PROJECT

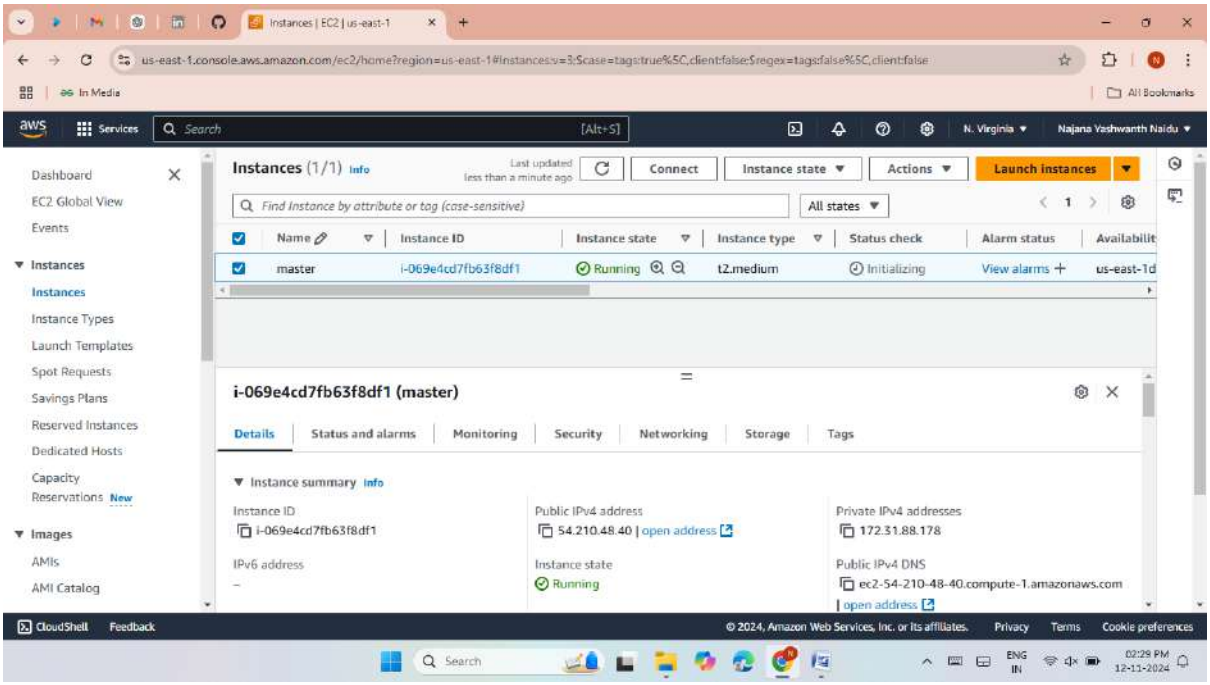
N Yashwanth Naidu

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).

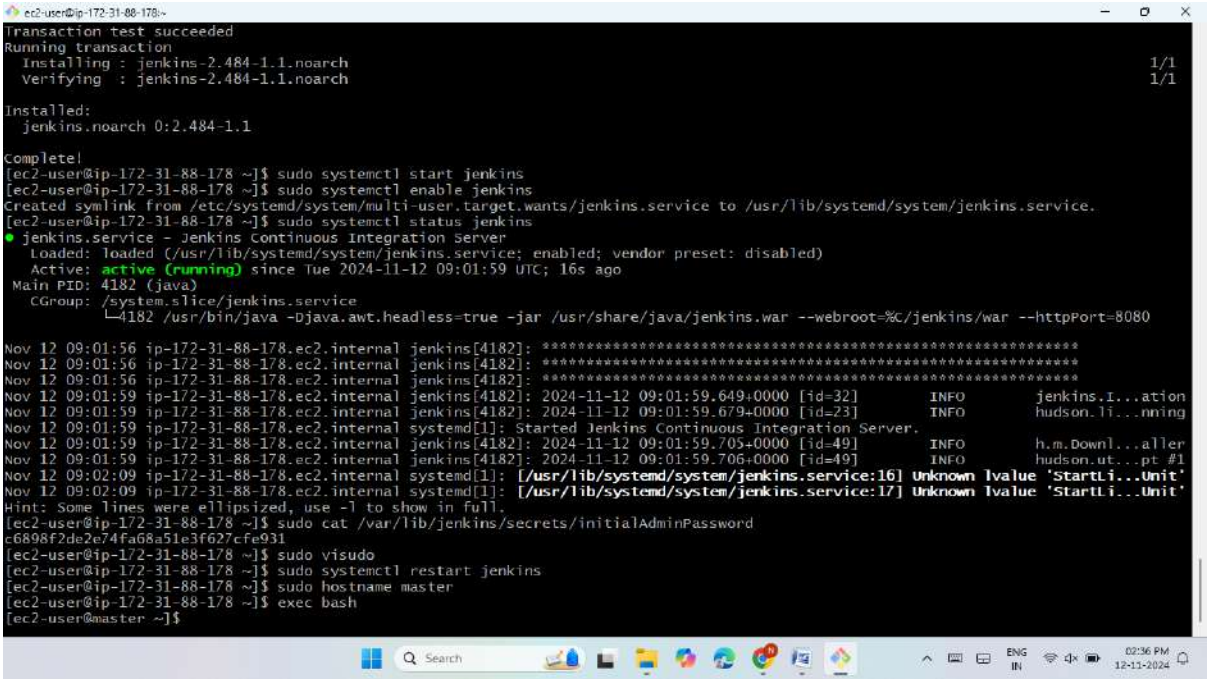
- First launch an master server with the Jenkins port 8080.



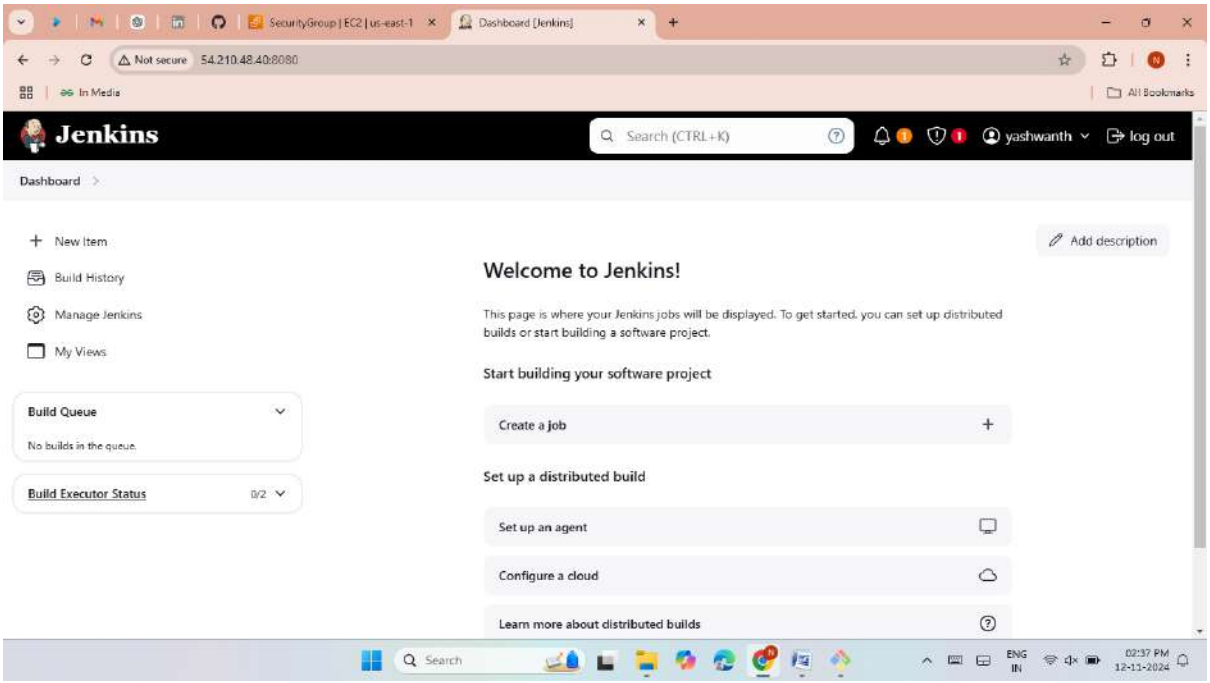
- Now install java and git.
- Now install Jenkins then start and enable the Jenkins.
- We can change the hostname with the commands

  Sudo hostname master

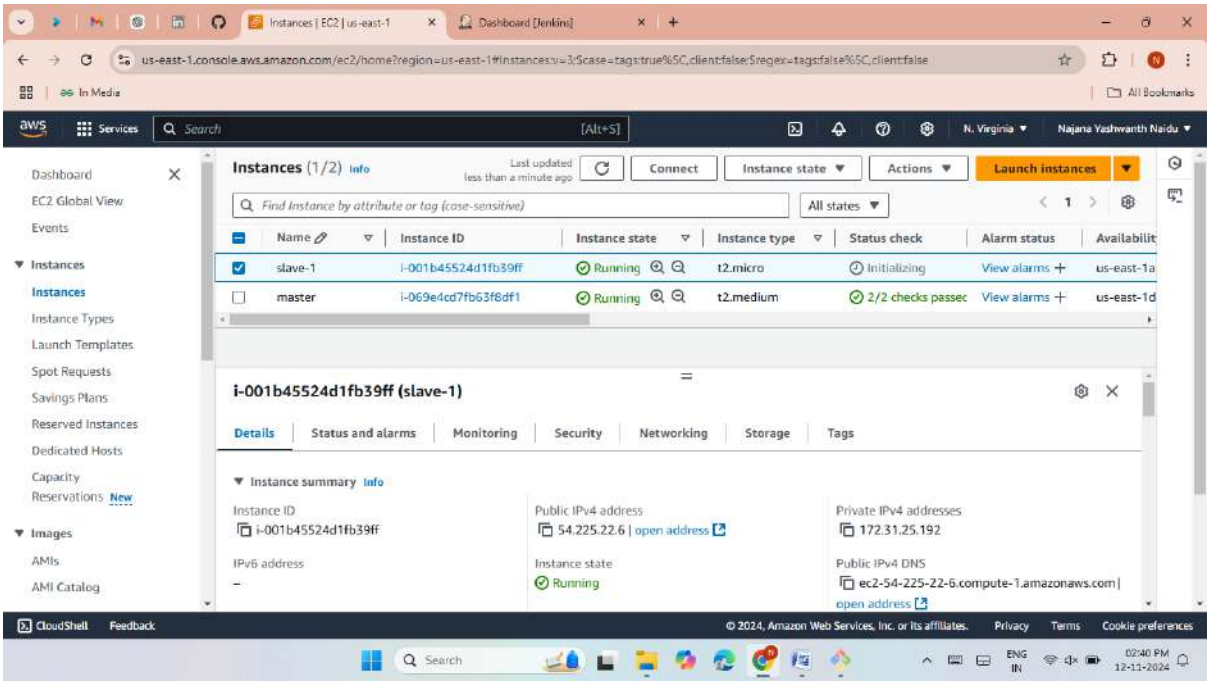
  Exec bash



- Here the Jenkins was launched successfully.

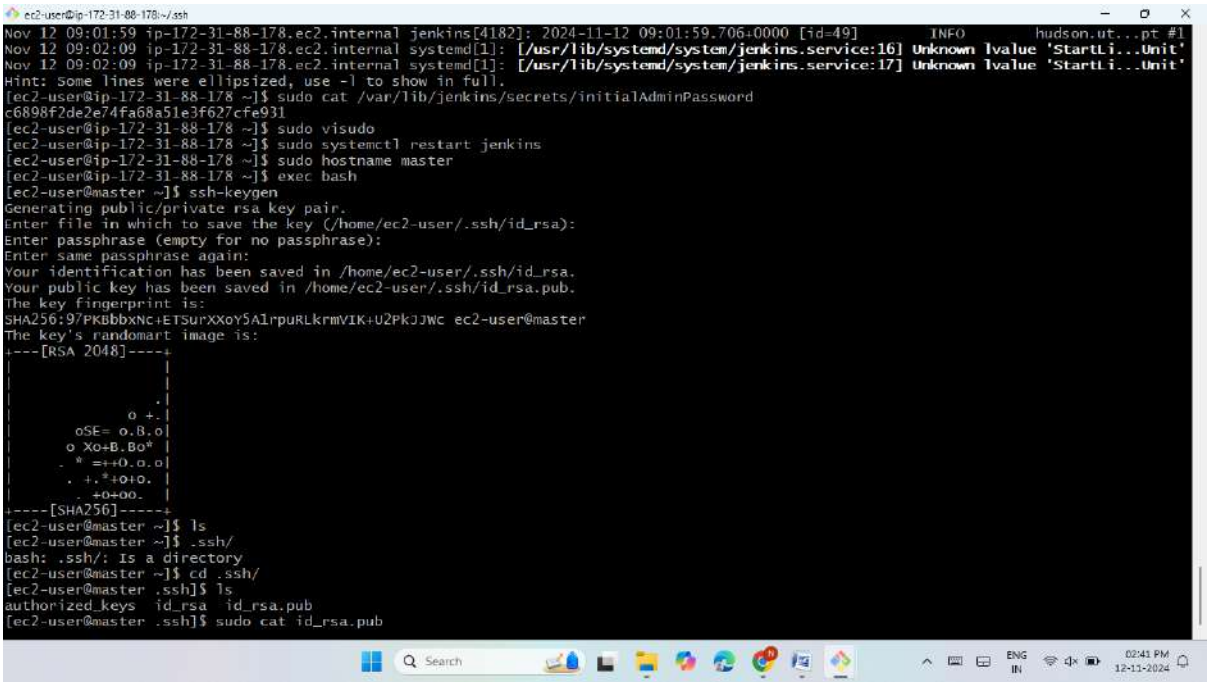


- Now launch an EC2 Instances and give name as slave1.

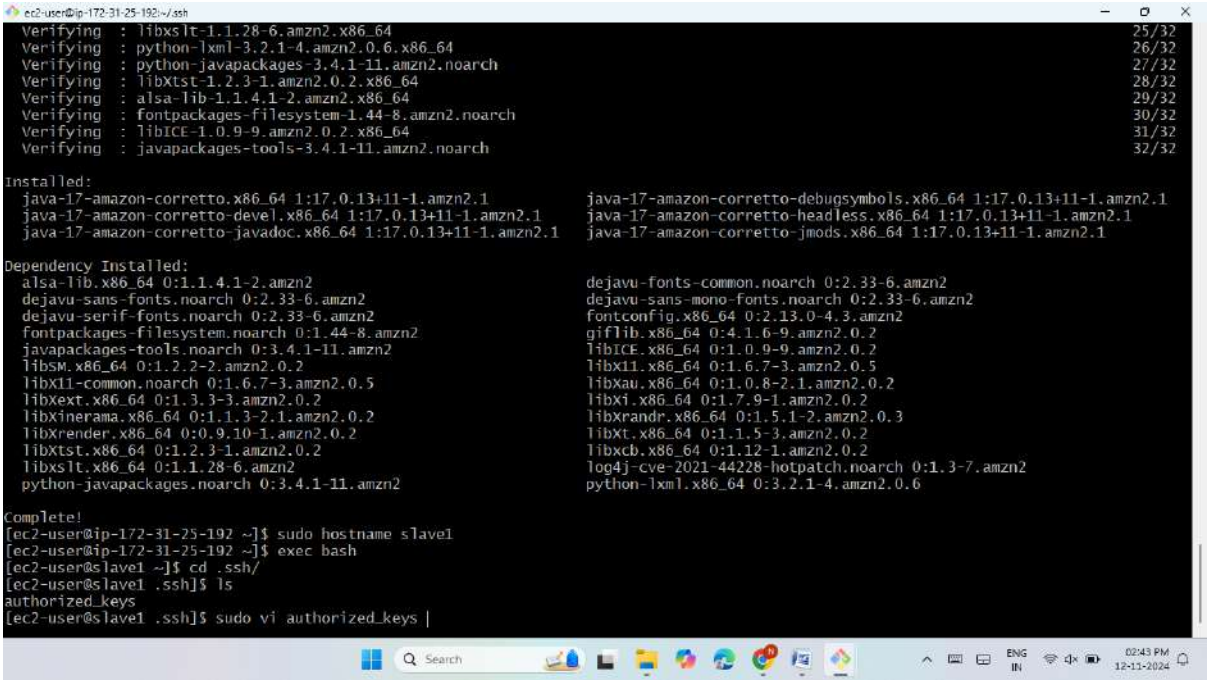


- Now connect to the Master server and then generate keys with the command.

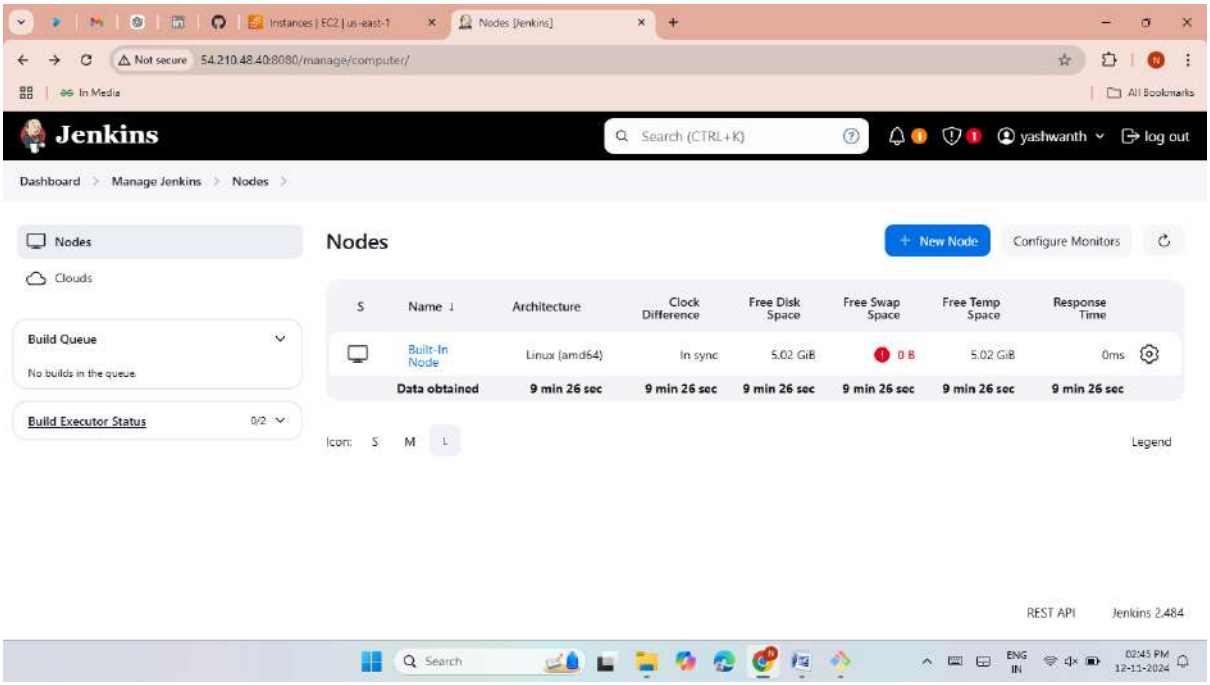
ssh-keygen



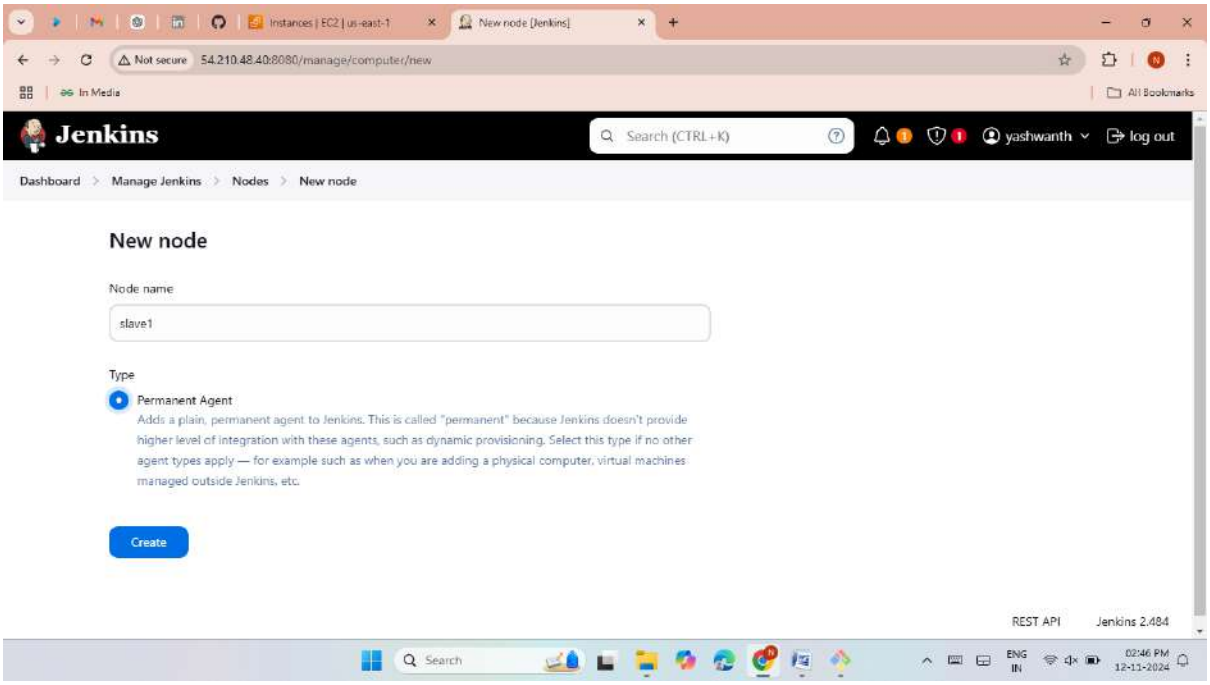
- Now connect Slave1 then install java.
- Now give private keys from master server.



