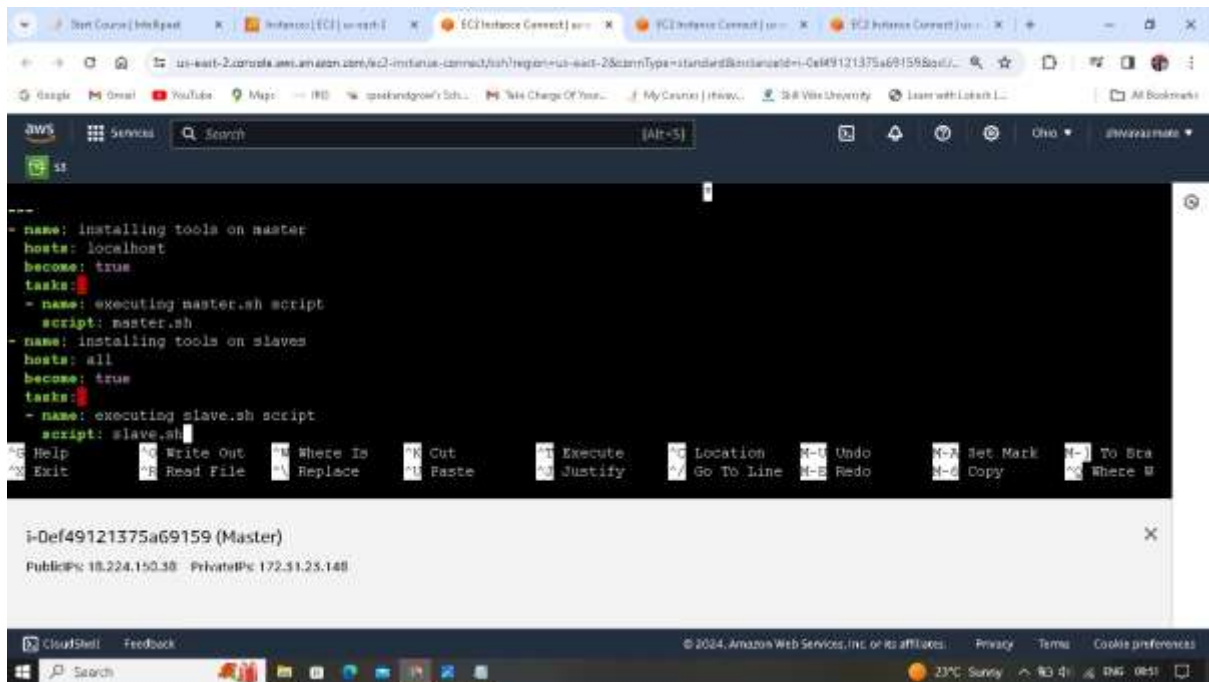
hosts: all

become: true

tasks:

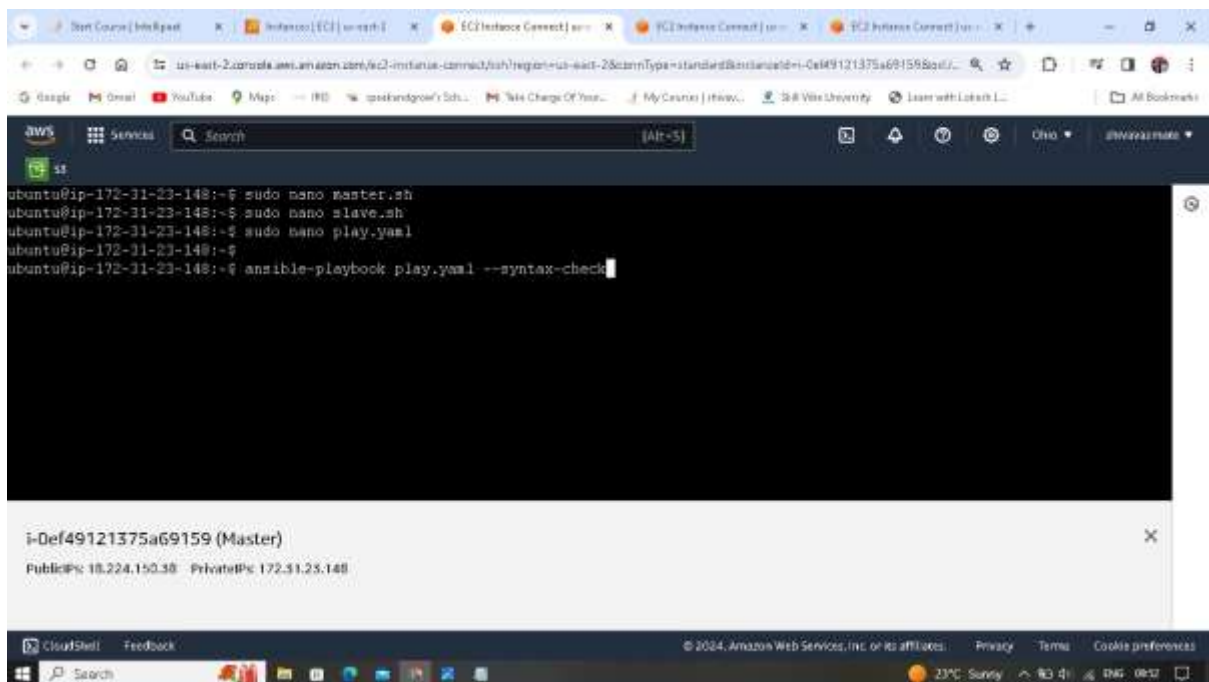
- name: executing slave.sh script

script: slave.sh

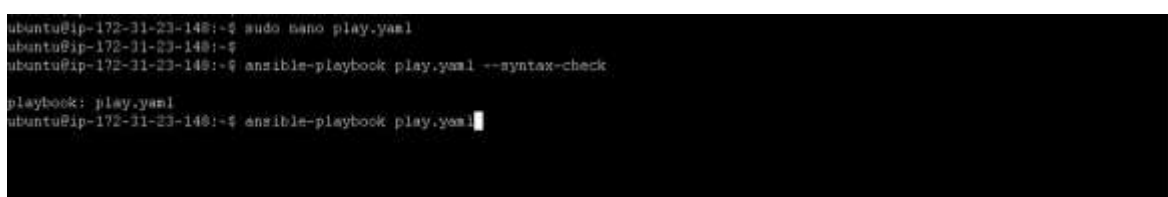


ctrl + s to save and ctrl + x to exit.

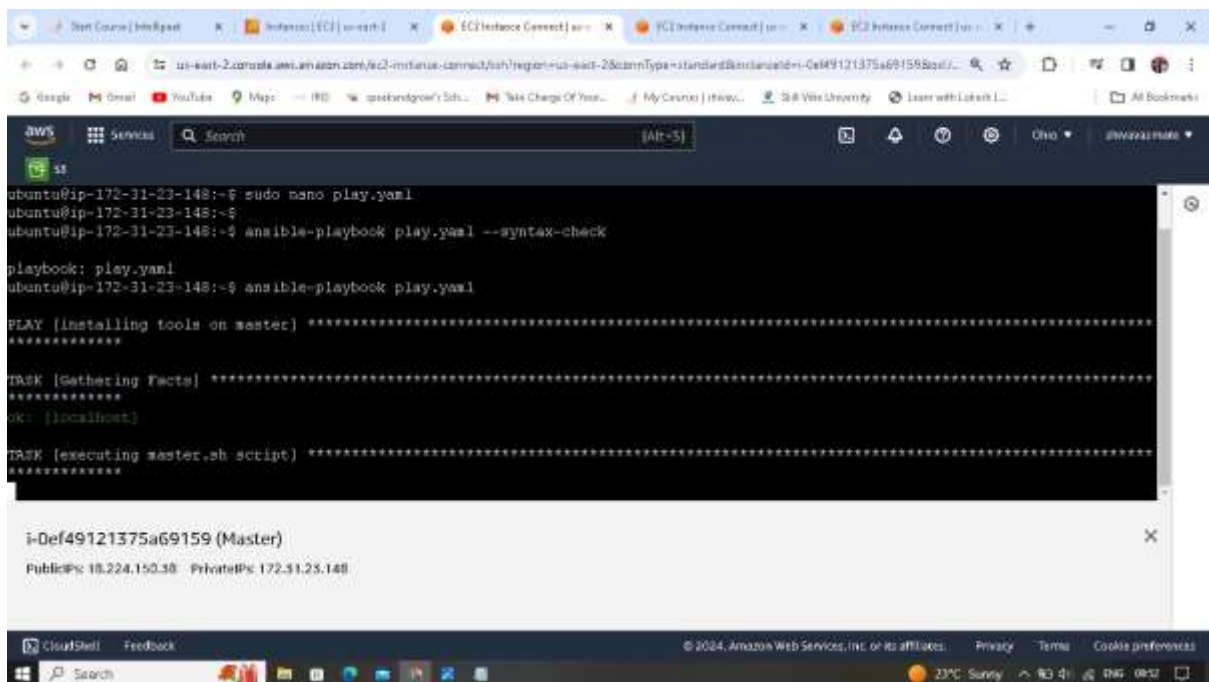
34. Check for any errors in the yaml file by `ansible-playbook play.yaml --syntax-check`



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



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



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output is as follows:

```
ubuntu@ip-172-31-23-148:~$ sudo nano play.yaml
ubuntu@ip-172-31-23-148:~$
ubuntu@ip-172-31-23-148:~$ ansible-playbook play.yaml --syntax-check

playbook: play.yaml
ubuntu@ip-172-31-23-148:~$ ansible-playbook play.yaml

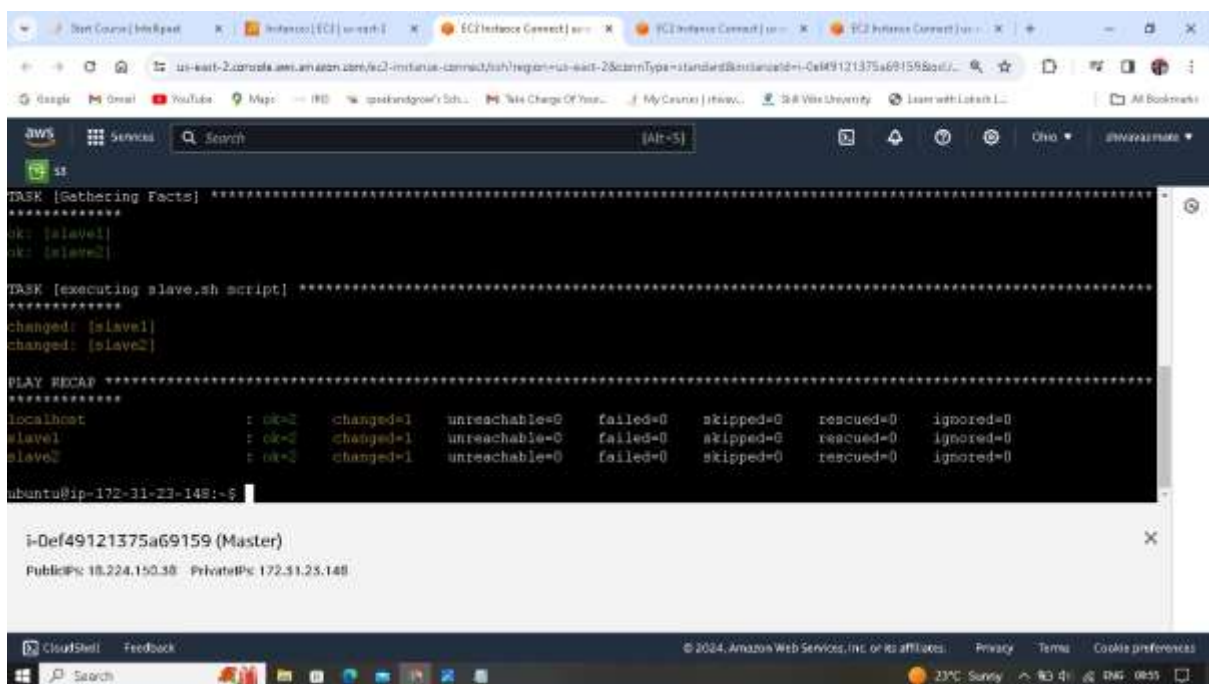
PLAY [installing tools on master] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [executing master.sh script] *****

i-Def49121375a69159 (Master)
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148
```

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



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output is as follows:

```
TASK [Gathering Facts] *****
ok: [slave1]
ok: [slave2]

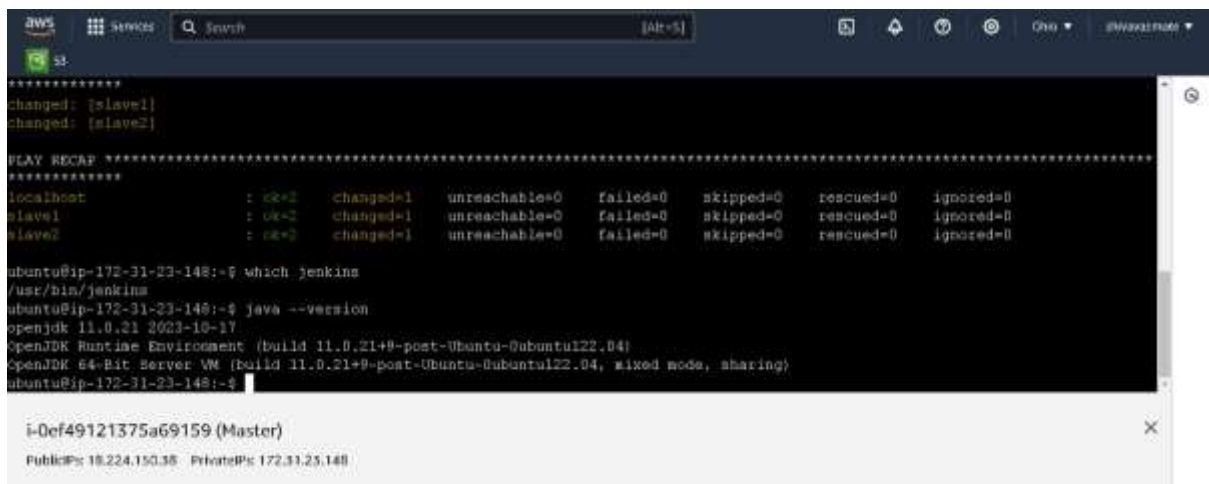
TASK [executing slave.sh script] *****
changed: [slave1]
changed: [slave2]

PLAY RECAP *****
localhost      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
slave1         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
slave2         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-23-148:~$

i-Def49121375a69159 (Master)
PublicIP: 18.224.150.38 PrivateIP: 172.31.23.148
```

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



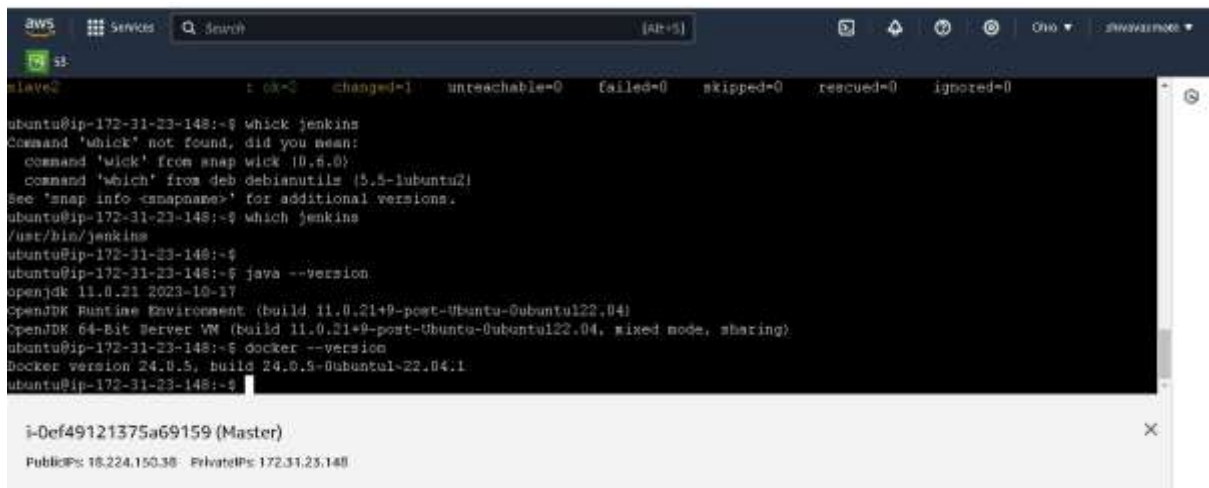
```
*****
changed: [slave1]
changed: [slave2]

PLAY RECAP *****
localhost                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
slave1                   : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
slave2                   : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-23-148:~$ which jenkins
/usr/bin/jenkins
ubuntu@ip-172-31-23-148:~$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-23-148:~$
```

i-0ef49121375a69159 (Master)  
PublicIPs: 18.224.150.38 PrivateIPs: 172.31.23.148

And for docker by docker --version

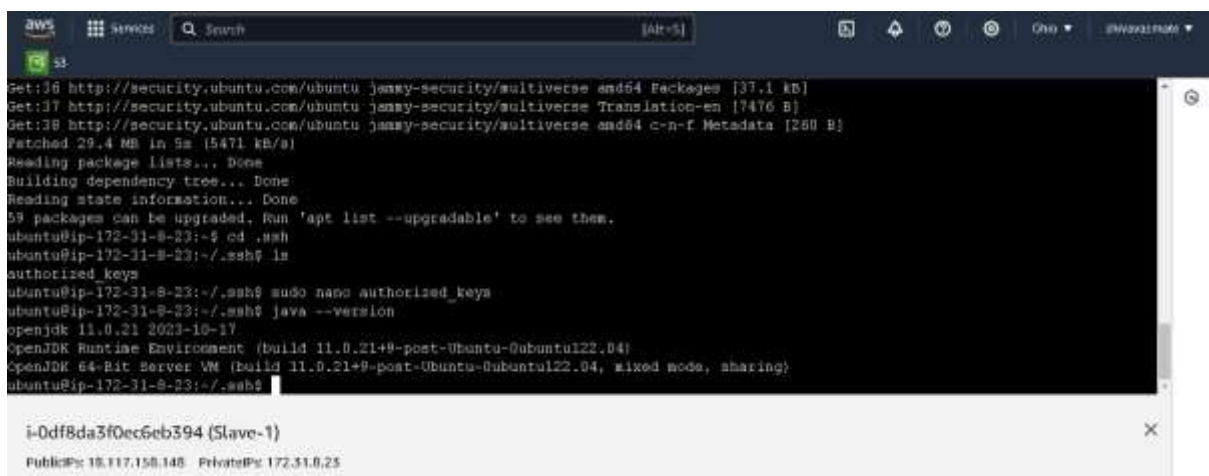


```
slave2                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-23-148:~$ which jenkins
Command 'which' not found, did you mean:
  Command 'wick' from snap wick (0.6.0)
  Command 'which' from deb debianutils (5.5-1ubuntu2)
See 'snap info <snapname>' for additional versions.
ubuntu@ip-172-31-23-148:~$ which jenkins
/usr/bin/jenkins
ubuntu@ip-172-31-23-148:~$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-23-148:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
ubuntu@ip-172-31-23-148:~$
```

i-0ef49121375a69159 (Master)  
PublicIPs: 18.224.150.38 PrivateIPs: 172.31.23.148

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



```
Get:16 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [280 B]
Fetched 29.4 MB in 9s (5471 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
59 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-8-23:~$ cd .ssh
ubuntu@ip-172-31-8-23:~/.ssh$ ls
authorized_keys
ubuntu@ip-172-31-8-23:~/.ssh$ sudo nano authorized_keys
ubuntu@ip-172-31-8-23:~/.ssh$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-8-23:~/.ssh$
```

i-0df8da3f0ec6eb394 (Slave-1)  
PublicIPs: 18.117.158.148 PrivateIPs: 172.31.8.23

And for docker by docker --version

```
Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

ubuntu@ip-172-31-8-23:~/.ssh$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1-22.04.1
ubuntu@ip-172-31-8-23:~/.ssh$
```

i-0df8da3f0ec6eb394 (Slave-1)  
PublicIP: 18.117.158.148 PrivateIP: 172.31.8.23

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

```
ubuntu@ip-172-31-5-15:~/.ssh$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-5-15:~/.ssh$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1-22.04.1
ubuntu@ip-172-31-5-15:~/.ssh$
```

i-0f55020c16517ab2c (Slave-2)  
PublicIP: 18.58.201.302 PrivateIP: 172.31.5.15

41. Go to master machine copy the public IP

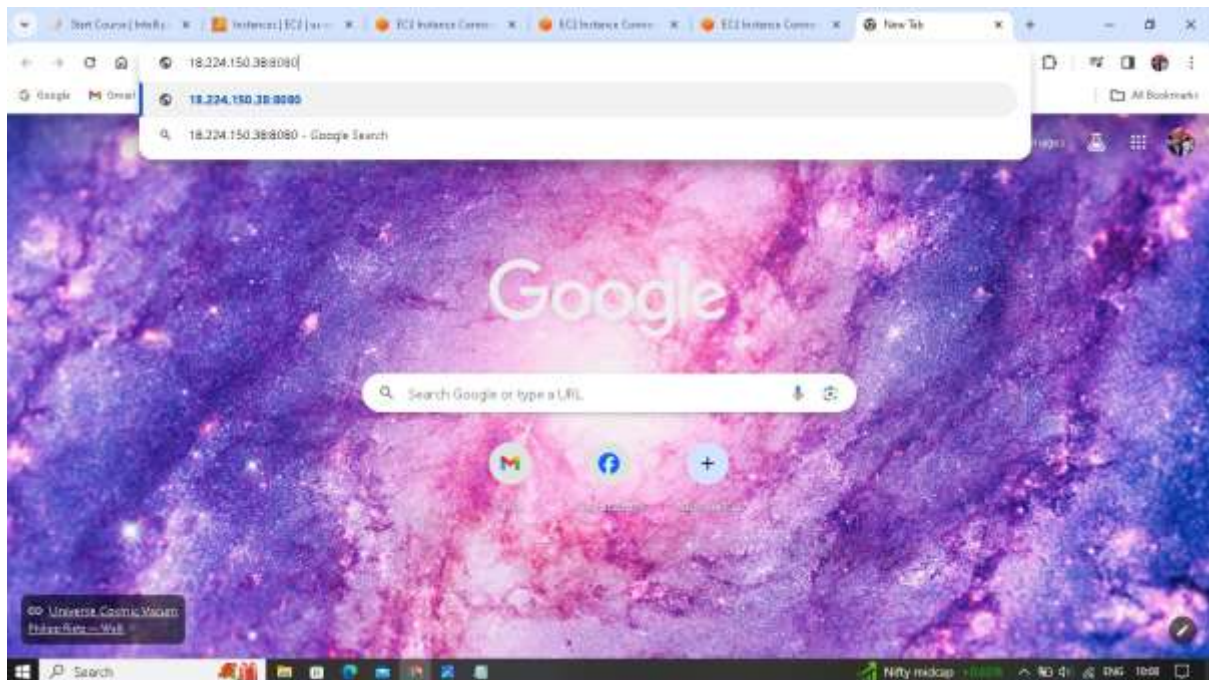
```
Slave2: 0 ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

ubuntu@ip-172-31-23-148:~$ which jenkins
Command 'which' not found, did you mean:
  command 'wick' from snap wick (0.6.0)
  command 'which' from deb debianutils (5.5-1ubuntu2)
See 'snap info <snapname>' for additional versions.
ubuntu@ip-172-31-23-148:~$ which jenkins
/usr/bin/jenkins
ubuntu@ip-172-31-23-148:~$
ubuntu@ip-172-31-23-148:~$ java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-23-148:~$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1-22.04.1
ubuntu@ip-172-31-23-148:~$
```

i-Def49121375a69159 (Master)  
PublicIP: 18.224.168.38 PrivateIP: 172.31.23.148



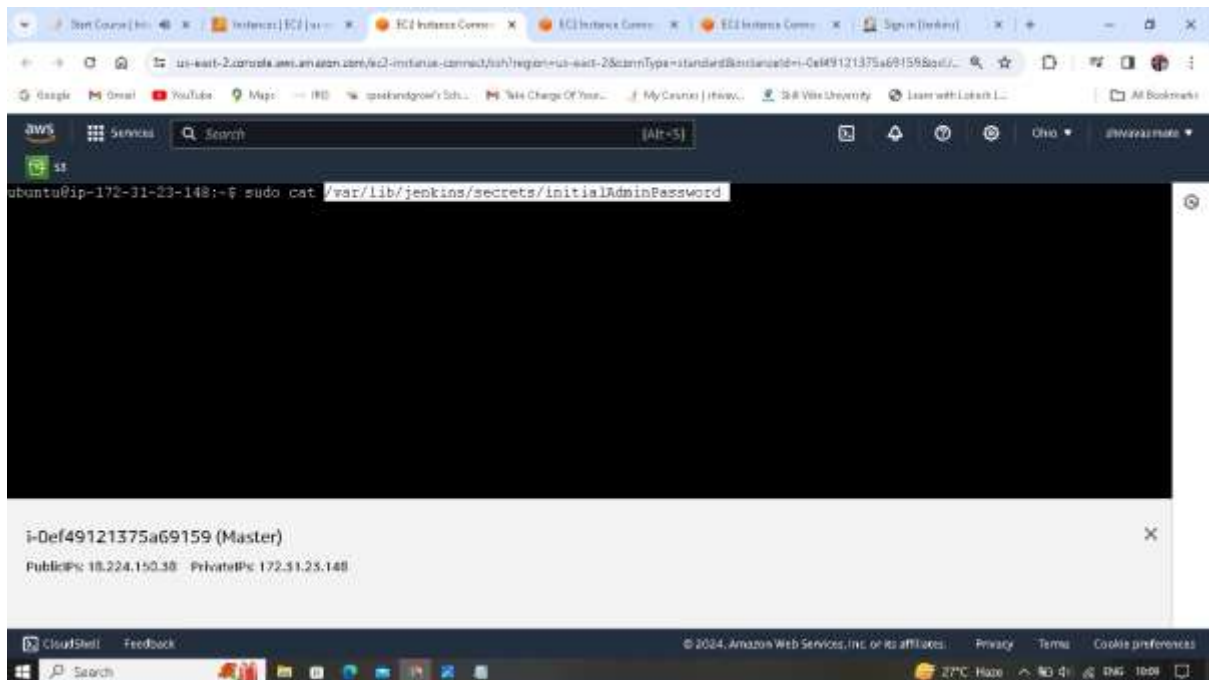
42. Paste it with port number:8080 in another browser.



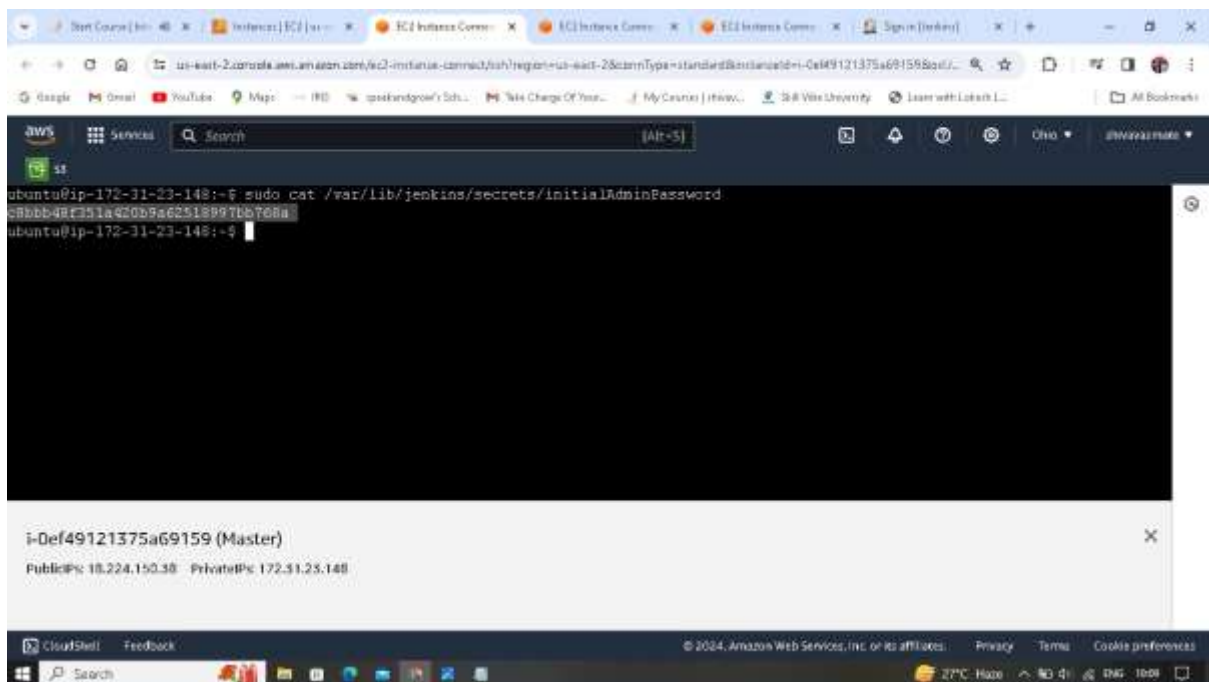
43. Copy the /var/lib/jenkins/secrets/initialadminPassword.



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



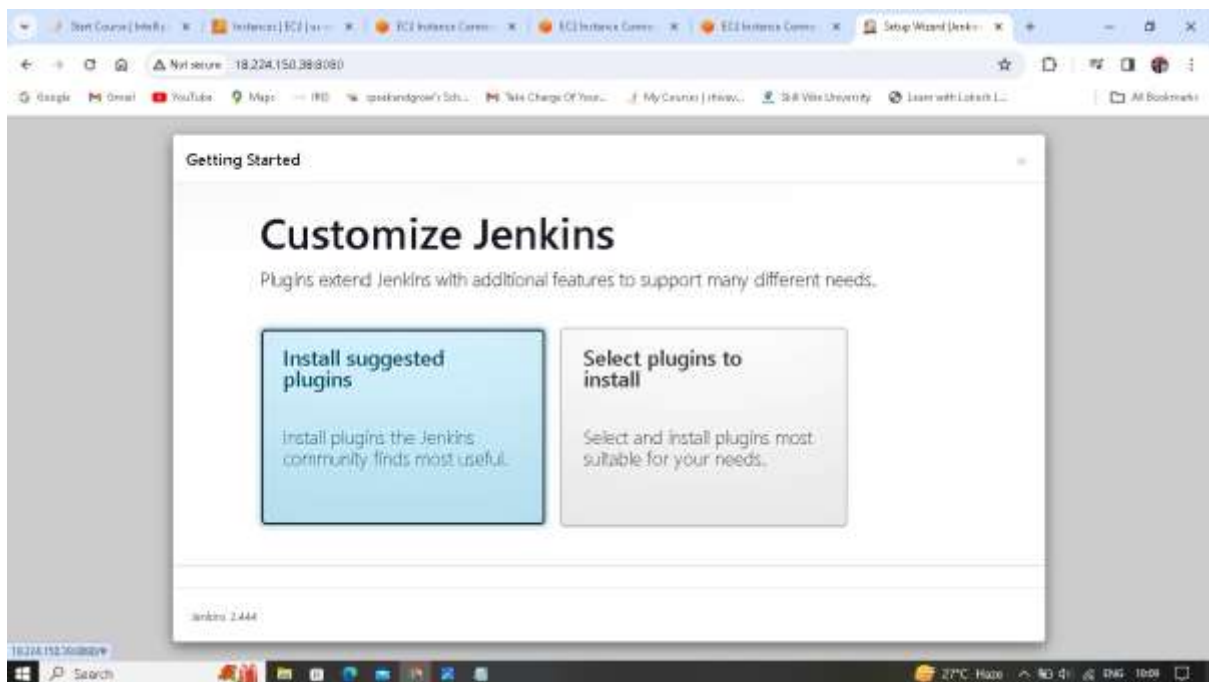
45. Copy the password



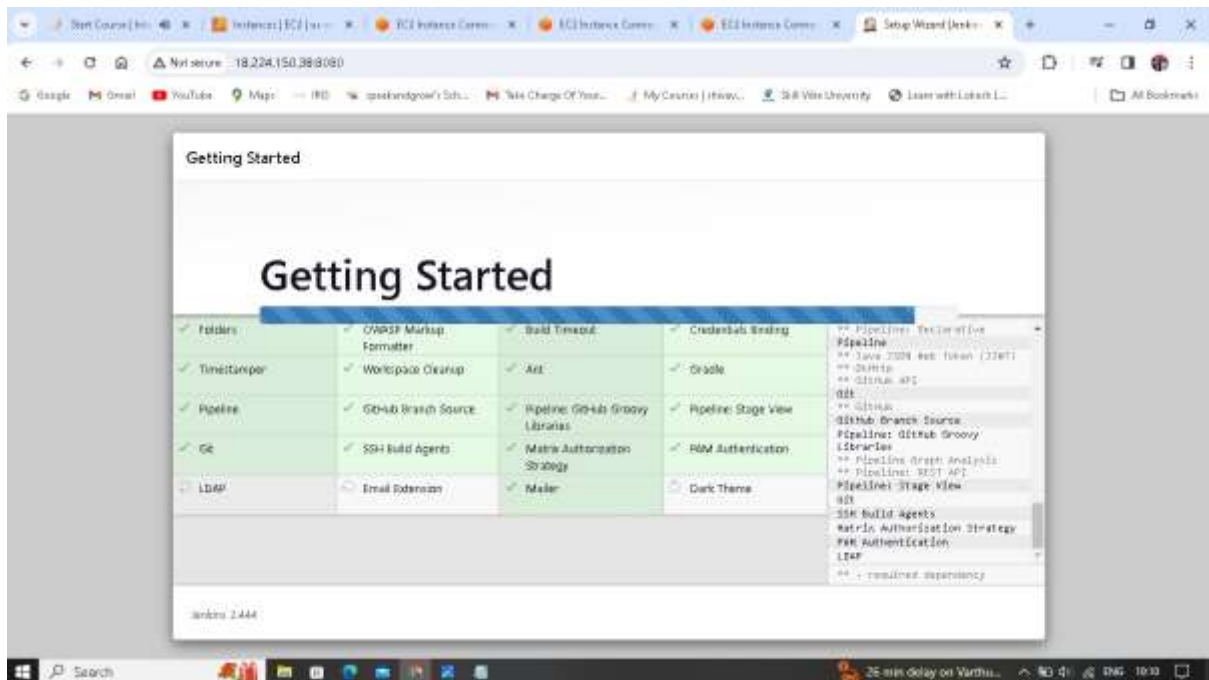
46. Paste it and click on continue.



