

PROJECT

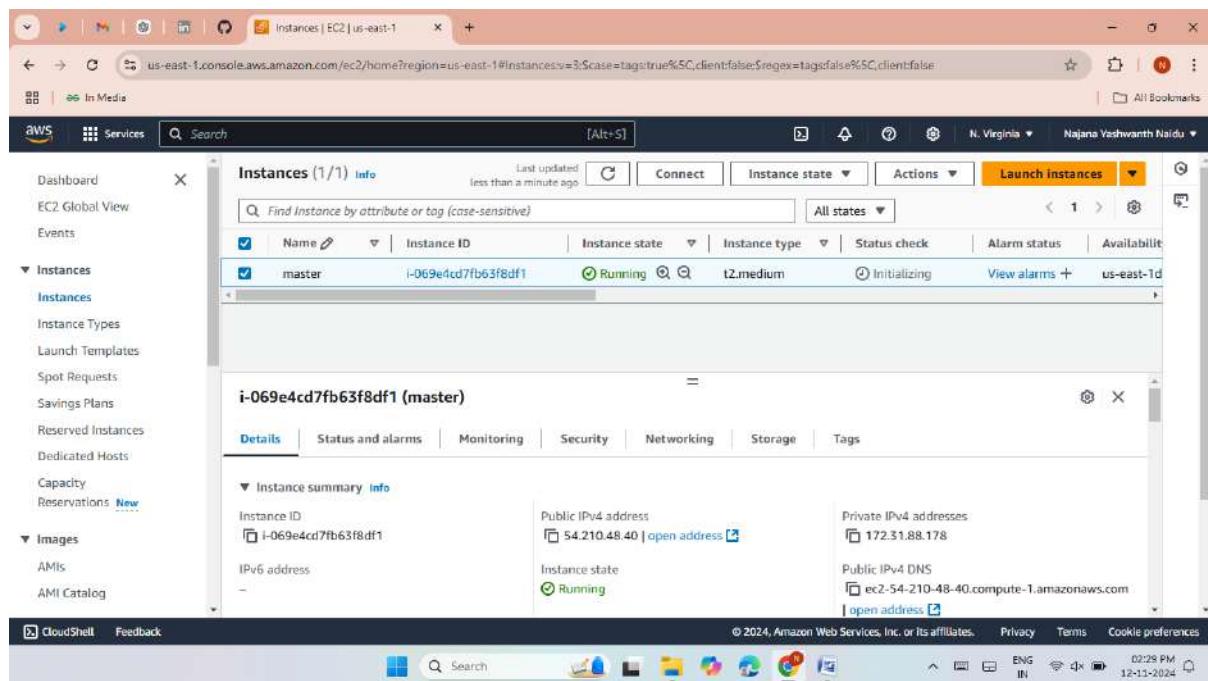
N Yashwanth Naidu

Jenkins master slave configuration

Method-1

Create one master and five slave nodes and install nginx (Slave-1), httpd (Slave-2), SonarQube (Slave-3), tomcat (Slave-4) and deploy any one Static application in any one of the web servers(httpd & nginx) (Slave-5).

- First launch an master server with the Jenkins port 8080.



- Now install java and git.
- Now install Jenkins then start and enable the Jenkins.
- We can change the hostname with the commands

Sudo hostname master

Exec bash

```

cc2-user@ip-172-31-88-178:~$ sudo systemctl start jenkins
transaction test succeeded
Running transaction
  Installing : jenkins-2.484-1.1.noarch
    Verifying : jenkins-2.484-1.1.noarch
      1/1
      1/1

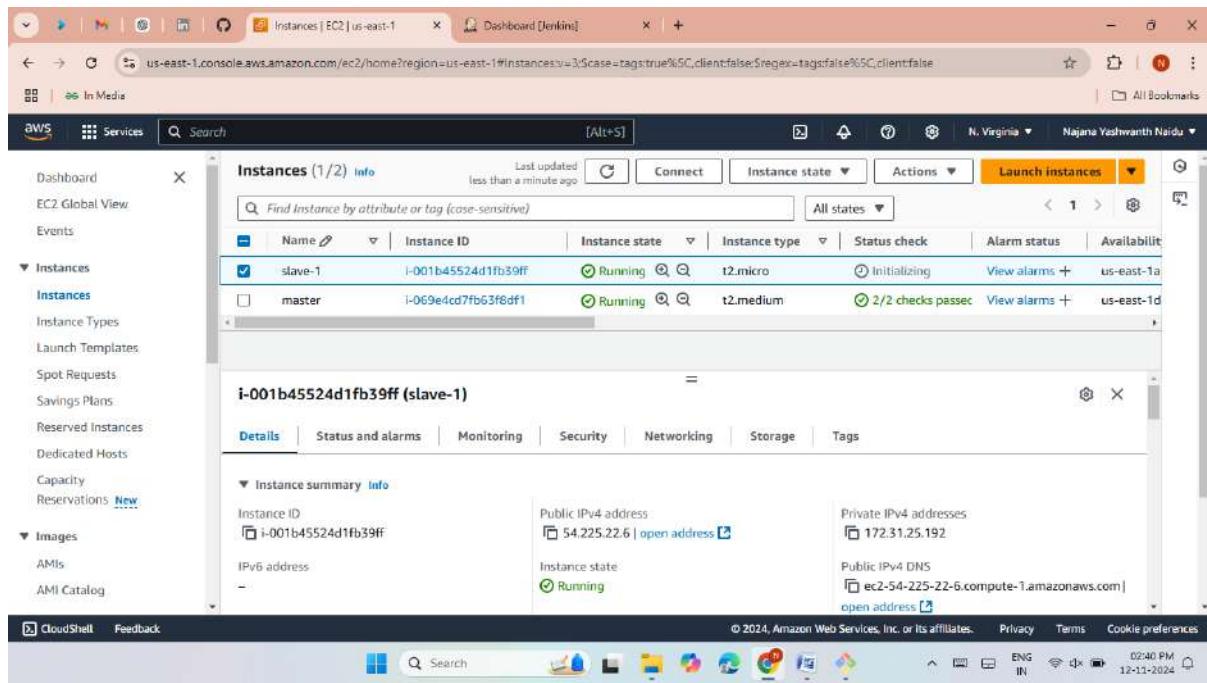
Installed:
  jenkins.noarch 0:2.484-1.1

Complete!
[ec2-user@ip-172-31-88-178 ~]$ sudo systemctl enable jenkins
Created symlink from /etc/systemd/system/multi-user.target.wants/jenkins.service to /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-88-178 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
     Active: active (running) since Tue 2024-11-12 09:01:59 UTC; 16s ago
       PID: 4182 (java)
      CGroup: /system.slice/jenkins.service
              └─ 4182 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Nov 12 09:01:56 ip-172-31-88-178.ec2.internal jenkins[4182]: *****
Nov 12 09:01:56 ip-172-31-88-178.ec2.internal jenkins[4182]: *****
Nov 12 09:01:56 ip-172-31-88-178.ec2.internal jenkins[4182]: *****
Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: 2024-11-12 09:01:59.649+0000 [id=32]      INFO      jenkins.I...
Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: 2024-11-12 09:01:59.679+0000 [id=23]      INFO      hudson.li...
Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: Started Jenkins Continuous Integration Server.
Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: 2024-11-12 09:01:59.709+0000 [id=49]      INFO      h.m.Downl...
Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: 2024-11-12 09:01:59.706+0000 [id=49]      INFO      hudson.ut...
Nov 12 09:02:09 ip-172-31-88-178.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] unknown Tvalue 'StartLi...
Nov 12 09:02:09 ip-172-31-88-178.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] Unknown Tvalue 'StartLi...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-88-178 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c6898f2de2e74fa68a51e3fg627cfef931
[ec2-user@ip-172-31-88-178 ~]$ sudo visudo
[ec2-user@ip-172-31-88-178 ~]$ sudo systemctl restart jenkins
[ec2-user@ip-172-31-88-178 ~]$ sudo hostname master
[ec2-user@ip-172-31-88-178 ~]$ exec bash
[ec2-user@master ~]$
```

➤ Here the Jenkins was launched successfully.

- Now launch an EC2 Instances and give name as slave1.



- Now connect to the Master server and then generate keys with the command.

ssh-keygen

```
ec2-user@ip-172-31-88-178:~$ Nov 12 09:01:59 ip-172-31-88-178.ec2.internal jenkins[4182]: 2024-11-12 09:01:59.706+0000 [id=49] INFO hudson.util...pt #1 Nov 12 09:02:09 ip-172-31-88-178.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] Unknown lvalue 'StartL...Unit' Nov 12 09:02:09 ip-172-31-88-178.ec2.internal systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] Unknown lvalue 'StartL...Unit' Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-88-178 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c6898f2de2e74fa68a51e3f627cf931
[ec2-user@ip-172-31-88-178 ~]$ sudo visudo
[ec2-user@ip-172-31-88-178 ~]$ sudo systemctl restart jenkins
[ec2-user@ip-172-31-88-178 ~]$ sudo hostname master
[ec2-user@ip-172-31-88-178 ~]$ exec bash
[ec2-user@master ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9/PKbbxNC+ETsurXXoy5A1rpvRLkrmVIK+U2PkJyW ec2-user@master
The key's randomart image is:
----[RSA 2048]----+
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
----[SHA256]----+
[ec2-user@master ~]$ ls
[ec2-user@master ~]$ .ssh/
bash: .ssh/: Is a directory
[ec2-user@master ~]$ cd .ssh/
[ec2-user@master .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub
[ec2-user@master .ssh]$ sudo cat id_rsa.pub
```

- Now connect Slave1 then install java.
- Now give private keys from master server.

```

Verifying : libxslt-1.1.28-6.amzn2.x86_64
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64
Verifying : python-javapackages-3.4.1-11.amzn2.noarch
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64
Verifying : alsa-lib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

Complete!
[ec2-user@ip-172-31-25-192 ~]$ sudo hostname slave1
[ec2-user@ip-172-31-25-192 ~]$ exec bash
[ec2-user@slave1 ~]$ cd .ssh/
[ec2-user@slave1 .ssh]$ ls
authorized_keys
[ec2-user@slave1 .ssh]$ sudo vi authorized_keys |

```

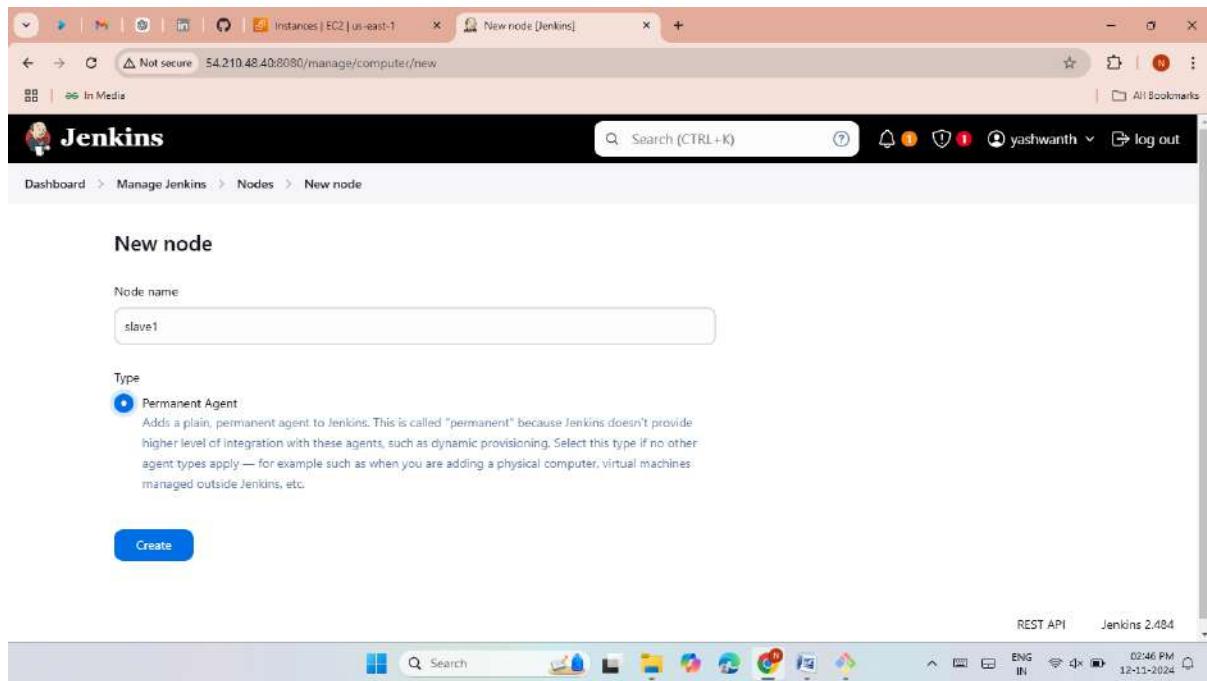
02:43 PM
12-11-2024

- Here the default nodes are there in the Jenkins.

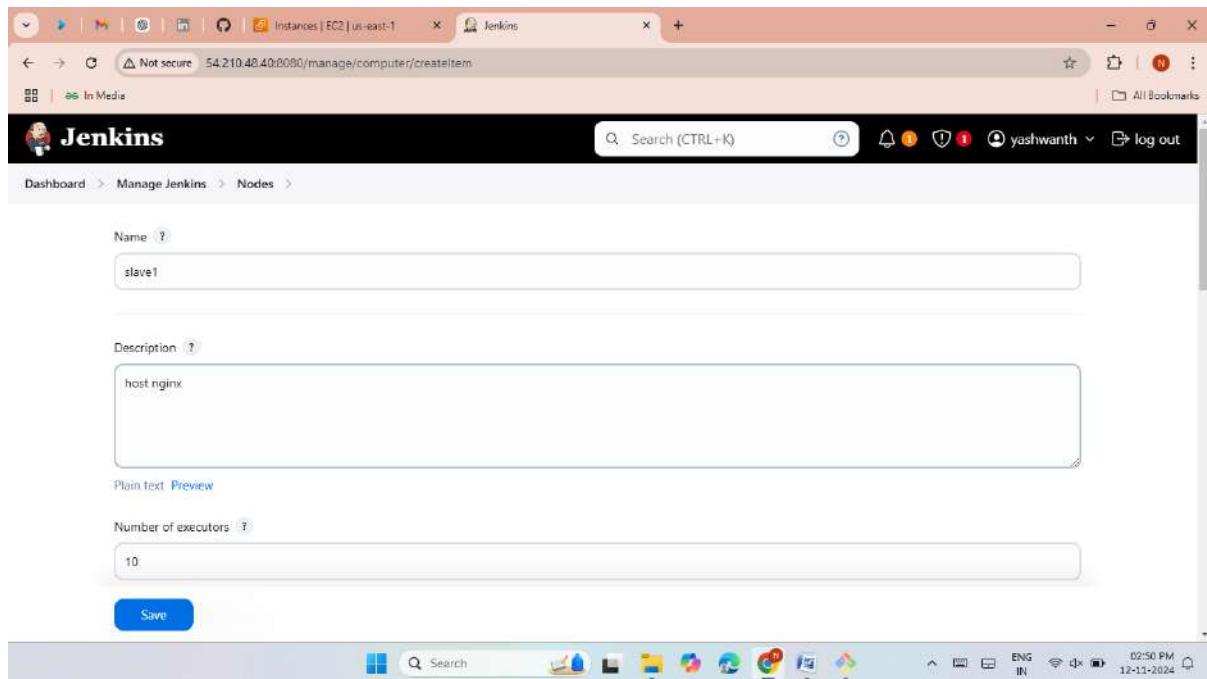
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.02 GiB	0 B	5.02 GiB	0ms
	Data obtained	9 min 26 sec	9 min 26 sec	9 min 26 sec	9 min 26 sec	9 min 26 sec	9 min 26 sec

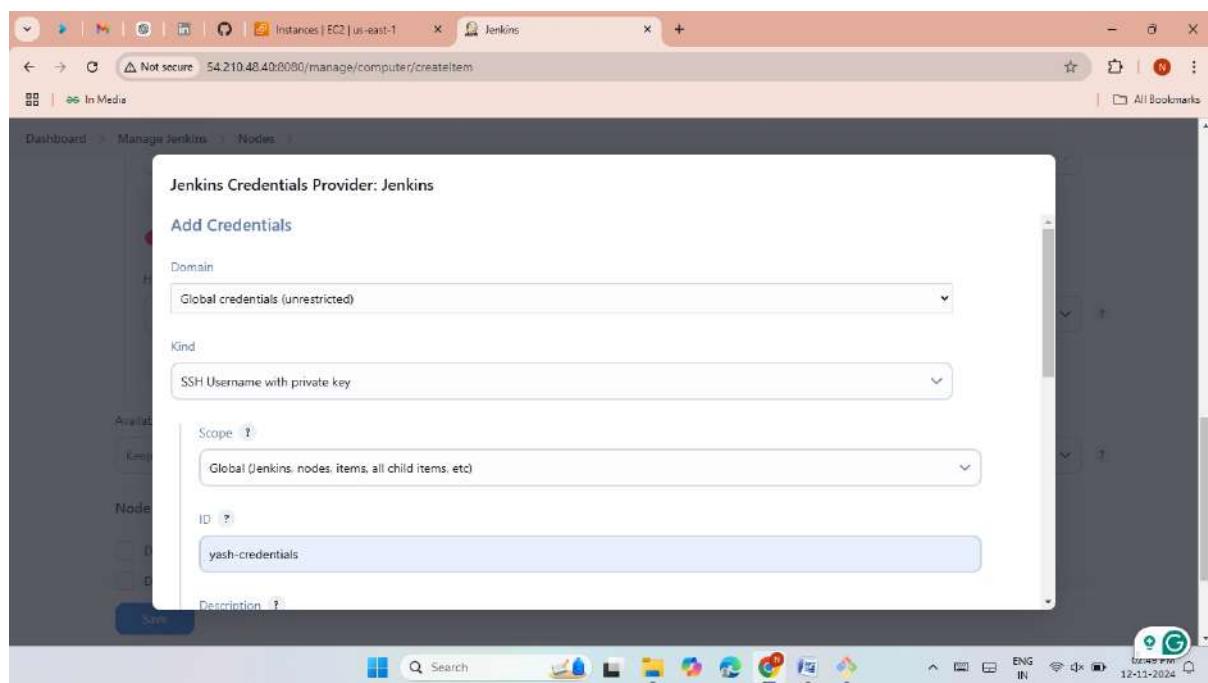
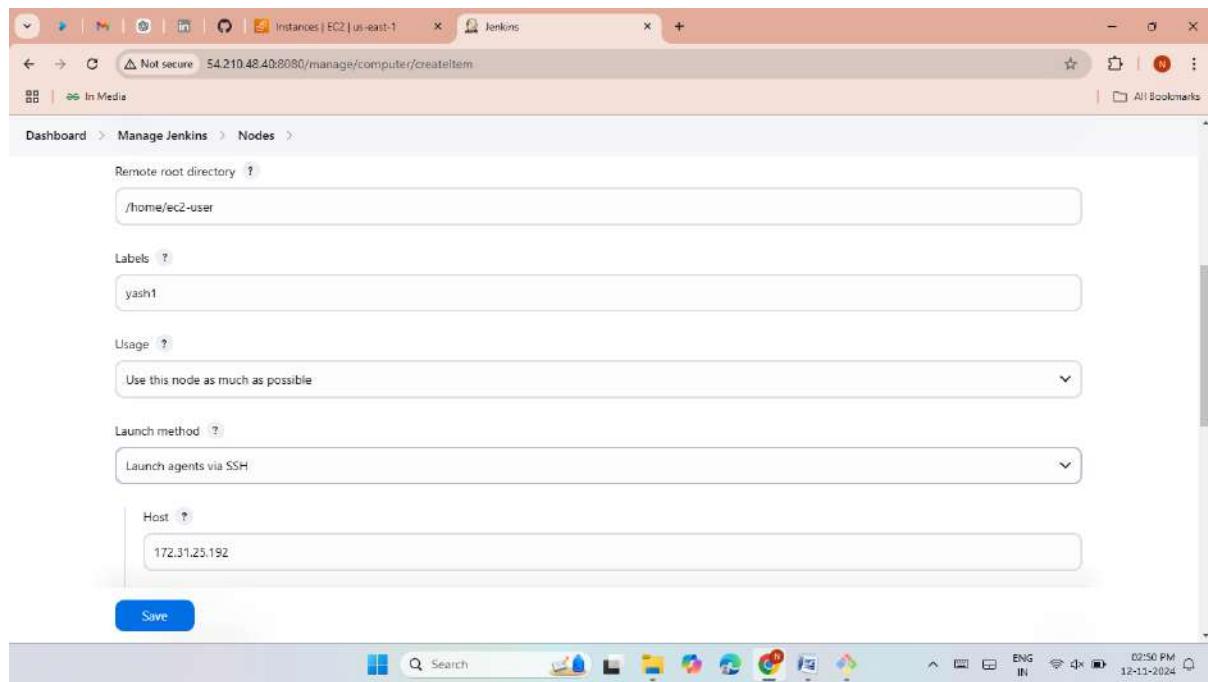
REST API Jenkins 2.484
02:45 PM
12-11-2024

- Now create a node for slave1 server.

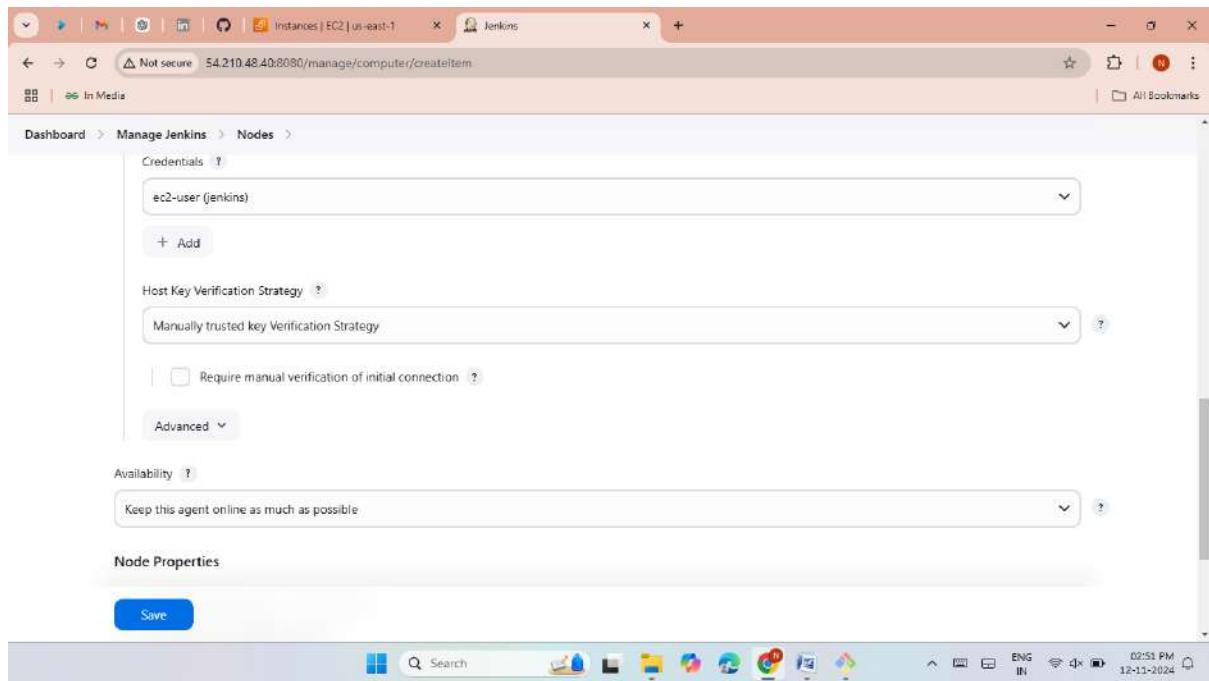


- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.





➤ Now click on save.



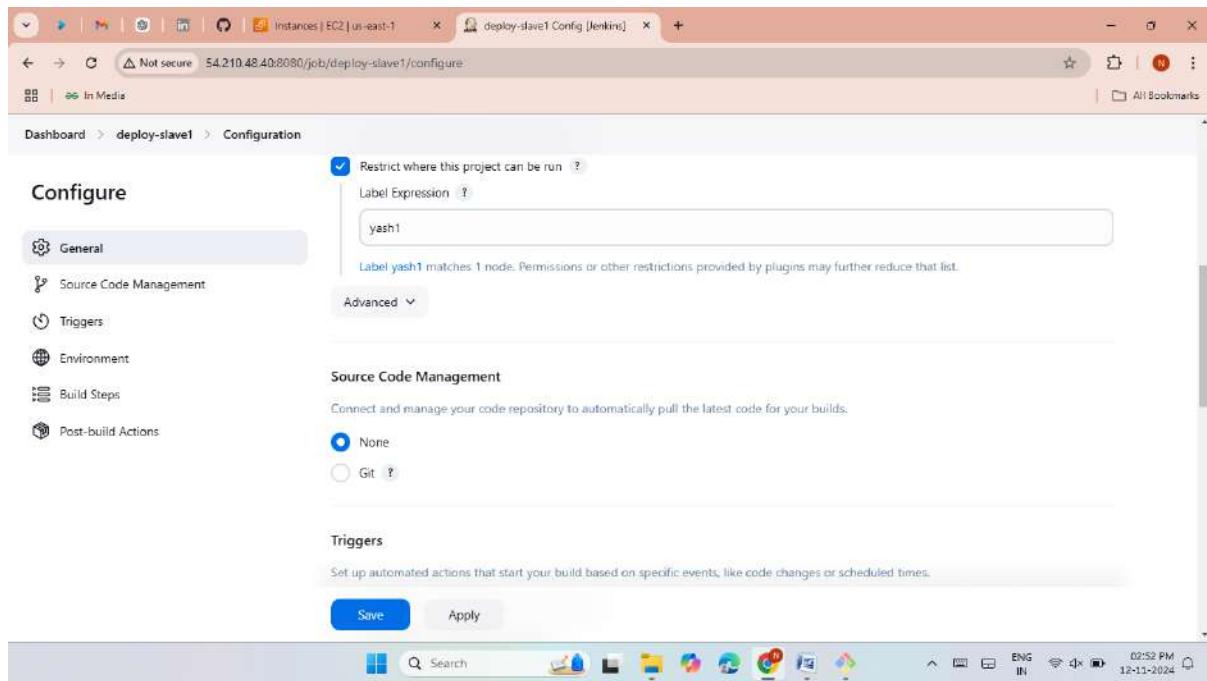
➤ Here the slave1 node is successfully created.

The screenshot shows the Jenkins 'Nodes' overview page. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node 0/2, slave1 0/10). The main area displays a table of nodes:

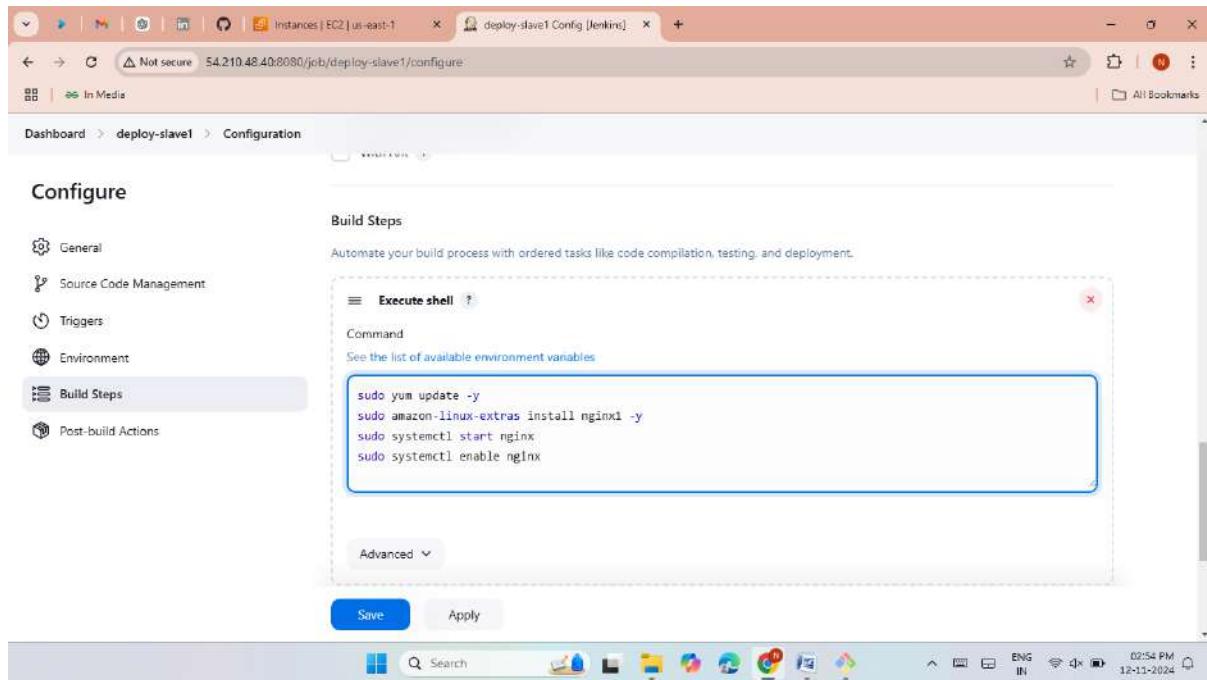
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	Built-In Node	Linux (amd64)	In sync	5.02 GiB	0 B	5.02 GiB	0ms
2	slave1	Linux (amd64)	In sync	5.41 GiB	0 B	5.41 GiB	52ms

At the bottom, there are links for 'REST API' and 'Jenkins 2.484'. The browser address bar shows 'Not secure 54.210.48.40:8080/manage/computer/_.'

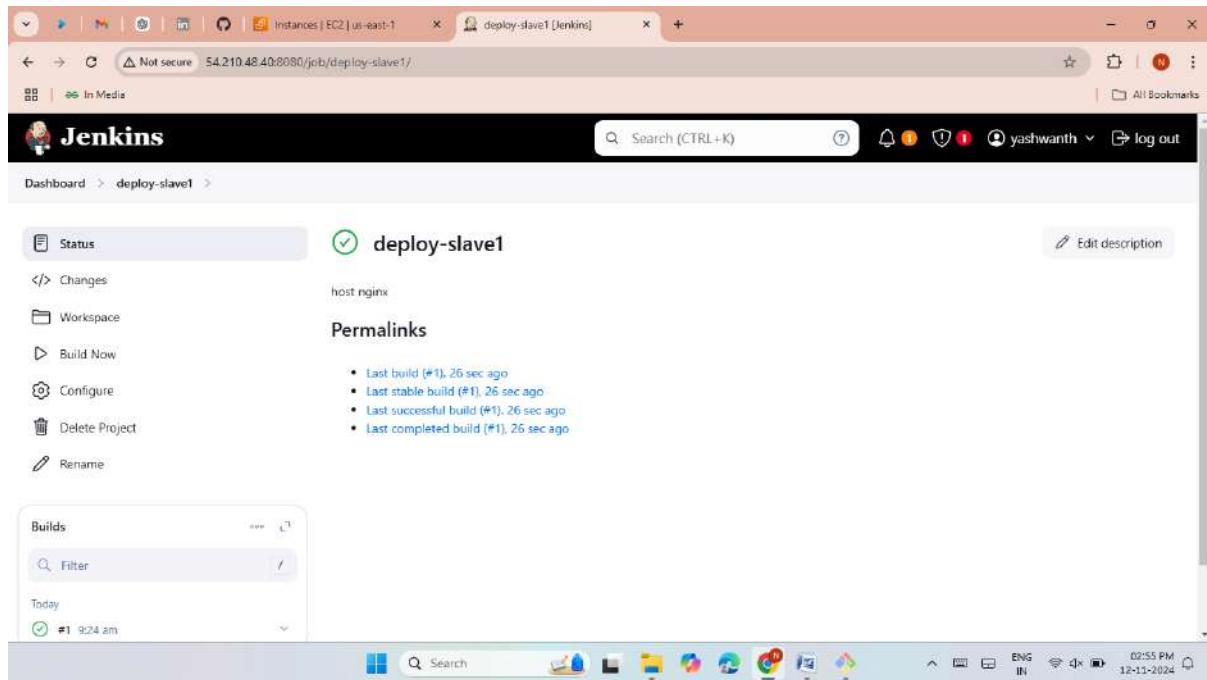
- Now create a job and give slave1 label.



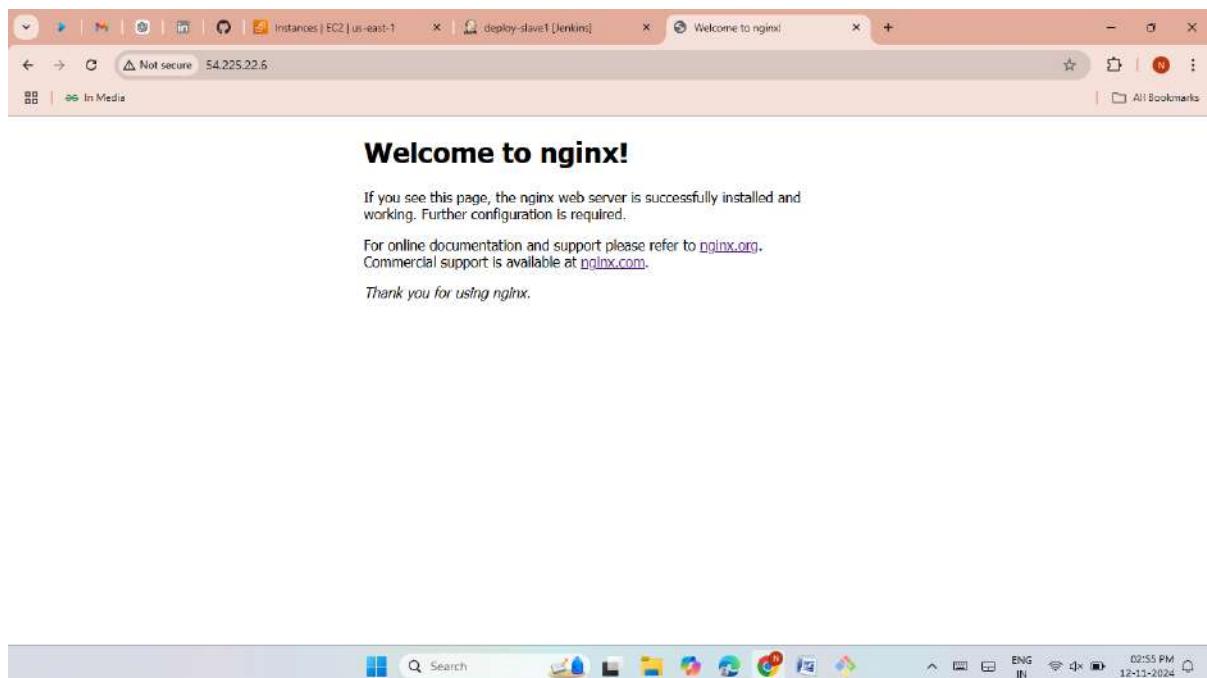
- Now install nginx and then start and enable.
- Now Click on save.



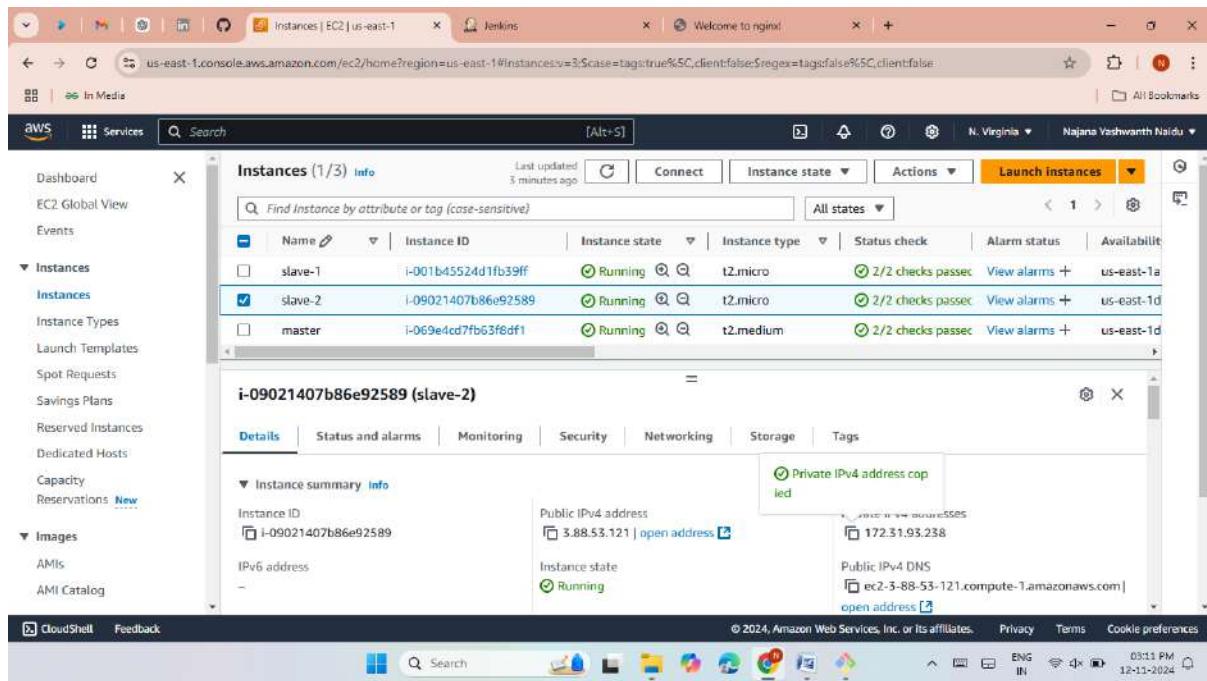
- Now click on build now.
- Here the job was success.



- Now copy the slave1 Public IP and past it on Google browser.
- The nginx web server was hosted successfully.



- Now launch an EC2 instances and give name as slave2.



- Now connect Slave2 then install java.
 - Now give private keys from master server.

```
[ec2-user@ip-172-31-93-238 ~]$ Read from remote host ec2-3-88-53-121.compute-1.amazonaws.com: Connection reset by peer
Connection to ec2-3-88-53-121.compute-1.amazonaws.com closed.
client_loop: send disconnect: Connection reset by peer

[yaswa@LAPTOP-GM3QJH0 MINGW64 ~/keys]
$ ssh -i "yash.pem" ec2-user@ec2-3-88-53-121.compute-1.amazonaws.com
Last login: Tue Nov 12 09:27:56 2024 from 49.43.203.141
  _#_
 /###\      Amazon Linux 2
 \##\`      AL2 End of Life is 2025-06-30.
  #/ \_     A newer version of Amazon Linux is available!
  / \_ /    Amazon Linux 2023, GA and supported until 2028-03-15.
  / \_ /    https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-93-238 ~]$ sudo yum -y install java-17*
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                         | 3.6 kB  00:00:00
Package 1:java-17-amazon-corretto-javadoc-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Package 1:java-17-amazon-corretto-debugsymbols-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Package 1:java-17-amazon-corretto-headless-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Package 1:java-17-amazon-corretto-devel-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Package 1:java-17-amazon-corretto-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Package 1:java-17-amazon-corretto-jmods-17.0.13+11-1.amzn2.1.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-93-238 ~]$ cd .ssh/
[ec2-user@ip-172-31-93-238 .ssh]$ ls
authorized_keys
[ec2-user@ip-172-31-93-238 .ssh]$ sudo vi authorized_keys
[ec2-user@ip-172-31-93-238 .ssh]$ cd
[ec2-user@ip-172-31-93-238 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-93-238 ~]$
```

- Now create a node for slave2 server.
- Give details for slave node.
- Here we have to give slave2 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.

Instances | EC2 | ui-east-1 Jenkins Welcome to nginx! Dashboard > Manage Jenkins > Nodes >

Name ? slave2

Description ? host httpd

Plain text Preview

Number of executors ? 10

Save

Instances | EC2 | ui-east-1 Jenkins Welcome to nginx! Dashboard > Manage Jenkins > Nodes >

Labels ? yash2

Usage ? Use this node as much as possible

Launch method ? Launch agents via SSH

Host ? 172.31.93.238

Credentials ? ec2-user (jenkins)

+ Add

Save

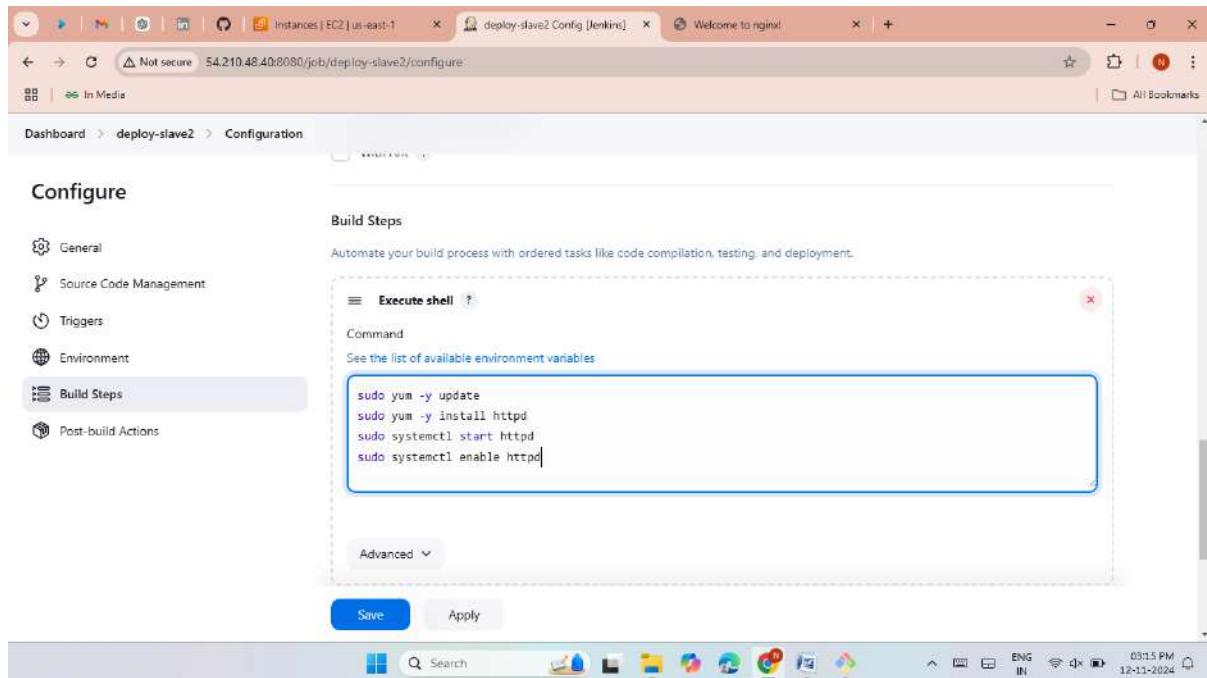
- Here the slave2 node is successfully created.

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (listing 'Built-In Node', 'slave1', and 'slave2' with counts of 0/2, 0/10, and 0/10 respectively). The main area is titled 'Nodes' and contains a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, Response Time, and Data obtained. The table shows three rows: 'Built-In Node' (Linux (amd64), In sync, 5.02 GiB, 0.8, 5.02 GiB, 0ms), 'slave1' (Linux (amd64), In sync, 5.39 GiB, 0.8, 5.39 GiB, 24ms), and 'slave2' (Linux (amd64), In sync, 5.41 GiB, 0.8, 5.41 GiB, 62ms). A legend at the bottom indicates that the 'Data obtained' column values range from 0.33 sec to 0.28 sec. The status bar at the bottom right shows the date and time as 03:13 PM 12-11-2024.

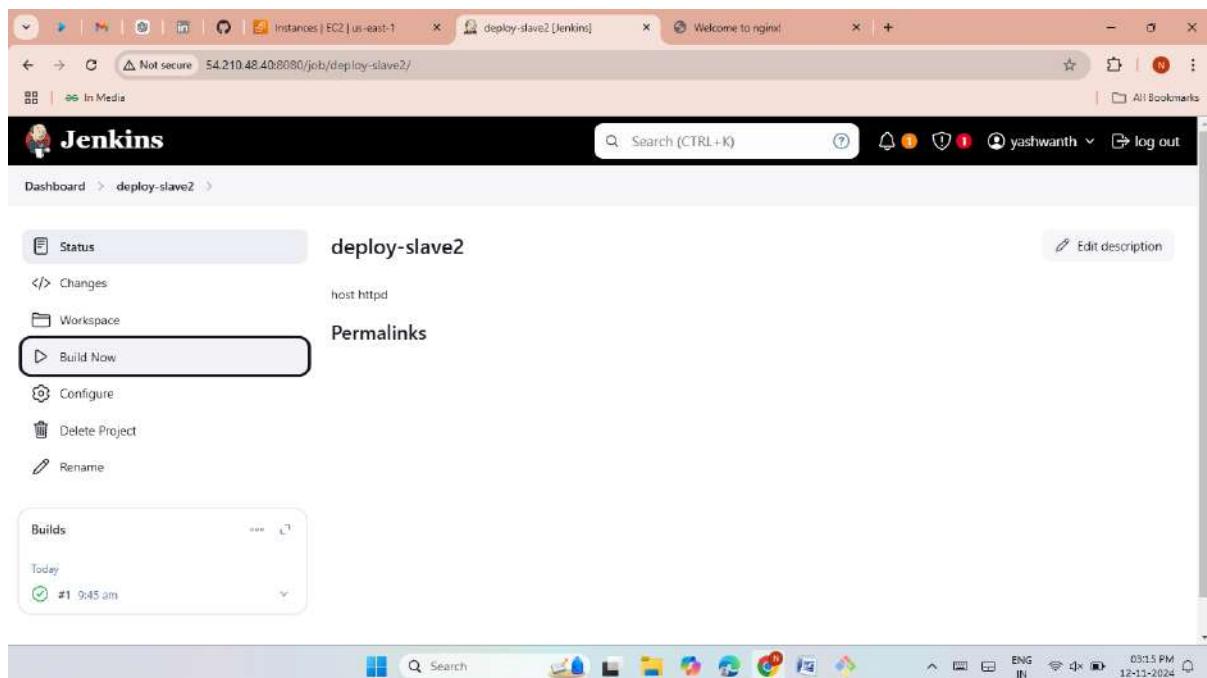
- Now create a job and give slave2 label.

The screenshot shows the Jenkins 'Configuration' page for a job named 'deploy-slave2'. The 'General' section is selected. Under 'Label Expression', the value 'yash2' is entered. A note below states: 'Label yash2 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' Other tabs visible include 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. At the bottom are 'Save' and 'Apply' buttons. The status bar at the bottom right shows the date and time as 03:13 PM 12-11-2024.

- Now install httpd and then start and enable.
- Now Click on save.



- Now click on build now.
- Here the job was success.



- Now copy the slave2 public ip and past it on google browser.
- The httpd web server was hosted successfully.



This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

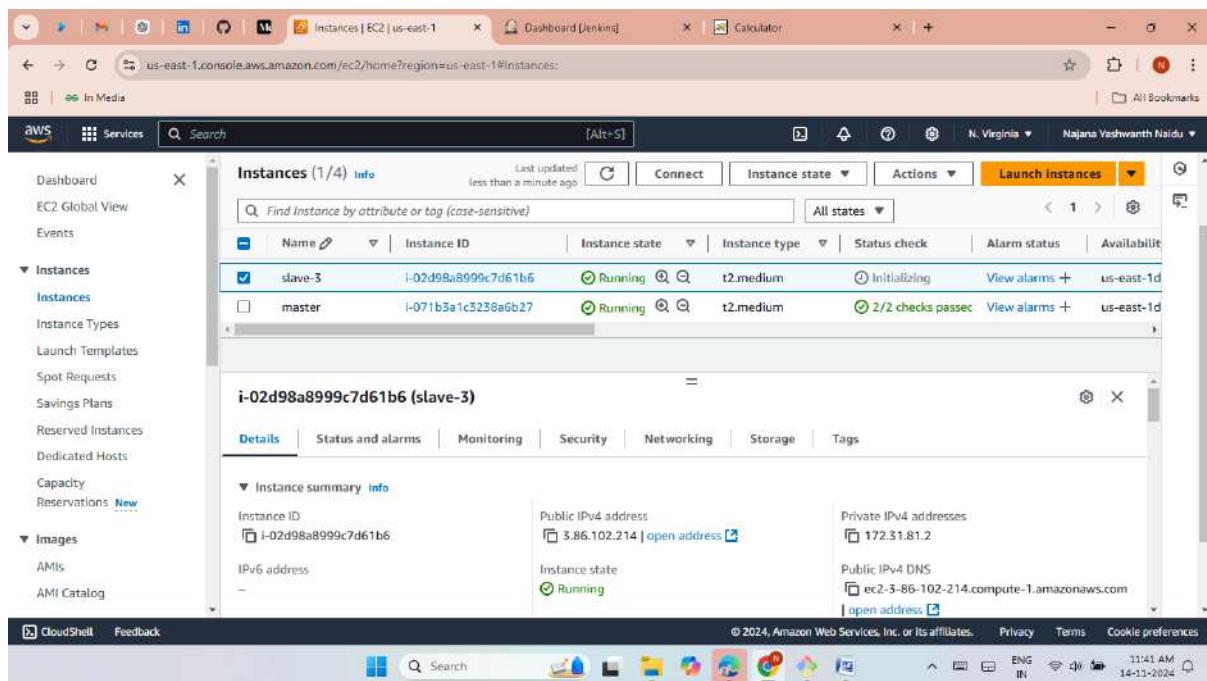
If you are the website administrator:

You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache HTTP Server:



- Now launch an EC2 instances and give name as slave3.



- Now connect Slave2 then install java.
 - Now give private keys from master server.

```
ec2-user@ip-172-31-81-2:~$ Verifying :alsa-lib-1.1.4.1-2.amzn2.x86_64 29/32
Verifying :fontpackages-filesystem-1.44-8.amzn2.noarch 30/32
Verifying :libICE-1.0.9-9.amzn2.0.x86_64 31/32
Verifying :javapackages-tools-3.4.1-11.amzn2.noarch 32/32

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

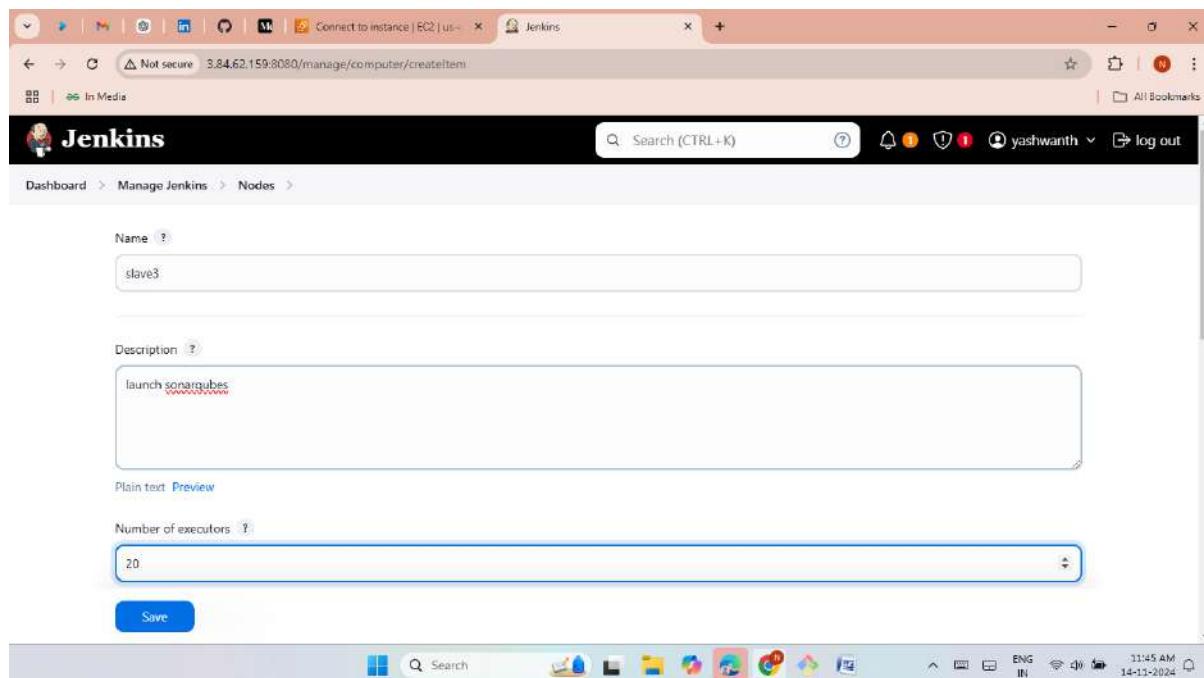
Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXiXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

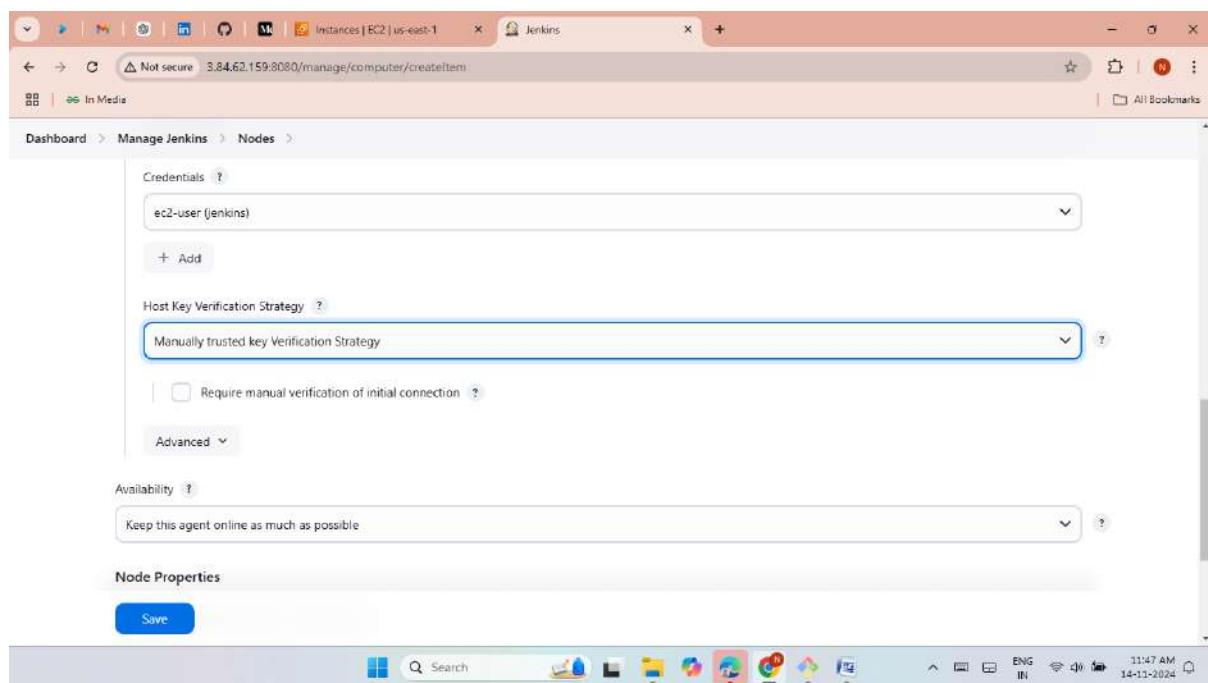
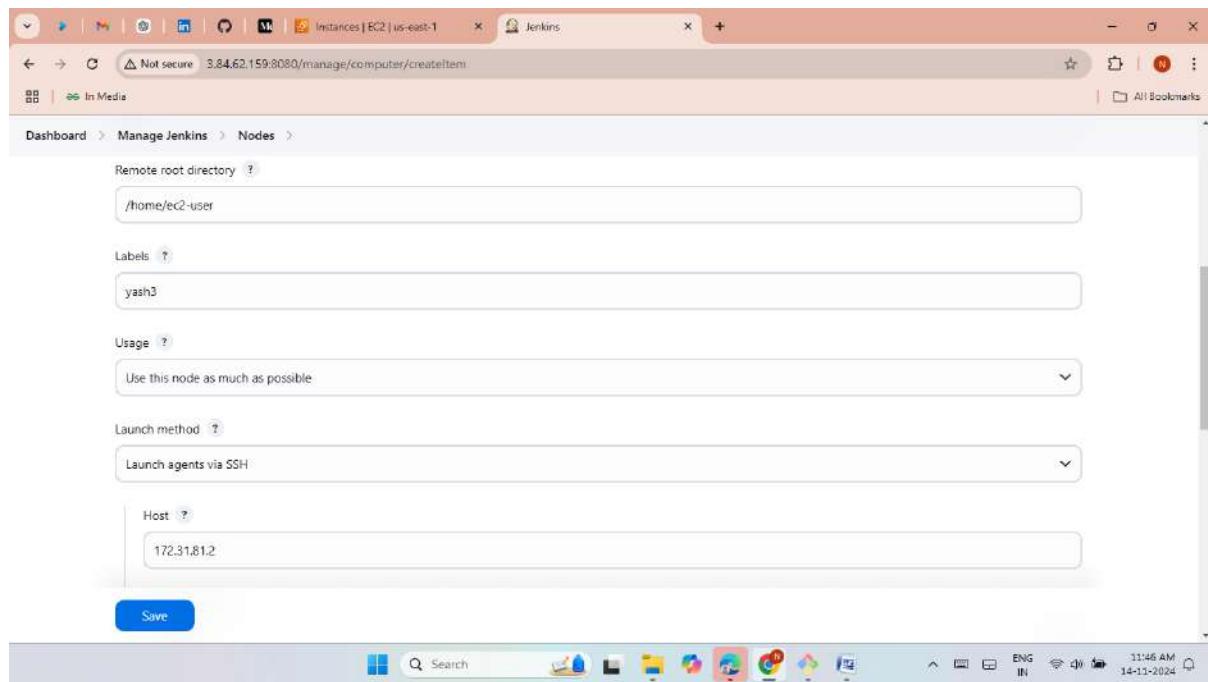
javalib.x86_64 0:1.0.9-9.amzn2.0.2
libICE.x86_64 0:1.0.9-9.amzn2.0.2
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXxi.x86_64 0:1.7.9-1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libxt.x86_64 0:1.1.5-3.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

dejavu-fonts-common.noarch 0:2.33-6.amzn2
dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2
Fontconfig.x86_64 0:2.13.0-4.3.amzn2
gifflib.x86_64 0:4.1.6-9.amzn2.0.2
libX11.x86_64 0:1.6.7-3.amzn2.0.5
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXxi.x86_64 0:1.7.9-1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libxt.x86_64 0:1.1.5-3.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

Complete!
[ec2-user@ip-172-31-81-2 ~]$ sudo hostname slave3
[ec2-user@ip-172-31-81-2 ~]$ exec bash
[ec2-user@slave3 ~]$ cd .ssh/
[ec2-user@slave3 .ssh]$ ls
authorized_keys
[ec2-user@slave3 .ssh]$ sudo vi authorized_keys
[ec2-user@slave3 .ssh]$ cd
[ec2-user@slave3 ~]$ pwd
/home/ec2-user
[ec2-user@slave3 ~]$ |
```

- Now create a node for slave3 server.
 - Give details for slave node.
 - Here we have to give slave3 path of root directory and private ip.
 - Now need to create credentials because already created in before slave.
 - Now click on save.





- Here the slave3 node is successfully created.

The screenshot shows the Jenkins 'Nodes' page. On the left, there are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node 0/2, slave3 0/20). The main area is titled 'Nodes' and contains a table with one row for each node. The table columns are: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The first node is 'Built-In Node' (Architecture: Linux (amd64), Free Disk Space: 4.83 GiB, Free Swap Space: 0 B, Free Temp Space: 4.83 GiB, Response Time: 0ms). The second node is 'slave3' (Architecture: Linux (amd64), Free Disk Space: 5.22 GiB, Free Swap Space: 0 B, Free Temp Space: 5.22 GiB, Response Time: 57ms). A legend at the bottom right indicates that the 'S' icon represents a Slave node. The status bar at the bottom right shows 'REST API Jenkins 2.485' and the system tray shows the date and time as '14-11-2024 11:47 AM'.