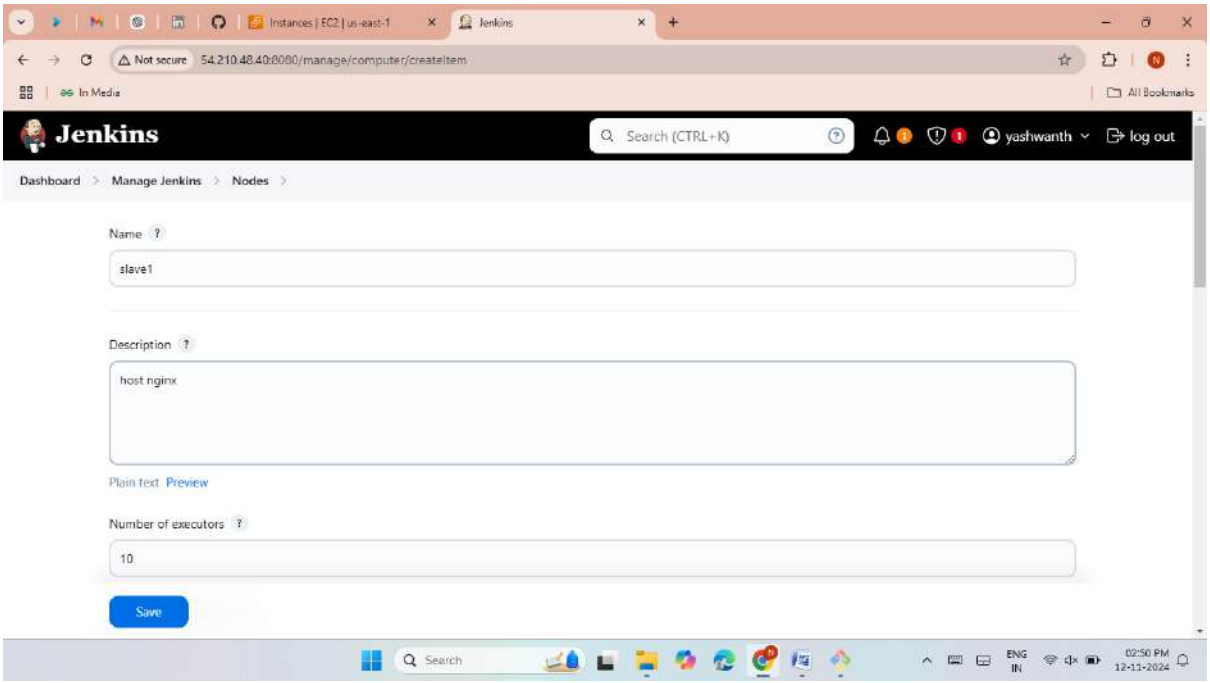
- Here the default nodes are there in the Jenkins.

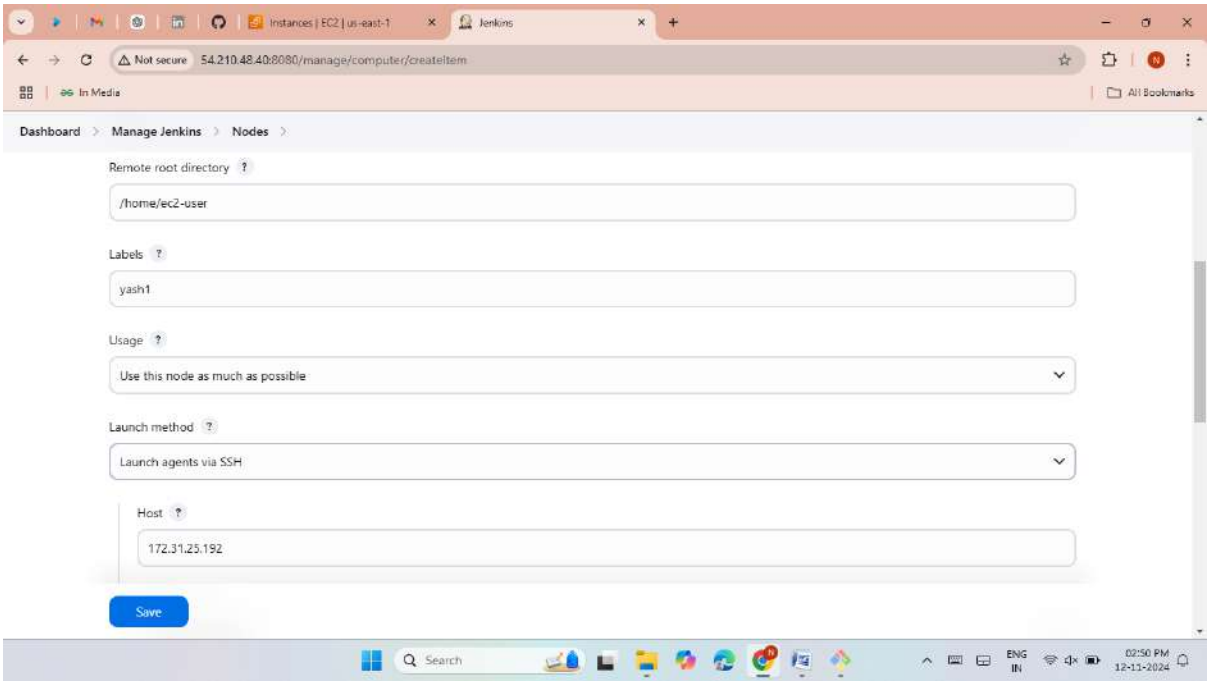


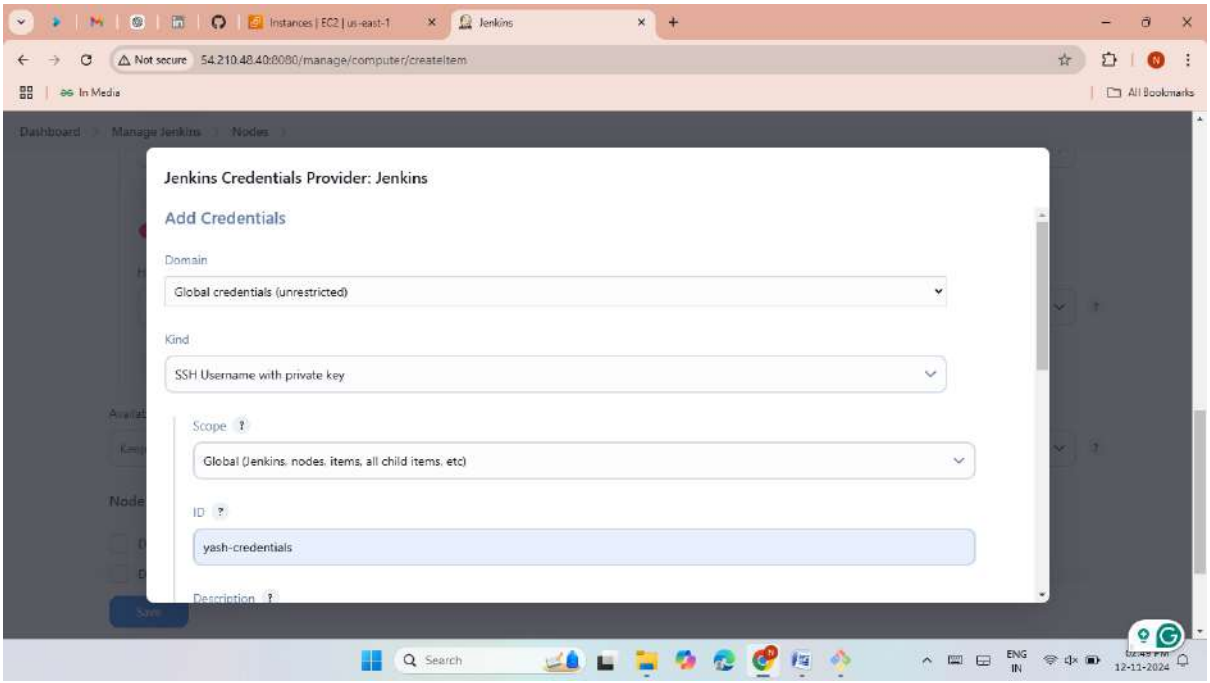
- Now create a node for slave1 server.



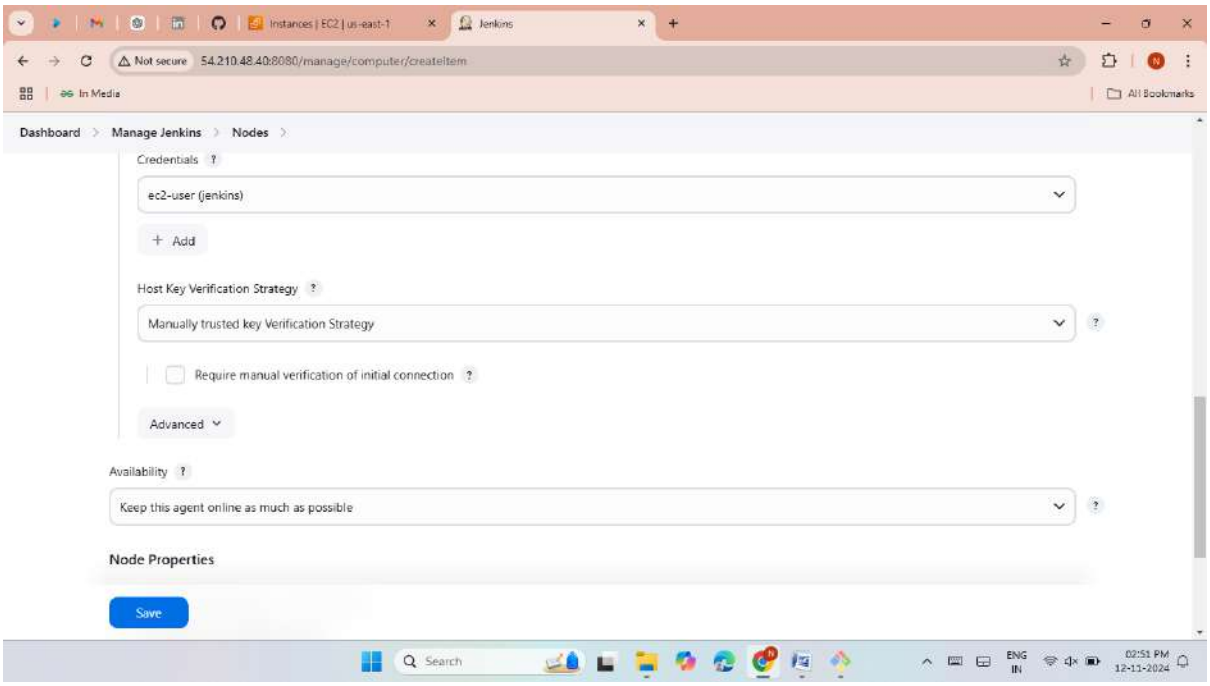
- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.



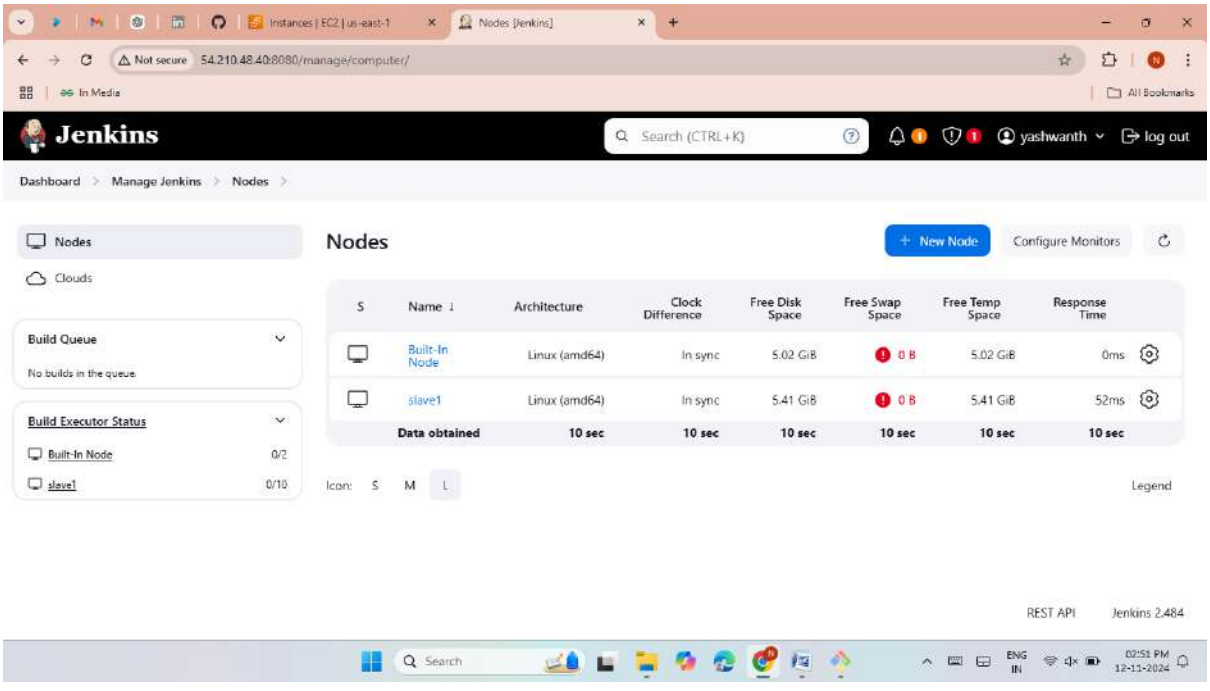

Instances | EC2 | us-east-1
Jenkins
Not secure 54.210.48.40:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Remote root directory
/home/ec2-user
Labels
yash1
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.31.25.192
Save
Search
ENG
IN
02:50 PM
12-11-2024


Instances | EC2 | us-east-1
Jenkins
Not secure 54.210.48.40:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
SSH Username with private key
Scope
Global (Jenkins, nodes, items, all child items, etc)
ID
yash-credentials
Description
Search
ENG
IN
12-11-2024

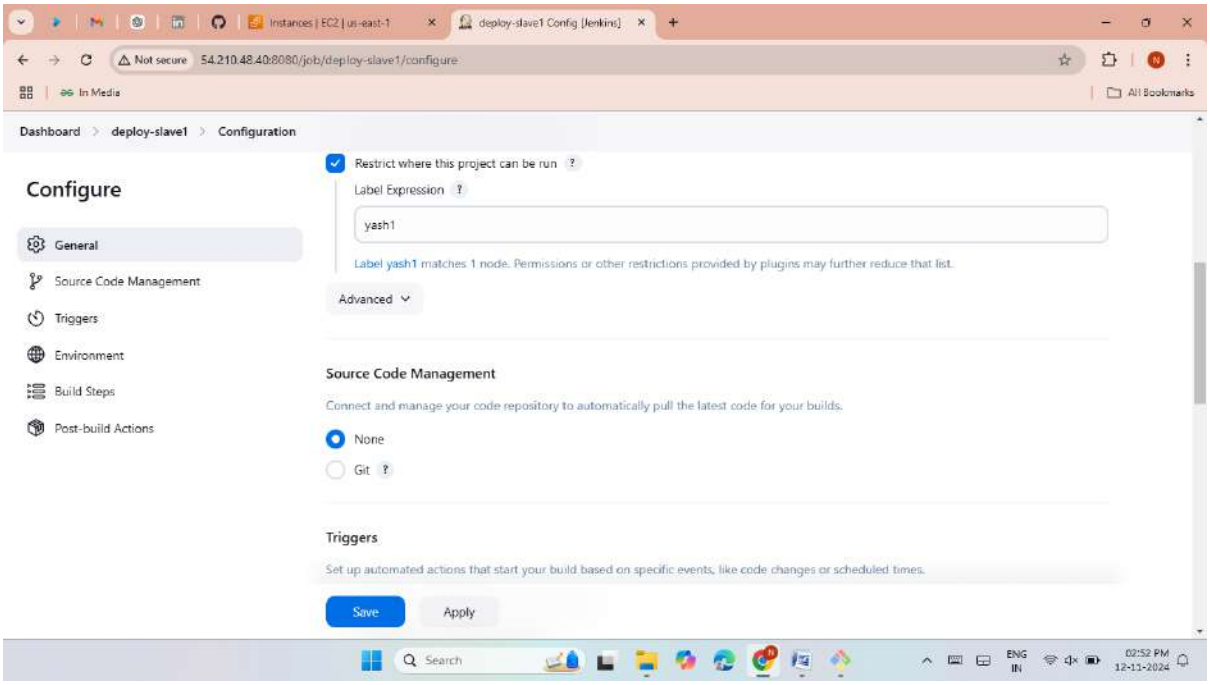
- Now click on save.