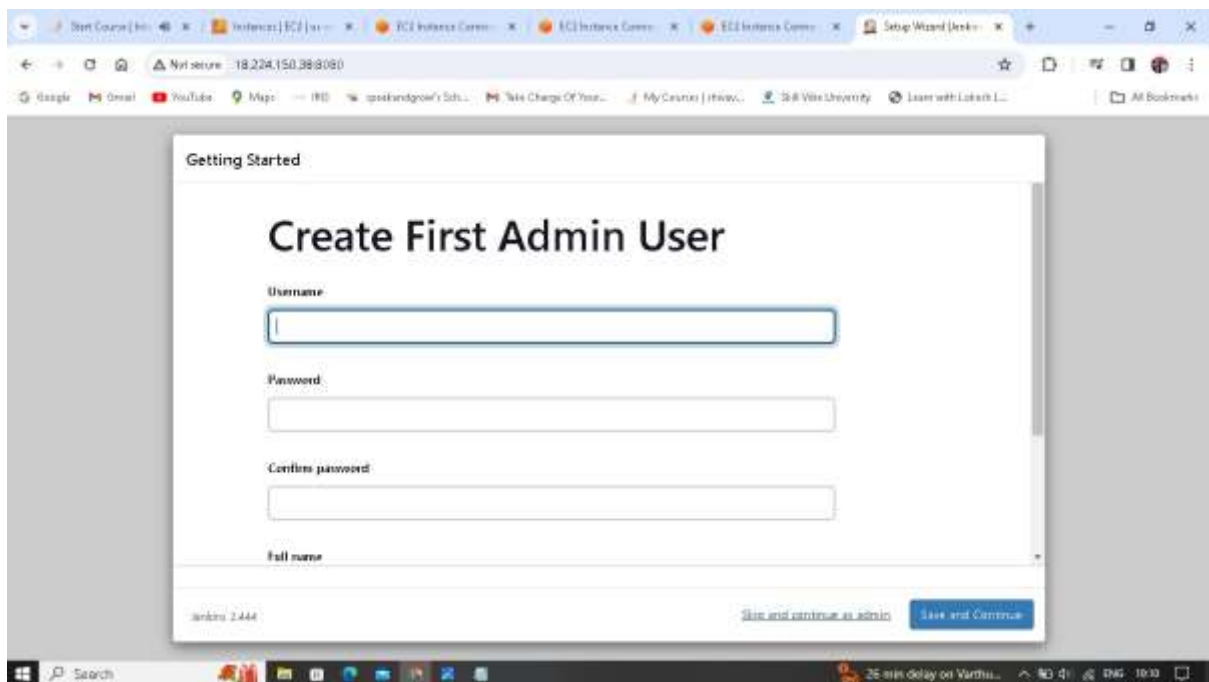
47. Click on Install suggested plugins



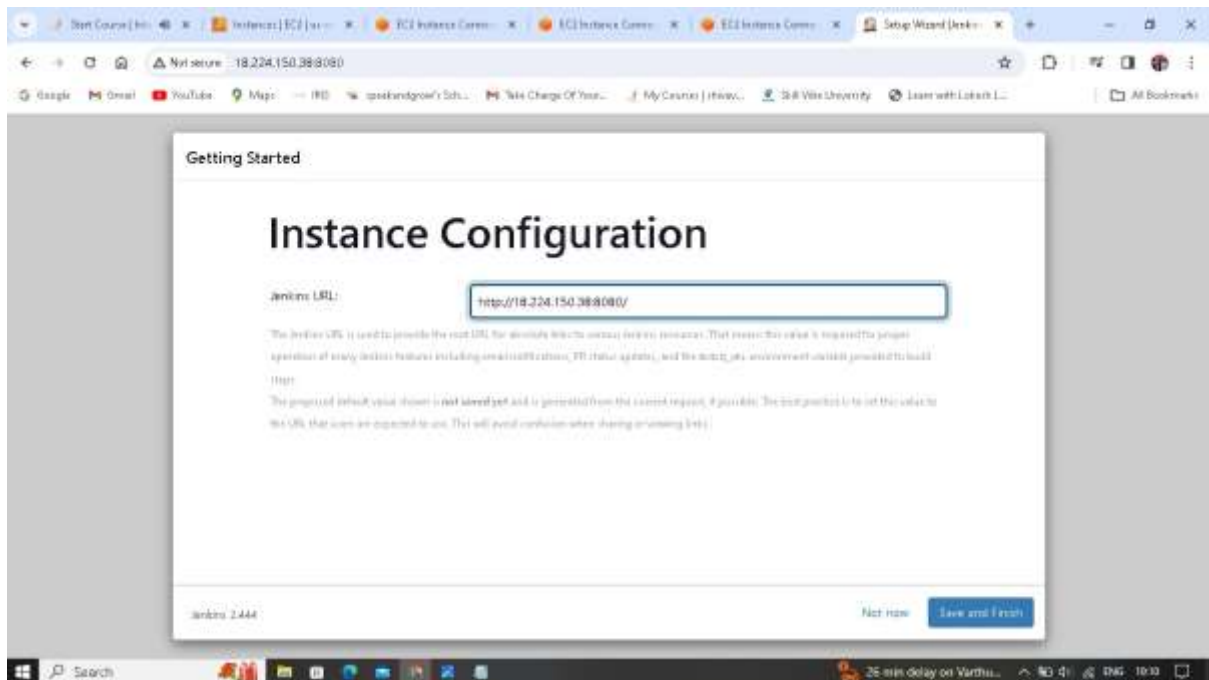
48. Let it install all plugins



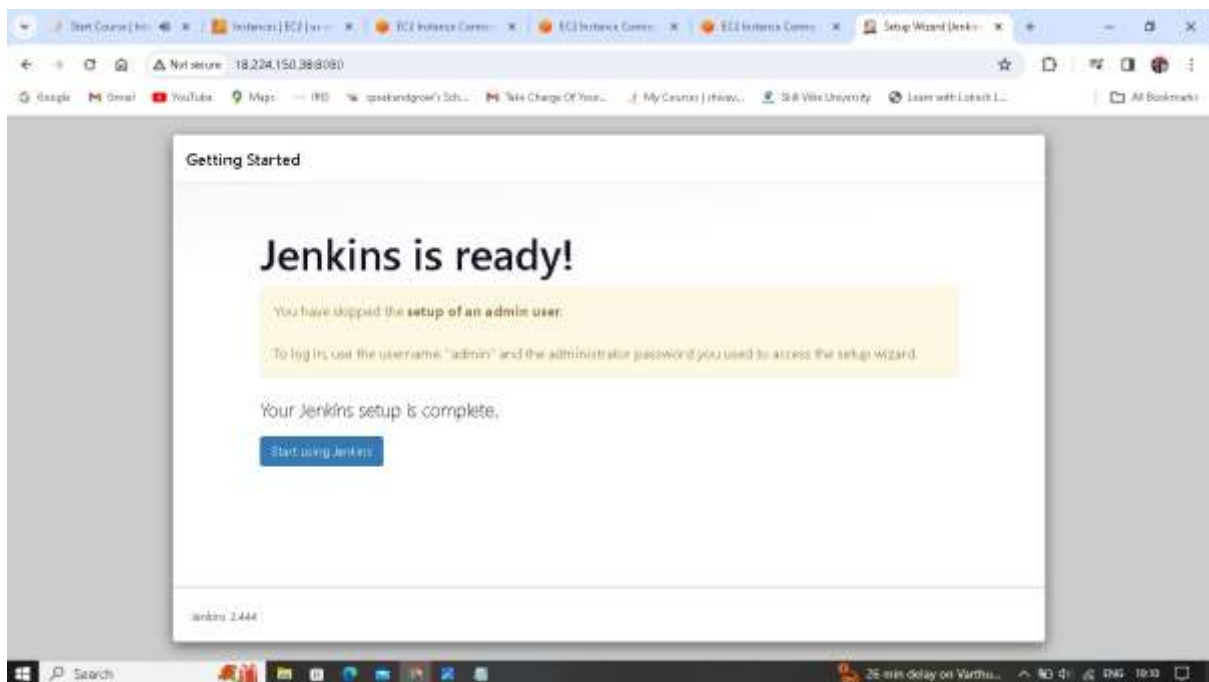
49. Click on skip and continue as admin.



50. Click on save and finish

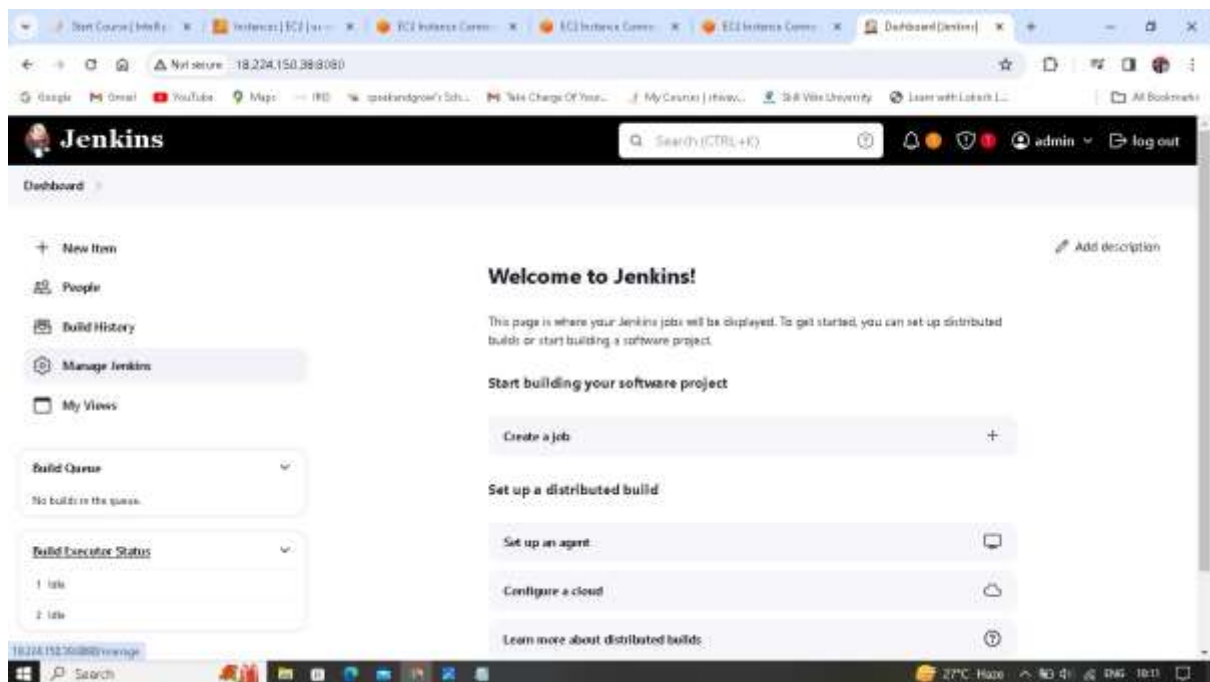


51 Jenkins is ready. Click on Start using Jenkins

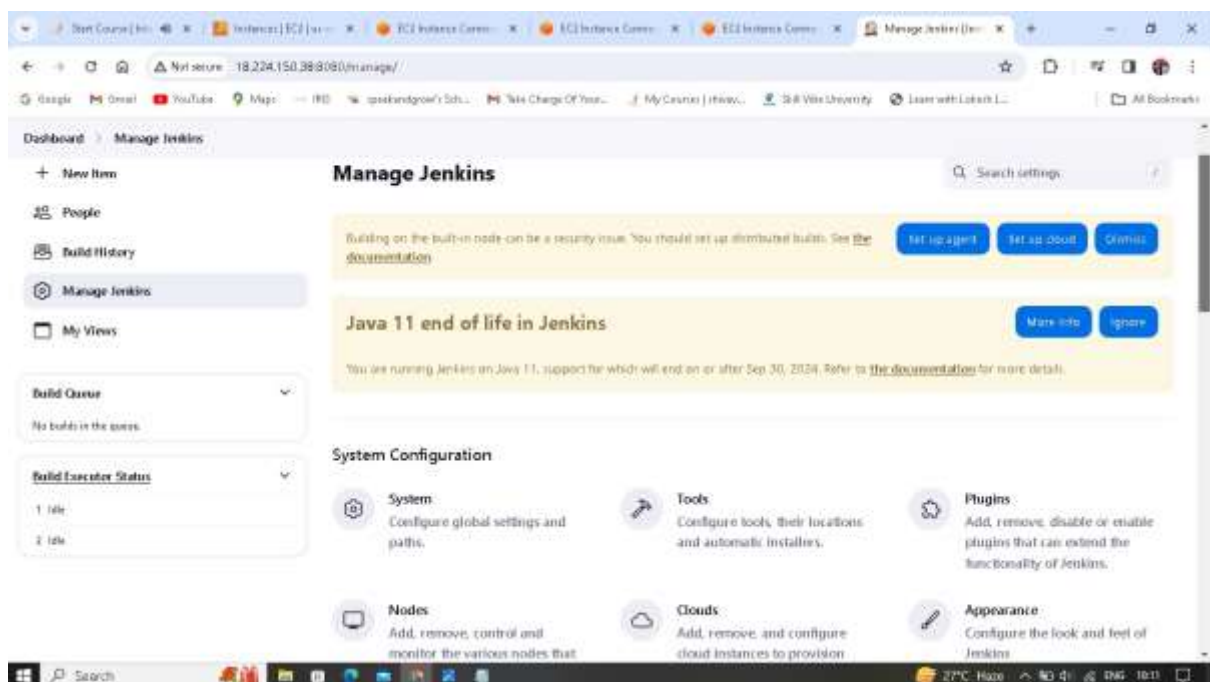




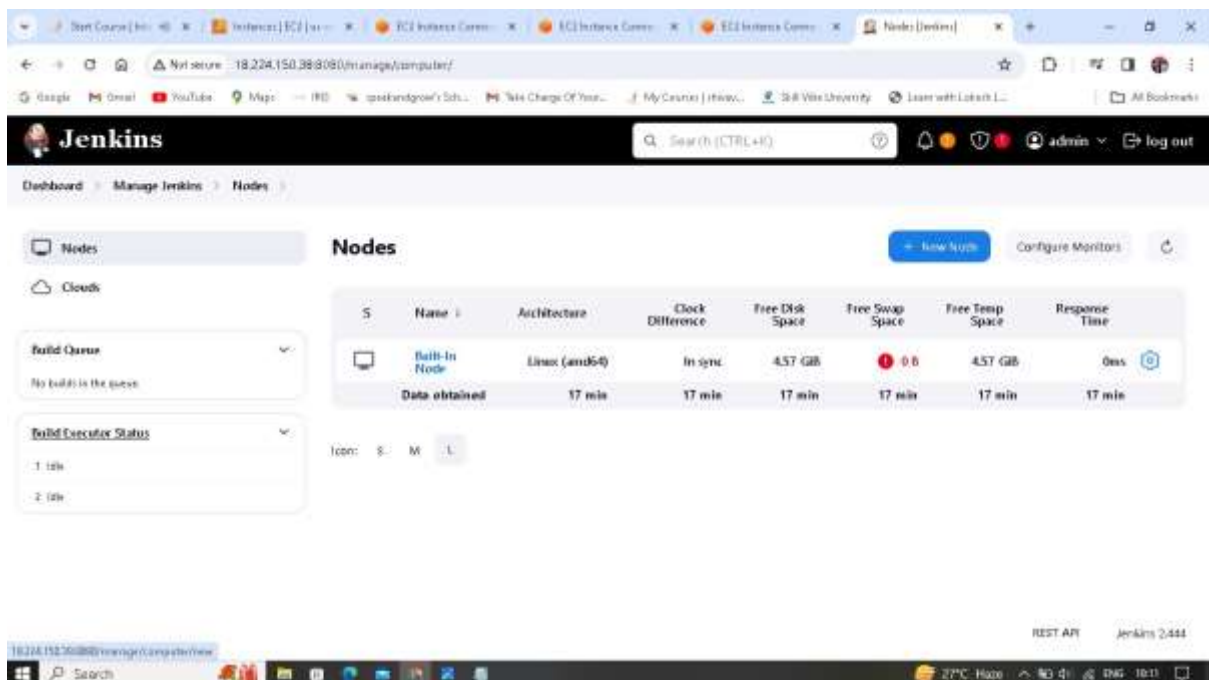
## 52. Click on Manage Jenkins



## 53. Click on Nodes



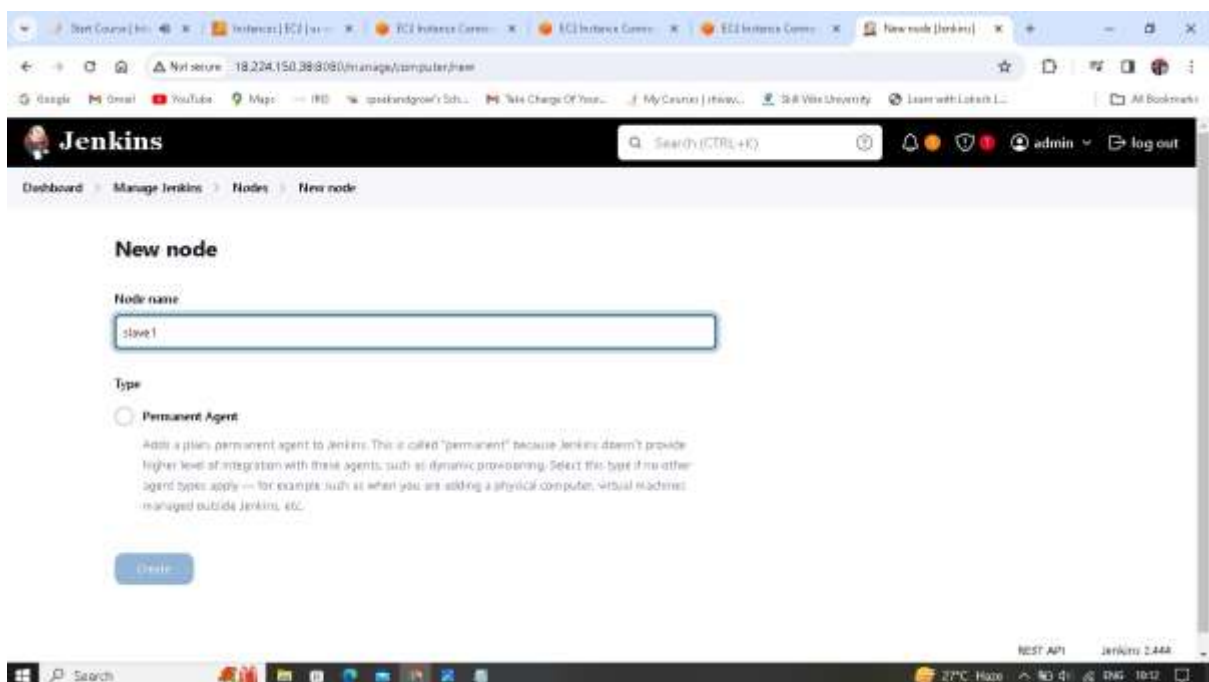
## 54. Click on +New Node



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and a user profile dropdown. The main content area is titled 'Nodes' and features a table of existing nodes. A 'New Node' button is visible in the top right corner of the nodes section.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.57 GB	0.6	4.57 GB	0ms
	Data obtained	17 min	17 min	17 min	17 min	17 min	17 min

## 55. Give Node name as slave1



The screenshot shows the 'New node' form in the Jenkins web interface. The 'Node name' field is filled with 'slave1'. The 'Type' section has 'Permanent Agent' selected. A 'Create' button is at the bottom of the form.

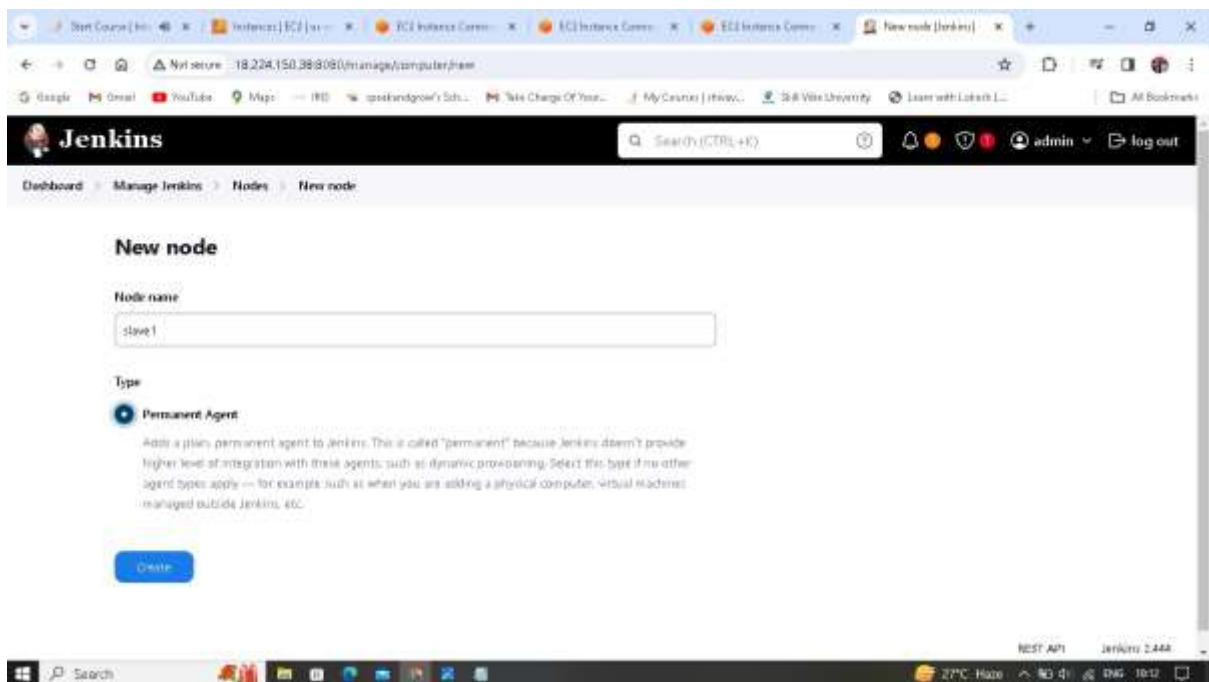
**New node**

Node name  
slave1

Type  
☒ Permanent Agent

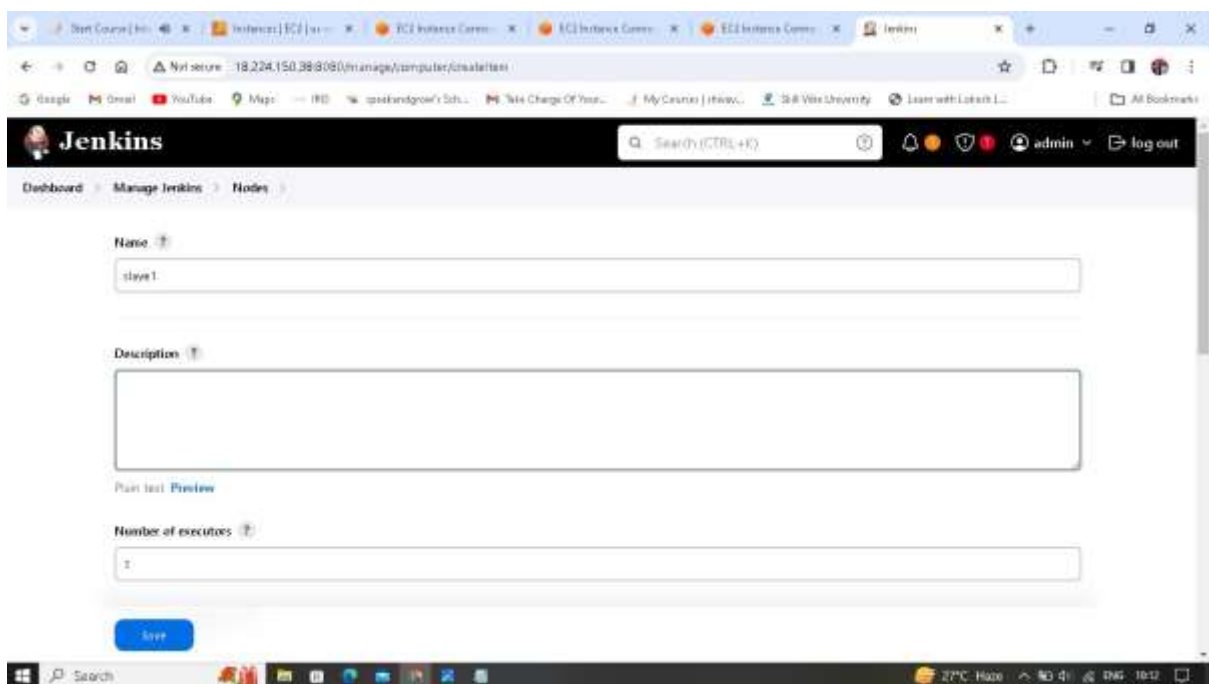
Create

56. Check in Permanent Agent and click on create



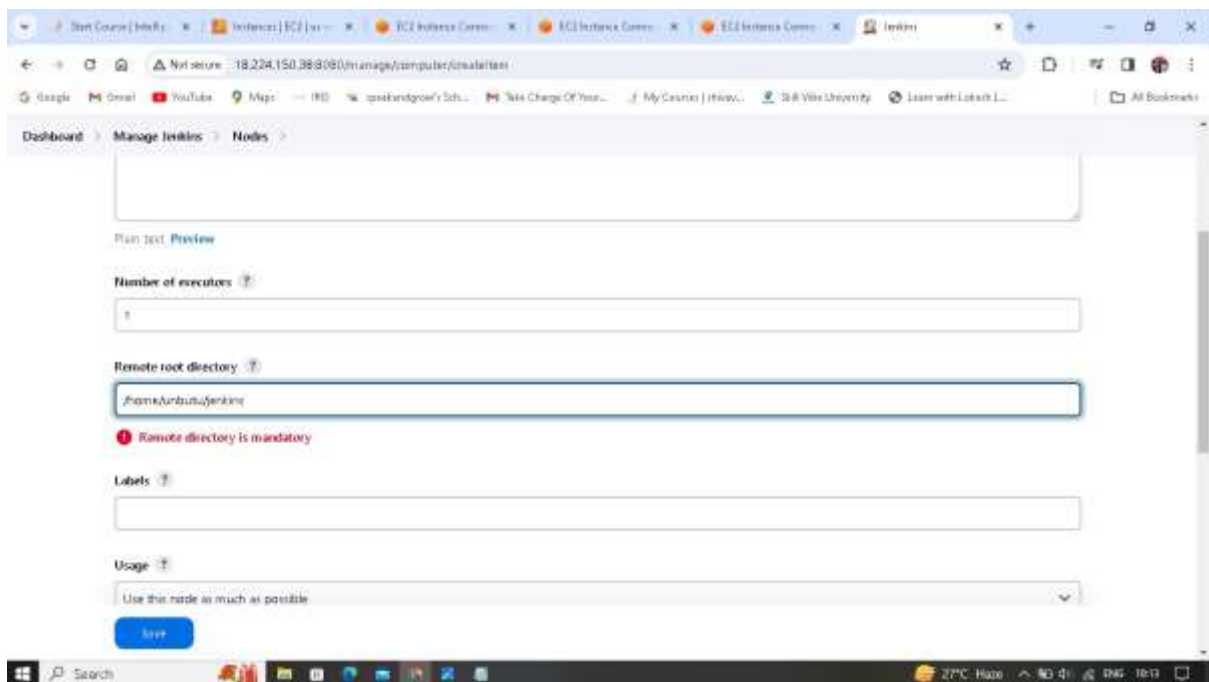
The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `http://18.224.150.38:8080/manage/computer/new`. The page title is "New node". Under the "Node name" field, the text "slave1" is entered. Under the "Type" section, the "Permanent Agent" radio button is selected. Below this, a descriptive paragraph explains that a permanent agent is a Java process that runs on the slave machine and provides a higher level of integration with Jenkins. At the bottom of the form, there is a blue "Create" button. The Jenkins logo and navigation menu are visible at the top.

57. Fill one by one



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `http://18.224.150.38:8080/manage/computer/new`. The page title is "New node". Under the "Name" field, the text "slave1" is entered. Under the "Description" field, there is a large empty text area. Below the description field, there is a "Number of executors" field with the value "1" entered. At the bottom of the form, there is a blue "Save" button. The Jenkins logo and navigation menu are visible at the top.

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



The screenshot shows the Jenkins 'Nodes' configuration page. The 'Remote root directory' field is set to '/home/ubuntu/jenkins'. A red error message below it states 'Remote directory is mandatory'. The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Save' button is visible at the bottom.

Dashboard > Manage Jenkins > Nodes >

Plan Text [Preview](#)

Number of executors

Remote root directory

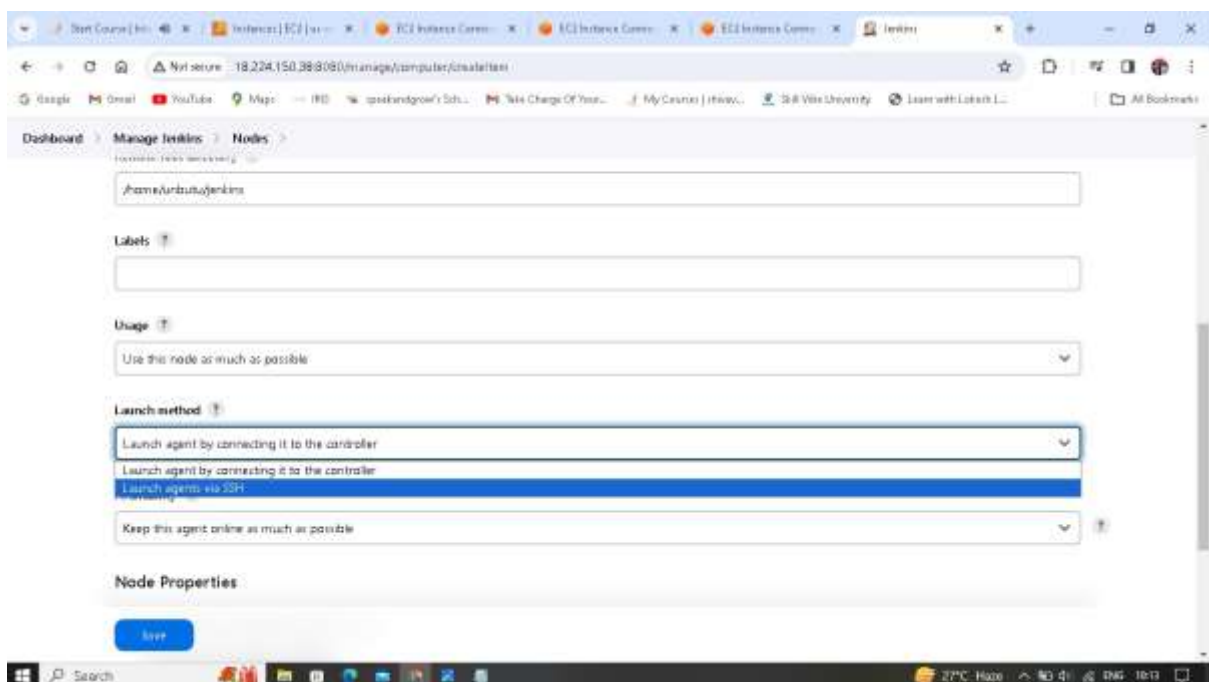
**Remote directory is mandatory**

Labels

Usage

[Save](#)

59. In Launch method choose Launch agents via SSH



The screenshot shows the Jenkins 'Nodes' configuration page with the 'Launch method' dropdown menu open. The option 'Launch agents via SSH' is selected. The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Save' button is visible at the bottom.