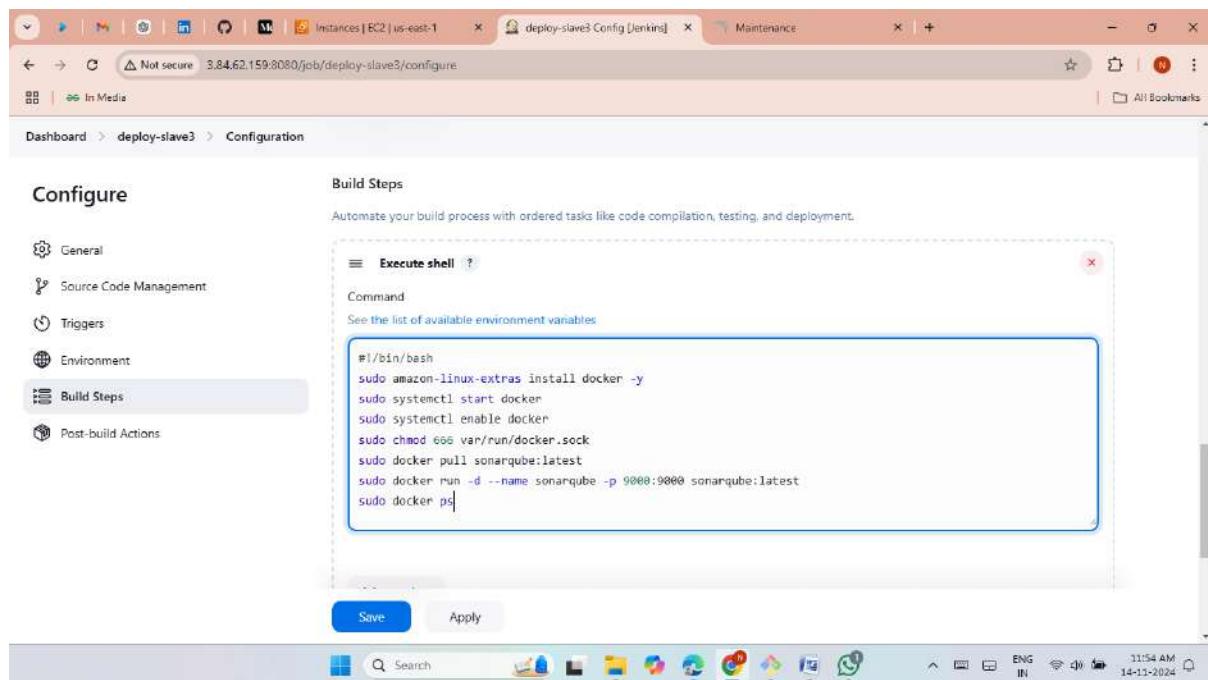
- Now create a job and give slave3 label.

The screenshot shows the Jenkins 'Configuration' page for the 'deploy-slave3' job. The 'Configure' section is open, showing the 'General' tab selected. Under 'General', the 'Restrict where this project can be run' checkbox is checked, and the 'Label Expression' field contains 'yash3'. A note below states 'Label yash3 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' The 'Source Code Management' section is set to 'None'. The 'Triggers' section is also visible. At the bottom, there are 'Save' and 'Apply' buttons. The status bar at the bottom right shows 'ENG IN 11:53 AM 14-11-2024'.

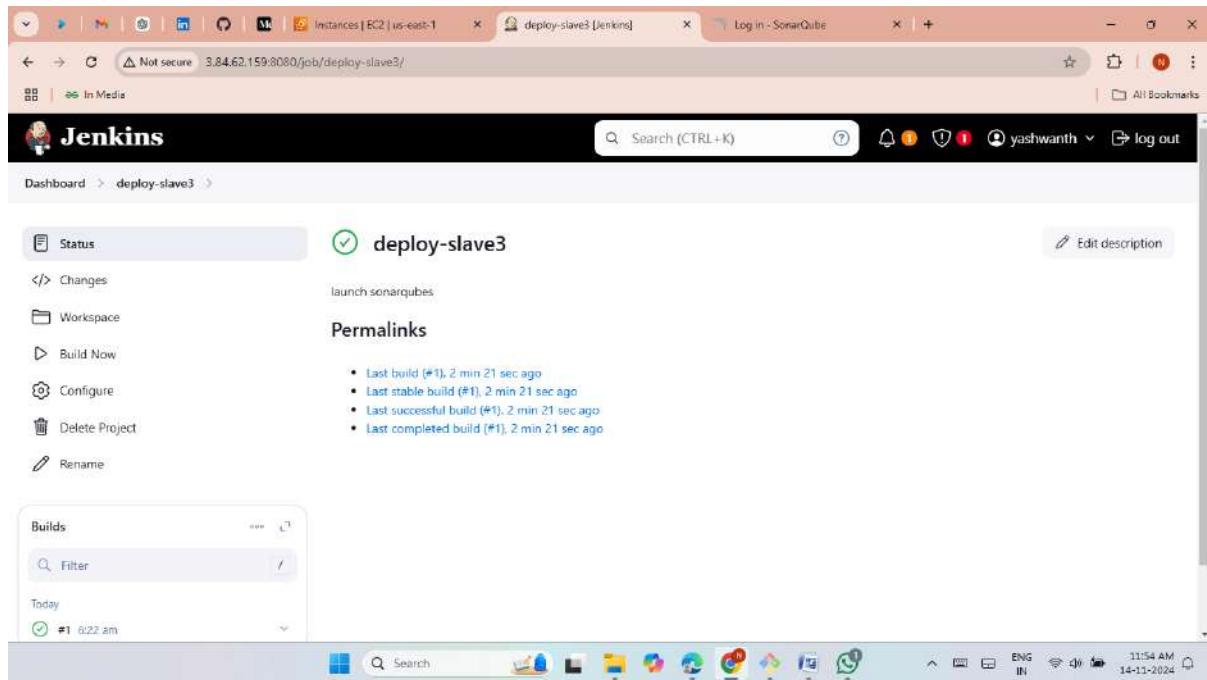
- Now install docker and then start and enable.
- Now pull sonarqube then run the sonarqubes.
- Now Click on save.

```
#!/bin/bash
```

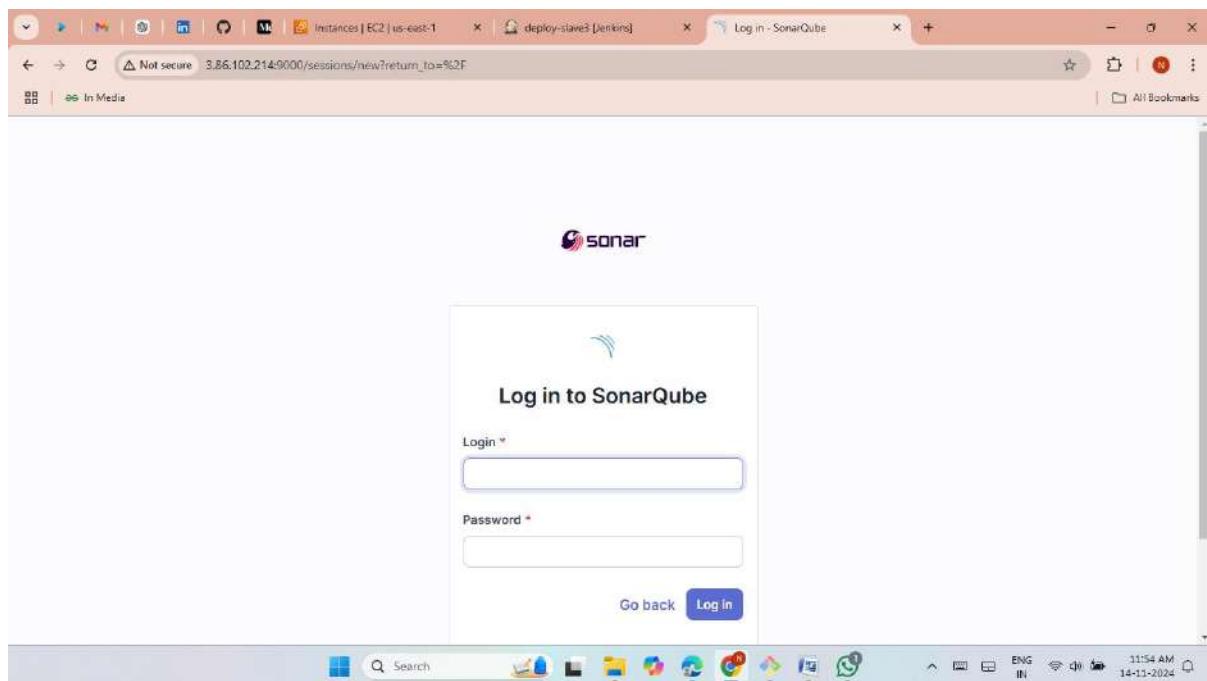
```
sudo amazon-linux-extras install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo chmod 666 var/run/docker.sock
sudo docker pull sonarqube:latest
sudo docker run -d --name sonarqube -p 9000:9000 sonarqube:latest
sudo docker ps
```



- Now click on build now.
- Here the job was success.



- Now copy the slave3 public ip and past it on Google browser.
- The sonarqubes web server was hosted successfully.



The screenshot shows the SonarQube interface for creating a new project. At the top, there are several import options: 'Import from Azure DevOps' (Setup), 'Import from Bitbucket Cloud' (Setup), 'Import from Bitbucket Server' (Setup), 'Import from GitHub' (Setup), and 'Import from GitLab' (Setup). Below these, there is a section for 'Create a local project'. The status bar at the bottom indicates the browser is not secure and shows the URL 3.86.102.214:9000/projects/create.

➤ Now launch an EC2 instances and give name as slave4.

The screenshot shows the AWS EC2 Instances page. On the left, there is a sidebar with various navigation links. The main area displays a table of instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. There are three instances listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
slave-3	i-00c6da07a75701c42	Terminated	t2.medium	-	View alarms +	us-east-1d
master	i-028192448460f511	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d
slave-4	i-05506abaebc0acd9a	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a

A modal window is open for the instance 'i-05506abaebc0acd9a (slave-4)', showing details like Public IPv4 address (54.90.198.153), Private IPv4 addresses (172.31.20.110), and Public IPv4 DNS (ec2-54-90-198-153.compute-1.amazonaws.com).

- Now connect Slave4 then install java.
- Now give private keys from master server.

```

Verifying : libxi-1.7.9-1.amzn2.0.2.x86_64          22/32
Verifying : log4j-cve-2021-44228-hotpatch-1.3-7.amzn2.noarch 23/32
Verifying : libX11-1.2.3-1.amzn2.x86_64           24/32
Verifying : libXSLT-1.1.28-6.amzn2.x86_64         25/32
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64   26/32
Verifying : python-javapackages-3.4.1-11.amzn2.noarch 27/32
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64       28/32
Verifying : alsa-lib-1.1.4-1-2.amzn2.x86_64        29/32
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch 30/32
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64        31/32
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch 32/32

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

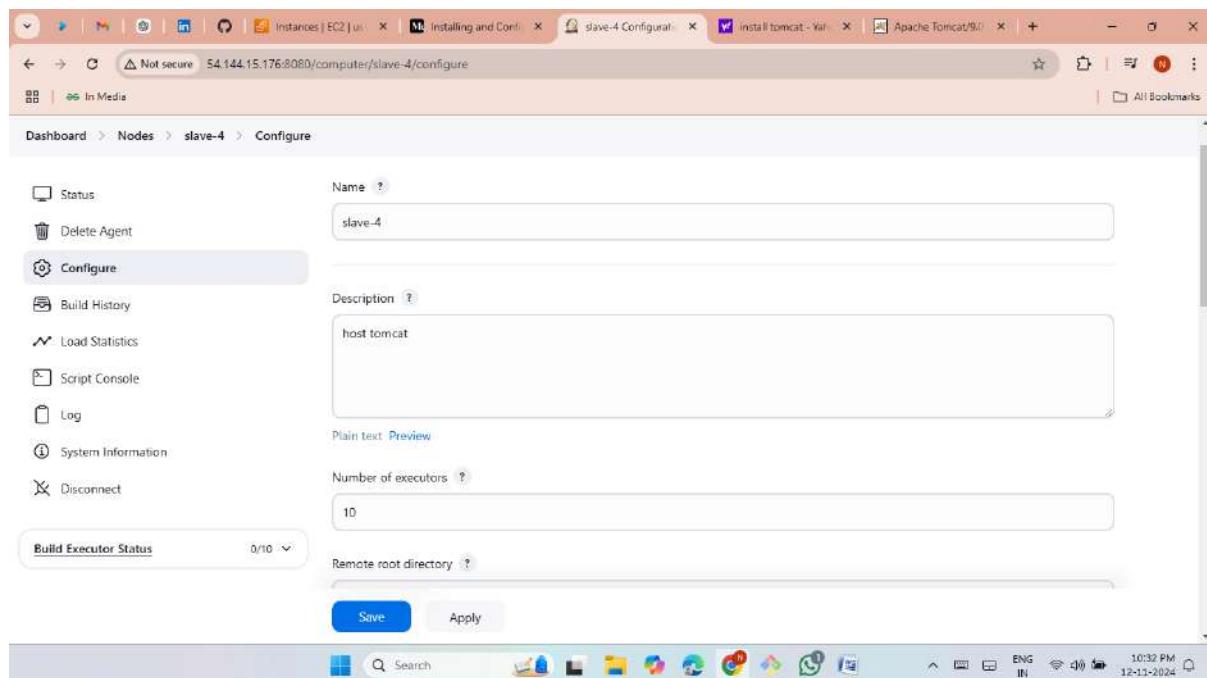
Complete!
[ec2-user@slave4 .ssh]$ ls
authorized_keys
[ec2-user@slave4 .ssh]$ sudo vi authorized_keys

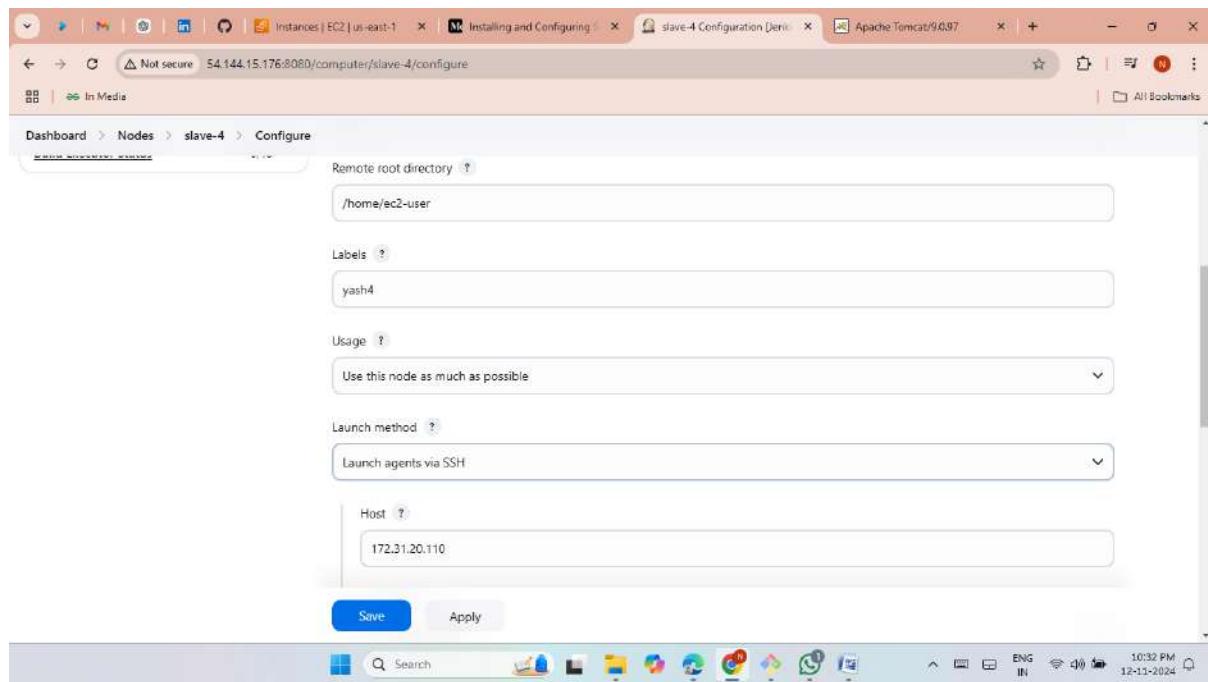
```

The screenshot shows a terminal window with the following content:

- Output of a package verification command showing success for all packages.
- Installed Java packages: java-17-amazon-corretto.x86_64, java-17-amazon-corretto-devel.x86_64, and java-17-amazon-corretto-javadoc.x86_64.
- Dependency Installed packages: alsa-lib.x86_64, dejavu-sans-fonts.noarch, dejavu-serif-fonts.noarch, fontpackages-filesystem.noarch, javapackages-tools.noarch, libSM.x86_64, libX11-common.noarch, libXext.x86_64, libXinerama.x86_64, libXrender.x86_64, libXtst.x86_64, libXslt.x86_64, and python-javapackages.noarch.
- File listing: authorized_keys.

- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.





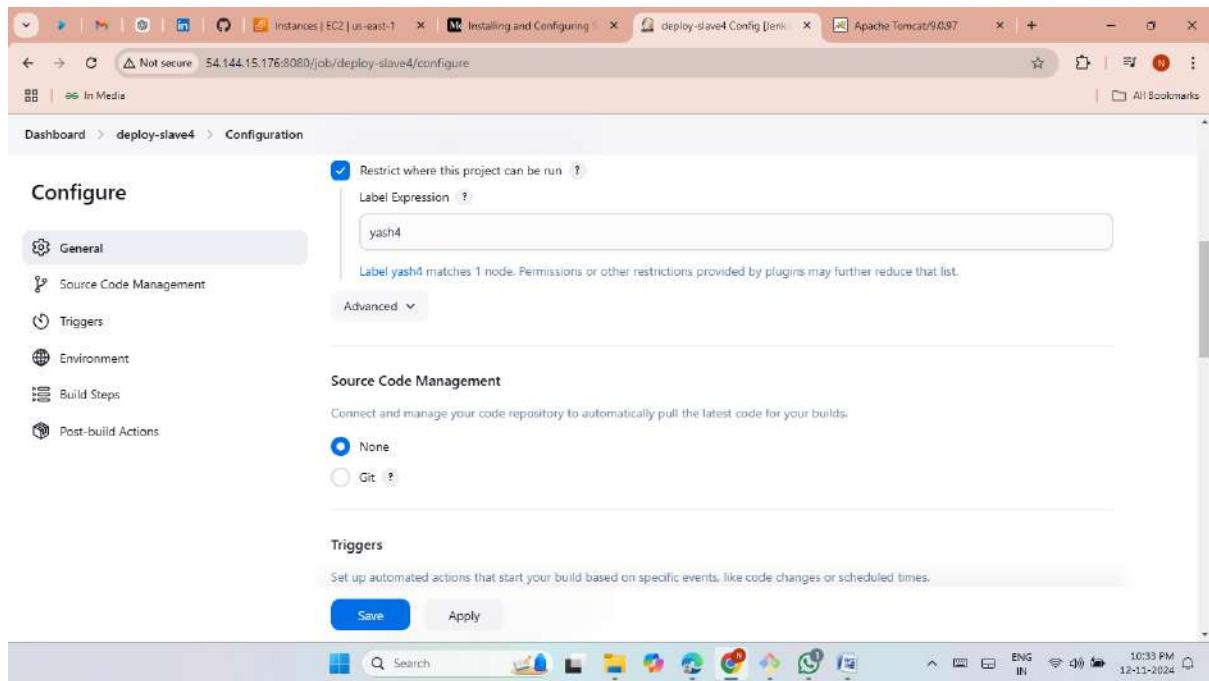
➤ Here the slave4 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.02 GiB	0 B	5.02 GiB	0ms
	slave-4	Linux (amd64)	In sync	5.40 GiB	0 B	5.40 GiB	87ms
	Data obtained	23 min	23 min	23 min	23 min	23 min	23 min

Legend: S, M, L

REST API Jenkins 2.485

- Now create a job and give slave4 label.



- Now install tomcat and then start the tomcat.
- Here the commands for tomcat.

```
sudo wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.97/bin/apache-tomcat-9.0.97.tar.gz
```

```
sudo tar -xvf apache-tomcat-9.0.97.tar.gz
```

```
sudo chmod +x apache-tomcat-9.0.97/bin
```

```
cd apache-tomcat-9.0.97
```

```
cd bin/
```

```
sudo ./startup.sh
```

Now Click on save.

The screenshot shows the Jenkins job configuration interface for a job named 'deploy-slave4'. On the left, there's a sidebar with options: General, Source Code Management, Triggers, Environment, Build Steps (which is selected), and Post-build Actions. The main area is titled 'Build Steps' with the sub-section 'Execute shell'. It contains a command box with the following script:

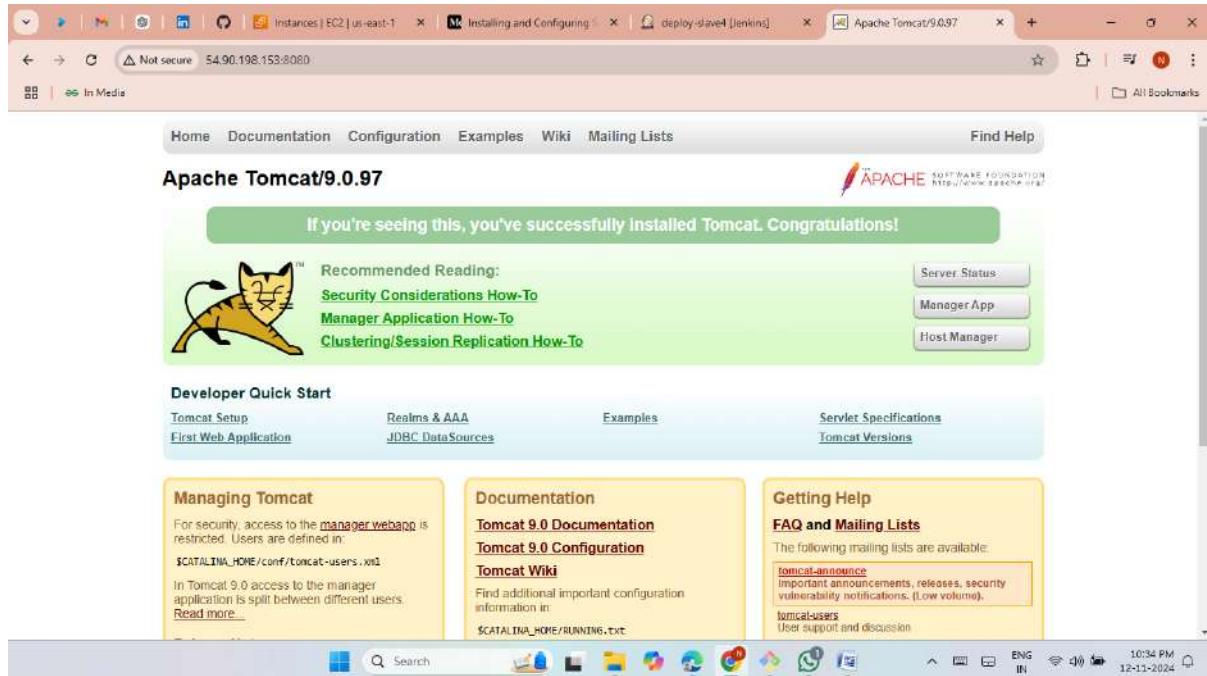
```
sudo wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.97/bin/apache-tomcat-9.0.97.tar.gz  
sudo tar -xvf apache-tomcat-9.0.97.tar.gz  
sudo chmod +x apache-tomcat-9.0.97/bin  
cd apache-tomcat-9.0.97  
cd bin/  
sudo ./startup.sh
```

Below the command box are 'Advanced' and 'Save' buttons. The status bar at the bottom indicates it's 10:33 PM on 12-13-2024.

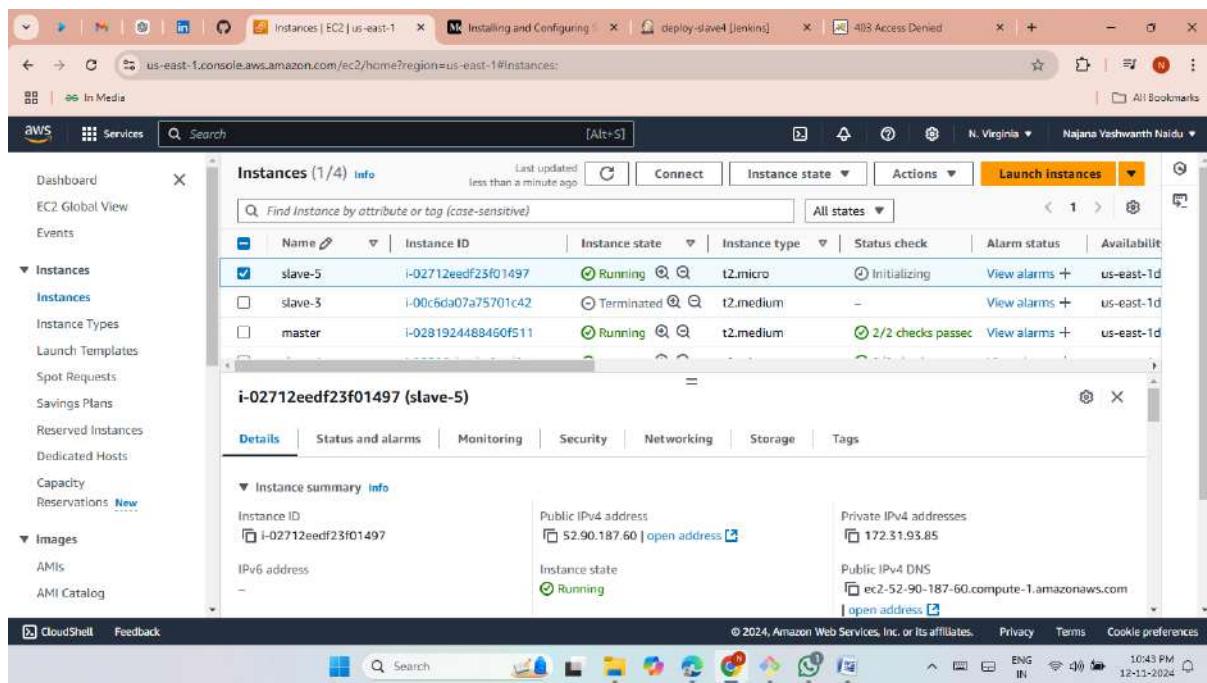
- Now click on build now.
- Here the job was success.

The screenshot shows the Jenkins job status page for 'deploy-slave4'. The top navigation bar includes links for Instances | EC2 | us-east-1, Installing and Configuring, deploy-slave4 [Jenkins], and Apache Tomcat/9.0.97. The main content area displays the job name 'deploy-slave4' with a green checkmark icon. Below it, the status message 'launched tomcat' is shown. A 'Permalinks' section lists several build history items. At the bottom, a 'Builds' table shows one entry from today at 4:59 pm. The status bar at the bottom indicates it's 10:33 PM on 12-13-2024.

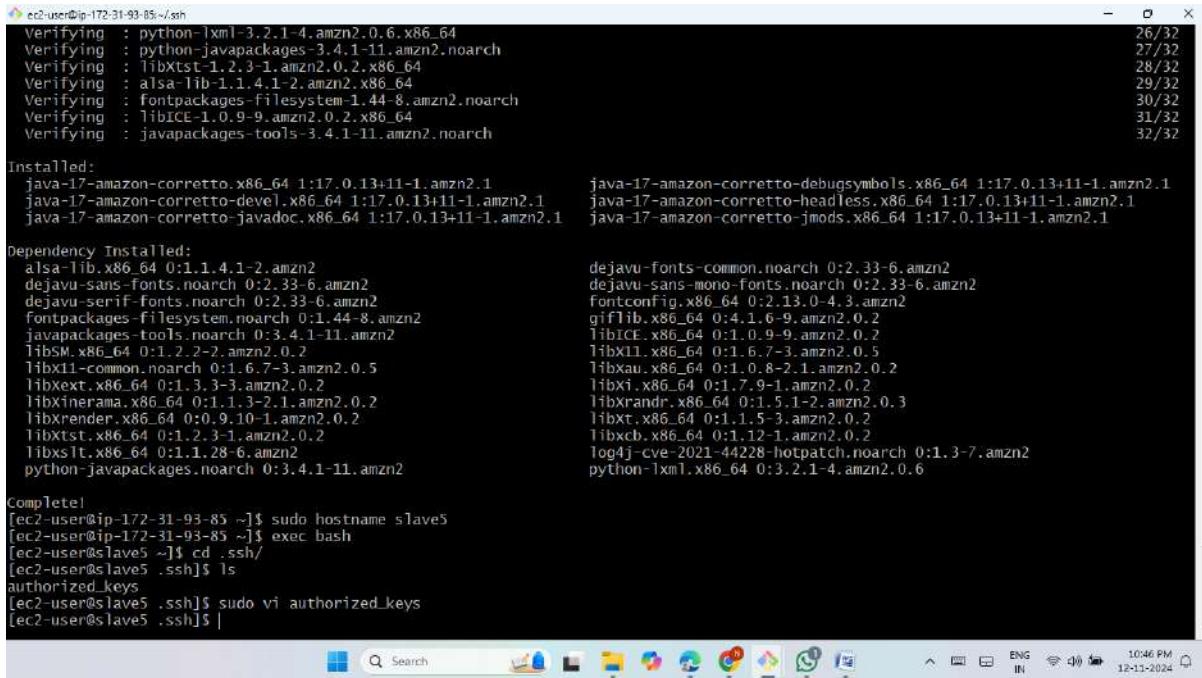
- Now copy the slave4 public ip with the tomcat port (8080) and past it on Google browser.
- The tomcat web server was hosted successfully.



- Now launch an ec2 instances and give name as slave5.



- Now connect Slave5 then install java.
- Now give private keys from master server.



```

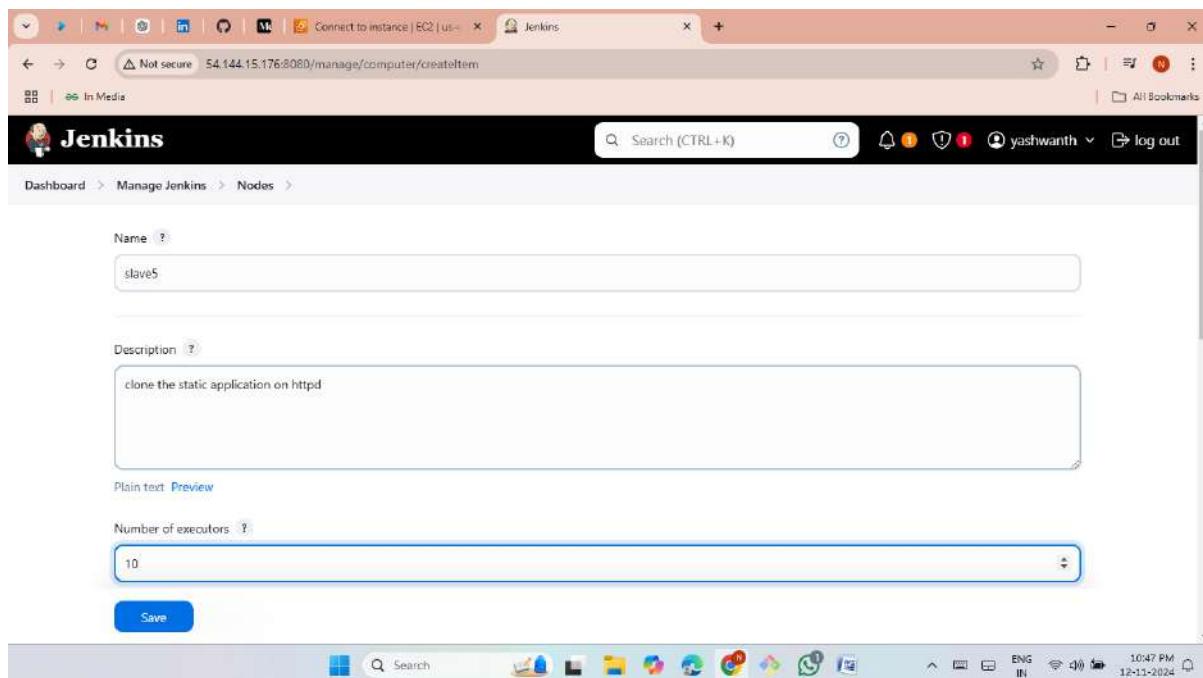
ec2-user@ip-172-31-93-85:~$ ssh
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64
Verifying : python-javapackages-3.4.1-11.amzn2.noarch
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64
Verifying : alsa-lib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch
Verifying : 26/32
Verifying : 27/32
Verifying : 28/32
Verifying : 29/32
Verifying : 30/32
Verifying : 31/32
Verifying : 32/32

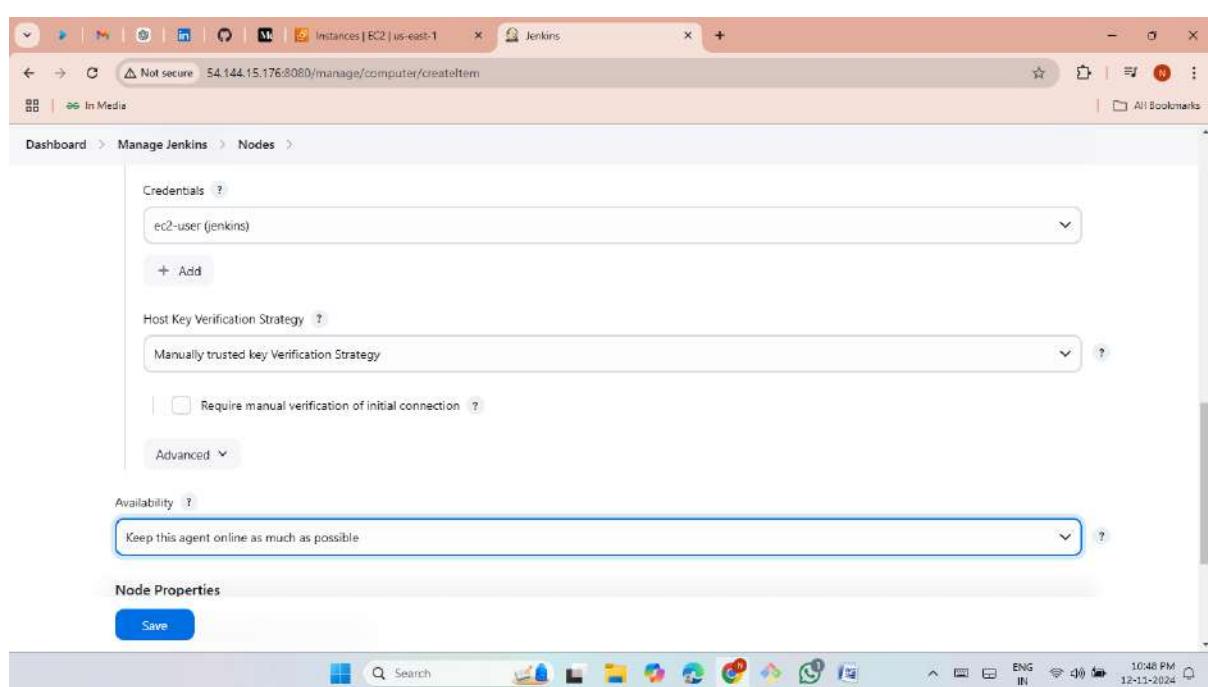
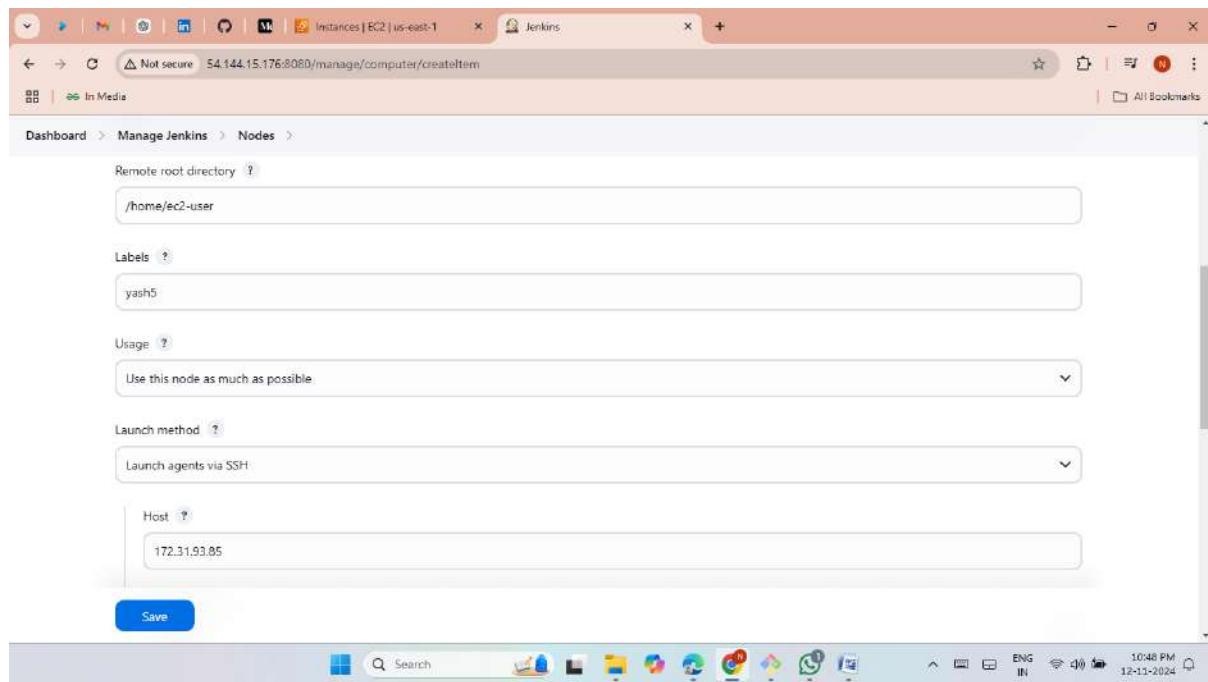
Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-jmods.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2
dejavu-fonts-common.noarch 0:2.33-6.amzn2
dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2
fontconfig.x86_64 0:2.13.0-4.3.amzn2
giflib.x86_64 0:4.1.6-9.amzn2.0.2
libICE.x86_64 0:1.0.9-9.amzn2.0.2
libX11.x86_64 0:1.6.7-3.amzn2.0.5
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXi.x86_64 0:1.7.9-1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libXt.x86_64 0:1.1.5-3.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

Complete!
[ec2-user@ip-172-31-93-85 ~]$ sudo hostname slave5
[ec2-user@ip-172-31-93-85 ~]$ exec bash
[ec2-user@slave5 ~]$ cd .ssh/
[ec2-user@slave5 .ssh]$ ls
authorized_keys
[ec2-user@slave5 .ssh]$ sudo vi authorized_keys
[ec2-user@slave5 .ssh]$ |
```

- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.





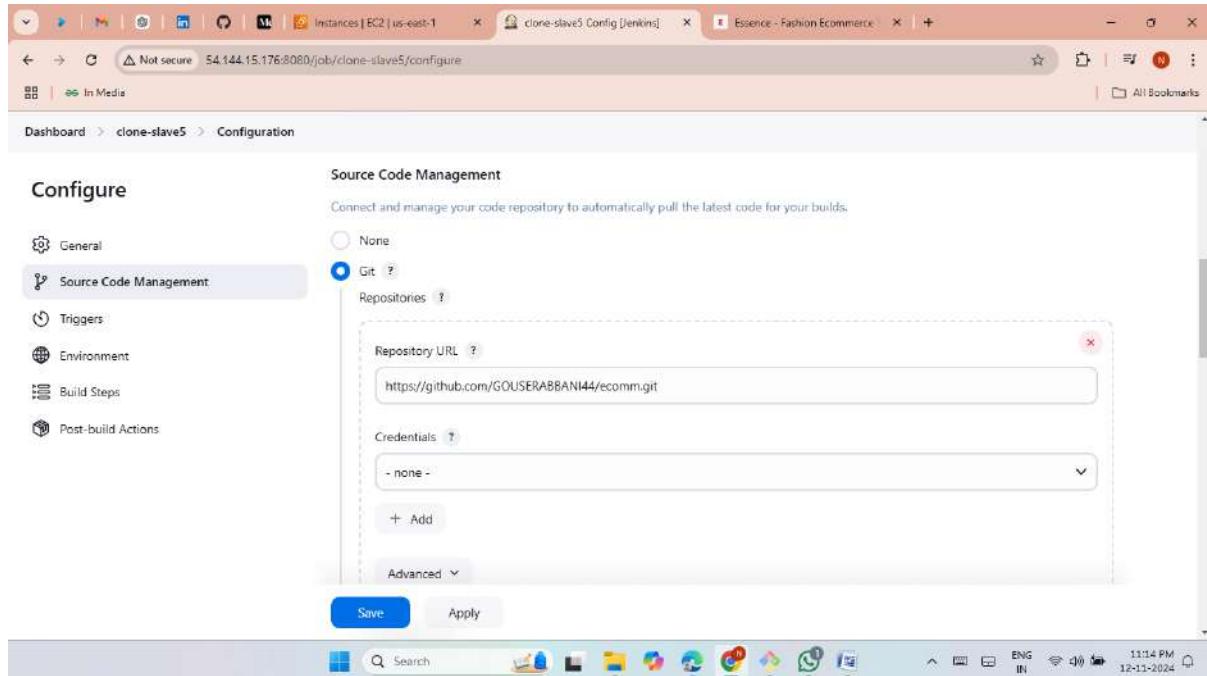
- Here the slave5 node is successfully created.

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (listing 'Built-In Node', 'slave-4', and 'slave5' with counts 0/2, 0/10, and 0/10 respectively). The main area is titled 'Nodes' and contains a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The table lists three nodes: 'Built-In Node' (Linux (amd64), In sync, 5.02 GiB free disk, 0.8 GiB swap, 5.02 GiB temp, 0ms response), 'slave-4' (Linux (amd64), In sync, 5.31 GiB free disk, 0.8 GiB swap, 5.31 GiB temp, 35ms response), and 'slave5' (Linux (amd64), In sync, 5.41 GiB free disk, 0.8 GiB swap, 5.41 GiB temp, 30ms response). A 'Data obtained' row at the bottom shows values: 4.5 sec, 4.5 sec, 4.6 sec, 4.5 sec, 4.5 sec, 4.6 sec, and 4.6 sec. Below the table are icons for sorting by column: S, M, L. At the bottom right are links for 'REST API' and 'Jenkins 2.485'. The system tray at the bottom shows the date and time as 12-11-2024 10:48 PM.

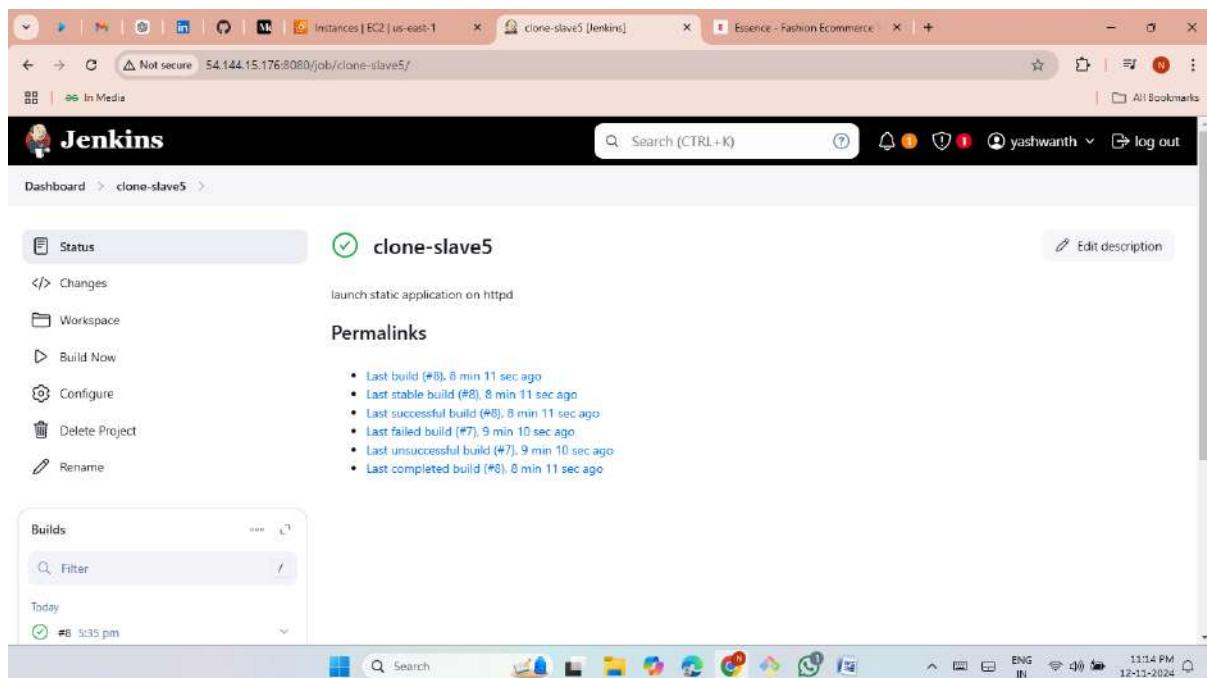
- Now create a job and give slave5 label.

The screenshot shows the 'clone-slave5' configuration page under 'clone-slave5 > Configuration'. The 'General' section is selected. Under 'Restrict where this project can be run', the 'Label Expression' field contains 'yash5'. A note below states 'Label yash5 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' There are 'Save' and 'Apply' buttons at the bottom. The system tray at the bottom shows the date and time as 12-11-2024 10:52 PM.

- Now clone the repo from github.
- Click on save.



- Click on build now.
- Here the job was success.



- Now create another job.
- Now install httpd and then start and enable.
- Now Click on save.

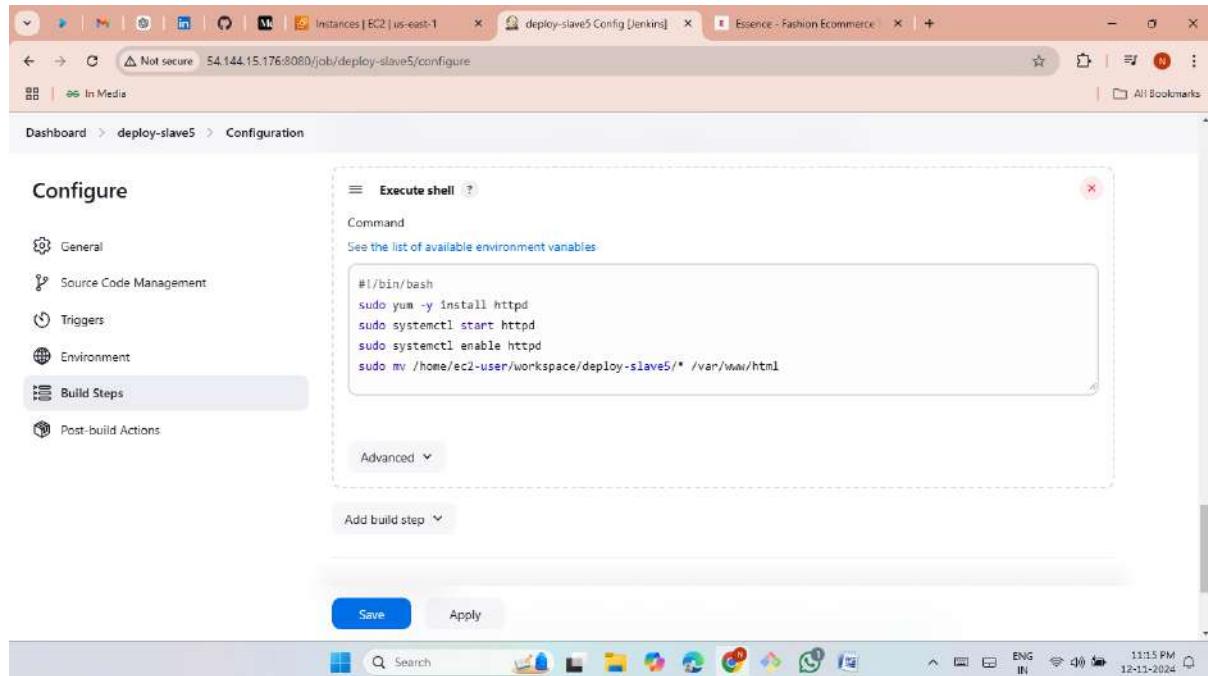
```
#!/bin/bash
```

```
sudo yum -y install httpd
```

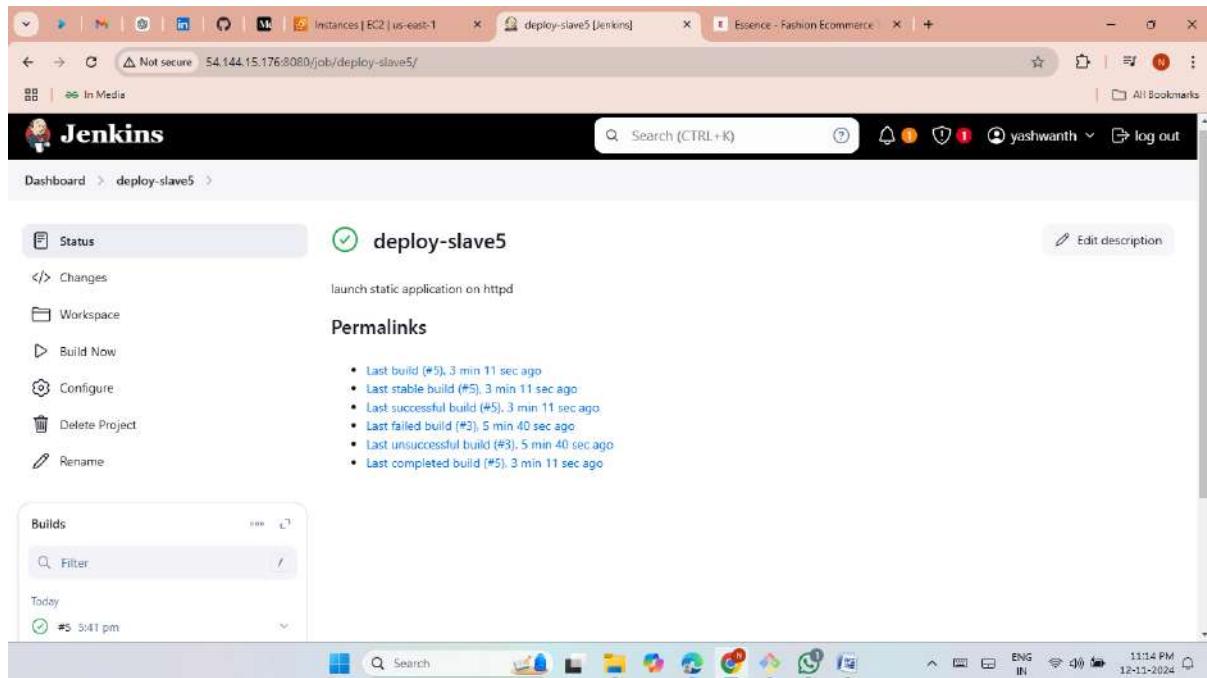
```
sudo systemctl start httpd
```

```
sudo systemctl enable httpd
```

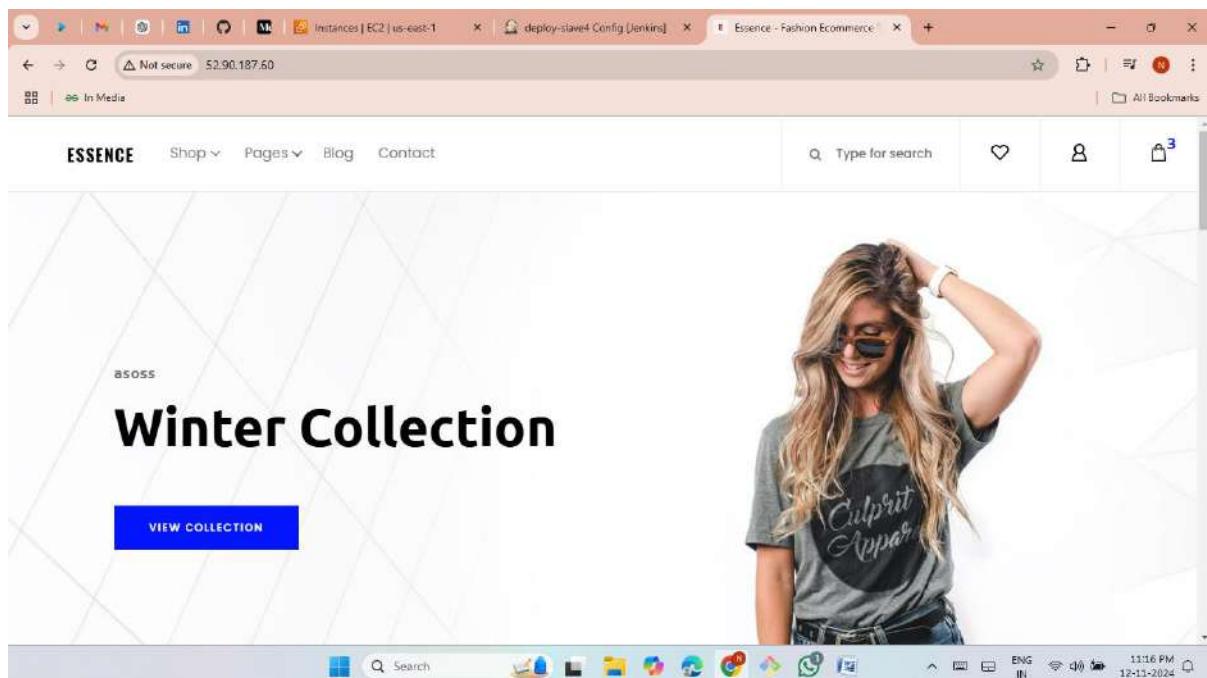
```
sudo mv /home/ec2-user/workspace/deploy-slave5/* /var/www/html
```



- Now click on build now.
- Here the job was success.



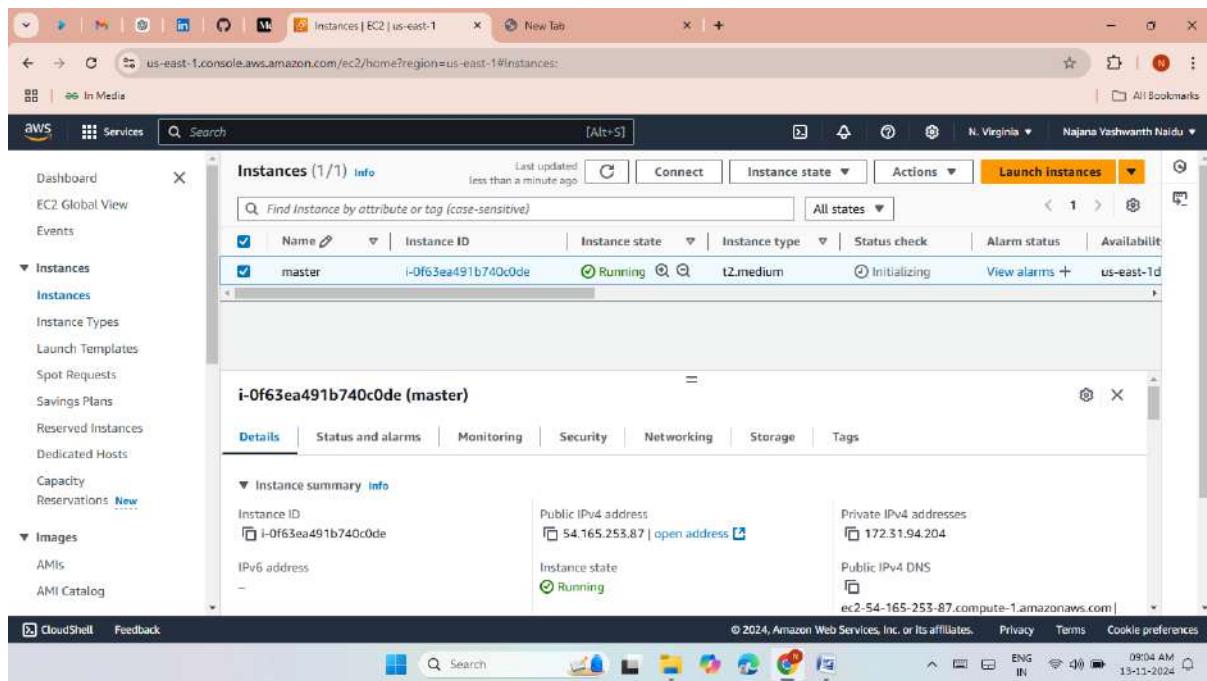
- Now copy the slave5 public ip and past it on Google browser.
- The ecomm web application was hosted successfully.



Method -2

Create one master and five slave nodes and deploy python webapplications through manually (Slave-1), bash script (Slave-2), terraform (Slave-3) and deploy java-based application through manually (Slave-4) and bash script (execute shell) (Slave-5).

- First launch an master server with the Jenkins port 8080.



- Now install java and git.
- Now install Jenkins then start and enable the Jenkins.
- We can change the hostname with the commands

Sudo hostname master

Exec bash

- Now connect to the Master server and then generate keys with the command.

ssh-keygen

```

cc2-user@ip-172-31-94-204:~/.ssh
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

complete!
[ec2-user@master ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:3mXXyFvjYcvPNWfOMVzc+QoRMuPWW+7dSTBmRDlkOc ec2-user@master
The key's randomart image is:
+---[RSA 2048]---+
|          *+=|
|         .&+|
|        =& @|
|       %E*o|
|      S   * =|
|     . . o   o=|
|     . . o=|
|           o B|
|             =.|
+---[SHA256]----+
[ec2-user@master ~$]
[ec2-user@master ~]$ cd .ssh/
[ec2-user@master .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub
[ec2-user@master .ssh]$ 

```

```

cc2-user@ip-172-31-94-204:~/
CGroup: /system.slice/jenkins.service
└─6631 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Nov 13 03:44:57 master jenkins[6631]: ****
Nov 13 03:44:57 master jenkins[6631]: ****
Nov 13 03:44:57 master jenkins[6631]: ****
Nov 13 03:45:00 master jenkins[6631]: 2024-11-13 03:45:00.778+0000 [id=49] INFO h.m.DownloadService$Downloader#staller
Nov 13 03:45:00 master jenkins[6631]: 2024-11-13 03:45:00.779+0000 [id=49] INFO hudson.util.Retriger#start: Per...empt #1
Nov 13 03:45:00 master jenkins[6631]: 2024-11-13 03:45:00.798+0000 [id=33] INFO jenkins.InitReactorRunner$1#on...ization
Nov 13 03:45:00 master jenkins[6631]: 2024-11-13 03:45:00.809+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onR...running
Nov 13 03:45:00 master systemd[1]: Started Jenkins Continuous Integration Server.
Nov 13 03:45:33 master systemd[1]: [/usr/lib/systemd/system/jenkins.service:16] Unknown lvalue 'StartLimitBurst' in section 'Unit'
Nov 13 03:45:33 master systemd[1]: [/usr/lib/systemd/system/jenkins.service:17] Unknown lvalue 'StartLimitIntervalSec' in se... 'Unit'
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@master ~]$ sudo visudo
[ec2-user@master ~]$ sudo systemctl restart jenkins
[ec2-user@master ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since Wed 2024-11-13 03:46:57 UTC; 4s ago
    Main PID: 8066 (java)
   CGroup: /system.slice/jenkins.service
          └─8066 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Nov 13 03:46:54 master jenkins[8066]: Jenkins initial setup is required. An admin user has been created and a password generated.
Nov 13 03:46:54 master jenkins[8066]: Please use the following password to proceed to installation:
Nov 13 03:46:54 master jenkins[8066]: 57983e65d05748cb8d043cb5539d9e6
Nov 13 03:46:54 master jenkins[8066]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Nov 13 03:46:54 master jenkins[8066]: ****
Nov 13 03:46:54 master jenkins[8066]: ****
Nov 13 03:46:54 master jenkins[8066]: 2024-11-13 03:46:57.488+0000 [id=30] INFO jenkins.InitReactorRunner$1#on...ization
Nov 13 03:46:57 master jenkins[8066]: 2024-11-13 03:46:57.502+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onR...running
Nov 13 03:46:57 master systemd[1]: Started Jenkins Continuous Integration Server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@master ~]$ 

```

- Here the Jenkins was launched successfully.