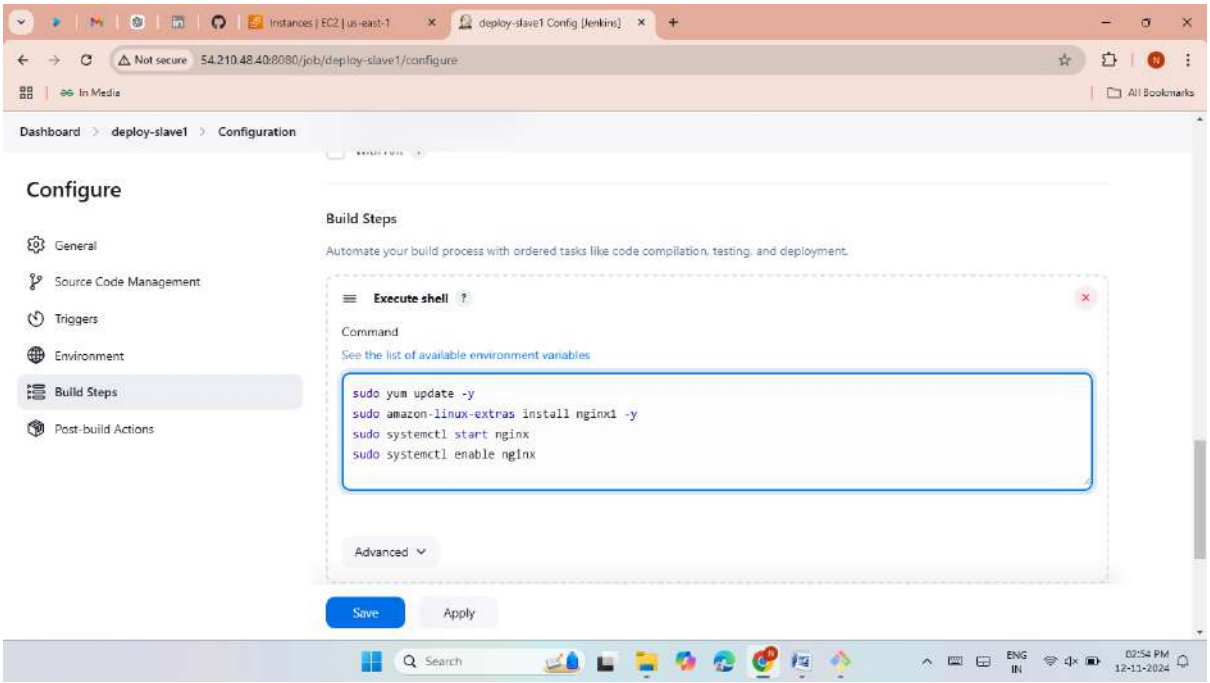
- Here the slave1 node is successfully created.



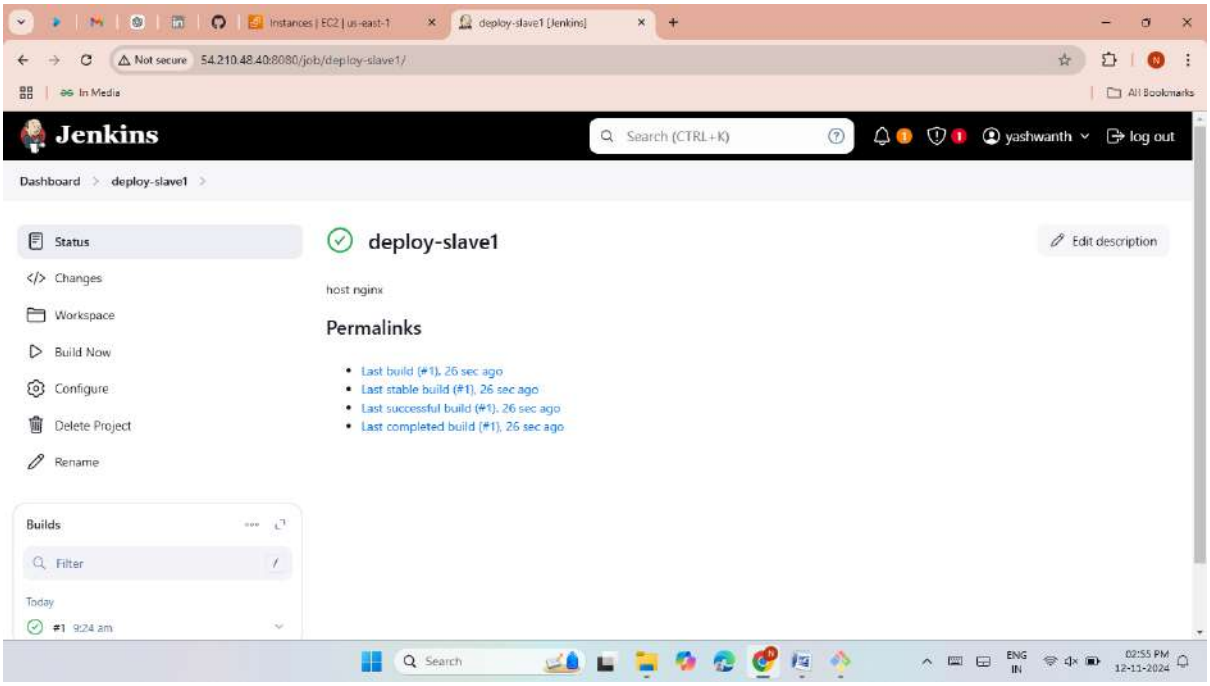
- Now create a job and give slave1 label.



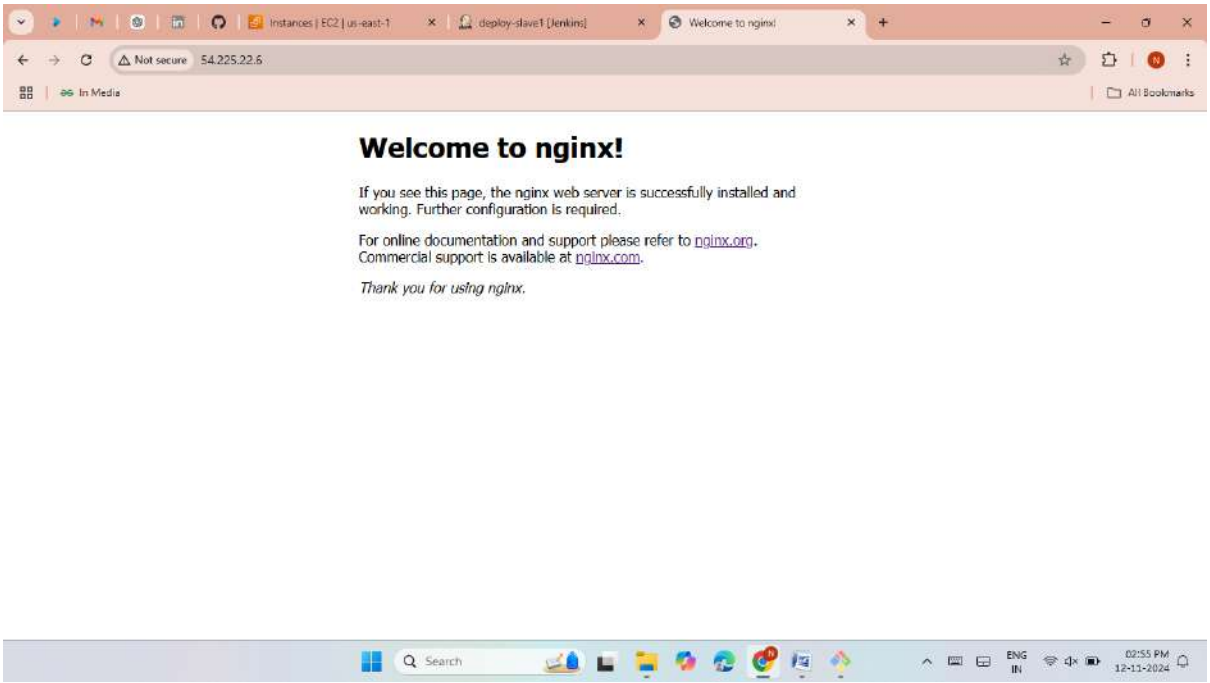
- Now install nginx and then start and enable.
- Now Click on save.



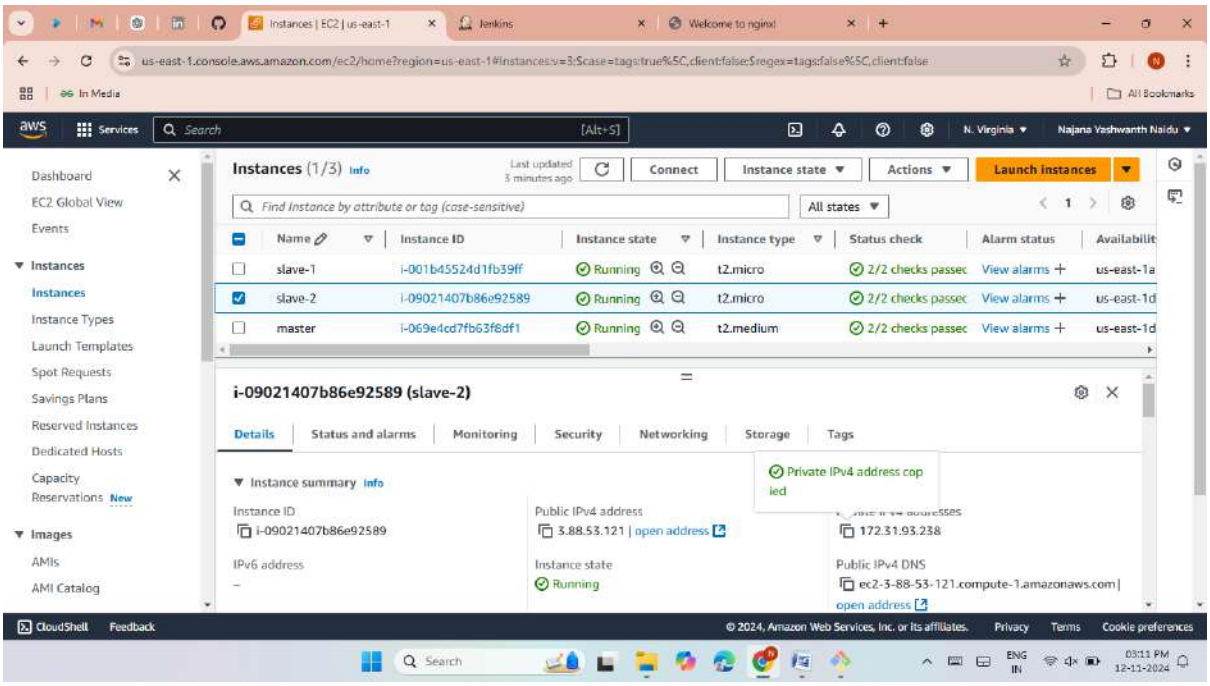
- Now click on build now.
- Here the job was success.



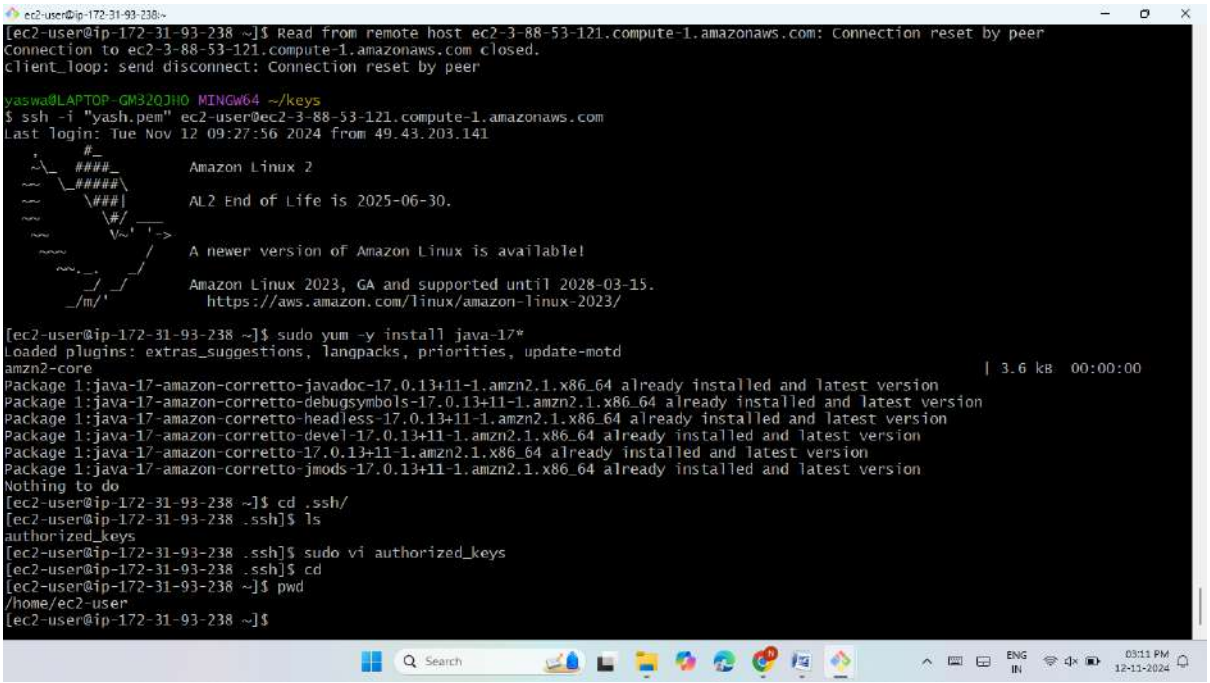
- Now copy the slave1 Public IP and past it on Google browser.
- The nginx web server was hosted successfully.



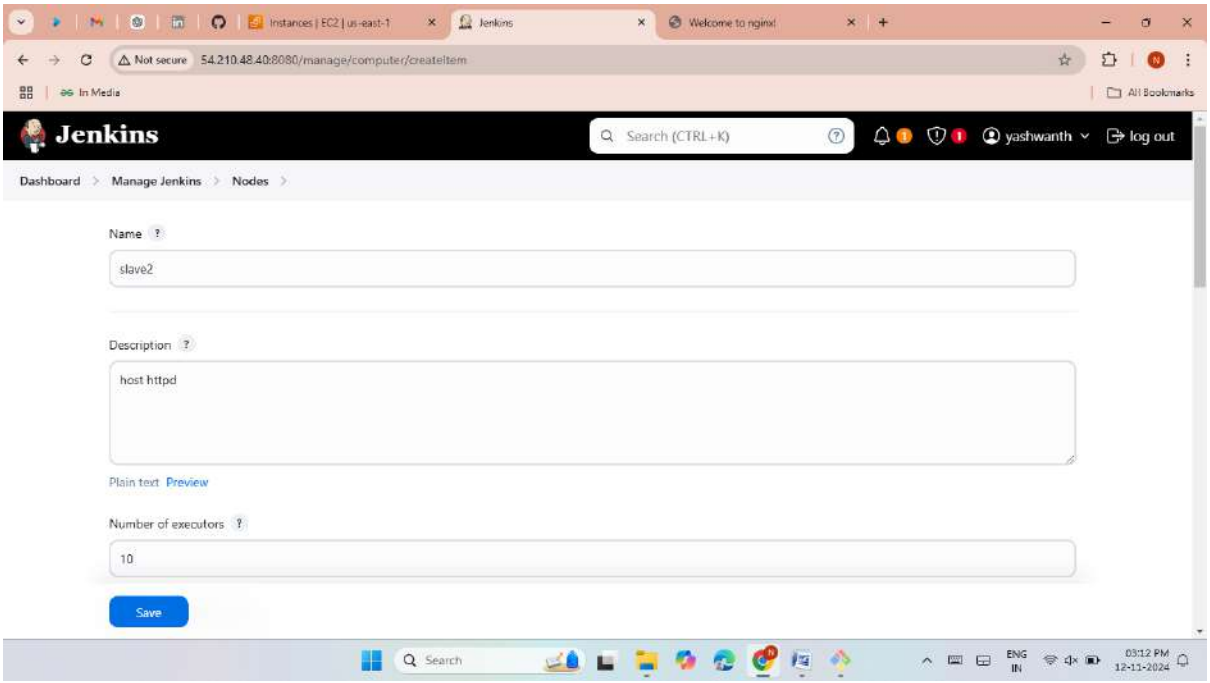
- Now launch an EC2 instances and give name as slave2.

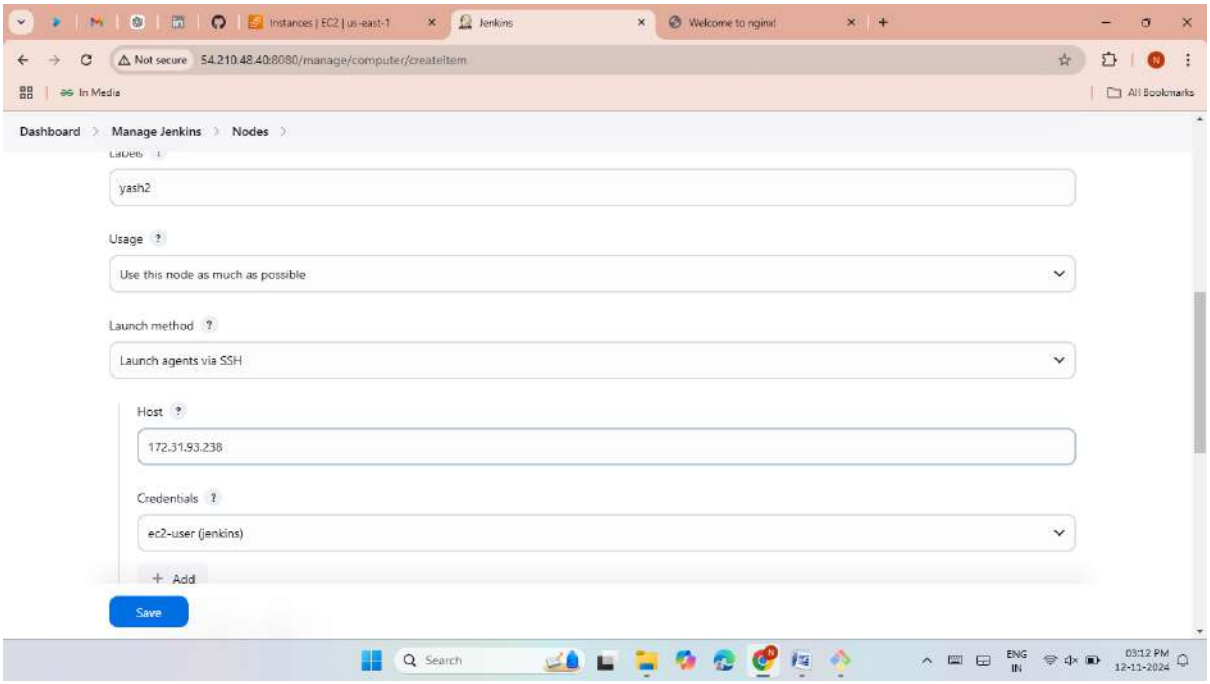


- Now connect Slave2 then install java.
- Now give private keys from master server.