Dashboard > Manage Jenkins > Nodes >

Labels

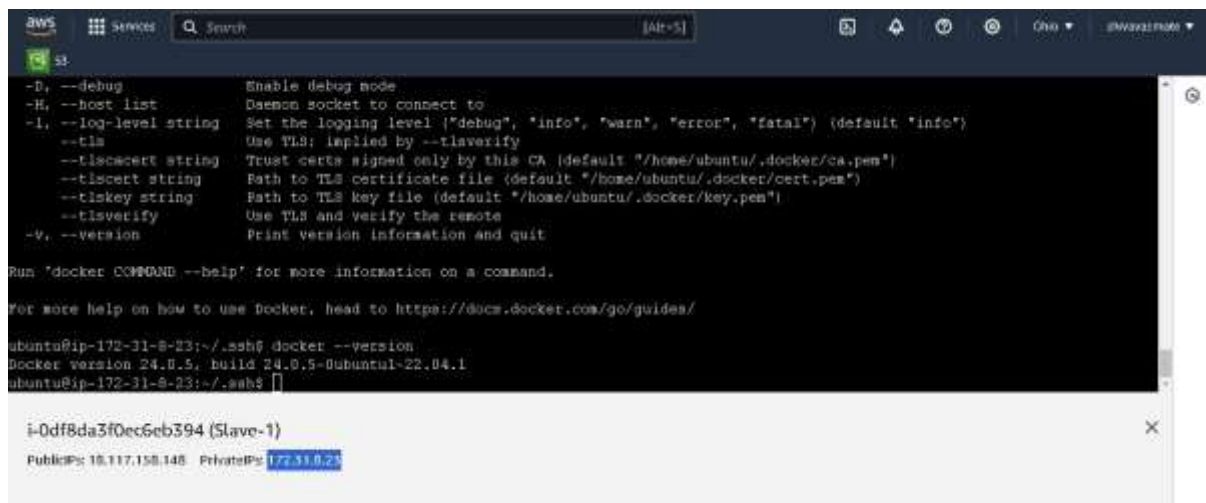
Usage

Launch method

Node Properties

[Save](#)

60. Go to Slave1 copy the private IP



The screenshot shows an AWS console interface. The main terminal window displays the Docker help text and the output of the command `docker --version`, which is `Docker version 24.0.5, build 24.0.5-0ubuntu1-22.04.1`. Below the terminal, a console output window for the instance `i-0df8da3f0ec5eb394 (Slave-1)` shows the public IP as `15.117.158.148` and the private IP as `172.31.0.23`, which is highlighted in blue.

```
-D, --debug          Enable debug mode
-H, --host list       Daemon socket to connect to
-l, --log-level string Set the logging level ("debug", "info", "warn", "error", "fatal") (default "info")
--tls                Use TLS; implied by --tlsv1verify
--tlscacert string    Trust certs signed only by this CA (default "/home/ubuntu/.docker/ca.pem")
--tlscert string      Path to TLS certificate file (default "/home/ubuntu/.docker/cert.pem")
--tlskey string        Path to TLS key file (default "/home/ubuntu/.docker/key.pem")
--tlsv1verify         Use TLS and verify the remote
-V, --version          Print version information and quit

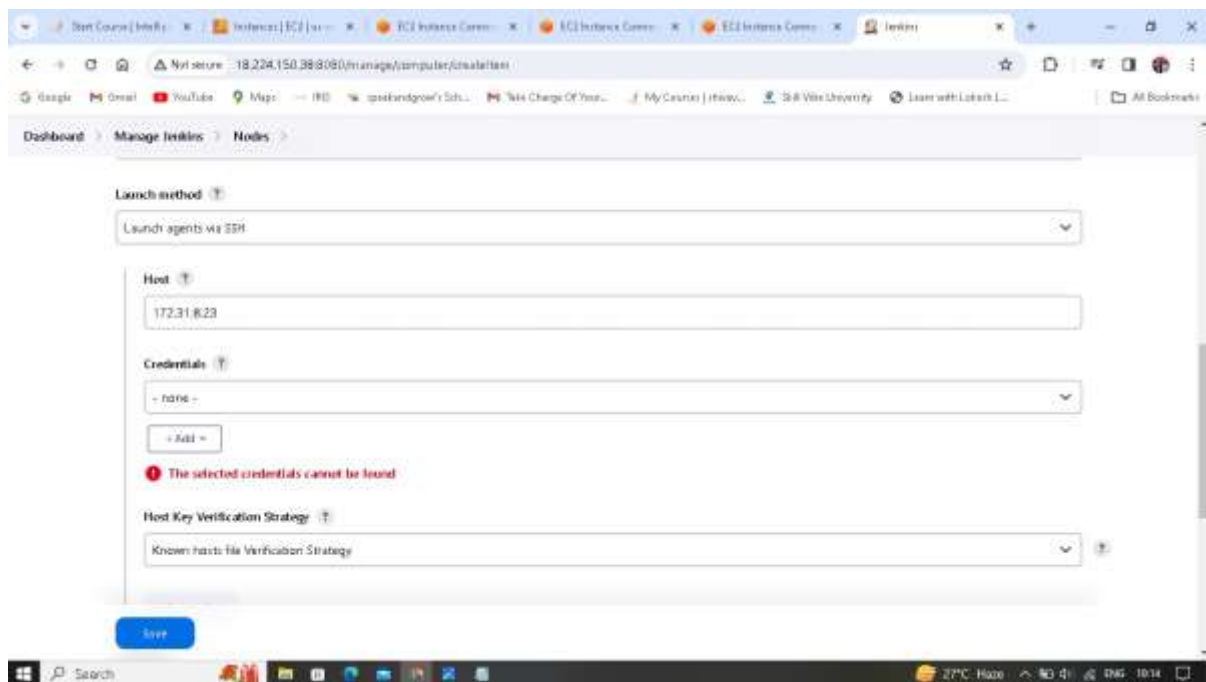
Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/go/guides/

ubuntu@ip-172-31-8-23:~/.ssh$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1-22.04.1
ubuntu@ip-172-31-8-23:~/.ssh$
```

i-0df8da3f0ec5eb394 (Slave-1)  
PublicIP: 15.117.158.148 PrivateIP: 172.31.0.23

61. Under Host paste it.



The screenshot shows the Jenkins 'Nodes' configuration page. The 'Launch method' is set to 'Launch agents via SSH'. The 'Host' field is filled with the private IP address '172.31.0.23'. The 'Credentials' dropdown is set to 'None', and there is a red error message below it: 'The selected credentials cannot be found'. The 'Host Key Verification Strategy' is set to 'Known hosts file Verification Strategy'. A 'Save' button is at the bottom.

Launch method: Launch agents via SSH

Host: 172.31.0.23

Credentials: None

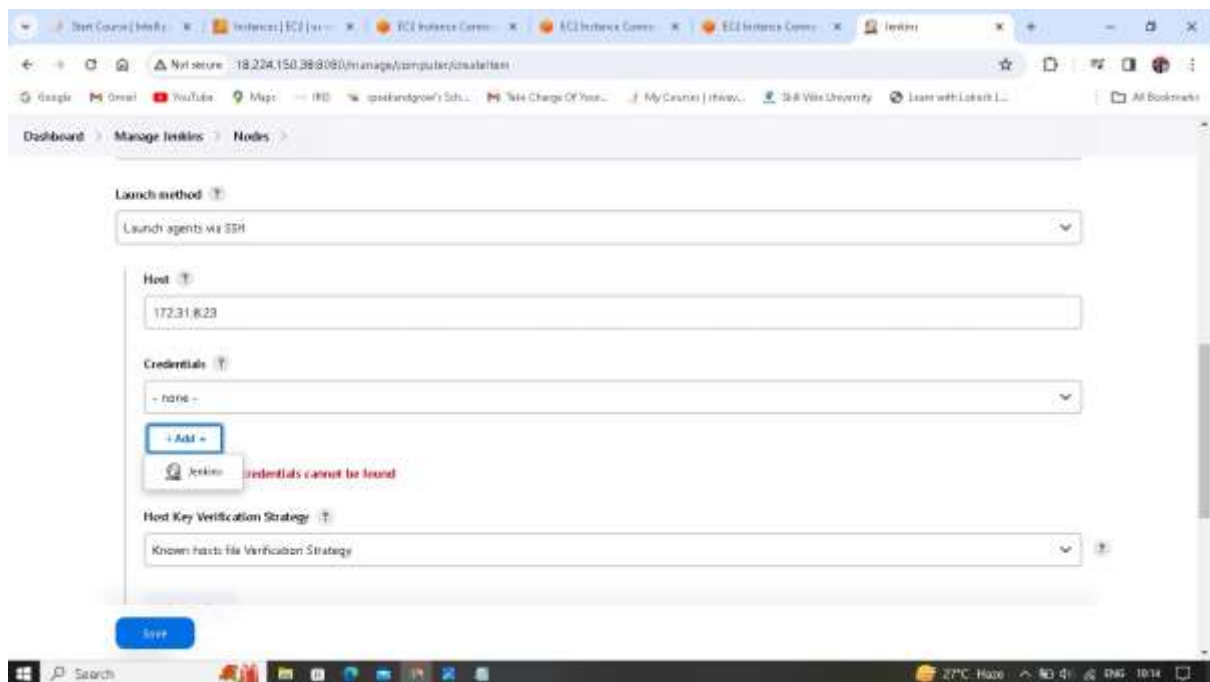
The selected credentials cannot be found

Host Key Verification Strategy: Known hosts file Verification Strategy

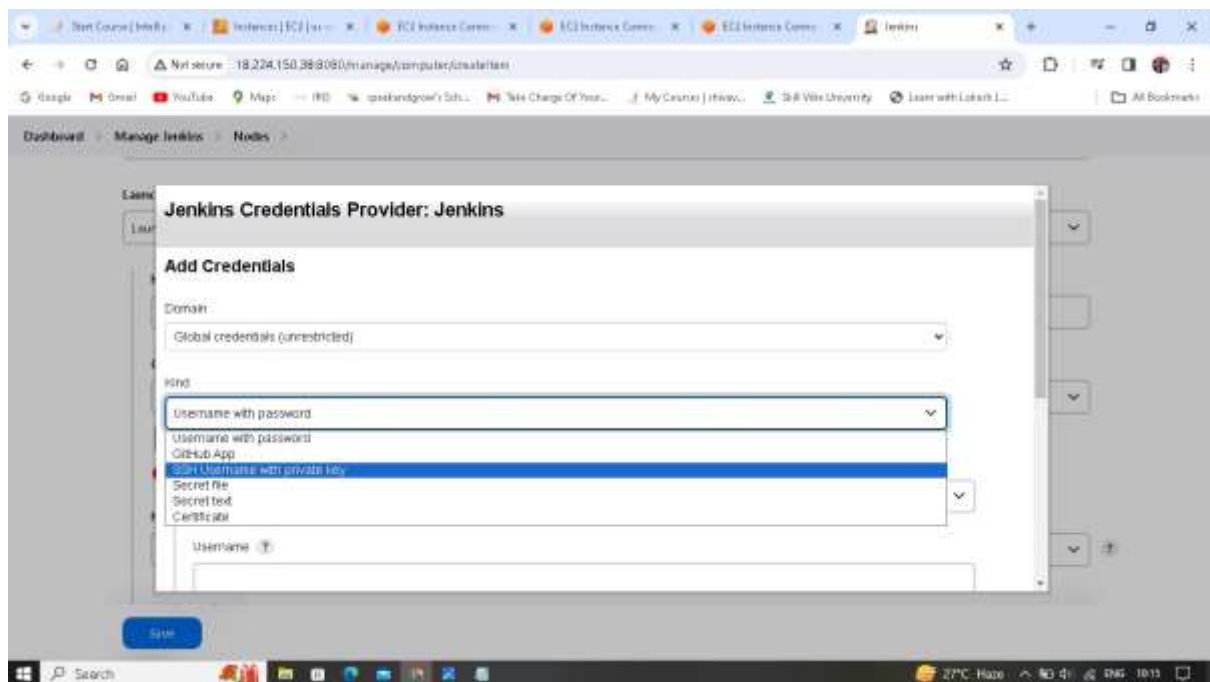
Save



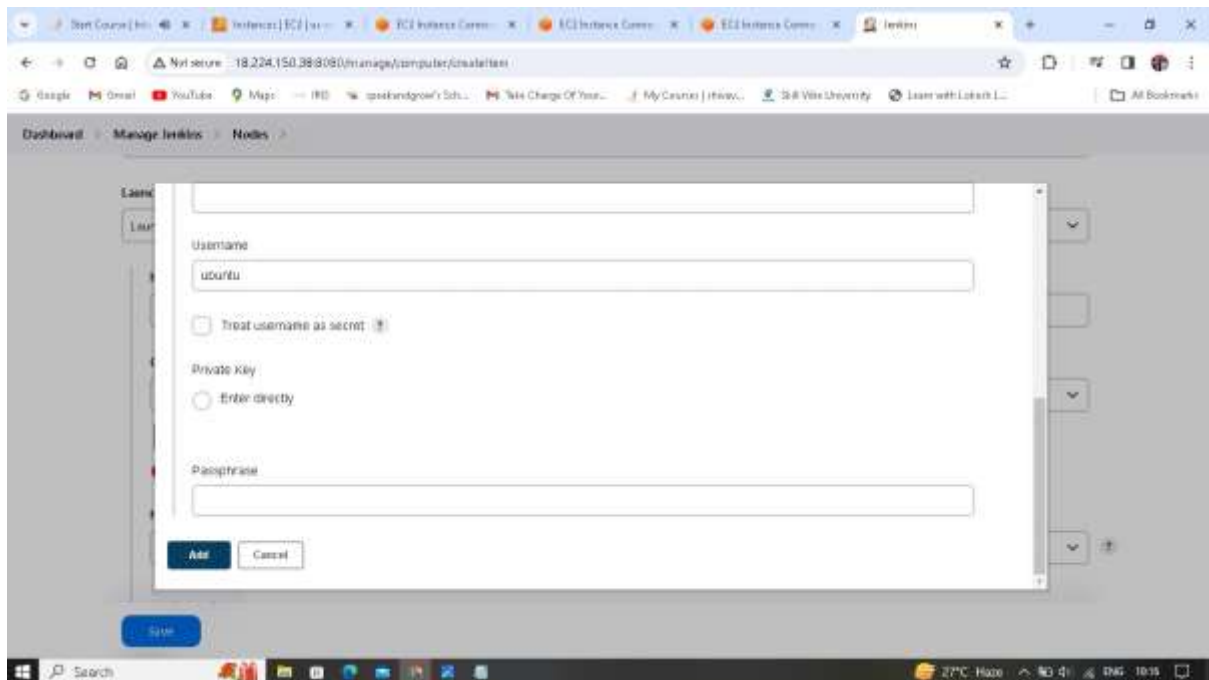
62. Under Credentials click on Add



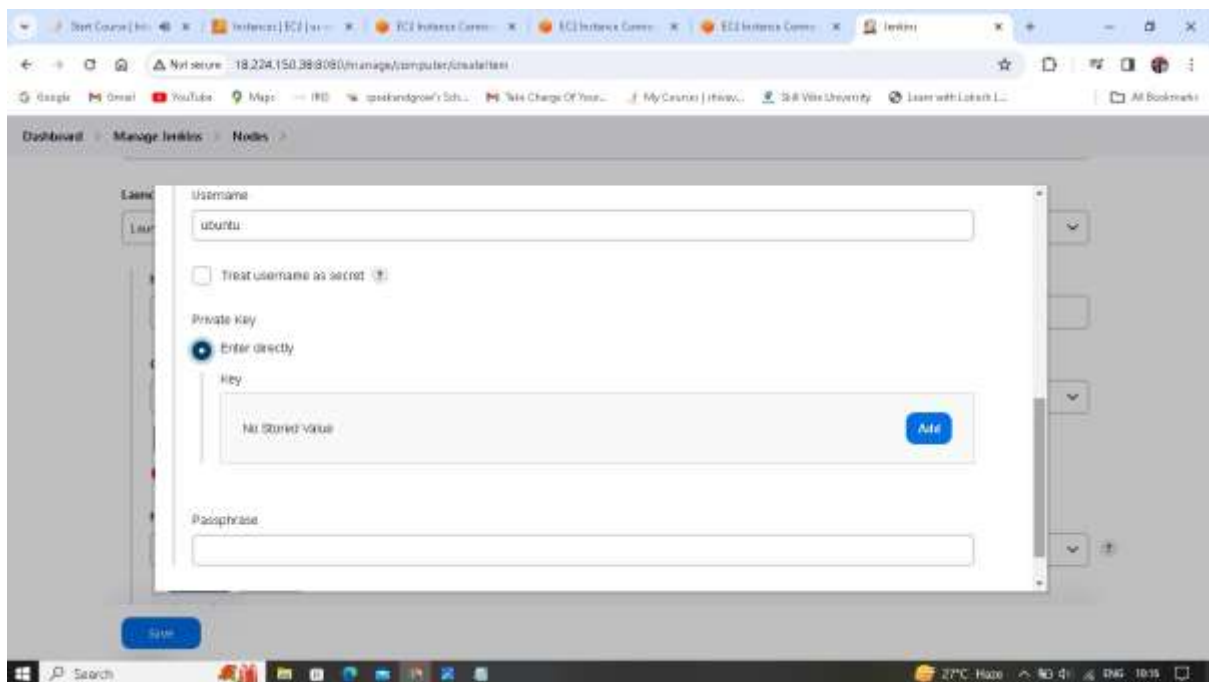
63. Under kind choose SSH Username with private key



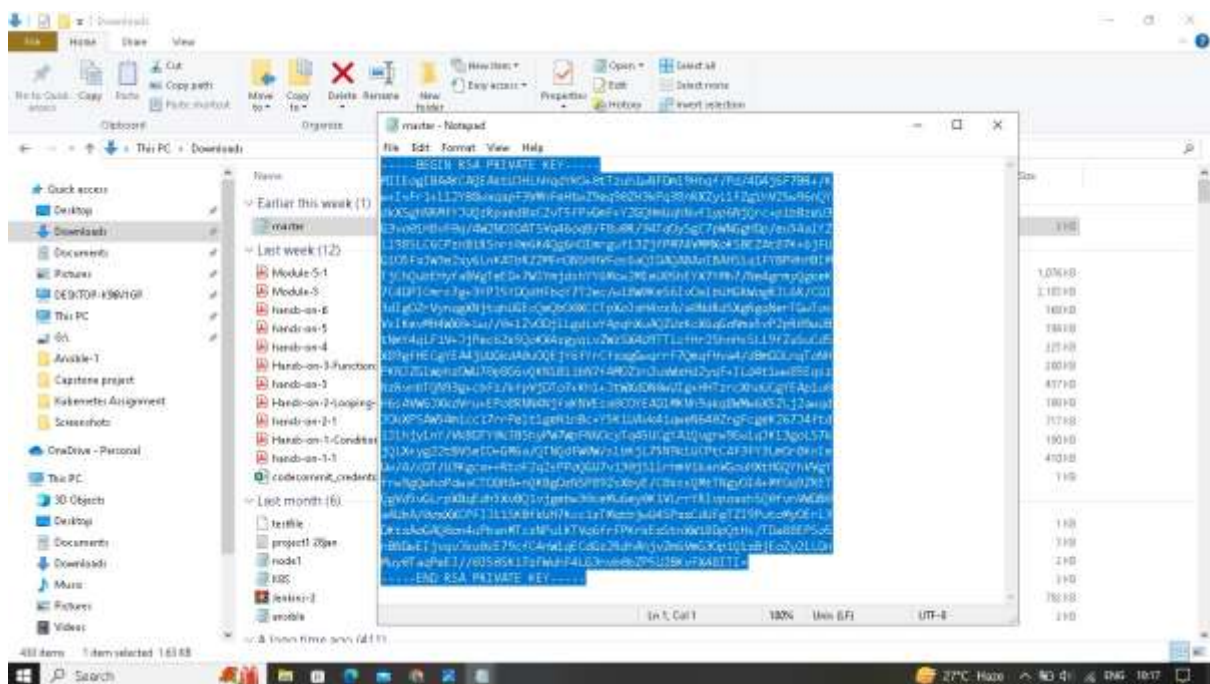
64. Under Username give as ubuntu



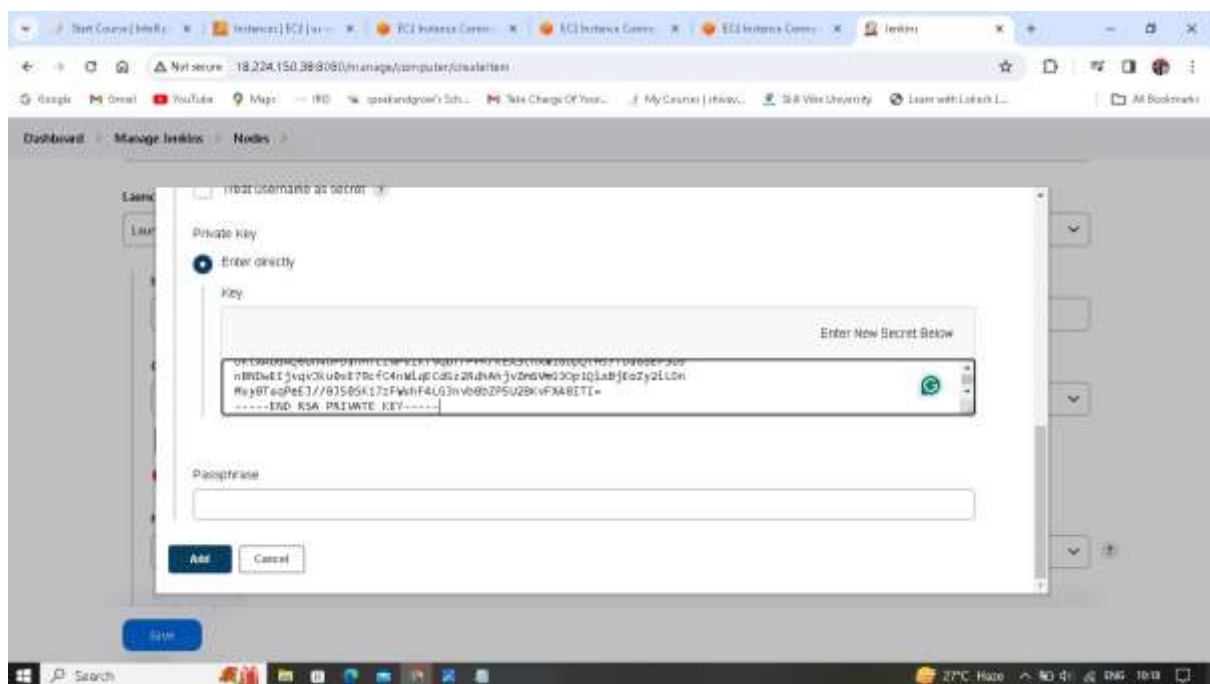
65. Check in Enter directly and click on Add.



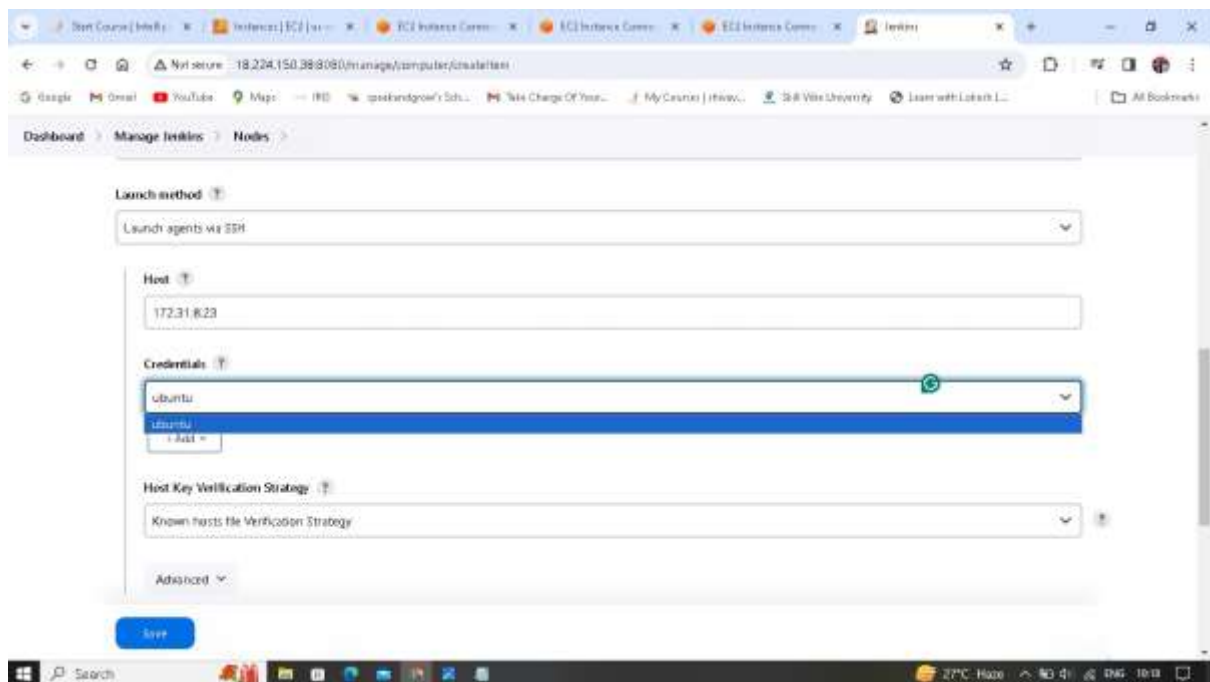
## 66. Copy the pem file content



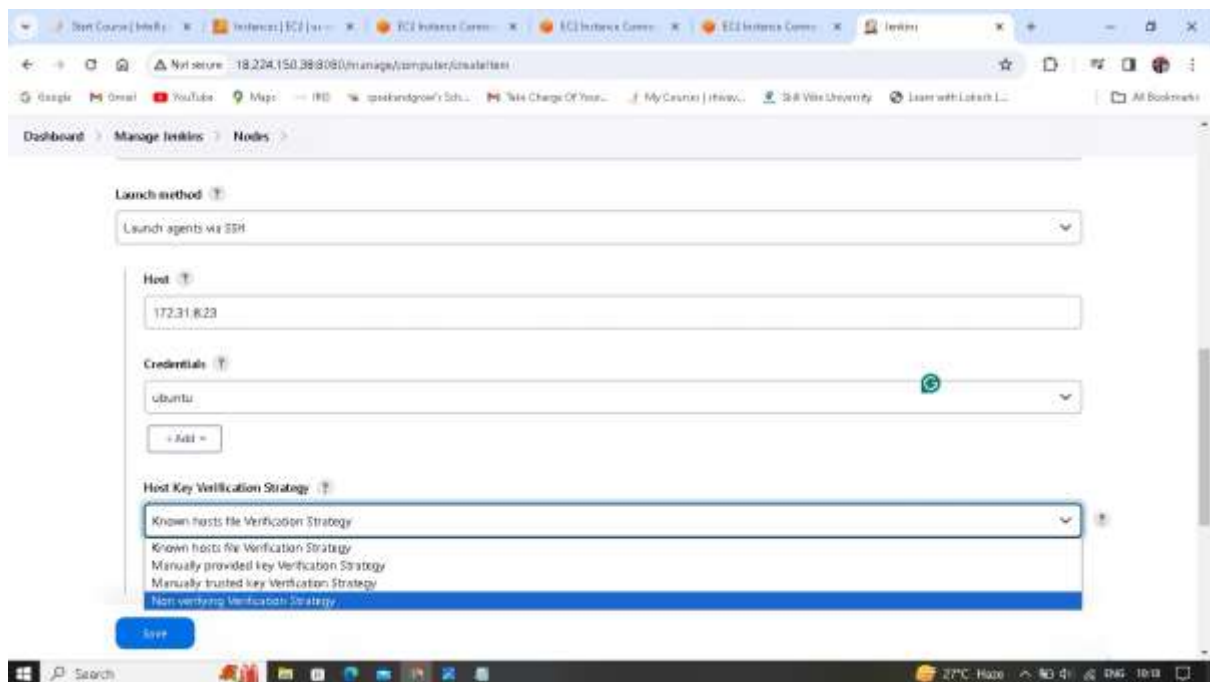
## 67. Paste it and click on Add.