The screenshot shows the Jenkins dashboard at 54.165.253.87:8080. The main heading is "Welcome to Jenkins!". Below it, a message says: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." There are several sections: "Build Queue" (No builds in the queue), "Build Executor Status" (9/2), "Create a job" (button with a plus sign), "Set up a distributed build" (with "Set up an agent" and "Configure a cloud" buttons), and "Start building your software project". A sidebar on the left includes links for "New Item", "Build History", "Manage Jenkins", and "My Views". The top right has a "log out" link. The bottom right shows system status: ENG IN, 08:19 AM, 13-11-2024.

- Now launch an Ec2 instances and give name as slave1.

The screenshot shows the AWS EC2 Instances page. The left sidebar lists options like Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations, Images, AMIs, and AMI Catalog. The main area shows a table of instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability zone
master	i-0114a3ead1beb63fc	Running	t2.medium	2/2 checks passed	View alarms	us-east-1d
slave-1	i-06696cf6410bd9132	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a

A modal window for instance **i-06696cf6410bd9132 (slave-1)** is open, showing details like Instance ID (i-06696cf6410bd9132), Public IPv4 address (35.172.179.81), Private IPv4 addresses (172.31.19.214), and Instance state (Running). The bottom of the screen shows the AWS navigation bar and system status: ENG IN, 05:02 PM, 13-11-2024.

- Now connect Slave1 then install java.
- Now give private keys from master server.

```

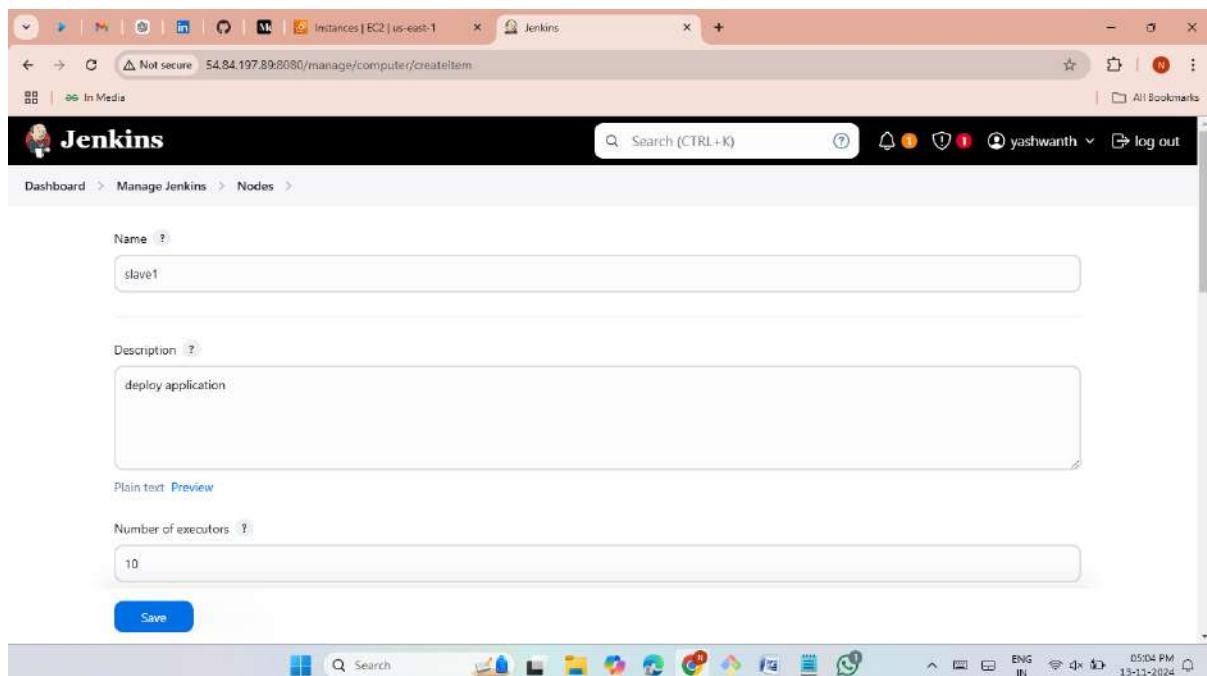
Verifying : libxtst-1.1.2-6.amzn2.x86_64 24/32
Verifying : libxslt-1.1.28-6.amzn2.x86_64 25/32
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64 26/32
Verifying : python-javapackages-3.4.1-11.amzn2.noarch 27/32
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64 28/32
Verifying : alsa-lib-1.1.4.1-2.amzn2.x86_64 29/32
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch 30/32
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64 31/32
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch 32/32

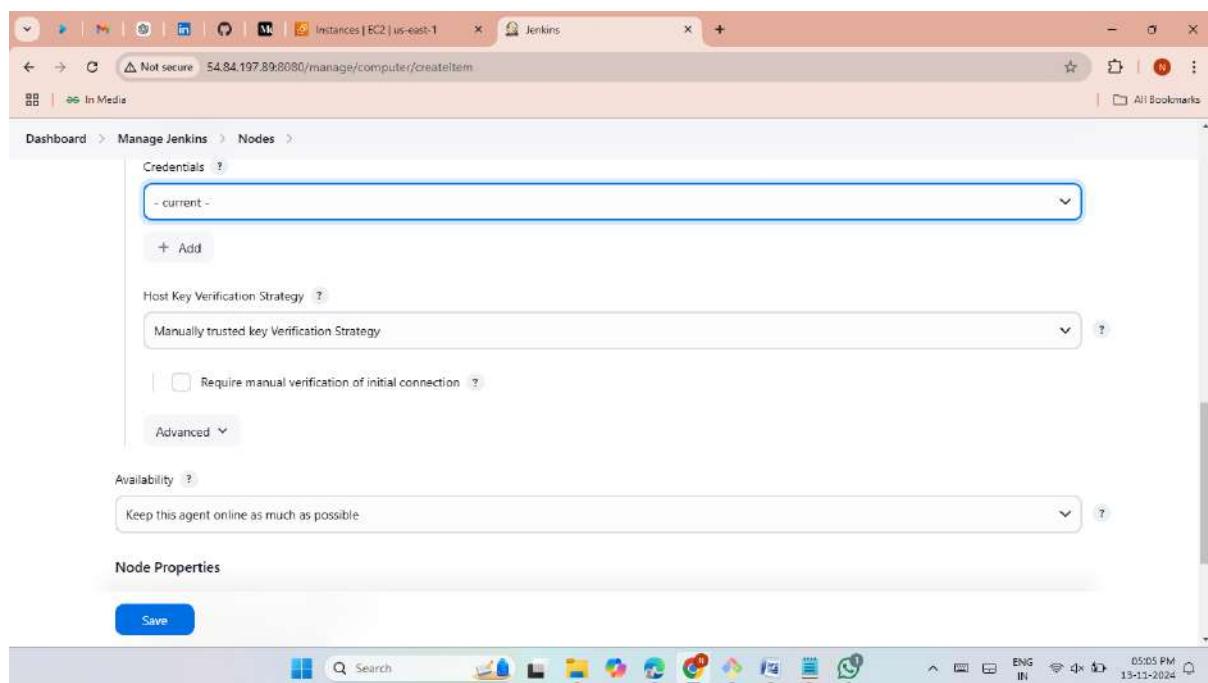
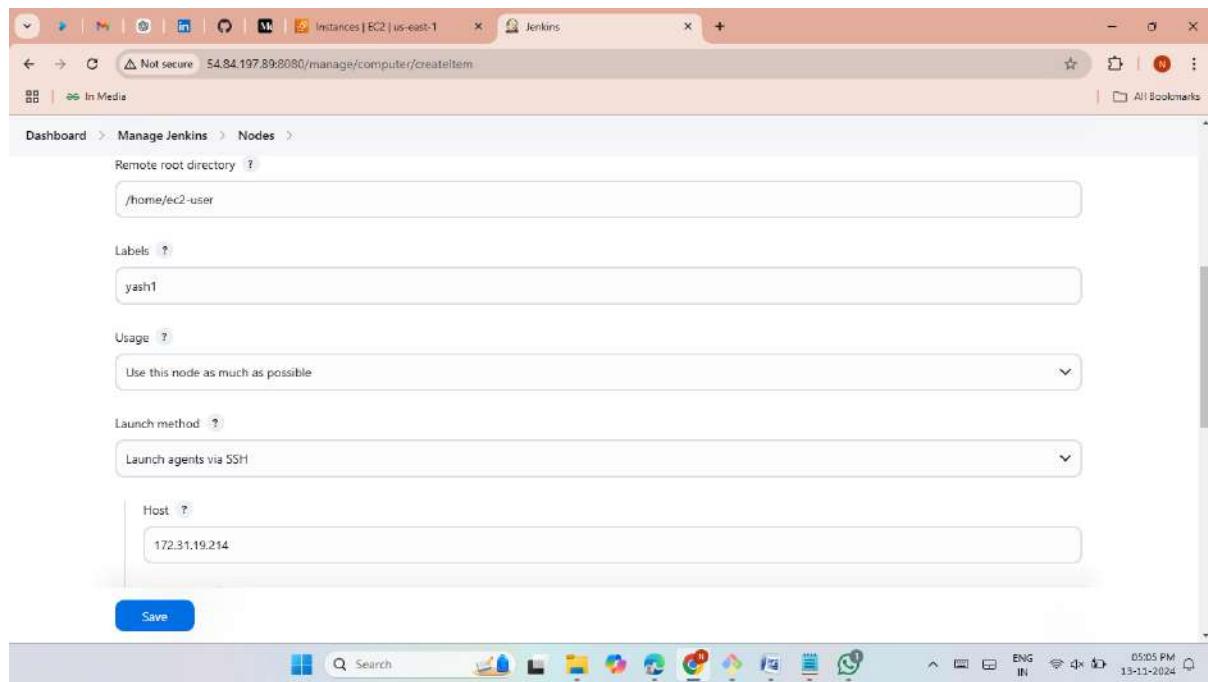
Installed:
  java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1      24/32
  java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1 25/32
  java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1 26/32
  java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1 27/32
  java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1 28/32
  java-17-amazon-corretto-jmods.x86_64 1:17.0.13+11-1.amzn2.1 29/32
  dejavu-fonts-common.noarch 0:2.33-6.amzn2 30/32
  dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2 31/32
  fontconfig.x86_64 0:2.13.0-4.3.amzn2 32/32
  giflib.x86_64 0:4.1.6-9.amzn2.0.2 33/32
  libICE.x86_64 0:1.0.9-9.amzn2.0.2 34/32
  libX11.x86_64 0:1.6.7-3.amzn2.0.5 35/32
  libXau.x86_64 0:1.0.8-2.1.amzn2.0.2 36/32
  libXi.x86_64 0:1.7.9-1.amzn2.0.2 37/32
  libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2 38/32
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2 39/32
  libXtst.x86_64 0:1.2.3-1.amzn2.0.2 40/32
  libXslt.x86_64 0:1.1.4.1-2.amzn2 41/32
  python-javapackages.noarch 0:3.4.1-11.amzn2 42/32

Dependency Installed:
  alsa-lib.x86_64 0:1.1.4.1-2.amzn2
  dejavu-sans-fonts.noarch 0:2.33-6.amzn2
  dejavu-serif-fonts.noarch 0:2.33-6.amzn2
  fontpackages-filesystem.noarch 0:1.44-8.amzn2
  javapackages-tools.noarch 0:3.4.1-11.amzn2
  libSM.x86_64 0:1.2.2-2.amzn2.0.2
  libX11-common.noarch 0:1.6.7-3.amzn2.0.5
  libXext.x86_64 0:1.3.3-3.amzn2.0.2
  libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2
  libXtst.x86_64 0:1.2.3-1.amzn2.0.2
  libXslt.x86_64 0:1.1.4.1-2.amzn2
  python-javapackages.noarch 0:3.4.1-11.amzn2

Complete!
[ec2-user@ip-172-31-19-214 ~]$ cd .ssh/
[ec2-user@ip-172-31-19-214 .ssh]$ ls
authorized_keys
[ec2-user@ip-172-31-19-214 .ssh]$ sudo vi authorized_keys
[ec2-user@ip-172-31-19-214 .ssh]$ 
```

- Now create a node for slave1 server.
- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.





- Here the slave1 node is successfully created.

The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node, slave1, slave3). The main area is titled 'Nodes' and contains a table with the following data:

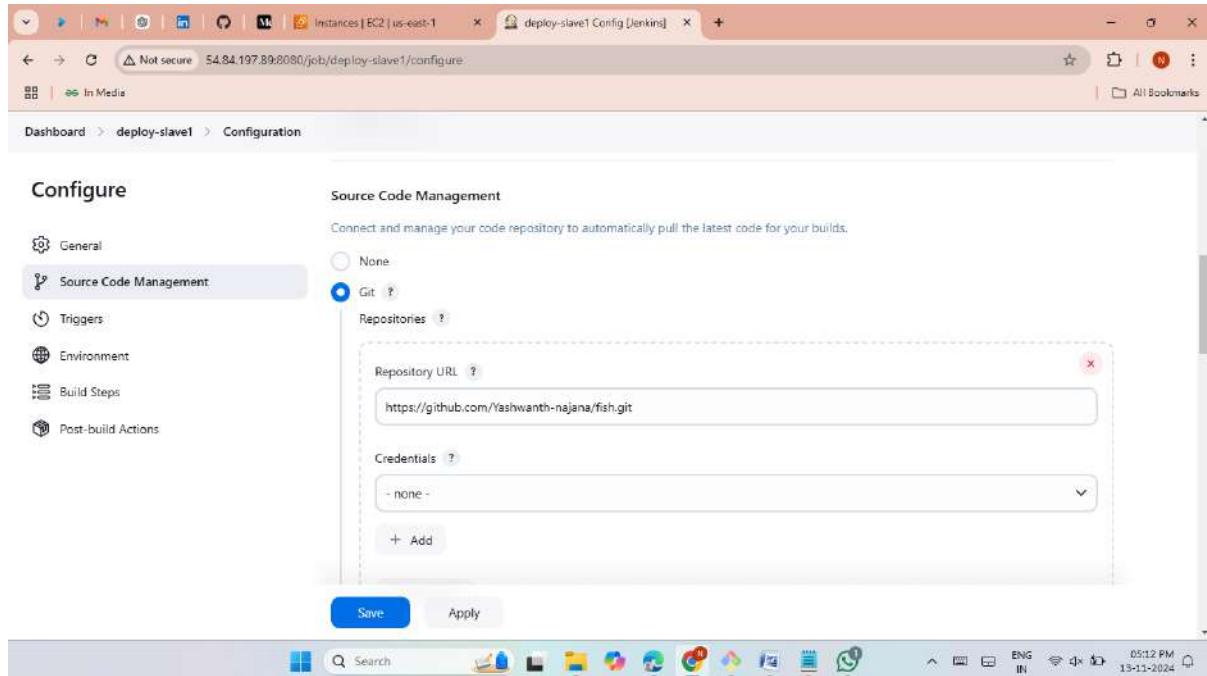
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.95 GiB	! 0.8	4.95 GiB	0ms
	slave1	Linux (amd64)	In sync	5.36 GiB	! 0.8	5.36 GiB	64ms
	slave3	Linux (amd64)	In sync	3.48 GiB	! 0.8	3.48 GiB	47ms
	Data obtained	1.1 sec	1.1 sec	1.1 sec	1 sec	1.1 sec	1.1 sec

At the bottom right of the table, there is a 'Legend' section with icons for S, M, and L.

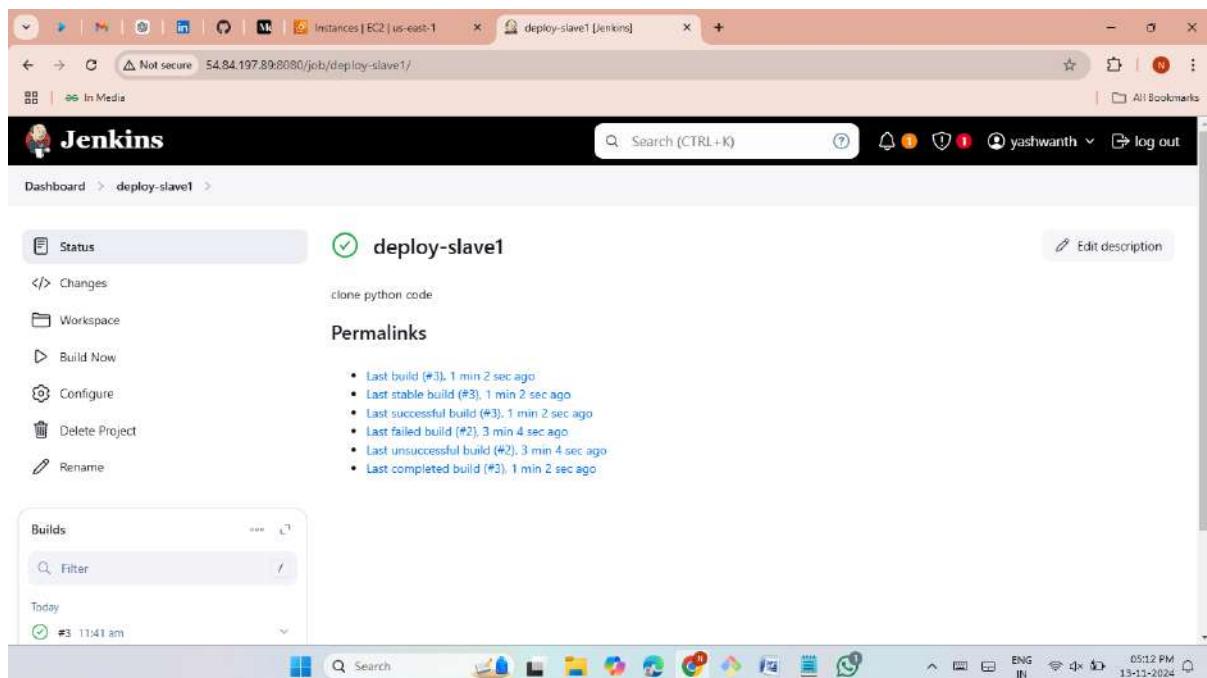
- Now create a job and give slave1 label.

The screenshot shows the Jenkins 'Configuration' page for the 'deploy-slave1' job. The 'General' section is selected. Under 'Restrict where this project can be run', the 'Label Expression' field contains 'yash1'. A note below states: 'Label yash1 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' The 'Source Code Management' section is also visible at the bottom.

- Now clone the python application repo from github.
- Click on save.



- Now click on build now.
- Here the job was success.



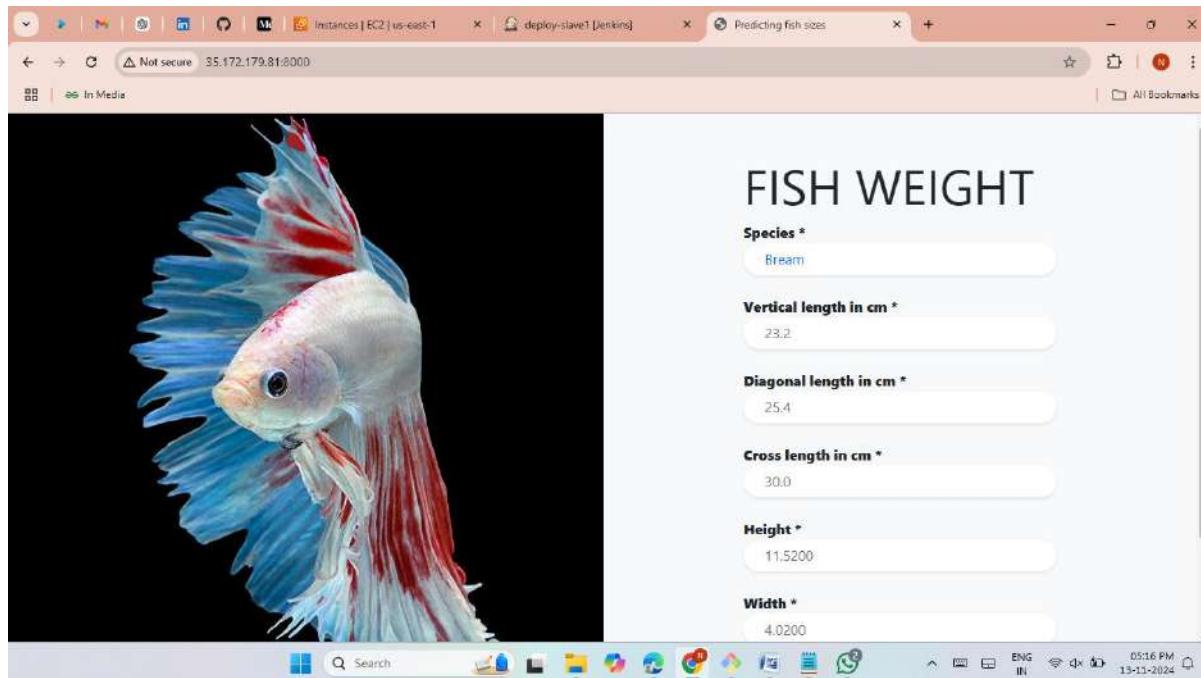
- Now run the python applications with the commands.
- Pip3 install -r requirements.txt
- Screen -m -d python3 app.py

```
ec2-user@ip-172-31-19-214:~/workspace/deploy-slave1
[ec2-user@ip-172-31-19-214 ~]$ ls
remoting remoting.jar workspace
[ec2-user@ip-172-31-19-214 ~]$ clear
[ec2-user@ip-172-31-19-214 ~]$ ls
remoting remoting.jar workspace
[ec2-user@ip-172-31-19-214 ~]$ cd workspace/
[ec2-user@ip-172-31-19-214 workspace]$ ls
deploy-slave1
[ec2-user@ip-172-31-19-214 workspace]$ cd deploy-slave1/
[ec2-user@ip-172-31-19-214 deploy-slave1]$ ls
app.py Fish Fish.csv Fish market.ipynb gitattributes one_joblib Procfile Random.pkl requirements.txt templates
[ec2-user@ip-172-31-19-214 deploy-slave1]$ sudo yum -y install python3-pip
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package python3-pip-20.2.2-1.amzn2.0.7.noarch already installed and latest version
Nothing to do
[ec2-user@ip-172-31-19-214 deploy-slave1]$ pip3 install -r requirements.txt
Defaulting to user installation because normal site-packages is not writeable
Collecting click
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting Flask
  Downloading Flask-2.2.5-py3-none-any.whl (101 kB)
Collecting importlib-metadata
  Downloading importlib_metadata-6.7.0-py3-none-any.whl (22 kB)
Collecting itsdangerous
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Jinja2
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
Collecting joblib
  Downloading joblib-1.3.2-py3-none-any.whl (302 kB)
Collecting MarkupSafe
  Downloading MarkupSafe-2.1.5-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Collecting numpy
  Downloading numpy-1.21.6-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)

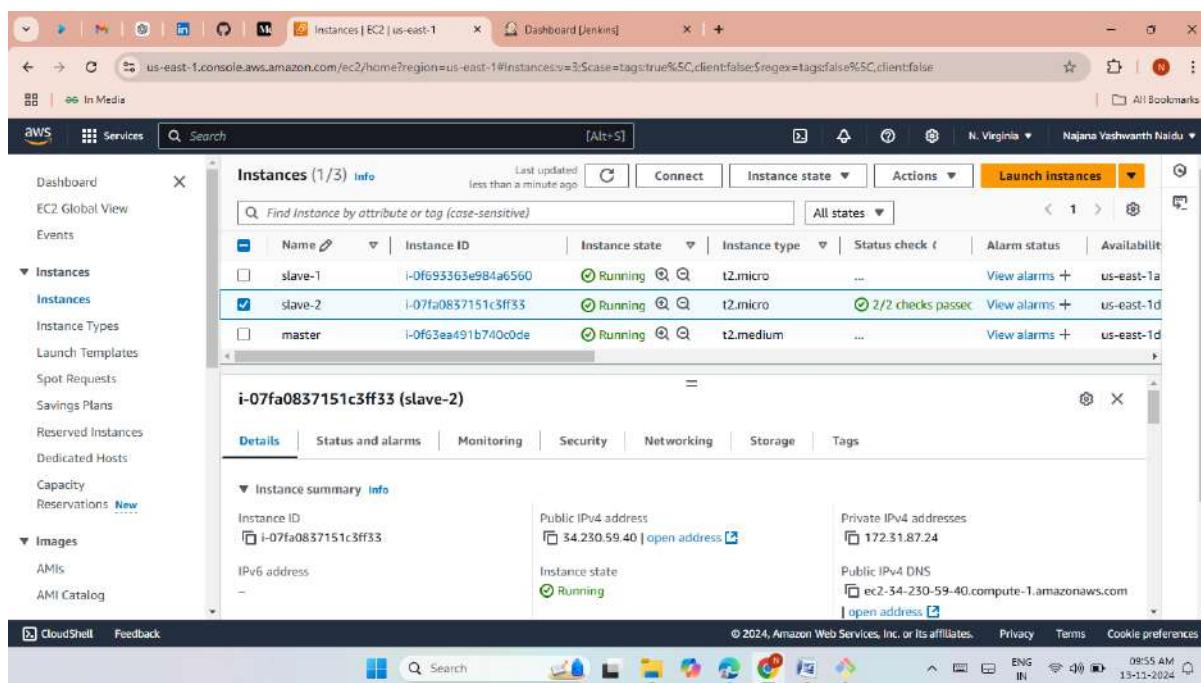
```

```
ec2-user@ip-172-31-19-214:~/workspace/deploy-slave1
[ec2-user@ip-172-31-19-214 ~]$ 15.7 MB 39.5 MB/s
Collecting pandas
  Downloading pandas-1.3.5-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.3 MB)
[ec2-user@ip-172-31-19-214 ~]$ 11.3 MB 12.5 MB/s
Collecting python-dateutil
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
[ec2-user@ip-172-31-19-214 ~]$ 229 kB 34.4 MB/s
Collecting pytz
  Downloading pytz-2024.2-py2.py3-none-any.whl (508 kB)
[ec2-user@ip-172-31-19-214 ~]$ 508 kB 37.6 MB/s
Collecting scikit-learn
  Downloading scikit_learn-1.0.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (24.8 MB)
[ec2-user@ip-172-31-19-214 ~]$ 24.8 MB 16.3 MB/s
Collecting scipy
  Downloading scipy-1.7.3-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (38.1 MB)
[ec2-user@ip-172-31-19-214 ~]$ 38.1 MB 38.8 MB/s
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting threadpoolctl
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Collecting Werkzeug
  Downloading werkzeug-2.2.3-py3-none-any.whl (233 kB)
[ec2-user@ip-172-31-19-214 ~]$ 233 kB 36.7 MB/s
Collecting zipp
  Downloading zipp-3.15.0-py3-none-any.whl (6.8 kB)
Collecting gunicorn
  Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
[ec2-user@ip-172-31-19-214 ~]$ 85 kB 8.2 MB/s
Collecting typing_extensions>=3.6.4: python_version < "3.8"
  Downloading typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Collecting packaging
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
[ec2-user@ip-172-31-19-214 ~]$ 53 kB 4.2 MB/s
Installing collected packages: typing_extensions, zipp, importlib-metadata, click, colorama, MarkupSafe, werkzeug, itsdangerous, Jinja2, Flask, joblib, numpy, six, python-dateutil, pytz, pandas, threadpoolctl, scipy, scikit-learn, packaging, gunicorn
Successfully installed Flask-2.2.5 Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-2.2.3 click-8.1.7 colorama-0.4.6 gunicorn-23.0.0 importlib-metadata-6.7.0 itsdangerous-2.1.2 joblib-1.3.2 numpy-1.21.6 packaging-24.0 pandas-1.3.5 python-dateutil-2.9.0.post0 pytz-2024.2 scikit-learn-1.0.2 scipy-1.7.3 six-1.16.0 threadpoolctl-3.1.0 typing_extensions-4.7.1 zipp-3.15.0
[ec2-user@ip-172-31-19-214 deploy-slave1]$ screen -m -d python3 app.py
[ec2-user@ip-172-31-19-214 deploy-slave1]$
```

- Now copy the slave1 public ip and past it on google browser.
 - The python application was hosted successfully.



- Now launch an ec2 instances and give name as slave2.



- Now connect Slave2 then install java.
- Now give private keys from master server.

```

cc2-user@ip-172-31-87-24:~/.ssh
verifying : libx11-common-1.6.7-3.amzn2.0.5.noarch
verifying : dejavu-sans-fonts-2.33-6.amzn2.noarch
verifying : fontconfig-2.13.0-4.3.amzn2.x86_64
Verifying : libXt-1.1.5-3.amzn2.0.2.x86_64
Verifying : giflib-4.1.6-9.amzn2.0.2.x86_64
Verifying : libXinerama-1.1.3-2.1.amzn2.0.2.x86_64
Verifying : libXi-1.7.9-1.amzn2.0.2.x86_64
Verifying : log4j-cve-2021-44228-hotpatch-1.3-7.amzn2.noarch
Verifying : libxml-1.7.1.28-6.amzn2.x86_64
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64
Verifying : python-javapackages-3.4.1-11.amzn2.noarch
Verifying : libXtst-1.2.3-1.amzn2.0.2.x86_64
Verifying : alsalib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch
Installed:
  java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1      java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1  java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1  java-17-amazon-corretto-jmods.x86_64 1:17.0.13+11-1.amzn2.1

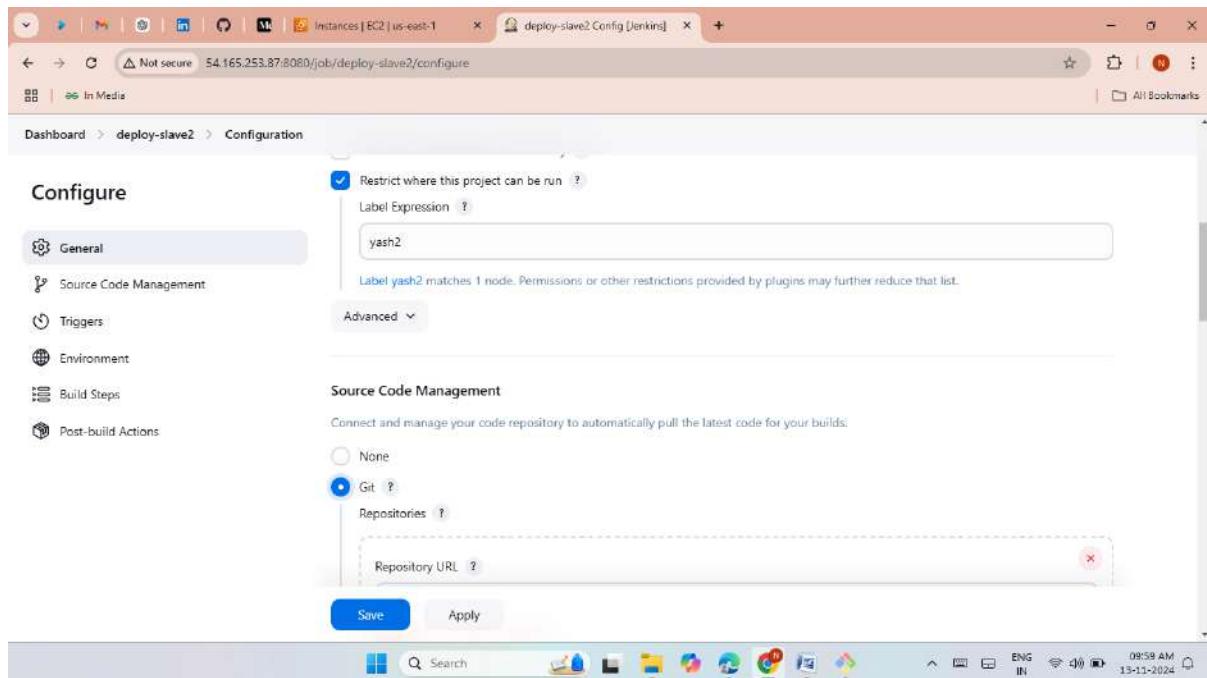
Dependency Installed:
  alsalib.x86_64 0:1.1.4.1-2.amzn2
  dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2
  fontpackages-filesystem.noarch 0:1.44-8.amzn2
  libICE.x86_64 0:1.0.9-9.amzn2.0.2
  libX11-common.noarch 0:1.6.7-3.amzn2.0.5
  libXi.x86_64 0:1.7.9-1.amzn2.0.2
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2
  libxcb.x86_64 0:1.12-1.amzn2.0.2
  python-javapackages.noarch 0:3.4.1-11.amzn2

Completed!
[ec2-user@slave2 ~]$ ls
[ec2-user@slave2 ~]$ cd .ssh/
[ec2-user@slave2 .ssh]$ ls
authorized_keys
[ec2-user@slave2 .ssh]$ sudo vi authorized_keys |
```

- Now create a node for slave2 server in jenkins.
- Give details for slave node.
- Here we have to give slave2 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave2 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	Built-In Node	Linux (amd64)	In sync	5.02 GiB	0 B	5.02 GiB	0ms
2	slave1	Linux (amd64)	In sync	4.53 GiB	0 B	4.53 GiB	26ms
3	slave2	Linux (amd64)	In sync	5.41 GiB	0 B	5.41 GiB	52ms
	Data obtained	2.6 sec	2.5 sec	2.5 sec	2.5 sec	2.5 sec	2.5 sec

- Now create a job and give slave2 label.



- Now clone the python application repo form github.
- Now install python and requirements.
- Now Click on save.

```
#!/bin/bash
```

```
sudo yum -y update
```

```
sudo yum install -y git
```

```
git clone https://github.com/Yashwanth-najana/fish.git /home/ec2-user/workspace/deploy-slave2/fish
```

```
cd /home/ec2-user/workspace/deploy-slave2/fish/
```

```
sudo yum -y install python3-pip
```

```
pip3 install -r requirements.txt
```

```
screen -m -d python3 app.py
```

The screenshot shows the Jenkins job configuration page for 'deploy-slave2'. The 'Build Steps' section is active, displaying an 'Execute shell' step. The command entered is:

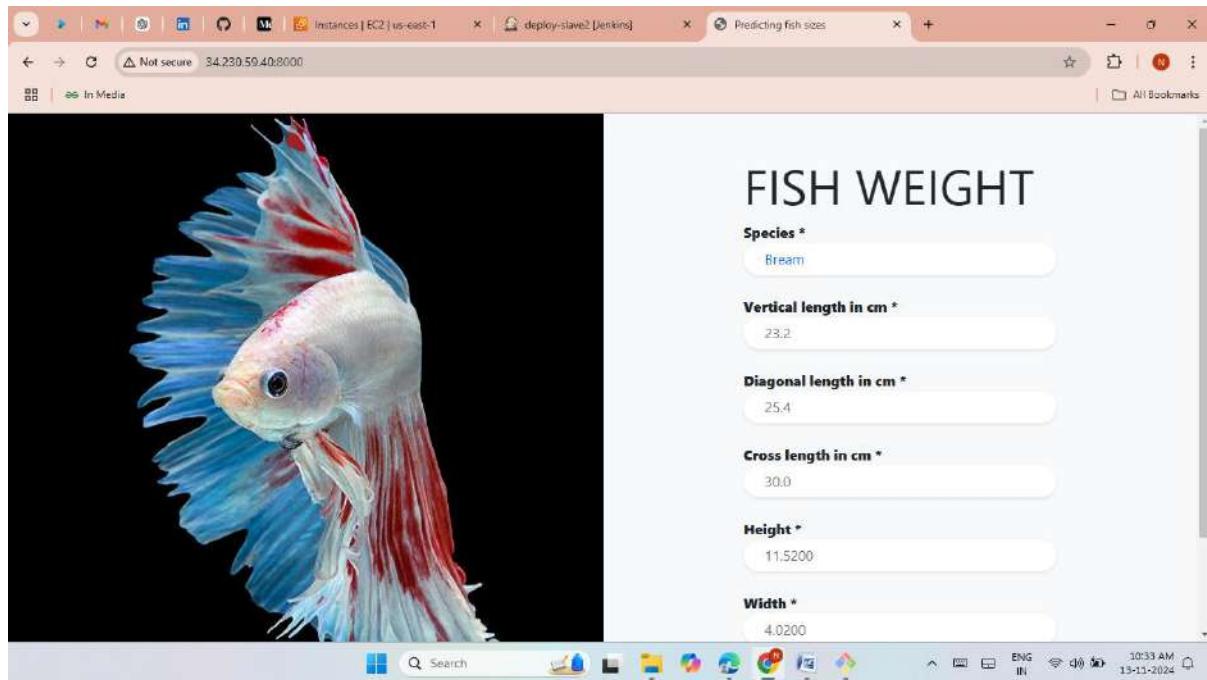
```
#!/bin/bash
sudo yum -y update
sudo yum install -y git
git clone https://github.com/Yashwanth-najana/fish.git /home/ec2-user/workspace/deploy-slave2/fish
cd /home/ec2-user/workspace/deploy-slave2/fish/
sudo yum -y install python3-pip
pip3 install -r requirements.txt
screen -m -d python3 app.py
```

Below the command, there are 'Save' and 'Apply' buttons.

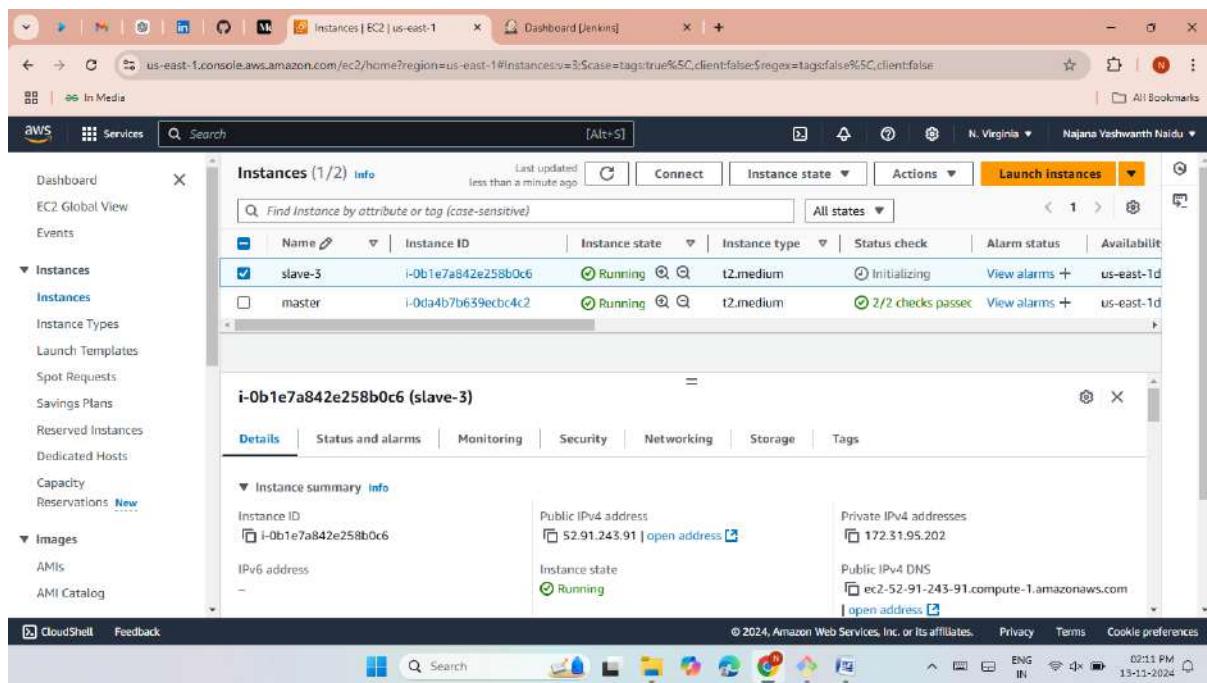
- Now click on build now.
- Here the job was success.

The screenshot shows the Jenkins job status page for 'deploy-slave2'. The job is marked as 'Success' (green checkmark). The description is 'launch python application using bash script'. The 'Permalink' section lists several builds, all of which are successful. The 'Builds' table shows one build from today at 5:02 AM. The Jenkins logo is visible in the top left, and the user 'yashwanth' is logged in.

- Now copy the slave2 public ip and past it on google browser.
- The python web application was hosted successfully.



- Now launch an ec2 instances and give name as slave3.



- Now connect Slave3 then install java.
- Now give private keys from master server.

```

Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64
Verifying : alsalib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-jmods.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsalib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libxtst.x86_64 0:1.2.3-1.amzn2.0.2
libxslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

Complete!
[ec2-user@slave3 ~]$ ls
[ec2-user@slave3 ~]$ cd .ssh/
[ec2-user@slave3 .ssh]$ ls
authorized_keys
[ec2-user@slave3 .ssh]$ sudo vi authorized_keys
[ec2-user@slave3 .ssh]$ cd
[ec2-user@slave3 ~]$ pwd
/home/ec2-user
[ec2-user@slave3 ~]$ |

```

- Now create a node for slave3 server.
- Give details for slave node.
- Here we have to give slave3 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave3 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.97 GiB	0 B	4.97 GiB	0ms
	slave3	Linux (amd64)	In sync	5.47 GiB	0 B	5.47 GiB	61ms
	Data obtained	3.9 sec	3.9 sec	3.8 sec	3.8 sec	3.8 sec	3.8 sec

- Now install the AWS credentials plugin for give access key and secret key to the job.

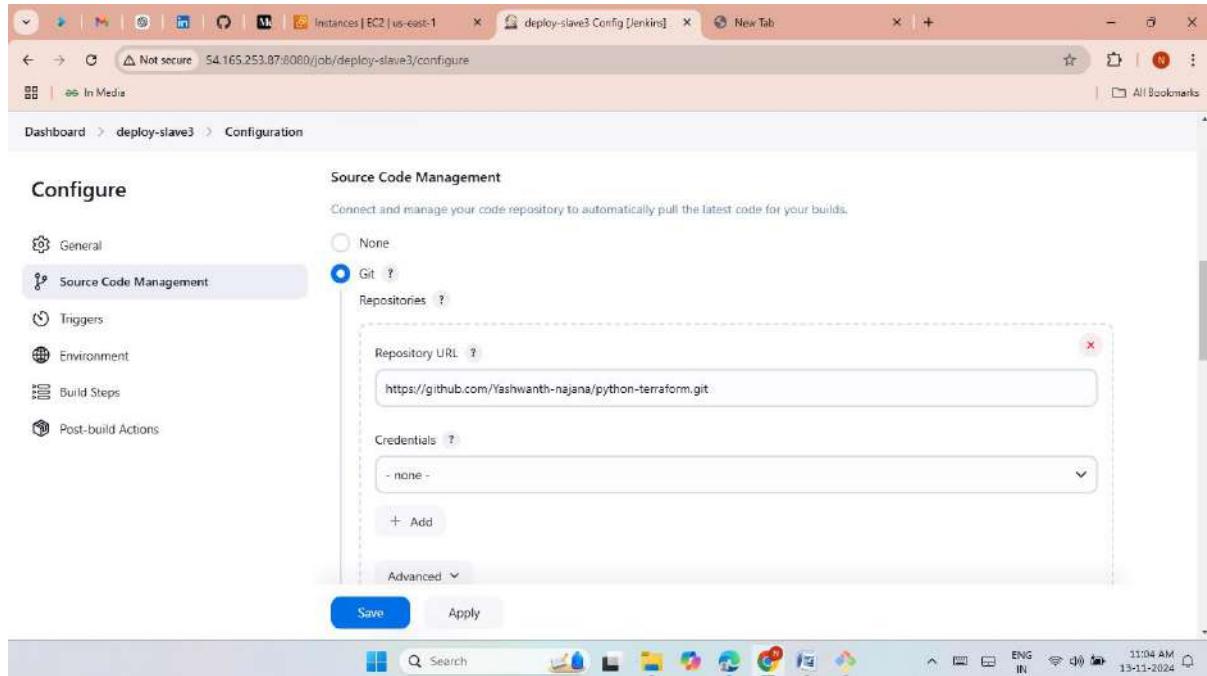
The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text 'aws'. Below it, a table lists several AWS-related plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	AWS Credentials 231.v08a_59f17d742 aws	7 mo 19 days ago
<input type="checkbox"/>	Amazon Web Services SDK :: Minimal 1.12.772-474.v7f79a_2046a_fb_ aws	29 days ago
<input type="checkbox"/>	Amazon Web Services SDK :: EC2 1.12.772-474.v7f79a_2046a_fb_ aws	29 days ago
	Amazon Web Services SDK :: FCR 1.12.772-474.v7f79a_2046a_fb_	

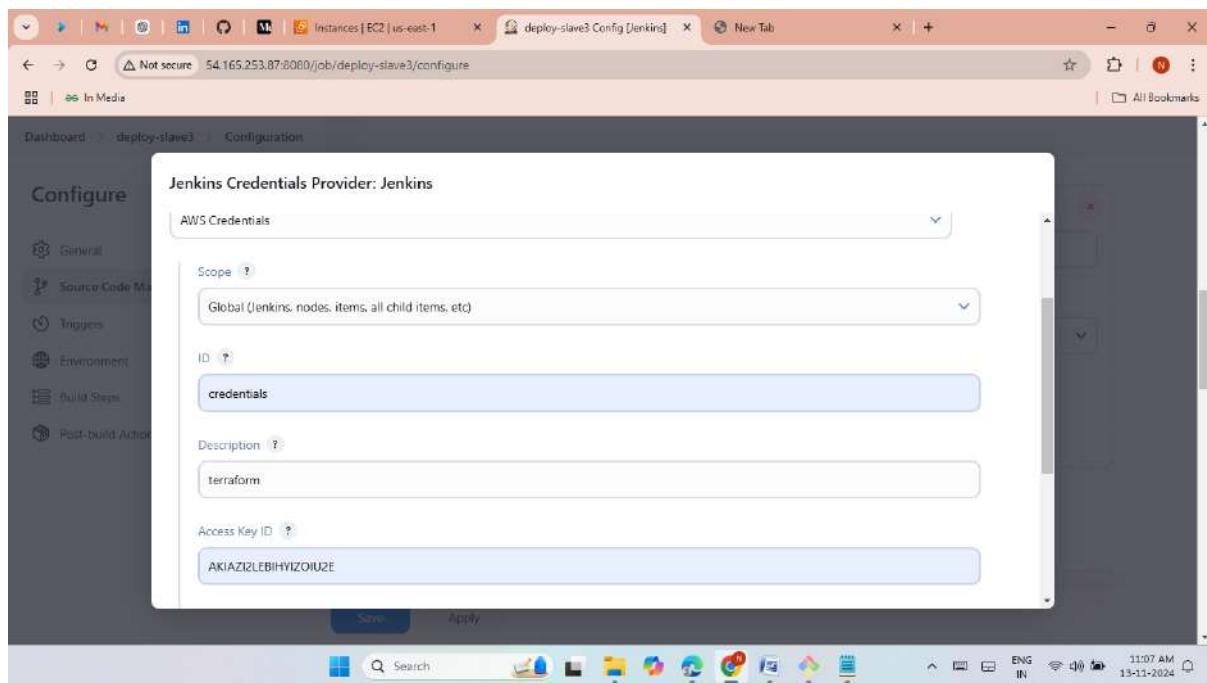
- Now create a job and give slave3 label.

The screenshot shows the Jenkins job configuration page for 'deploy-slave3'. Under the 'Configure' section, the 'General' tab is selected. In the 'Label Expression' field, the value 'yash3' is entered. A note below the field states: 'Label yash3 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' At the bottom of the page, there are 'Save' and 'Apply' buttons.

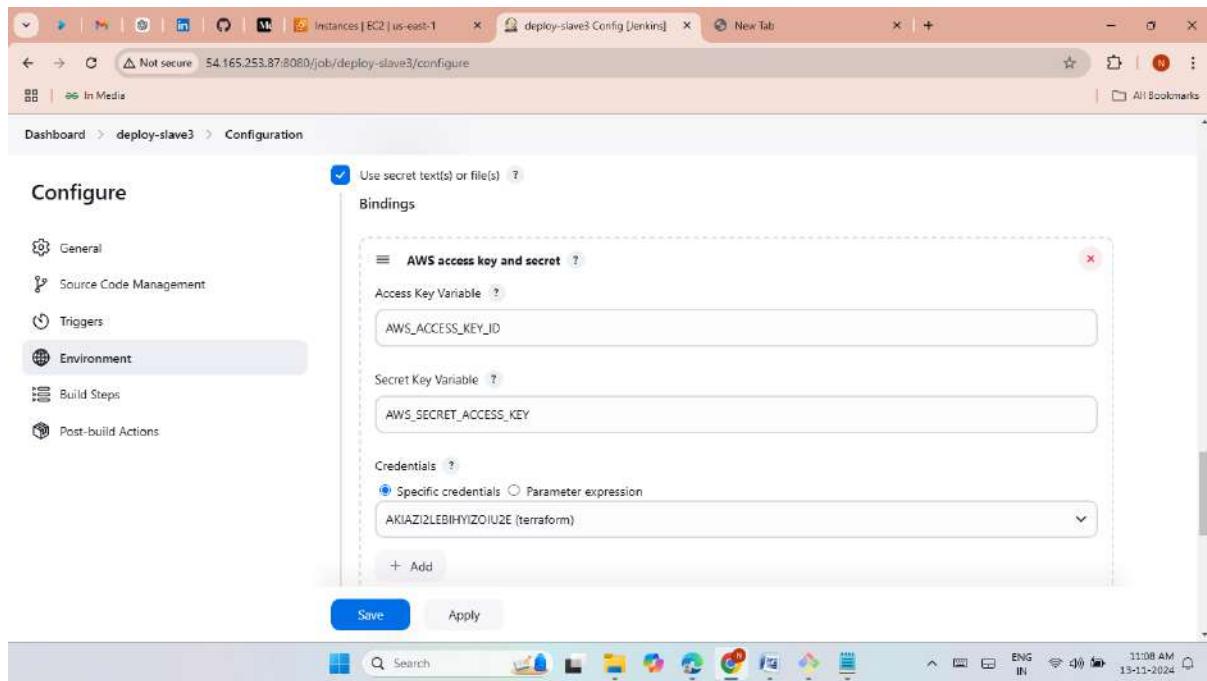
- Now clone the Terraform script for launch EC2 instance along with the python application then it creates VPC, Subnets, IGW, Route Table repo from github.



- Now create a credentials with IAM access key and secret key.



- Now give created access key and secret key.



- Now install terraform in execut shell.
- Now run the terraform with help of init, fmt, validate, apply commands.
- Now Click on save.

```
#!/bin/bash

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum install -y terraform

pwd

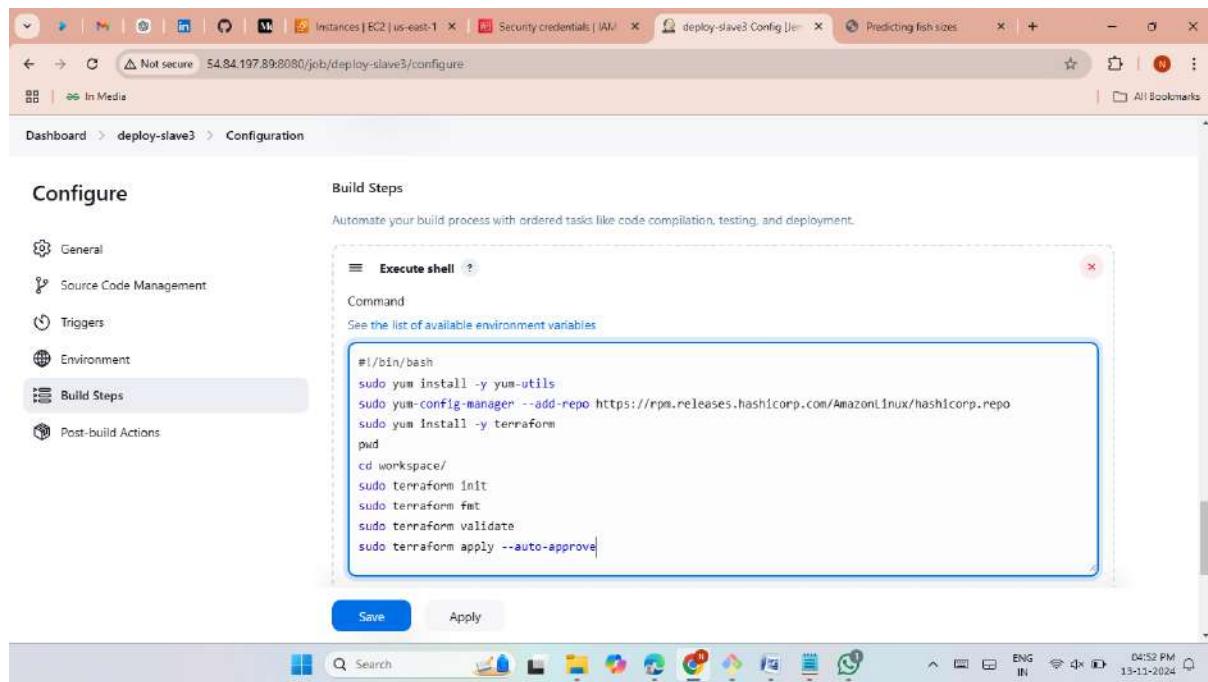
cd workspace/

sudo terraform init

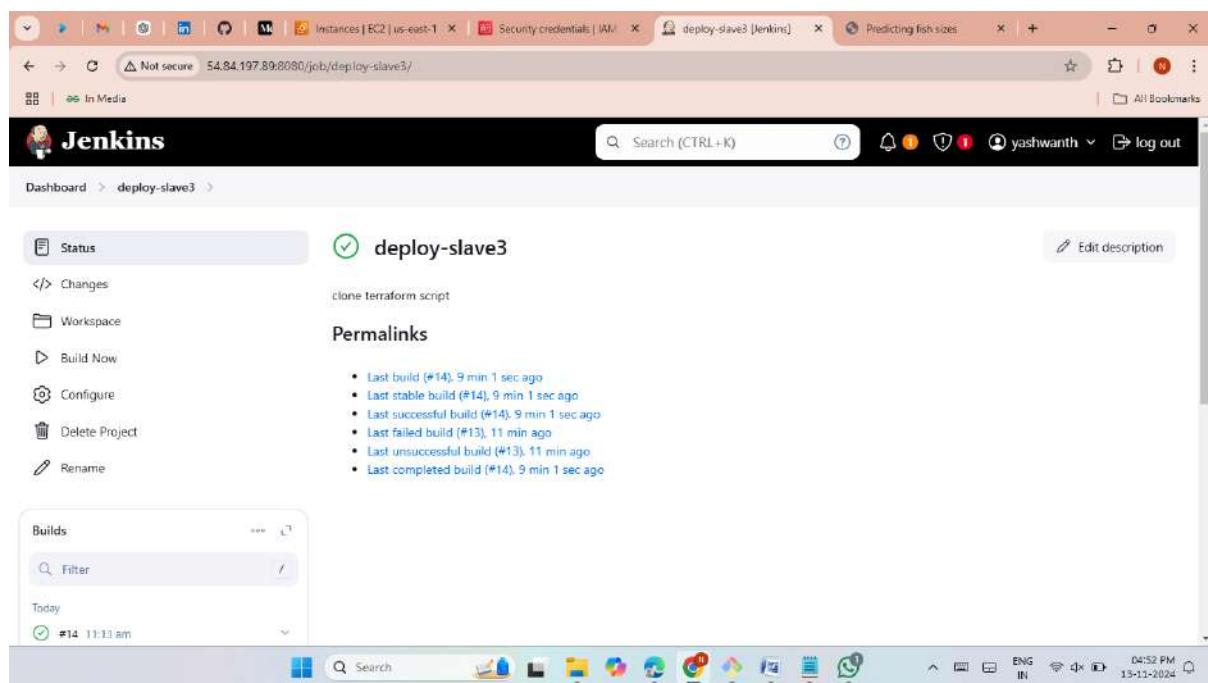
sudo terraform fmt

sudo terraform validate

sudo terraform apply --auto-approve
```



- Now click on build now.
- Here the job was success.



- Here the instance, VPC, Subnets, IGW, Route table was successfully created with help of Terraform script.

The screenshot shows the AWS Management Console with the EC2 service selected. The left sidebar shows navigation options like Dashboard, EC2 Global View, Events, Instances, Images, and CloudShell. The main pane displays the 'Instances (1/5) info' table with three entries:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
slave-3	i-0958f71df1b9a47c5	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
yash	i-088295817166ded69	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a
master	i-0114a3ead1beb63fc	Running	t2.medium	2/2 checks passed	View alarms	us-east-1d

A modal window for the instance 'yash' is open, showing its details. It includes sections for Instance summary, Networking, and Tags. The Public IPv4 address is listed as 18.212.96.252.

- Now copy the created instance public ip and past it on google browser.
- The python web application was hosted successfully.

The screenshot shows a web browser window with the URL `18.212.96.252:8000`. The page has a black header with the text 'FISH WEIGHT'. Below the header is a large image of a betta fish. To the right of the image is a form with the following fields:

- Species ***: Bream
- Vertical length in cm ***: 23.2
- Diagonal length in cm ***: 25.4
- Cross length in cm ***: 30.0
- Height ***: 11.5200
- Width ***: 4.0200

- Now launch an ec2 instances and give name as slave4.

The screenshot shows the AWS Management Console with the EC2 Instances page open. The left sidebar shows various services like Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, and Reservations. The main content area shows a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. Two instances are listed: 'slave-4' (running, t2.micro) and 'master' (running, t2.medium). Below the table, a detailed view for 'slave-4' is expanded, showing its instance ID (i-0c8e39fc408fe257c), Public IPv4 address (3.87.71.198), Private IPv4 addresses (172.31.91.221), and Public IPv4 DNS (ec2-3-87-71-198.compute-1.amazonaws.com). The bottom of the screen shows the Windows taskbar with various icons and the date/time (14-11-2024, 08:31 AM).

- Now connect Slave4 then install java.
- Now give private keys from master server.

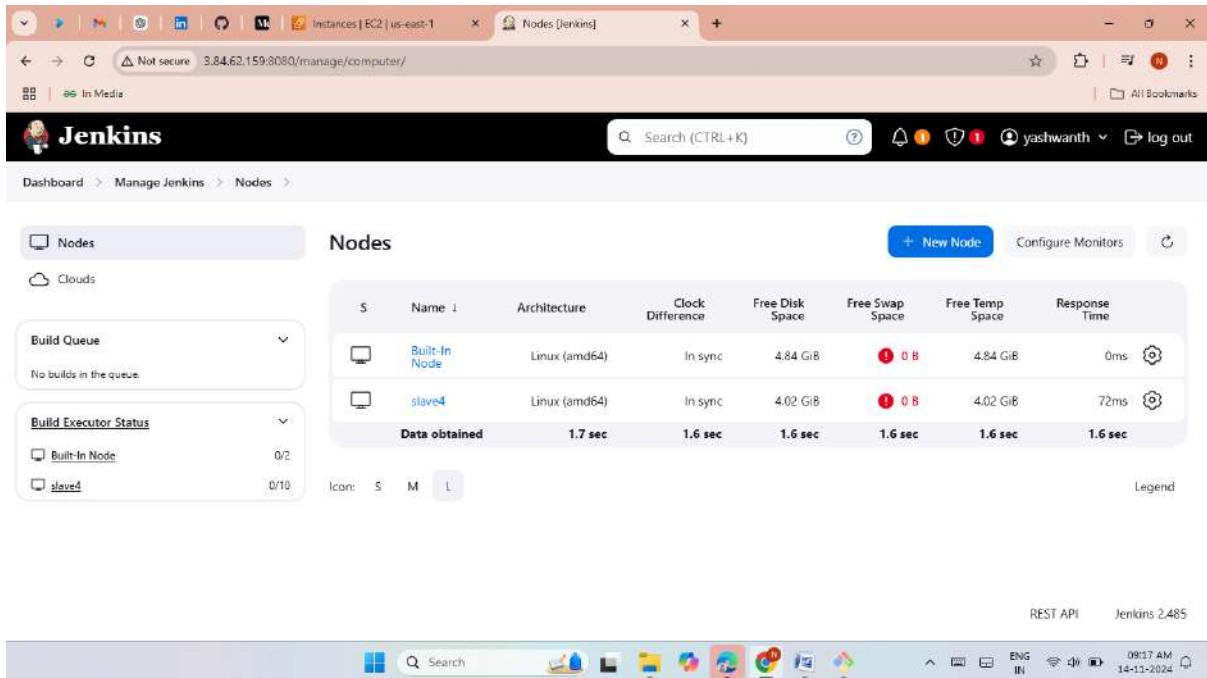
```

libcomposite.x86_64 0:0.4.4-4.1.amzn2.0.2
libdamage.x86_64 0:1.1.4-4.1.amzn2.0.2
libfixes.x86_64 0:5.0.3-1.amzn2.0.2
libXi.x86_64 0:1.7.9-1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXxf86dg.x86_64 0:1.1.4-2.1.amzn2.0.2
libXxf86vm.x86_64 0:1.1.4-1.amzn2.0.2
libfontenc.x86_64 0:1.1.3-3.amzn2.0.2
libglvnd-eql.x86_64 1:1.0.1-0.1.git5baale5.amzn2.0.1
libthai.x86_64 0:0.1.14-9.amzn2.0.2
libwayland-server.x86_64 0:1.17.0-1.amzn2.0.1
libxshmfence.x86_64 0:1.2-1.amzn2.0.2
libkctp-tools.x86_64 0:1.17-2.amzn2.0.2
mesa-libEGL.x86_64 0:18.3.4-5.amzn2.0.1
mesa-libgbm.x86_64 0:18.3.4-5.amzn2.0.1
mozsil7.x86_64 0:17.0.0-20.amzn2.0.1
pccsc-lite-libs.x86_64 0:1.8.8-7.amzn2
polkit.x86_64 0:0.112-26.amzn2.1
python-javapackages.noarch 0:3.4.1-11.amzn2
ttmkfdir.x86_64 0:3.0.9-42.amzn2.0.2
xorg-x11-font-utils.x86_64 1:7.5-21.amzn2
xorg-x11-utils.x86_64 0:7.5-23.amzn2

Complete!
[ec2-user@ip-172-31-91-221 ~]$ sudo hostname slave4
me Slave4
[ec2-user@ip-172-31-91-221 ~]$ exec bash
[ec2-user@Slave4 ~]$ cd .ssh/
[ec2-user@Slave4 .ssh]$ ls
authorized_keys
[ec2-user@Slave4 .ssh]$ sudo vi authorized_keys
[ec2-user@Slave4 .ssh]$ cd
[ec2-user@Slave4 ~]$ pwd
/home/ec2-user
[ec2-user@Slave4 ~]$ |

```

- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave4 node is successfully created.