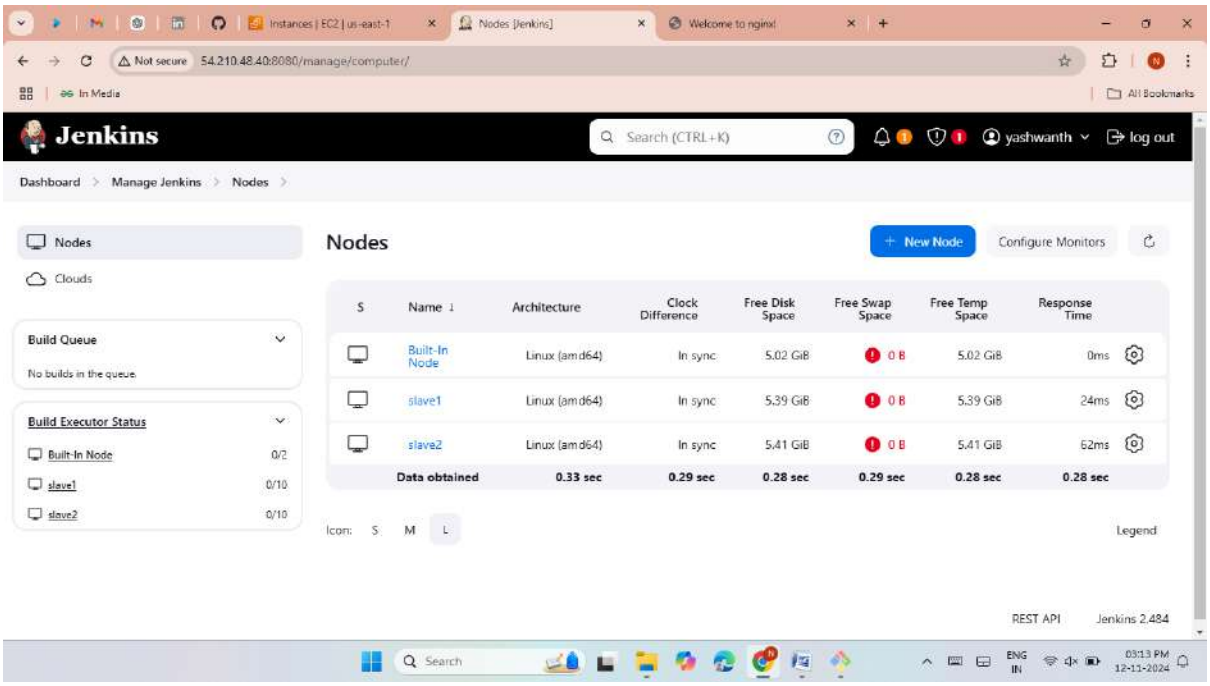


- Now create a node for slave2 server.
- Give details for slave node.
- Here we have to give slave2 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.

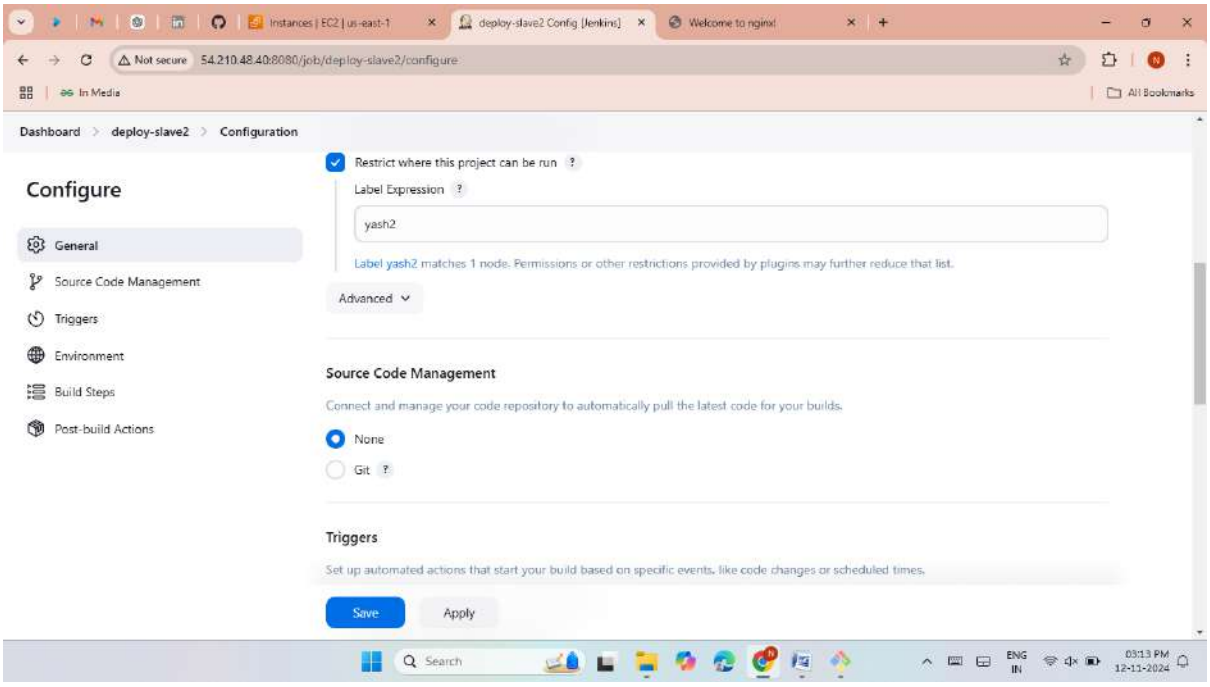




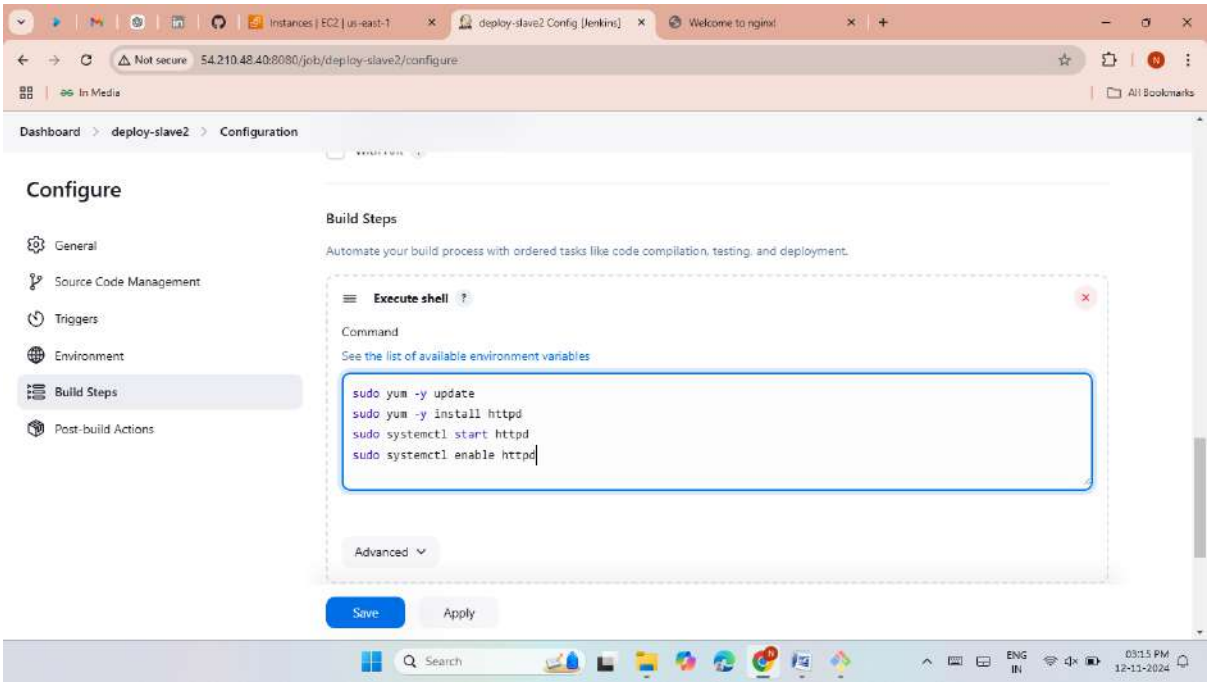
- Here the slave2 node is successfully created.



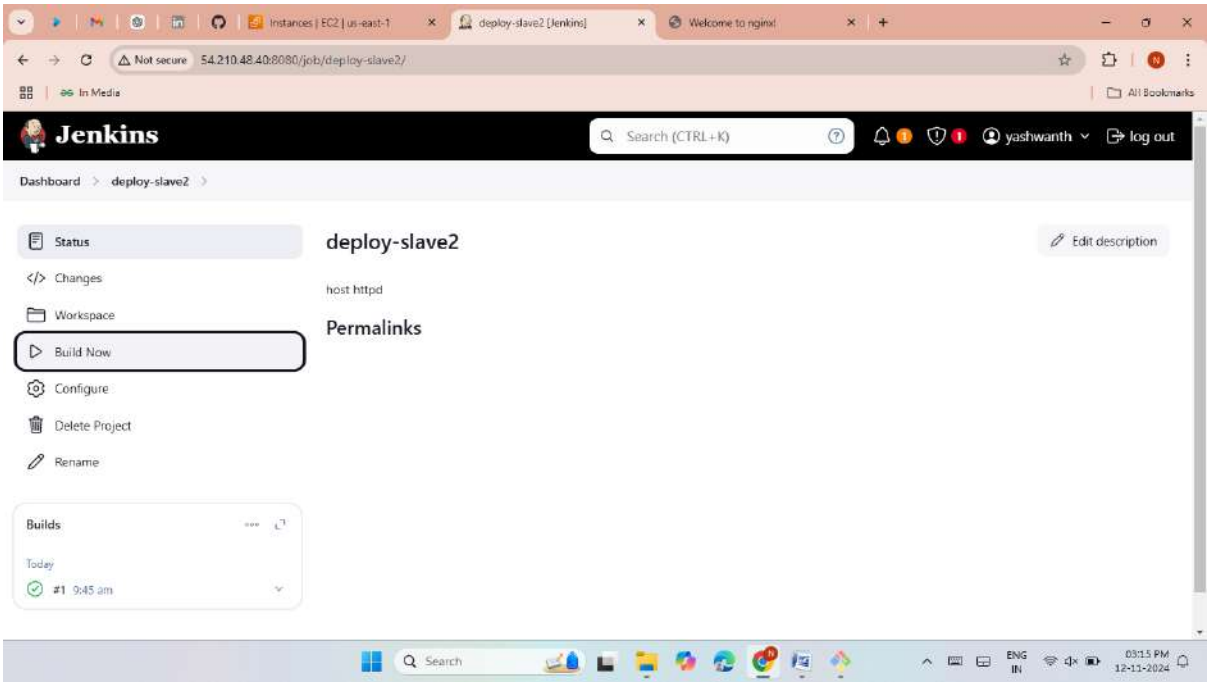
- Now create a job and give slave2 label.



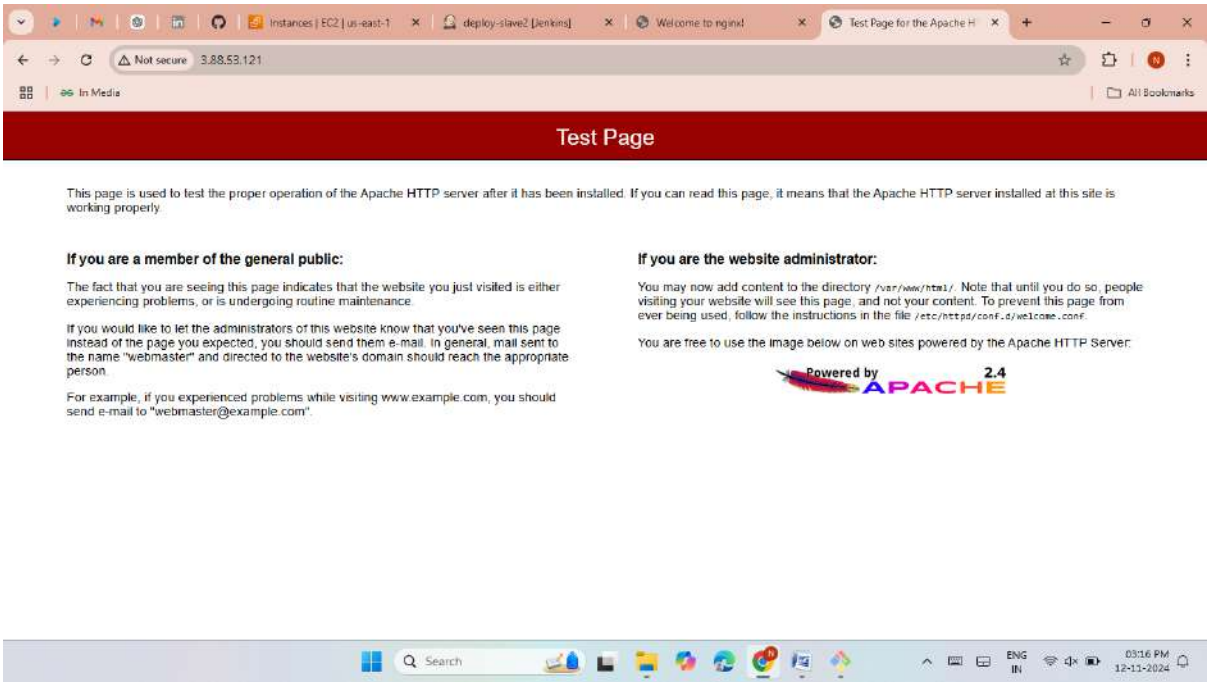
- Now install httpd and then start and enable.
- Now Click on save.



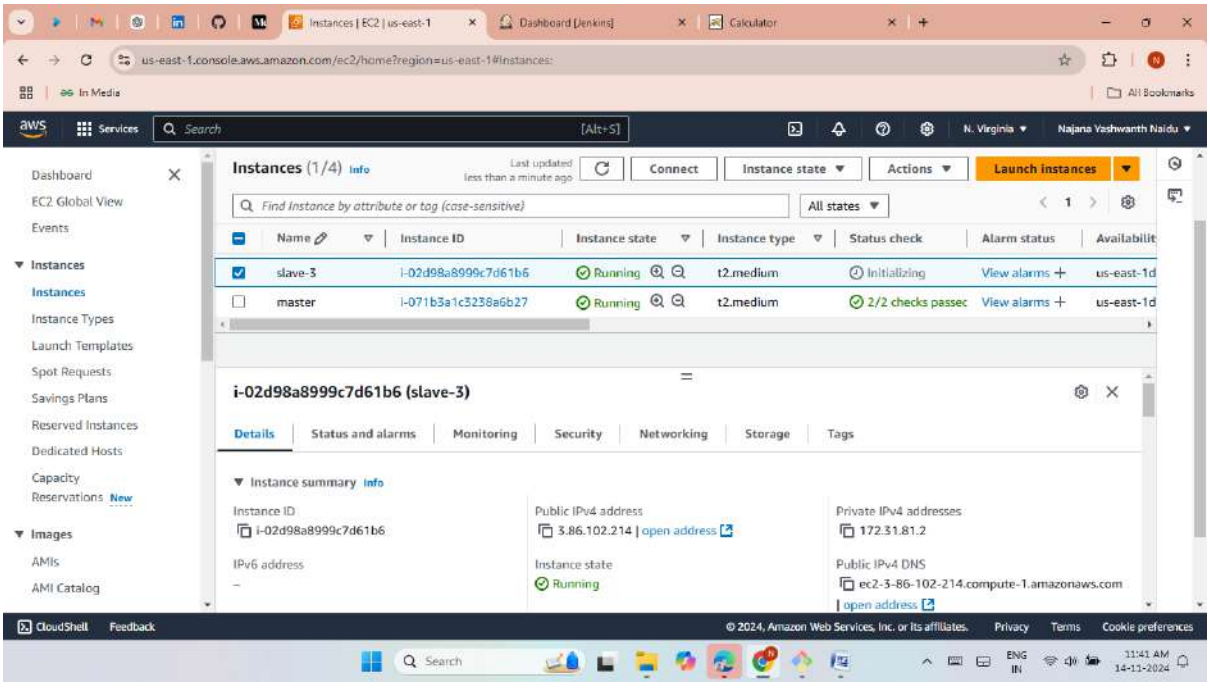
- Now click on build now.
- Here the job was success.



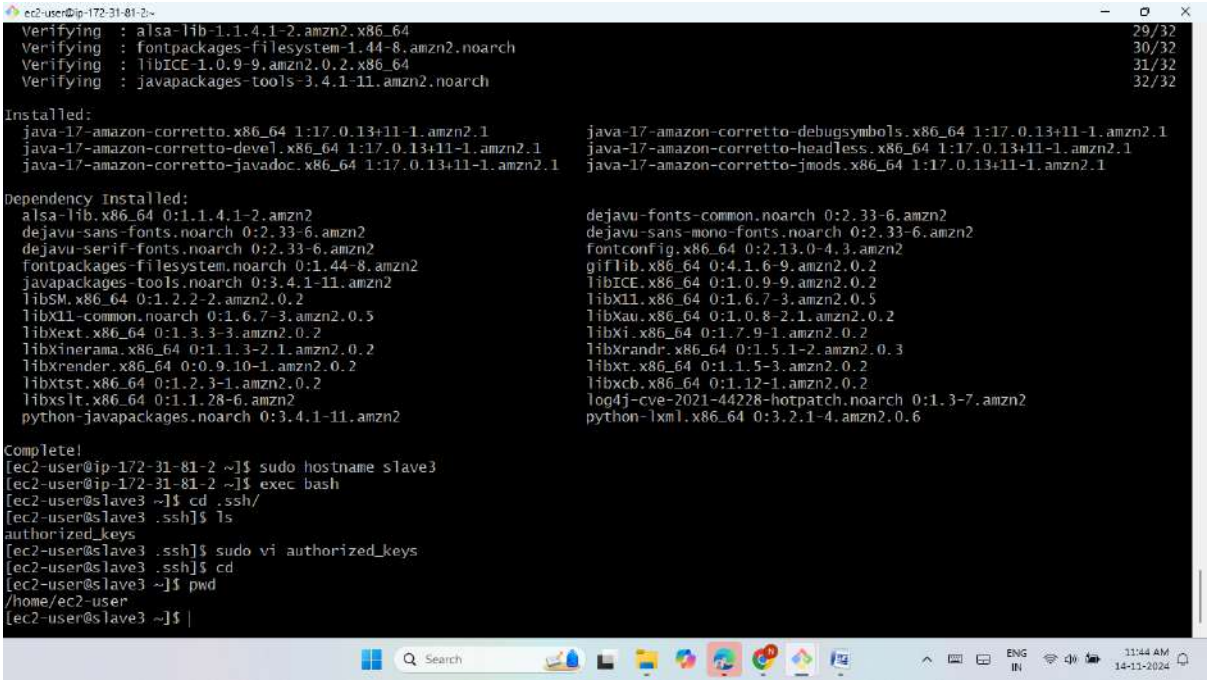
- Now copy the slave2 public ip and past it on google browser.
- The httpd web server was hosted successfully.



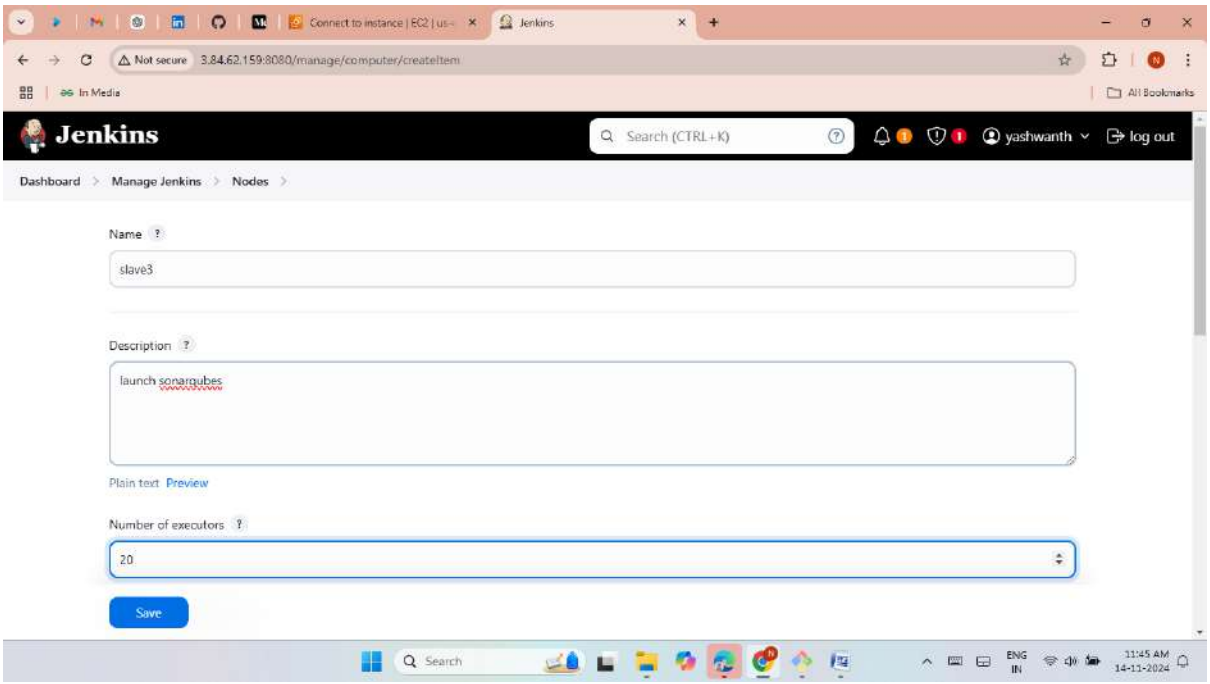
- Now launch an EC2 instances and give name as slave3.