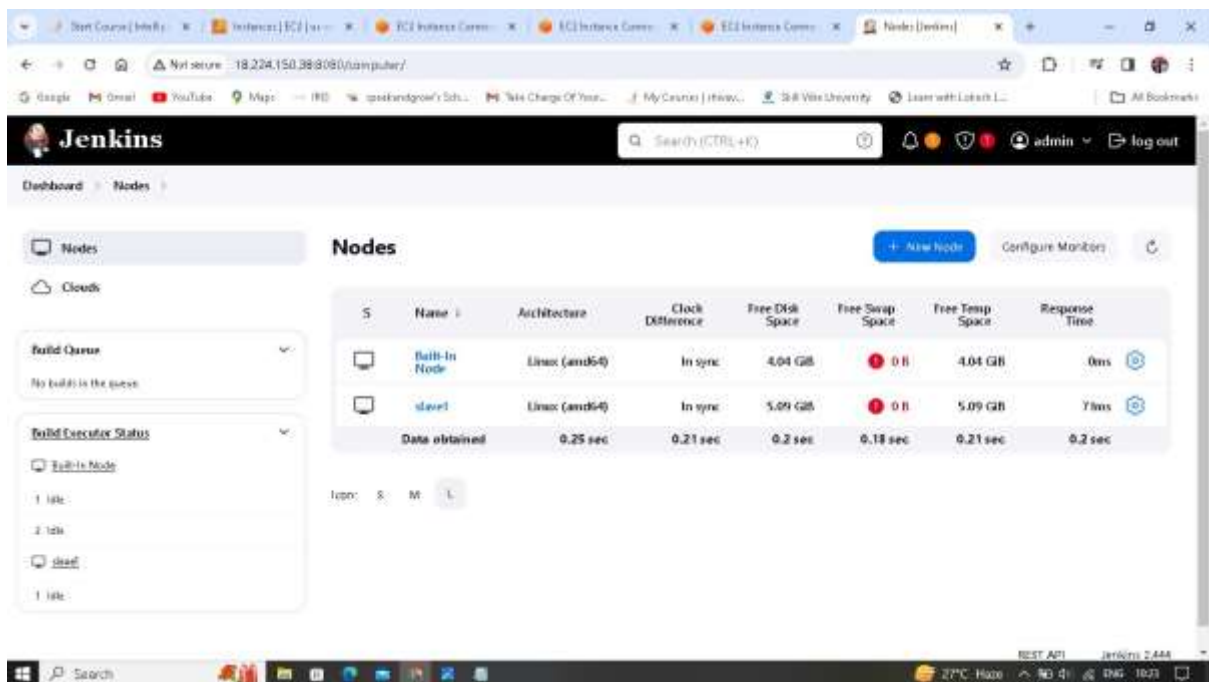
68. Now choose the Credentials as ubuntu



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



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

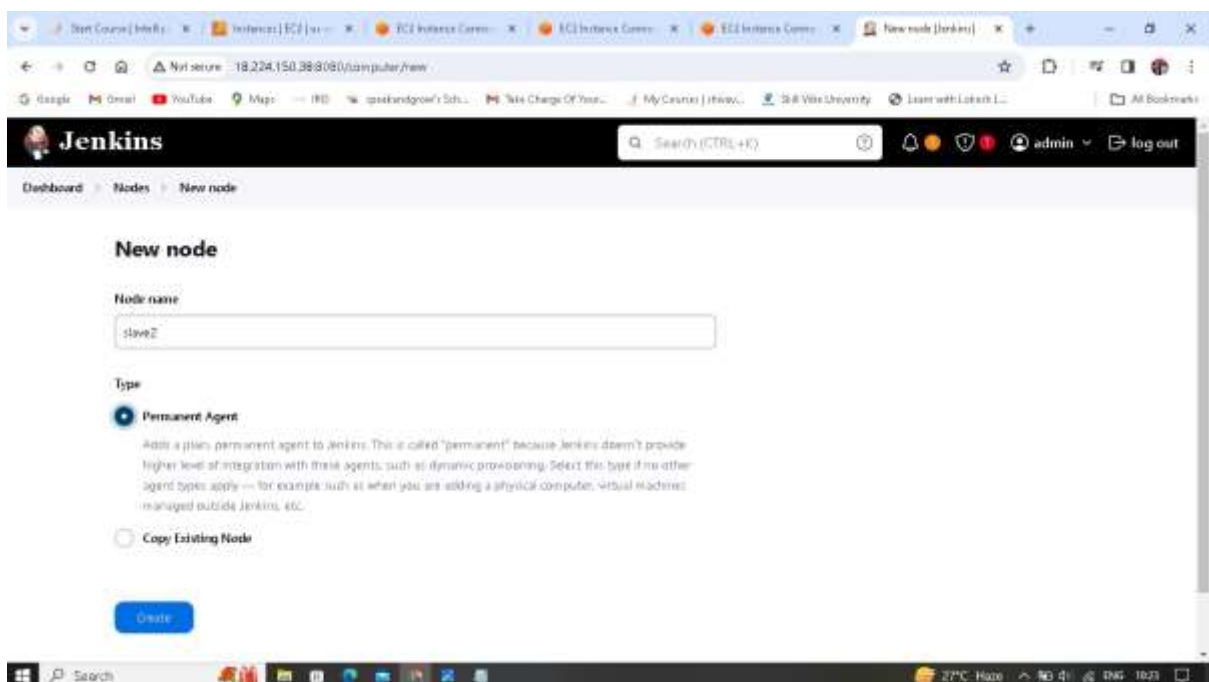


The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and a user profile 'admin' with a 'log out' button. The main content area is titled 'Nodes' and features a '+ New node' button and a 'Configure Monitors' link. A table lists the current nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.04 GB	0 B	4.04 GB	0ms
	slave1	Linux (amd64)	In sync	5.09 GB	0 B	5.09 GB	7ms
Data obtained		0.25 sec	0.21 sec	0.2 sec	0.18 sec	0.21 sec	0.2 sec

Below the table, there are tabs for 'Info', 'Logs', and 'L'. On the left sidebar, there are sections for 'Nodes', 'Clouds', 'Build Queue' (showing 'No builds in the queue'), and 'Build Executor Status' (listing 'Built-In Node' with 1 idle, 'slave1' with 2 idle, and 'dead' with 1 idle).

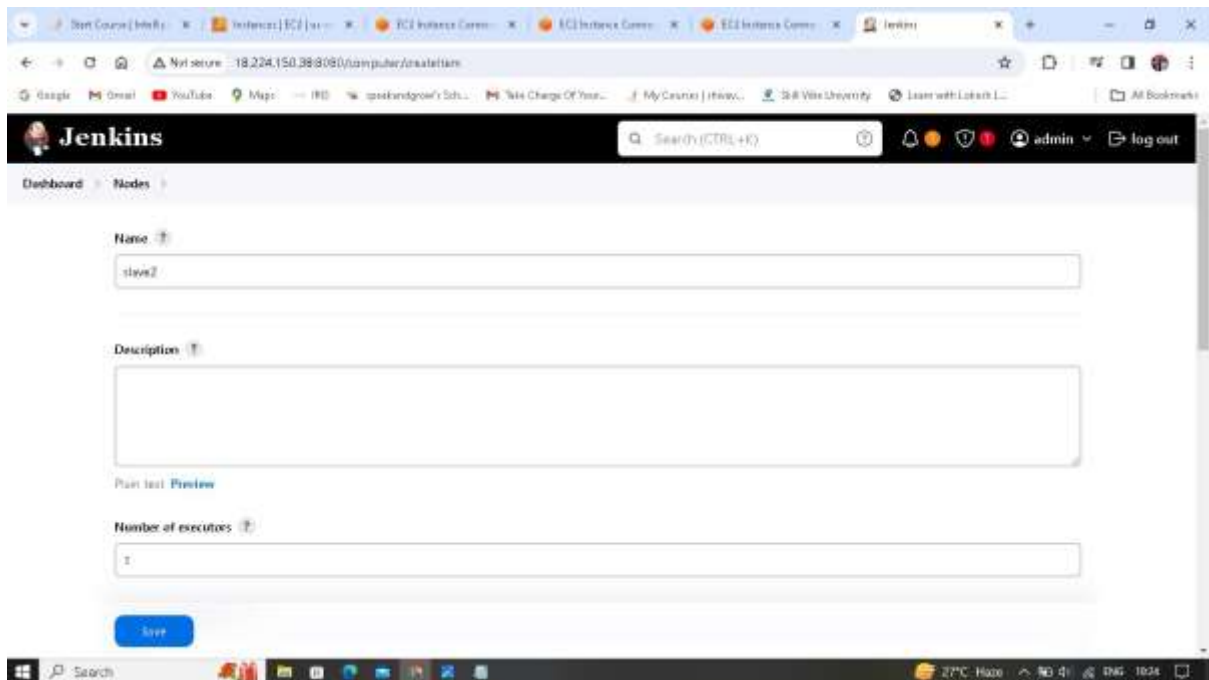
71. Give Node name as slave2, Permanent Agent type and click on create.



The screenshot shows the 'New node' form in the Jenkins interface. The 'Node name' field contains 'slave2'. Under the 'Type' section, 'Permanent Agent' is selected with a radio button. Below this, there is a descriptive text: 'Add a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply --- for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.' The 'Copy Existing Node' option is unselected. A blue 'Create' button is at the bottom of the form.




72.Fill it




The screenshot shows the Jenkins 'New Node' configuration page. The 'Name' field contains 'slave2'. The 'Description' field is empty. The 'Number of executors' field contains '1'. The 'Save' button is at the bottom.


Dashboard > Nodes >

Name 

slave2

Description 

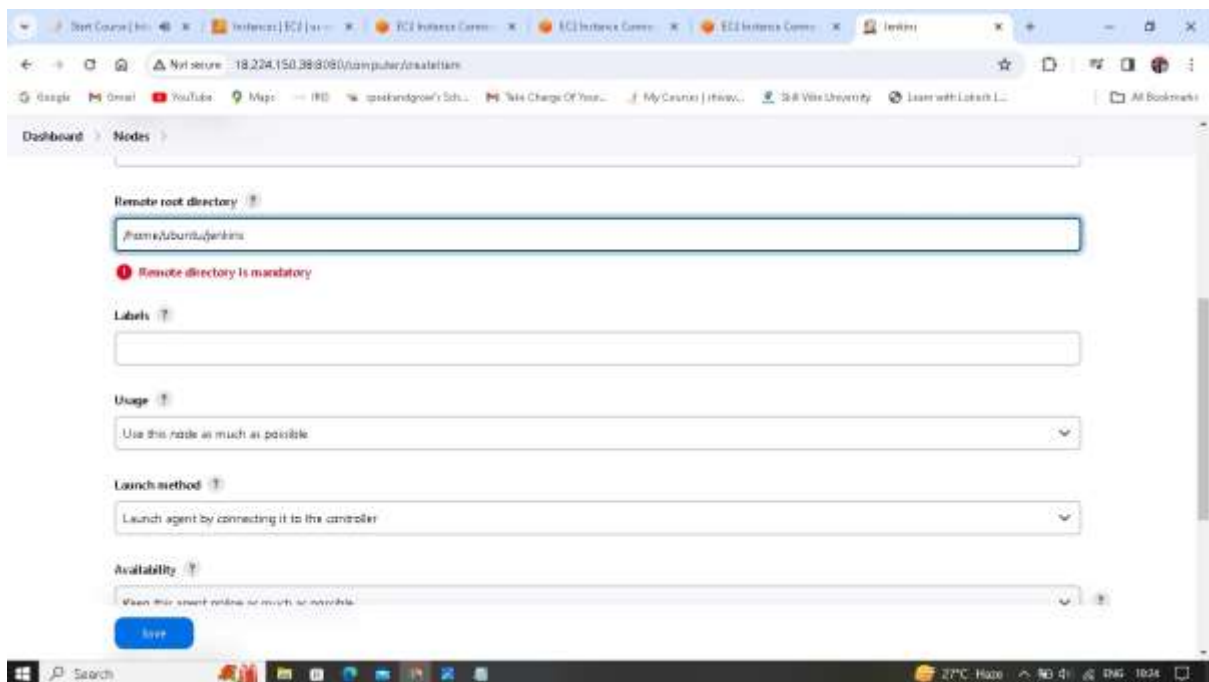
Plain text [Preview](#)

Number of executors 

1


[Save](#)

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





The screenshot shows the Jenkins 'New Node' configuration page. The 'Remote root directory' field contains '/home/ubuntu/jenkins'. A red error message 'Remote directory is mandatory' is displayed below the field. The 'Labels' field is empty. The 'Usage' dropdown is set to 'Use this node as much as possible'. The 'Launch method' dropdown is set to 'Launch agent by connecting it to the controller'. The 'Availability' dropdown is set to 'When this agent online or much as possible'. The 'Save' button is at the bottom.


Dashboard > Nodes >

Remote root directory 


/home/ubuntu/jenkins

 Remote directory is mandatory


Labels 

Usage 

Use this node as much as possible

Launch method 

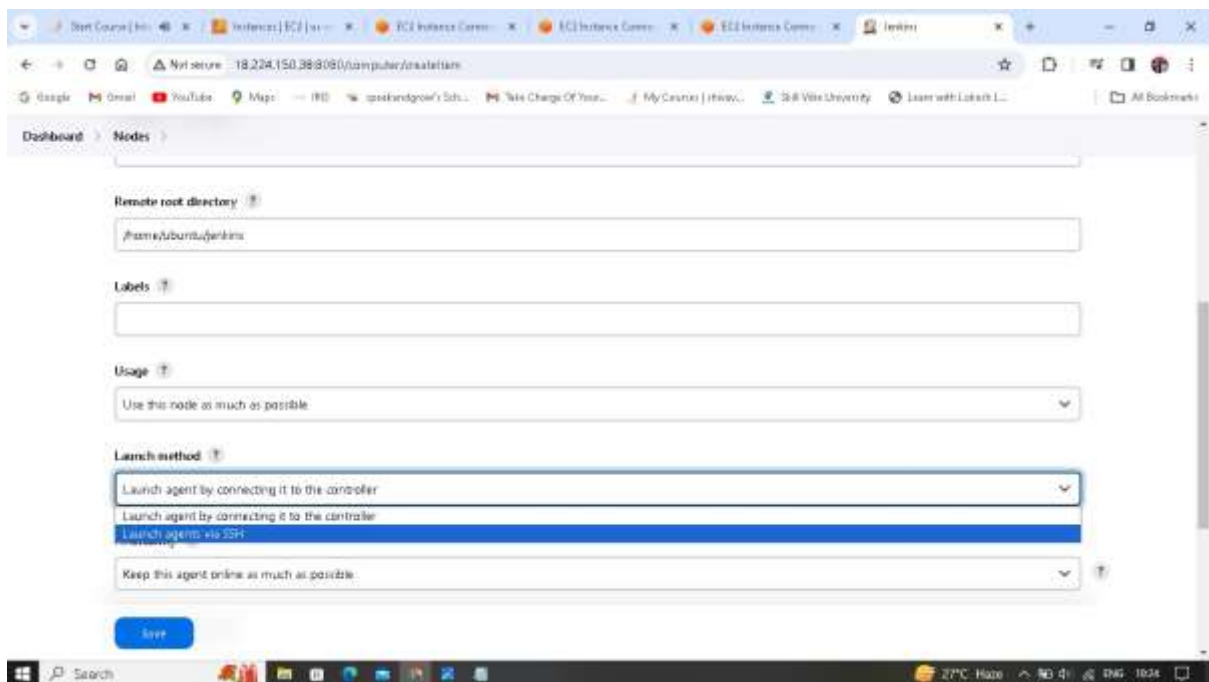
Launch agent by connecting it to the controller

Availability 

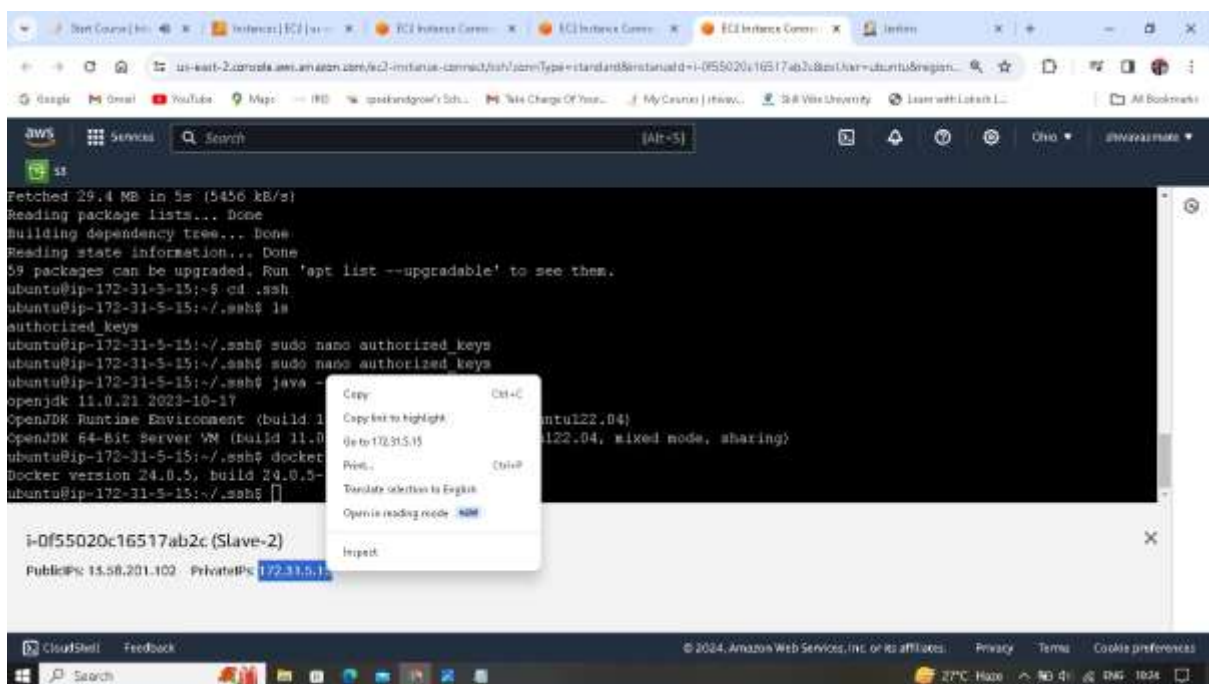
When this agent online or much as possible

[Save](#)

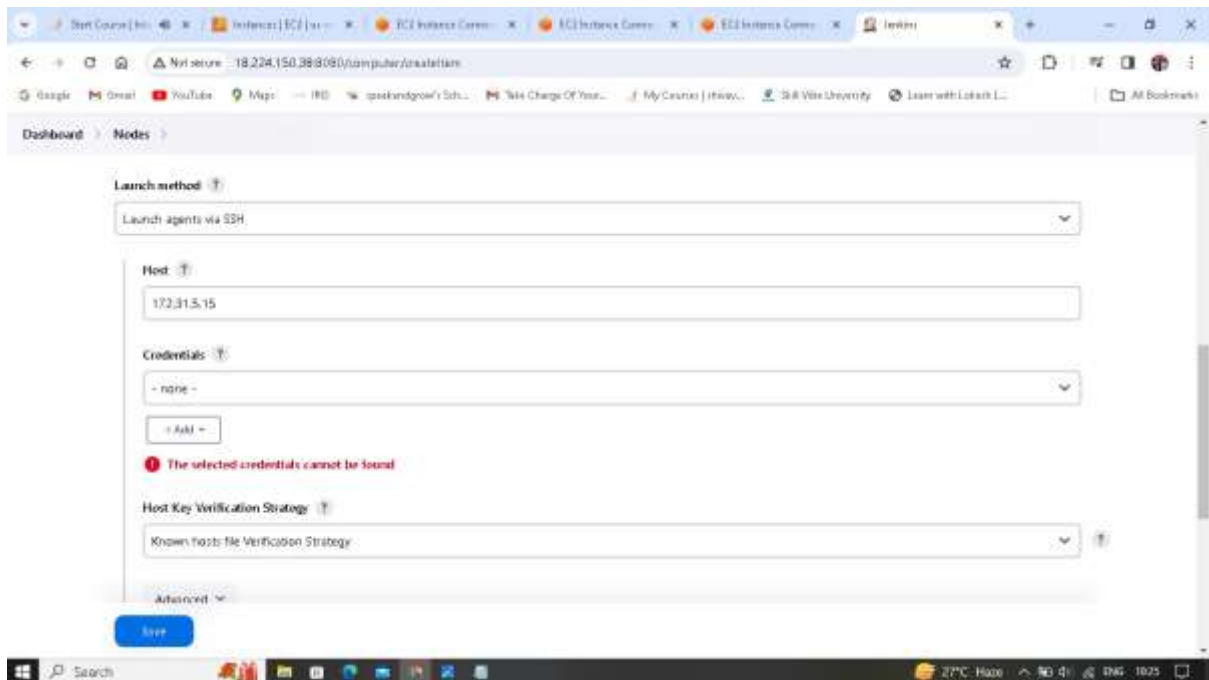
74. In Launch method choose Launch agents via SSH



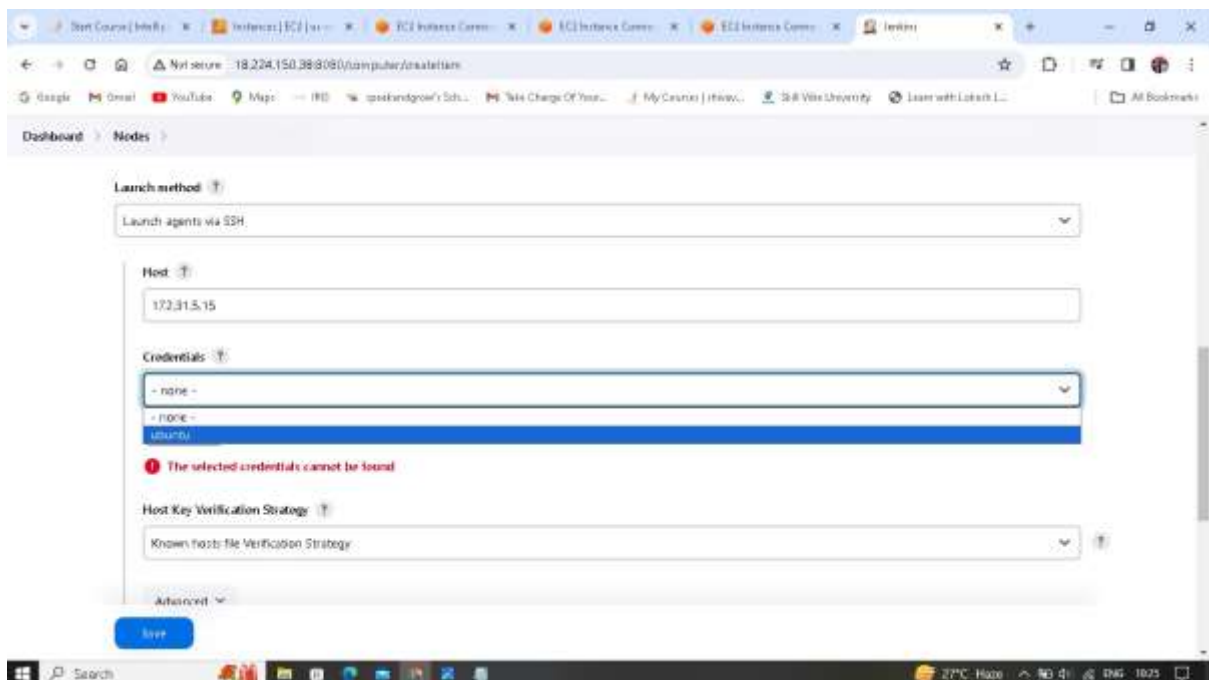
75. Go to Slave2 copy the private IP



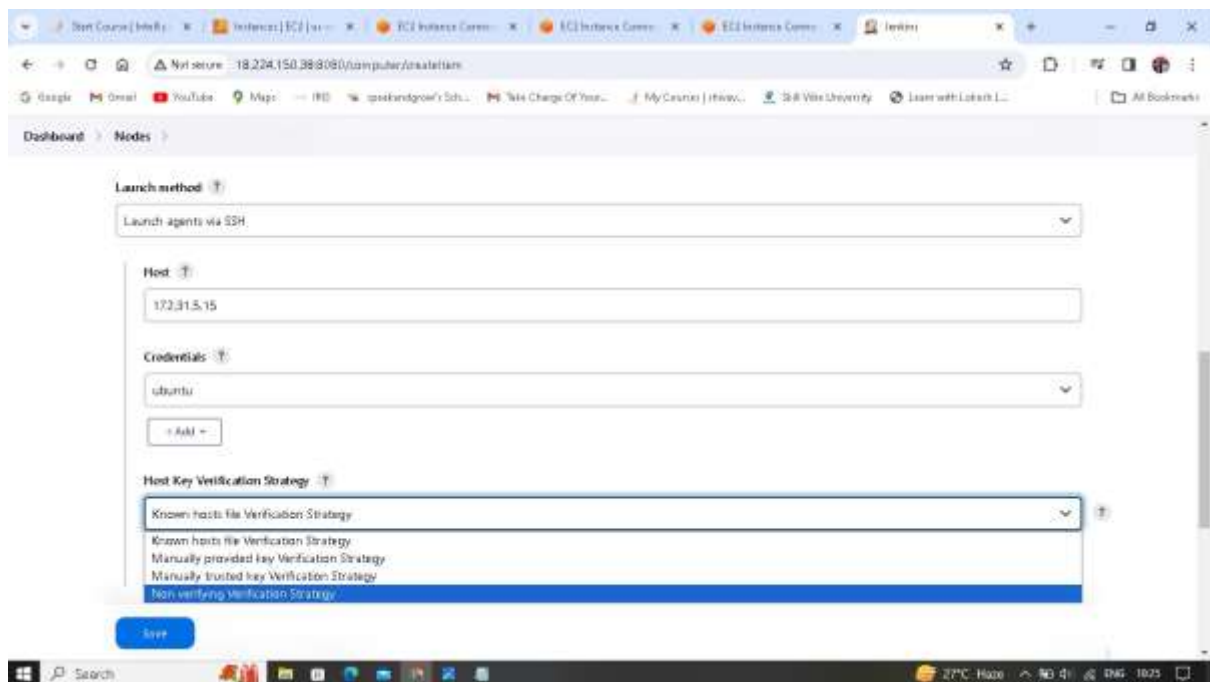
76. Under Host paste it.



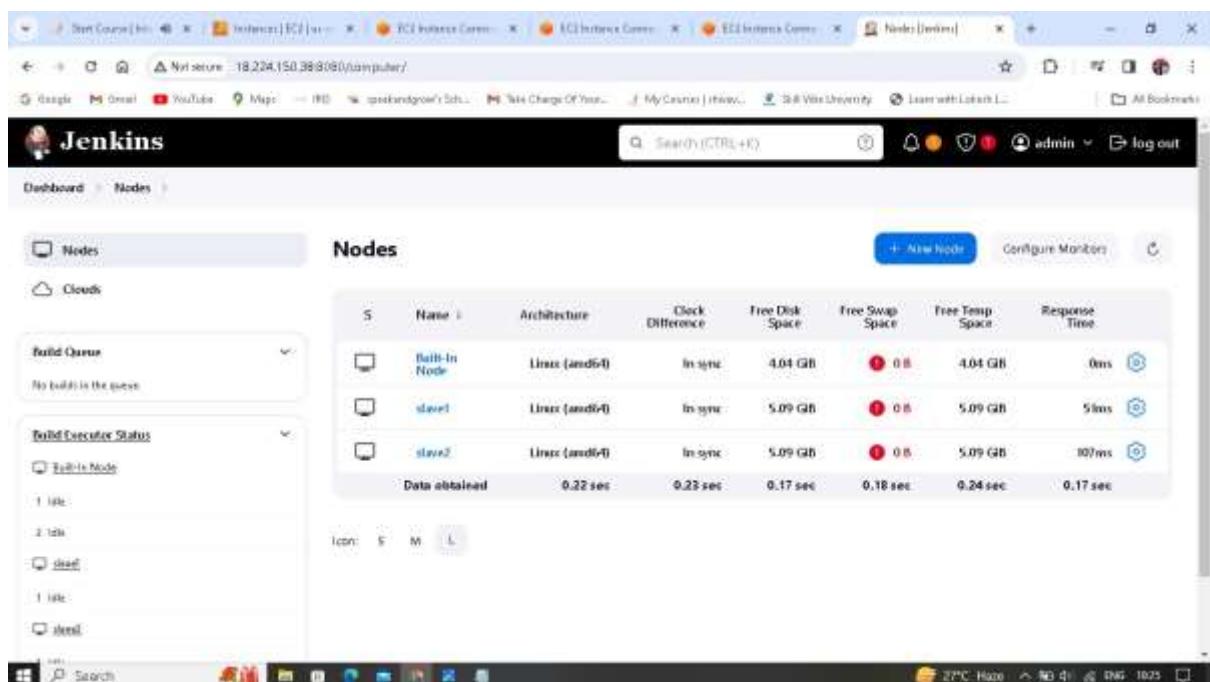
77. This time just choose ubuntu for Credentials.



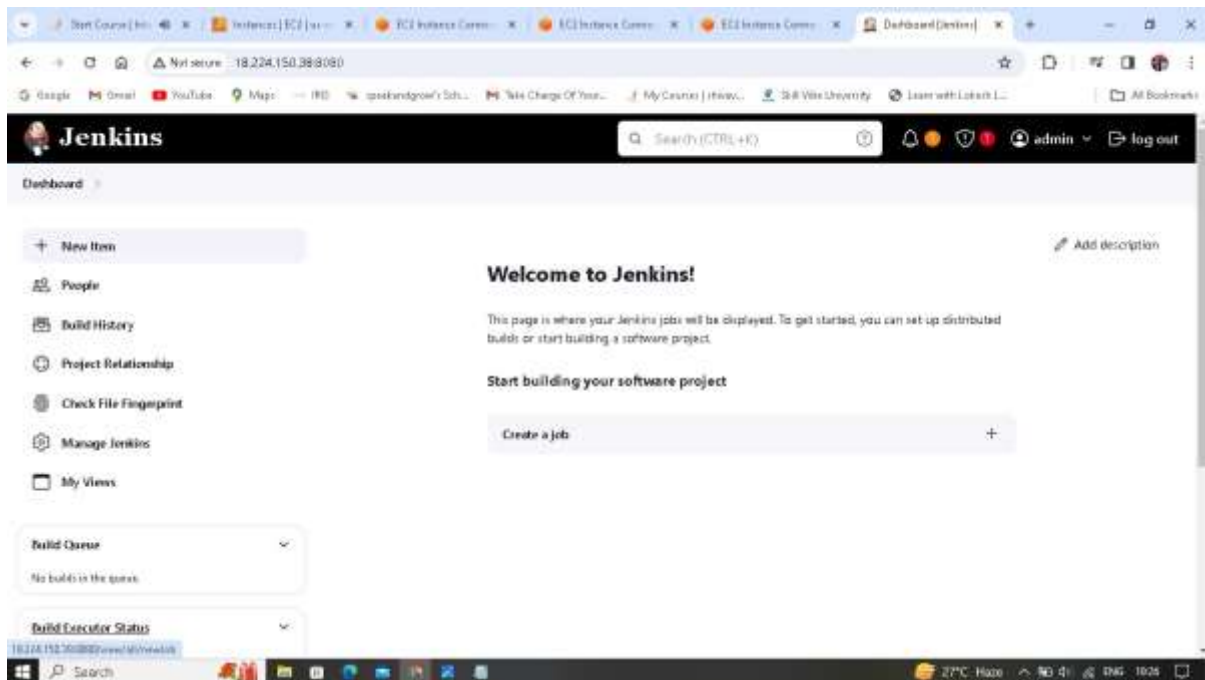
78. Under Host Key Verification Strategy choose Non verifying and click on save



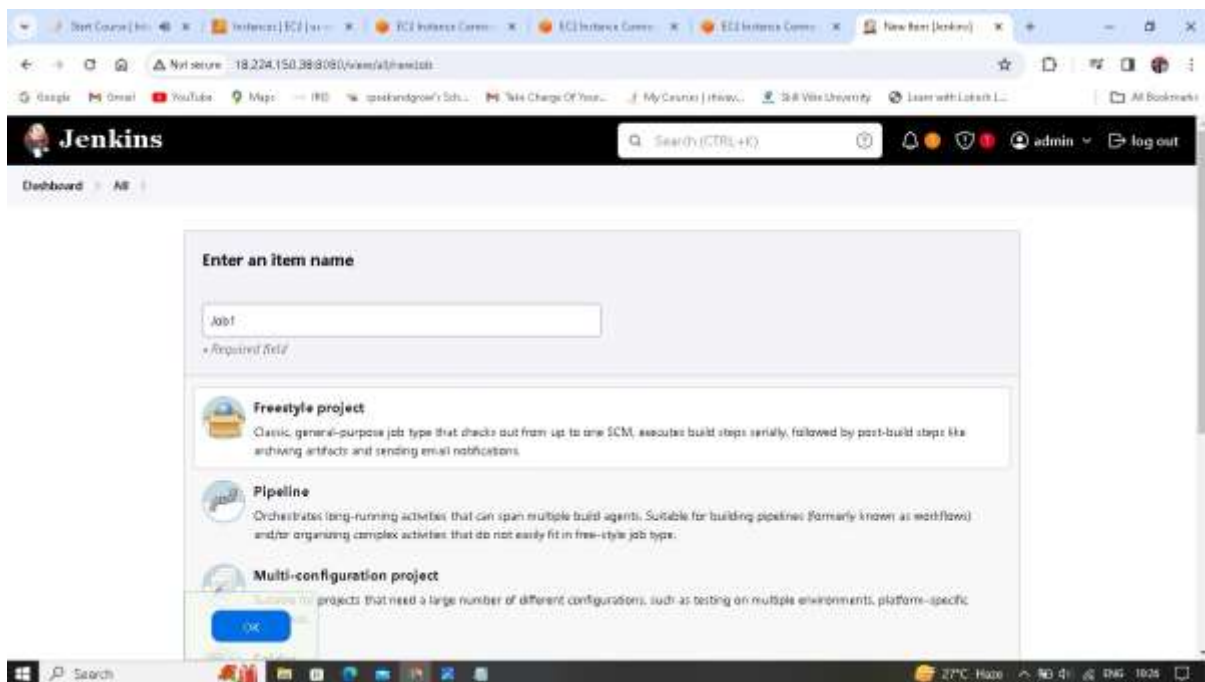
79. Now slave2 also running.