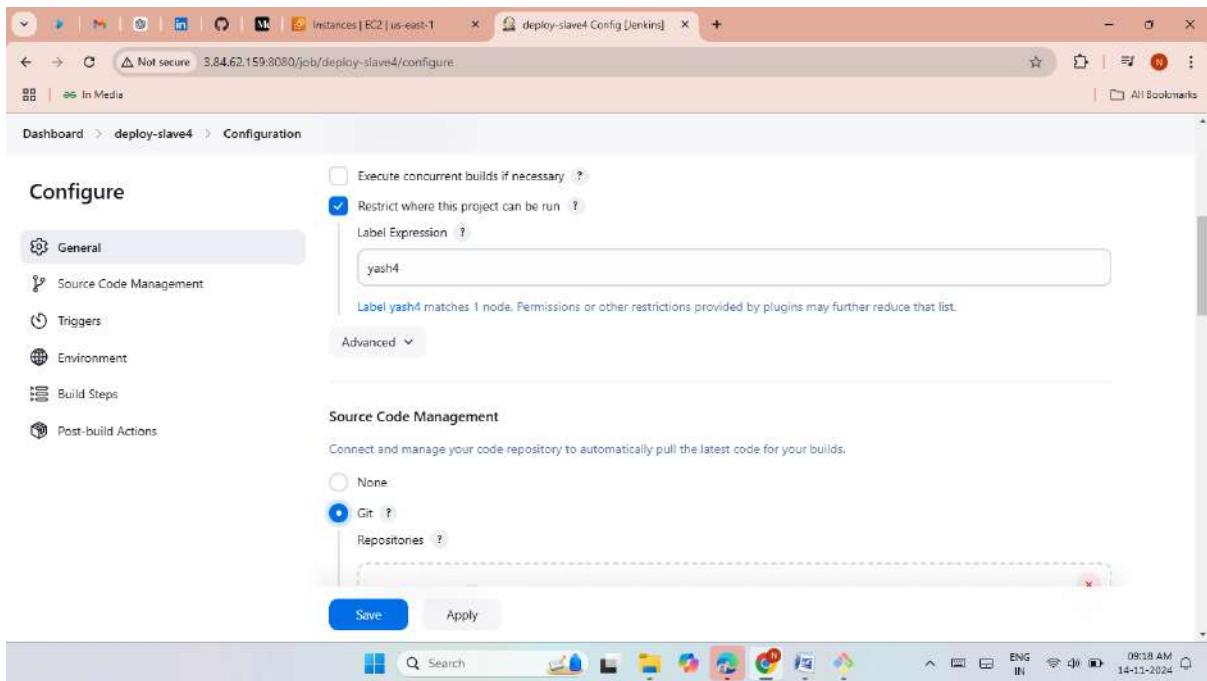


The screenshot shows the Jenkins 'Nodes' page. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (Built-In Node: 0/2, slave4: 0/10). The main area displays a table of nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.84 GiB	0 B	4.84 GiB	0ms
	slave4	Linux (amd64)	In sync	4.02 GiB	0 B	4.02 GiB	72ms
	Data obtained		1.7 sec	1.6 sec	1.6 sec	1.6 sec	1.6 sec

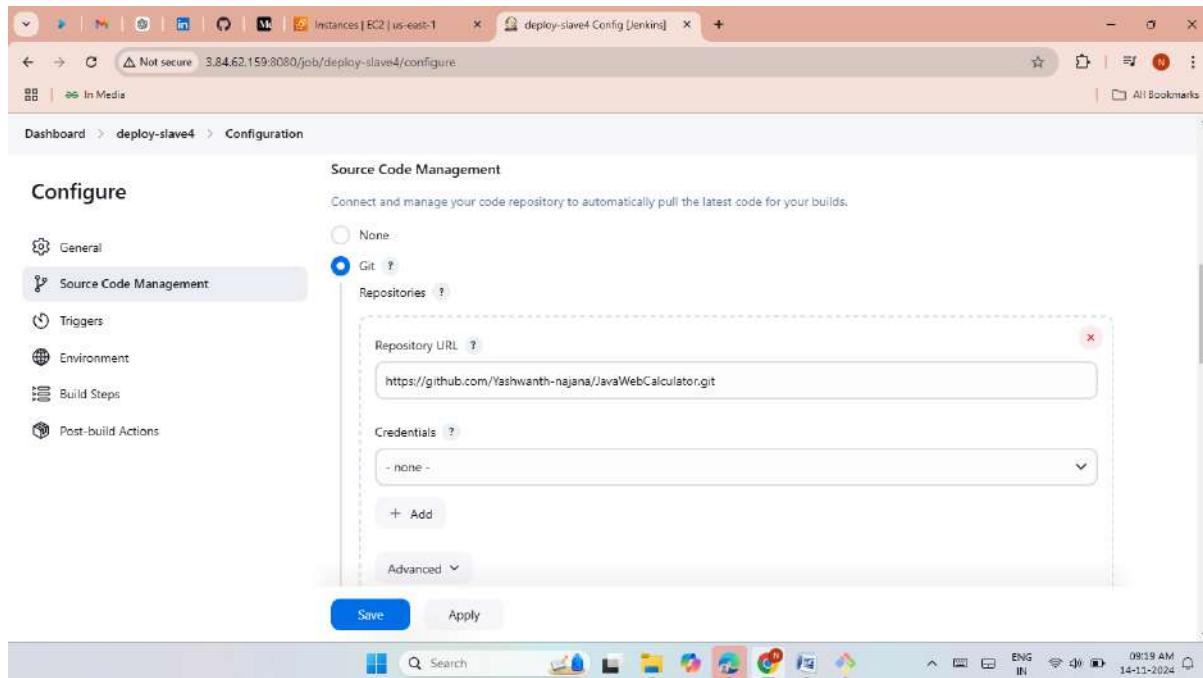
At the bottom right of the table, there is a 'Legend' section with icons for 'S' (Slave), 'M' (Master), and 'L' (Label). The Jenkins header includes 'Dashboard', 'Manage Jenkins', 'Nodes', 'yashwanth', and 'log out'. The browser address bar shows 'Not secure 3.84.62.159:8080/manage/computer/'.

- Now create a job and give slave4 label.

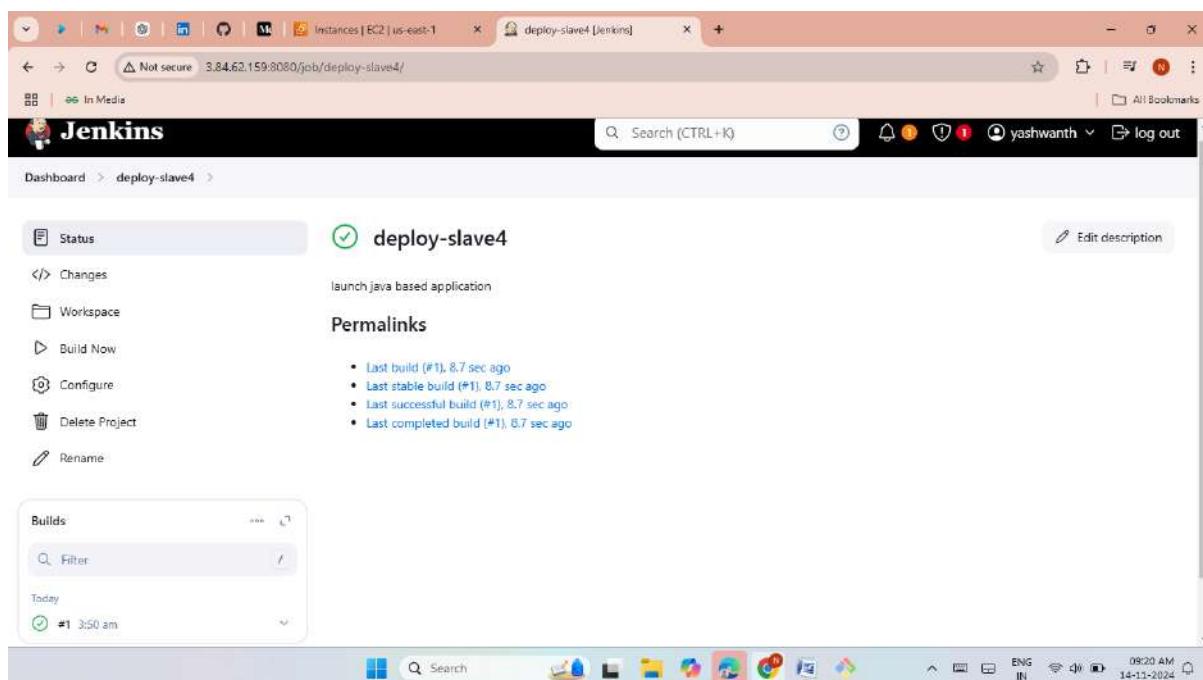


The screenshot shows the Jenkins 'Configuration' page for the 'deploy-slave4' job. The 'General' section is selected. Under 'Restrict where this project can be run', the 'Label Expression' field contains 'yash4'. A note below states: 'Label yash4 matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' The 'Source Code Management' section shows 'Git' selected with 'None' as the repository. At the bottom, there are 'Save' and 'Apply' buttons. The Jenkins header includes 'Dashboard', 'deploy-slave4', 'Configuration', 'yashwanth', and 'log out'. The browser address bar shows 'Not secure 3.84.62.159:8080/job/deploy-slave4/configure'.

- Now clone the java application repo from github.
- Now Click on save.



- Now click on build now.
- Here the job was success.



- Now launch java applications manually with help of some commands
- Install maven with the command.

sudo yum -y install maven

```
cc2-user@ip-172-31-91-221:~/workspace/deploy-slave4
[ec2-user@slave4 ~]$ ls
remoting remoting.jar workspace
[ec2-user@slave4 ~]$ cd workspace/
[ec2-user@slave4 workspace]$ ls
deploy-slave4
[ec2-user@slave4 workspace]$ cd deploy-slave4/
[ec2-user@slave4 deploy-slave4]$ ls
pom.xml  src
[ec2-user@slave4 deploy-slave4]$ sudo yum -y install maven
```

- Now package the application with the command
- Mvn package

```
cc2-user@ip-172-31-91-221:~/workspace/deploy-slave4
nekohtml.noarch 0:1.9.14-13.amzn2
plexus-cipher.noarch 0:1.7-5.amzn2
plexus-component-api.noarch 0:1.0-0.16.alpha15.amzn2
plexus-containers-container-default.noarch 0:1.5.5-14.amzn2
plexus-interpolation.noarch 0:1.15-8.amzn2
plexus-utils.noarch 0:3.0.9-9.amzn2
regexp.noarch 0:1.5-13.amzn2
sisu-inject-plexus.noarch 0:2.3.0-11.amzn2
tomcat-servlet-3.0-api.noarch 0:7.0.76-10.amzn2.0.9
xbean.noarch 0:3.13-6.amzn2
xml-commons-apis.noarch 0:1.4.01-16.amzn2
objectweb-asm.noarch 0:3.3.1-9.amzn2
plexus-classworlds.noarch 0:2.4.2-8.amzn2
plexus-containers-component-annotations.noarch 0:1.5.5-14.amzn2
plexus-interactivity.noarch 0:1.0-0.14.alpha6.amzn2
plexus-sec-dispatcher.noarch 0:1.4-13.amzn2
qdox.noarch 0:1.12.1-10.amzn2
sisu-inject-bean.noarch 0:2.3.0-11.amzn2
sif4j.noarch 0:1.7.4-4.amzn2
xalan-j2.noarch 0:2.7.1-23.1.amzn2
xerces-j2.noarch 0:2.11.0-17.amzn2.0.2
xml-commons-resolver.noarch 0:1.2-15.amzn2

Complete!
[ec2-user@slave4 deploy-slave4]$ mvn
[INFO] Scanning for projects...
[INFO]
[INFO] BUILD FAILURE
[INFO]
[INFO] Total time: 0.123s
[INFO] Finished at: Thu Nov 14 03:53:10 UTC 2024
[INFO] Final Memory: 4M/15M
[INFO]
[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format <plugin-prefix>:<goal> or <plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: validate, initialize, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes, generate-test-sources, process-test-sources, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare-package, package, pre-integration-test, integration-test, post-integration-test, verify, install, deploy, pre-clean, clean, post-clean, pre-site, site, post-site, site-deploy. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoGoalsSpecifiedException
[ec2-user@slave4 deploy-slave4]$ mvn package
```

```

ec2-user@ip-172-31-91-221:~/workspace/deploy-slave4/target
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/3.2.0/plexus-io-3.2.0.jar (74 KB at 2108.2 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.8/xz-1.8.jar (107 KB at 4609.2 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.jar (210 KB at 4195.1 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/commons-io/2.6/commons-io-2.6.jar (210 KB at 4195.1 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/3.1.1/maven-filtering-3.1.1.jar (50 KB at 261.9 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.26/plexus-interpolation-1.26.jar (84 KB at 3788.5 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-mapping/3.0.0/maven-mapping-3.0.0.jar (11 KB at 797.6 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.3.0/plexus-utils-3.3.0.jar (258 KB at 7345.2 KB/sec)
[INFO] Packaging webapp
[INFO] Assembling webapp [webapp] in [/home/ec2-user/workspace/deploy-slave4/target/webapp-1.8]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/ec2-user/workspace/deploy-slave4/src/main/webapp]
[INFO] Building war: /home/ec2-user/workspace/deploy-slave4/target/webapp-1.8.war
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 9.087s
[INFO] Finished at: Thu Nov 14 04:41:19 UTC 2024
[INFO] Final Memory: 16M/88M
[INFO]
[ec2-user@slave4 deploy-slave4]$ ls
pom.xml  src  target
[ec2-user@slave4 deploy-slave4]$ cd target/
[ec2-user@slave4 target]$ ls
classes  generated-test-sources  maven-status  surefire-reports  webapp-1.8
generated-sources  maven-archiver  surefire  test-classes  webapp-1.8.war
[ec2-user@slave4 target]$
```

➤ Now install tomcat tar file from Google browser.

➤ Unzip the tomcat tar file with the command.

Sudo tar -xvf <file name>

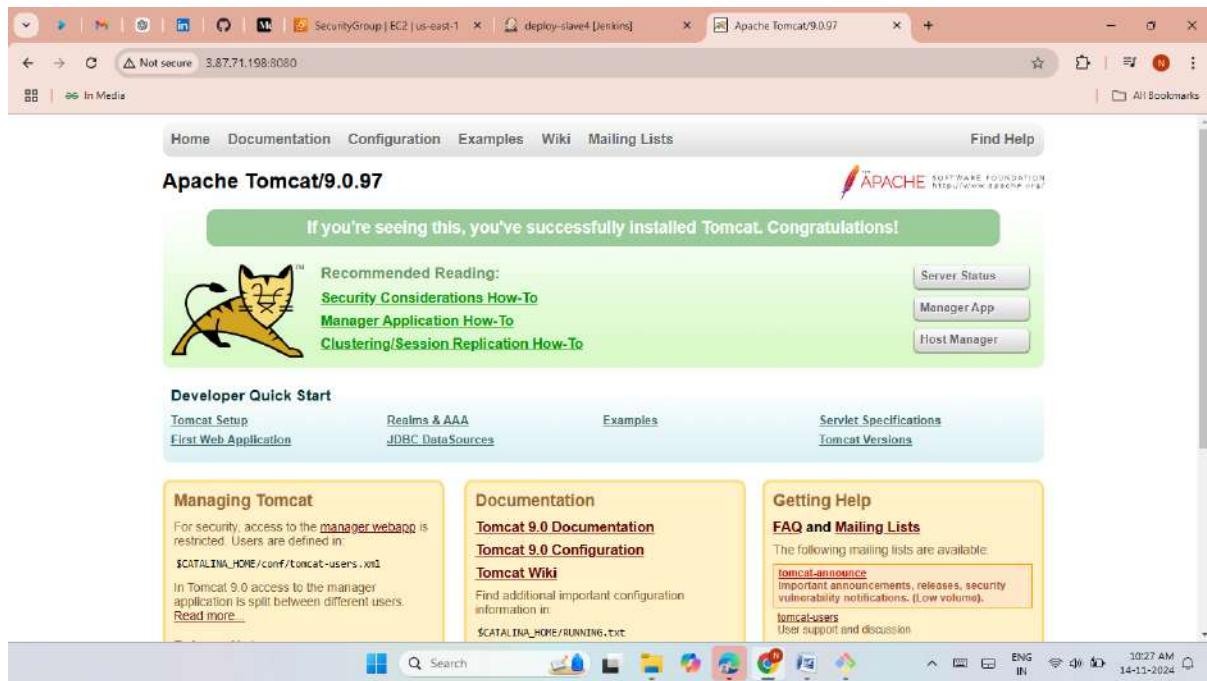
➤ Now start the tomcat with the command.

./startup.sh

```

ec2-user@ip-172-31-91-221:~/workspace/apache-tomcat-9.0.97/bin
/home/ec2-user/workspace/apache-tomcat-9.0.97/bin/catalina.sh: Line 504: /home/ec2-user/workspace/apache-tomcat-9.0.97/logs/catalina.out: Permission denied
[ec2-user@slave4 bin]$ cd
[ec2-user@slave4 ~]$ ls -ld /home/ec2-user/workspace/apache-tomcat-9.0.97/logs
drwxr-x--- 2 root root 6 Nov 6 19:55 /home/ec2-user/workspace/apache-tomcat-9.0.97/logs
[ec2-user@slave4 ~]$ sudo chmod 755 /home/ec2-user/workspace/apache-tomcat-9.0.97/logs
[ec2-user@slave4 ~]$ sudo chown -R ec2-user:ec2-user /home/ec2-user/workspace/apache-tomcat-9.0.97/logs
[ec2-user@slave4 ~]$ cd workspace/
[ec2-user@slave4 workspace]$ cd deploy-slave4/
[ec2-user@slave4 deploy-slave4]$ ls
pom.xml  src  target
[ec2-user@slave4 deploy-slave4]$ cd ..
[ec2-user@slave4 workspace]$ cd ..
[ec2-user@slave4 ~]$ ls
remoting  remoting.jar  workspace
[ec2-user@slave4 ~]$ cd workspace/
[ec2-user@slave4 workspace]$ ls
apache-tomcat-9.0.97.tar.gz  deploy-slave4
[ec2-user@slave4 workspace]$ cd apache-tomcat-9.0.97/
[ec2-user@slave4 apache-tomcat-9.0.97]$ ls
BUILDING.txt  conf  CONTRIBUTING.md  lib  LICENSE  logs  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  temp  webapps  work
[ec2-user@slave4 apache-tomcat-9.0.97]$ cd bin/
[ec2-user@slave4 bin]$ ls
bootstrap.jar  ciphers.sh  daemon.sh  setclasspath.bat  startup.sh  version.bat
catalina.bat  commons-daemon.jar  digest.bat  setclasspath.sh  tomcat-juli.jar  version.sh
catalina.sh  commons-daemon-native.tar.gz  digest.sh  shutdown.bat  tomcat-native.tar.gz
catalina-tasks.xml  configtest.bat  makebase.bat  shutdown.sh  tool-wrapper.bat
ciphers.bat  configtest.sh  makebase.sh  startup.bat  tool-wrapper.sh
[ec2-user@slave4 bin]$ ./startup.sh
Using CATALINA_BASE:  /home/ec2-user/workspace/apache-tomcat-9.0.97
Using CATALINA_HOME:  /home/ec2-user/workspace/apache-tomcat-9.0.97
Using CATALINA_TMPDIR: /home/ec2-user/workspace/apache-tomcat-9.0.97/temp
Using JRE_HOME:      /usr
Using CLASSPATH:     /home/ec2-user/workspace/apache-tomcat-9.0.97/bin/bootstrap.jar:/home/ec2-user/workspace/apache-tomcat-9.0.97/b
in/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
[ec2-user@slave4 bin]$
```

- Here the tomcat was hosted successfully.



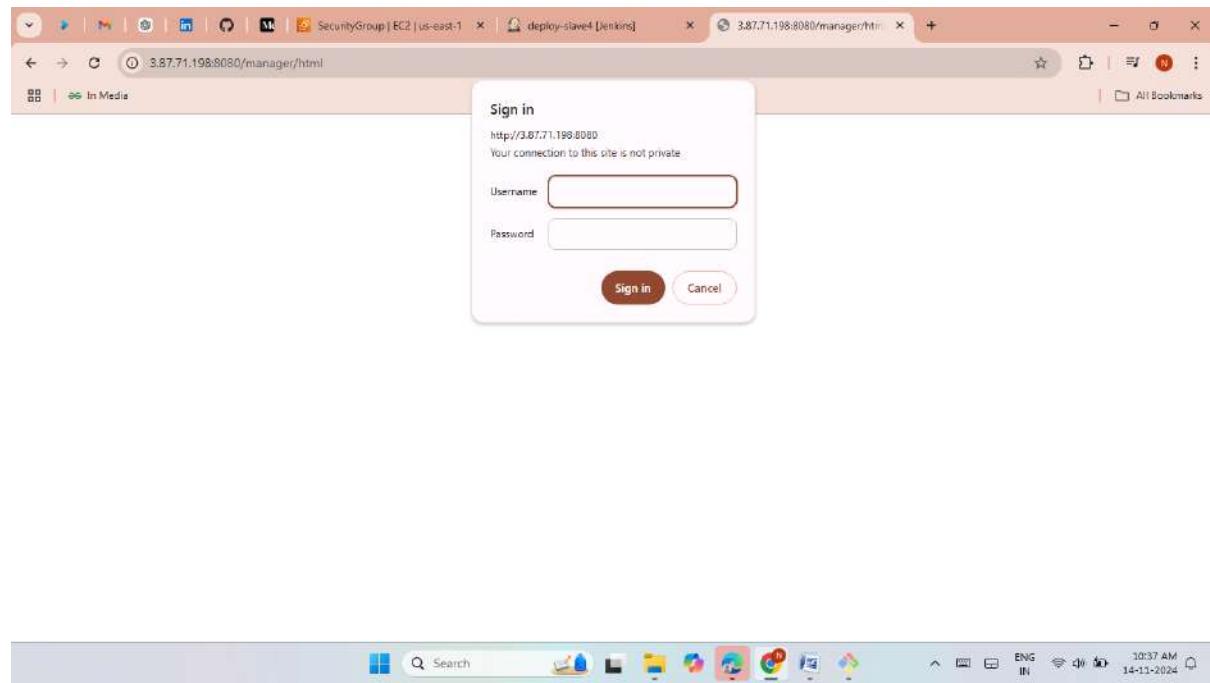
- Now copy the .war files from target to webapps.

```
sudo mv *.war /home/ec2-user/workspace/deploy-slave5/apache-tomcat-9.0.97/webapps/
```

```
[ec2-user@ip-172-31-91-221:~/workspace/apache-tomcat-9.0.97/webapps]
[ec2-user@slave4 ~]$ ls
remoting remotng.jar workspace
[ec2-user@slave4 workspace]$ ls
apache-tomcat-9.0.97.tar.gz deploy-slave4
[ec2-user@slave4 workspace]$ cd deploy-slave4/
[ec2-user@slave4 deploy-slave4]$ ls
pom.xml src target
[ec2-user@slave4 deploy-slave4]$ cd target/
[ec2-user@slave4 target]$ ls
apache-tomcat-9.0.97.tar.gz.asc generated-sources maven-archiver surefire test-classes webapp-1.8.war
classes generated-test-sources maven-status surefire-reports webapp-1.8
[ec2-user@slave4 target]$ sudo mv *.war /home/ec2-user/workspace/apache-tomcat-9.0.97/webapps/
[ec2-user@slave4 target]$ ls
apache-tomcat-9.0.97.tar.gz.asc generated-sources maven-archiver surefire test-classes
classes generated-test-sources maven-status surefire-reports webapp-1.8
[ec2-user@slave4 target]$ cd ..
[ec2-user@slave4 deploy-slave4]$ ls
[ec2-user@slave4 deploy-slave4]$ cd ..
[ec2-user@slave4 workspace]$ ls
apache-tomcat-9.0.97.tar.gz deploy-slave4
[ec2-user@slave4 workspace]$ cd apache-tomcat-9.0.97/
[ec2-user@slave4 apache-tomcat-9.0.97]$ ls
> /c
[ec2-user@slave4 apache-tomcat-9.0.97]$ ls
BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@slave4 apache-tomcat-9.0.97]$ cd w
bash: cd: w: No such file or directory
[ec2-user@slave4 apache-tomcat-9.0.97]$ cd webapps/
bash: cd: webapps/: Permission denied
[ec2-user@slave4 apache-tomcat-9.0.97]$ sudo chmod 777 webapps/
[ec2-user@slave4 apache-tomcat-9.0.97]$ ls
BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@slave4 apache-tomcat-9.0.97]$ cd webapps/
[ec2-user@slave4 webapps]$ ls
docs examples host-manager manager ROOT webapp-1.8 webapp-1.8.war
[ec2-user@slave4 webapps]$
```

- Now go to host-manager and manager context file then remove some permission.

- Click on manager apps in tomcat the pages was changed.



- Now give role name, username and password in tomcat-users.xml file.

```
ec2-user@ip-172-31-91-221:~/workspace/apache-tomcat-9.0.97/conf
[ec2-user@slave4 apache-tomcat-9.0.97]$ ls
BUILDING.txt  CONTRIBUTING.md  lib  LICENSE  logs  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  temp  work
[ec2-user@slave4 apache-tomcat-9.0.97]$ cd conf/
[ec2-user@slave4 conf]$ ls
Catalina  catalina.properties  jaspic-providers.xml  logging.properties  tomcat-users.xml  web.xml
catalina.policy  context.xml  jaspic-providers.xsd  server.xml  tomcat-users.xsd
[ec2-user@slave4 conf]$ sudo vi tomcat-users.xml
[ec2-user@slave4 conf]$
```

```

cc2-user@ip-172-31-91-221:~/workspace/apache-tomcat-9.0.97/conf
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary.

Built-in Tomcat manager roles:
- manager-gui - allows access to the HTML GUI and the status pages
- manager-script - allows access to the HTTP API and the status pages
- manager-jmx - allows access to the JMX proxy and the status pages
- manager-status - allows access to the status pages only

The users below are wrapped in a comment and are therefore ignored. If you
wish to configure one or more of these users for use with the manager web
application, do not forget to remove the <!-- ... --> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="" roles="manager-gui"/>
<user username="robot" password="" roles="manager-script"/>
-->
<!--
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- ... --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->
<role rolename="manager-gui"/>
<user username="yash" password="yash" roles="manager-gui"/>
</tomcat-users>
~ 
-- INSERT --

```

57,59 Bot

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with various icons. Below the taskbar, a terminal window is open, displaying configuration code for Tomcat's `tomcat-users.xml` file. The code includes built-in manager roles and a sample user named 'yash' with role 'manager-gui'. Below the terminal, a web browser window is open to the URL <http://3.87.71.198:8080/manager/html>. The browser title bar says '/manager'. The page itself is the 'Tomcat Web Application Manager' interface, showing a list of applications deployed to the server, including 'Welcome to Tomcat', 'Tomcat Documentation', 'Servlet and JSP Examples', and 'Tomcat Host Manager Application'. The Apache Software Foundation logo is visible in the top right corner of the browser window.

➤ Now click on webapps-1.8.

The screenshot shows the Tomcat Manager interface. The 'Deployments' section lists several applications:

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/webapp-1.8	None specified	Servlet	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

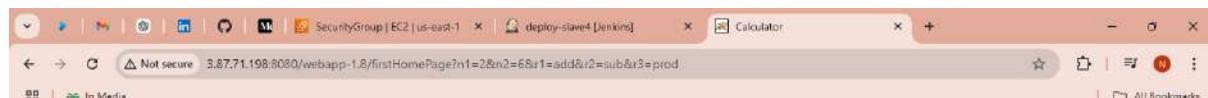
The 'Deploy' section contains fields for Context Path, Version (for parallel deployment), XML Configuration file path, and WAR or Directory path. A 'Deploy' button is present. The URL in the address bar is <http://3.87.71.198:8080/manager/html#/>.

➤ The java web calculator was hosted successfully.

The screenshot shows a Java web application titled 'Calculator'. The form contains the following fields:

- first number:
- Second number:
- Sravani-Addition
- Mohammed-subraction
- Vimod-Multiplication
-

The URL in the address bar is <http://3.87.71.198:8080/webapp-1.8/>.



Addition

8

Subtraction

4

Multiplication

12

Calculator

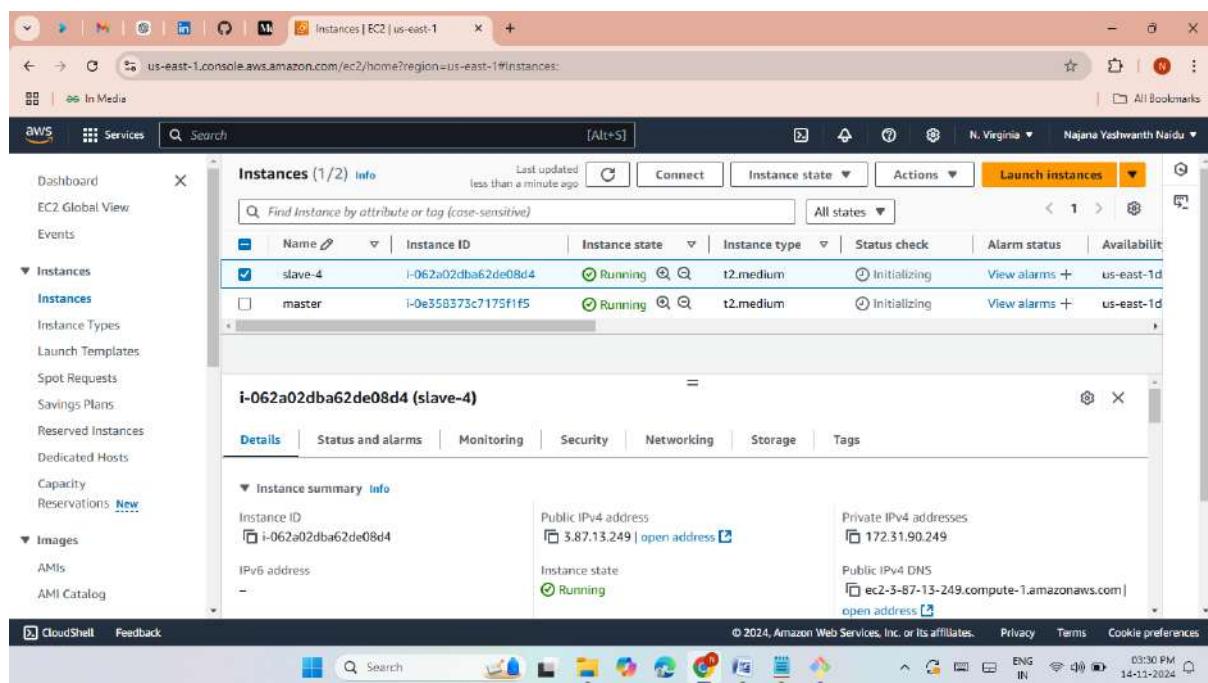
first number:

Second number:

Sravani-Addition
 Mohammed-subtraction
 Vinod-Multiplication



- Here I have done another method in slave4.
- Now launch an ec2 instances and give name as slave4.



- Now connect Slave4 then install java.
- Now give private keys from master server.

```

ec2-user@ip-172-31-90-249:~$ 
Verifying : python-javapackages-3.4.1-11.amzn2.noarch          27/32
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64                  28/32
Verifying : alsalib-1.1.4-1-2.amzn2.x86_64                   29/32
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch      30/32
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64                 31/32
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch          32/32

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libxtst.x86_64 0:1.2.3-1.amzn2.0.2
libxsdt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

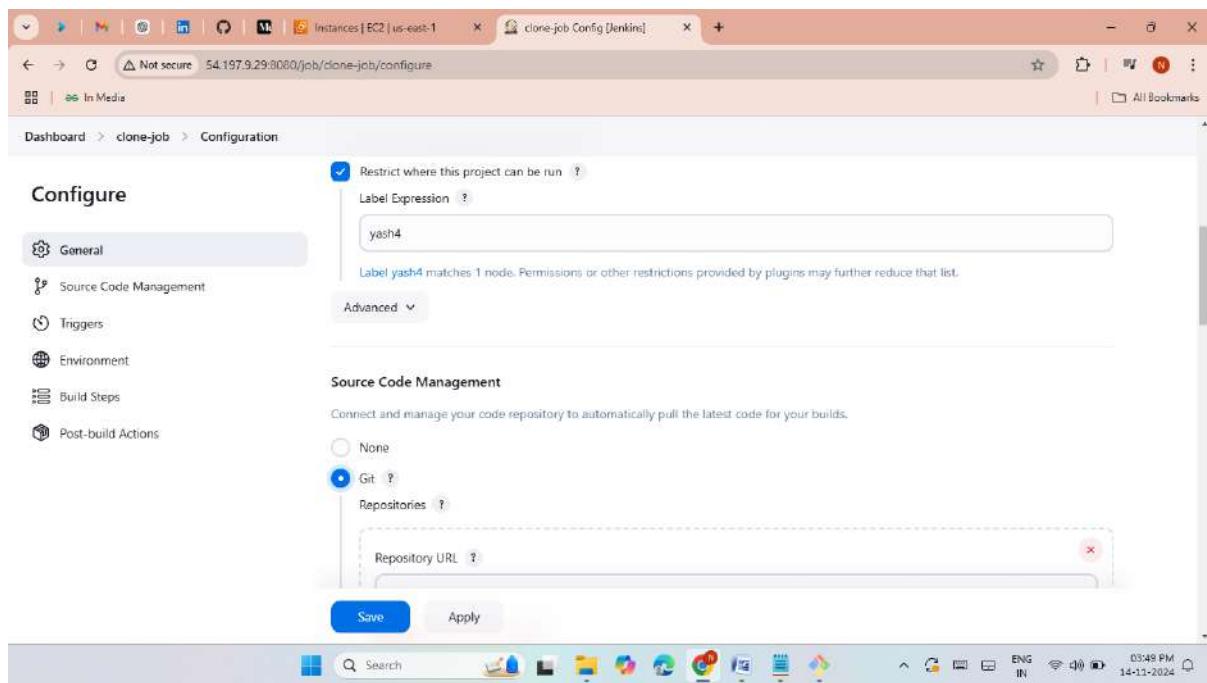
Completed!
[ec2-user@ip-172-31-90-249 ~]$ sudo hostname slave4
[ec2-user@ip-172-31-90-249 ~]$ exec bash
[ec2-user@slave4 ~]$ cd .ssh/
[ec2-user@slave4 .ssh]$ ls
authorized_keys
[ec2-user@slave4 .ssh]$ sudo vi authorized_keys
[ec2-user@slave4 .ssh]$ cd
[ec2-user@slave4 ~]$ pwd
/home/ec2-user
[ec2-user@slave4 ~]$ 

```

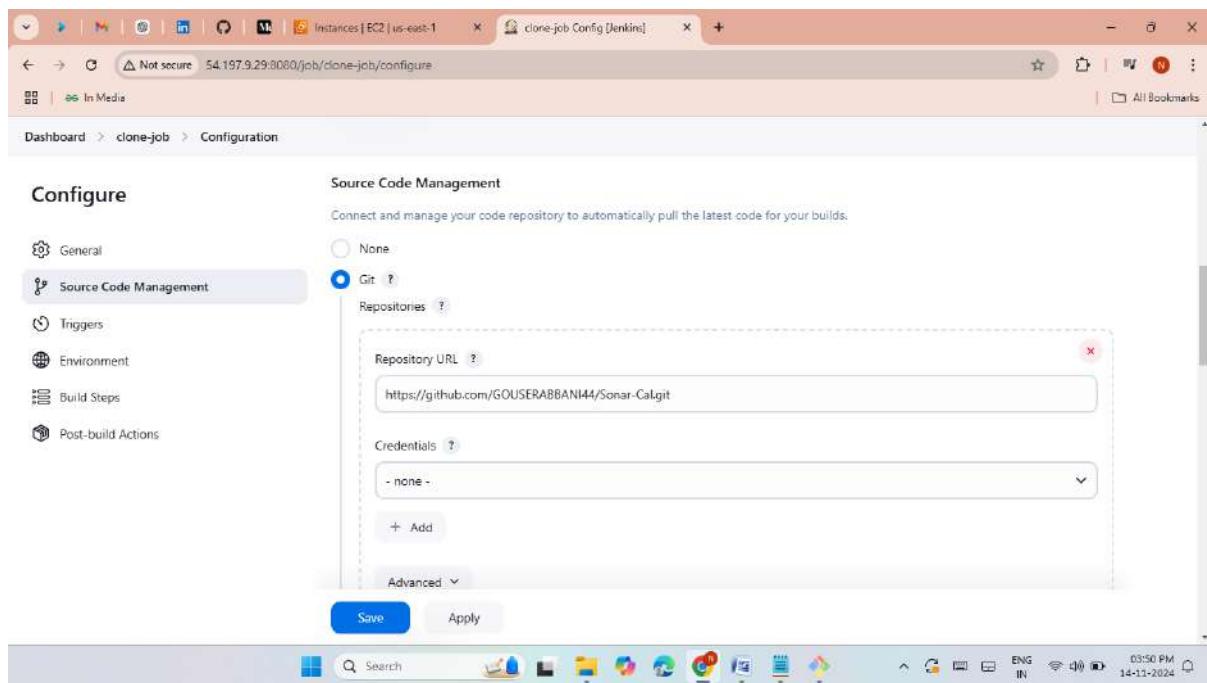
- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave4 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.64 GiB	0 B	4.64 GiB	0ms
	slave4	Linux (amd64)	In sync	5.23 GiB	0 B	5.23 GiB	53ms

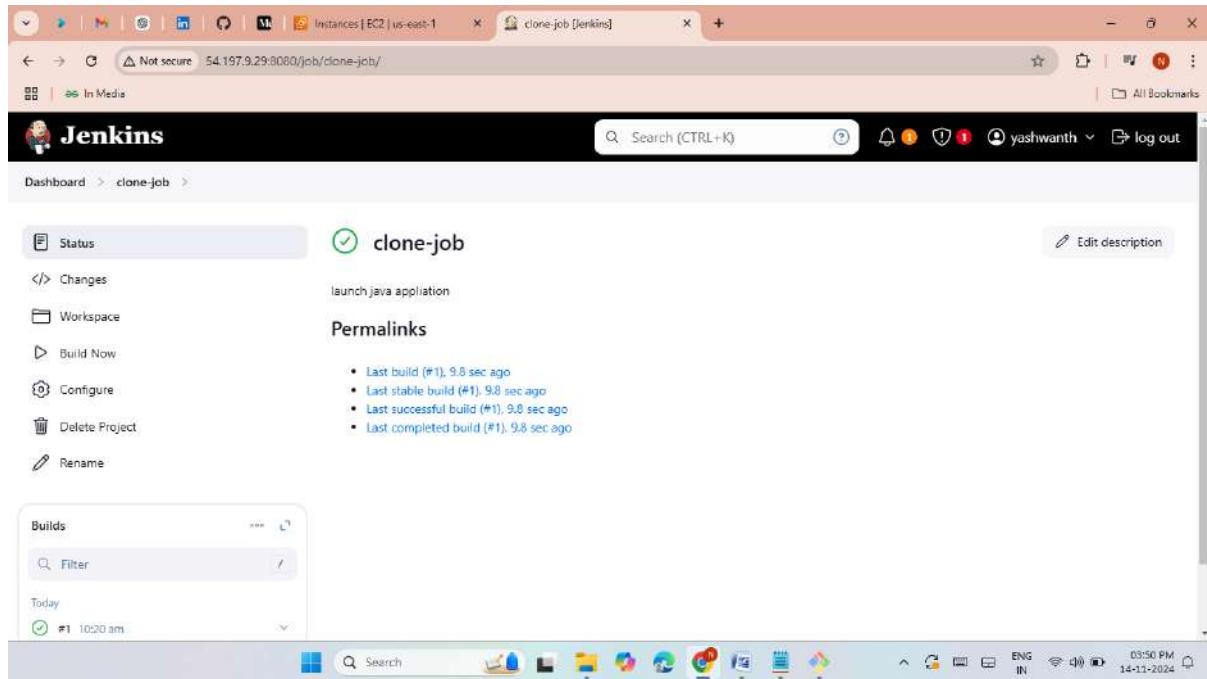
- Now create a cone-job and give slave4 label.



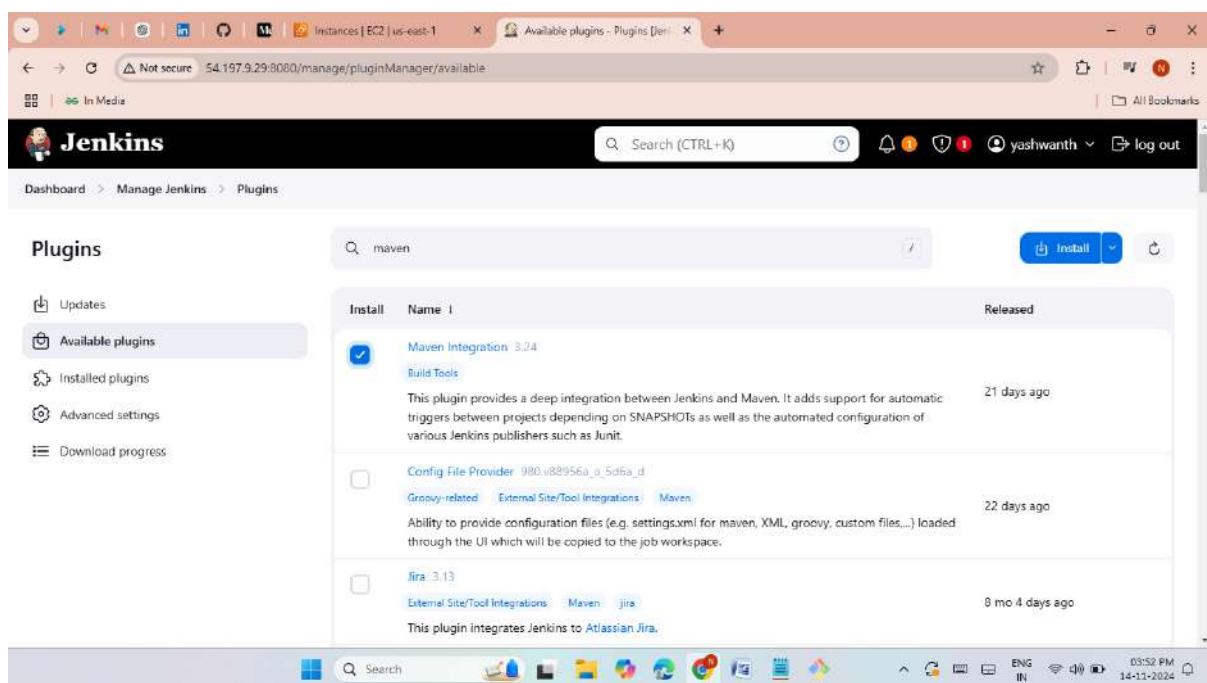
- Now clone the java application repo form github.
- Now Click on save.



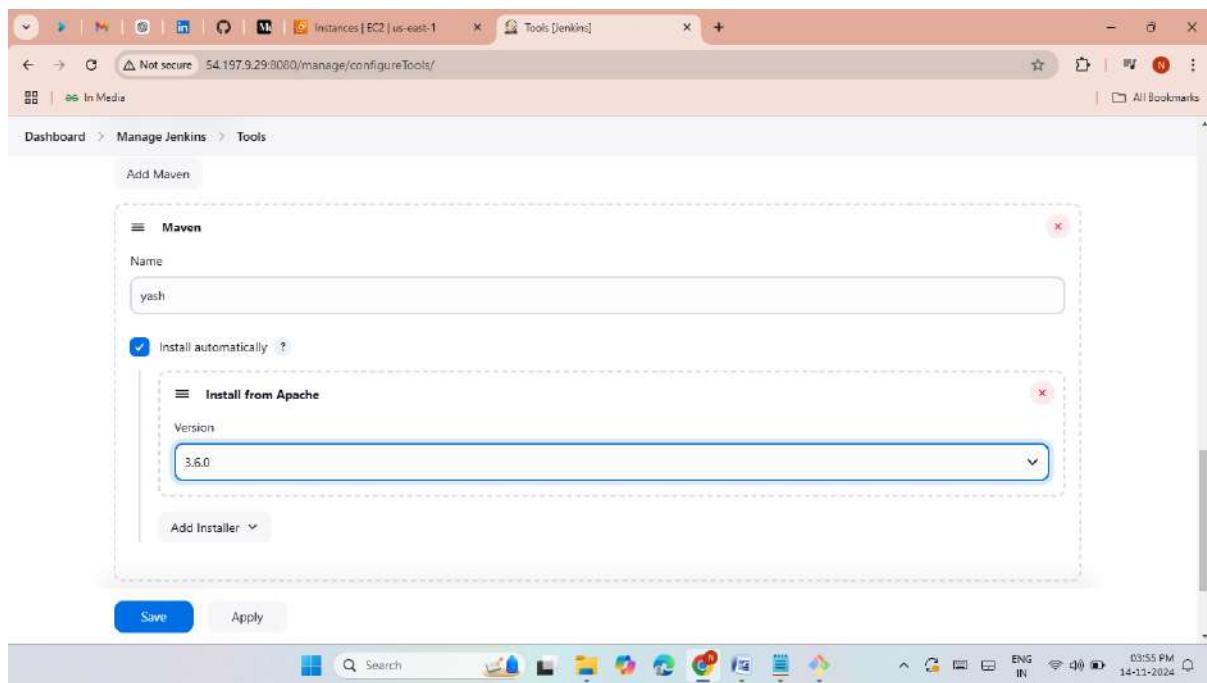
- Now click on build now.
- Here the job was success.



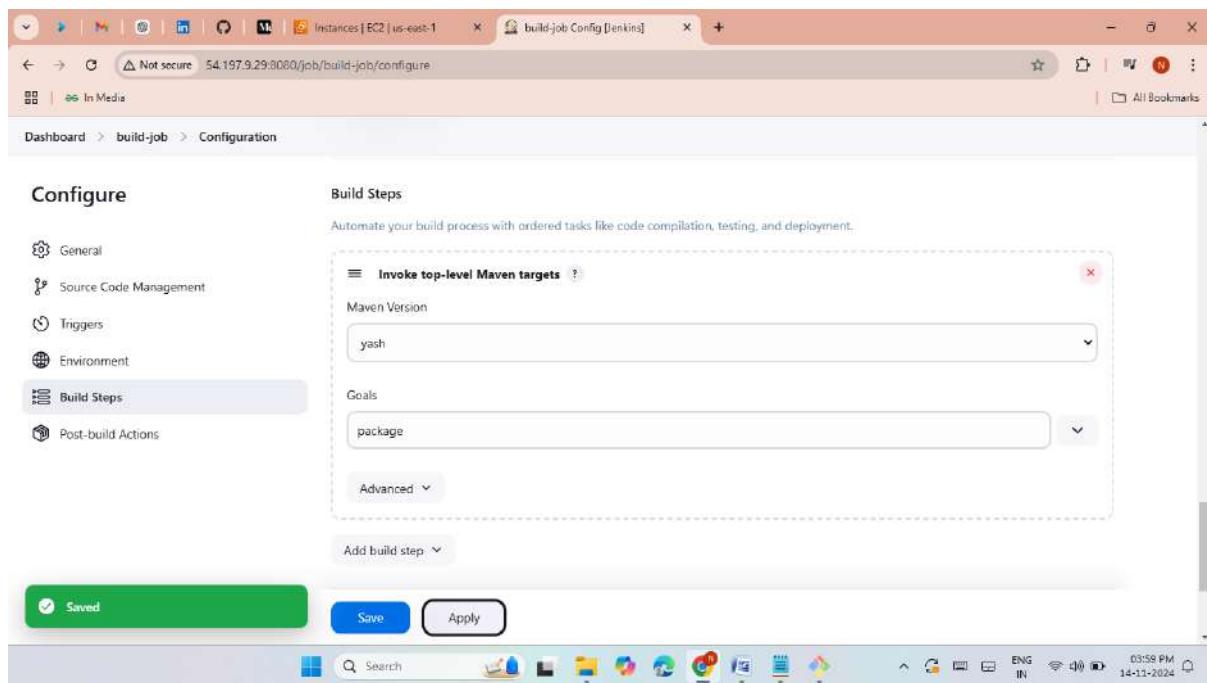
- Now install maven integration plugin for package the code.



- Now go to tools then add maven along with the version.



- Now create a build-job and copy the clone job created information.
- In build step add the created maven version and set the goles.
- Now click on apply.



- Now click on build now.
- Here the job was success.

The screenshot shows the Jenkins interface for the 'build-job' project. The status bar at the top indicates 'Not secure'. The main content area displays the project details: 'Status' (green with a checkmark), 'Changes' (empty), 'Workspace' (empty), 'Build Now' (button), 'Configure' (link), 'Delete Project' (link), and 'Rename' (link). Below this is a 'Permalinks' section with a list of recent builds. A large 'Builds' section shows a single entry from 'Today' at 10:29 am, marked with a green checkmark. The bottom of the screen shows a Windows taskbar with various icons and the system tray.

- Launch an another EC2 instance for install sonarqubes.
- In this step we can install sonarqubes in the same server.

The screenshot shows the AWS EC2 Instances page. The sidebar on the left lists various services, with 'Instances' selected. The main content area shows a table of instances. The 'sonarqubes' instance (ID: i-033523a4573793bc6) is selected and expanded. The 'Details' tab is active, showing the instance summary. Key details include:

Attribute	Value
Instance ID	i-033523a4573793bc6
Public IPv4 address	34.203.190.139
Private IPv4 addresses	172.31.84.133
Instance state	Running

➤ Now connect the created server and install sonarqube zip file from Google.

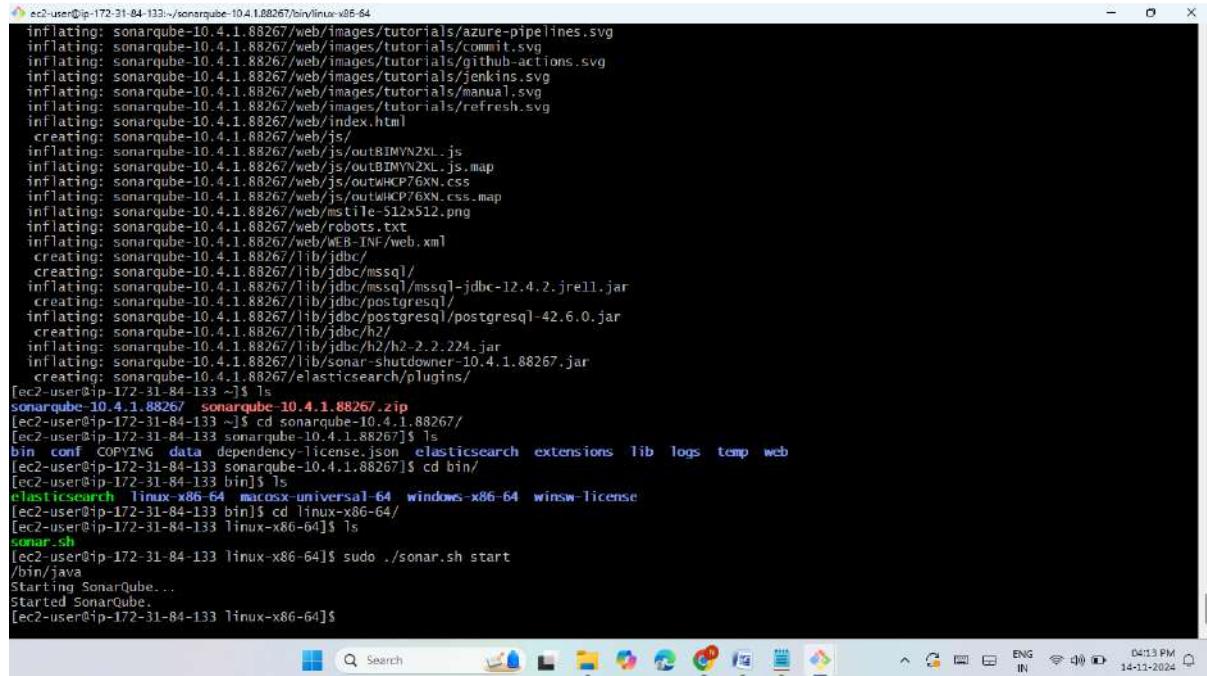
➤ Unzip the file with the command

Sudo unzip <file>

➤ Now go to sonarqubes→bin→linux-x86-64

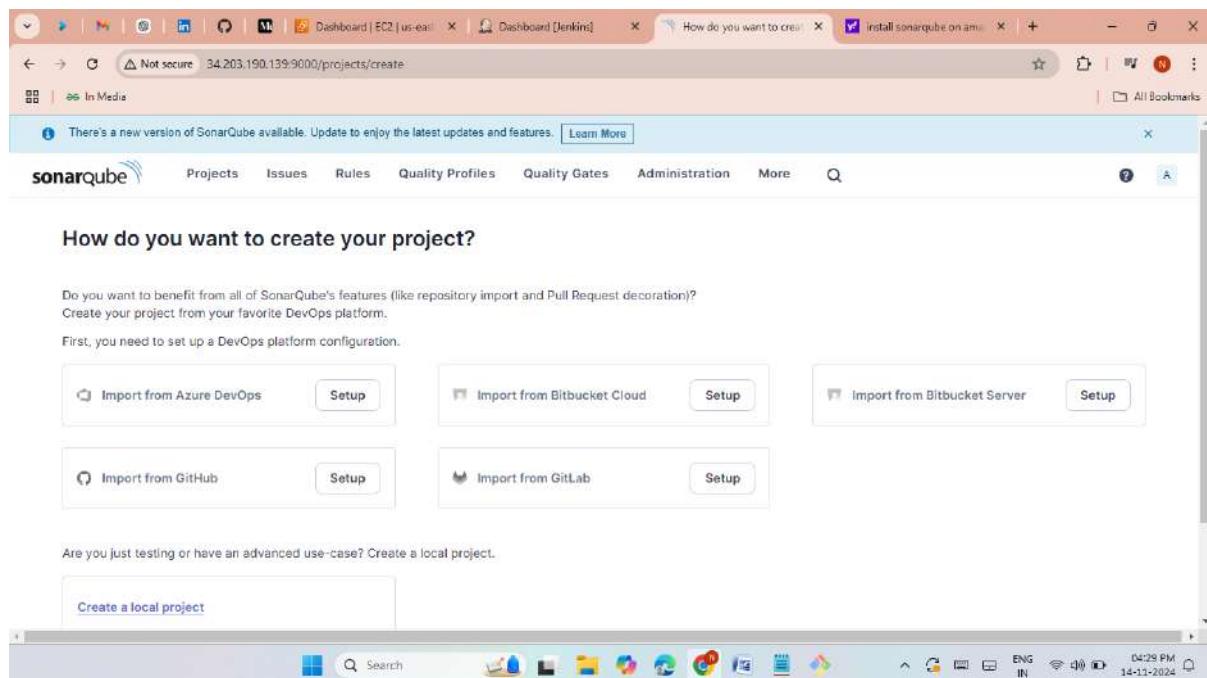
➤ Run the sonarqube with the command

Sudo ./sonar.sh start

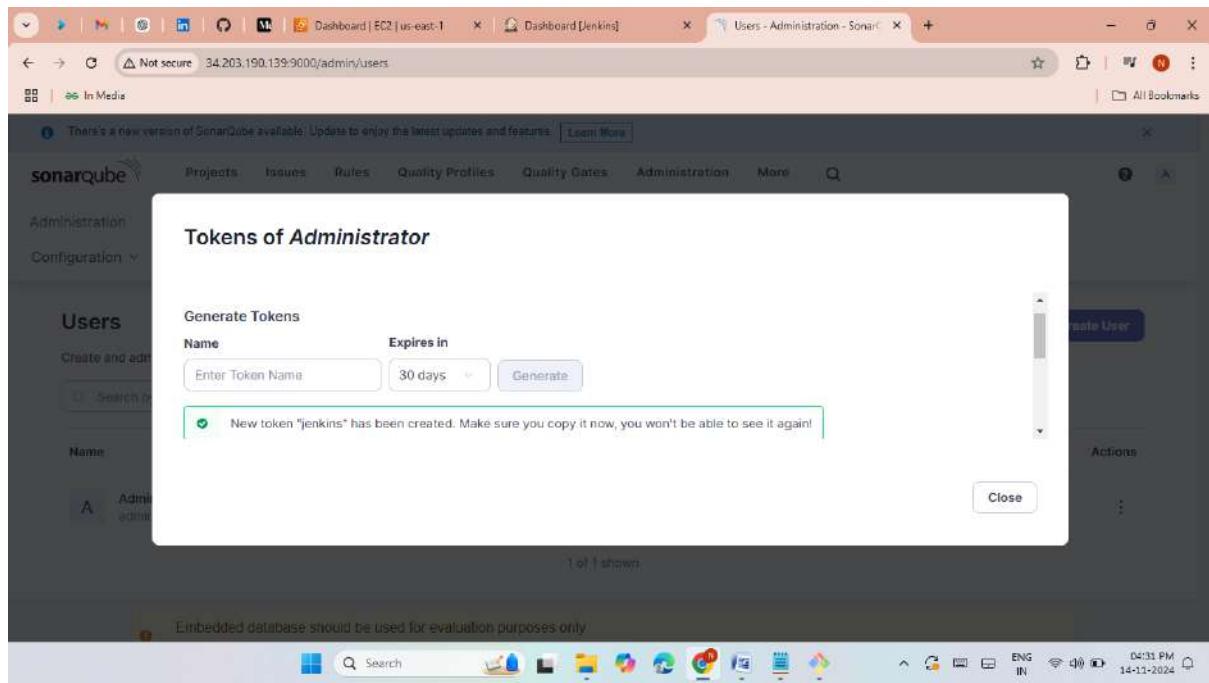


```
ec2-user@ip-172-31-84-133:~/sonarqube-10.4.1.88267/bin/linux-x86-64
inflating: sonarqube-10.4.1.88267/web/images/tutorials/azure-pipelines.svg
inflating: sonarqube-10.4.1.88267/web/images/tutorials/commit.svg
inflating: sonarqube-10.4.1.88267/web/images/tutorials/github-actions.svg
inflating: sonarqube-10.4.1.88267/web/images/tutorials/jenkins.svg
inflating: sonarqube-10.4.1.88267/web/images/tutorials/manual.svg
inflating: sonarqube-10.4.1.88267/web/images/tutorials/refresh.svg
inflating: sonarqube-10.4.1.88267/web/index.html
creating: sonarqube-10.4.1.88267/web/js/
inflating: sonarqube-10.4.1.88267/web/js/outLIMYN2XL.js.map
inflating: sonarqube-10.4.1.88267/web/js/outLIMYN2XL.js
inflating: sonarqube-10.4.1.88267/web/js/outuHCP76KN.css.map
inflating: sonarqube-10.4.1.88267/web/js/outuHCP76KN.css
inflating: sonarqube-10.4.1.88267/web/mstile-512x512.png
inflating: sonarqube-10.4.1.88267/web/robots.txt
inflating: sonarqube-10.4.1.88267/web/META-INF/web.xml
creating: sonarqube-10.4.1.88267/lib/jdbc/
creating: sonarqube-10.4.1.88267/lib/jdbc/mysql/
inflating: sonarqube-10.4.1.88267/lib/jdbc/mysql/mssql-jdbc-12.4.2.jre11.jar
creating: sonarqube-10.4.1.88267/lib/jdbc/postgresql/
inflating: sonarqube-10.4.1.88267/lib/jdbc/postgresql/postgresql-42.6.0.jar
creating: sonarqube-10.4.1.88267/lib/jdbc/h2/
inflating: sonarqube-10.4.1.88267/lib/jdbc/h2/h2-2.2.224.jar
inflating: sonarqube-10.4.1.88267/lib/sonar-shutdowner-10.4.1.88267.jar
creating: sonarqube-10.4.1.88267/elasticsearch/plugins/
[ec2-user@ip-172-31-84-133 ~]$ ls
sonarqube-10.4.1.88267 sonarqube-10.4.1.88267.zip
[ec2-user@ip-172-31-84-133 ~]$ cd sonarqube-10.4.1.88267/
[ec2-user@ip-172-31-84-133 sonarqube-10.4.1.88267]$ ls
bin config COPING data dependency-license.json elasticsearch extensions lib logs temp web
[ec2-user@ip-172-31-84-133 sonarqube-10.4.1.88267]$ cd bin/
[ec2-user@ip-172-31-84-133 bin]$ ls
elasticsearch_linux-x86_64 macOS-universal-64 windows-x86_64 winsw-license
[ec2-user@ip-172-31-84-133 bin]$ cd linux-x86-64/
[ec2-user@ip-172-31-84-133 Linux-x86-64]$ ls
sonar.sh
[ec2-user@ip-172-31-84-133 linux-x86-64]$ sudo ./sonar.sh start
/bin/java
Starting SonarQube...
Started SonarQube.
[ec2-user@ip-172-31-84-133 linux-x86-64]$
```

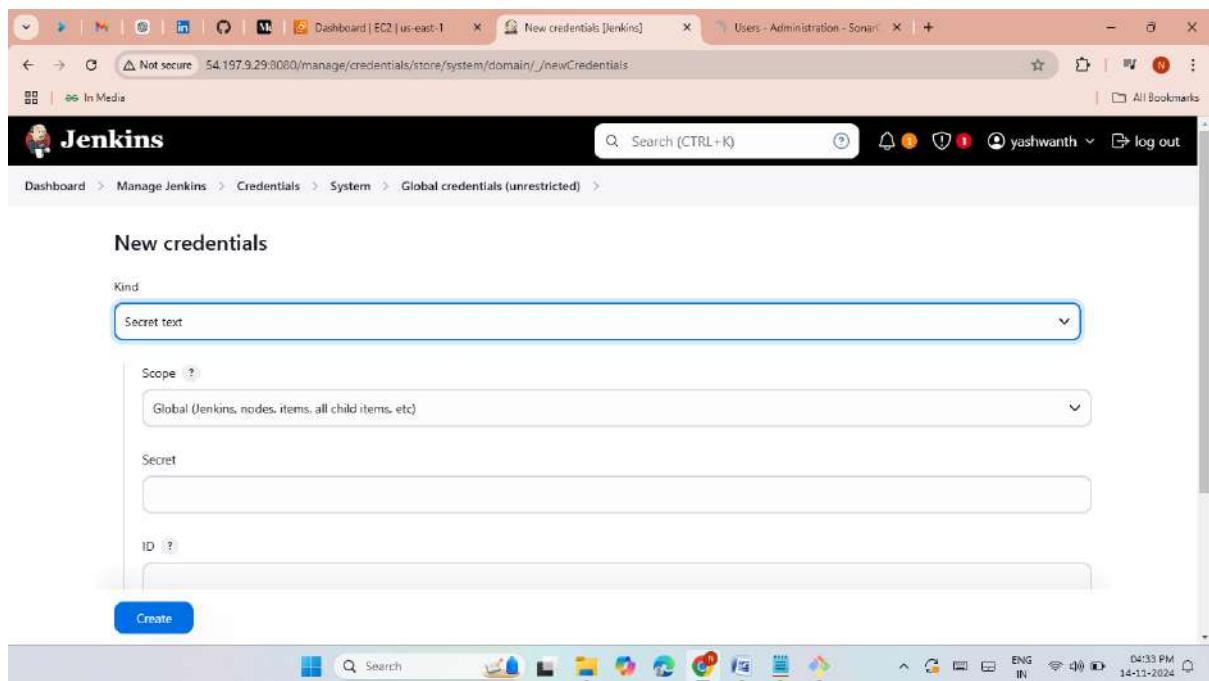
➤ Now launch the sonarqube with the ip address along with the port.