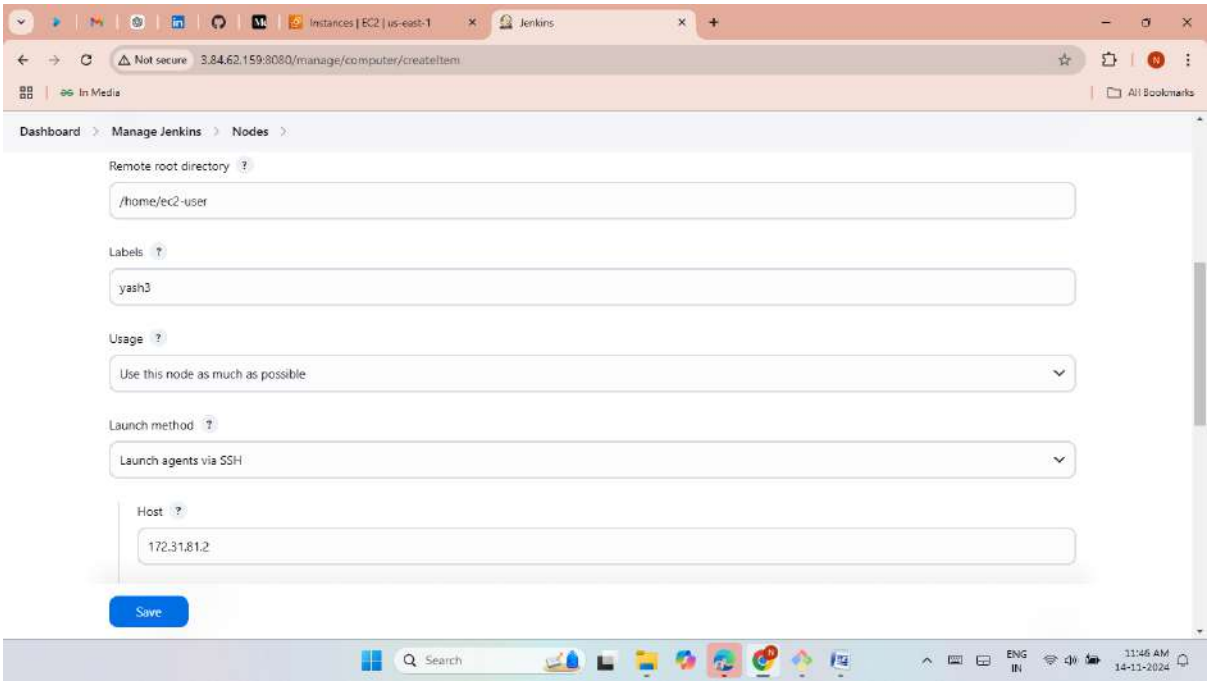


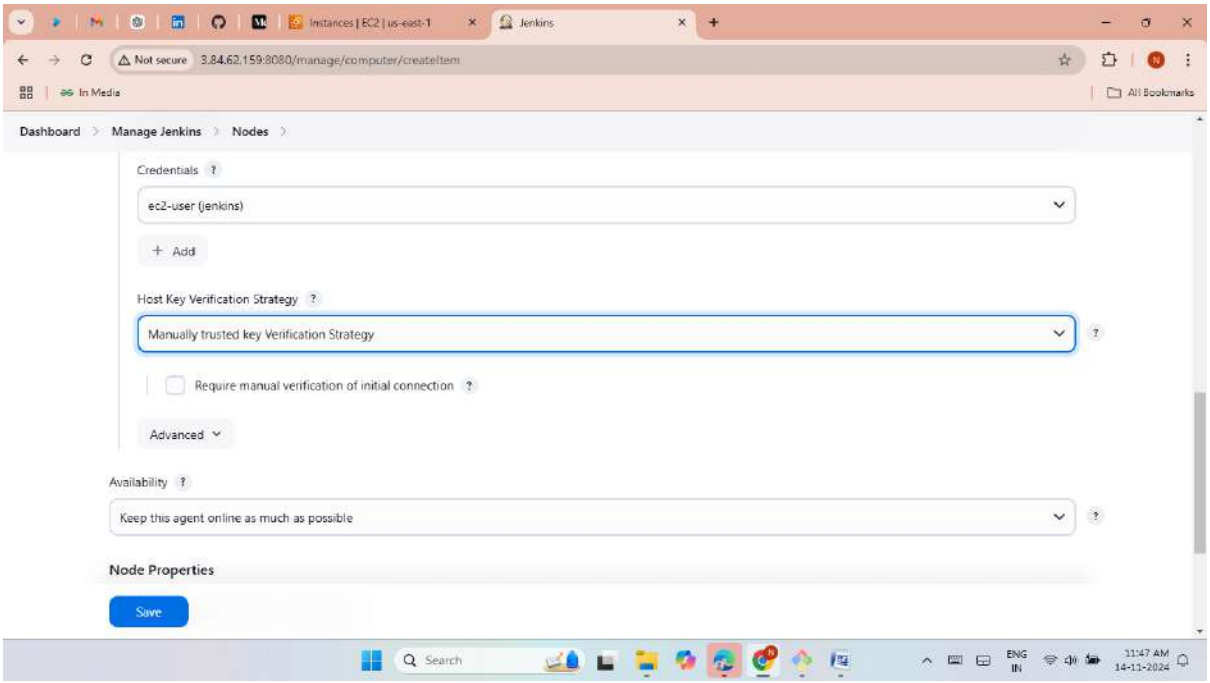
- Now connect Slave2 then install java.
- Now give private keys from master server.



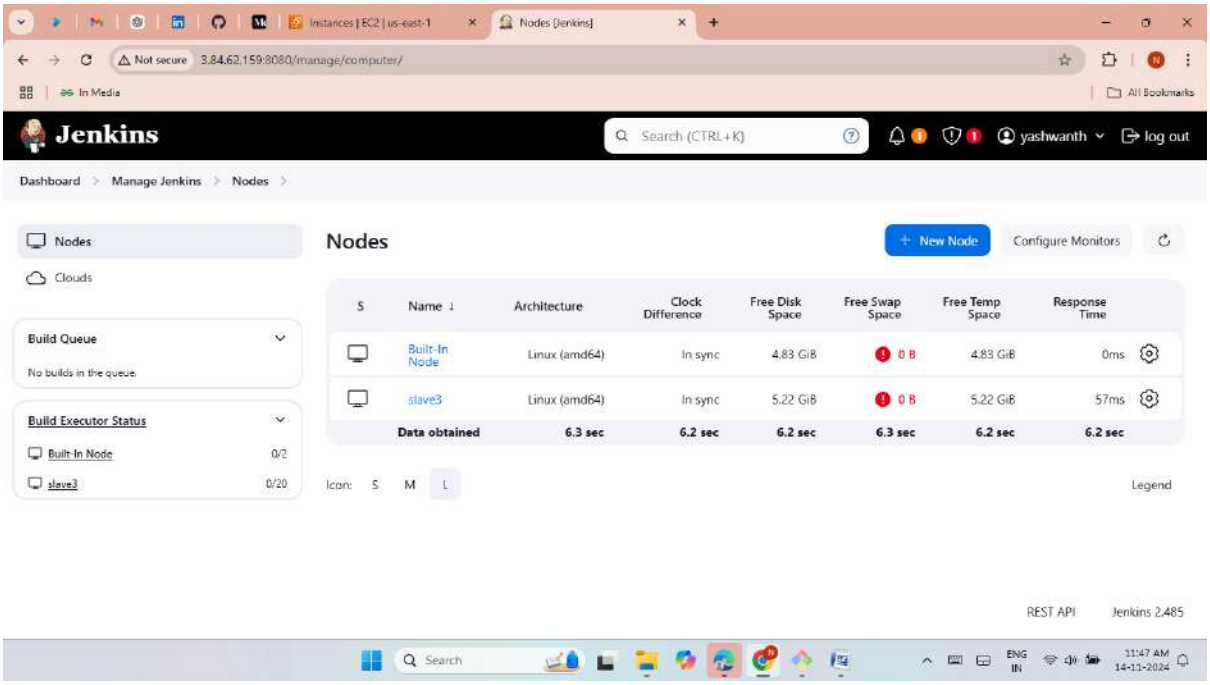
- Now create a node for slave3 server.
- Give details for slave node.
- Here we have to give slave3 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.



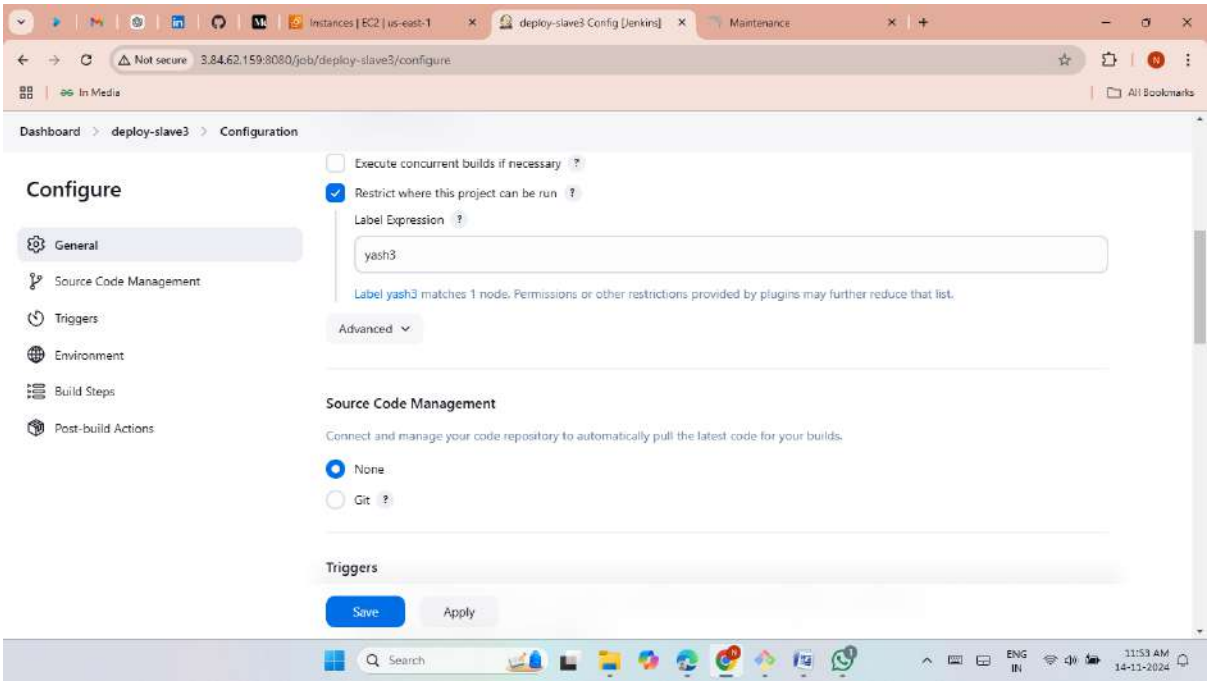

Instances | EC2 | us-east-1
Jenkins
Not secure 3.84.62.159:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Remote root directory
/home/ec2-user
Labels
yash3
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.31.81.2
Save
Search
ENG
IN
11:46 AM
14-11-2024


Instances | EC2 | us-east-1
Jenkins
Not secure 3.84.62.159:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Credentials
ec2-user (jenkins)
+ Add
Host Key Verification Strategy
Manually trusted key Verification Strategy
Require manual verification of initial connection
Advanced
Availability
Keep this agent online as much as possible
Node Properties
Save
Search
ENG
IN
11:47 AM
14-11-2024

- Here the slave3 node is successfully created.

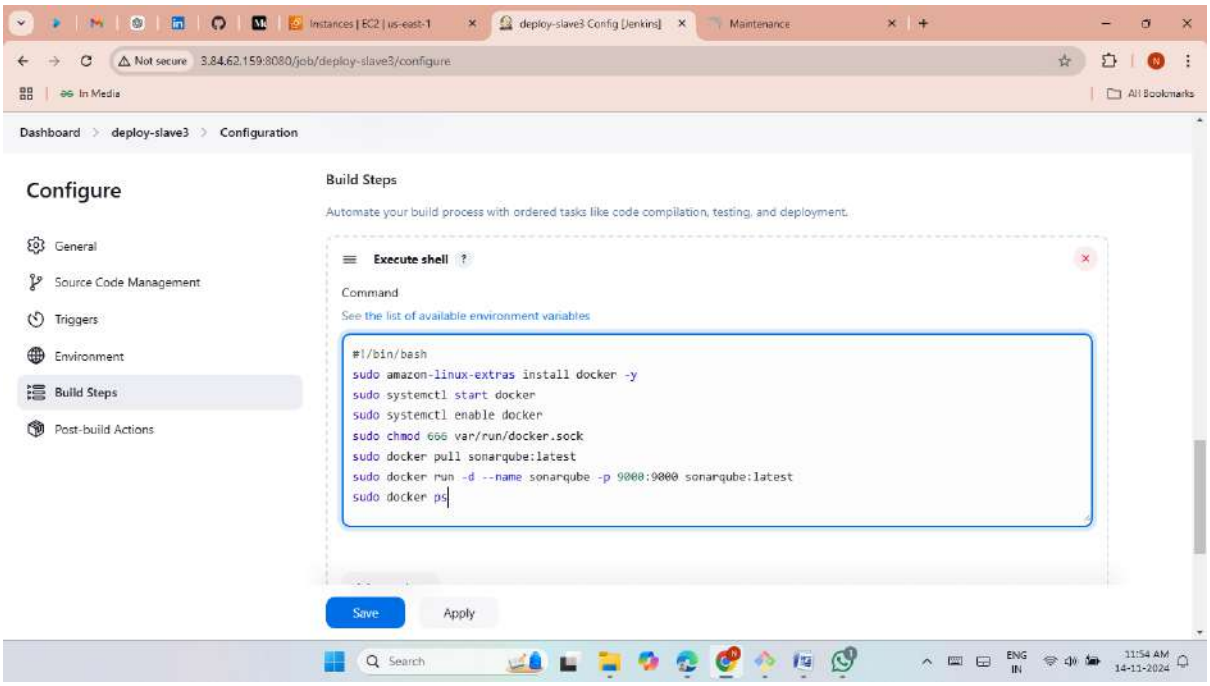


- Now create a job and give slave3 label.

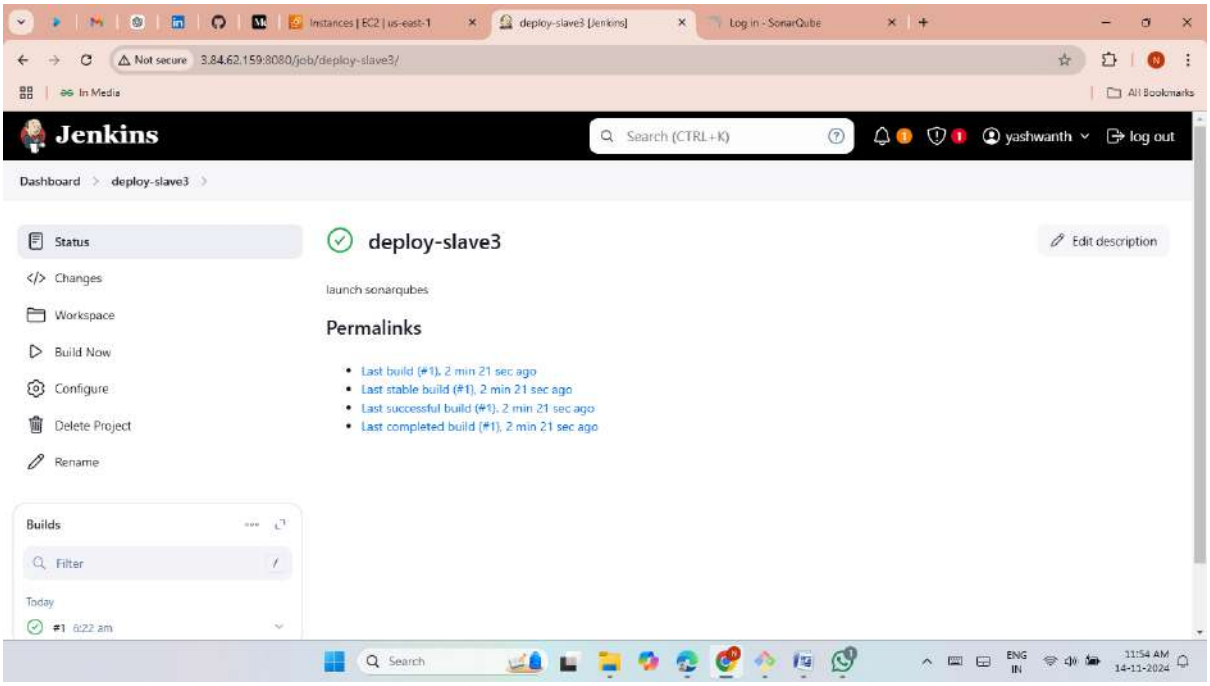


- Now install docker and then start and enable.
- Now pull sonarqube then run the sonarqubes.
- Now Click on save.