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

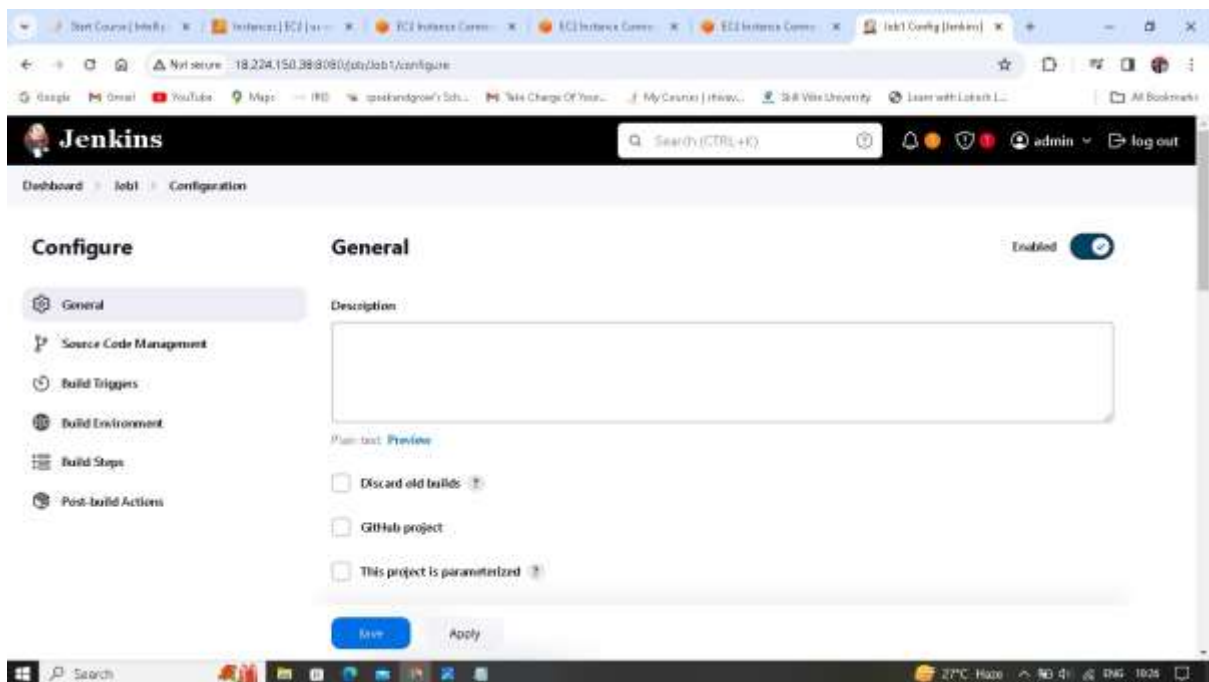


81.Enter an item name as Job1 and choose Freestyle project click on ok.

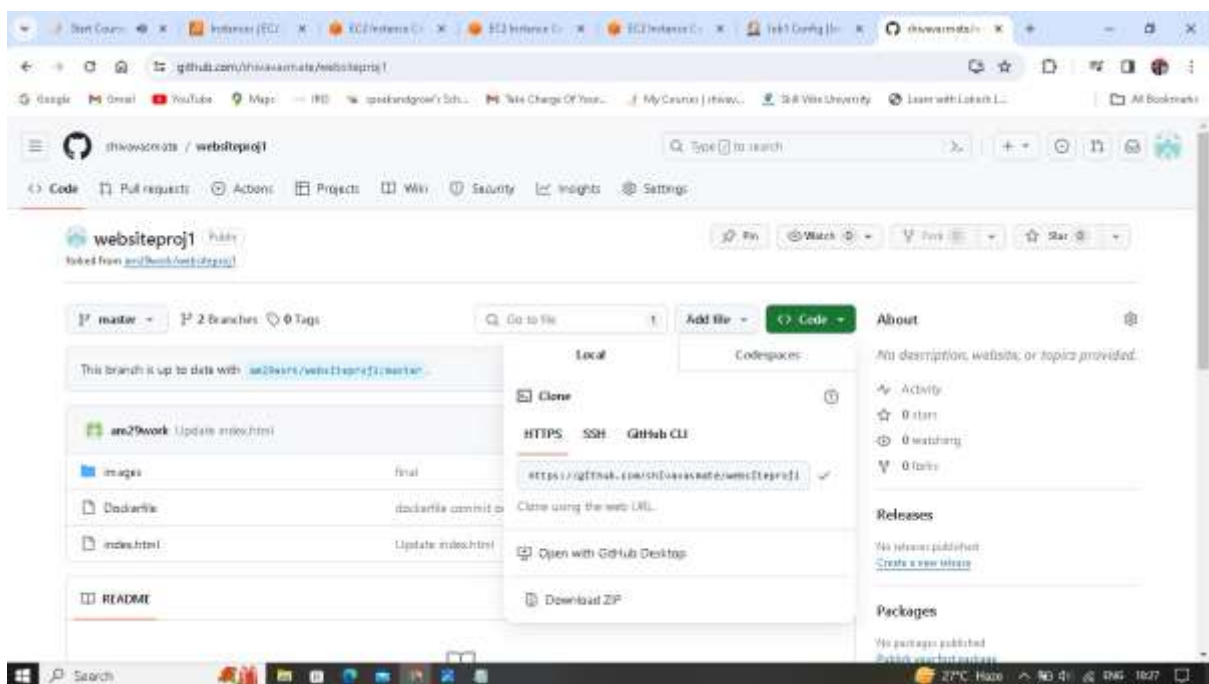




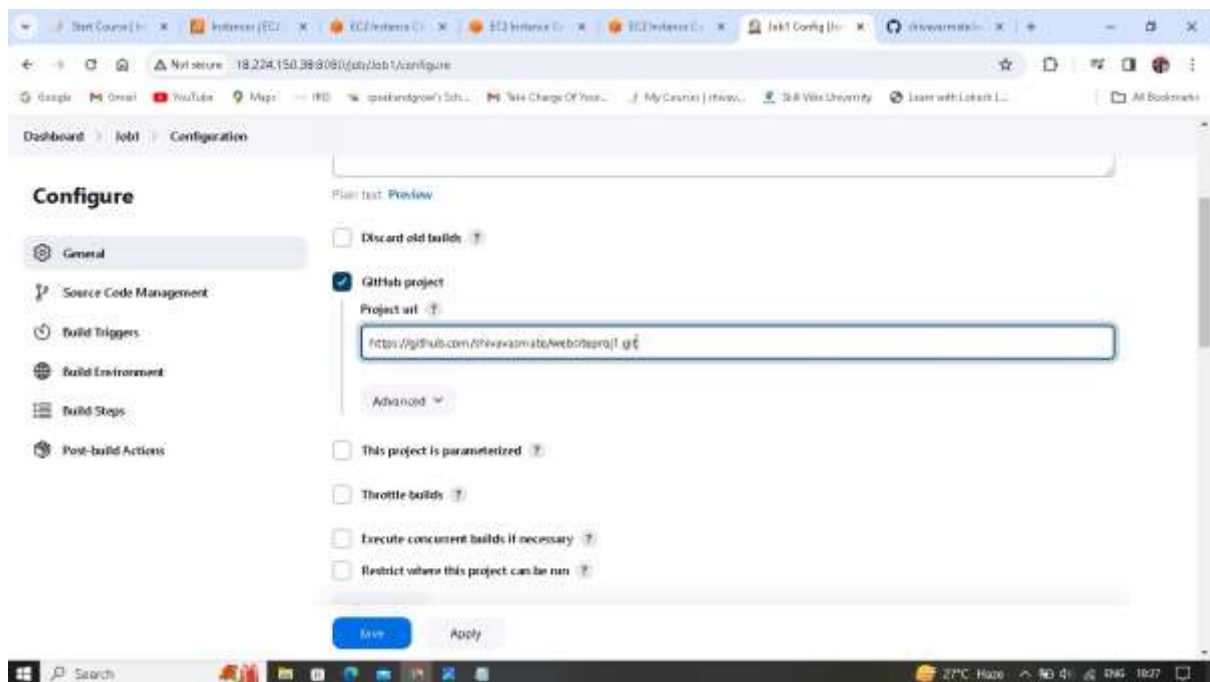
82.Fill one by one. check in GitHub project.



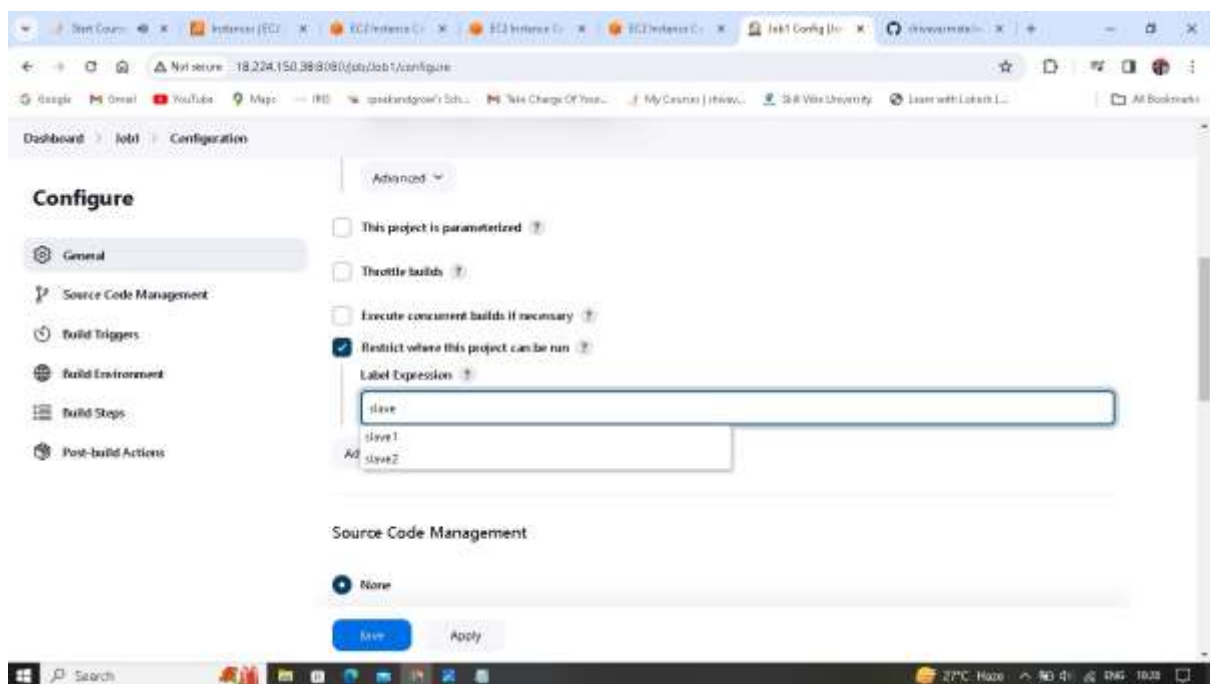
83.Go to github copy the url



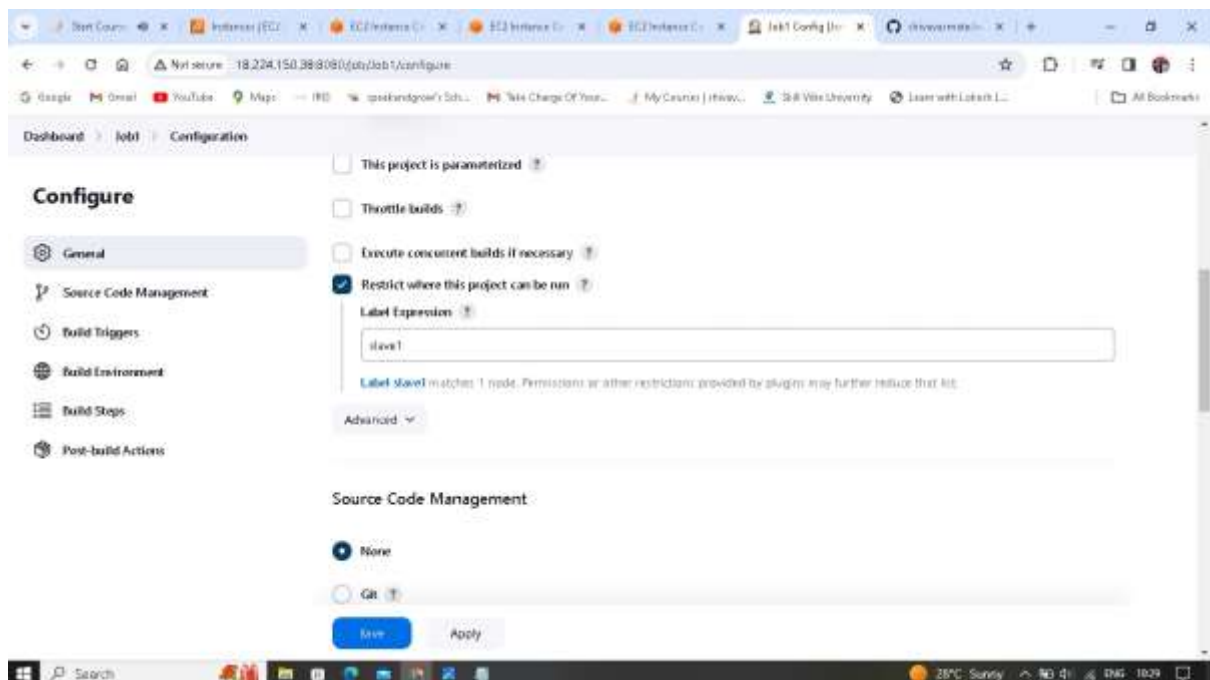
84.Paste it



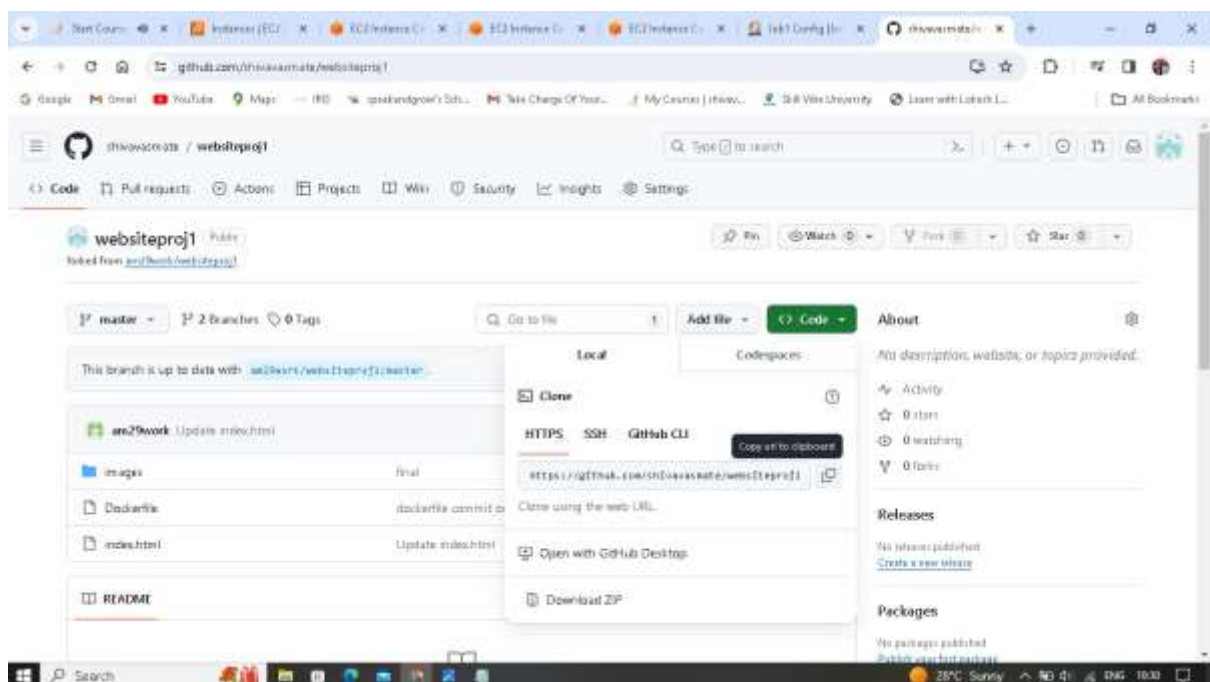
85.Under Restrict where this project can be run type slave1.



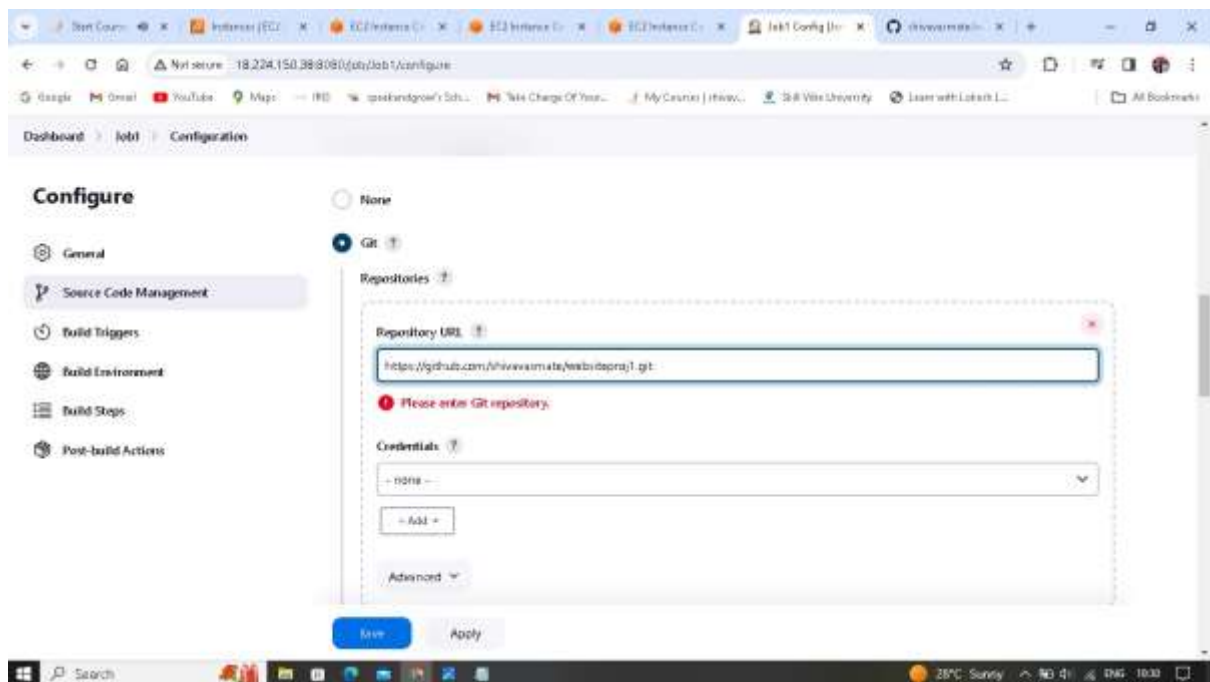
86. Blue colour will appear as Label slave1 and under source code management check in Git.



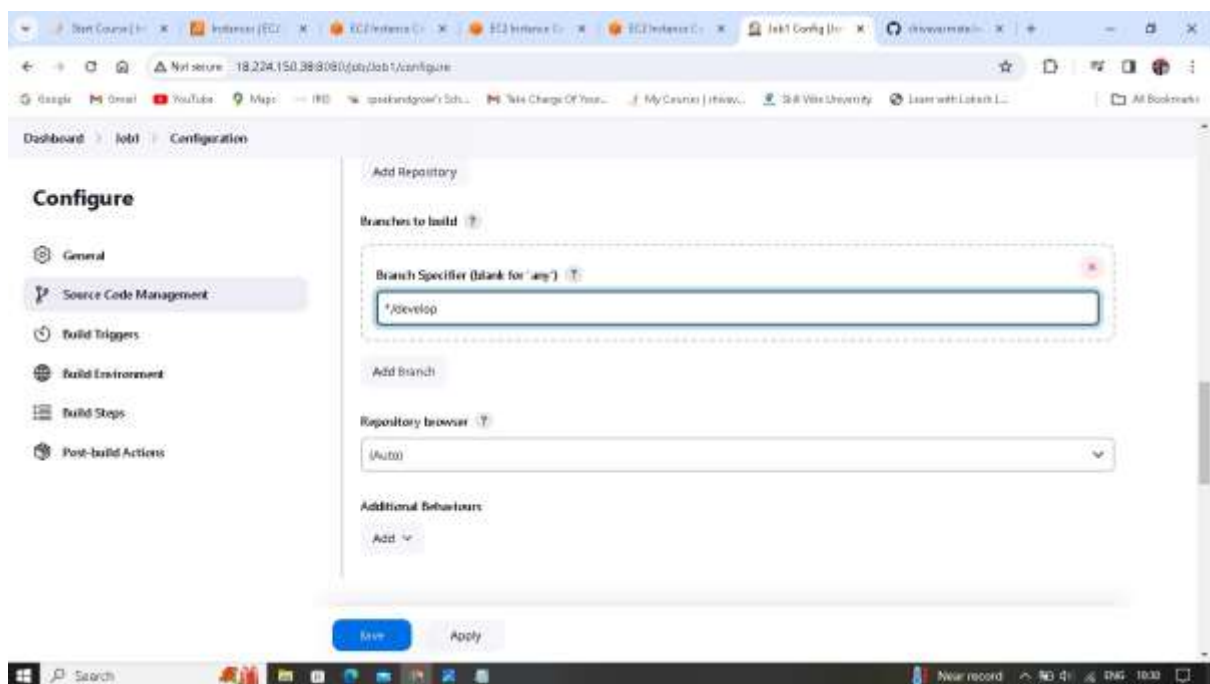
87. Copy the URL for github.



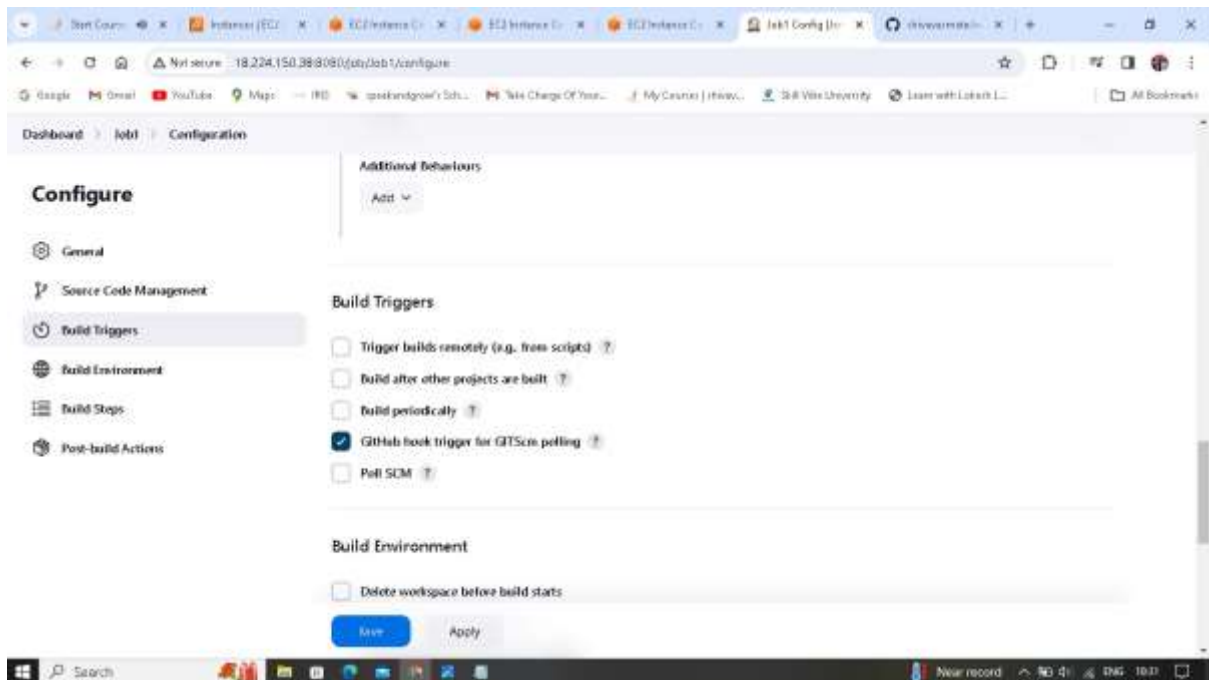
## 88. Paste it in Repository URL



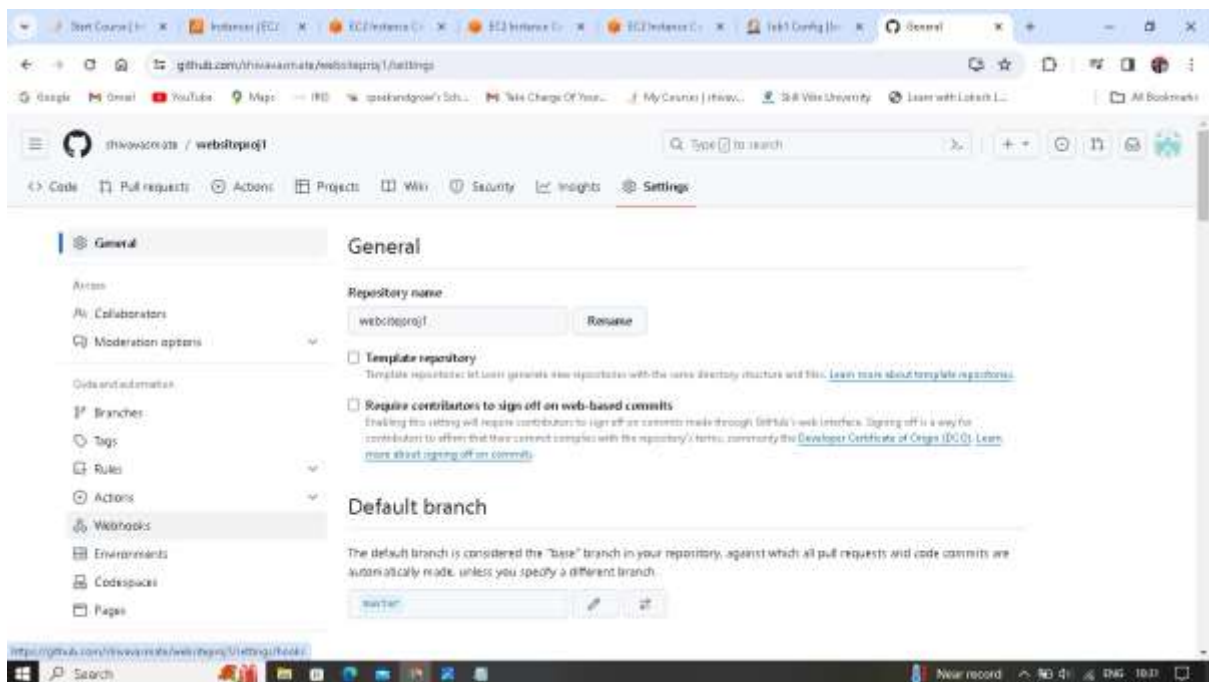
## 89. Under Branches to build choose \*/develop as Branch Specifier.



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

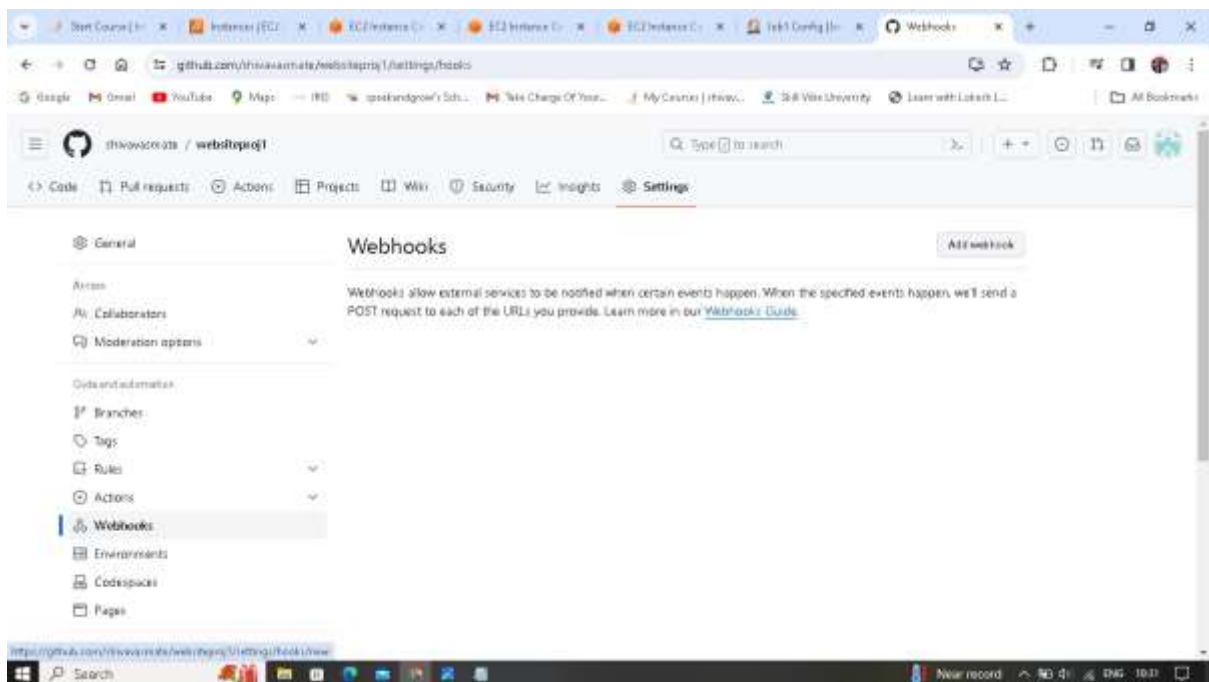


91. For it go to github under repository setting

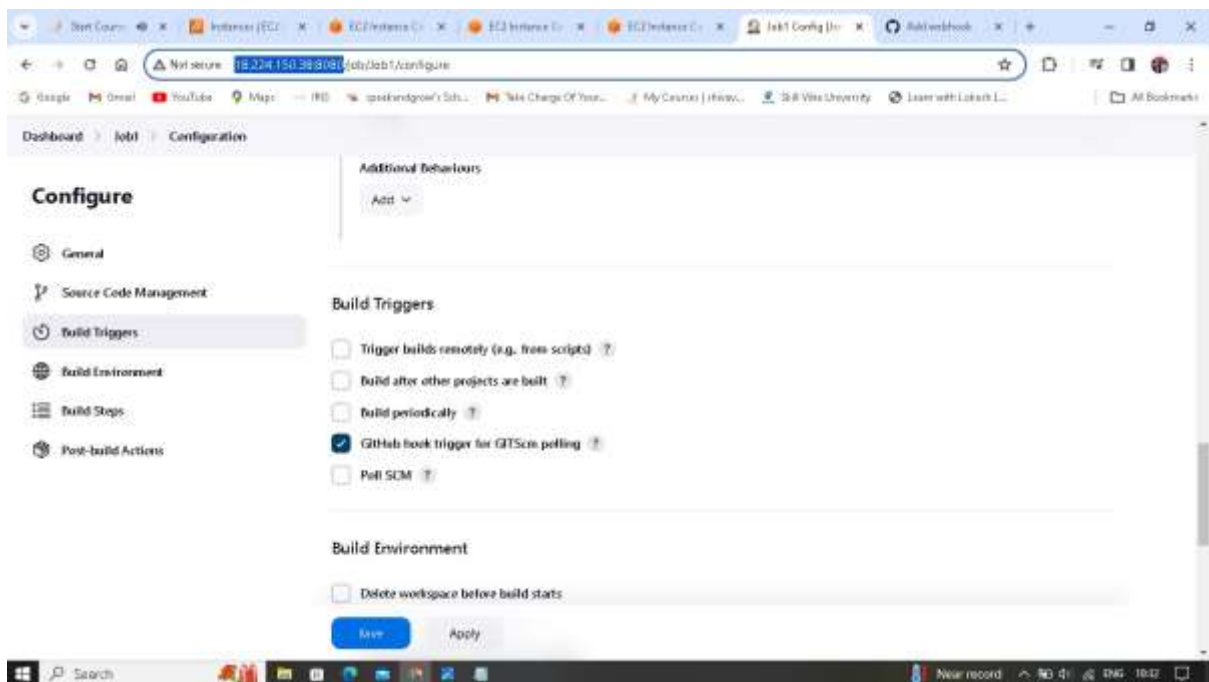




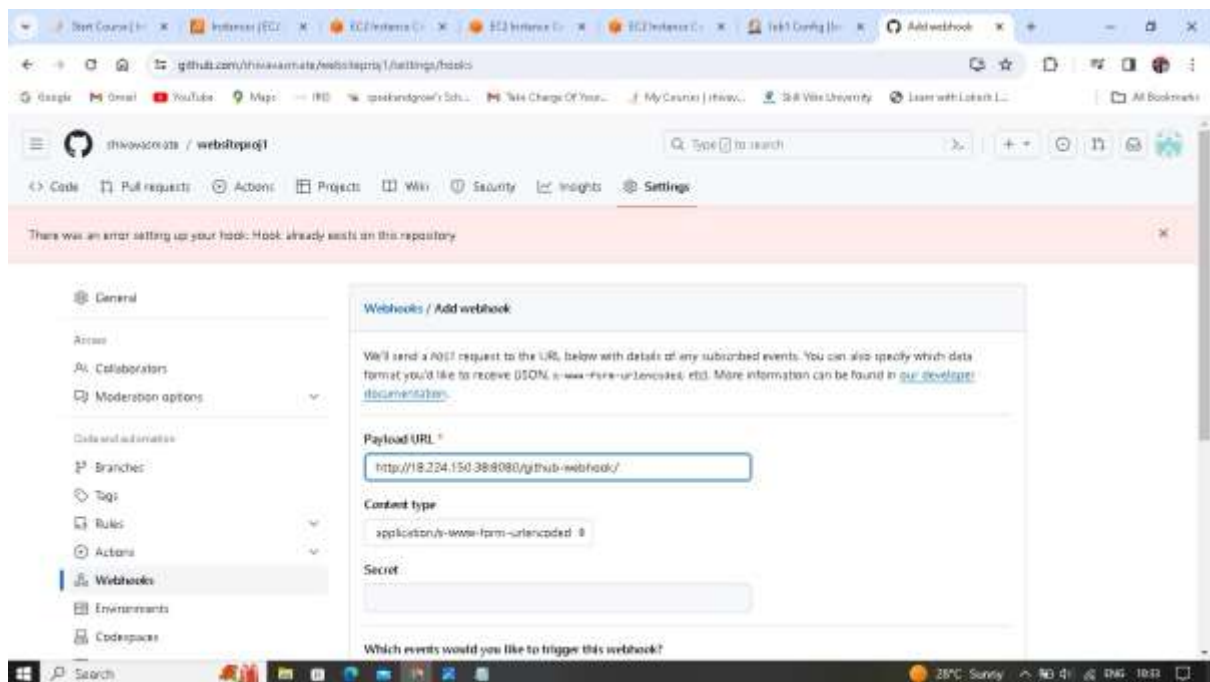
92. On left side click on Webhooks under it click on add webhook.