- Now generate token for access it on Jenkins.



- Now create a credentials in Jenkins.
- Select secret text
- Past the sonarqubes token into the secret.



- Here the sonarqubes credentials was created.

The screenshot shows the Jenkins Global credentials (unrestricted) page. It lists two entries:

ID	Name	Kind	Description
yash-credentials	ec2-user (jenkins)	SSH Username with private key	jenkins
sonarqubes	sonarqubes	Secret text	

At the bottom, there are icons for S, M, and L.

At the top right, there is a REST API link and a Jenkins 2.485 status bar.

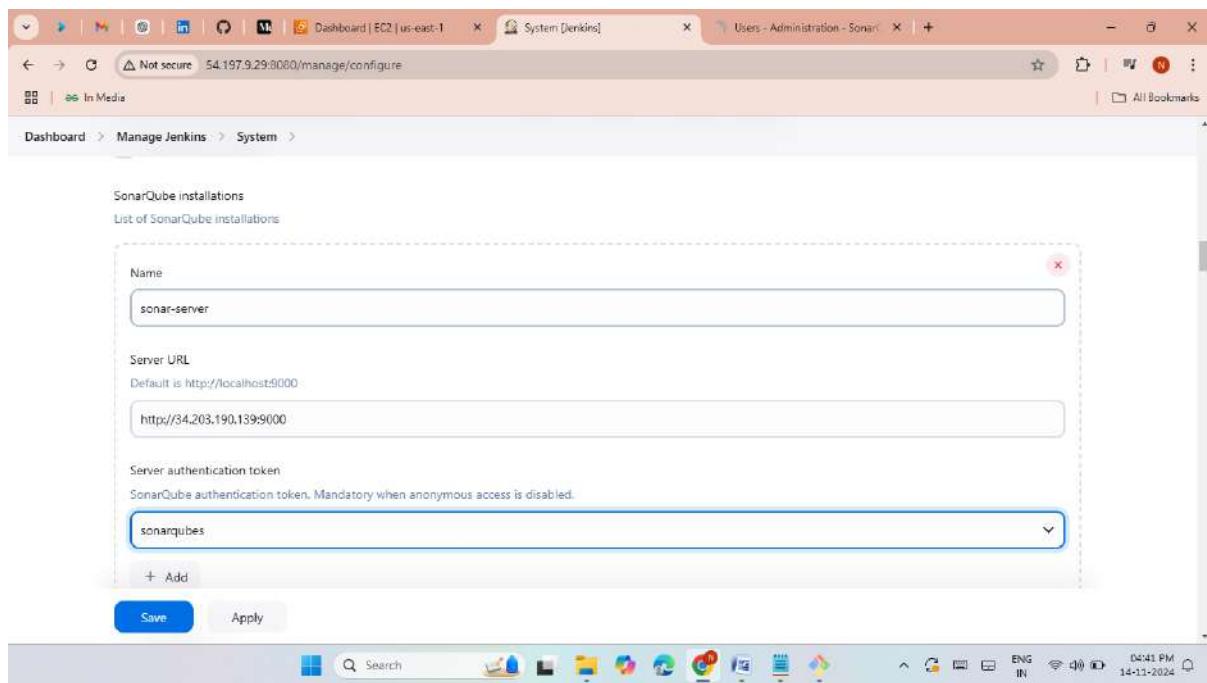
- Now install sonarqube scanner plugin.

The screenshot shows the Jenkins Available plugins page. The search bar contains "sonarqube". The "Available plugins" tab is selected. The SonarQube Scanner plugin is highlighted with a checked checkbox and is being installed.

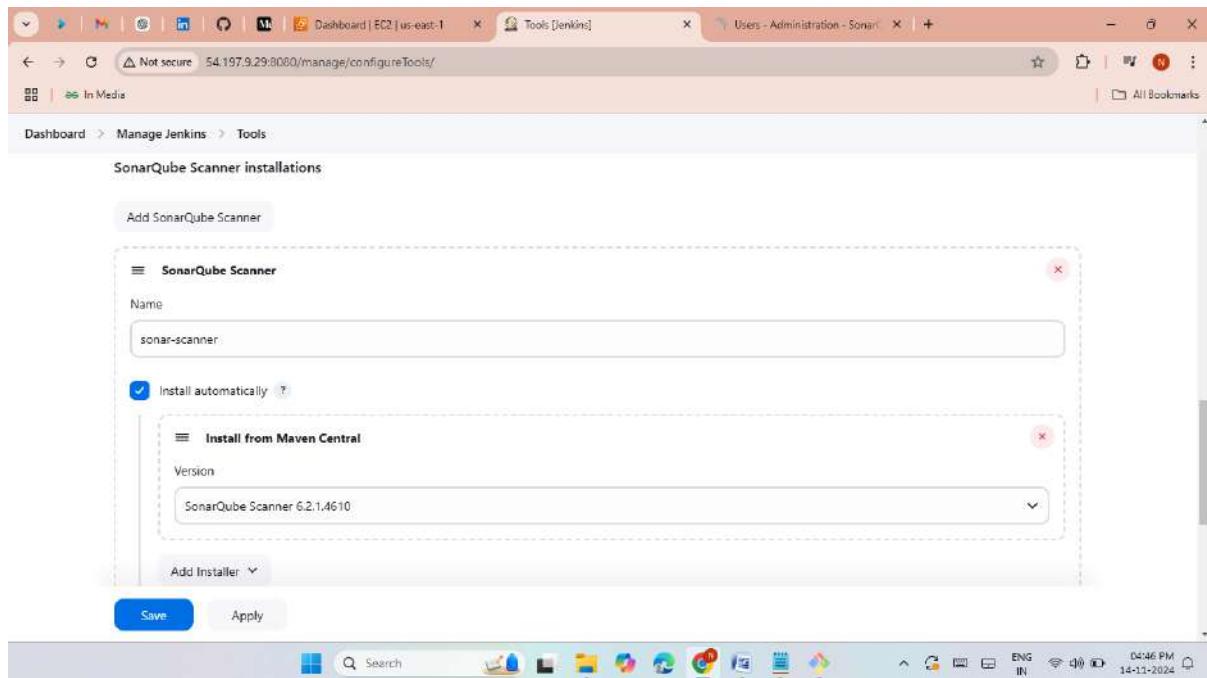
Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2	External Site/Tool Integrations Build Reports 8 mo 29 days ago
<input type="checkbox"/>	Sonar Gerrit 308.v9b.f1cb_e42305	External Site/Tool Integrations This plugin allows to submit issues from SonarQube to Gerrit as comments directly. 5 mo 12 days ago
<input type="checkbox"/>	SonarQube Generic Coverage 1.0	TODO 5 yr 3 mo ago

At the bottom, there is a REST API link and a Jenkins 2.485 status bar.

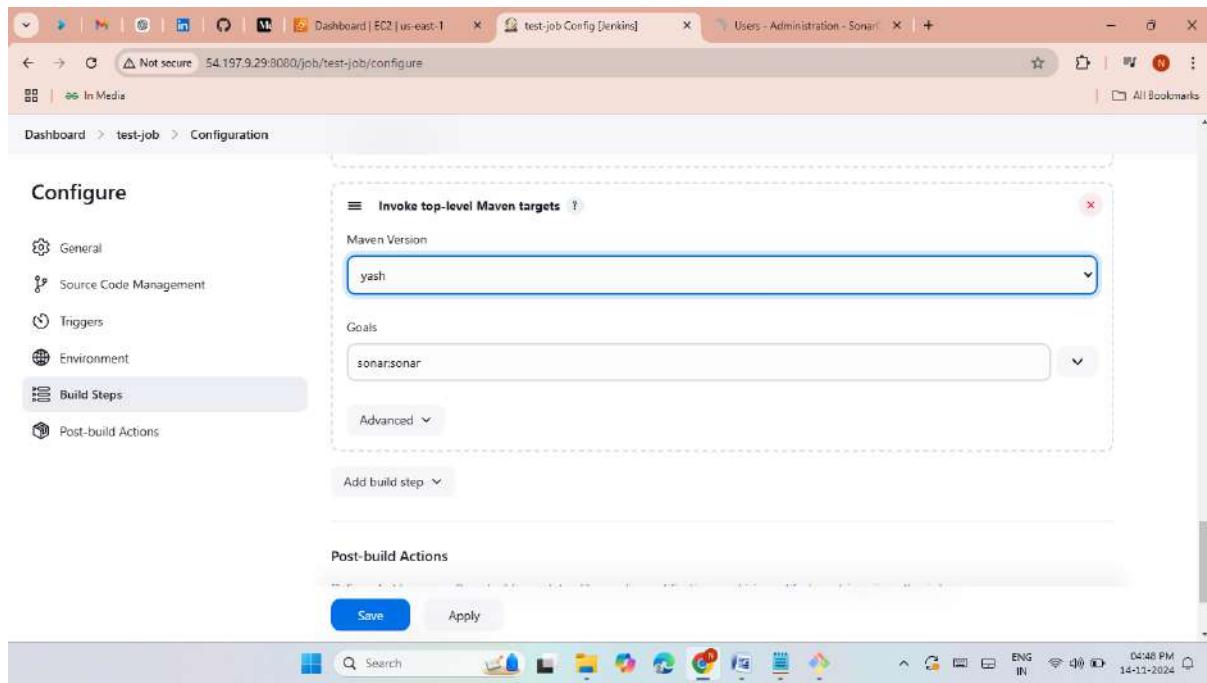
➤ Now go to system then add sonarqube tokens and server url.



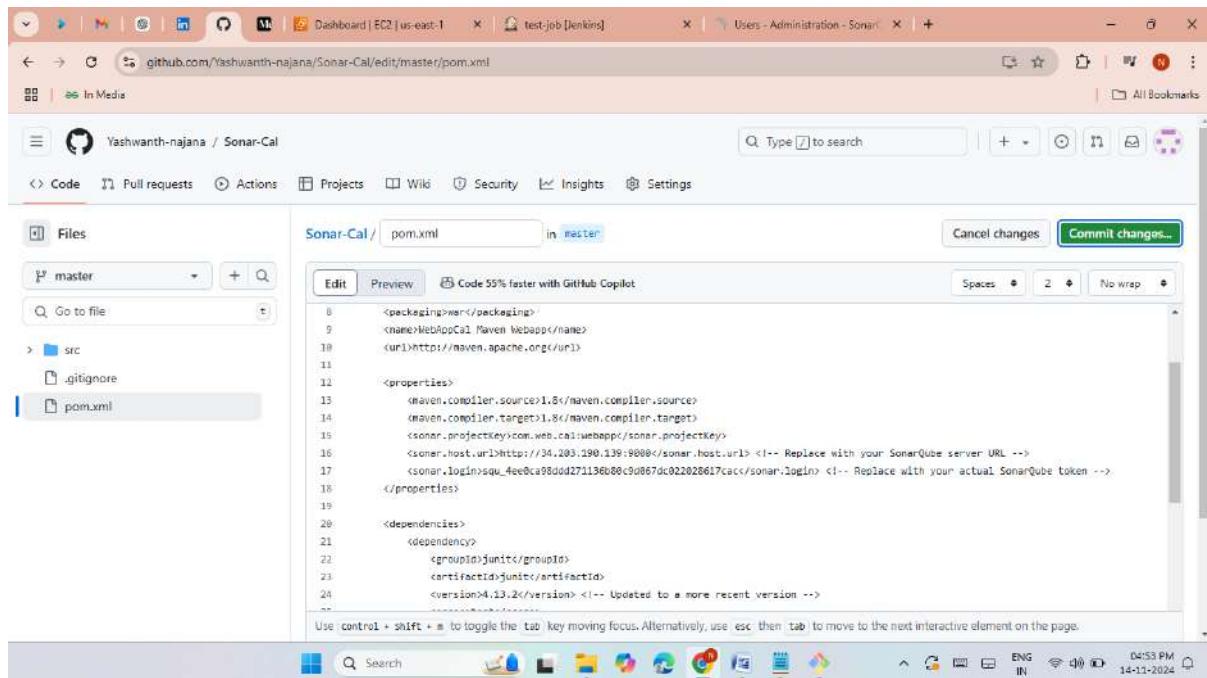
➤ Now go to tools then add sonarqube scanner.



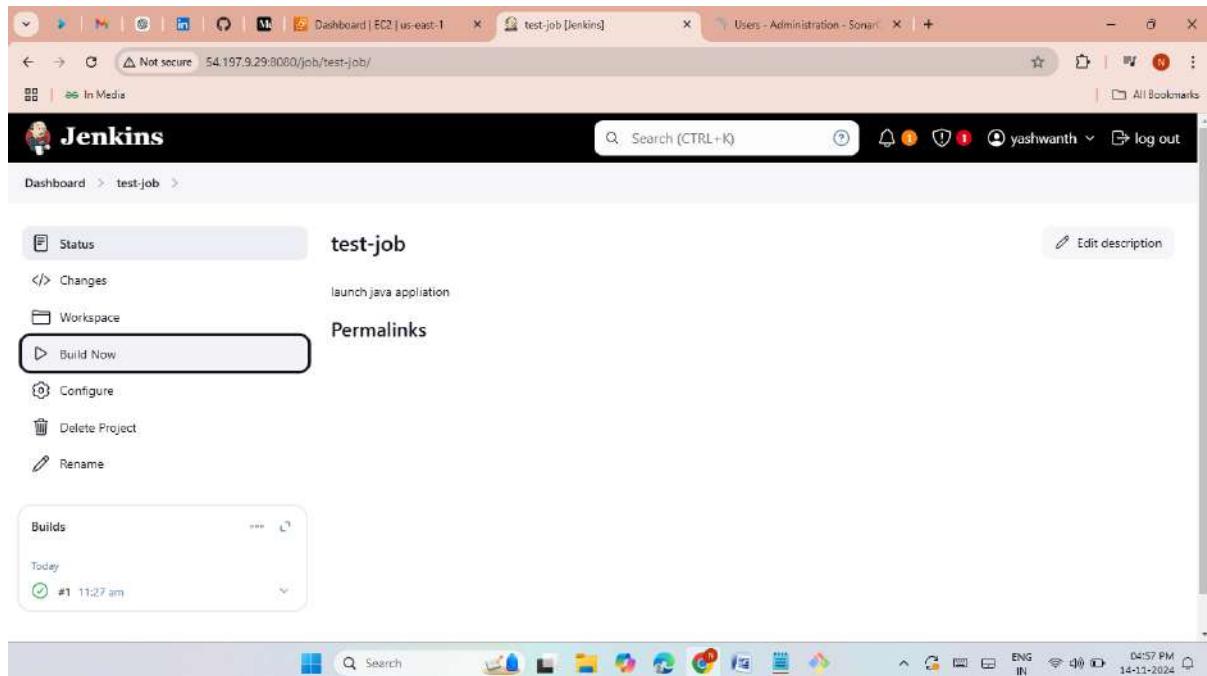
- Now create test-job then add maven version and gole for sonarqube in build step.



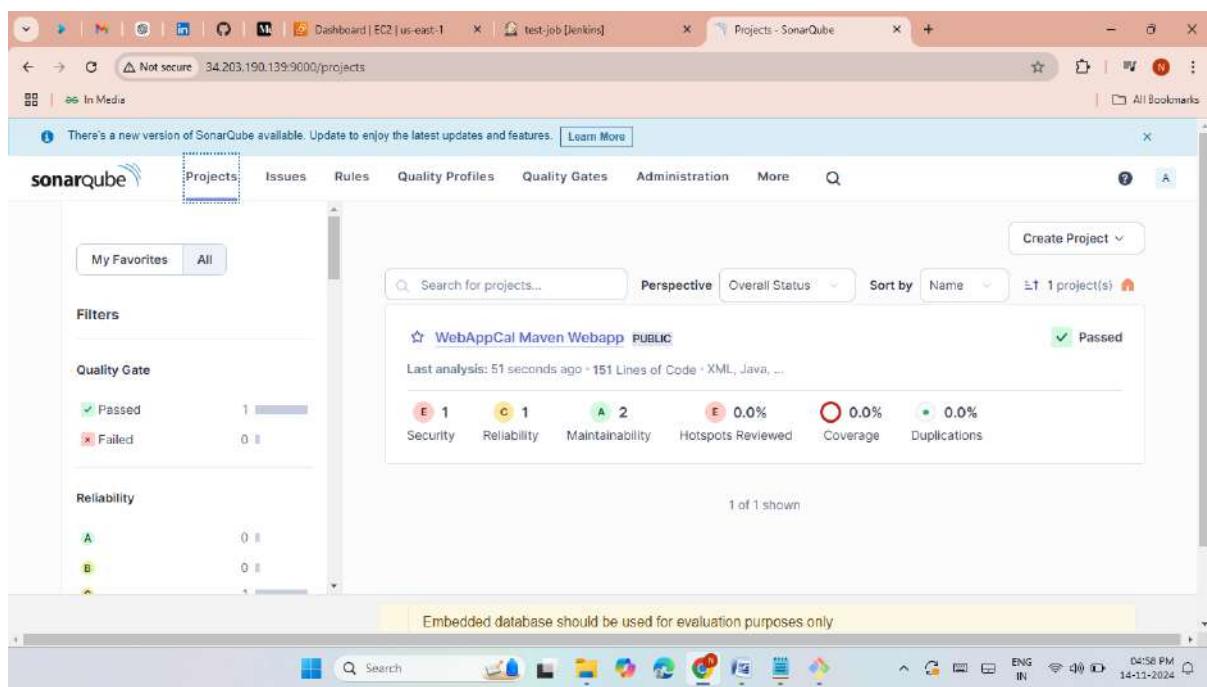
- Go to pom.xml file in github then give sonarqube ip address and sonarqube secret key.



- Now click on build now.
- The test job was success.



- Here we can see the result in sonarqubes.



- Now install deploy to container plugin.

- Install tomcat in slave server then start the tomcat.
- Give roles and passwords in tomcat-user.xml
- Remove permissions in context.xml file.

```
ec2-user@slave4:~/apache-tomcat-9.0.97/conf
you must define such a user - the username and password are arbitrary.

Built-in Tomcat manager roles:
- manager-gui      - allows access to the HTML GUI and the status pages
- manager-script   - allows access to the HTTP API and the status pages
- manager-jmx      - allows access to the JMX proxy and the status pages
- manager-status   - allows access to the status pages only

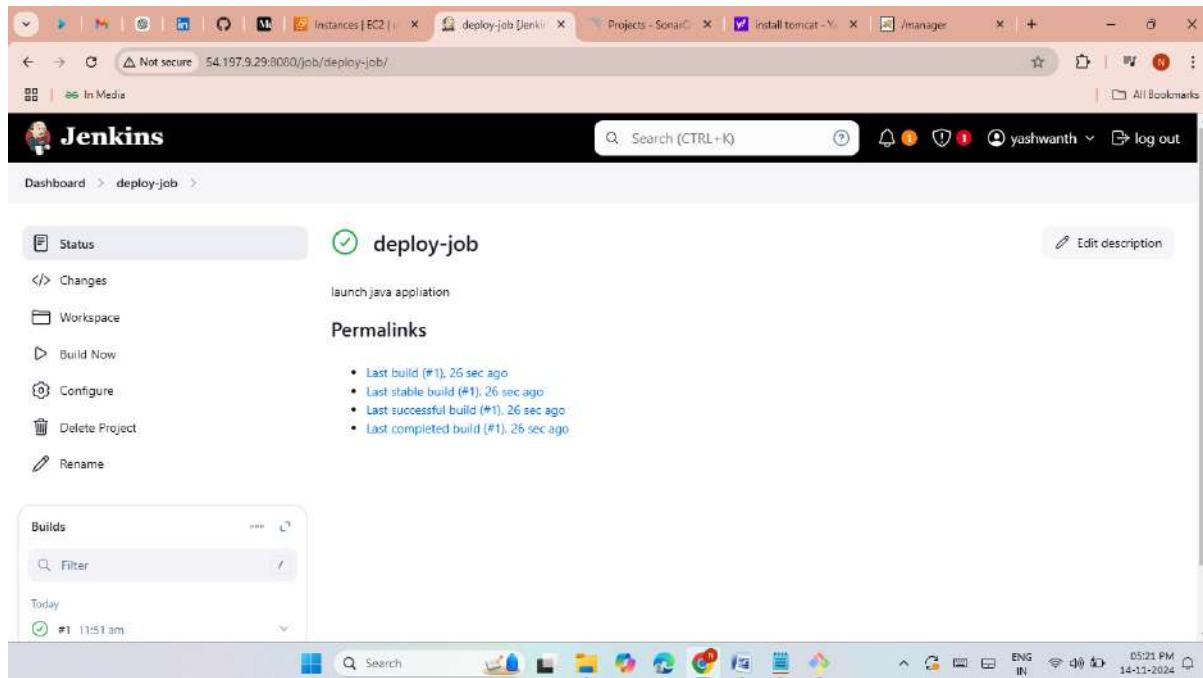
The users below are wrapped in a comment and are therefore ignored. If you
wish to configure one or more of these users for use with the manager web
application, do not forget to remove the <!-- ... --> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="" roles="manager-gui"/>
<user username="robot" password="" roles="manager-script"/>
-->
<!--
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- ... --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="yash" password="yash" roles="manager-gui"/>
<user username="yash1" password="yash1" roles="manager-script"/>
</tomcat-users>
-->
:wq!
```

- Now create a deploy-job then add post-build action.
- Copy .war files and select tomcat version and add credentials (manager-script).
- Now add tomcat URL then click on save.

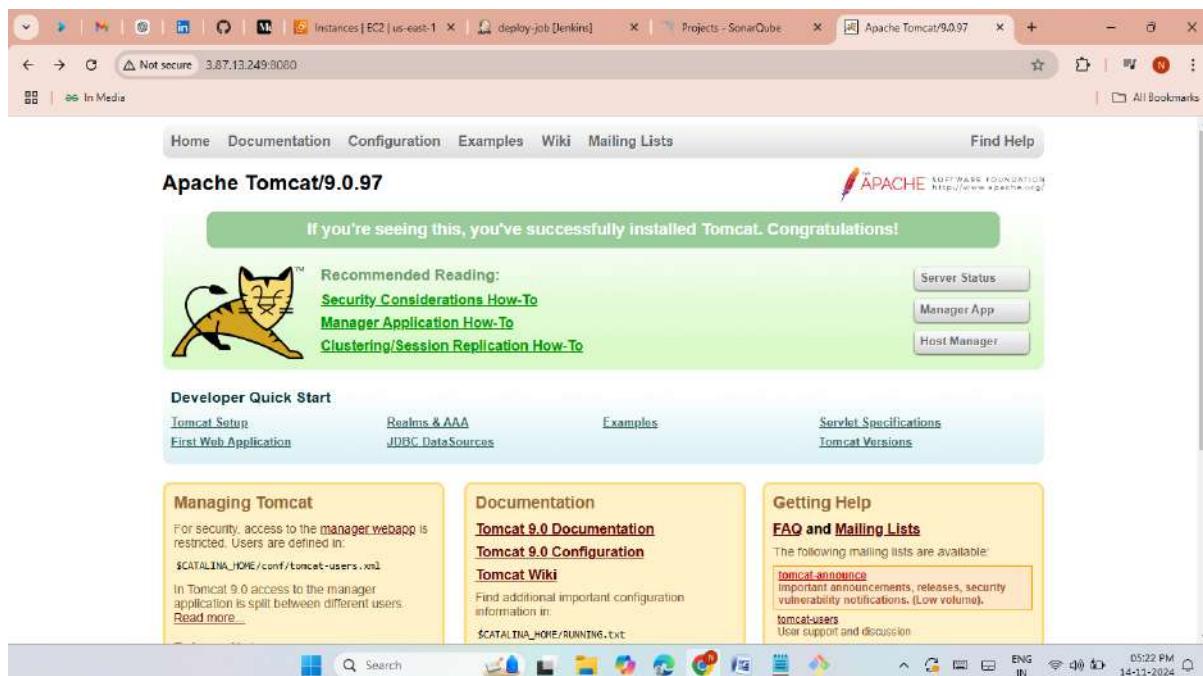
The screenshot shows the Jenkins job configuration interface. The left sidebar lists various configuration tabs: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The Post-build Actions tab is currently selected. Under the Post-build Actions section, there is a heading "Post-build Actions" with a sub-instruction: "Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs." Below this, there is a "Deploy war/ear to a container" section. It contains fields for "WAR/EAR files" (with a placeholder "**/*.war") and "Context path" (empty). A "Containers" section is expanded, showing a "Tomcat 9.x Remote" entry. Under "Tomcat 9.x Remote", there is a "Credentials" field containing "yash1*****". At the bottom of the "Deploy war/ear to a container" section are "Save" and "Apply" buttons. The status bar at the bottom right indicates the date and time as 14-11-2024 05:20 PM.

This screenshot shows the same Jenkins job configuration interface as the previous one, but with a different focus. The "Containers" section is now expanded, showing the "Tomcat 9.x Remote" entry. Under "Tomcat 9.x Remote", there is a "Tomcat URL" field containing "http://3.87.13.249:8080". Below the "Tomcat URL" field is an "Advanced" dropdown menu. At the bottom of the "Containers" section are "Save" and "Apply" buttons. The status bar at the bottom right indicates the date and time as 14-11-2024 05:23 PM.

- Now click on build now.
- Here the job was success.



- Now copy the slave4 public ip and past it on google browser.
- The tomcat was hosted successfully.
- Click on manager app.



➤ Now click on web-apps.1.8.

The screenshot shows the Tomcat Manager interface. In the 'Applications' section, there is a table with the following data:

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	2	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/webapp-1.8	None specified	Servlet	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

➤ Here the java web calculator was hosted successfully.

The screenshot shows the Java web calculator application. The form contains the following fields:

- first number:
- Second number :
- addition
- subtraction
- product
-



Addition

8

Subtraction

4

Multiplication

12

Calculator

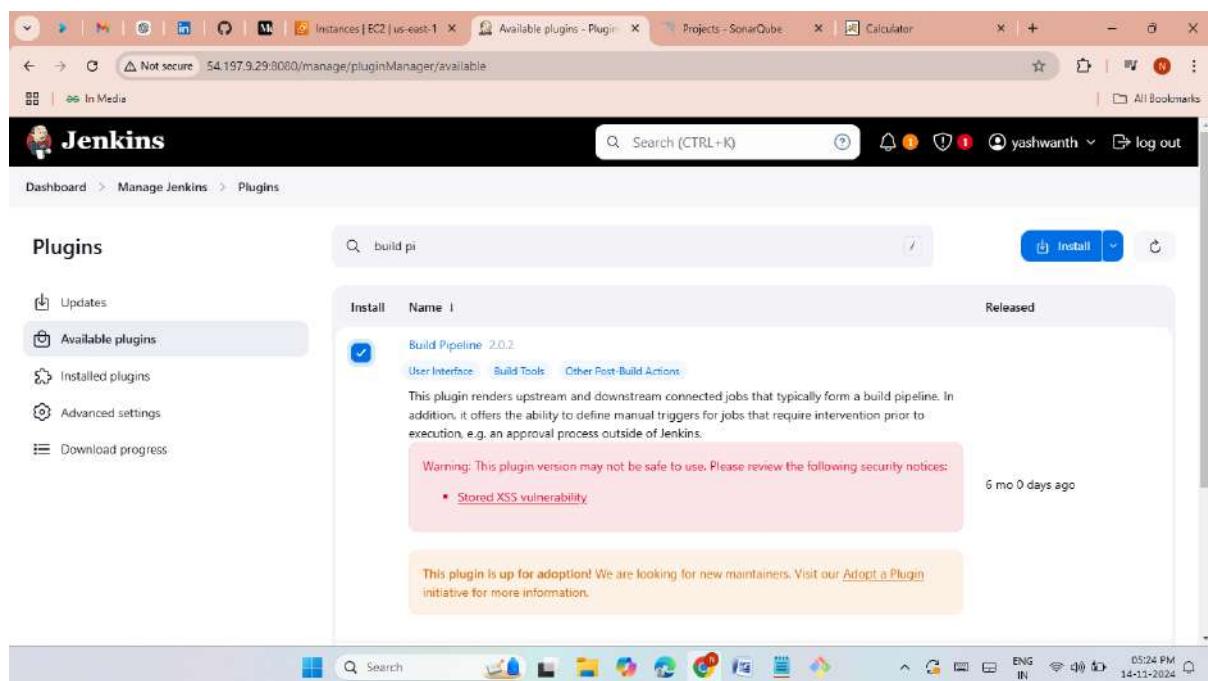
first number:

Second number:

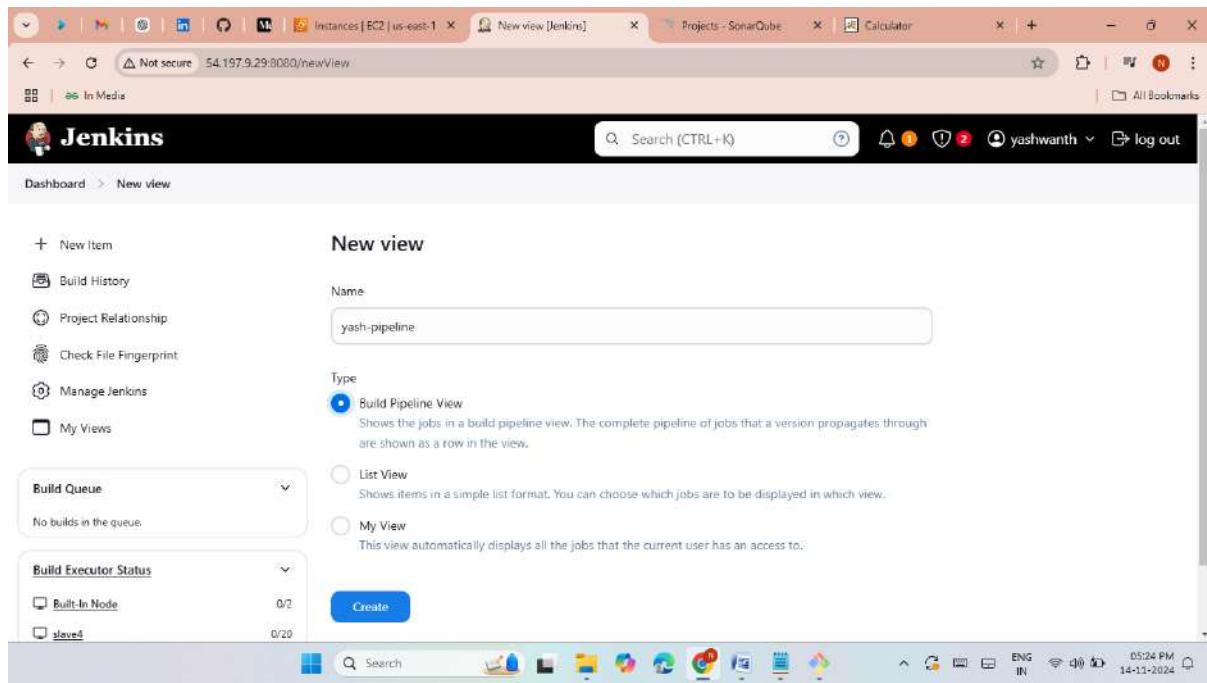
addition
 subtraction
 product



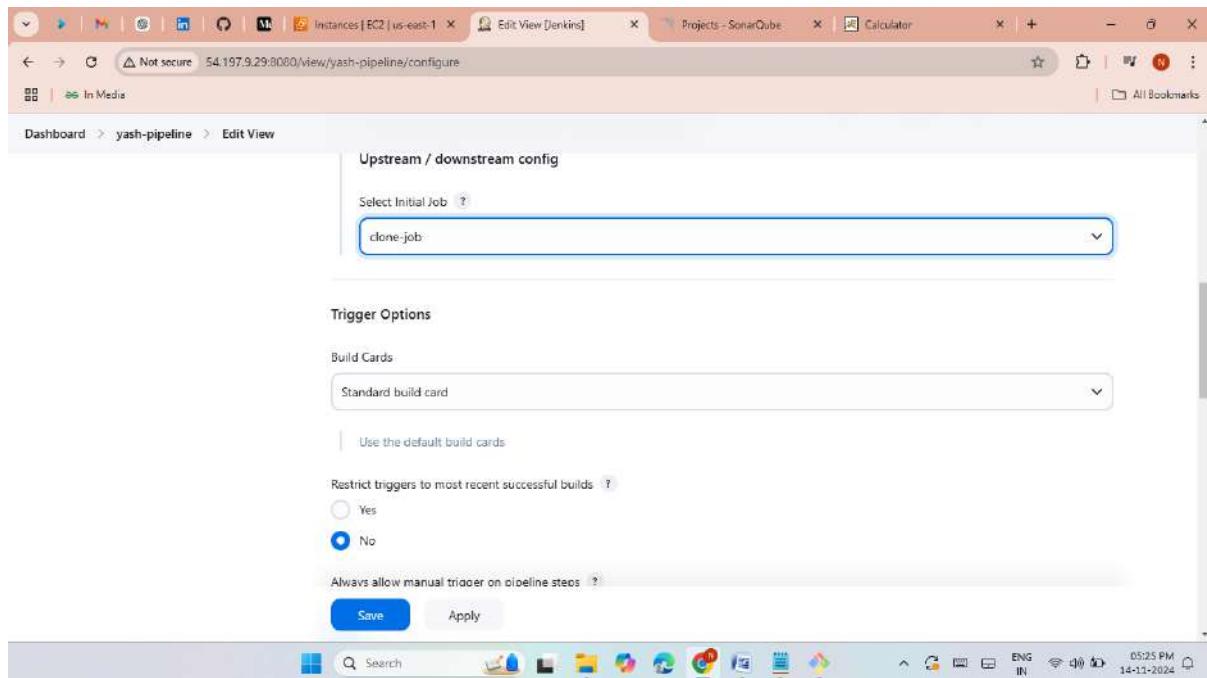
➤ Now install build pipeline plugin to create pipeline.



- Create a pipeline for all the jobs.



- Select initial job.



➤ Go to clone job add downstream project.

The screenshot shows the Jenkins interface for the 'clone-job' configuration. The top navigation bar includes tabs for 'Instances | EC2 | us-east-1', 'clone-job [Jenkins]', 'Projects - SonarQube', 'Calculator', and a search bar. The main content area displays the 'clone-job' configuration with the following details:

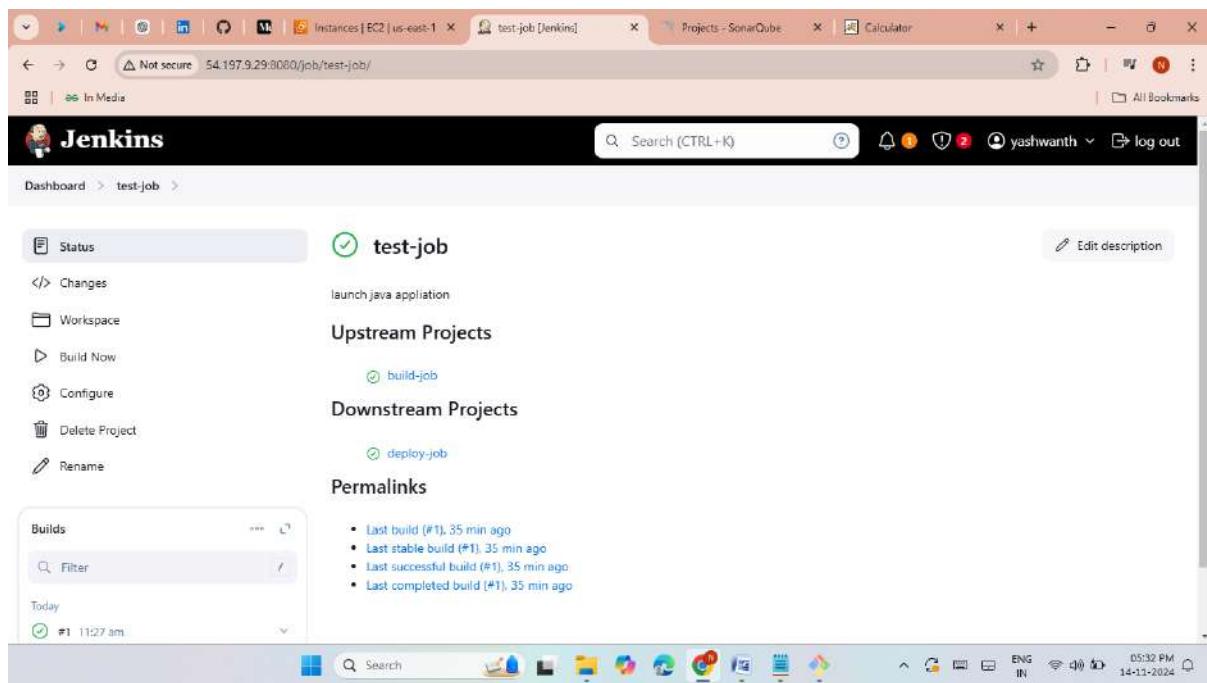
- Status:** Green checkmark icon, indicating the job is healthy.
- Description:** 'launch java application'
- Downstream Projects:** A list containing 'build-job'.
- Permalinks:** A list of recent builds:
 - Last build (#2), 35 min ago
 - Last stable build (#2), 35 min ago
 - Last successful build (#2), 35 min ago
 - Last completed build (#2), 35 min ago
- Builds:** A table showing build history for today, with the most recent entry being build #2 at 11:25 am.

➤ Go to build job add downstream project and upstream project.

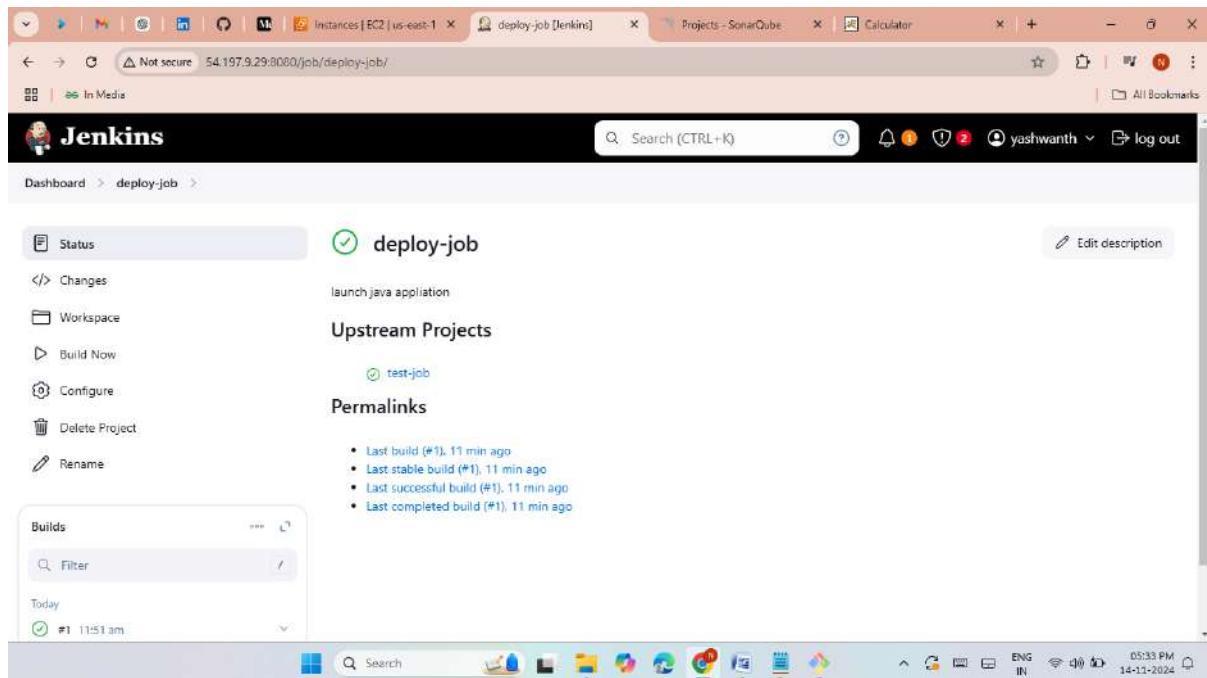
The screenshot shows the Jenkins interface for the 'build-job' configuration. The top navigation bar includes tabs for 'Instances | EC2 | us-east-1', 'build-job [Jenkins]', 'Projects - SonarQube', 'Calculator', and a search bar. The main content area displays the 'build-job' configuration with the following details:

- Status:** Green checkmark icon, indicating the job is healthy.
- Description:** 'launch java application'
- Upstream Projects:** A list containing 'clone-job'.
- Downstream Projects:** A list containing 'test-job'.
- Permalinks:** A list of recent builds:
 - Last build (#2), 35 min ago
 - Last stable build (#2), 35 min ago
 - Last successful build (#2), 35 min ago
 - Last completed build (#2), 35 min ago
- Builds:** A table showing build history for today, with the most recent entry being build #2 at 11:26 am.

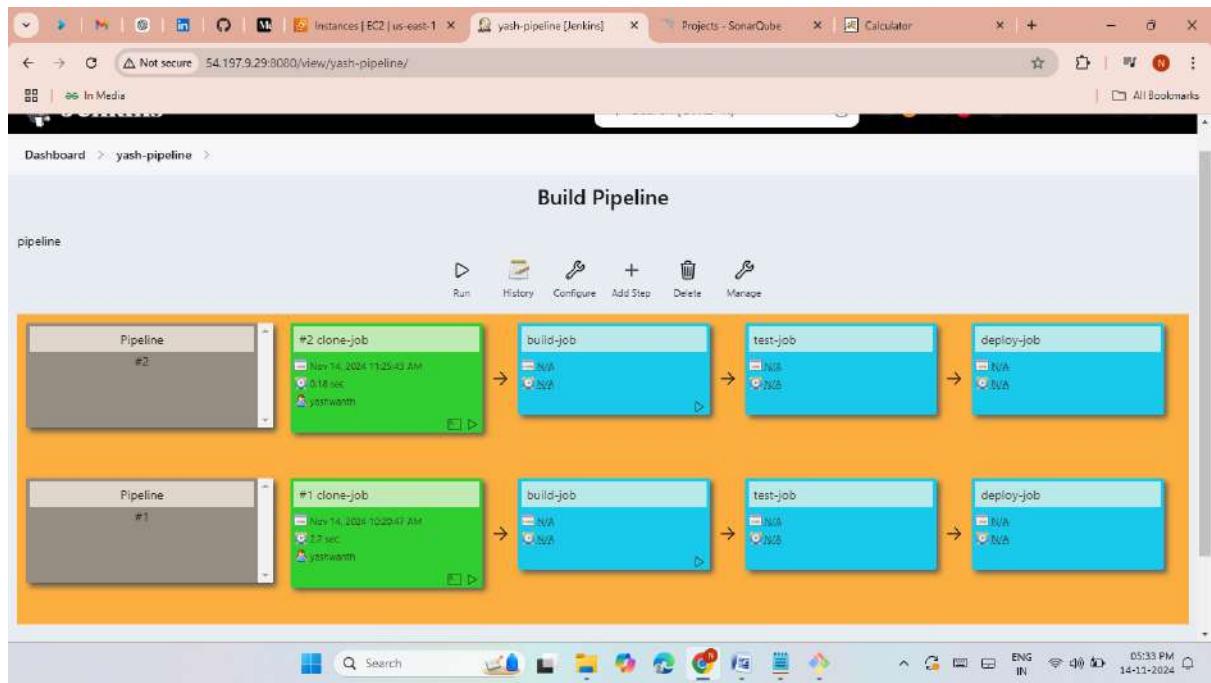
- Go to test job add downstream project and upstream project.



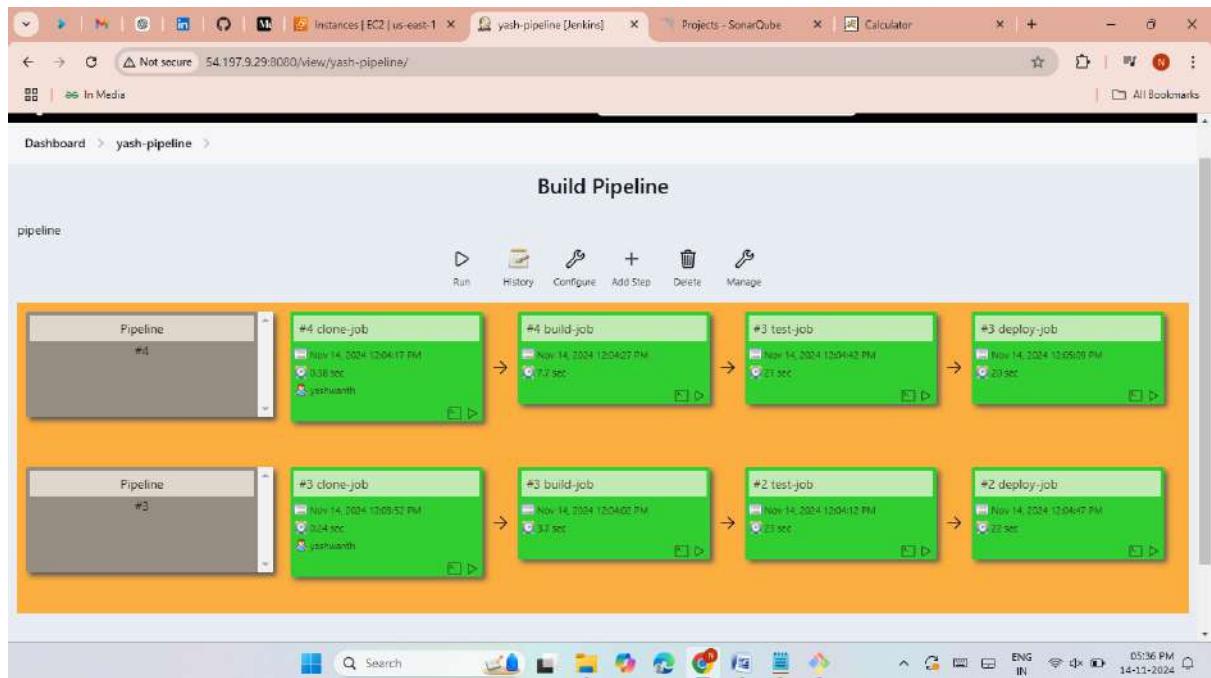
- Go to build job add upstream project.



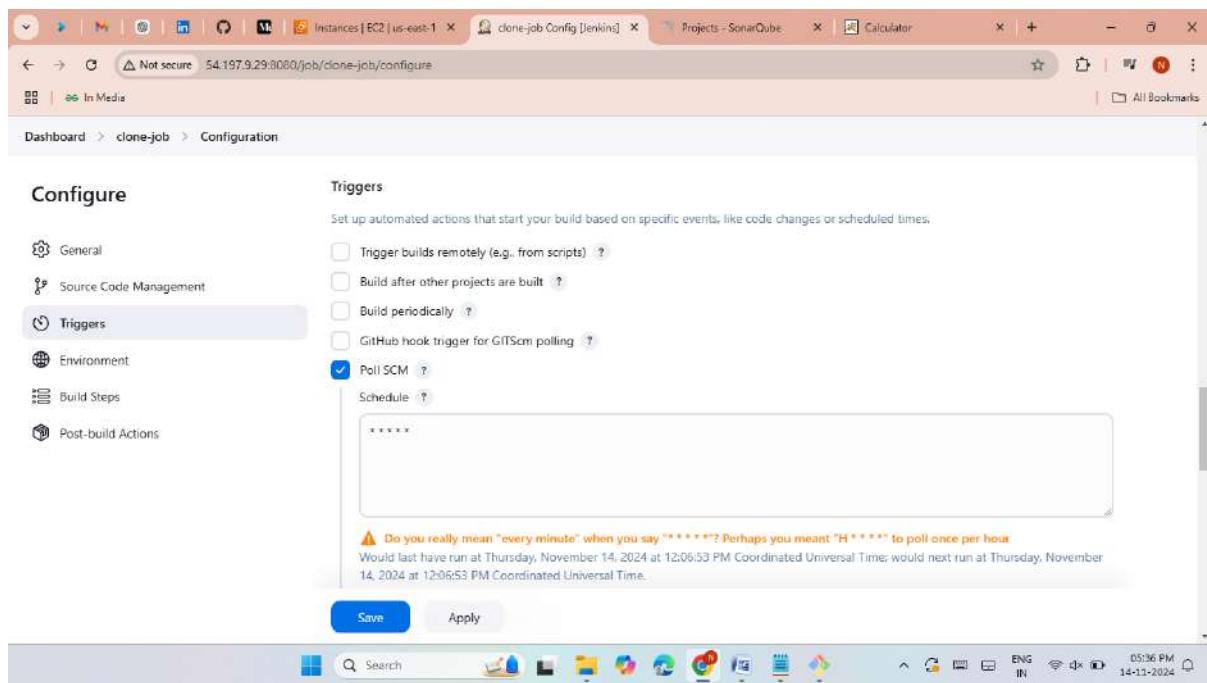
- Here the pipeline was successfully created.



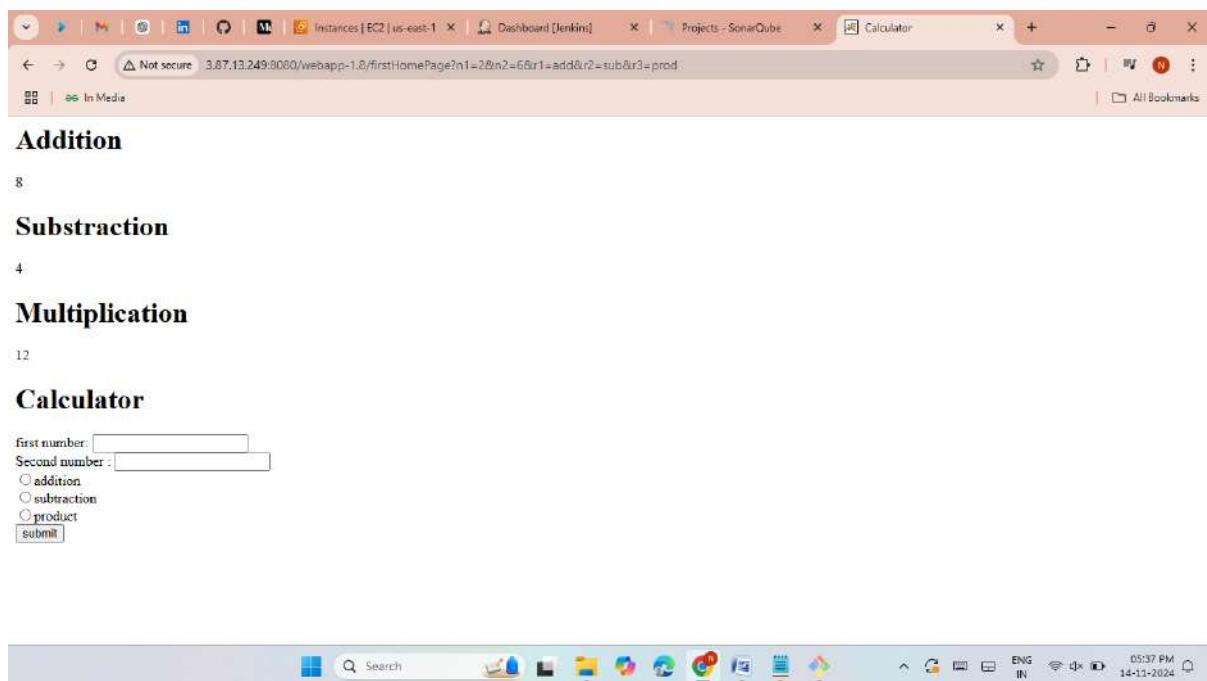
- Click on run the jobs was run onebyone.



➤ Now add poll SCM.



➤ Here the main output.



- I have changed the code in github.

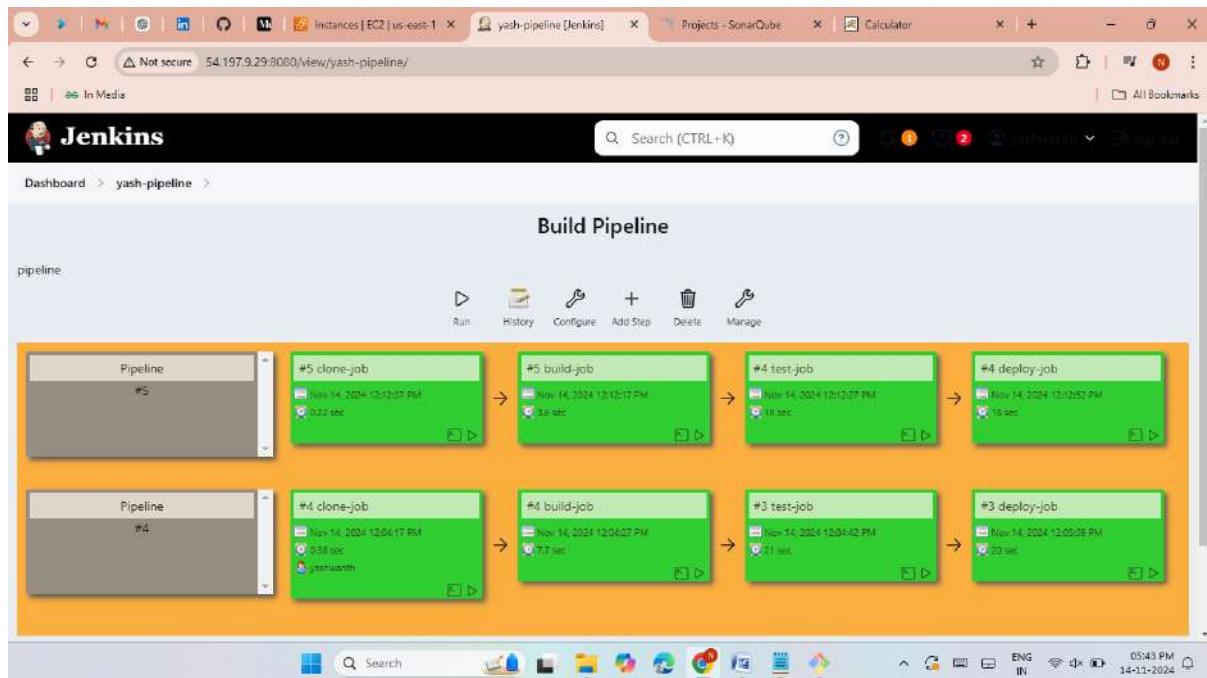
```

1 package mypackage;
2
3 import java.io.*;
4 import javax.servlet.*;
5 import javax.servlet.http.*;
6
7 public class Calculator extends HttpServlet {
8
9     public long mulFunc(long first, long second){
10
11         return first*second;
12     }
13
14     public long subFunc(long first, long second){
15
16         return second-first;
17     }
18
19     public long addFunc(long first, long second){
20
21         return first+second;
22     }

```

Use control + shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.

- After 1 min the job was run automatically because of poll scm.



- Here the java calculator was changed successfully.