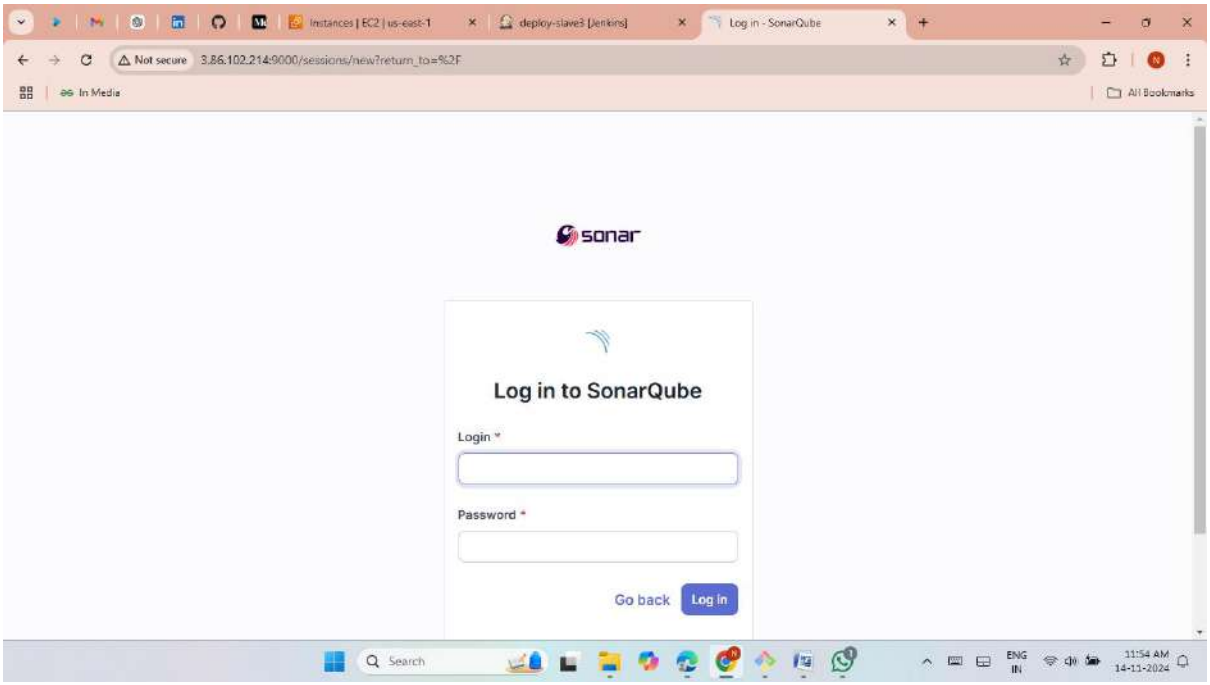
#!/bin/bash
sudo amazon-linux-extras install docker -y
sudo systemctl start docker
sudo systemctl enable docker
sudo chmod 666 var/run/docker.sock
sudo docker pull sonarqube:latest
sudo docker run -d --name sonarqube -p 9000:9000 sonarqube:latest
sudo docker ps

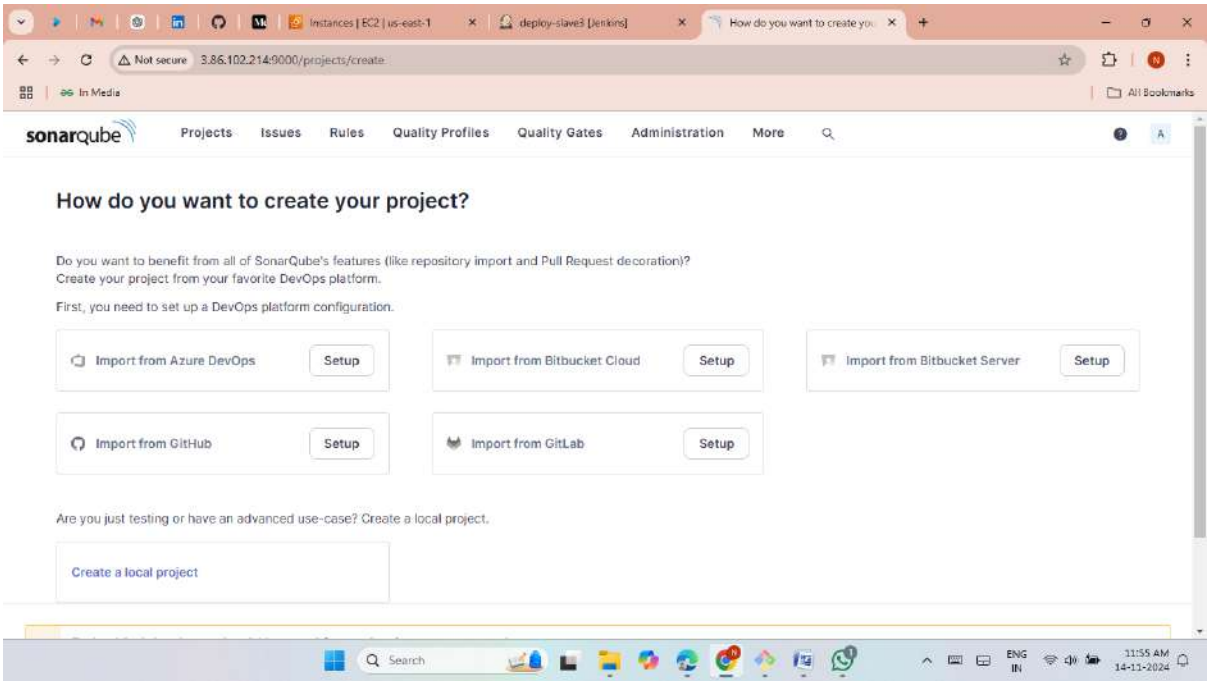


- Now click on build now.
- Here the job was success.

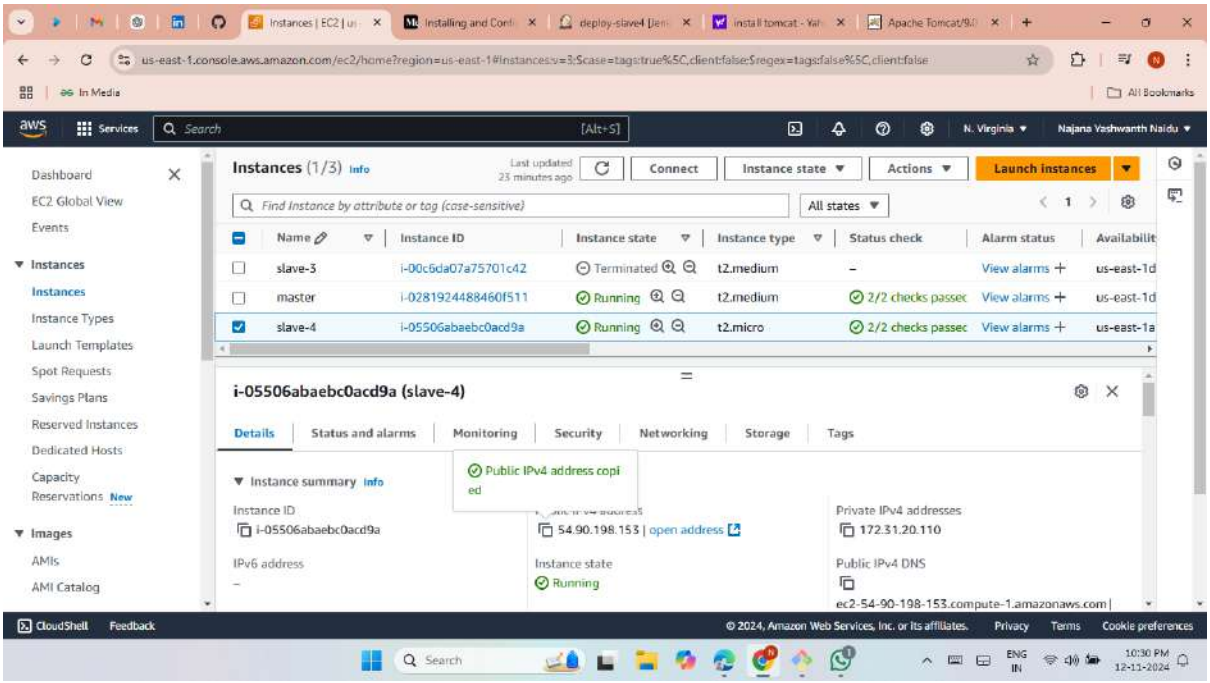


- Now copy the slave3 public ip and past it on Google browser.
- The sonarqubes web server was hosted successfully.

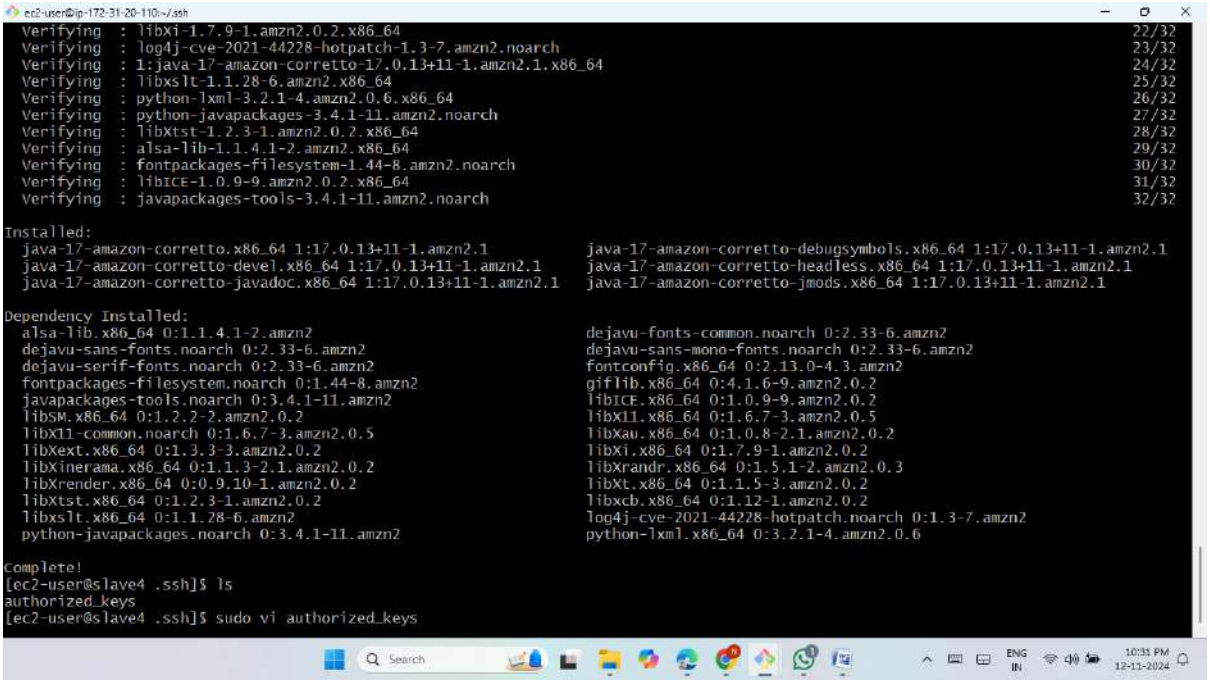




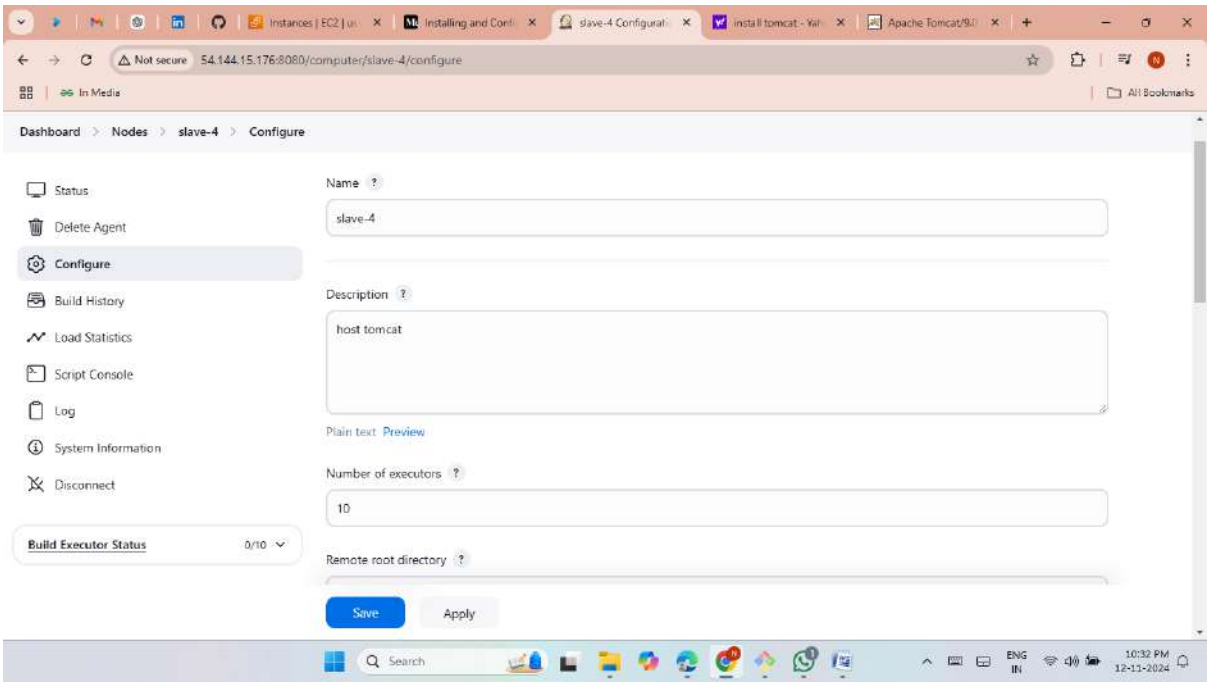
- Now launch an EC2 instances and give name as slave4.

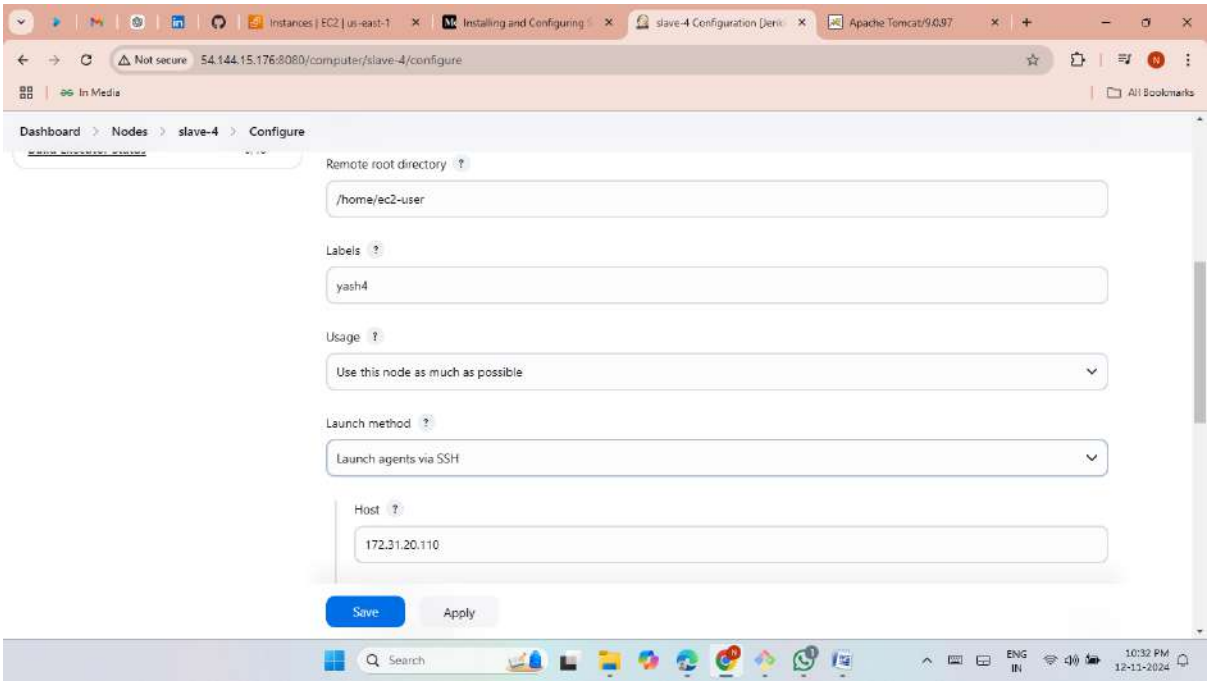


- Now connect Slave4 then install java.
- Now give private keys from master server.

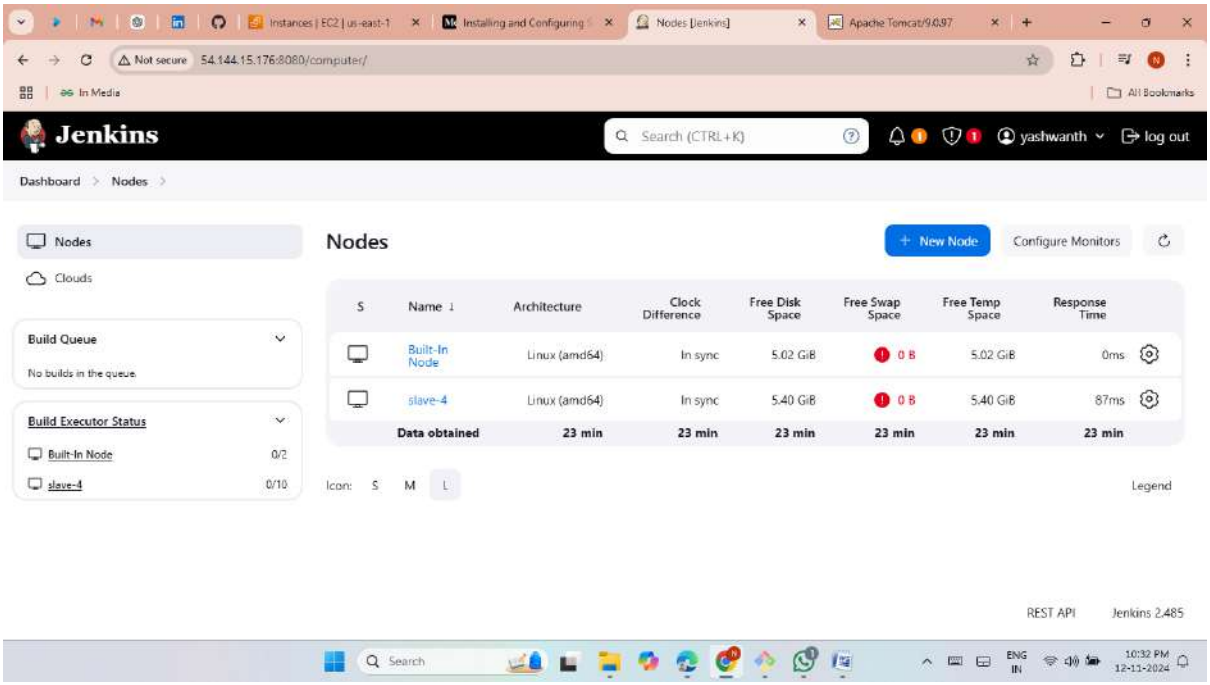


- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.

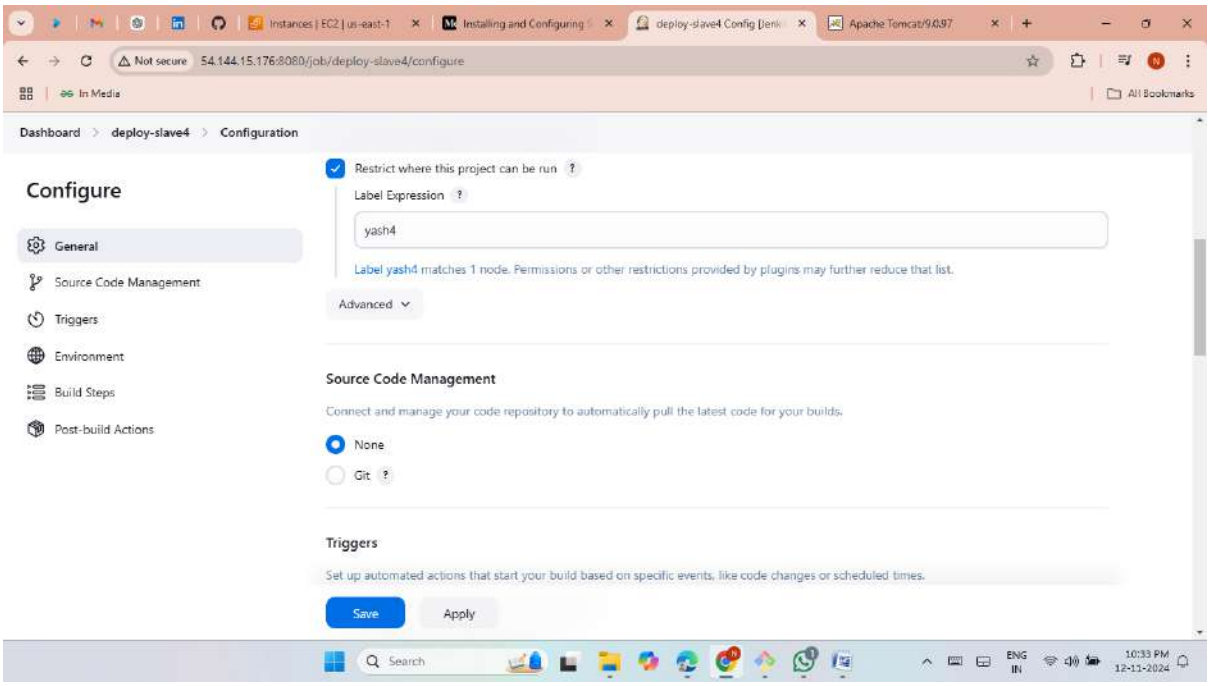




- Here the slave4 node is successfully created.