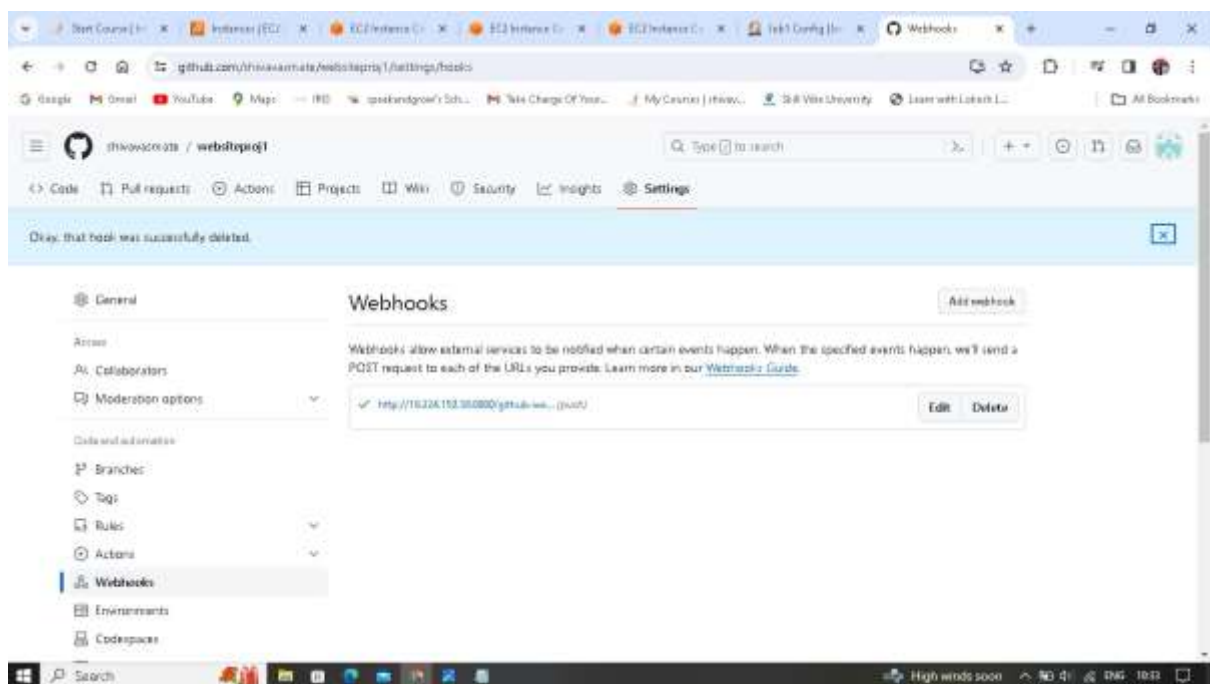
93. Copy the URL of Jenkins



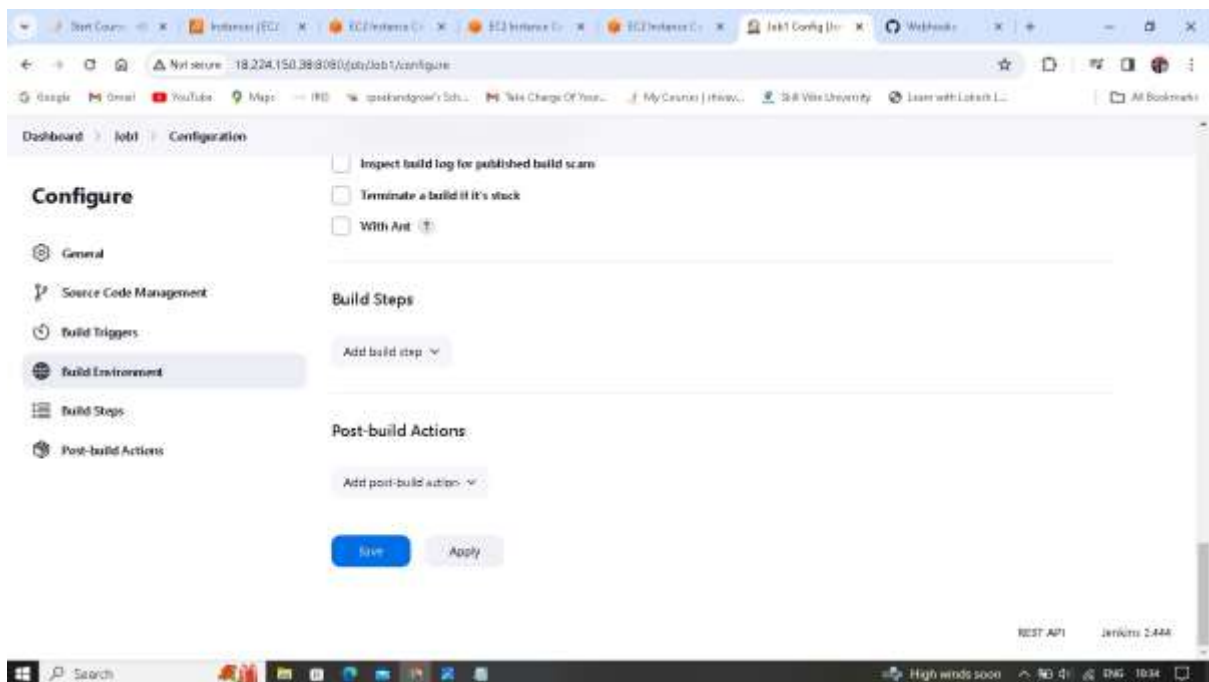
94. Under Payload URL paste it with /github-webhook/ and click active



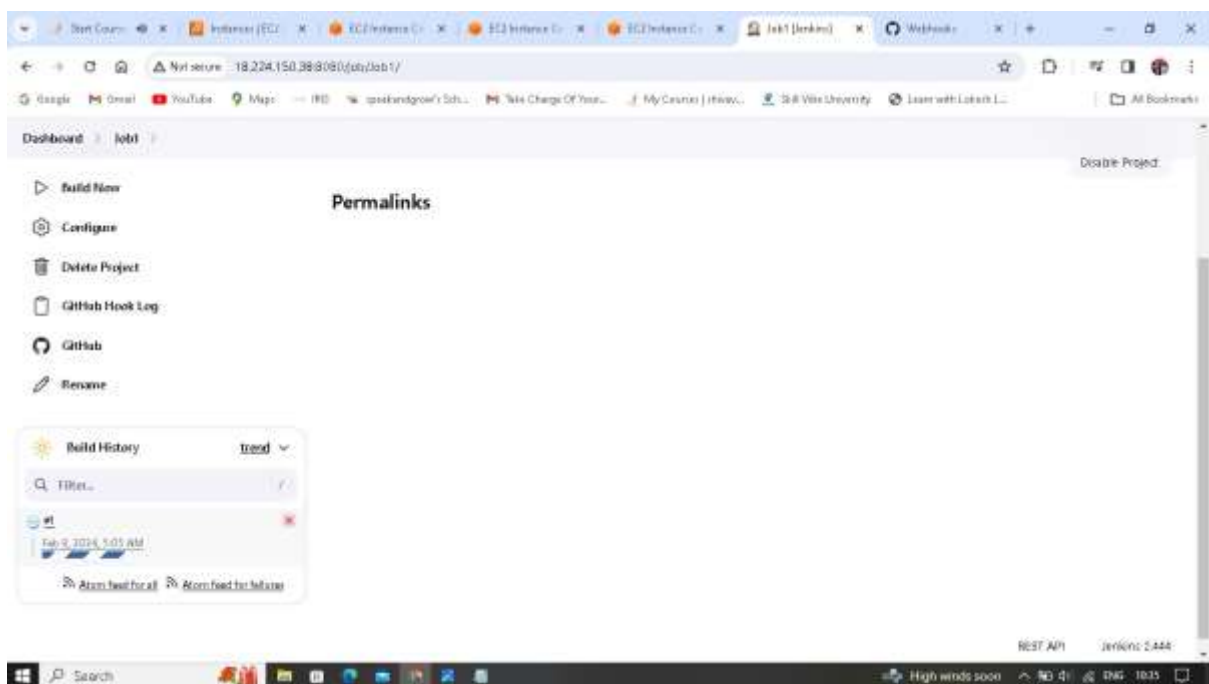
95. Check for tick mark for it.



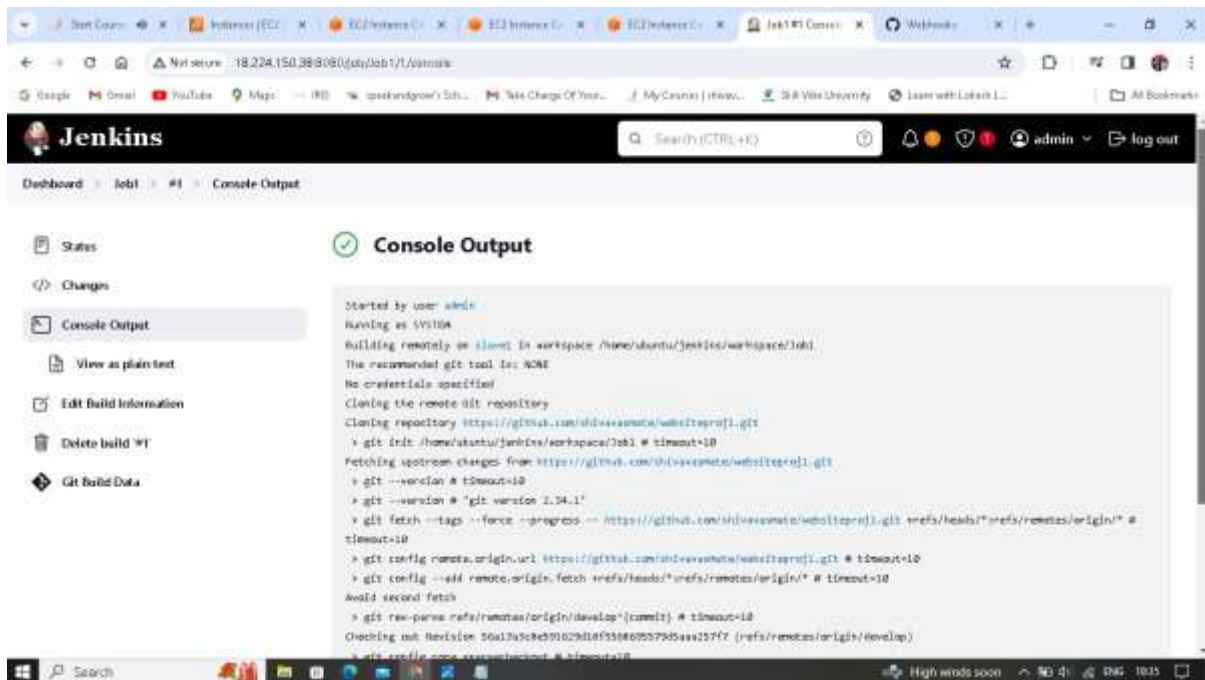
96. Click on both Apply and Save.



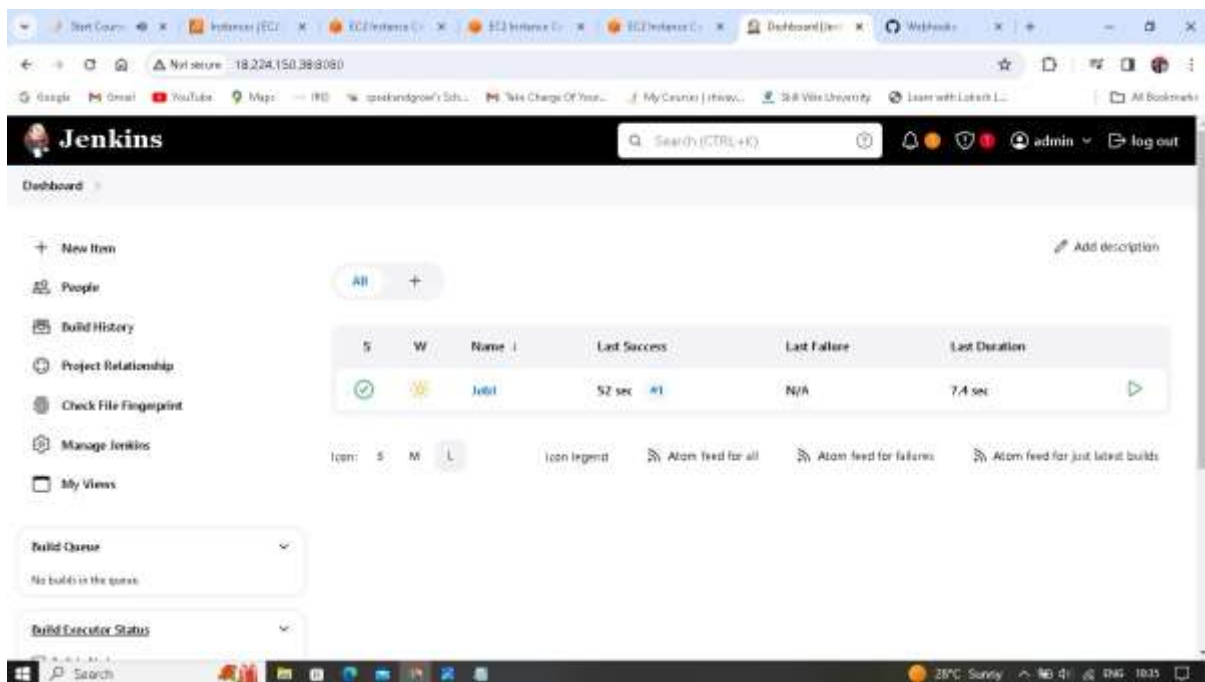
97. Click on Build Now #1 is running



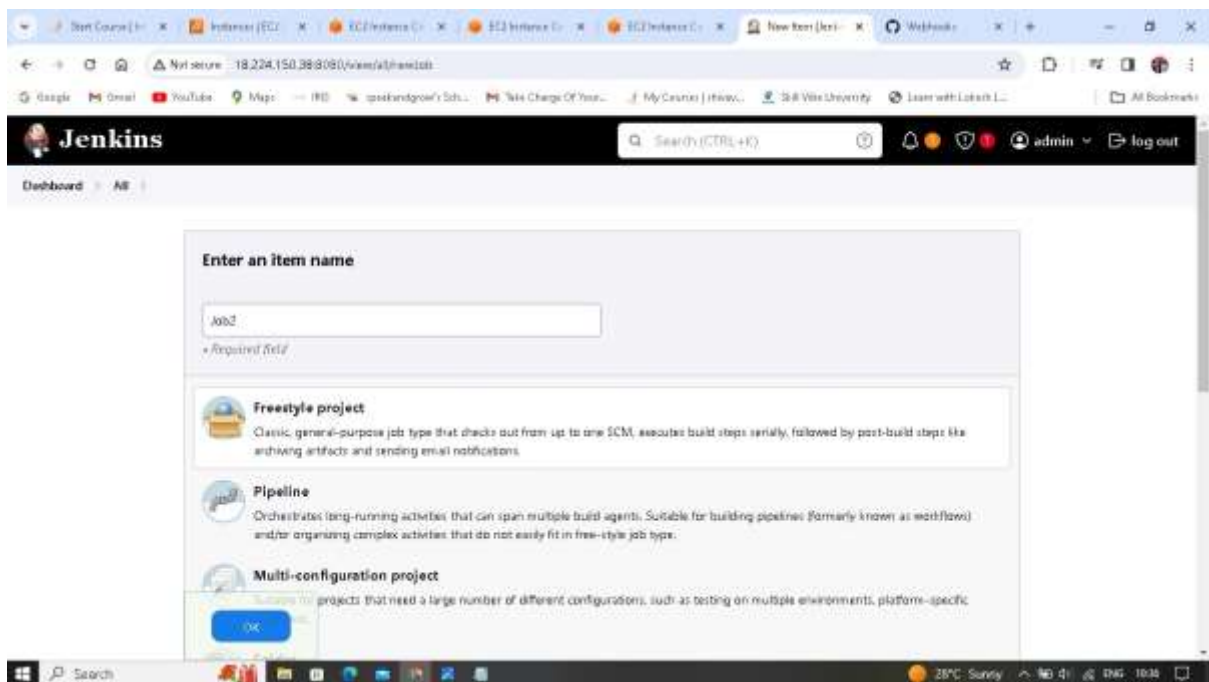
98. Job1 is successful is created.



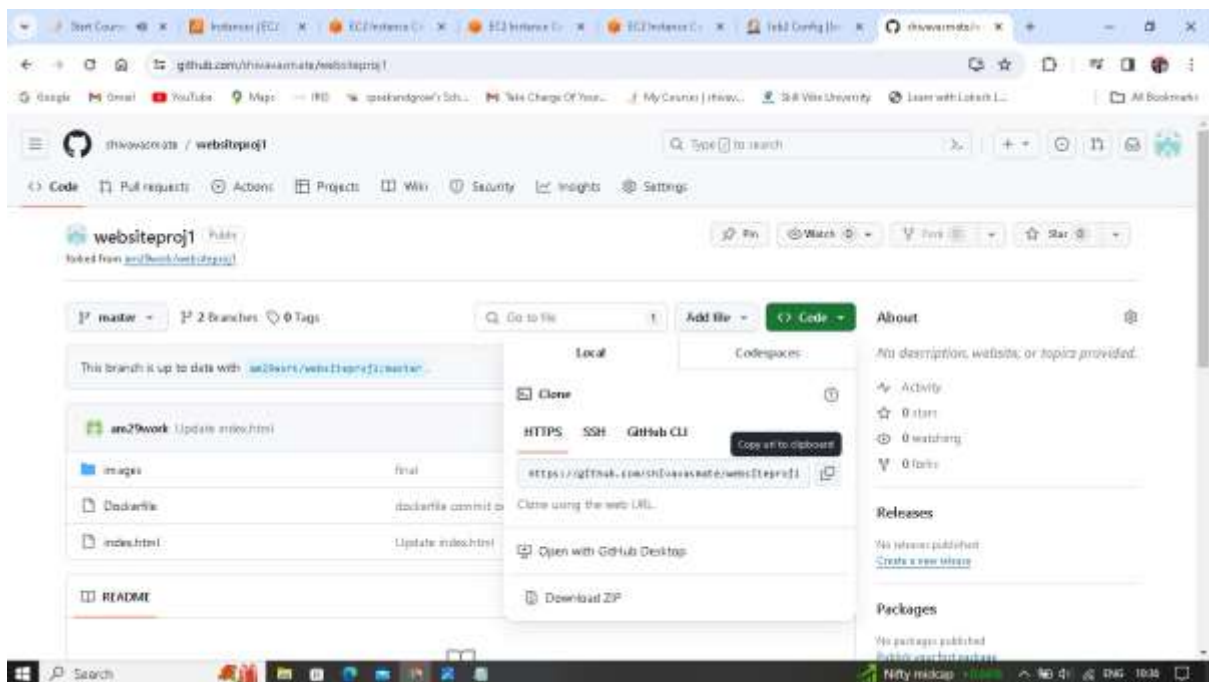
99. Go to Dashboard to new job click on +New Item.



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

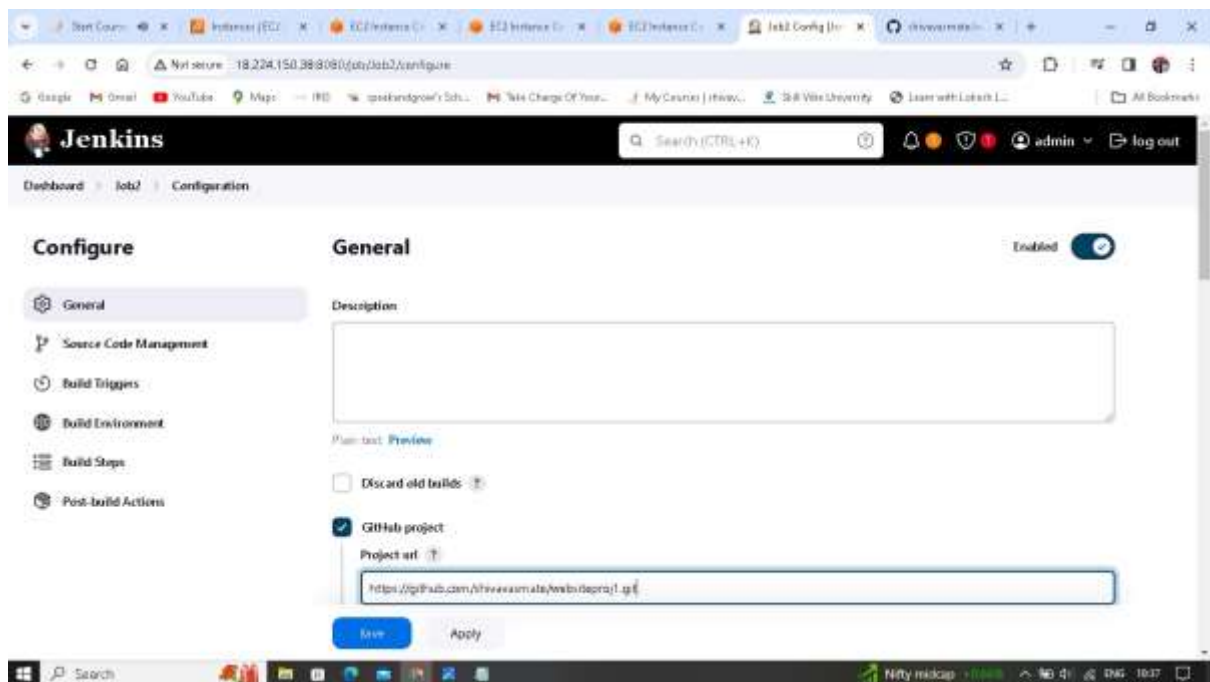


101. Copy the URL from github repo.

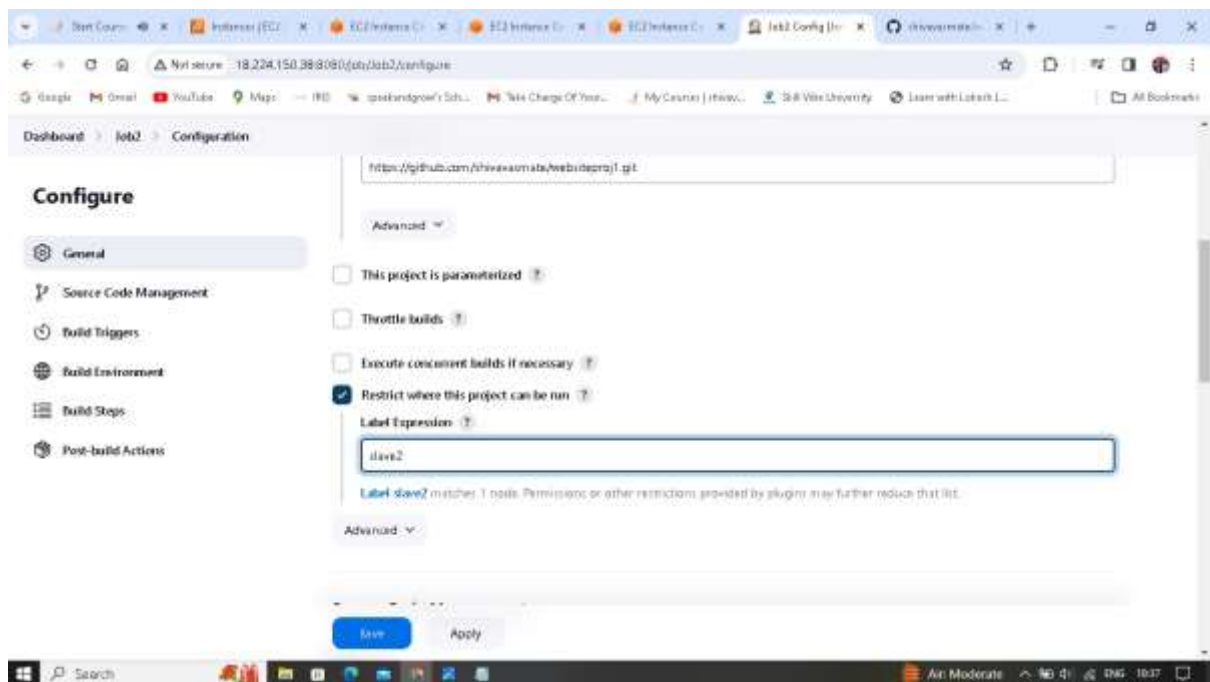




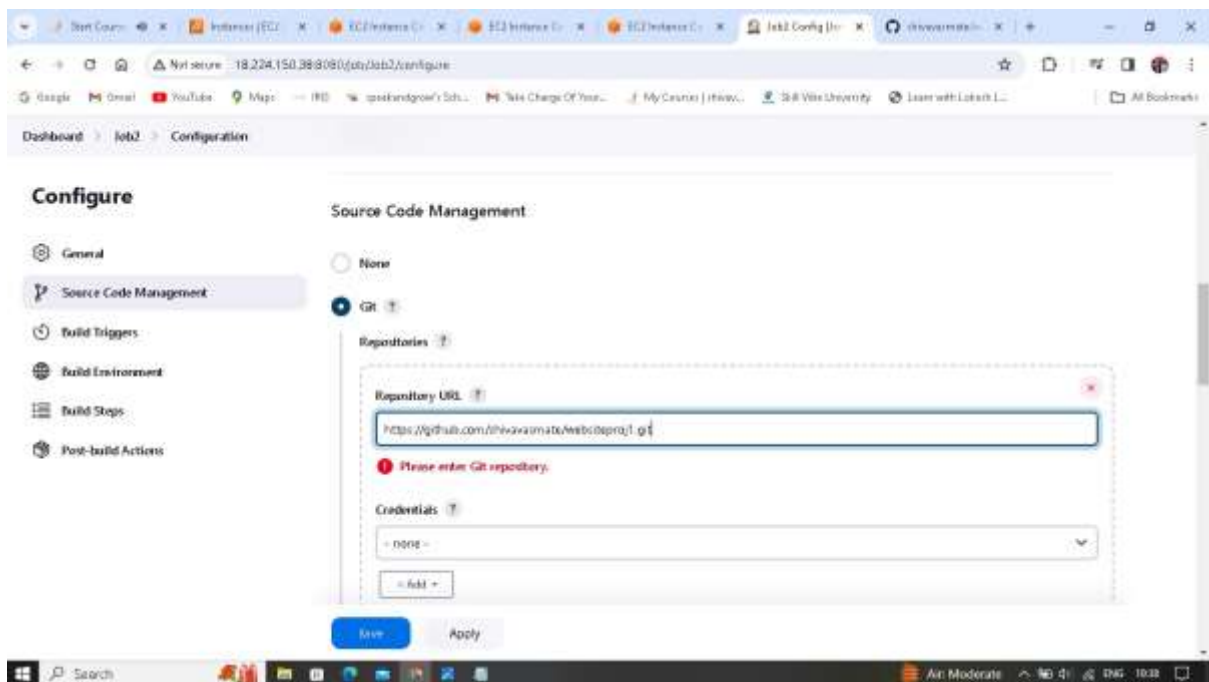
102. Paste it



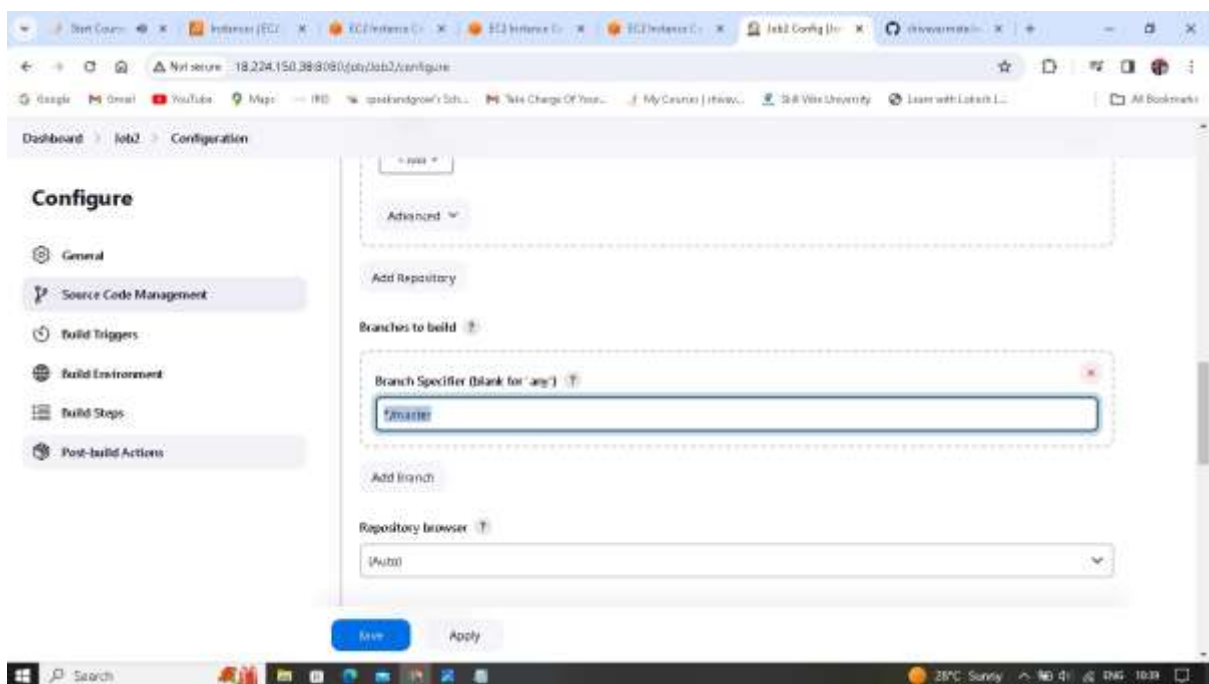
103. This time type slave2.