Multiplication

12

Subtraction

4

Addition

8

Calculator

first number:

Second number:

addition

subtraction

product

- Now I have added webhooks.

Yashwanth-najana / Sonar-Cal

Type to search

Code Full requests Actions Projects Wiki Security Insights Settings

Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at <https://docs.github.com/webhooks/ping-event>.

General	Webhooks	Add webhook
Access	Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide .	
Collaborators		
Moderation options		
Code and automation		
Branches		
Tags		
Rules		
Actions		
Webhooks	<input checked="" type="checkbox"/> http://54.197.9.29:8080/github-webhook/push <small>Last delivery was successful.</small>	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
Environments		
Codespaces		

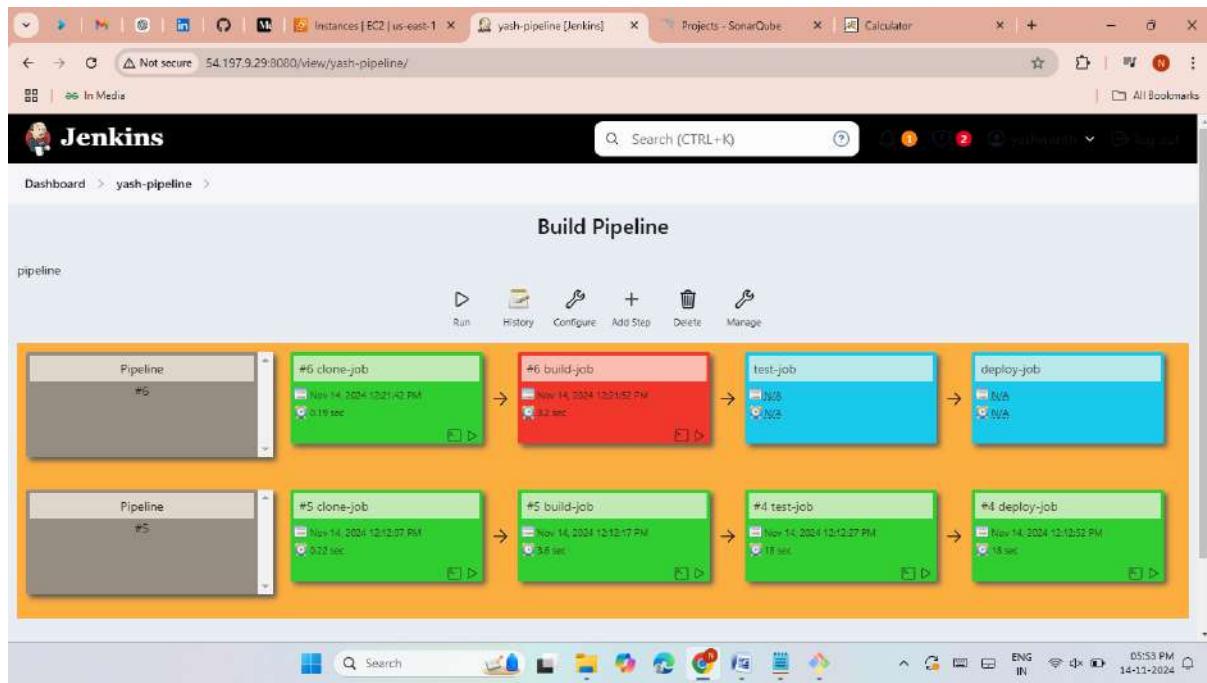
The screenshot shows the Jenkins job configuration page for a job named 'clone-job'. The 'Triggers' section is selected, displaying options like 'Trigger builds remotely', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling' (which is checked), and 'Poll SCM'. The 'Environment' section is also visible, containing settings for workspace deletion and timestamps. At the bottom are 'Save' and 'Apply' buttons.

➤ Now change the code in github.

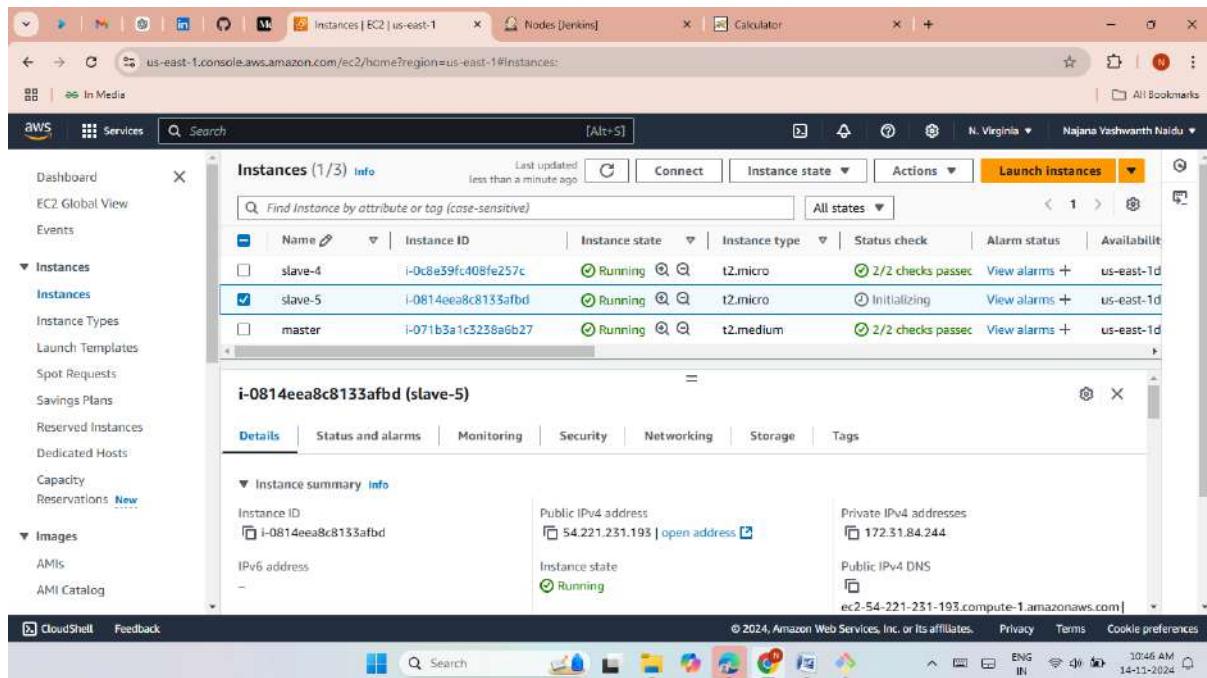
The screenshot shows the GitHub code editor for a Java file named 'Calculator.java' located in the 'src/main/java/mypackage' directory of a repository named 'Sonar-Cal'. The code implements a simple calculator with methods for multiplication, addition, and subtraction. The GitHub Copilot feature is shown as a suggestion above the code.

```
public class Calculator extends HttpServlet {
    public long mulFunc(long first, long second){
        return first*second;
    }
    public long addFunc(long first, long second){
        return second+first;
    }
    public long subFunc(long first, long second){
        return first-second;
    }
}
```

- Here the jobs were changed with in seconds because of webhooks.



- Now launch an ec2 instances and give name as slave5.



- Now connect Slave5 then install java.
- Now give private keys from master server.

```

ec2-user@ip-172-31-84-244:~$ sudo yum install -y java-1.8.0-openjdk-headless
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : git-core-2.40.1-1.amzn2.0.3.x86_64 1/6
  Installing : git-core-doc-2.40.1-1.amzn2.0.3.noarch 2/6
  Installing : perl-Error-0.17020-2.amzn2.noarch 3/6
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 4/6
  Installing : perl-Git-2.40.1-1.amzn2.0.3.noarch 5/6
  Installing : git-2.40.1-1.amzn2.0.3.x86_64 6/6
  Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64 1/6
  Verifying : git-2.40.1-1.amzn2.0.3.x86_64 2/6
  Verifying : perl-Error-0.17020-2.amzn2.noarch 3/6
  Verifying : git-core-2.40.1-1.amzn2.0.3.x86_64 4/6
  Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch 5/6
  Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch 6/6

Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3

Dependency Installed:
  git-core.x86_64 0:2.40.1-1.amzn2.0.3           git-core-doc.noarch 0:2.40.1-1.amzn2.0.3           perl-Error.noarch 1:0.17020-2.amzn2
  perl-git.noarch 0:2.40.1-1.amzn2.0.3           perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2

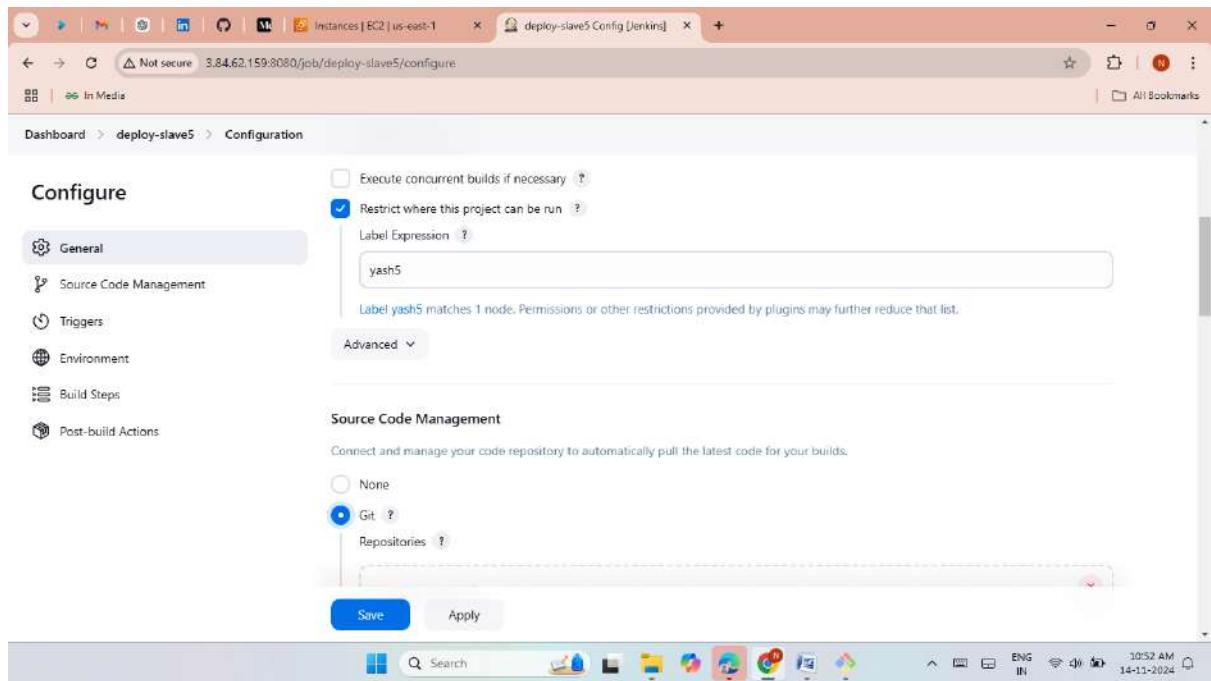
Complete!
[ec2-user@ip-172-31-84-244 ~]$ sudo hostname slave5
[ec2-user@ip-172-31-84-244 ~]$ exec bash
[ec2-user@slave5 ~]$ ls
[ec2-user@slave5 ~]$ cd .ssh/
[ec2-user@slave5 .ssh]$ ls
authorized_keys
[ec2-user@slave5 .ssh]$ sudo vi authorized_keys
[ec2-user@slave5 .ssh]$ cd
[ec2-user@slave5 ~]$ pwd
/home/ec2-user
[ec2-user@slave5 ~]$

```

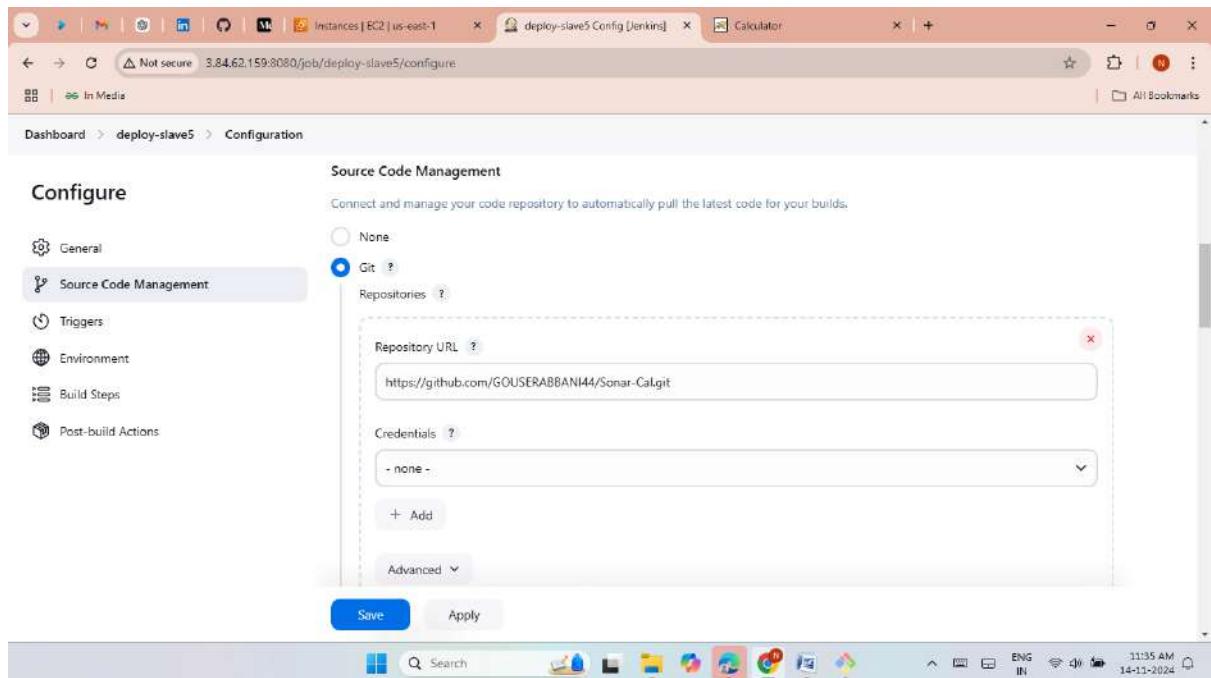
- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave5 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.83 GiB	0 B	4.83 GiB	0ms
	slave4	Linux (amd64)	In sync	3.35 GiB	0 B	3.35 GiB	22ms
	slave5	Linux (amd64)	In sync	5.36 GiB	0 B	5.36 GiB	77ms
	Data obtained	0.81 sec	0.8 sec	0.8 sec	0.83 sec	0.81 sec	0.8 sec

➤ Now create a job and give slave5 label.



➤ Now clone the java web calculator repo from github.



- Now write a script for install tomcat and maven to run the java calculator.
- Now Click on save.

```
#!/bin/bash

sudo wget https://downloads.apache.org/tomcat/tomcat-9/v9.0.97/bin/apache-tomcat-9.0.97.tar.gz

sudo tar -xvf apache-tomcat-9.0.97.tar.gz

sudo chmod +x apache-tomcat-9.0.97/bin

sudo yum -y install maven

mvn package

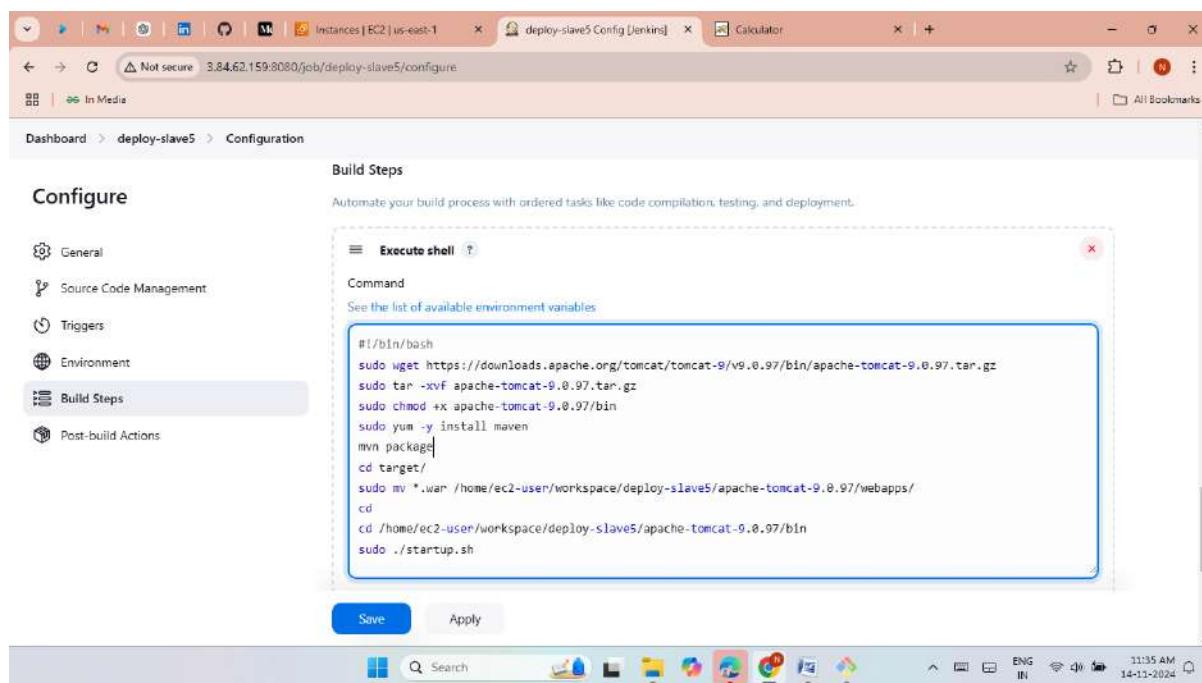
cd target/

sudo mv *.war /home/ec2-user/workspace/deploy-slave5/apache-tomcat-9.0.97/webapps/

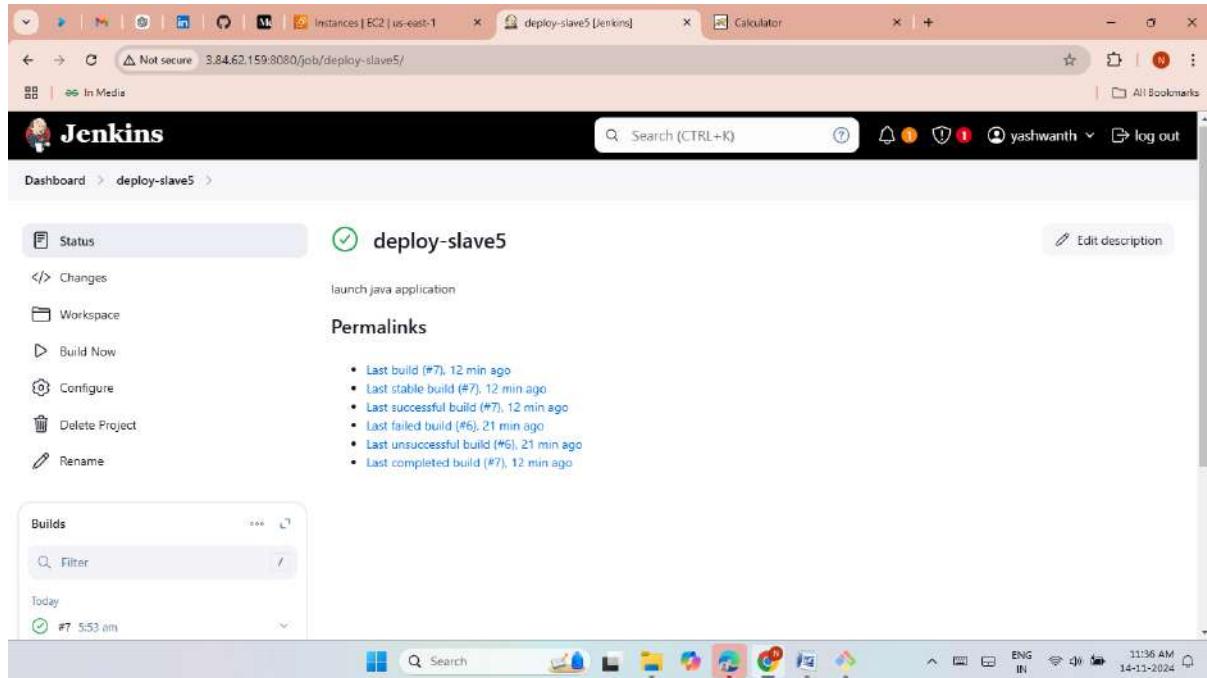
cd

cd /home/ec2-user/workspace/deploy-slave5/apache-tomcat-9.0.97/bin

sudo ./startup.sh
```



- Now click on build now.
- Here the job was success.



- Now remove some permissions in host-manager and manger context.xml files.

```
ec2-user@ip-172-31-84-244:~/workspace/deploy-slave5/apache-tomcat-9.0.97
[ec2-user@slave5 deploy-slave5]$ cd apache-tomcat-9.0.97/
[ec2-user@slave5 apache-tomcat-9.0.97]$ ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@slave5 apache-tomcat-9.0.97]$ cd conf/
bash: cd: cq: No such file or directory
[ec2-user@slave5 apache-tomcat-9.0.97]$ cd conf/
bash: cd: conf/: Permission denied
[ec2-user@slave5 apache-tomcat-9.0.97]$ sudo chmod 777 conf
[ec2-user@slave5 apache-tomcat-9.0.97]$ cd conf
[ec2-user@slave5 conf]$ ls
catalina.policy context.xml jaspic-providers.xsd server.xml tomcat-users.xsd
catalina.properties jaspic-providers.xml logging.properties tomcat-users.xml web.xml
[ec2-user@slave5 conf]$ sudo vi tomcat-users.xml
[ec2-user@slave5 conf]$ cd ..
[ec2-user@slave5 apache-tomcat-9.0.97]$ sudo find / -name context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562658200/apache-tomcat-9.0.97/conf/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562658200/apache-tomcat-9.0.97/webapps/docs/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562658200/apache-tomcat-9.0.97/webapps/examples/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562658200/apache-tomcat-9.0.97/webapps/host-manager/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562658200/apache-tomcat-9.0.97/webapps/manager/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562966981/apache-tomcat-9.0.97/conf/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562966981/apache-tomcat-9.0.97/webapps/docs/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562966981/apache-tomcat-9.0.97/webapps/examples/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562966981/apache-tomcat-9.0.97/webapps/host-manager/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731562966981/apache-tomcat-9.0.97/webapps/manager/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731563046516/apache-tomcat-9.0.97/conf/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731563046516/apache-tomcat-9.0.97/webapps/docs/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731563046516/apache-tomcat-9.0.97/webapps/examples/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731563046516/apache-tomcat-9.0.97/webapps/host-manager/META-INF/context.xml
/home/ec2-user/workspace/deploy-slave5_ws-cleanup_1731563046516/apache-tomcat-9.0.97/webapps/manager/META-INF/context.xml
[ec2-user@slave5 apache-tomcat-9.0.97]$ sudo vi
```

- Now create manager-gui role along with username and password.

```
ec2-user@ip-172-31-84-244:~/workspace/deploy-slave5/apache-tomcat-9.0.97/conf
[ec2-user@Slave5 apache-tomcat-9.0.97]$ ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd w
bash: cd: w: No such file or directory
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd webapps/
bash: cd: webapps/: Permission denied
[ec2-user@Slave5 apache-tomcat-9.0.97]$ sudo chmod 777 webapps/
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd webapps/
[ec2-user@Slave5 webapps]$ ls
docs examples host-manager manager ROOT webapp-1.8 webapp-1.8.war
[ec2-user@Slave5 webapps]$ cd ..
[ec2-user@Slave5 apache-tomcat-9.0.97]$ ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd ..
[ec2-user@Slave5 deploy-slave5]$ ls
apache-tomcat-9.0.97 apache-tomcat-9.0.97.tar.gz pom.xml src target
[ec2-user@Slave5 deploy-slave5]$ cd ..
[ec2-user@Slave5 workspace]$ ls
deploy-slave5 deploy-slave5_ws-cleanup_1731562658200 deploy-slave5_ws-cleanup_1731562966981 deploy-slave5_ws-cleanup_1731563046516
[ec2-user@Slave5 workspace]$ cd deploy-slave5
[ec2-user@Slave5 deploy-slave5]$ ls
apache-tomcat-9.0.97 apache-tomcat-9.0.97.tar.gz pom.xml src target
[ec2-user@Slave5 deploy-slave5]$ cd apache-tomcat-9.0.97/
[ec2-user@Slave5 apache-tomcat-9.0.97]$ ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps work
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd cq
bash: cd: cq: No such file or directory
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd conf/
bash: cd: conf/: Permission denied
[ec2-user@Slave5 apache-tomcat-9.0.97]$ sudo chmod 777 conf
[ec2-user@Slave5 apache-tomcat-9.0.97]$ cd conf
[ec2-user@Slave5 conf]$ ls
catalina.policy context.xml jaspic-providers.xsd server.xml tomcat-users.xml
catalina.properties jaspic-providers.xml logging.properties tomcat-users.xml web.xml
[ec2-user@Slave5 conf]$ sudo vi tomcat-users.xml
[ec2-user@Slave5 conf]$
```

```

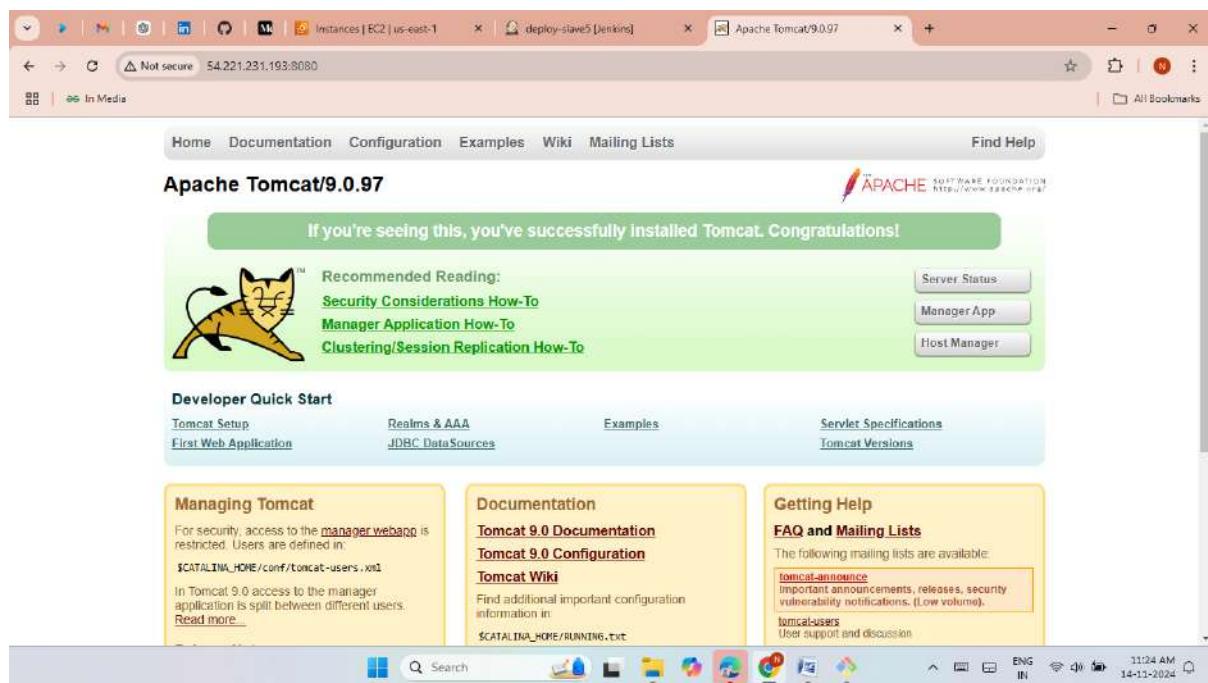
cc2-user@ip-172-31-84-244:~/workspace/deploy-slave5/apache-tomcat-9.0.97/conf

Built-in Tomcat manager roles:
- manager-gui - allows access to the HTML GUI and the status pages
- manager-script - allows access to the HTTP API and the status pages
- manager-jmx - allows access to the JMX proxy and the status pages
- manager-status - allows access to the status pages only

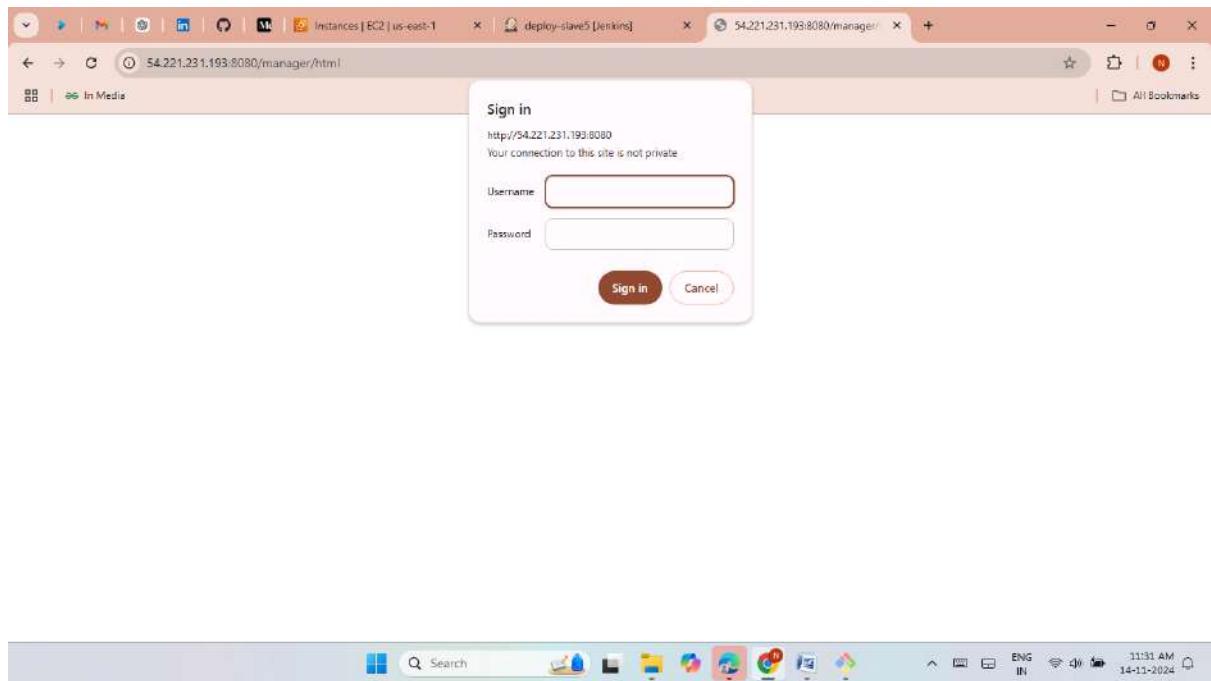
The users below are wrapped in a comment and are therefore ignored. If you
wish to configure one or more of these users for use with the manager web
application, do not forget to remove the <!-- ... --> that surrounds them. You
will also need to set the passwords to something appropriate.
-->
<!--
<user username="admin" password="" roles="manager-gui"/>
<user username="robot" password="" roles="manager-script"/>
-->
<!--
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- ... --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="" roles="tomcat"/>
<user username="both" password="" roles="tomcat,role1"/>
<user username="role1" password="" roles="role1"/>
-->
<role rolename="manager-gui"/>
<user username="yash" password="yash" roles="manager-gui"/>
</tomcat-users>
~-- INSERT --
56,30 Bot

```

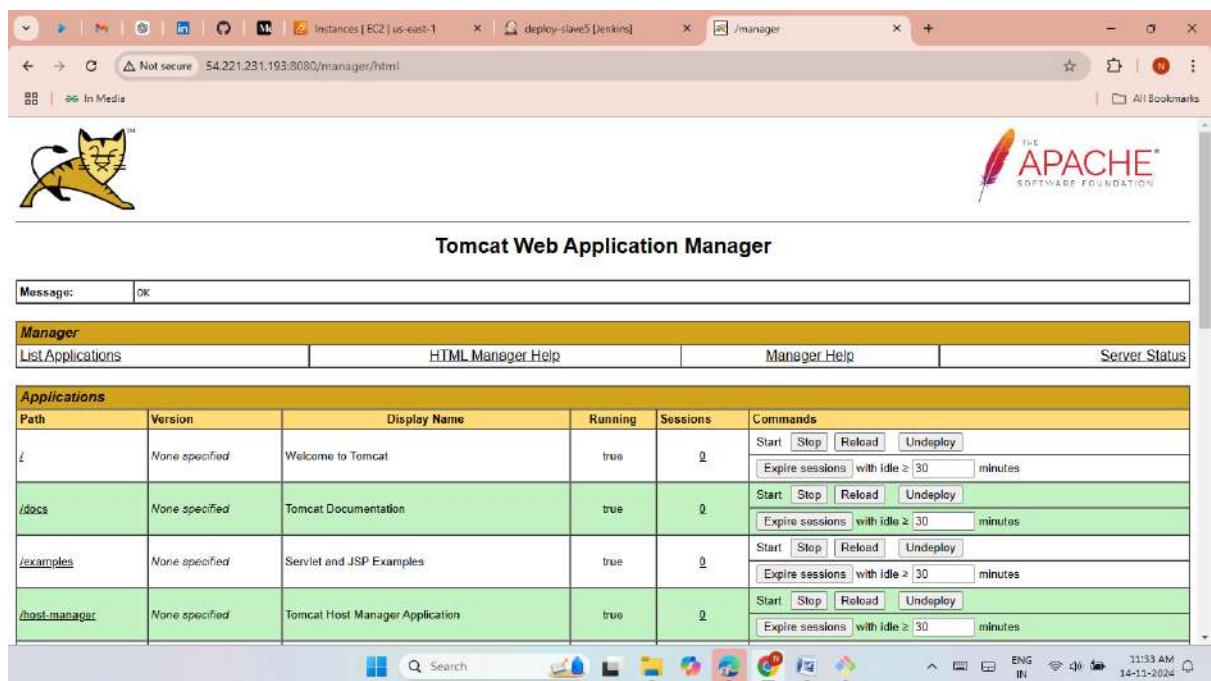
- Now copy the slave5 public ip and past it on google browser.
- The tomcat web server was hosted successfully.
- Click on manager app.



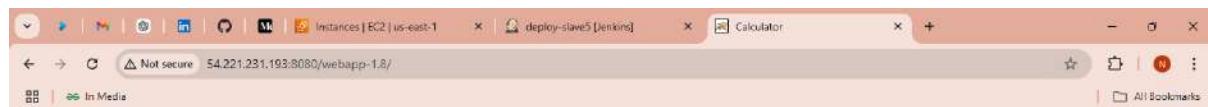
- Give manager-gui username and password.



- Now click on webapps-1.8.



- Here the java web calculator was hosted successfully.



Calculator

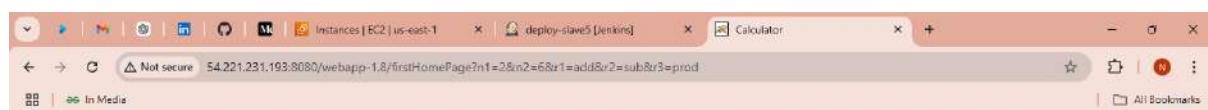
first number:

Second number :

Sravani-Addition

Mohammed-subtraction

Vinod-Multiplication



Addition

8

Substraction

4

Multiplication

12

Calculator

first number:

Second number :

Sravani-Addition

Mohammed-subtraction

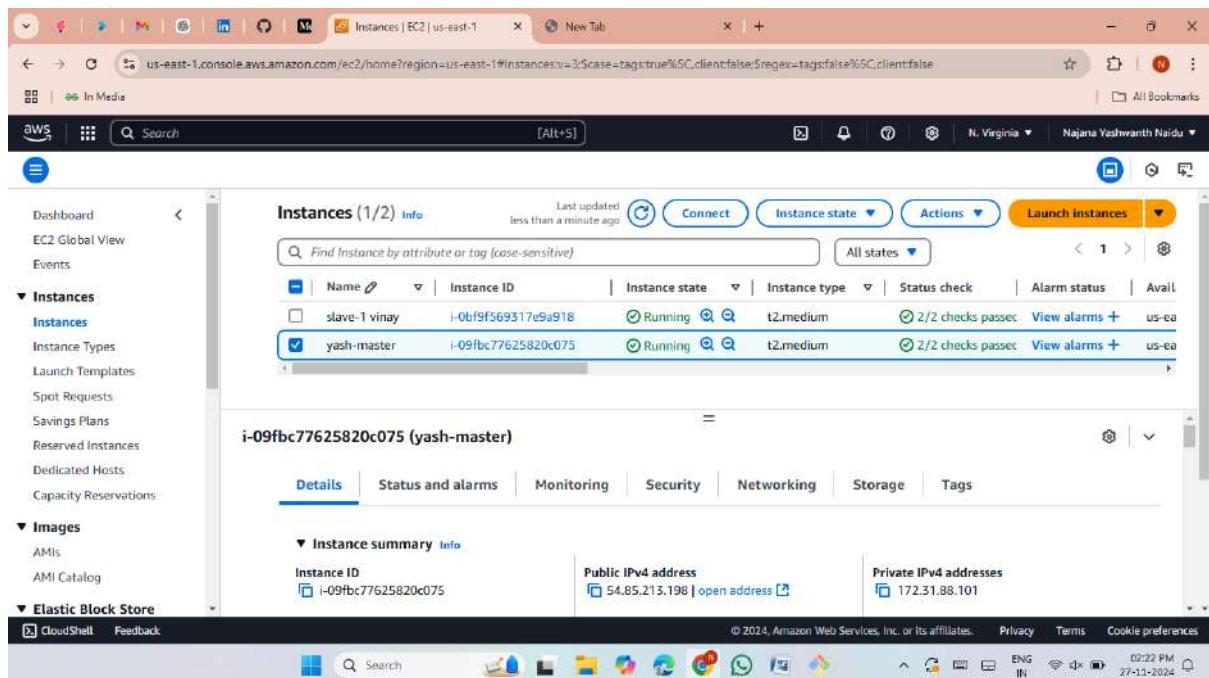
Vinod-Multiplication



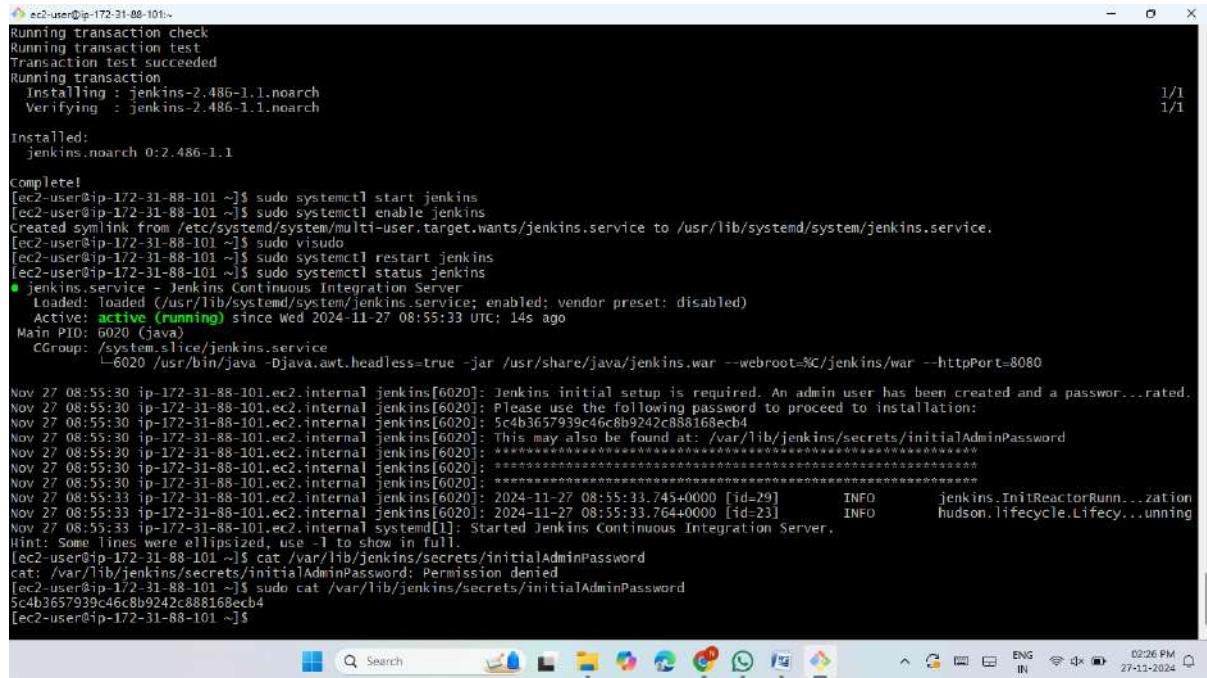
Method-3

Jostna Create one master and Slave-5 and Yaswanth creates 2nd slave node and Vinay creates 3rd slave and Ramsai creates 4th slave Rajesh creates slave-1 Same AWS account and Jostna deploy word press with bash script(slave-5) and yashwanth deploy word press with Docker Compose file (slave-2), and Vinay deploy word press with Terraform (slave-3) and Ramsai deploy the word press with poll-scm, build periodically, webhooks (slave-4) and Rajesh deploy the word press with k8's (slave-1) With in the same account and within the different AWS Accounts.

- Here I have worked as a master and slave-5 in a team project.
- Launch an ec2 instance along with required ports.



- Now connect master then install java.
- Install Jenkins then start and enable.
- Generate keys with the command ssh-keygen.



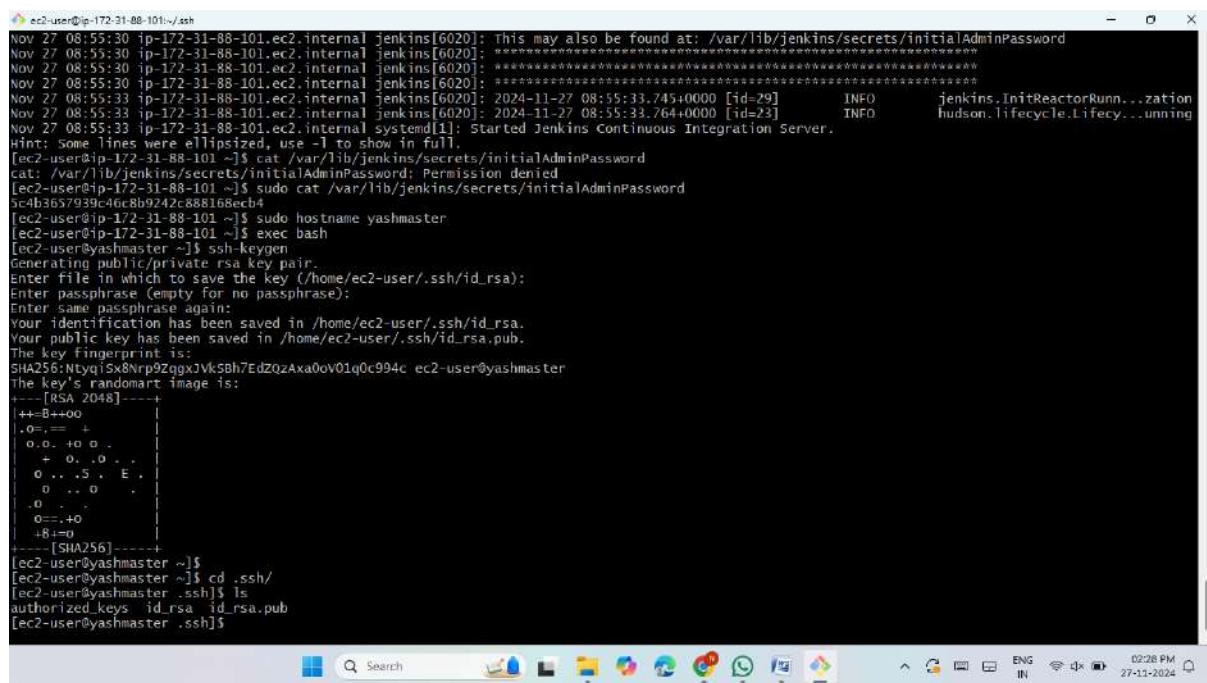
```

ec2-user@ip-172-31-88-101:~$ sudo systemctl start jenkins
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.486-1.1.noarch
  Verifying : jenkins-2.486-1.1.noarch

Installed:
  jenkins.noarch 0:2.486-1.1

Complete!
[ec2-user@ip-172-31-88-101 ~]$ sudo systemctl enable jenkins
Created symlink from /etc/systemd/system/multi-user.target.wants/jenkins.service to /usr/lib/systemd/system/jenkins.service.
[ec2-user@ip-172-31-88-101 ~]$ sudo visudo
[ec2-user@ip-172-31-88-101 ~]$ sudo systemctl restart jenkins
[ec2-user@ip-172-31-88-101 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: Loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
  Active: active (running) since wed 2024-11-27 08:55:33 UTC; 14s ago
    Main PID: 6020 (Java)
      CGroup: /system.slice/jenkins.service
             └─6020 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenkins/war --httpPort=8080

Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: Jenkins initial setup is required. An admin user has been created and a password...rated.
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: Please use the following password to proceed to installation:
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: 5c4b3657939c46c8b9242c888168ecb4
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: ****
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: ****
Nov 27 08:55:30 ip-172-31-88-101.ec2.internal jenkins[6020]: ****
Nov 27 08:55:33 ip-172-31-88-101.ec2.internal jenkins[6020]: 2024-11-27 08:55:33.745+0000 [id=29]           INFO      jenkins.InitReactorRunn...zation
Nov 27 08:55:33 ip-172-31-88-101.ec2.internal jenkins[6020]: 2024-11-27 08:55:33.764+0000 [id=23]           INFO      hudson.lifecycle.Lifecy...nning
Nov 27 08:55:33 ip-172-31-88-101.ec2.internal systemd[1]: Started Jenkins Continuous Integration Server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-88-101 ~]$ cat /var/lib/jenkins/secrets/initialAdminPassword
cat: /var/lib/jenkins/secrets/initialAdminPassword: Permission denied
[ec2-user@ip-172-31-88-101 ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
5c4b3657939c46c8b9242c888168ecb4
[ec2-user@ip-172-31-88-101 ~]$
```



```

ec2-user@ip-172-31-88-101:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_rsa.
Your public key has been saved in /home/ec2-user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ntyq15x8Nr9ZgxDVksBh7EdZQzAxa0oV01q994c ec2-user@yashmaster
The key's randomart image is:
++-[RSA 2048]----+
|++B++oo |
|.o, == + |
|o.o .+o o . |
| + o .. o .. |
| o ... . E . |
| o .. o . |
| .o . |
| o==,+o |
| +B+=o |
++-[SHA256]----+
[ec2-user@yashmaster ~]$ 
[ec2-user@yashmaster ~]$ cd .ssh/
[ec2-user@yashmaster .ssh]$ ls
authorized_keys id_rsa id_rsa.pub
[ec2-user@yashmaster .ssh]$
```

- Here the Jenkins was hosted successfully.

The screenshot shows the Jenkins dashboard at <http://54.85.213.198:8080>. The title bar says "Instances | EC2 | us-east-1" and "Dashboard [Jenkins]". The dashboard features a "Welcome to Jenkins!" message and sections for "Build Queue" (empty), "Create a job", "Manage Jenkins", "My Views", "Build Executor Status" (0/2), "Set up a distributed build", "Set up an agent", "Configure a cloud", and "Learn more about distributed builds". The status bar at the bottom shows "02:27 PM 27-12-2024".

- Now create a credentials with private key form master.

The screenshot shows the "Global credentials (unrestricted)" page at http://54.85.213.198:8080/manage/credentials/store/system/domain/_/. The title bar says "System > Global credentials (unrestricted)". The page lists a single credential: "yash-credentials" (ID), "ec2-user (jenkins)" (Name), "SSH Username with private key" (Kind), and "jenkins" (Description). A blue "Add Credentials" button is visible. The status bar at the bottom shows "02:30 PM 27-12-2024".

- Here I have worked as a slave-5 in a team project.
- Launch an ec2 instance along with required ports.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The main area displays 'Instances (1/3) Info' with a table showing three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
yash-slave-5	i-0ebd6d83e4a58a839	Running	t2.micro	Initializing	View alarms +	us-east-1
slave-1 vinay	i-0bf569317e9a918	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1
yash-master	i-09fb77625820c075	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1

Below the table, there's a detailed view for the first instance (yash-slave-5) with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Details tab shows the Instance summary with Public IPv4 address (3.88.174.132) and Private IPv4 addresses (172.31.81.67).

- Now connect Slave5 then install java.
- Now give private keys from master server.

```

Verifying : log4j-cve-2021-44228-hotpatch-1.3-7.amzn2.noarch 23/32
Verifying : j:java-17-amazon-corretto-17.0.13+11-1.amzn2.1.x86_64 24/32
Verifying : libxslt-1.2.28-6.amzn2.x86_64 25/32
Verifying : python-xml-3.2.1-4.amzn2.0.6.x86_64 26/32
Verifying : python-javapackages-3.4.1-11.amzn2.noarch 27/32
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64 28/32
Verifying : alsalib-1.1.4.1-2.amzn2.x86_64 29/32
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch 30/32
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64 31/32
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch 32/32

Installed:
  java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1      java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1  java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1  java-17-amazon-corretto-jmDNS.x86_64 1:17.0.13+11-1.amzn2.1

Dependency installed:
  alsalib.x86_64 0:1.1.4.1-2.amzn2          dejavu-fonts-common.noarch 0:2.33-6.amzn2      dejavu-sans-fonts.noarch 0:2.33-6.amzn2
  dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2  dejavu-serif-fonts.noarch 0:2.33-6.amzn2  fontconfig.x86_64 0:2.13.0-4.3.amzn2
  fontpackages-filesystem.noarch 0:1.44-8.amzn2  giflib.x86_64 0:4.1.6-9.amzn2.0.2        javapackages-tools.noarch 0:3.4.1-11.amzn2
  libICE.x86_64 0:1.0.9-9.amzn2.0.2         libSM.x86_64 0:1.2.2-2.amzn2.0.2        libX11.x86_64 0:1.6.7-3.amzn2.0.5
  libX11-common.noarch 0:1.6.7-3.amzn2.0.5   libXau.x86_64 0:1.0.8-2.1.amzn2.0.2       libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
  libXi.x86_64 0:1.7.9-1.amzn2.0.2         libXext.x86_64 0:1.1.5-3.amzn2.0.2       libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2    libXt.x86_64 0:1.1.28-6.amzn2              libXtst.x86_64 0:1.2.3-1.amzn2.0.2
  libxcb.x86_64 0:1.12-1.amzn2.0.2        libXtst.x86_64 0:1.1.28-6.amzn2.0.6     log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2

Complete!
[ec2-user@ip-172-31-81-67 ~]$ sudo hostname yashslave5
[ec2-user@ip-172-31-81-67 ~]$ exec bash
[ec2-user@yashslave5 ~]$ cd .ssh/
[ec2-user@yashslave5 .ssh]$ ls
authorized_keys
[ec2-user@yashslave5 .ssh]$ sudo vi authorized_keys
[ec2-user@yashslave5 .ssh]$ authorized_keys
[ec2-user@yashslave5 .ssh]$ cd
[ec2-user@yashslave5 ~]$ pwd
/home/ec2-user
[ec2-user@yashslave5 ~]$ |

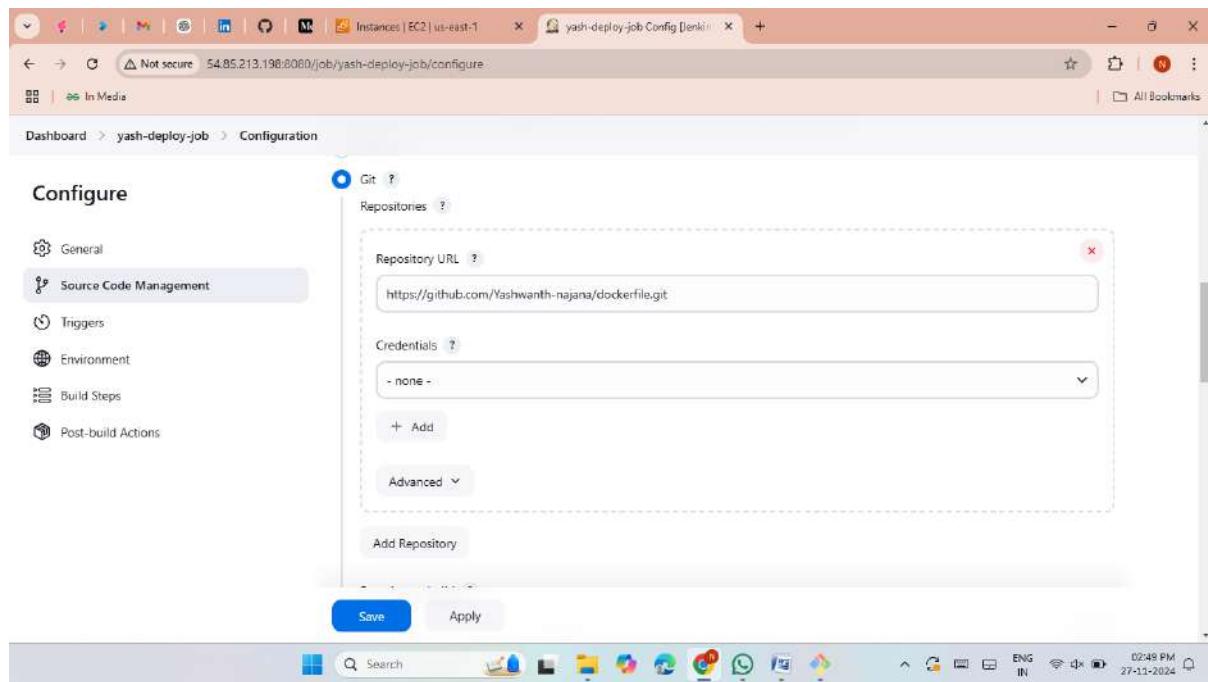
```

- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here the slave5 node is successfully created.

The screenshot shows the Jenkins interface for managing nodes. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (listing 'Built-In Node', 'slave-1', and 'yash-slave5'). The main area is titled 'Nodes' and displays a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. The table contains three rows: 'Built-In Node' (Linux (amd64), In sync, 4.96 GiB, 0 B, 4.96 GiB, 0ms), 'slave-1' (Linux (amd64), In sync, 5.05 GiB, 0 B, 5.05 GiB, 13ms), and 'yash-slave5' (Linux (amd64), In sync, 5.36 GiB, 0 B, 5.36 GiB, 66ms). A 'New Node' button is at the top right. Below the table are icons for 'S' (Small), 'M' (Medium), and 'L' (Large). At the bottom right is a 'Legend' link. The browser address bar shows 'Not secure 54.85.213.198:8080/manage/computer/'.

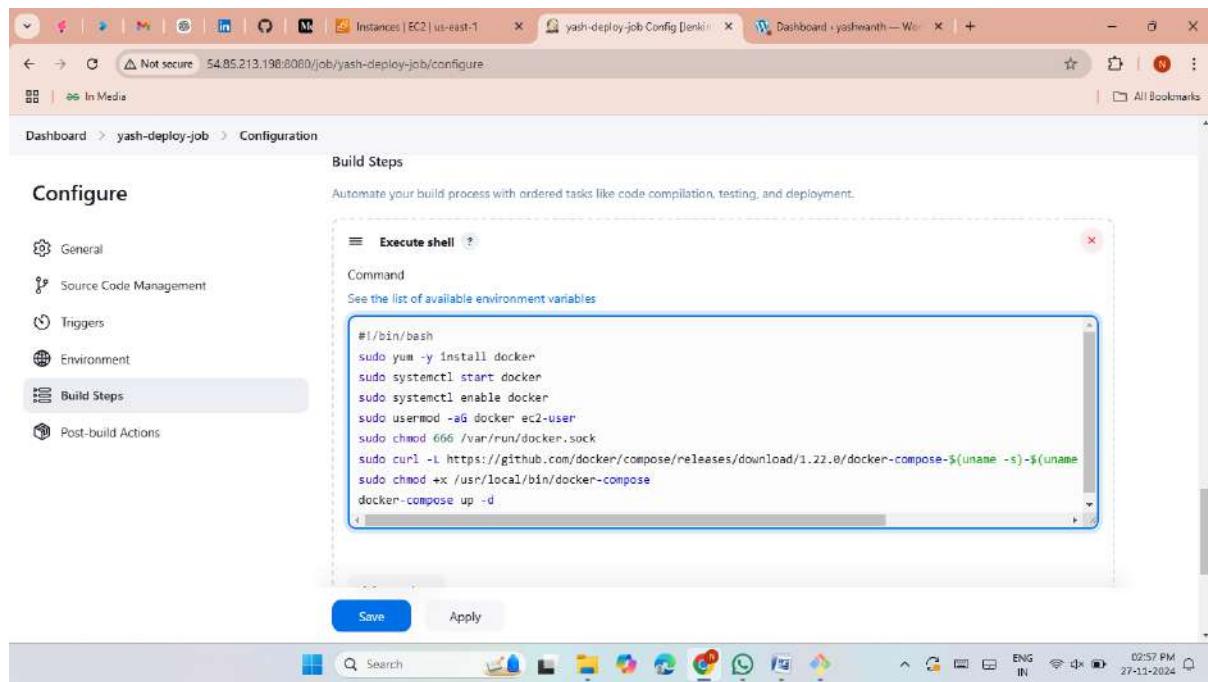
- Now create a job and give slave5 label.
- Now clone the wordpress application repo from github.

The screenshot shows the Jenkins job configuration page for 'yash-deploy-job'. The left sidebar lists 'General', 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. Under 'General', there is a 'Label Expression' field containing 'yash'. Below it, a note says 'Label yash matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' Under 'Source Code Management', the 'Git' option is selected, and the 'Repository URL' field is empty. At the bottom are 'Save' and 'Apply' buttons. The browser address bar shows 'Not secure 54.85.213.198:8080/job/yash-deploy-job/configure'.

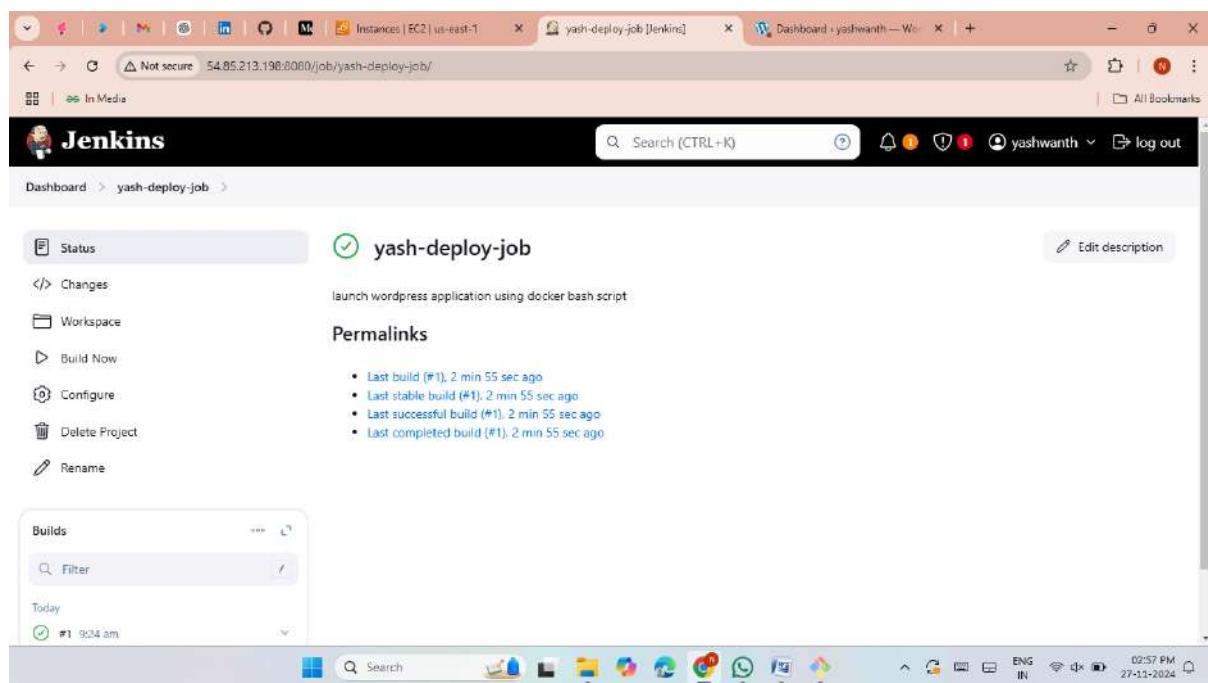


- Now run the following commands to create image and run container.