- 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

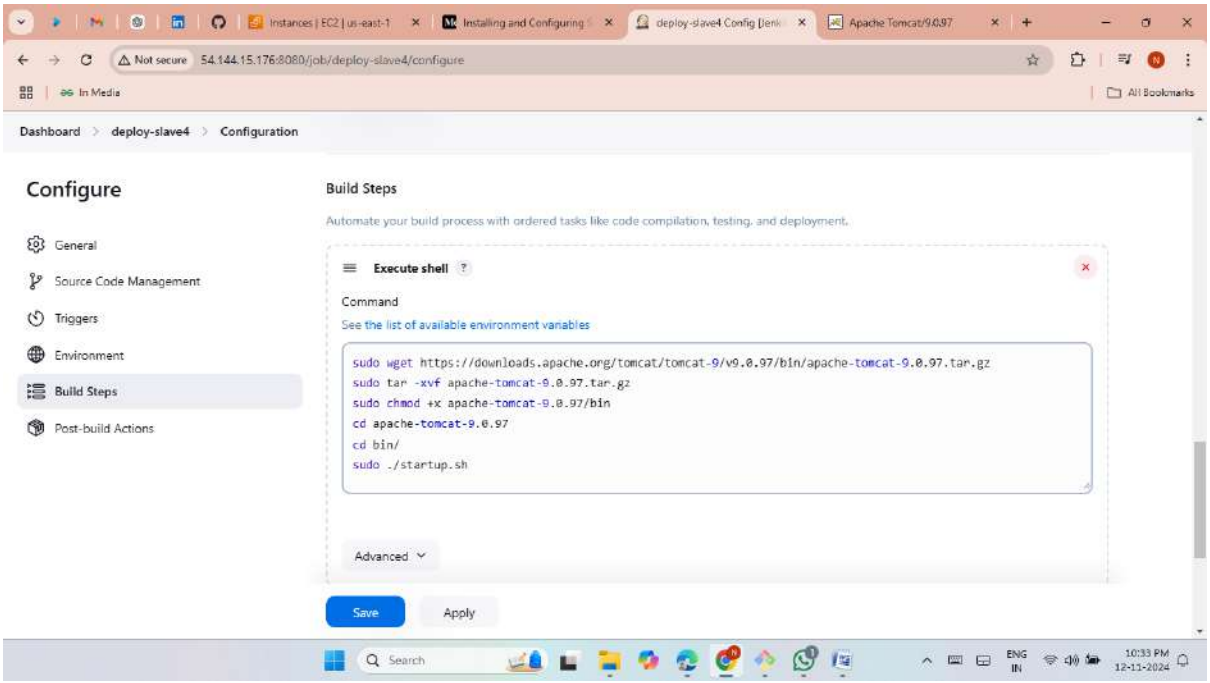
sudo chmod +x apache-tomcat-9.0.97/bin

cd apache-tomcat-9.0.97

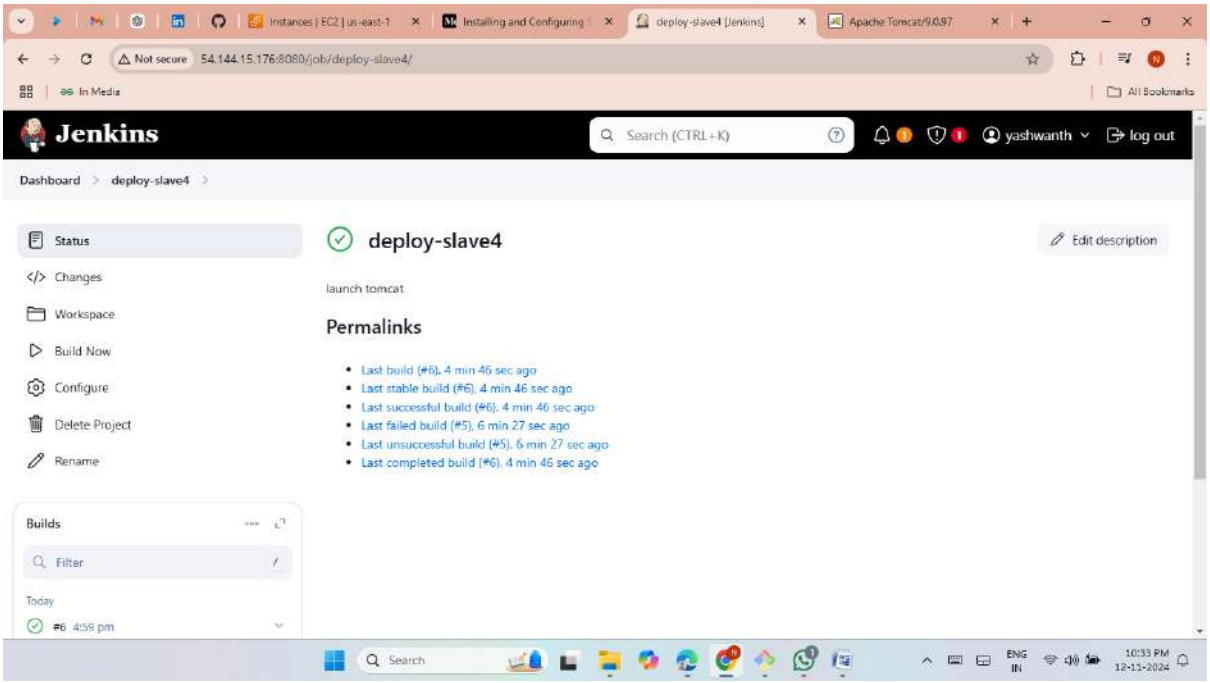
cd bin/

sudo ./startup.sh

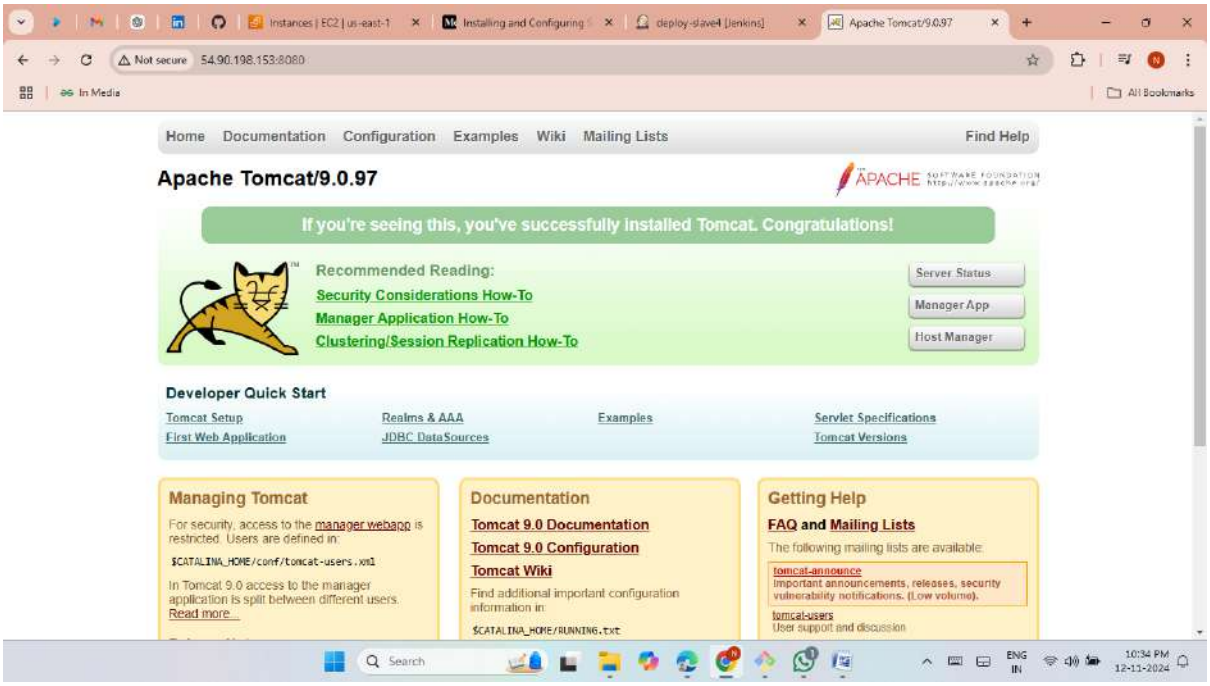
Now Click on save.



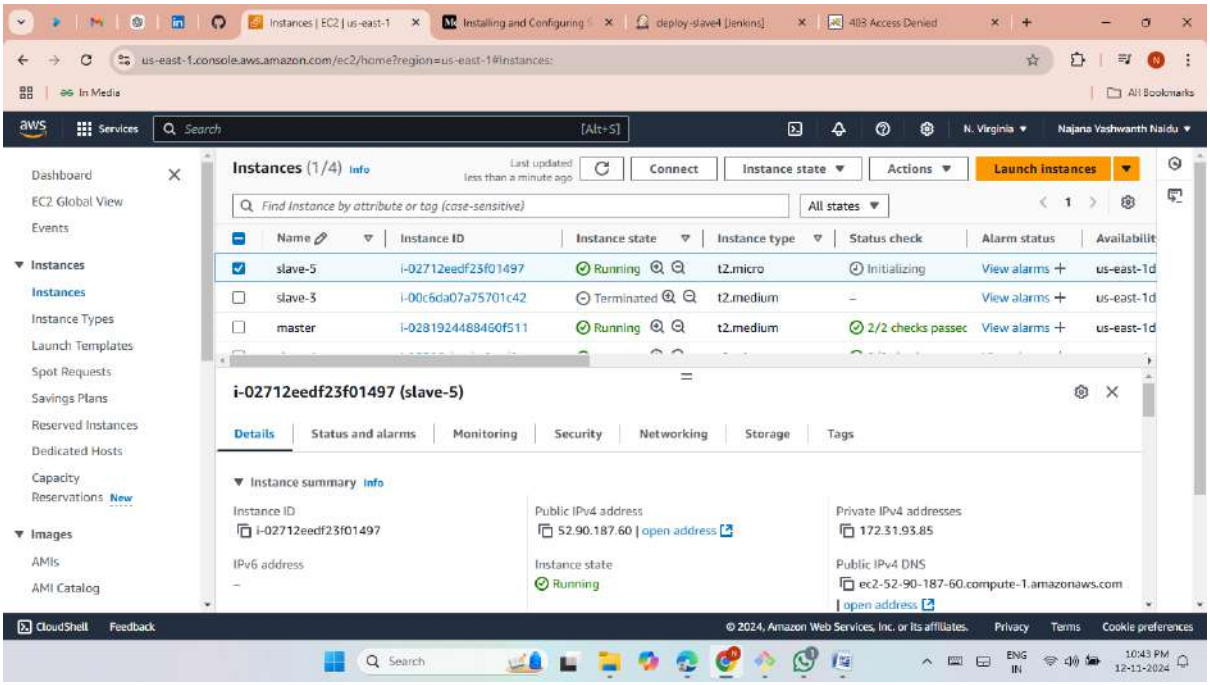
- Now click on build now.
- Here the job was success.



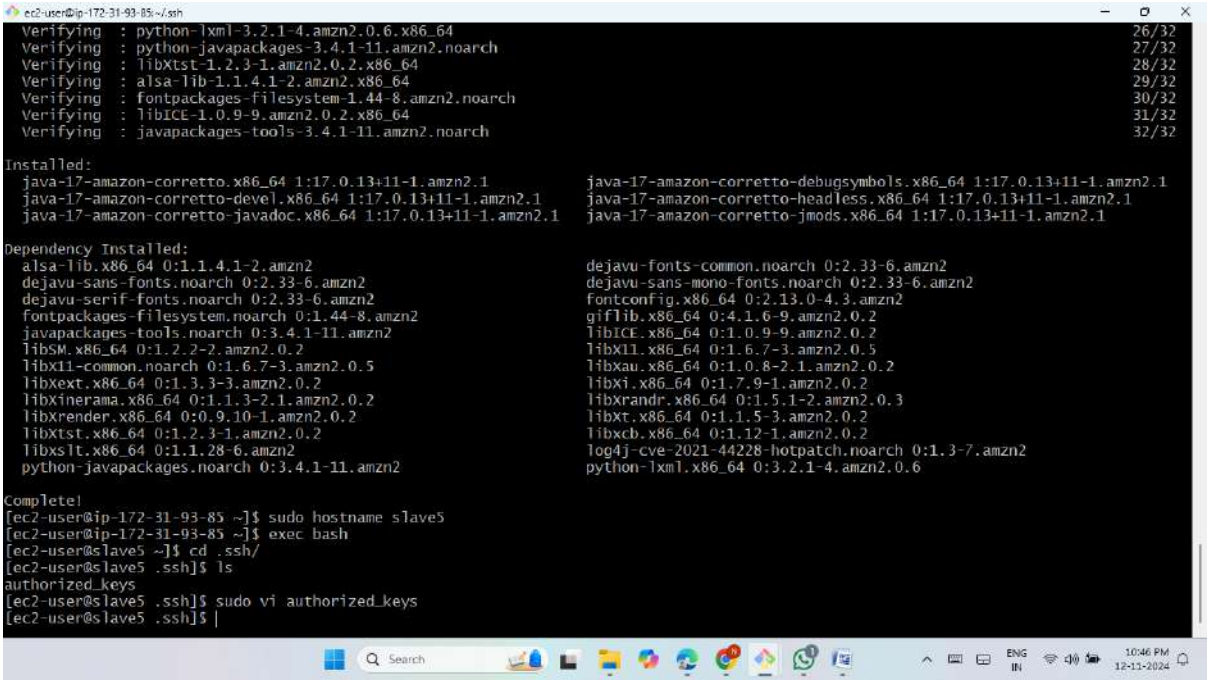
- Now copy the slave4 public ip with the tomcat port (8080) and past it on Google browser.
- The tomcat web server was hosted successfully.



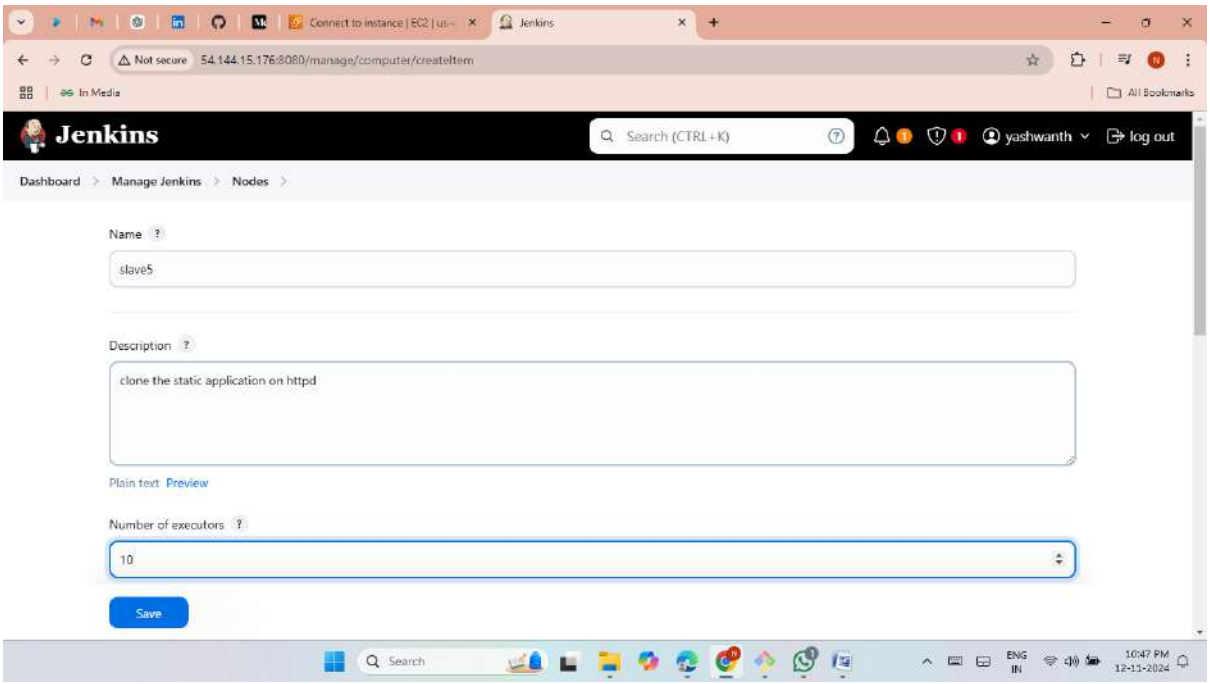
- Now launch an ec2 instances and give name as slave5.

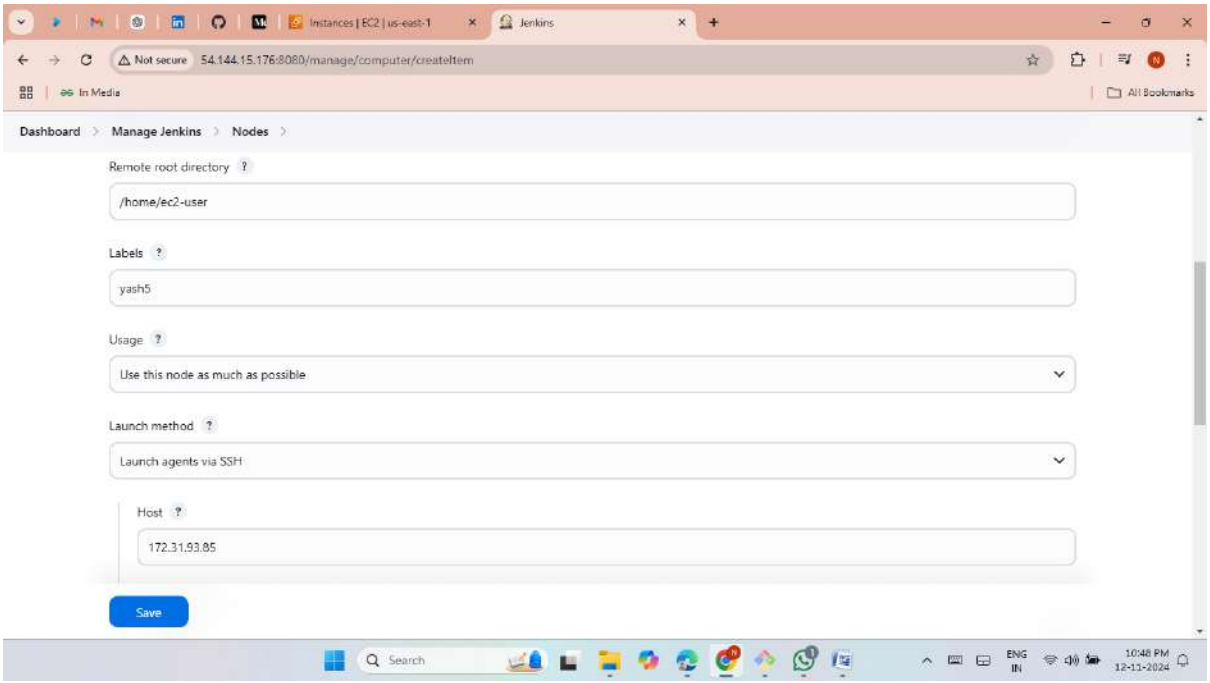


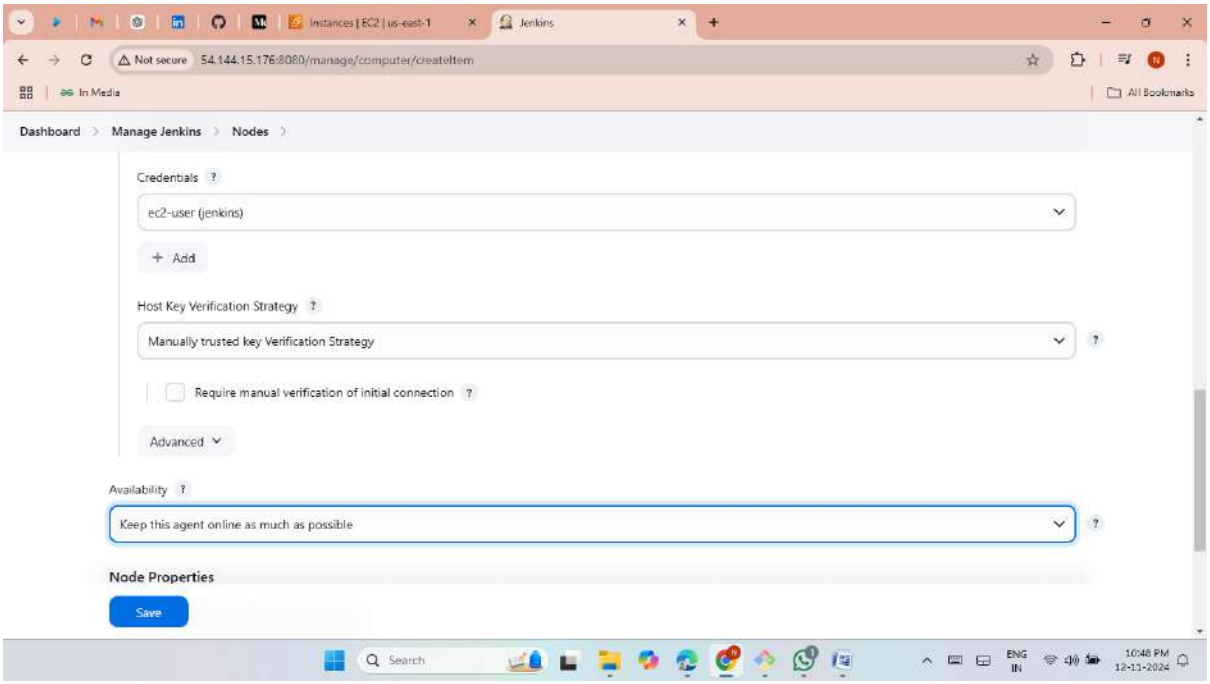
- Now connect Slave5 then install java.
- Now give private keys from master server.