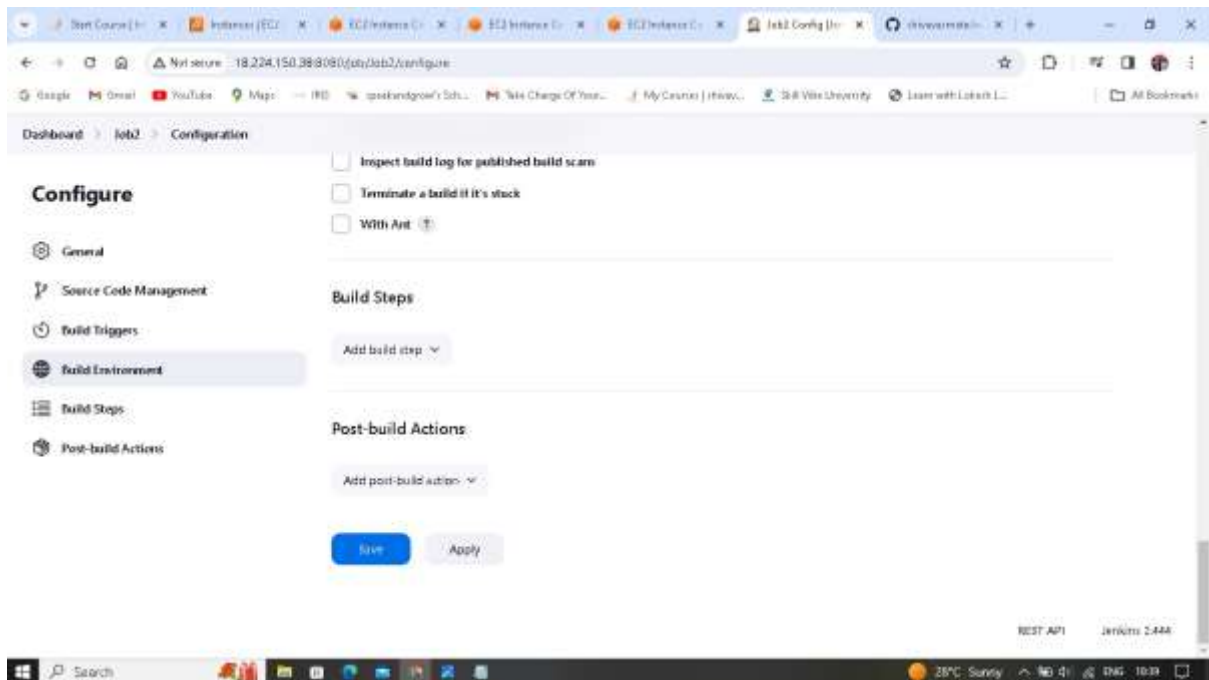
104. Under Source code management paste Repository URL from github repo.



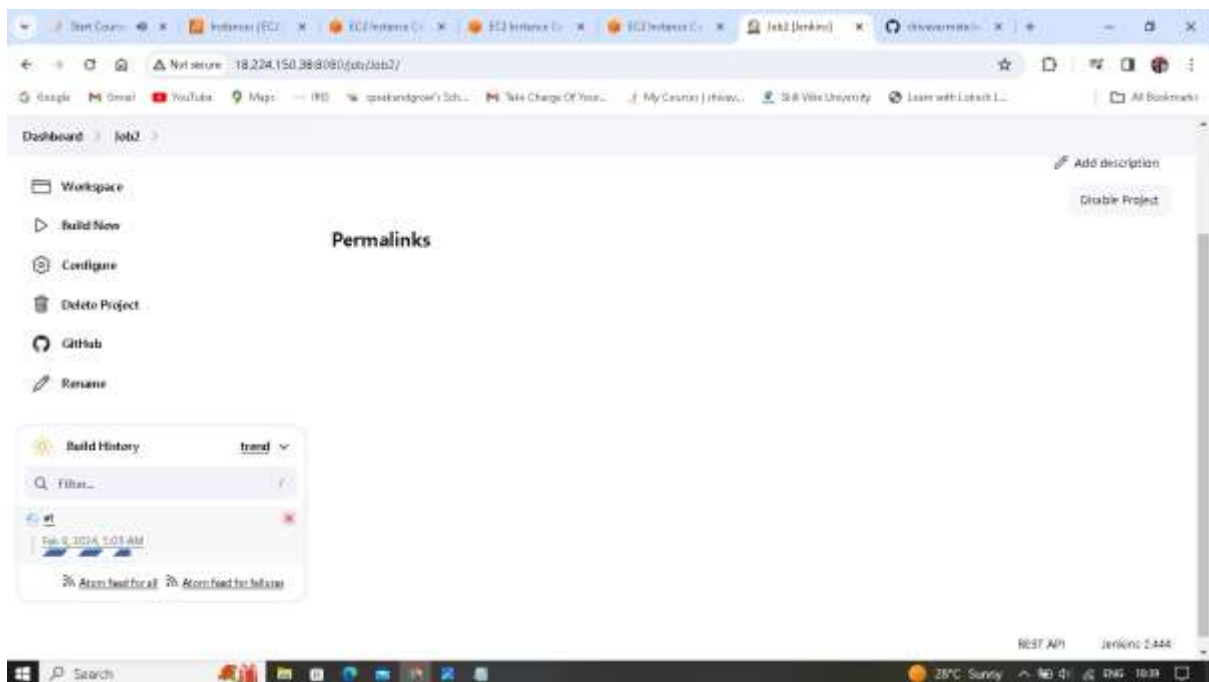
105. This time Branch specifier give \*/master.



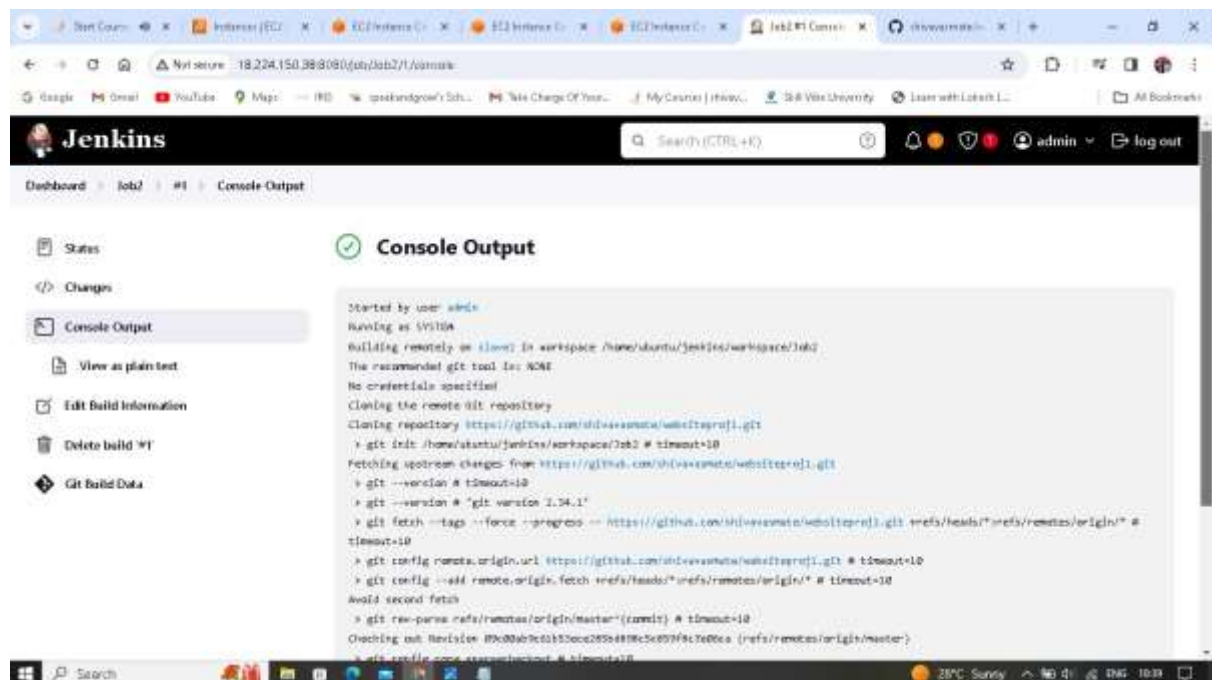
106. Click on both Apply and Save.



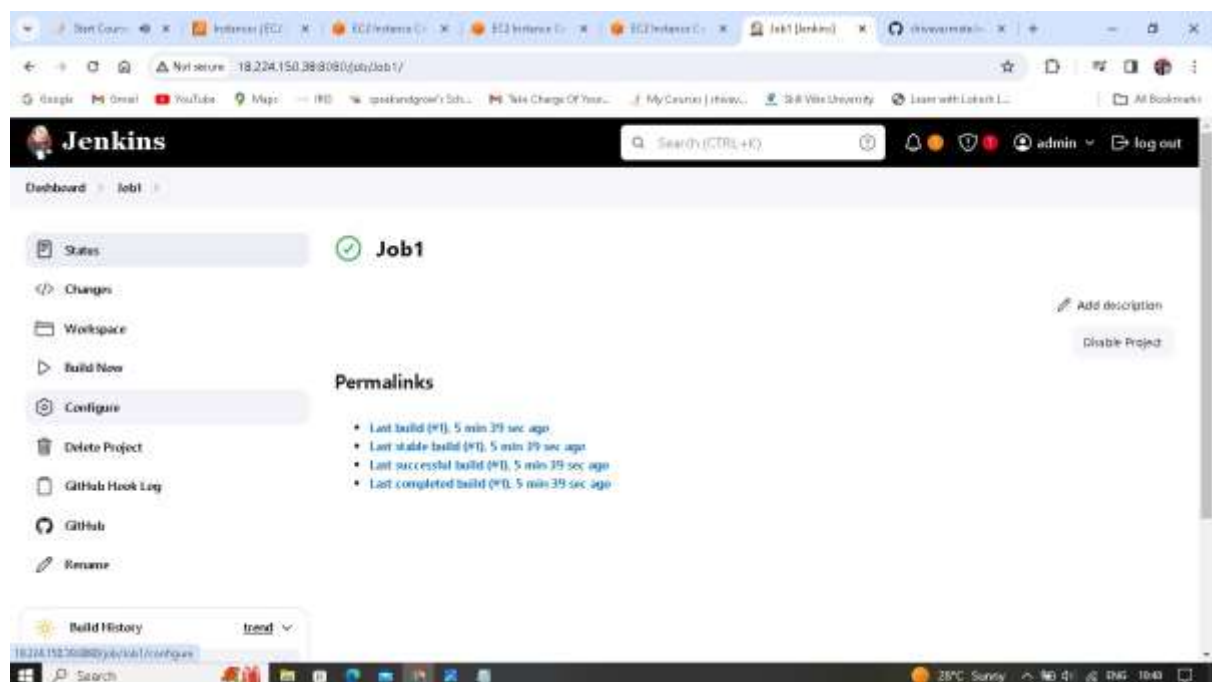
107. Click on Build Now #1 is running



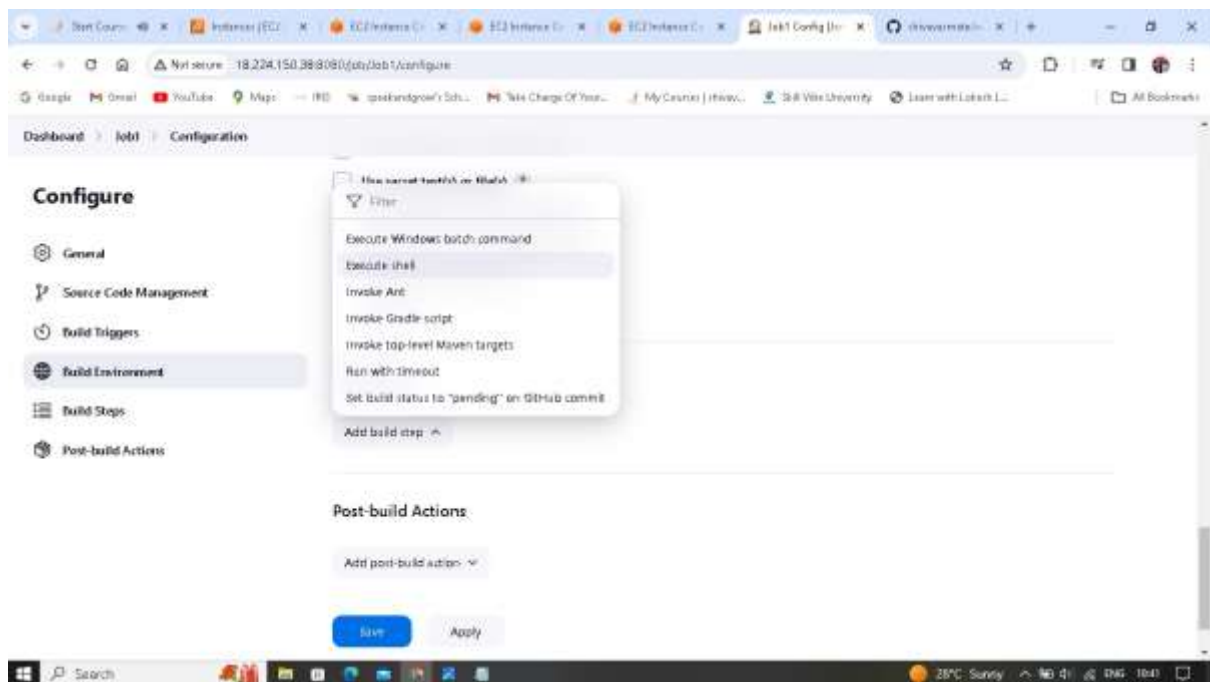
108. Job2 is successfully running.



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



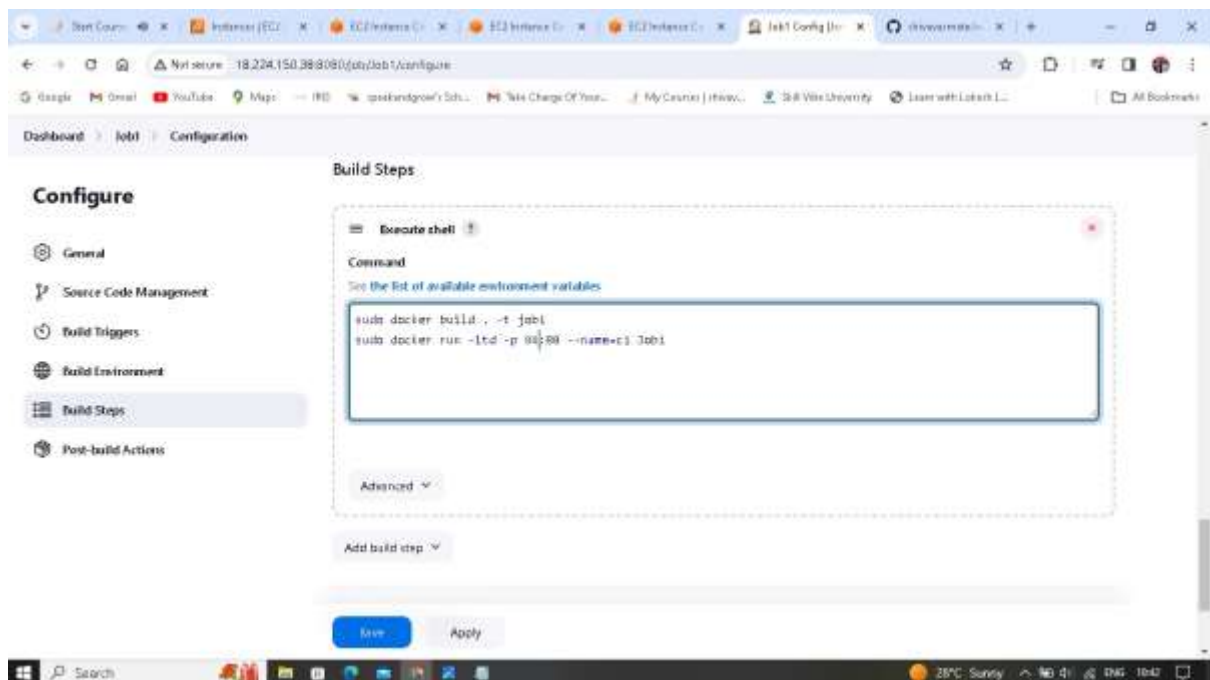
110. Go at the end under Add build step choose Execute shell.



111. Under it type

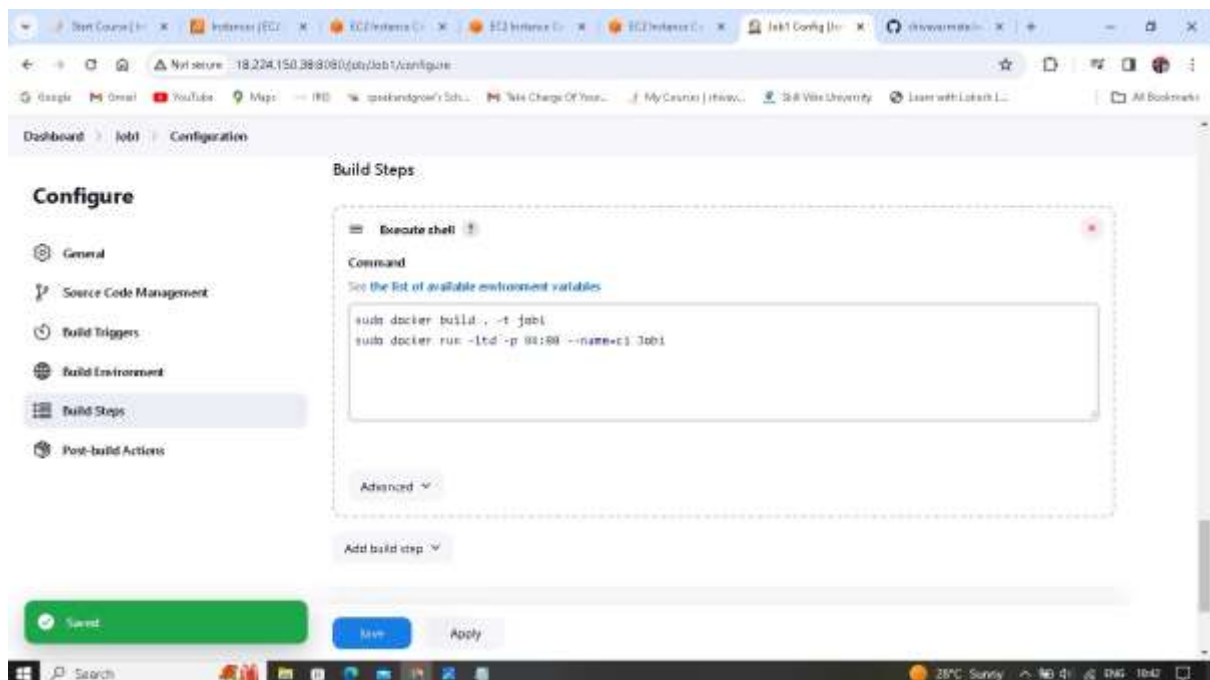
```
sudo docker build . -t job1
```

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

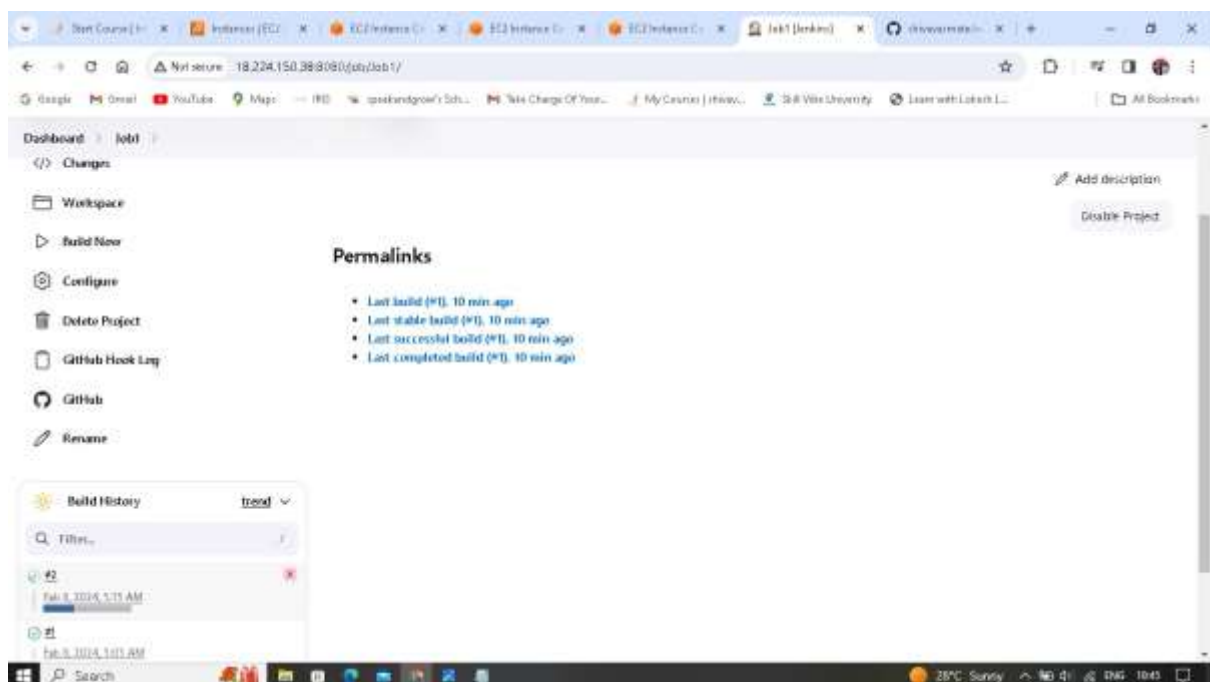




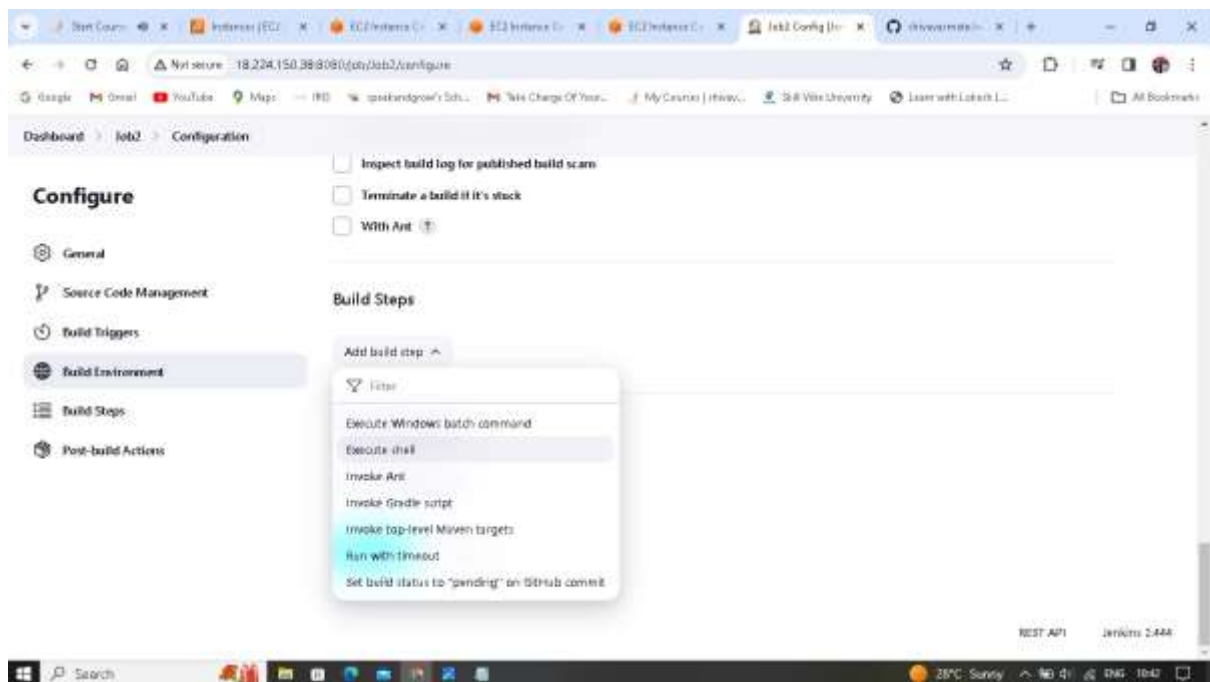
112. Click on both Apply and Save.



113. Click on Build Now



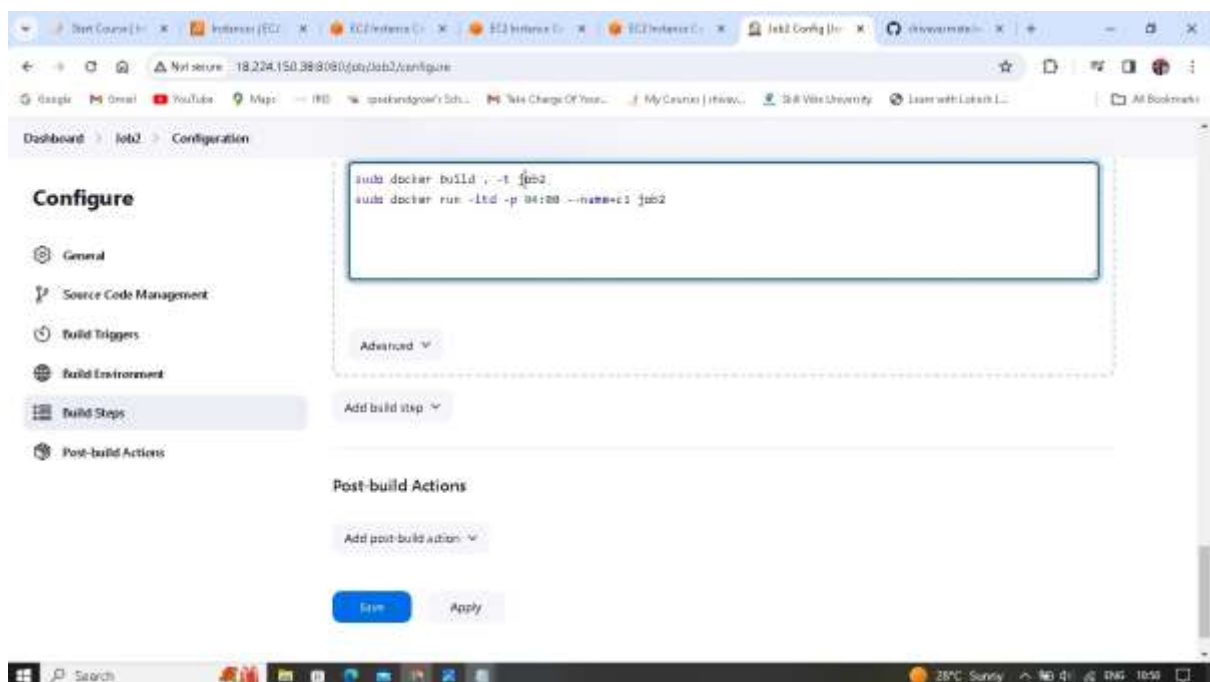
114. Similarly for Job2 go to Build Steps choose Execute shell.



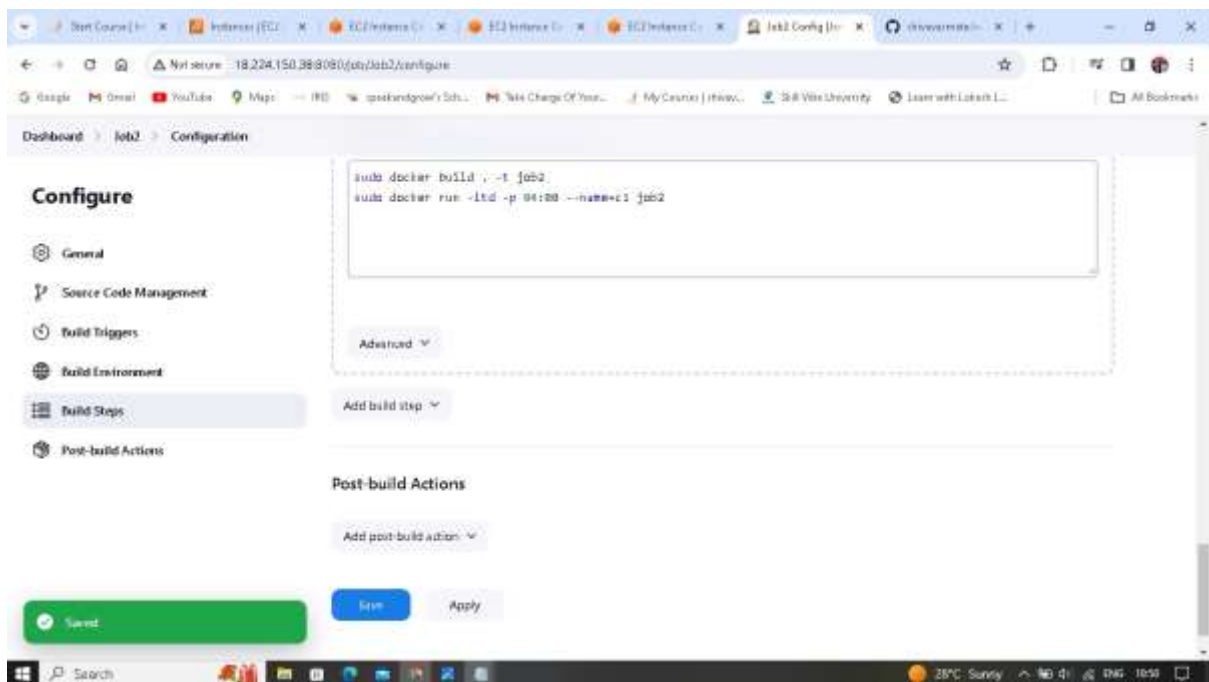
115. Under it type

```
sudo docker build . -t job2
```

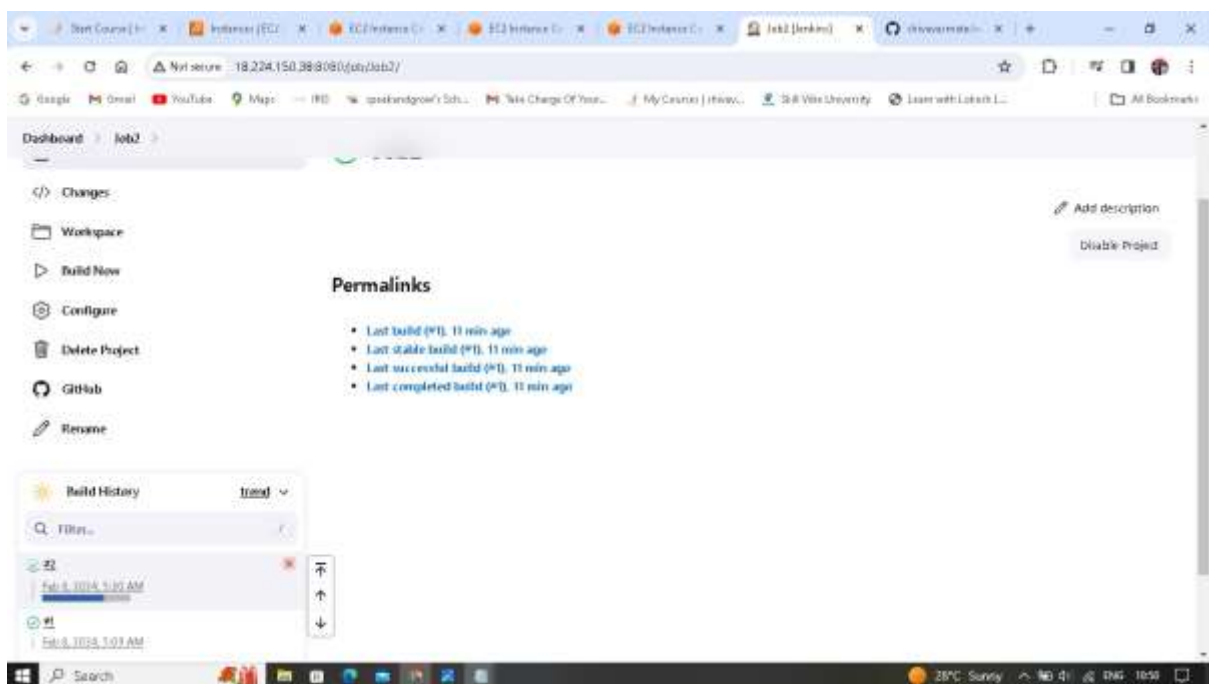
```
sudo docker run -itd -p 84:80 --name=c1 job2
```



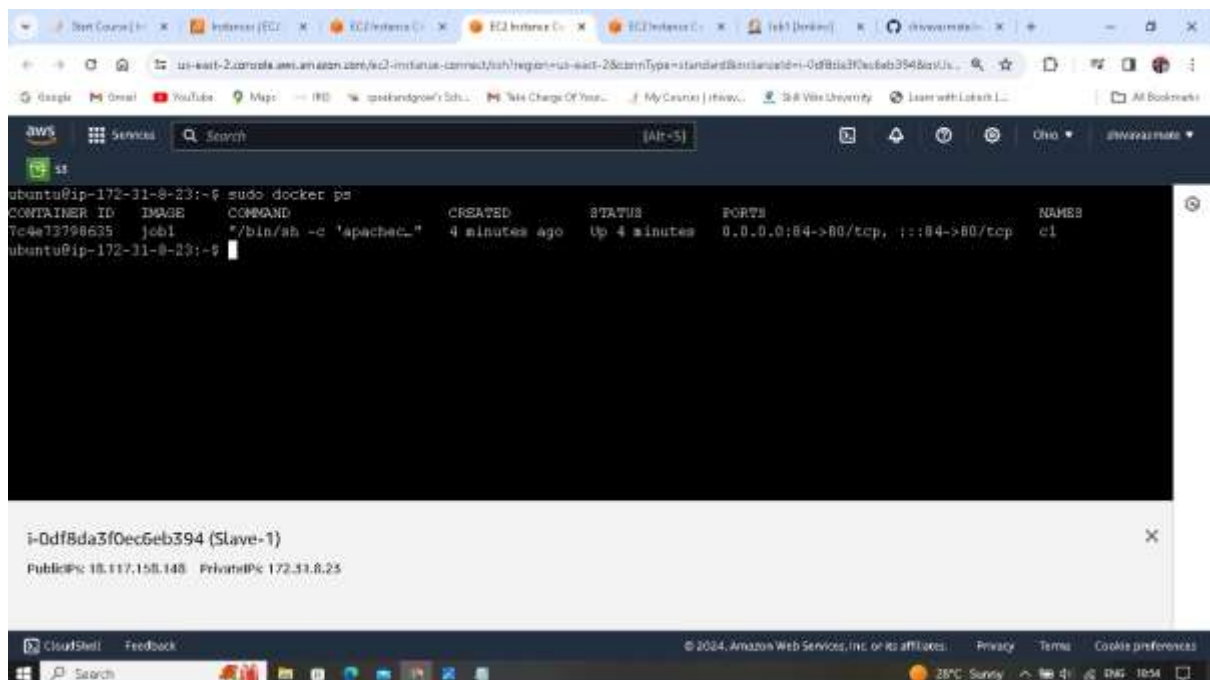
116. Click on both Apply and Save.