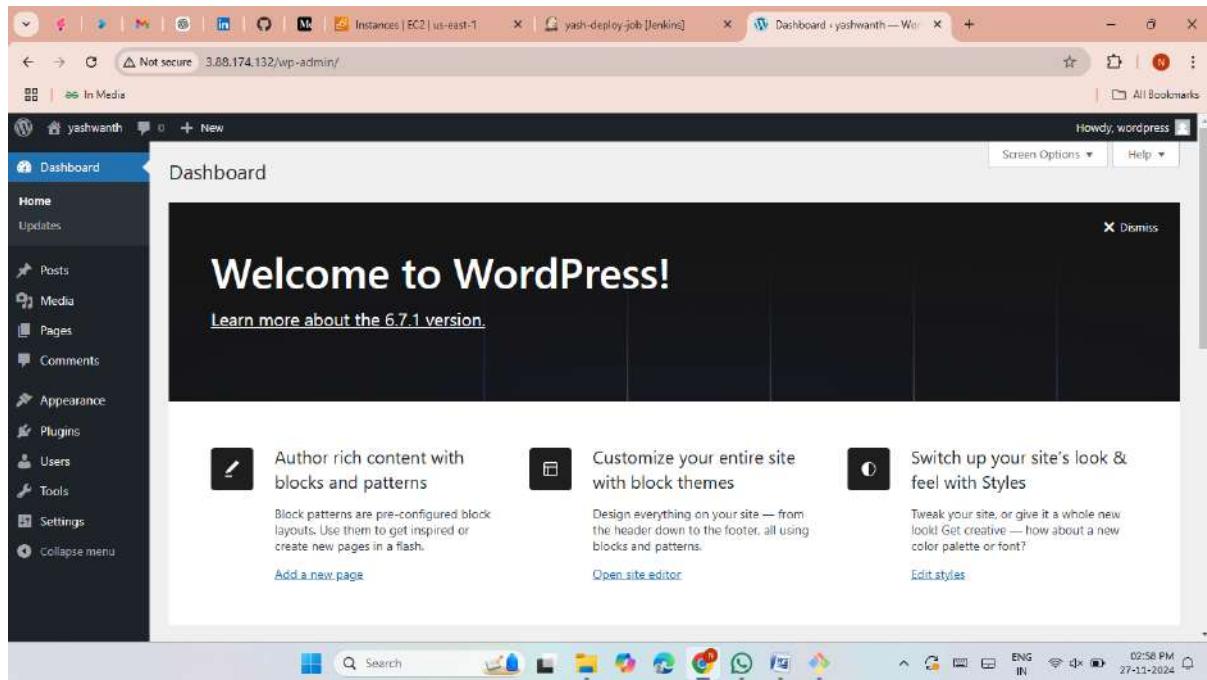
```
#!/bin/bash  
  
sudo yum -y install docker  
  
sudo systemctl start docker  
  
sudo systemctl enable docker  
  
sudo usermod -aG docker ec2-user  
  
sudo chmod 666 /var/run/docker.sock  
  
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose  
  
sudo chmod +x /usr/local/bin/docker-compose  
  
docker-compose up -d
```



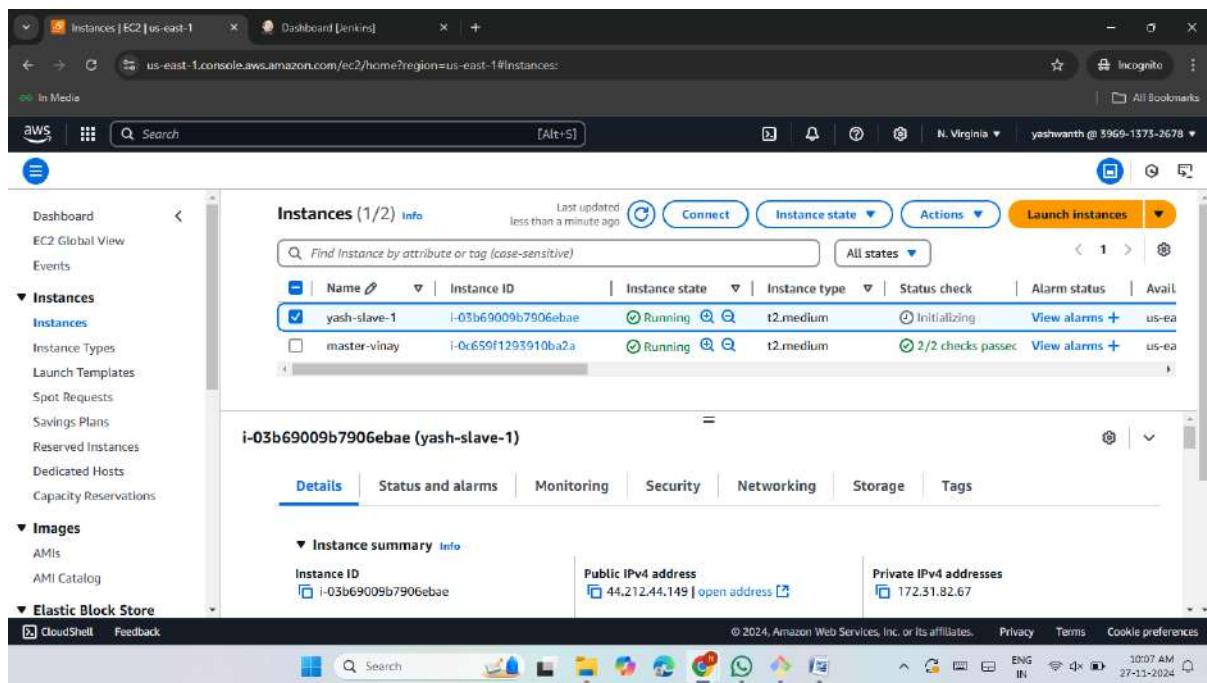
- Now click on build now.
- Here the job was success.



- Now copy the slave5 public ip and past it on google browser.
- The wordpress was hosted successfully.



- Here I have worked as a slave-1 in a team project.
- Launch an ec2 instance along with required ports.



- Now connect Slave1 then install java.
- Now give private keys from master server.

```

ec2-user@ip-172-31-82-67:~/.ssh
Verifying : python-lxml-3.2.1-4.amzn2.0.6.x86_64
Verifying : python-javapackages-3.4.1-11.amzn2.noarch
Verifying : libxtst-1.2.3-1.amzn2.0.2.x86_64
Verifying : alsalib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch
Verifying : 26/32
Verifying : 27/32
Verifying : 28/32
Verifying : 29/32
Verifying : 30/32
Verifying : 31/32
Verifying : 32/32

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-debugsymbols.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-headless.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-jmods.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
Fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2
dejavu-fonts-common.noarch 0:2.33-6.amzn2
dejavu-sans-mono-fonts.noarch 0:2.33-6.amzn2
fontconfig.x86_64 0:2.13.0-4.3.amzn2
giflib.x86_64 0:4.1.6-9.amzn2.0.2
libICE.x86_64 0:1.0.9-9.amzn2.0.2
libX11.x86_64 0:1.6.7-3.amzn2.0.5
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXi.x86_64 0:1.7.9-1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libXt.x86_64 0:1.1.5-3.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.amzn2
python-lxml.x86_64 0:3.2.1-4.amzn2.0.6

Complete!
[ec2-user@ip-172-31-82-67 ~]$ sudo hostname yashslavel
[ec2-user@ip-172-31-82-67 ~]$ exec bash
[ec2-user@yashslavel ~]$ cd .ssh/
[ec2-user@yashslavel .ssh]$ ls
authorized_keys
[ec2-user@yashslavel .ssh]$ sudo vi authorized_keys
[ec2-user@yashslavel .ssh]$ pwd
/home/ec2-user/.ssh
[ec2-user@yashslavel .ssh]$ |

```

- Now create a node for slave1 server.
- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here the slave1 node is successfully created.

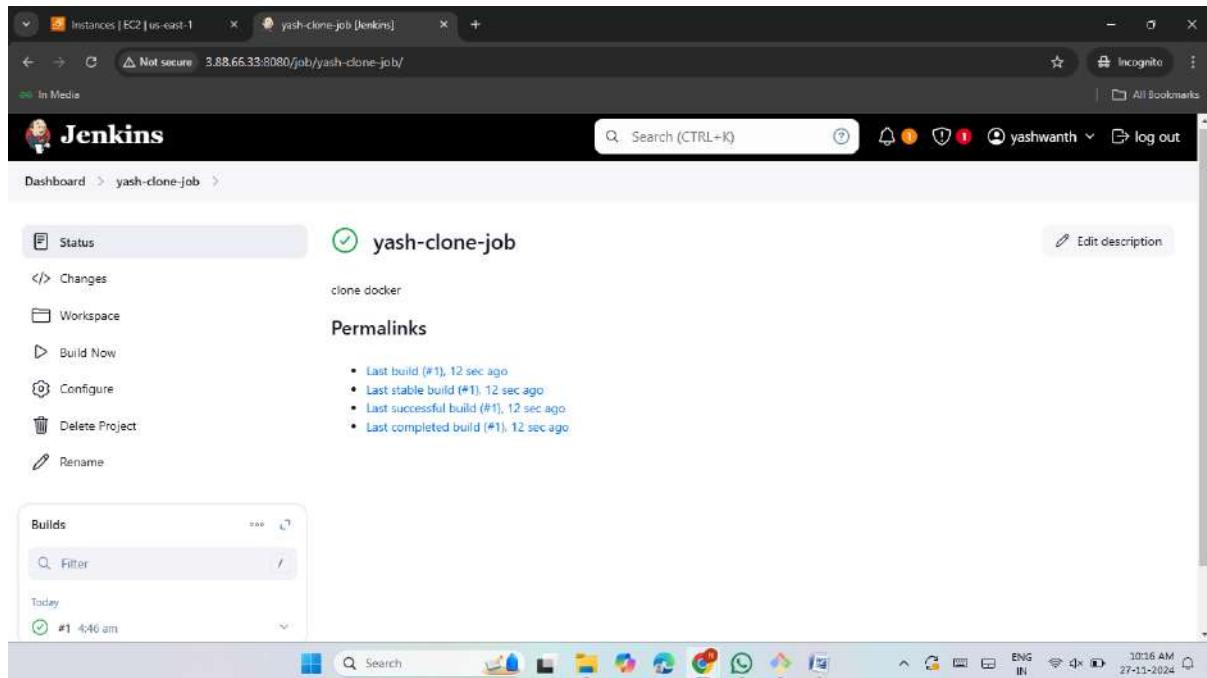
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.97 GiB	0 B	4.97 GiB	0ms
	yash-slave1	Linux (amd64)	In sync	5.36 GiB	0 B	5.36 GiB	60ms
	Data obtained	3.8 sec	3.7 sec	3.7 sec	3.7 sec	3.7 sec	3.7 sec

- Now create a job and give slave1 label.
- Now clone the wordpress application repo form github.
- Now Click on save.

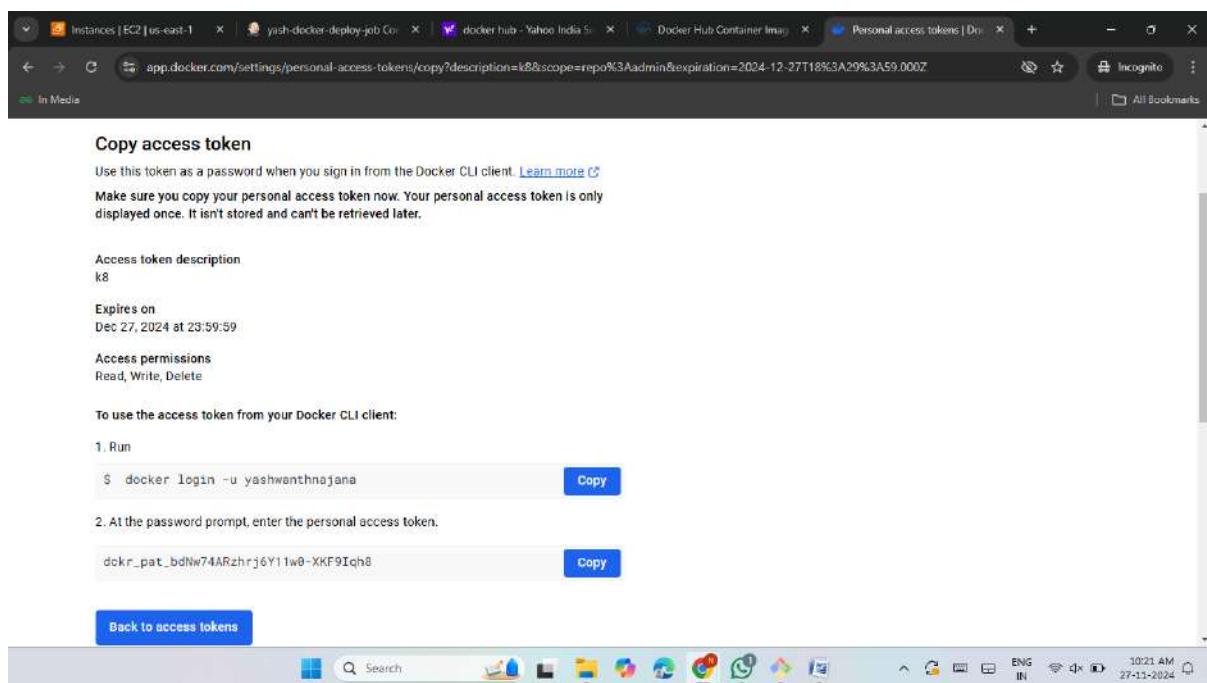
The screenshot shows the Jenkins job configuration page for 'yash-clone-job'. The 'General' tab is selected. Under 'Restrict where this project can be run', the 'Label Expression' field contains 'yash'. A note below states: 'Label yash matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' Below the General tab, there are tabs for Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions.

The screenshot shows the Jenkins job configuration page for 'yash-clone-job'. The 'Source Code Management' tab is selected. It is configured to use 'Git' with a repository URL of 'https://github.com/Yashwanth-najana/dockerfile.git'. The 'Credentials' dropdown is set to '- none -'. Below the Source Code Management tab, there are tabs for General, Triggers, Environment, Build Steps, and Post-build Actions.

- Now click on build now.
- Here the job was success.



- Now create a token in dockerhub for Jenkins.



- Now run the following commands to create image and run container.
- Push the images into dockerhub.

```
#!/bin/bash

sudo yum -y install docker

sudo systemctl start docker

sudo systemctl enable docker

sudo usermod -aG docker ec2-user

sudo chmod 666 /var/run/docker.sock

sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

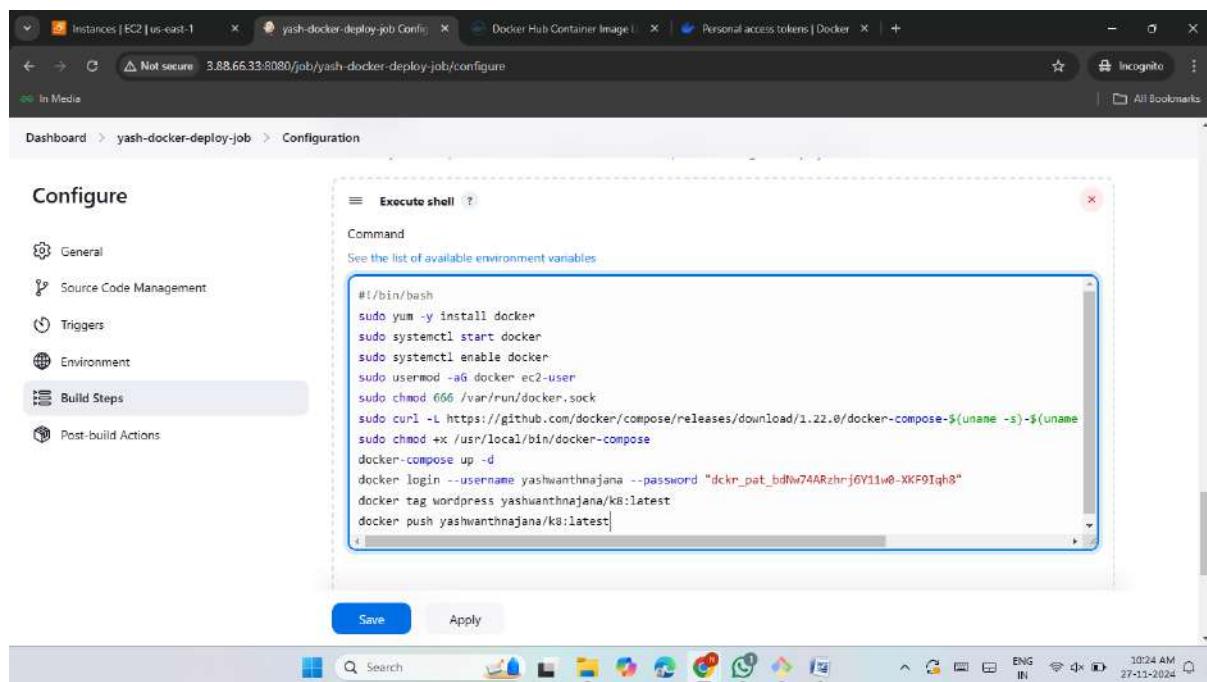
sudo chmod +x /usr/local/bin/docker-compose

docker-compose up -d

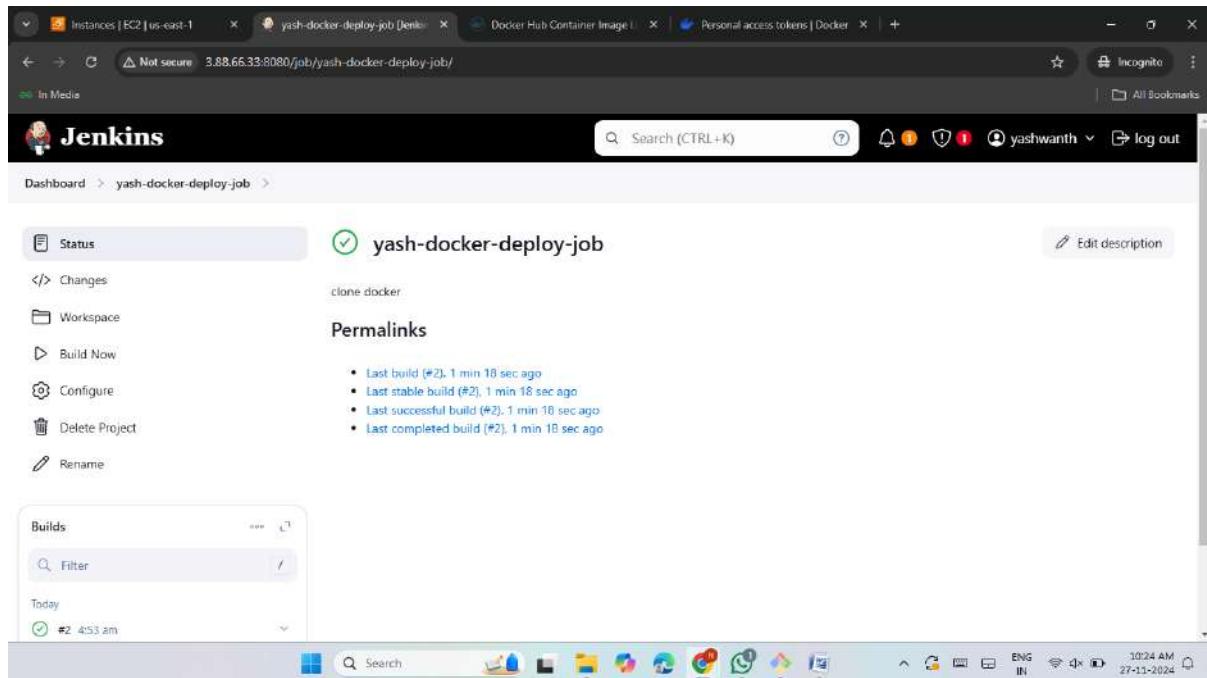
docker login --username yashwanthnajana --password "dckr_pat_bdNw74ARzhrj6Y11w0-XKF9Iqh8"

docker tag wordpress yashwanthnajana/k8:latest

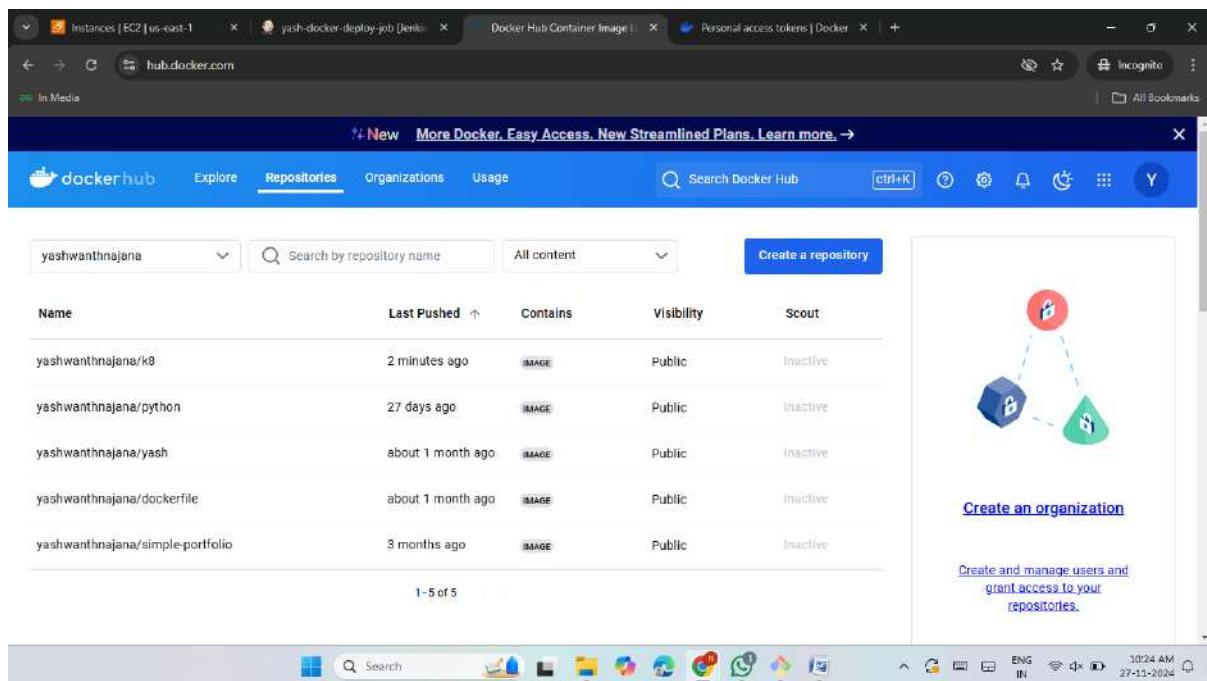
docker push yashwanthnajana/k8:latest
```



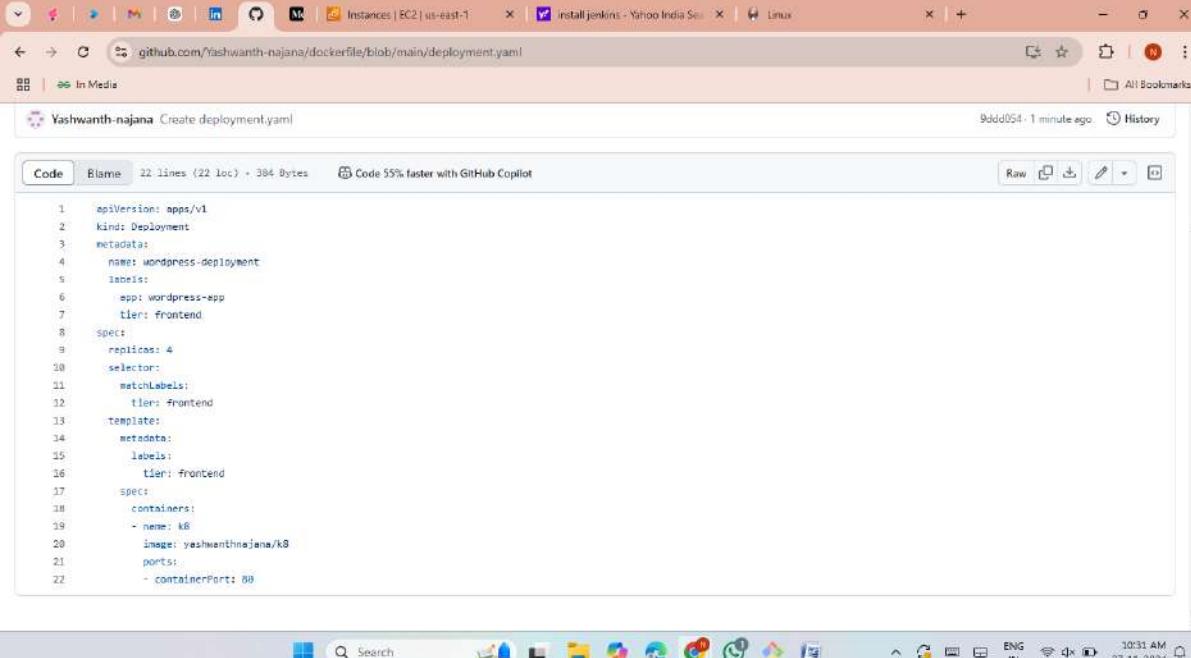
- Now click on build now.
- Here the job was success.



- Here the images is successfully pushed into dockerhub.



- Create a deployment.yaml file into github.

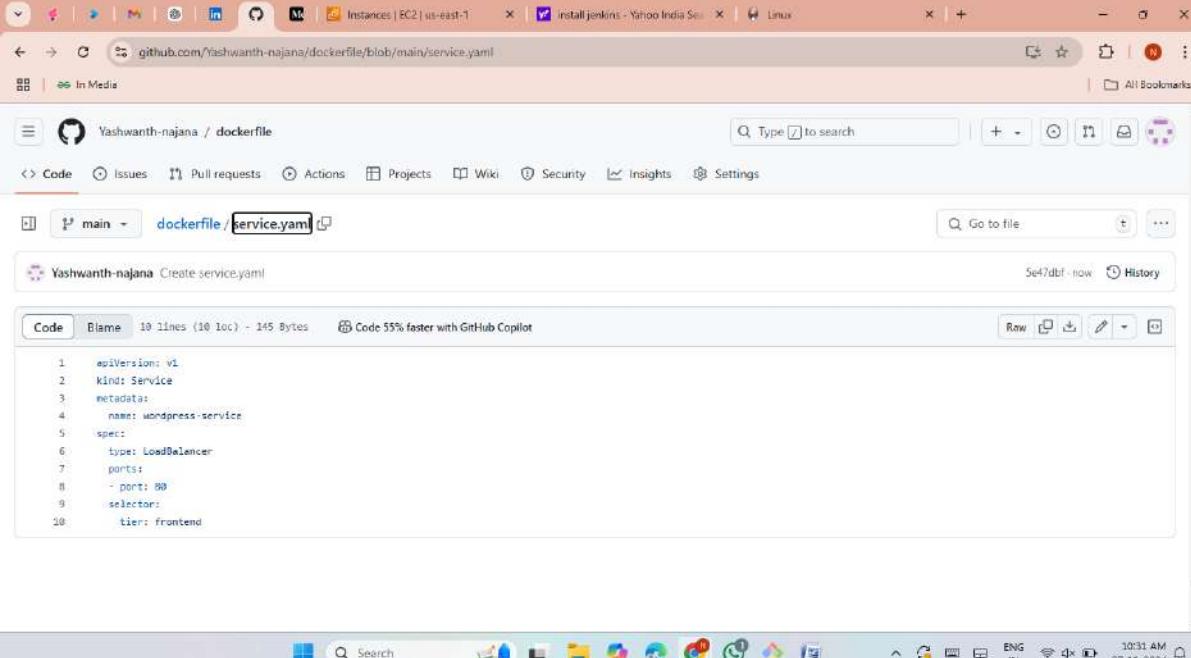


```

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: wordpress-deployment
5   labels:
6     app: wordpress-app
7     tier: frontend
8 spec:
9   replicas: 4
10  selector:
11    matchLabels:
12      tier: frontend
13  template:
14    metadata:
15      labels:
16        tier: frontend
17    spec:
18      containers:
19        - name: k8
20          image: yashwanthnajana/k8
21          ports:
22            - containerPort: 80

```

- Now create service.yaml file in github.



```

1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: wordpress-service
5 spec:
6   type: LoadBalancer
7   ports:
8     - port: 80
9   selector:
10    tier: frontend

```

- Now create role with admin full permissions then attach to slave server.

The screenshot shows the AWS IAM Roles page. A green success message at the top says "Role yash-role created." Below it, the "Roles (1/3)" section lists a single role named "yash-role". The role has "AWS Service: ec2" selected under "Trusted entities". At the bottom of the page, there are sections for "Access AWS from your non AWS workloads", "X.509 Standard", and "Temporary credentials". The status bar at the bottom right shows the date and time as "© 2024, Amazon Web Services, Inc. or its affiliates. 10:38 AM 27-11-2024".

The screenshot shows the AWS EC2 Instances page. A green success message at the top says "Successfully attached yash-role to instance i-03b69009b7906ebae". The "Instances (1/2)" table shows two instances: "master-vinay" and "yash-slave-1". The "yash-slave-1" row is selected. The instance details for "i-03b69009b7906ebae (yash-slave-1)" are shown below, including the Public IPv4 address (44.212.44.149) and Private IPv4 address (172.31.82.67). The status bar at the bottom right shows the date and time as "© 2024, Amazon Web Services, Inc. or its affiliates. 10:39 AM 27-11-2024".

- Now write the following command to install kops and kubectl.
- Run the deployment and service files.

```

#!/bin/bash

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repodata/repomd.xml.key
EOF

sudo yum install kubectl -y

curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s
https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d
''' -f 4)/kops-linux-amd64

chmod +x kops

sudo mv kops /usr/local/bin/kops

aws s3 mb s3://yash-k8

export KOPS_STATE_STORE=s3://yash-k8

kops create cluster --name yashwanth-k8s.local --state s3://yash-k8 --zones us-east-1a,us-
east-1b --node-count 2 --yes

kops validate cluster

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl get all

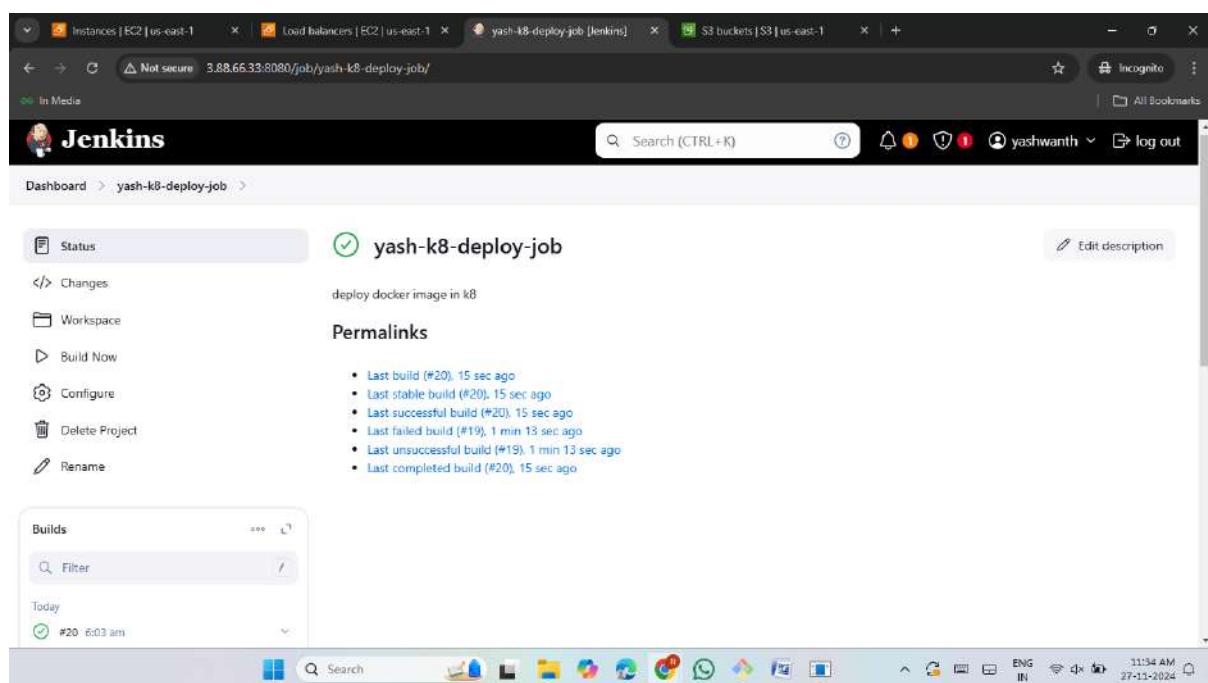
```

The screenshot shows the AWS Lambda configuration interface for a job named 'yash-k8-deploy-job'. The 'Build Steps' tab is selected. A large code editor window displays a bash script:

```
#!/bin/bash
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repo/repodata/repomd.xml.key
EOF
sudo yum install kubectl -y
curl -Lo kops https://github.com/kubernetes/kops/releases/download/v1.31.0/kops-1.31.0-linux-amd64.tar.gz
tar -xvf kops-1.31.0-linux-amd64.tar.gz
sudo mv kops /usr/local/bin/kops
aws s3 mb s3://yash-k8
export KOPS_STATE_STORE=s3://yash-k8
kops create cluster --name yashwanth-k8s.local --state s3://yash-k8 --zones us-east-1a,us-east-1b --node-count=2
kops validate cluster
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
kubectl get all
```

Below the code editor are 'Save' and 'Apply' buttons. The task bar at the bottom shows various icons and the date/time: 11:34 AM 27-11-2024.

➤ Here the job was success.



- Here the loadbalancer was successfully created.

The screenshot shows the AWS CloudWatch Metrics Insights interface. A search bar at the top contains the query: `CloudWatch Metrics Insights metrics @ "last hour"`. The results table has three columns: `Time`, `CloudWatch Metrics Insights metrics`, and `Value`. The data shows a single metric named `CloudWatch Metrics Insights metrics` with a value of `1` across all time points from `1 hour ago` to the present. The interface includes various navigation and configuration buttons.

➤ Here the s3 was created.

The screenshot shows the AWS S3 console under the 'Amazon S3' section. On the left sidebar, there are links for 'Buckets', 'Access Grants', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. Below these, there are sections for 'Block Public Access settings for this account' and 'Storage Lens' with 'Dashboards', 'Storage Lens groups', and 'AWS Organizations settings'. The main content area has a heading 'Account snapshot - updated every 24 hours' with a link to 'All AWS Regions'. It also mentions 'Storage lens provides visibility into storage usage and activity trends' with a 'Learn more' link. There are two tabs: 'General purpose buckets' (selected) and 'Directory buckets'. Below this, a table lists 'General purpose buckets (1)'. The table has columns for 'Name', 'AWS Region', 'IAM Access Analyzer', and 'Creation date'. A search bar at the top of the table says 'Find buckets by name'. The single bucket listed is 'yash-k8' in the 'US East (N. Virginia) us-east-1' region, created on November 27, 2024, at 11:03:15 UTC+05:30. Action buttons for the bucket include 'Info', 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'.

➤ Here the required instances also created.

The screenshot shows the AWS EC2 Instances page with the following details:

- Instances (11) Info:** Last updated less than a minute ago.
- Actions:** Connect, Instance state, Actions, Launch instances.
- Filter:** All states.
- Table Headers:** Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Available.
- Table Data:**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
yash-slave-1	i-03b69009b7906beae	Running	t2.medium	2/2 checks passed	View alarms	us-east-1
nodes-us-east-1a	i-045b5d957349d15e5	Running	t3.medium	3/3 checks passed	View alarms	us-east-1
nodes-us-east-1b	i-0a7266e3255e6beba	Running	t3.medium	3/3 checks passed	View alarms	us-east-1
- Select an instance:** A dropdown menu is open, listing the three instances: control-plane-us-east-1a, nodes-us-east-1a, and nodes-us-east-1b.

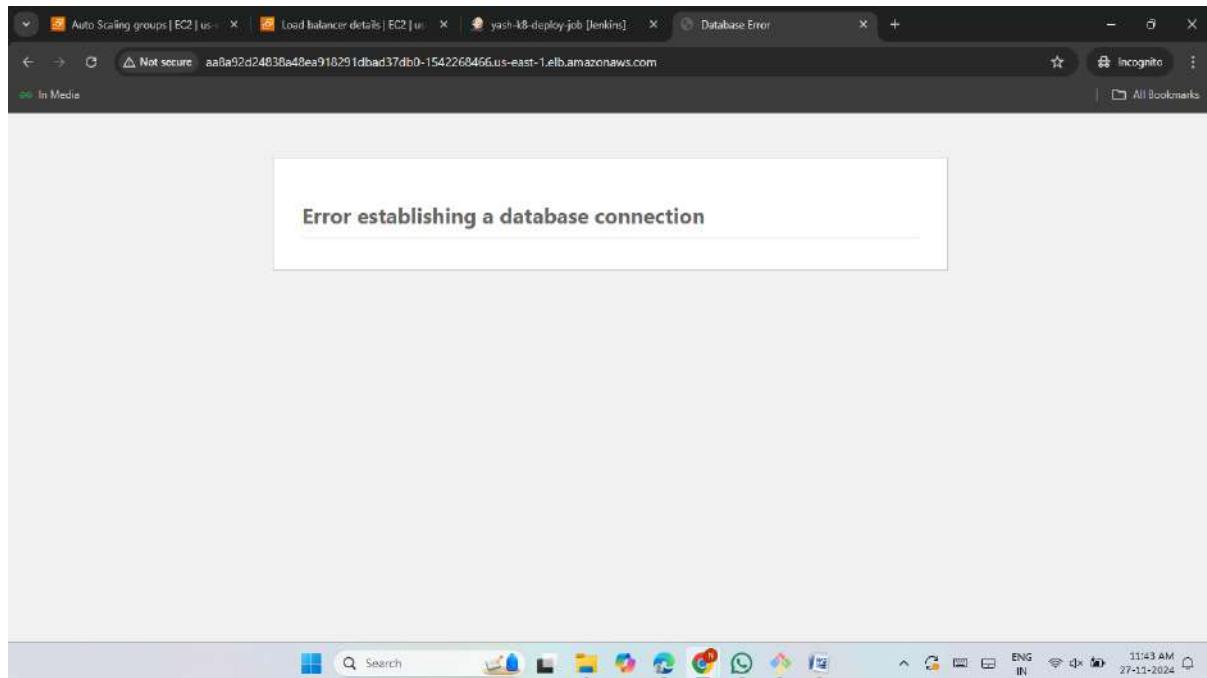
➤ Here the auto scaling was created.

The screenshot shows the AWS Auto Scaling groups page with the following details:

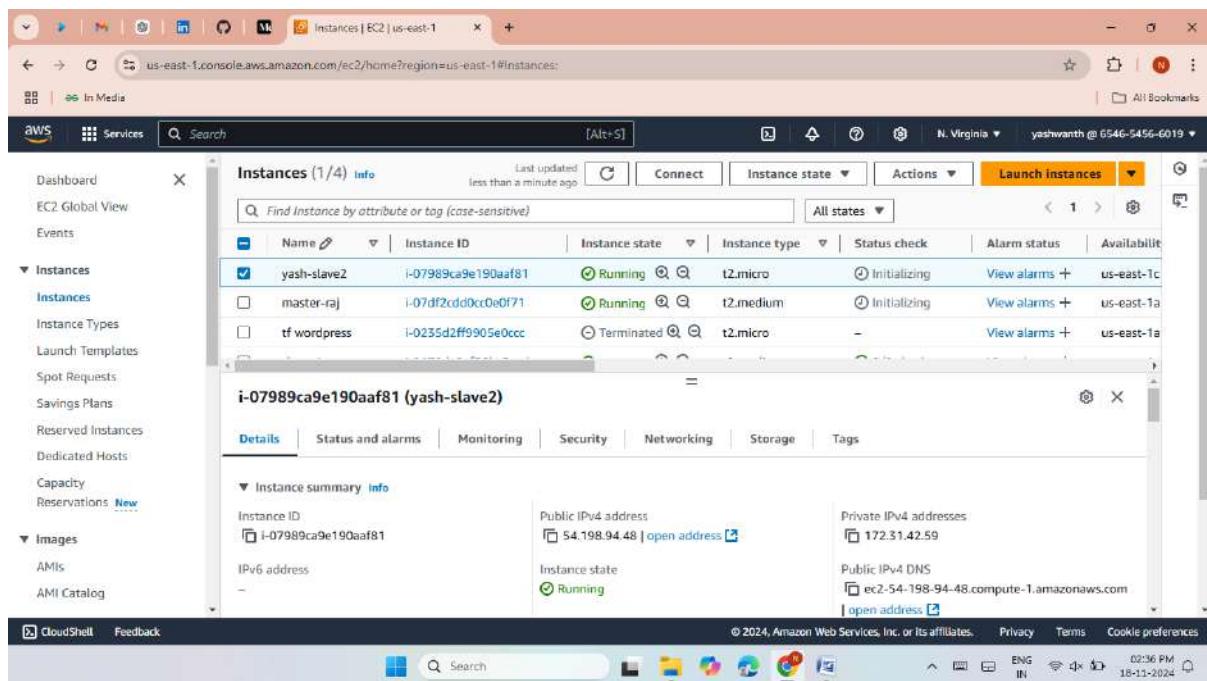
- Auto Scaling groups (3) Info:** Last updated less than a minute ago.
- Actions:** Launch configurations, Launch templates, Actions, Create Auto Scaling group.
- Filter:** All states.
- Table Headers:** Name, Launch template/configuration, Instances, Status, Desired capacity.
- Table Data:**

Name	Launch template/configuration	Instances	Status	Desired capacity
control-plane-us-east-1a	control-plane-us-east-1a.masters.yashwanth-k8s.local	1	-	1
nodes-us-east-1a	nodes-us-east-1a.yashwanth-k8s.local	1	-	1
nodes-us-east-1b	nodes-us-east-1b.yashwanth-k8s.local	1	-	1

- Now copy the slave1 public ip and past it on google browser.
- The wordpress was hosted successfully.



- Here I have worked as a slave-2 in a team project.
- Launch an ec2 instance along with required ports.



- Now connect Slave2 then install java.
- Now give private keys from master server.

```

Verifying : alsa-lib-1.1.4.1-2.amzn2.x86_64
Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch
Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64
Verifying : javapackages-tools-3.4.1-11.amzn2.noarch

Installed:
java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
alsa-lib.x86_64 0:1.1.4.1-2.amzn2
dejavu-sans-fonts.noarch 0:2.33-6.amzn2
dejavu-serif-fonts.noarch 0:2.33-6.amzn2
fontpackages-filesystem.noarch 0:1.44-8.amzn2
javapackages-tools.noarch 0:3.4.1-11.amzn2
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libXslt.x86_64 0:1.1.28-6.amzn2
python-javapackages.noarch 0:3.4.1-11.amzn2

Complete!
[ec2-user@ip-172-31-42-59 ~]$ sudo hostname yashslave2
[ec2-user@ip-172-31-42-59 ~]$ exec bash
[ec2-user@yashslave2 ~]$ cd .ssh/
[ec2-user@yashslave2 .ssh]$ ls
authorized_keys
[ec2-user@yashslave2 .ssh]$ sudo vi authorized_keys
[ec2-user@yashslave2 .ssh]$ cd
[ec2-user@yashslave2 ~]$ pwd
/home/ec2-user
[ec2-user@yashslave2 ~]$

```

- Now create a node for slave2 server.
- Give details for slave node.
- Here we have to give slave2 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here the slave2 node is successfully created.

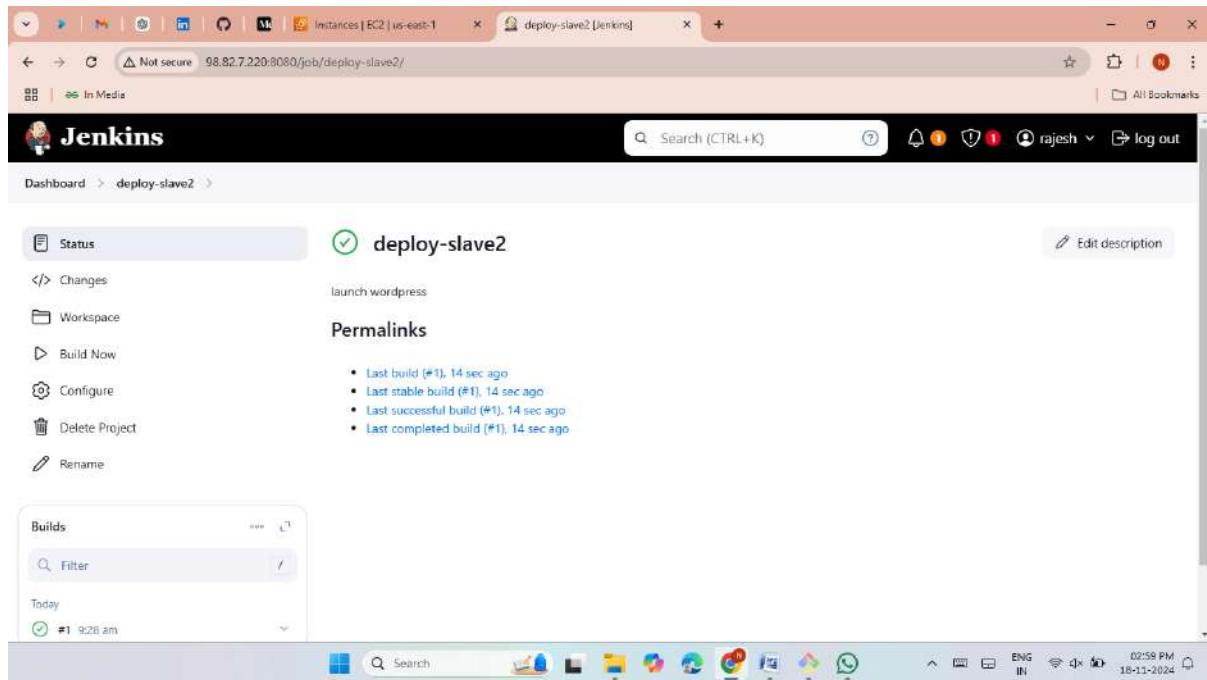
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	5.02 GiB	0 B	5.02 GiB	0ms
	Slave5-JP	Linux (amd64)	In sync	5.86 GiB	0 B	5.86 GiB	23ms
	yash-slave2	Linux (amd64)	In sync	5.36 GiB	0 B	5.36 GiB	22ms
	Data obtained	40 ms	40 ms	39 ms	40 ms	39 ms	38 ms

- Now create a job and give slave2 label.
- Now clone the wordpress application repo form github.
- Now Click on save.

The screenshot shows the Jenkins configuration page for the 'clone-slave2' job. The 'General' section is selected. Under 'Label Expression', the value 'yash' is entered, with a note below stating 'Label yash matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' The 'Source Code Management' section is expanded, showing 'Git' selected with a 'Repository' dropdown. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration page for the 'deploy-slave2' job. The 'Source Code Management' section is selected. Under 'Git', the 'Repository URL' is set to 'https://github.com/Yashwanth-naJana/dockerfile.git'. The 'Credentials' dropdown is set to '- none -'. At the bottom are 'Save' and 'Apply' buttons.

- Now click on build now.
- Here the job was success.



- Now run the following commands to create image and run container.

```
sudo yum -y install docker
sudo systemctl start docker
sudo systemctl enable docker
sudo chmod 666 /var/run/docker.sock
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose up -d
```

```

ec2-user@yashslave2:~$ docker.x86_64 0:25.0.6-1.amzn2.0.2
Dependency Installed:
  containerd.x86_64 0:1.7.23-1.amzn2.0.2           libcgroup.x86_64 0:0.41-21.amzn2          pigz.x86_64 0:2.3.4-1.amzn2.0.1
  runc.x86_64 0:1.1.14-1.amzn2

Complete!
[ec2-user@yashslave2 ~]$ sudo systemctl start docker
[ec2-user@yashslave2 ~]$ sudo systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@yashslave2 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@yashslave2 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
     Active: active (running) since Mon 2024-11-18 09:30:17 UTC; 33s ago
       Docs: https://docs.docker.com
      Main PID: 13658 (dockerd)
        CGroup: /system.slice/docker.service
              └─13658 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Nov 18 09:30:17 yashslave2 systemd[1]: Starting Docker Application Container Engine...
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.456300625Z" level=info msg="Starting up"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.547898087Z" level=info msg="Loading containers: start."
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.765201324Z" level=info msg="Loading containers: done."
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.780312822Z" level=warning msg="WARNING: bridge-nf-call-...sabled"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.781016672Z" level=warning msg="WARNING: bridge-nf-call-...sabled"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.781231099Z" level=info msg="Docker daemon" commit=b08a5...=25.0.6
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.781456830Z" level=info msg="Daemon has completed initialization"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.812751675Z" level=info msg="API listen on /run/docker.sock"
Nov 18 09:30:17 yashslave2 systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@yashslave2 ~]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
[ec2-user@yashslave2 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[ec2-user@yashslave2 ~]$ 

```

03:01 PM IN 18-11-2024

```

ec2-user@yashslave2:~$ created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@yashslave2 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@yashslave2 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
     Active: active (running) since Mon 2024-11-18 09:30:17 UTC; 33s ago
       Docs: https://docs.docker.com
      Main PID: 13658 (dockerd)
        CGroup: /system.slice/docker.service
              └─13658 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Nov 18 09:30:17 yashslave2 systemd[1]: Starting Docker Application Container Engine...
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.456300625Z" level=info msg="Starting up"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.547898087Z" level=info msg="Loading containers: start."
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.765201324Z" level=info msg="Loading containers: done."
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.780312822Z" level=warning msg="WARNING: bridge-nf-call-...sabled"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.781016672Z" level=warning msg="WARNING: bridge-nf-call-...sabled"
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.781231099Z" level=info msg="Docker daemon" commit=b08a5...=25.0.6
Nov 18 09:30:17 yashslave2 dockerd[13658]: time="2024-11-18T09:30:17.812751675Z" level=info msg="Daemon has completed initialization"
Nov 18 09:30:17 yashslave2 systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@yashslave2 ~]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
[ec2-user@yashslave2 ~]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[ec2-user@yashslave2 ~]$ sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)
) o /usr/local/bin/docker-compose
% Total % Received % Xferd Average Speed Time Time Current
          Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 0 0 0 0
100 11.2M 100 11.2M 0 0 44.8M 0 0 0 0 0 44.8M
[ec2-user@yashslave2 ~]$ sudo chmod +x /usr/local/bin/docker-compose
[ec2-user@yashslave2 ~]$ ls
remoting_remoting.jar workspace
[ec2-user@yashslave2 ~]$ 

```

03:02 PM IN 18-11-2024

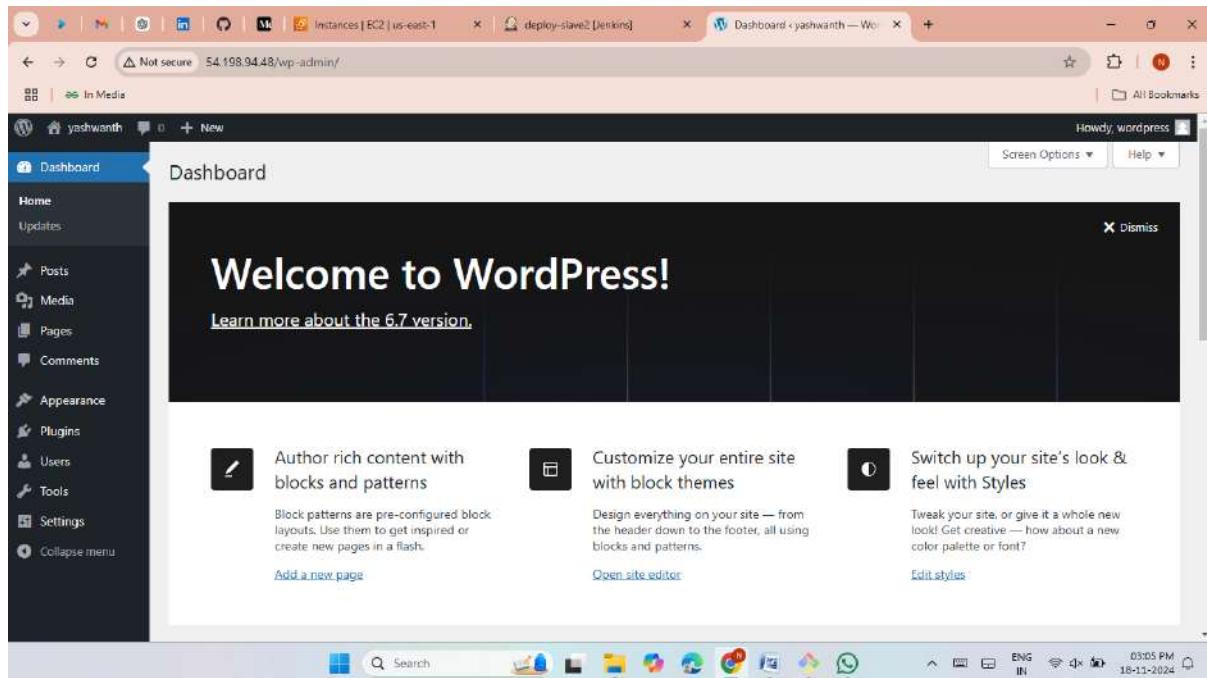
```

ec2-user@yashslave2:~/workspace/deploy-slave2
400f75d99009: Pull complete
c532b66579ca: Pull complete
9ac50b358b78: Pull complete
d8ecb69d311b: Pull complete
c047dfa11941: Pull complete
ef42461ff613: Pull complete
0c524fd1548c: Pull complete
50edb1570ef: Pull complete
cb08836f49b3: Pull complete
9c2382214998: Pull complete
2e3e5308e30: Pull complete
0f217c2e0f35: Pull complete
4f4fb700ef54: Pull complete
765e6e0a4dab: Pull complete
0e27'8e629e5: Pull complete
d58171113cc3: Pull complete
c767d278ce20: Pull complete
d1c531eb0a2b: Pull complete
cc989667550: Pull complete
fdf8706bad06: Pull complete
418893168d7: Pull complete
Digest: sha256:0f89dd041bddad11a01345850366403e0f268a1dabbca58beee786b3f3456348fd
Status: Downloaded newer image for wordpress:latest
Creating deploy-slave2_wordpress_1 ... done
Creating deploy-slave2_db_1 ... done
[ec2-user@yashslave2 deploy-slave2]$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
wordpress latest 3b0d909b6d7f 5 days ago 701MB
mysql 8.0.19 0c27e8e5fcfa 4 years ago 546MB
[ec2-user@yashslave2 deploy-slave2]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
938632d8ee46 mysql:8.0.19 "docker-entrypoint.s..." 15 seconds ago Up 13 seconds 3306/tcp, 33060/tcp
-1slave2_db_1 b72a2f4d39cc wordpress:latest "docker-entrypoint.s..." 15 seconds ago Up 13 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp deploy
-1slave2_wordpress_1 [ec2-user@yashslave2 deploy-slave2]$ 

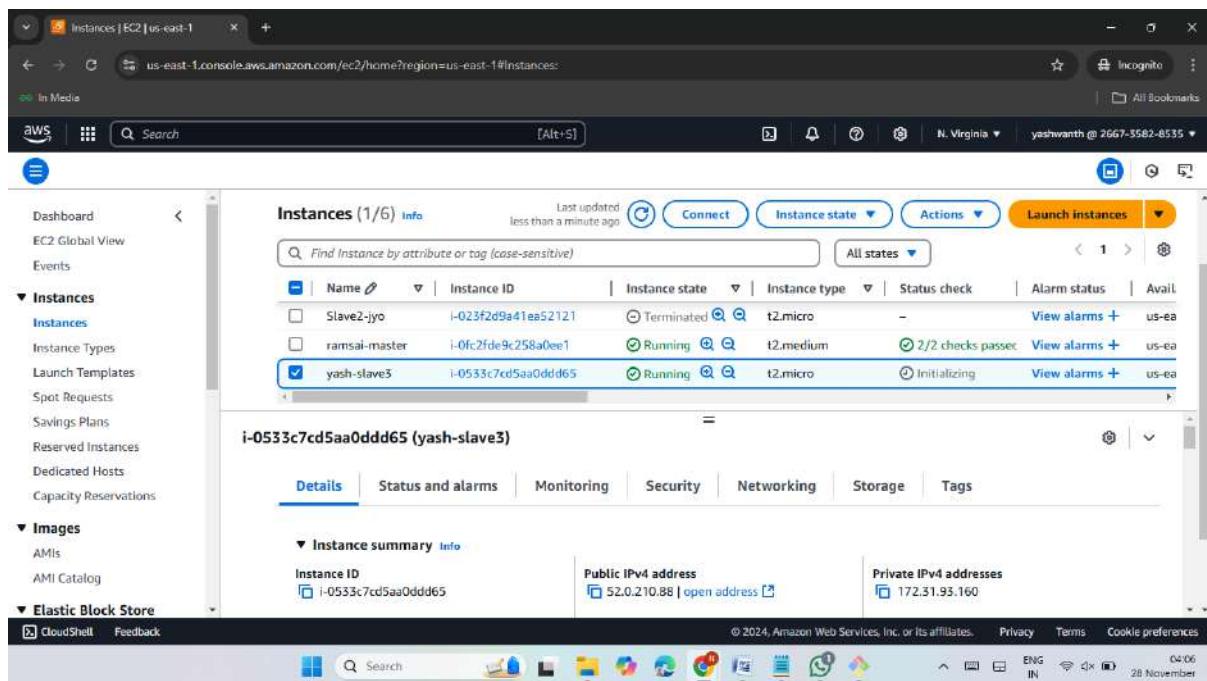
```

03:03 PM IN 18-11-2024

- Now copy the slave2 public ip and past it on google browser.
- The wordpress was hosted successfully.



- Here I have worked as a slave-3 in a team project.
- Launch an ec2 instance along with required ports.