116. Click on Build Now



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

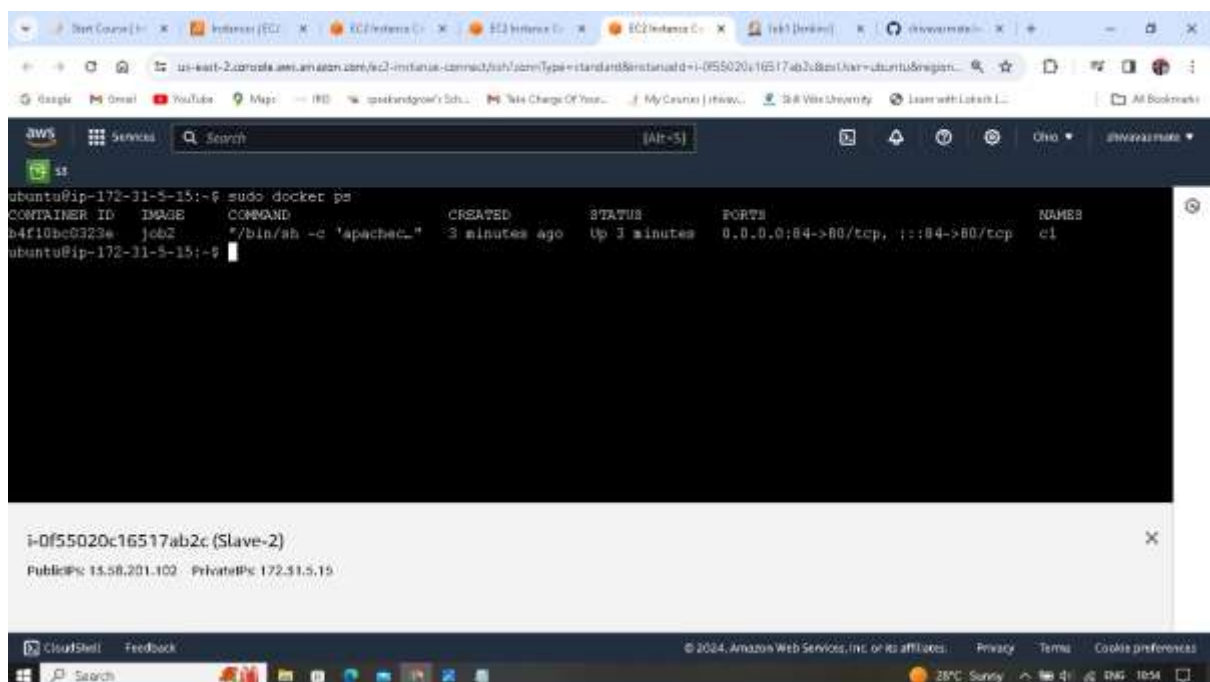


The screenshot shows the AWS CloudShell interface for an EC2 instance named 'Slave1'. The terminal window displays the command 'sudo docker ps' and its output, which is a table of running containers. Below the terminal, a metadata box for the instance 'i-0df8da3f0ec6eb394' is visible, showing its Public IP as 18.117.158.148 and Private IP as 172.31.8.25.

```
ubuntu@ip-172-31-8-23:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
7c4e73798635   job1     "/bin/sh -c 'apache2..." 4 minutes ago    Up 4 minutes    0.0.0.0:84->80/tcp, :::84->80/tcp    c1
ubuntu@ip-172-31-8-23:~$
```

i-0df8da3f0ec6eb394 (Slave-1)  
PublicIP: 18.117.158.148 PrivateIP: 172.31.8.25

118. Similarly for Slave2 machine.



The screenshot shows the AWS CloudShell interface for an EC2 instance named 'Slave2'. The terminal window displays the command 'sudo docker ps' and its output, which is a table of running containers. Below the terminal, a metadata box for the instance 'i-0f55020c16517ab2c' is visible, showing its Public IP as 15.58.201.102 and Private IP as 172.31.5.15.

```
ubuntu@ip-172-31-5-15:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
b4f10bc0323e   job2     "/bin/sh -c 'apache2..." 3 minutes ago    Up 3 minutes    0.0.0.0:84->80/tcp, :::84->80/tcp    c1
ubuntu@ip-172-31-5-15:~$
```

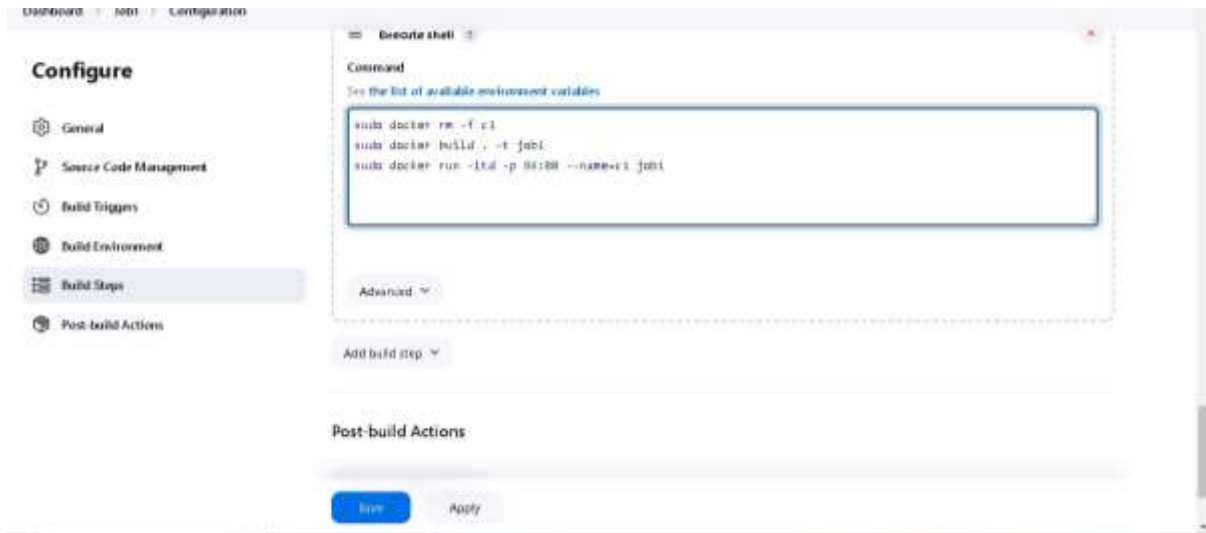
i-0f55020c16517ab2c (Slave-2)  
PublicIP: 15.58.201.102 PrivateIP: 172.31.5.15

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
```

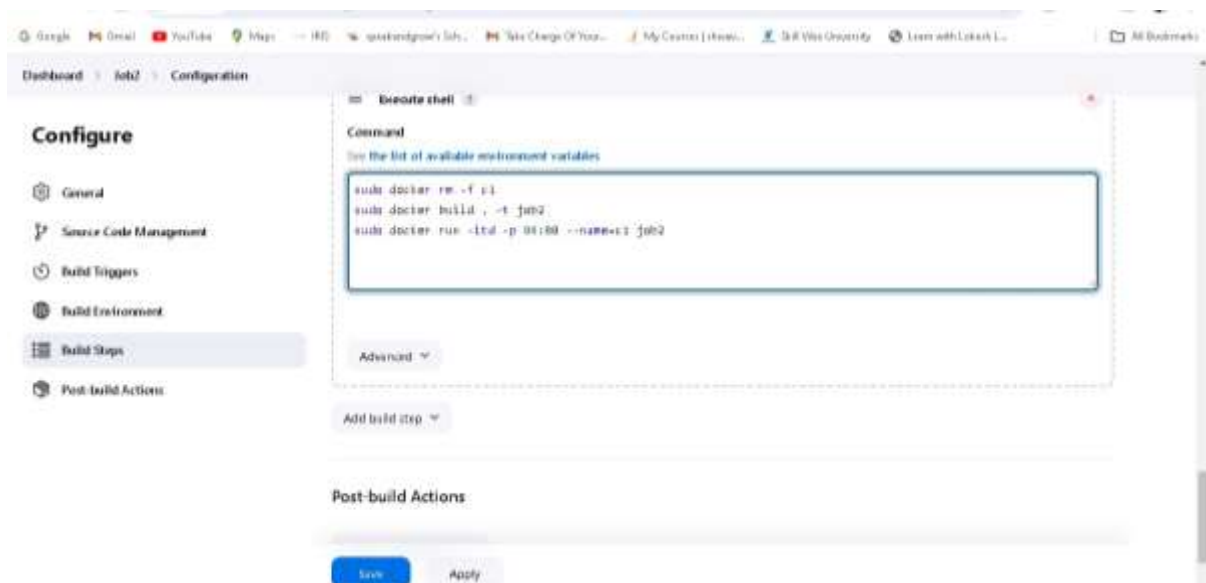


120. For Job2 edit the execute shell by

```
sudo docker rm -f c1
```

```
sudo docker build . -t job2
```

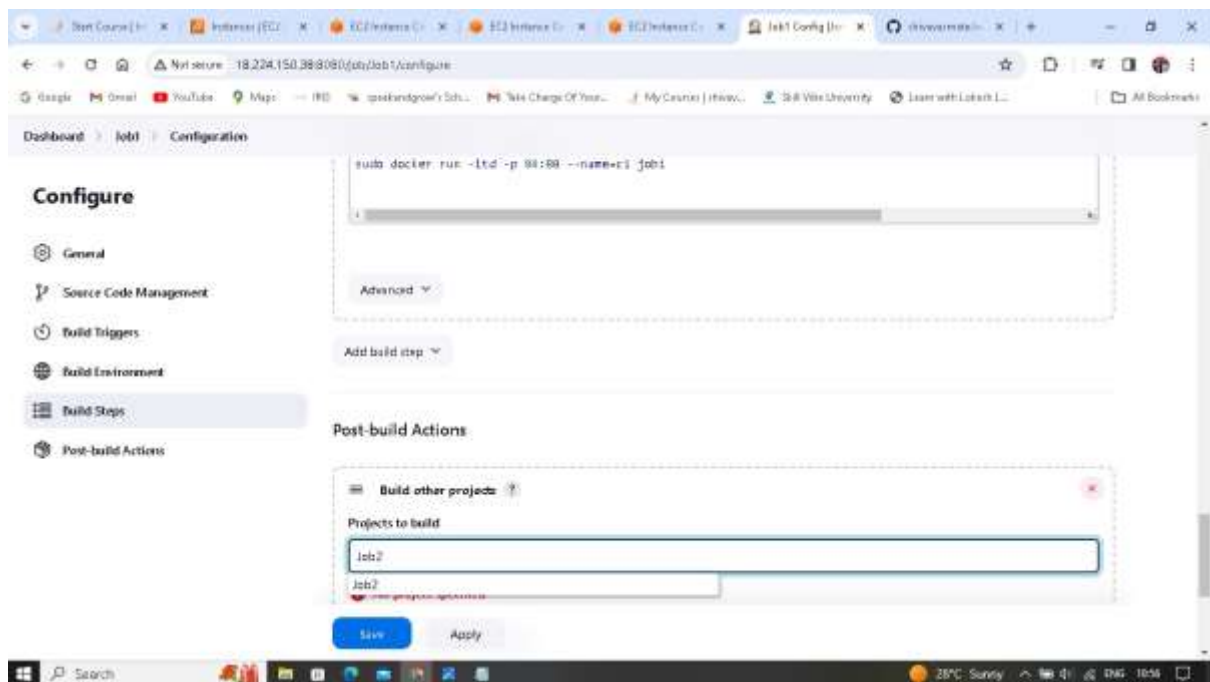
```
sudo docker run -itd -p 84:80 --name=c1 job2
```



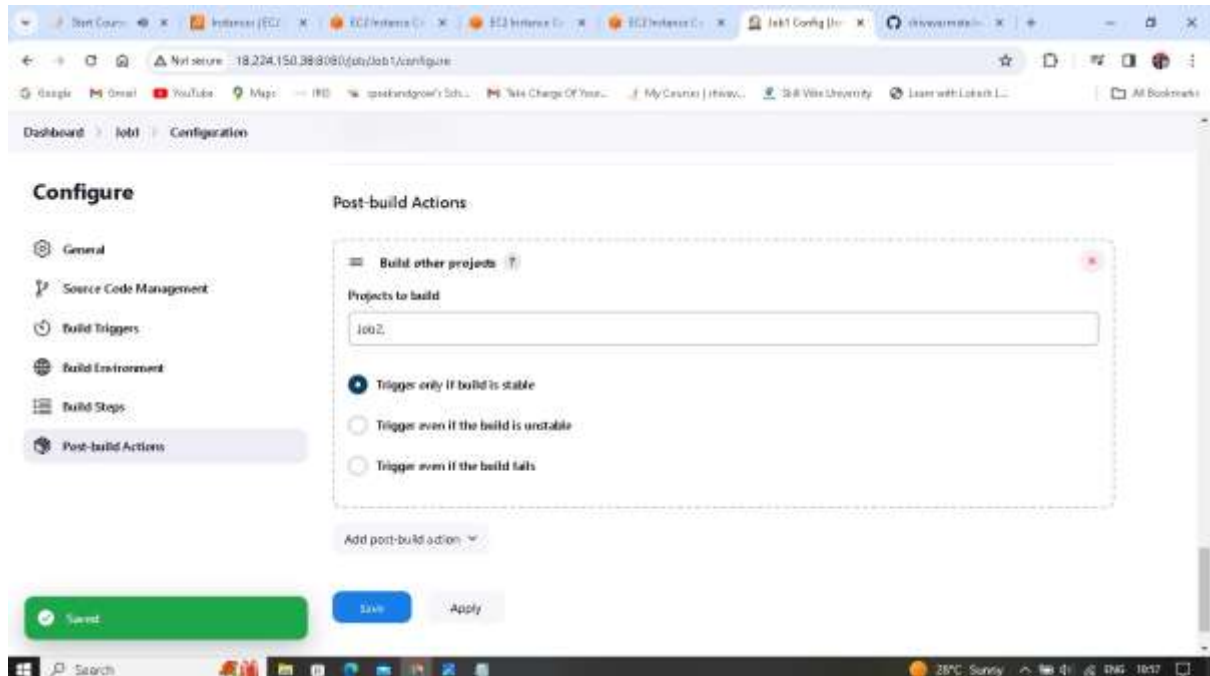
Second time it runs no error will appear. Only one docker container will run only on one port.



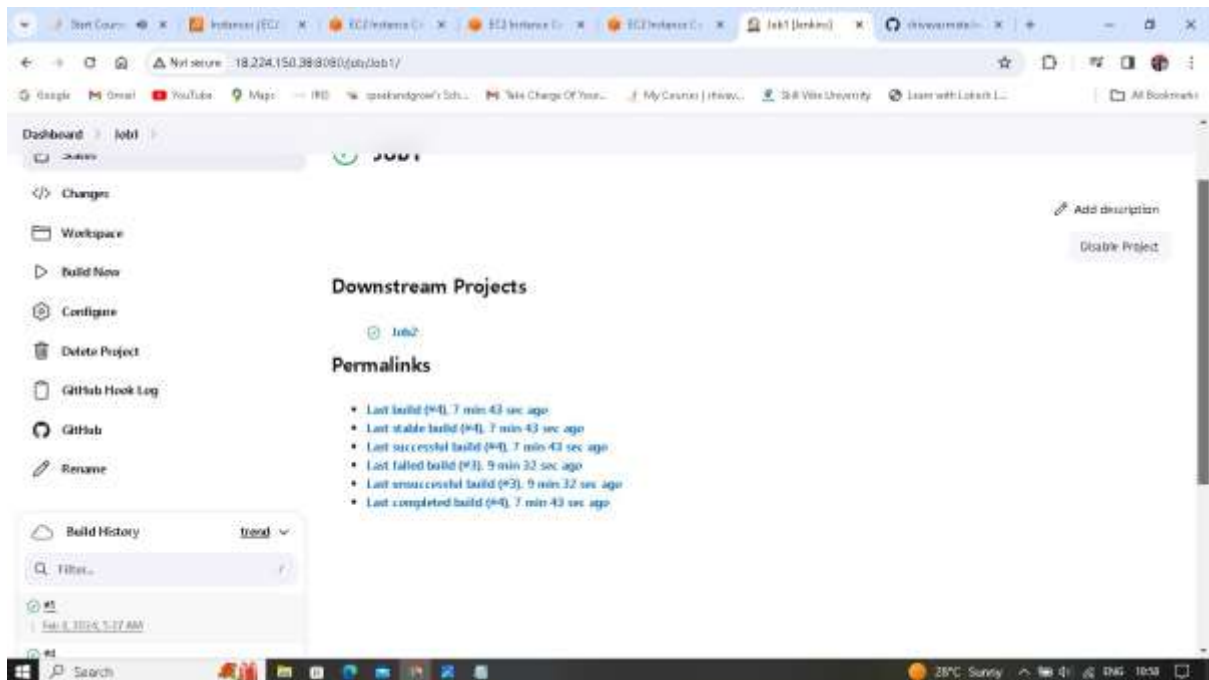
121.Go to Job1 configure under Post build Action. Give Job2 as project to build.  
such that after the job1 the job2 trigger automatically.



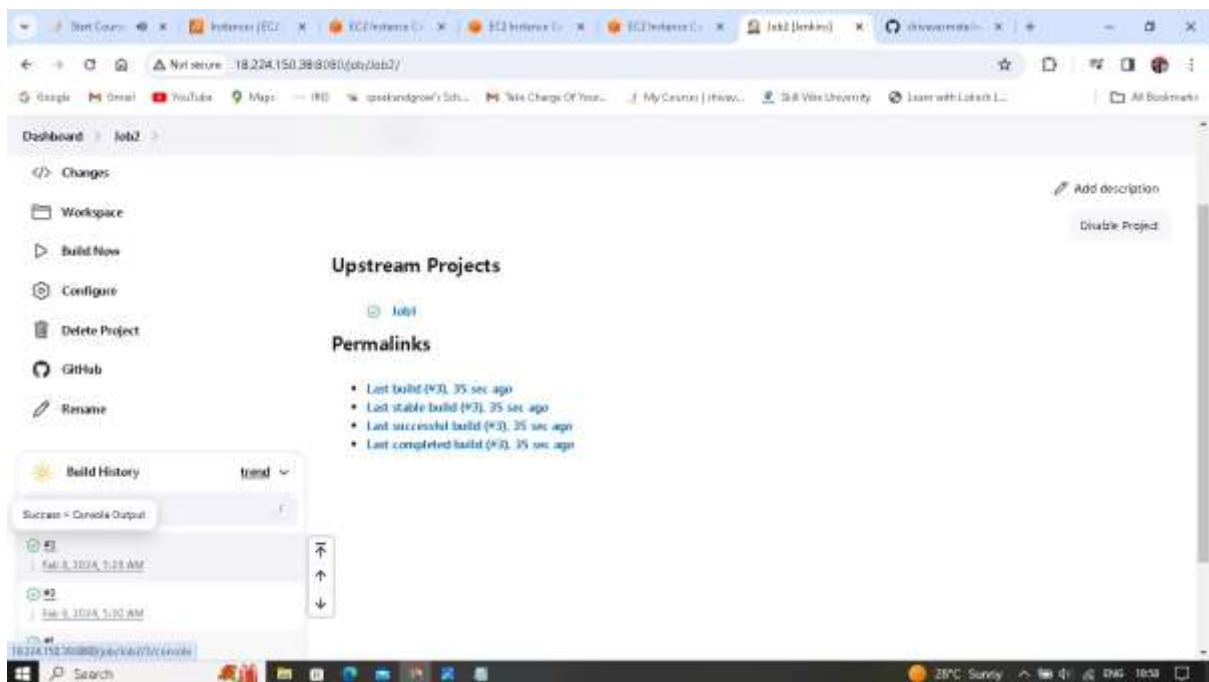
122.Click on both Apply and Save.



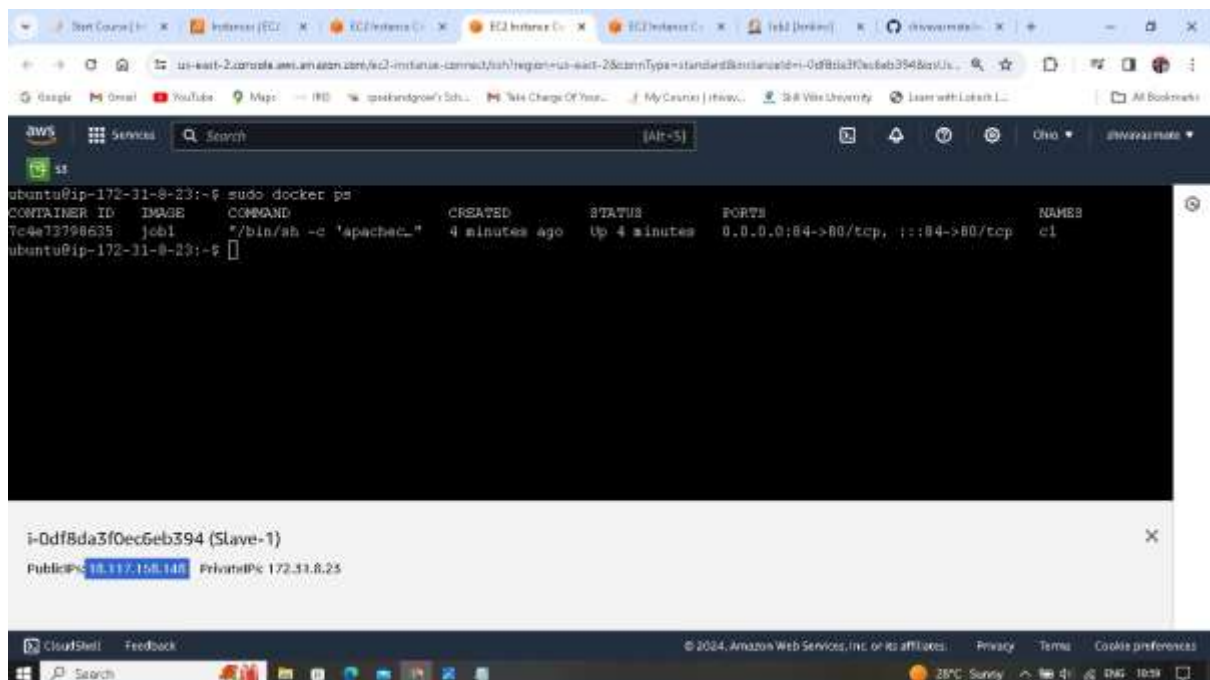
123. Click on Build Now #5 is ran successful.



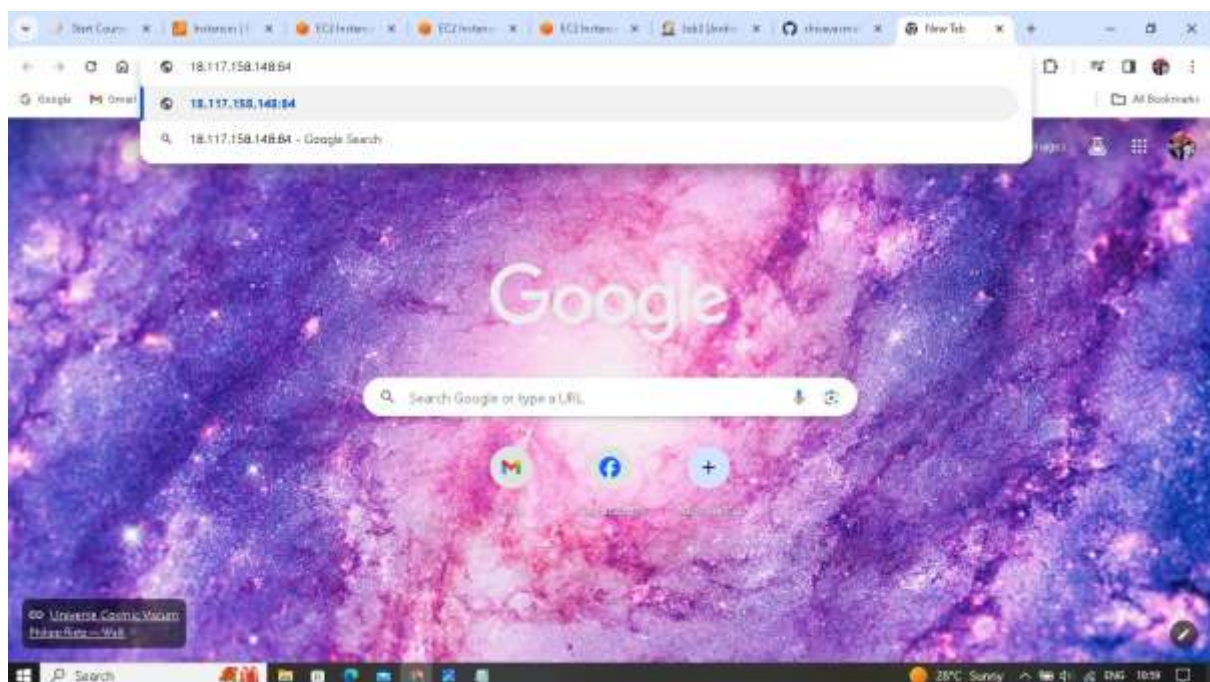
124. Job1 triggers #3 is also successful for the Job2.



125.To Verify copy the public IP of Slave1



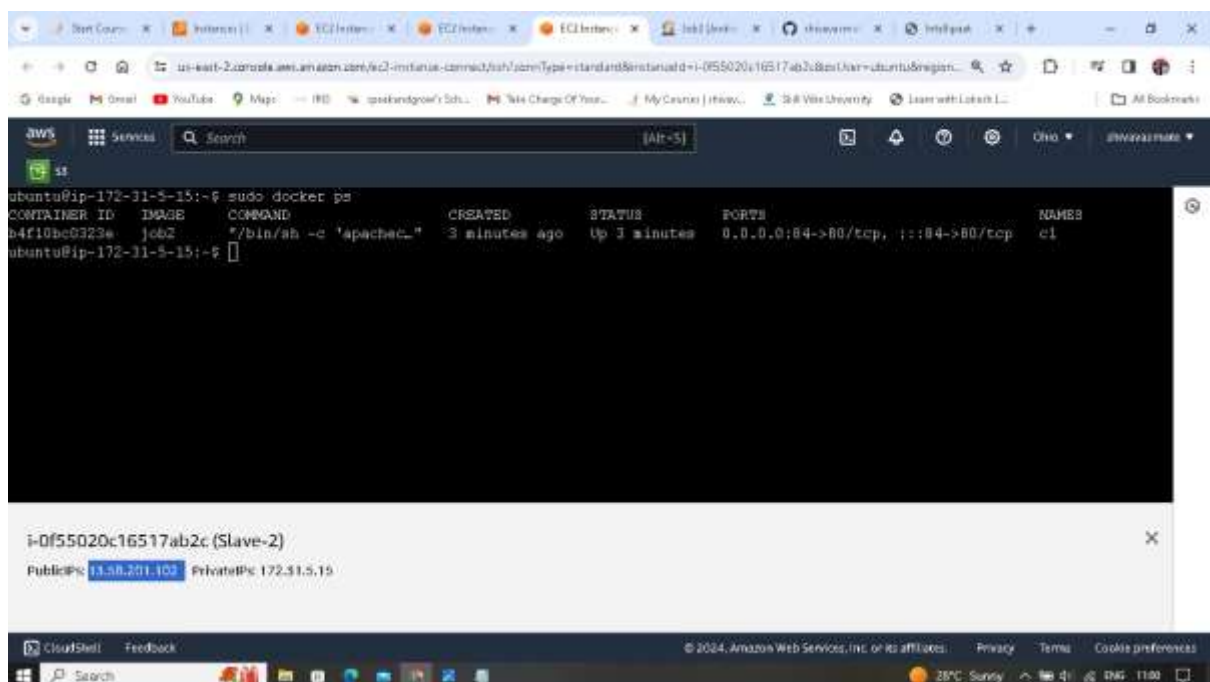
126.Paste it with port :84 in another browser.



127. Docker image is executed

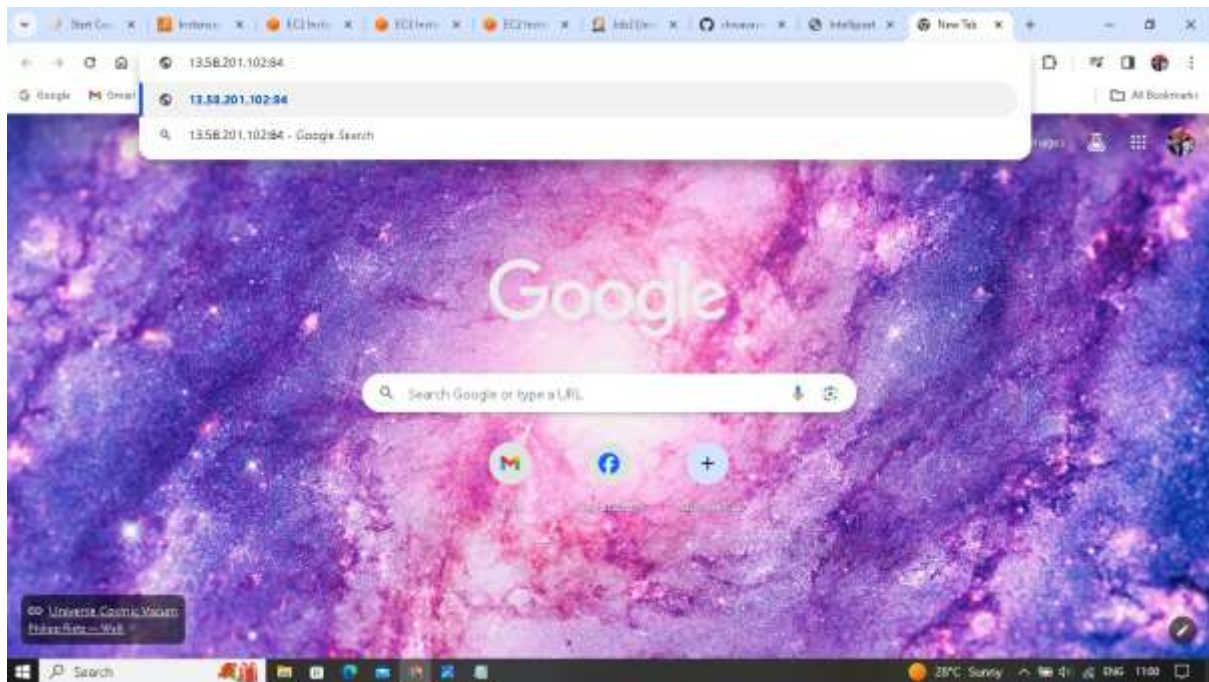


128. Similarly copy the public IP of Slave2





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



130. Docker image execute successfully.