- Now connect Slave3 then install java.
- Now give private keys from master server.

```

ec2-user@ip-172-31-93-160:~ Verifying : alsa-lib-1.4.1-2.amzn2.x86_64 29/32
ec2-user@ip-172-31-93-160:~ Verifying : fontpackages-filesystem-1.44-8.amzn2.noarch 30/32
ec2-user@ip-172-31-93-160:~ Verifying : libICE-1.0.9-9.amzn2.0.2.x86_64 31/32
ec2-user@ip-172-31-93-160:~ Verifying : javapackages-tools-3.4.1-11.amzn2.noarch 32/32

Installed:
  java-17-amazon-corretto.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-devel.x86_64 1:17.0.13+11-1.amzn2.1
  java-17-amazon-corretto-javadoc.x86_64 1:17.0.13+11-1.amzn2.1

Dependency Installed:
  alsa-lib.x86_64 0:1.4.1-2.amzn2
  dejavu-sans-fonts.noarch 0:2.33-6.amzn2
  dejavu-serif-fonts.noarch 0:2.33-6.amzn2
  fontpackages-filesystem.noarch 0:1.44-8.amzn2
  javapackages-tools.noarch 0:3.4.1-11.amzn2
  libSM.x86_64 0:1.2.2-2.amzn2.0.2
  libX11-common.noarch 0:1.6.7-3.amzn2.0.5
  libXext.x86_64 0:1.3.3-3.amzn2.0.2
  libXinerama.x86_64 0:1.3.2-2.amzn2.0.2
  libXrender.x86_64 0:0.9.10-1.amzn2.0.2
  libXtst.x86_64 0:1.2.3-1.amzn2.0.2
  libXslt.x86_64 0:1.1.28-6.amzn2
  python-javapackages.noarch 0:3.4.1-11.amzn2

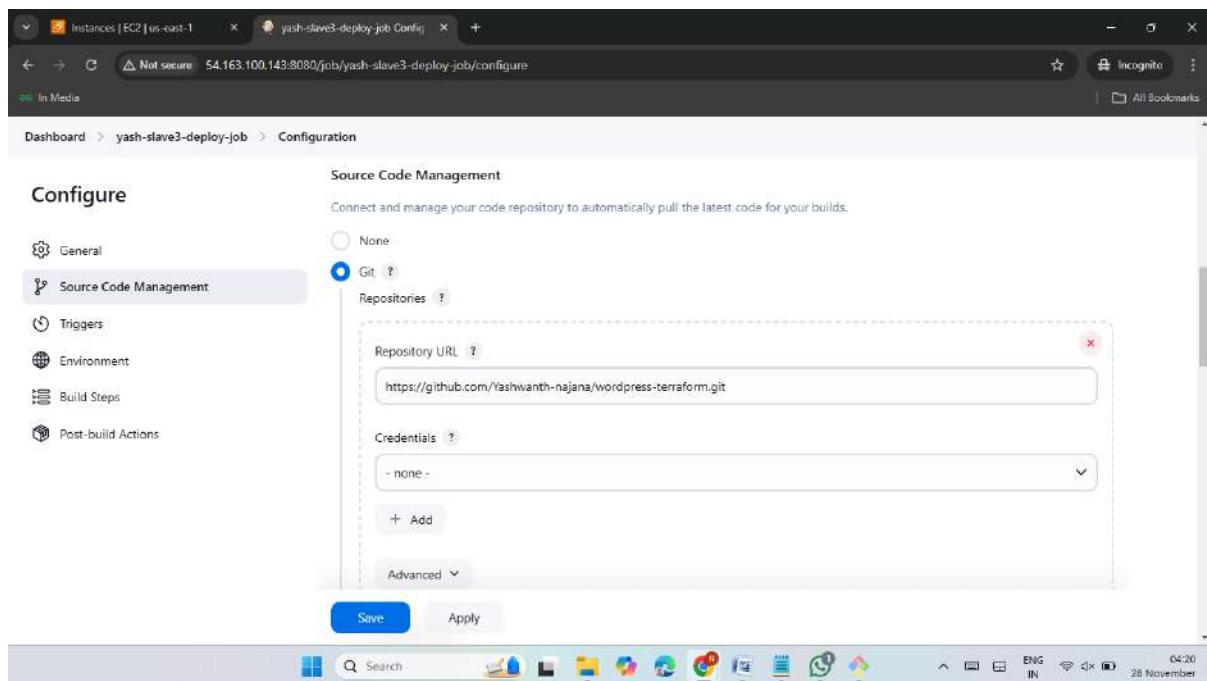
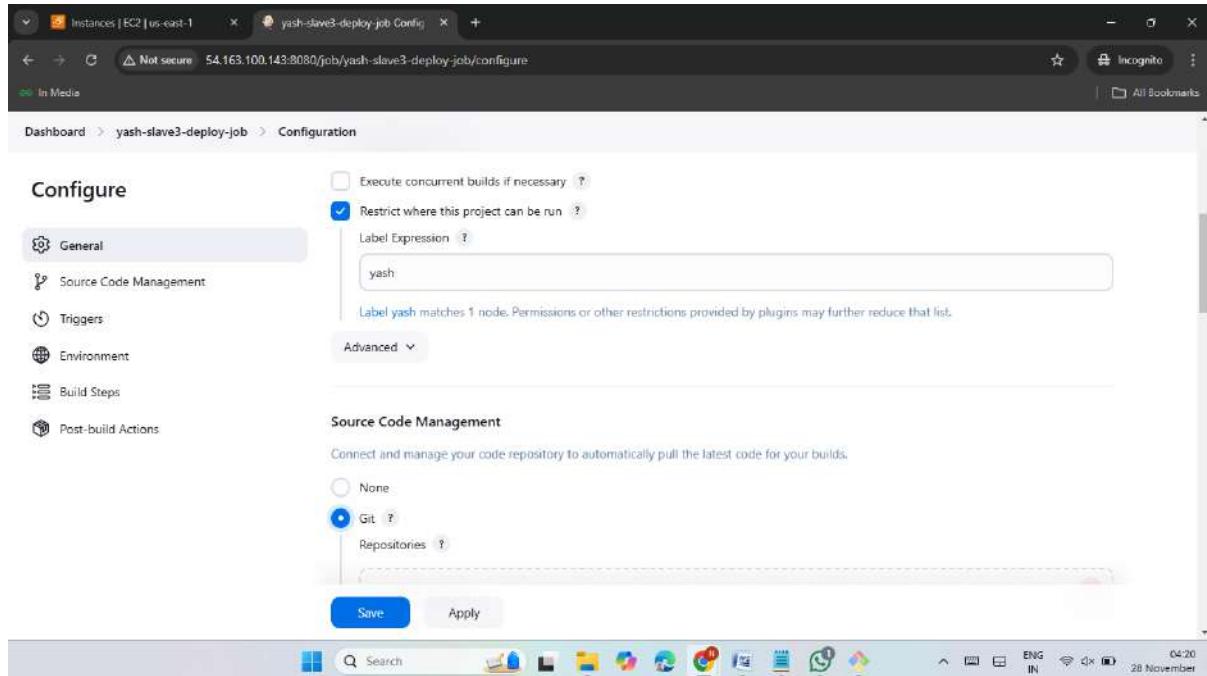
Complete!
[ec2-user@ip-172-31-93-160 ~]$ sudo hostname yashslave3
[ec2-user@ip-172-31-93-160 ~]$ exec bash
[ec2-user@yashslave3 ~]$ cd .ssh/
[ec2-user@yashslave3 .ssh]$ ls
authorized_keys
[ec2-user@yashslave3 .ssh]$ sudo vi authorized_keys
[ec2-user@yashslave3 .ssh]$ cd
[ec2-user@yashslave3 ~]$ ped
bash: ped: command not found
[ec2-user@yashslave3 ~]$ pwd
/home/ec2-user
[ec2-user@yashslave3 ~]$

```

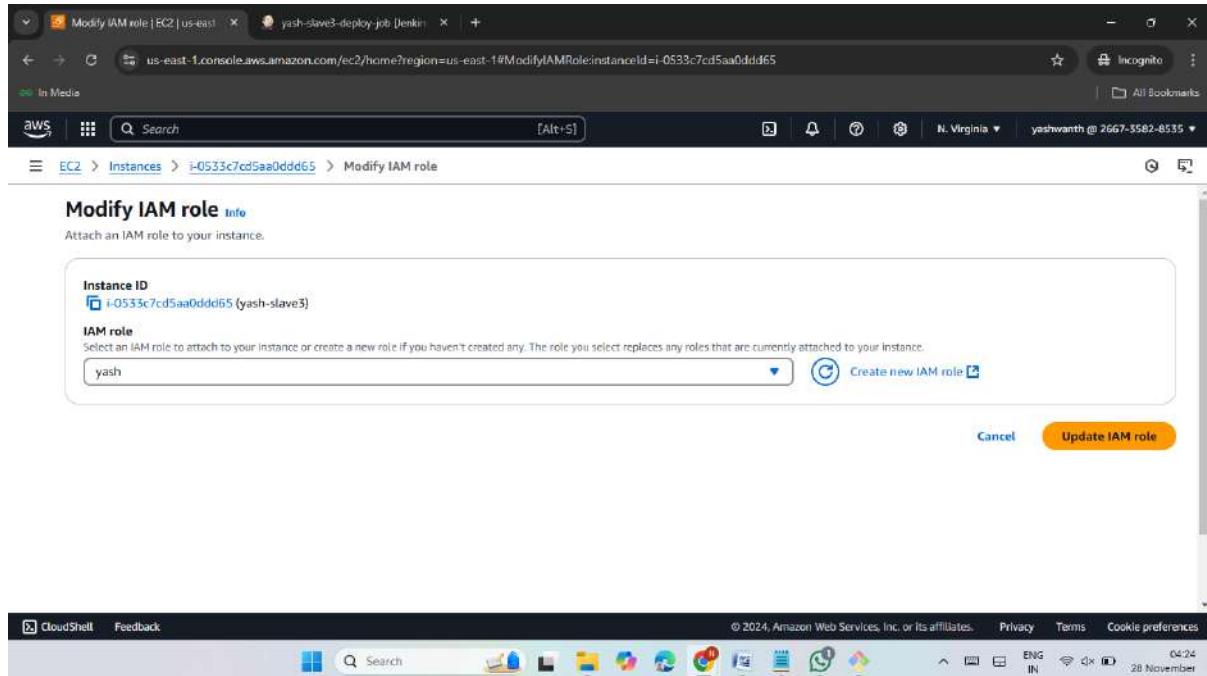
- Now create a node for slave3 server.
- Give details for slave node.
- Here we have to give slave3 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here the slave3 node is successfully created.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	4.97 GiB	0 B	4.97 GiB	0ms
	slave-4	Linux (amd64)	In sync	4.66 GiB	0 B	4.66 GiB	34ms
	Slave2-jyo	Linux (amd64)	In sync	3.57 GiB	0 B	3.57 GiB	32ms
	yash-slave-3	Linux (amd64)	In sync	5.36 GiB	0 B	5.36 GiB	36ms
	Data obtained	0.26 sec	0.26 sec	0.26 sec	0.26 sec	0.25 sec	0.26 sec

- Now create a job and give slave3 label.
- Now clone the wordpress application repo form github.
- Now Click on save.



- Create IAM role with full access then attach to the slave-3 server for give permissions to the terraform resources.



- Now run the following commands to create instances, vpc, subnets, igw, route, security groups.

```
#!/bin/bash
```

```
sudo yum install -y yum-utils shadow-utils
```

```
sudo yum-config-manager --add-repo
```

```
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
```

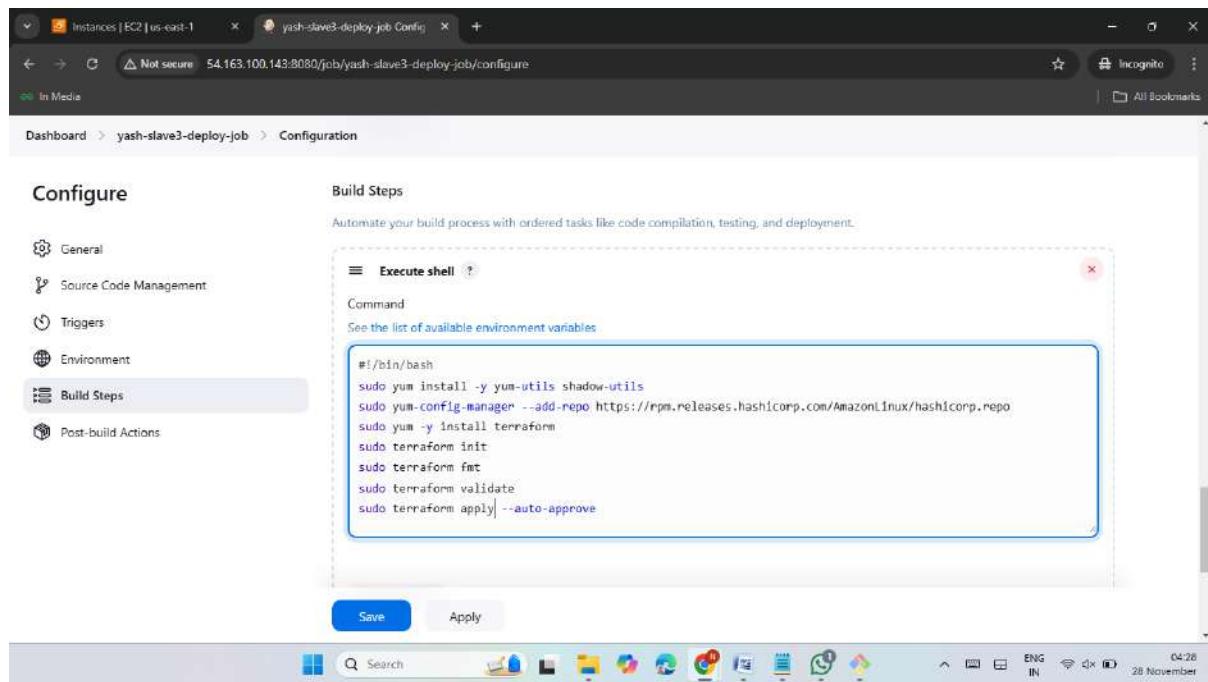
```
sudo yum -y install terraform
```

```
sudo terraform init
```

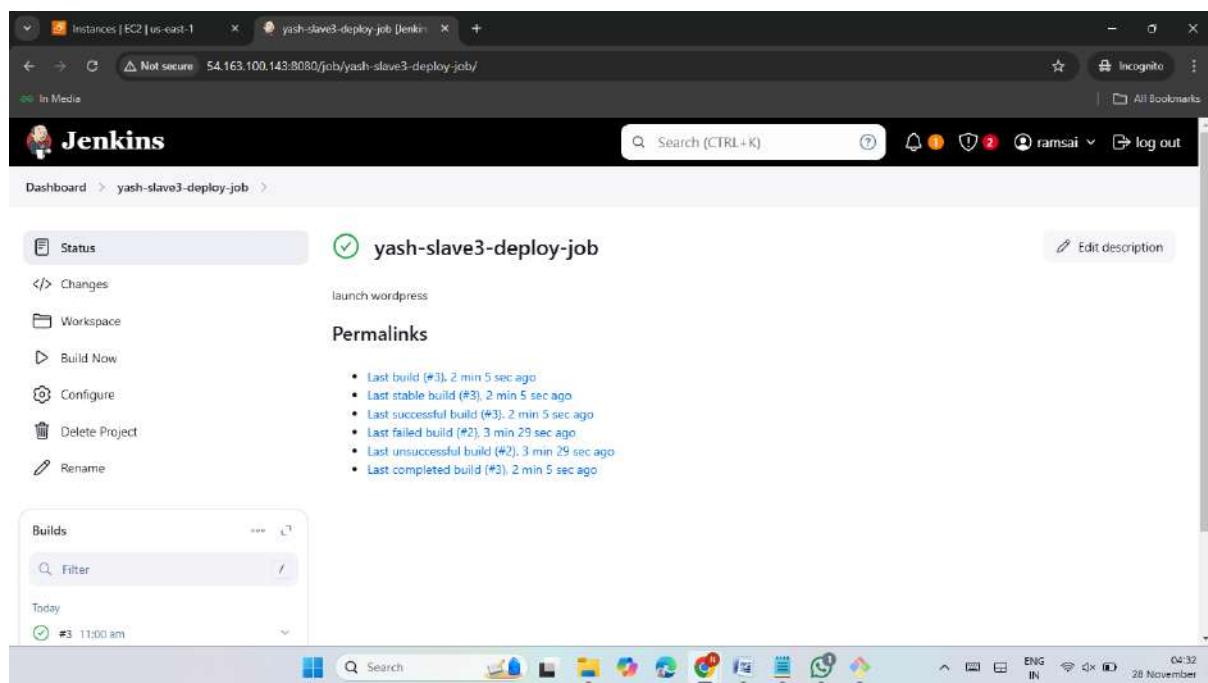
```
sudo terraform fmt
```

```
sudo terraform validate
```

```
sudo terraform apply --auto-approve
```



- Now click on build now.
- Here the job was success.



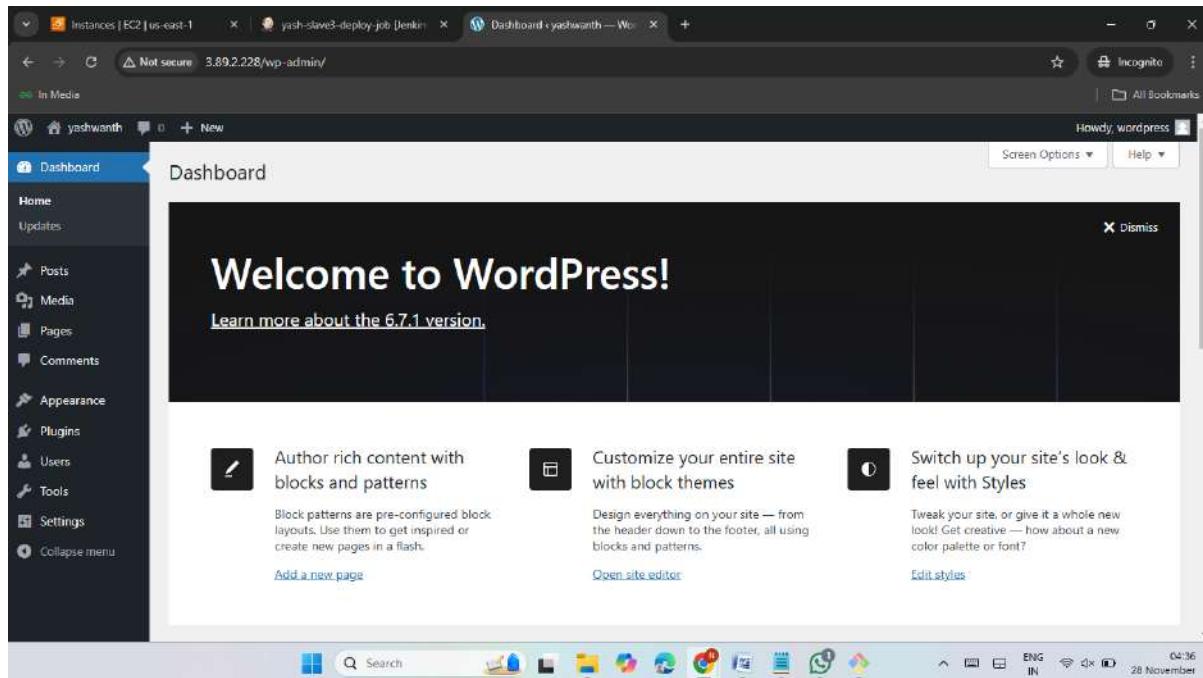
➤ Here the instance was launched.

The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like Dashboard, EC2 Global View, Events, Instances (with sub-options for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and Elastic Block Store (CloudShell, Feedback). The main content area displays a table titled "Instances (1/10) Info" with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability. One instance, "wordpress" (Instance ID: i-07b828b39de995a3b), is listed as "Running". Below the table, a detailed view for "i-07b828b39de995a3b (wordpress)" is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The "Details" tab is selected, showing the Instance ID (i-07b828b39de995a3b), Public IPv4 address (3.89.2.228), and Private IPv4 addresses (10.0.1.89).

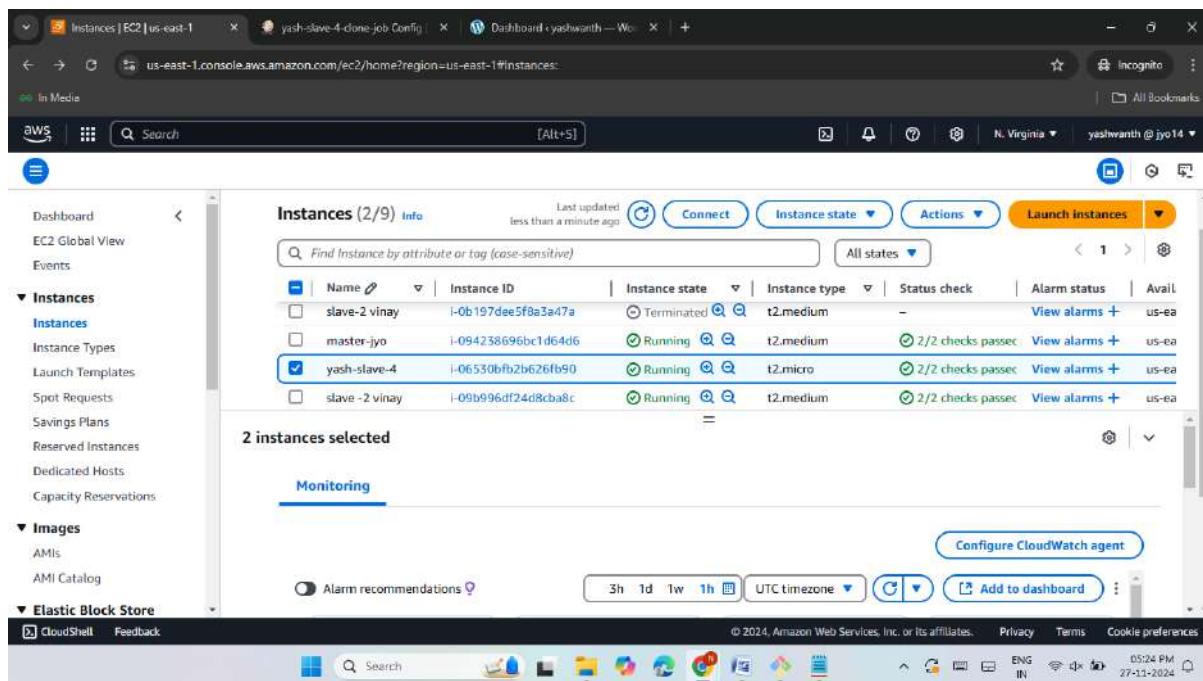
➤ Here the VPC was launched.

The screenshot shows the AWS VPC dashboard. The left sidebar includes options like EC2 Global View (Filter by VPC), Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only Internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints), and CloudShell, Feedback. The main content area displays a table titled "Your VPCs (1/3) Info" with columns for Name, VPC ID, State, Block Public Access, and IPv4 CIDR. A VPC named "yashvpc" (VPC ID: vpc-08ceec730d2f7b8dc) is listed as "Available" with Block Public Access set to "Off" and IPv4 CIDR 10.0.0.0/16. Below the table, a detailed view for "vpc-08ceec730d2f7b8dc / yashvpc" is shown with tabs for Details, Resource map, CIDs, Flow logs, Tags, and Integrations. The "Details" tab is selected, showing the VPC ID (vpc-08ceec730d2f7b8dc), State (Available), Block Public Access (Off), and DNS hostnames (Disabled).

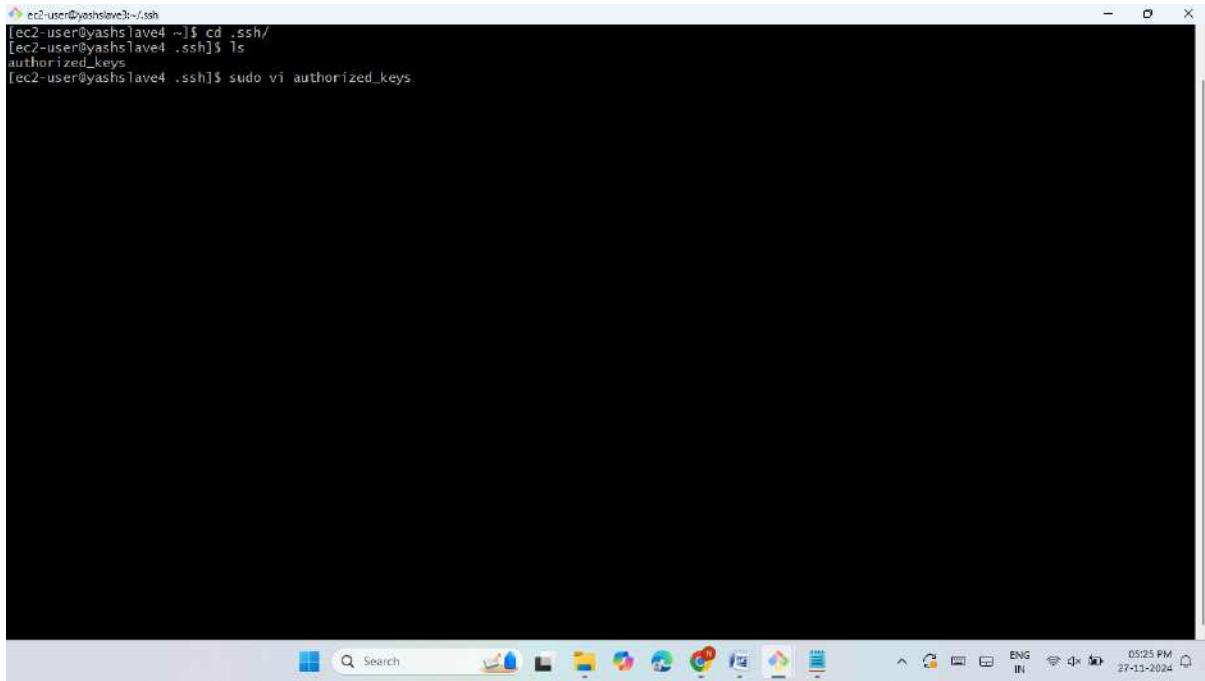
- Now copy the created instance public ip and past it on Google browser.
- The wordpress was hosted successfully.



- Here I have worked as a slave-4 in a team project.
- Launch an ec2 instance along with required ports.



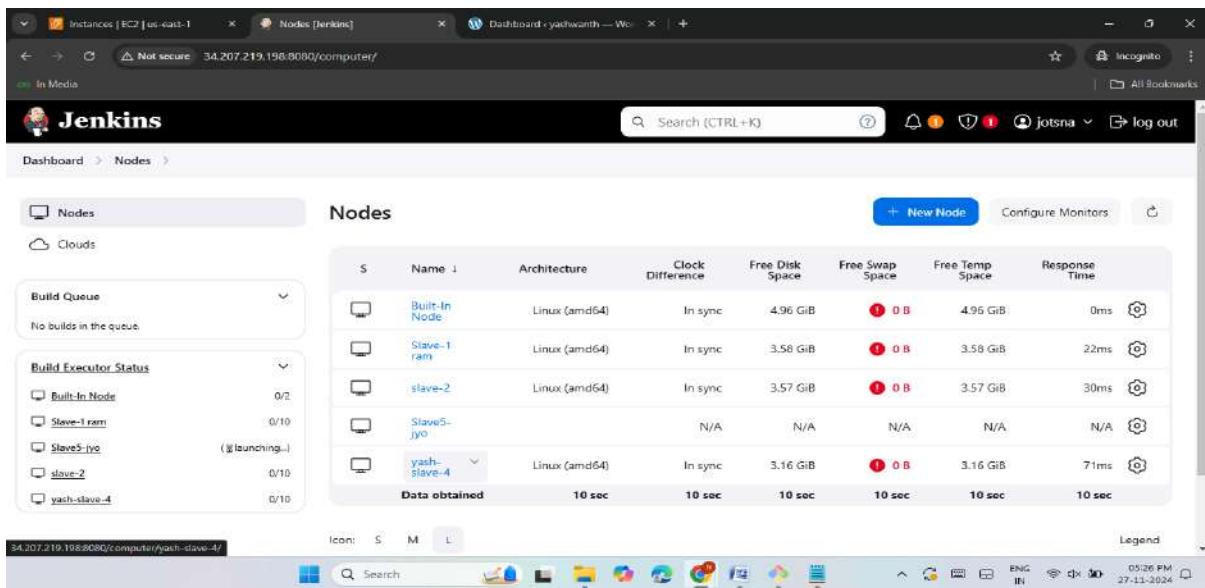
- Now connect Slave4 then install java.
- Now give private keys from master server.



```
ec2-user@yashslave4:~/ssh
[ec2-user@yashslave4 ~]$ cd .ssh/
[ec2-user@yashslave4 .ssh]$ ls
authorized_keys
[ec2-user@yashslave4 .ssh]$ sudo vi authorized_keys
```

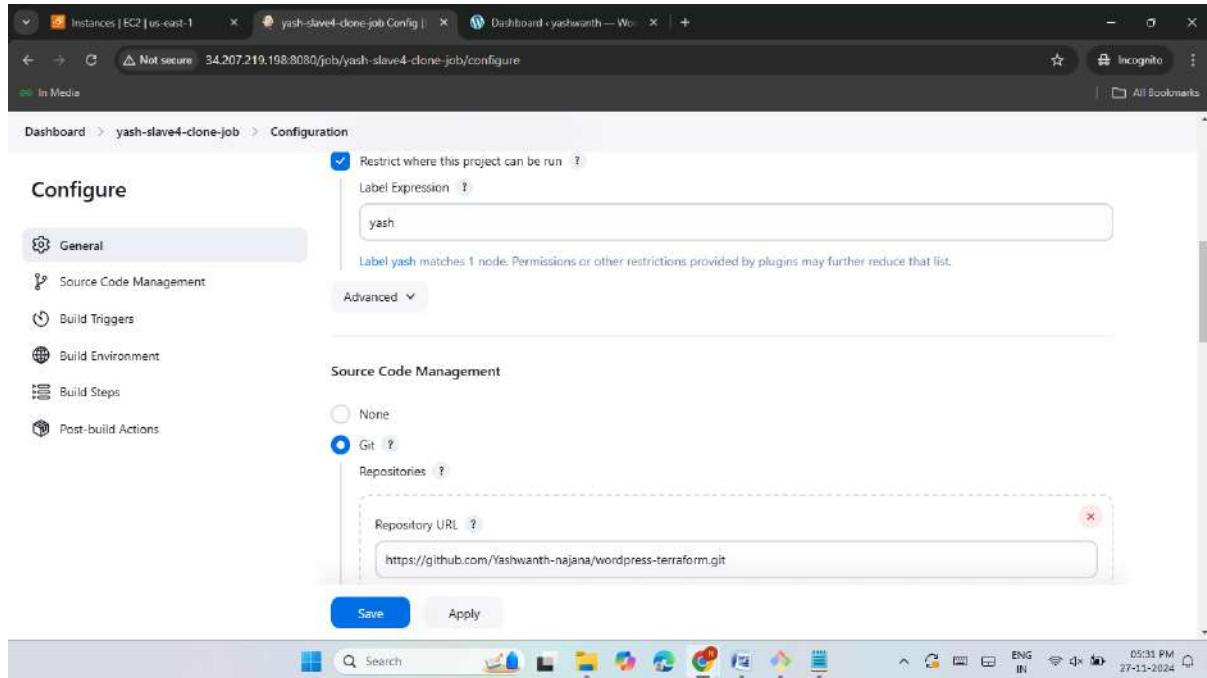
The screenshot shows a Windows desktop environment with a terminal window open. The terminal window title is 'ec2-user@yashslave4:~/ssh'. Inside, the user has run several commands: 'cd .ssh/' to change directory, 'ls' to list files, and 'sudo vi authorized_keys' to edit the file. The terminal window is part of a larger desktop interface with a taskbar at the bottom showing various application icons.

- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here the slave4 node is successfully created.

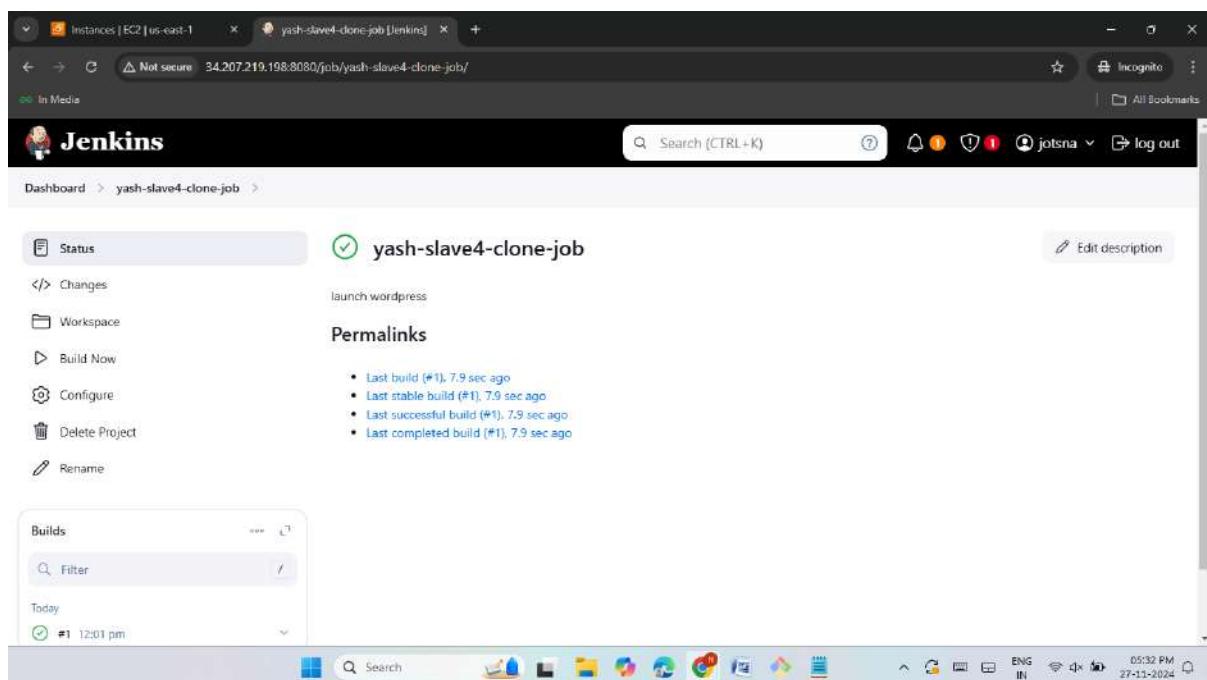


The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with tabs for 'Instances | EC2 | us-east-1', 'Nodes [Jenkins]', and 'Dashboard - yashwaran - Wo...'. Below the navigation, there's a search bar and a 'log out' link. The main area is titled 'Nodes' and contains a table of nodes. The table columns are: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, Response Time, and a 'Configure Monitors' button. The table lists several nodes, including 'Built-In Node', 'Slave-1 ram', 'slave-2', 'Slave5-jyo', and 'yash-slave-4'. The 'yash-slave-4' row shows 'Data obtained' and '10 sec' for all metrics. On the left side, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (listing 'Built-In Node', 'Slave-1 ram', 'Slave5-jyo', 'slave-2', and 'yash-slave-4'). The bottom of the screen shows a Windows taskbar with various icons.

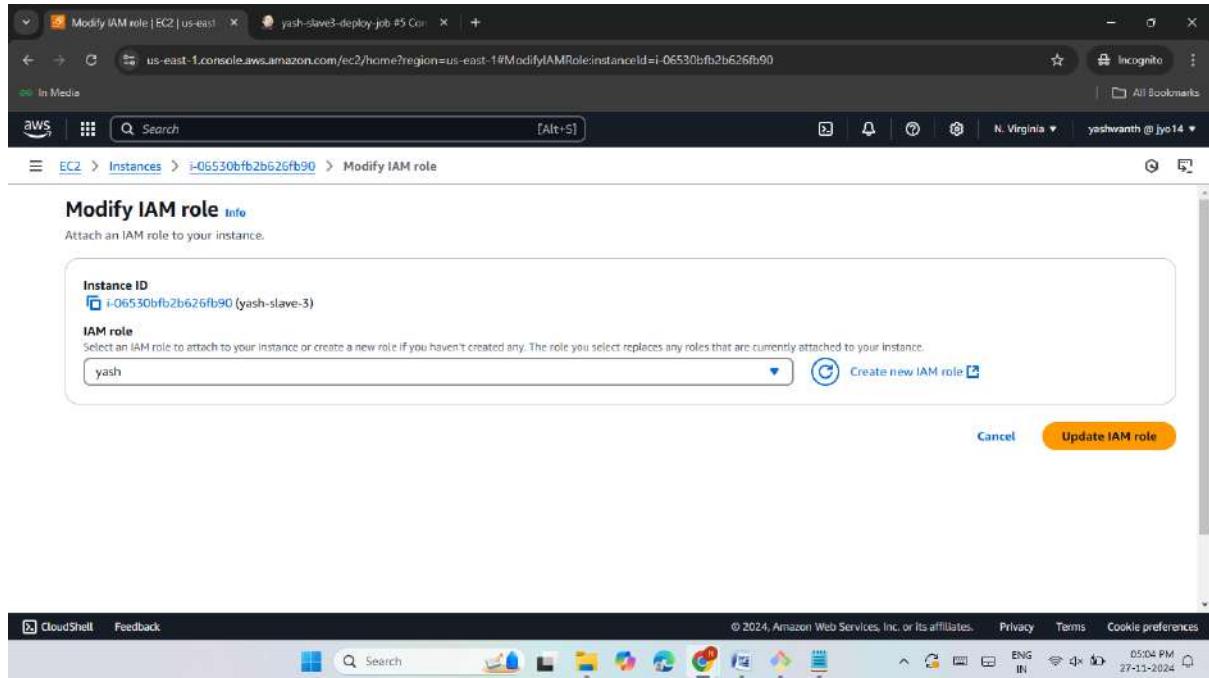
- Now create a job and give slave4 label.
- Now clone the wordpress application repo from github.
- Now Click on save.



- Now click on build now.
- Here the job was success.



- Create IAM role with full access then attach to the slave-4 server for give permissions to the terraform resources.



- Create a another job.
- Now run the following commands to create instances, vpc, subnets, igw, route, security groups.

```
#!/bin/bash
```

```
sudo yum install -y yum-utils shadow-utils
```

```
sudo yum-config-manager --add-repo
```

```
https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
```

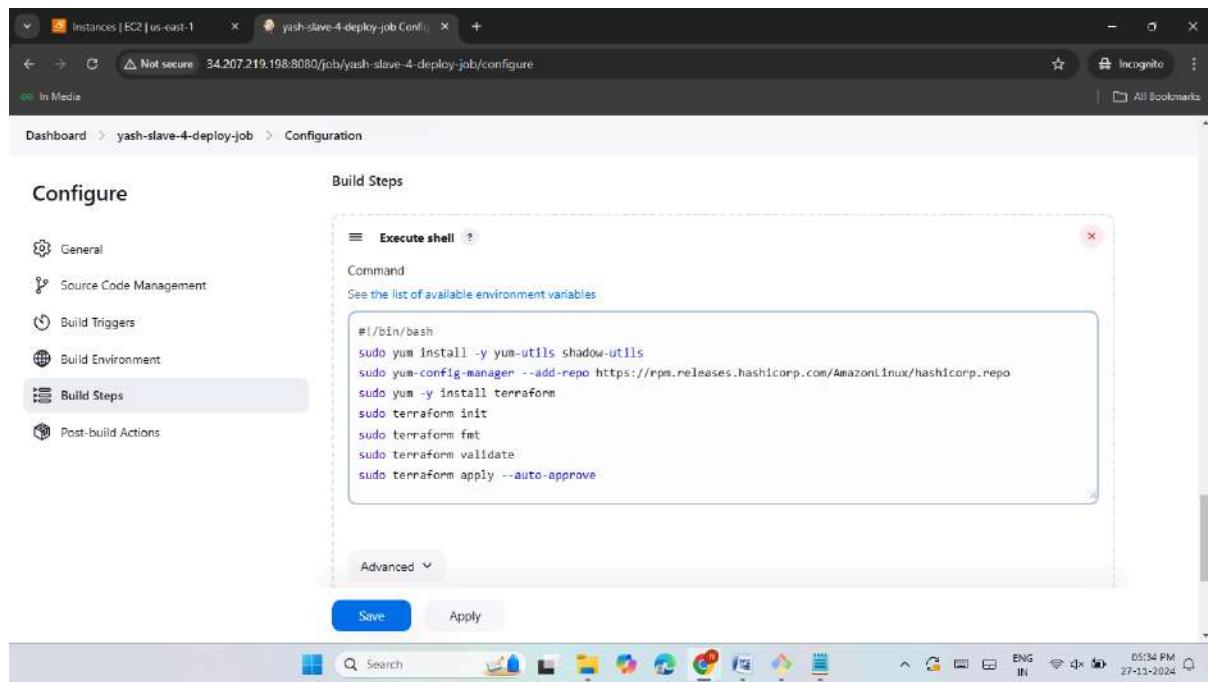
```
sudo yum -y install terraform
```

```
sudo terraform init
```

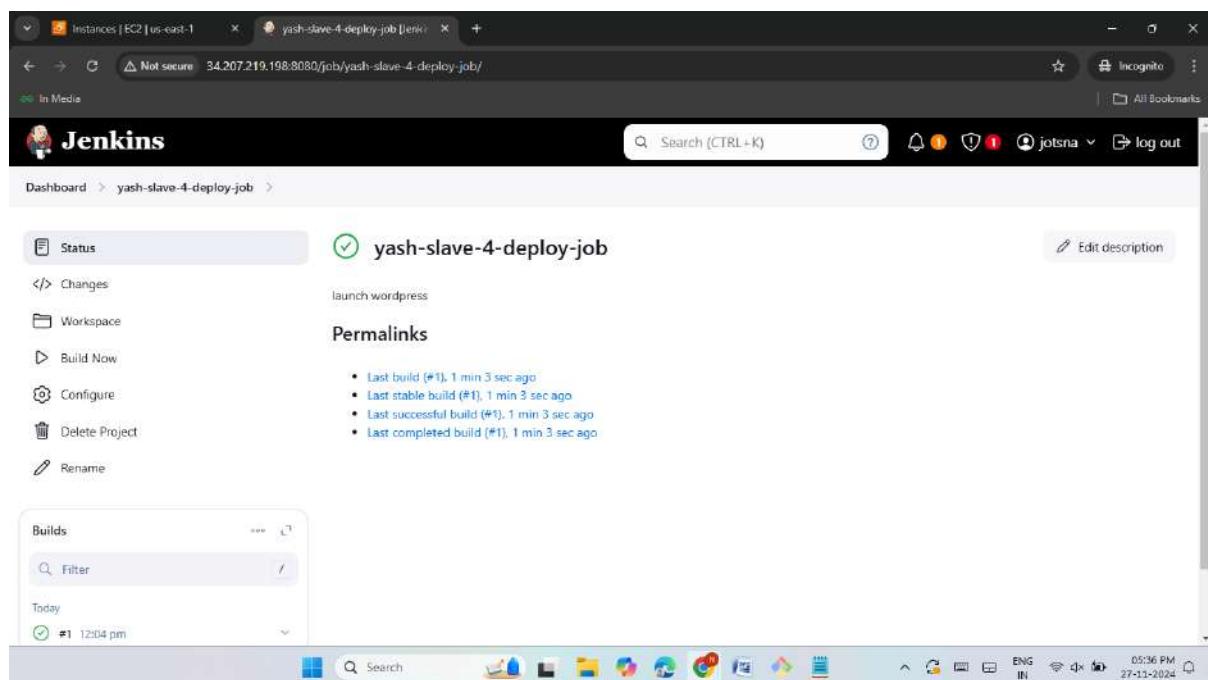
```
sudo terraform fmt
```

```
sudo terraform validate
```

```
sudo terraform apply --auto-approve
```



- Now click on build now.
- Here the job was success.



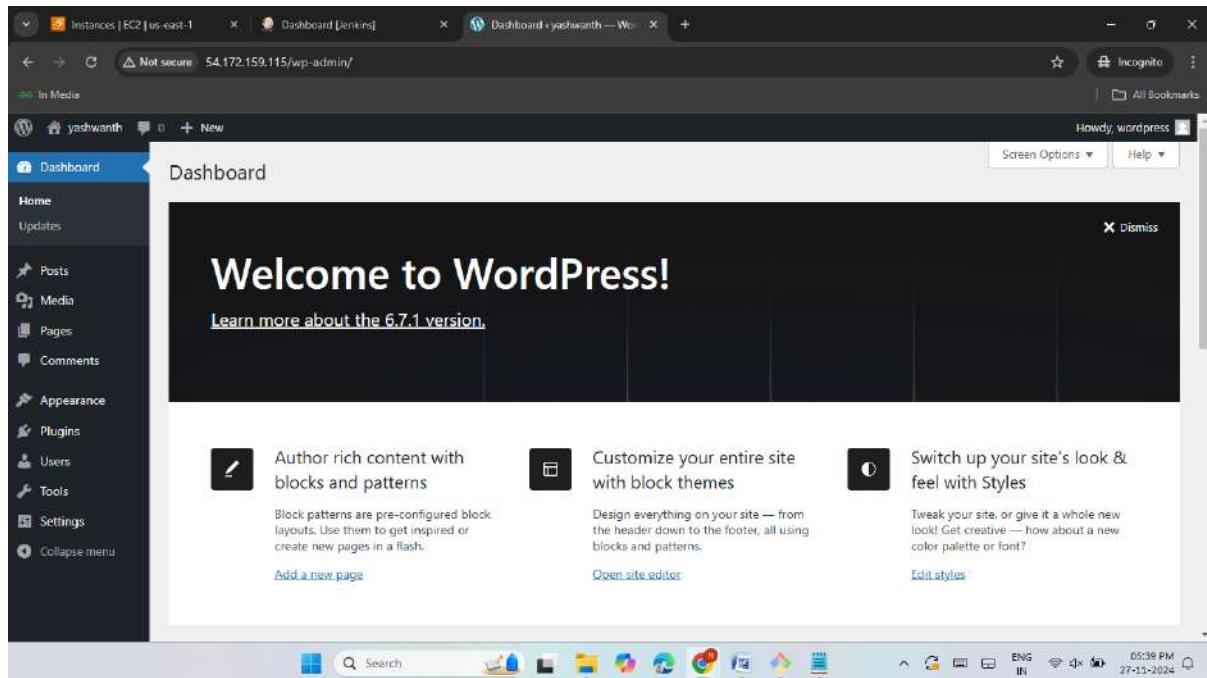
- Here the instance was launched.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store. The main content area is titled "Instances (2/13)" and shows a table of instances. Two instances are selected: "slave - 2 vinay" (terminated, t2.medium) and "Slave5-jyo" (terminated, t2.micro). A third instance, "Slave-1 raim" (running, t2.medium), has a checkbox next to it. A fourth instance, "wordpress" (running, t2.micro), has a checked checkbox. Below the table, it says "2 instances selected". There are tabs for Monitoring and CloudWatch Metrics. At the bottom, there are buttons for Configure CloudWatch agent, Alarm recommendations, and Add to dashboard. The status bar at the bottom right shows the date and time as 27-11-2024 05:36 PM.

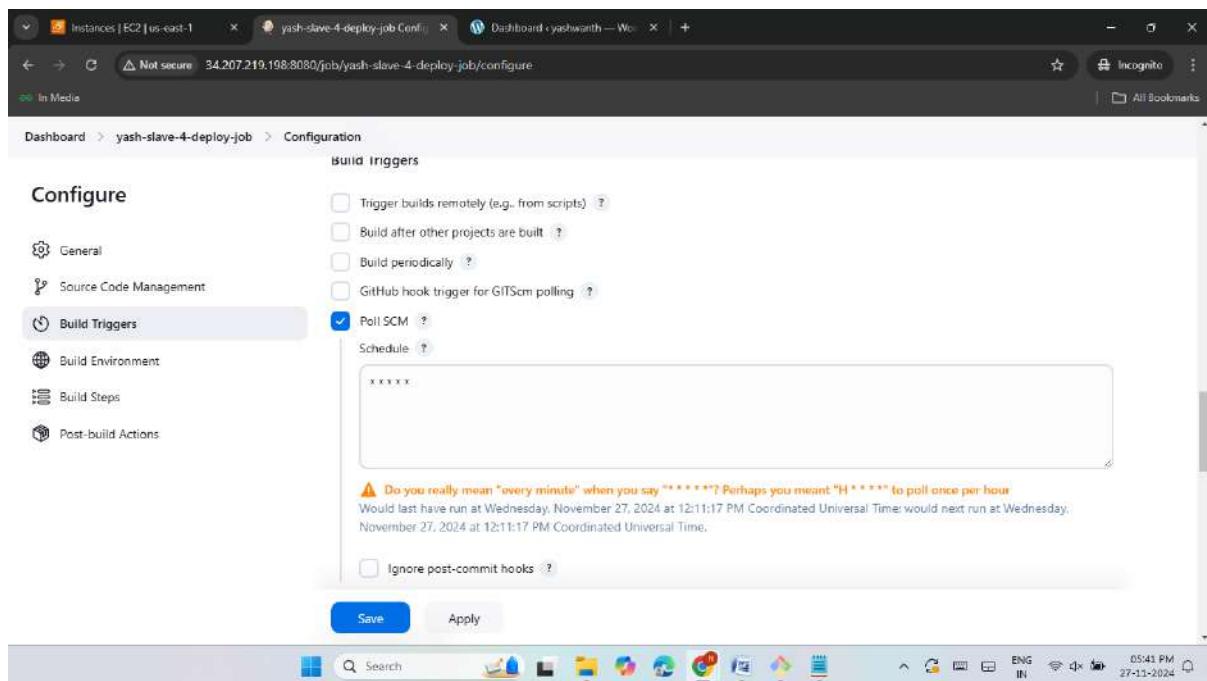
- Here the VPC was launched.

The screenshot shows the AWS VPC Console. On the left, there's a navigation sidebar with EC2 Global View (selected), Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints), and CloudShell/Feedback. The main content area is titled "Your VPCs (5)" and shows a table of VPCs. Five VPCs are listed: "yashvpc" (available, 10.0.0.0/16), "yashvpc" (available, 10.0.0.0/16), "ramsay-k8.local" (available, 172.20.0.0/16), "-" (available, 172.31.0.0/16), and "yashvpc" (available, 10.0.0.0/16). Below the table, it says "Select a VPC above". The status bar at the bottom right shows the date and time as 27-11-2024 05:37 PM.

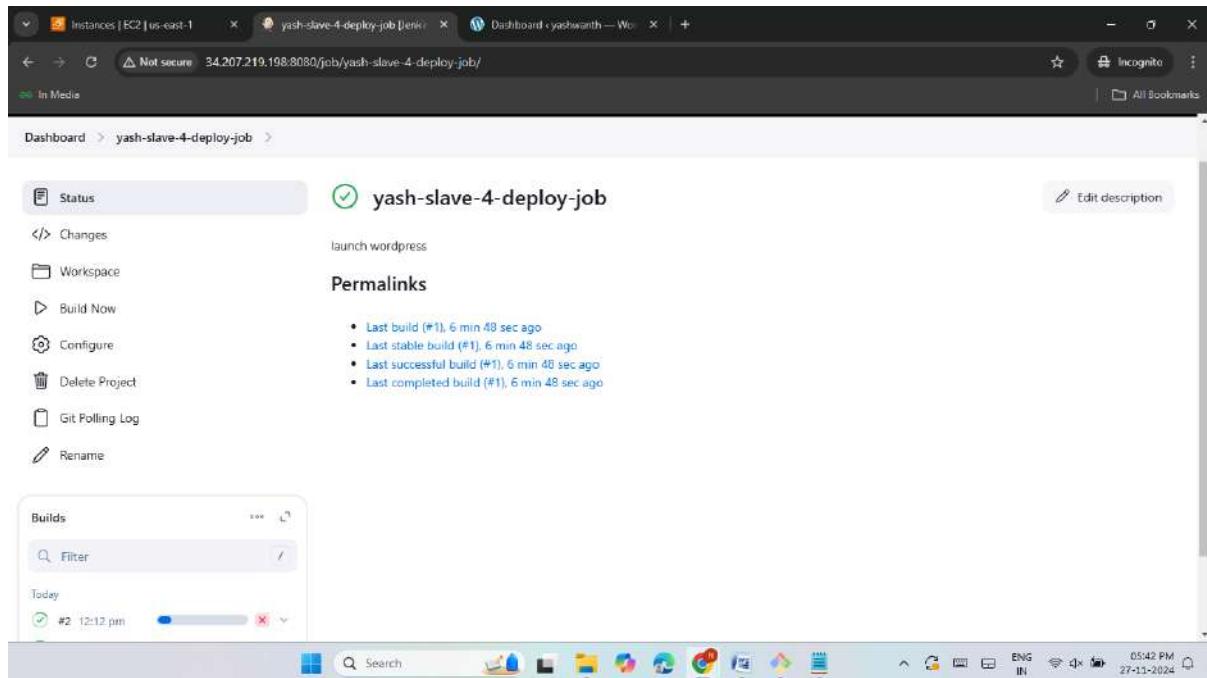
- Now copy the created instance public ip and past it on google browser.
- The wordpress was hosted successfully.



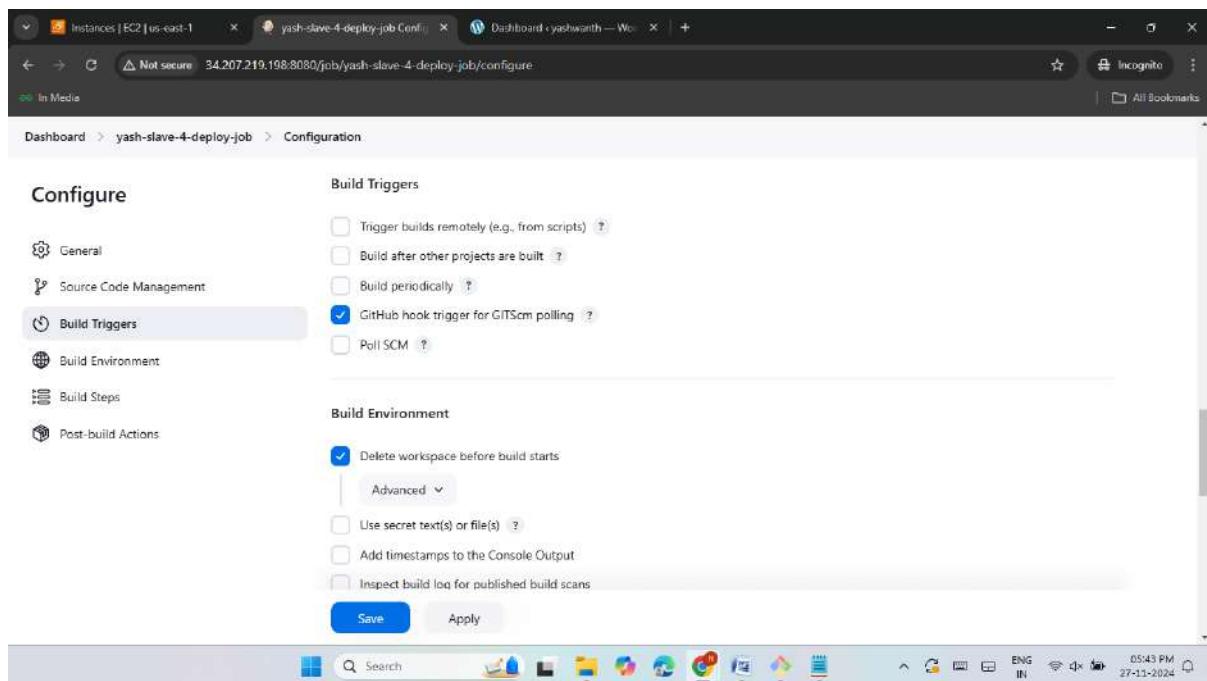
- Now add Poll SCM.



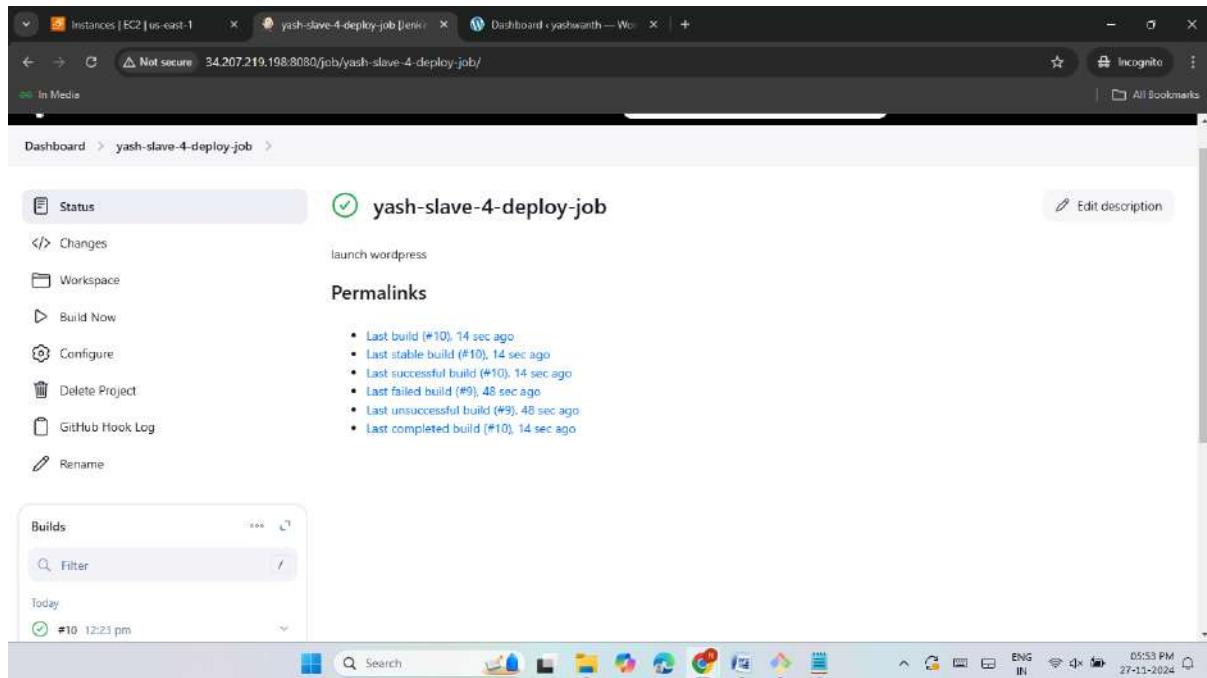
- When we change the code in github. The jobs was automatically runs after one min because of Poll SCM.



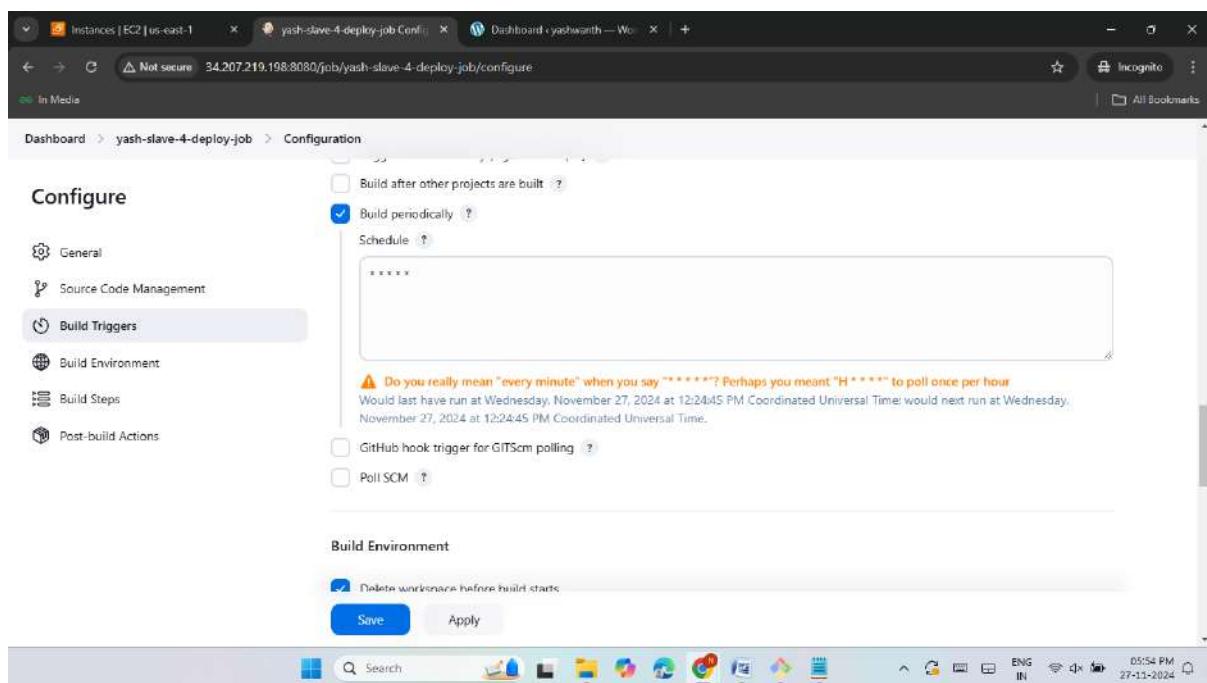
- Create webhooks in github then add into the Jenkins job.



- When we change the code in github. The jobs was automatically runs within seconds because of webhooks.



- Now add build periodically in Jenkins job.



- The jobs was automatically runs every one min because of build periodically.

The screenshot shows a Jenkins job named "yash-slave-4-deploy-job". The job status is "Green" with a checkmark icon. The job description is "launch wordpress". On the left, there's a sidebar with options like "Changes", "Workspace", "Build Now", "Configure", "Delete Project", and "Rename". Below the sidebar is a "Builds" section with a table showing the last 11 builds. The most recent build (#11) is shown with a green checkmark and the timestamp "12:25 pm". The Jenkins interface includes a search bar, a "Permalinks" section with a bulleted list of build links, and a "Edit description" button. At the top, the browser title is "yash-slave-4-deploy-job [jenk]" and the address bar shows "Not secure 34.207.219.198:8080/job/yash-slave-4-deploy-job/". The bottom of the screen shows a Windows taskbar with various icons and the system tray indicating the date and time as "27-11-2024 05:55 PM".