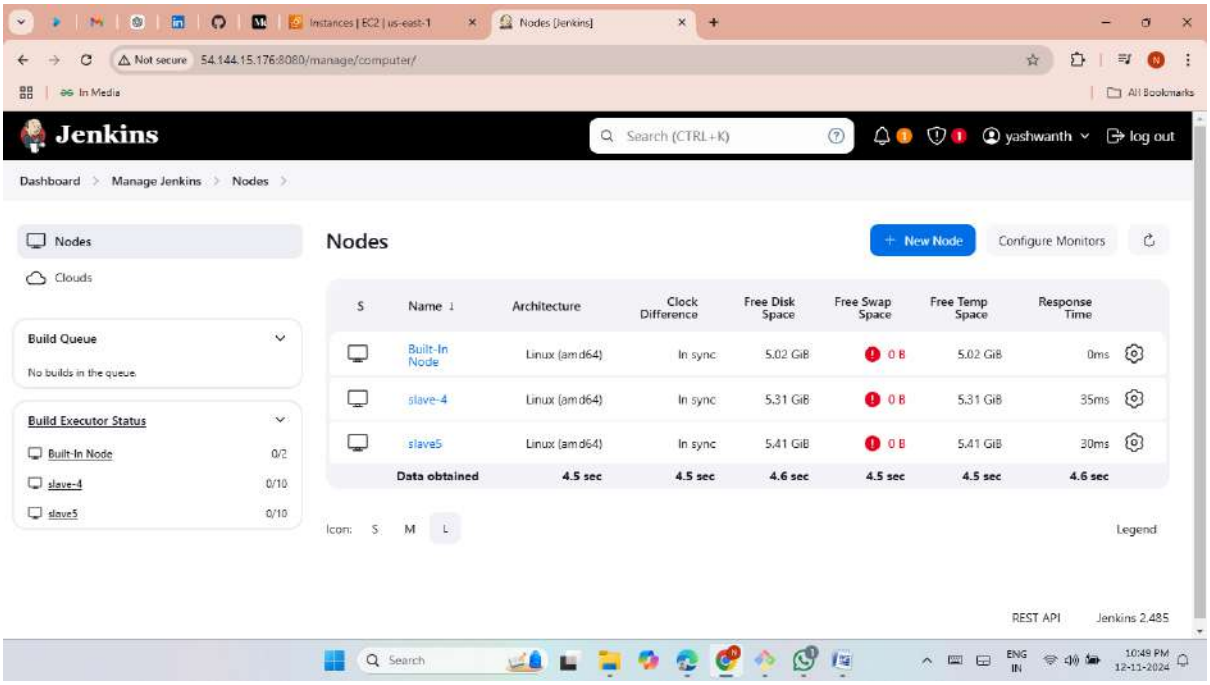
- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.



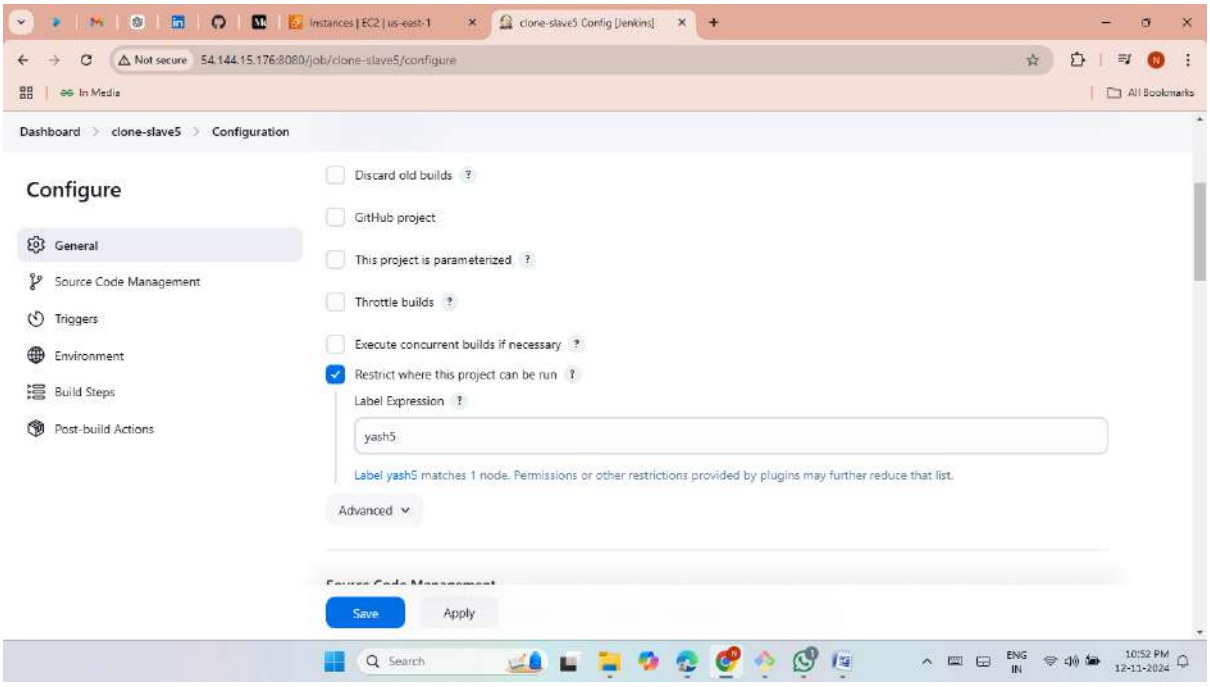

Instances | EC2 | us-east-1
Jenkins
Not secure 54.144.15.176:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Remote root directory
/home/ec2-user
Labels
yash5
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.31.93.85
Save
Search
ENG
IN
10:48 PM
12-11-2024


Instances | EC2 | us-east-1
Jenkins
Not secure 54.144.15.176:8080/manage/computer/createItem
In Media
All Bookmarks
Dashboard > Manage Jenkins > Nodes >
Credentials
ec2-user (jenkins)
+ Add
Host Key Verification Strategy
Manually trusted key Verification Strategy
Require manual verification of initial connection
Advanced
Availability
Keep this agent online as much as possible
Node Properties
Save
Search
ENG
IN
10:48 PM
12-11-2024

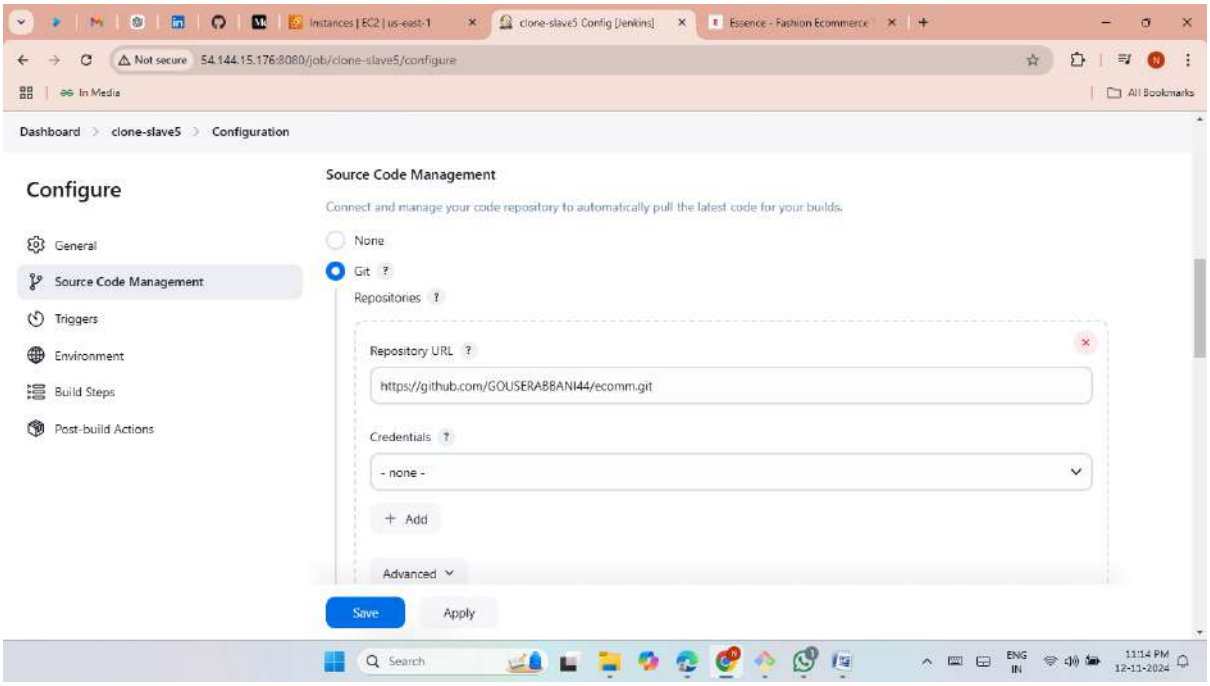
- Here the slave5 node is successfully created.



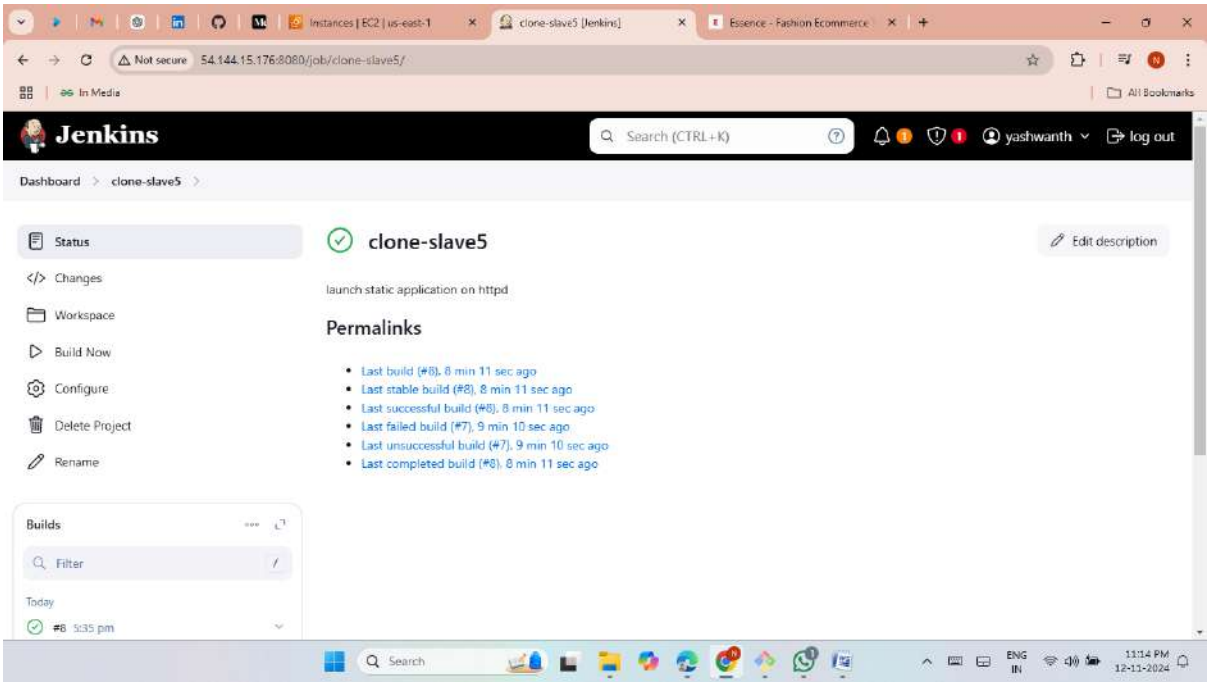
- Now create a job and give slave5 label.



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



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



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

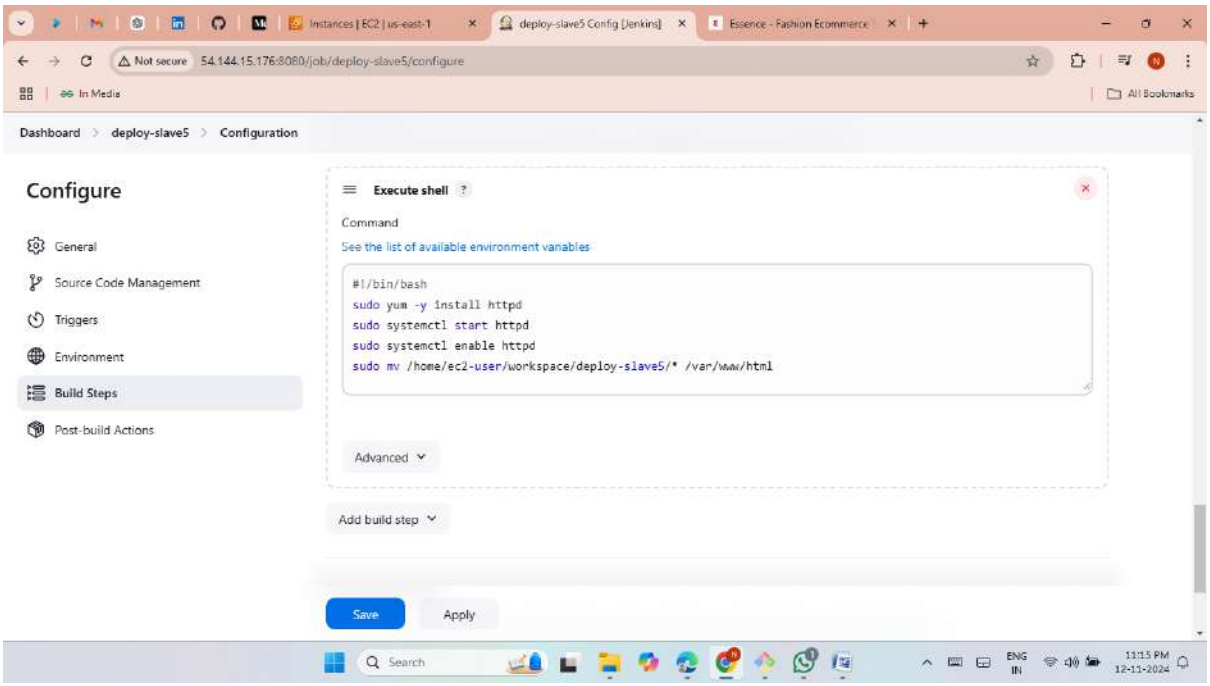
#!/bin/bash

sudo yum -y install httpd

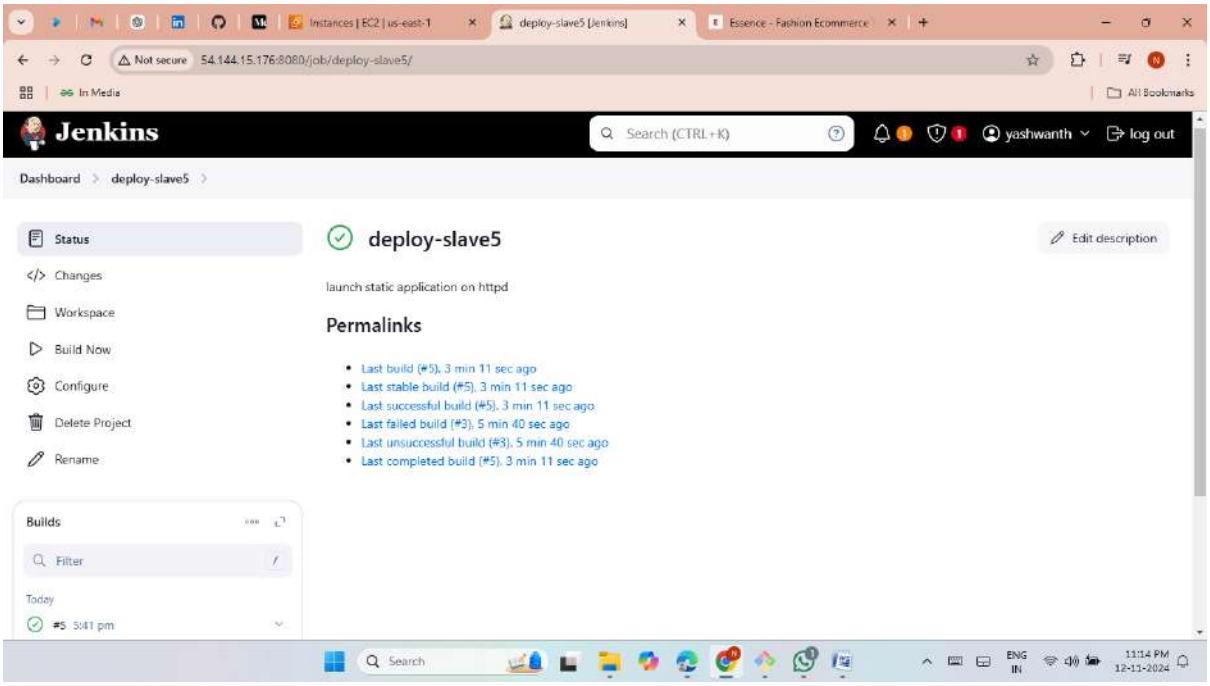
sudo systemctl start httpd

sudo systemctl enable httpd

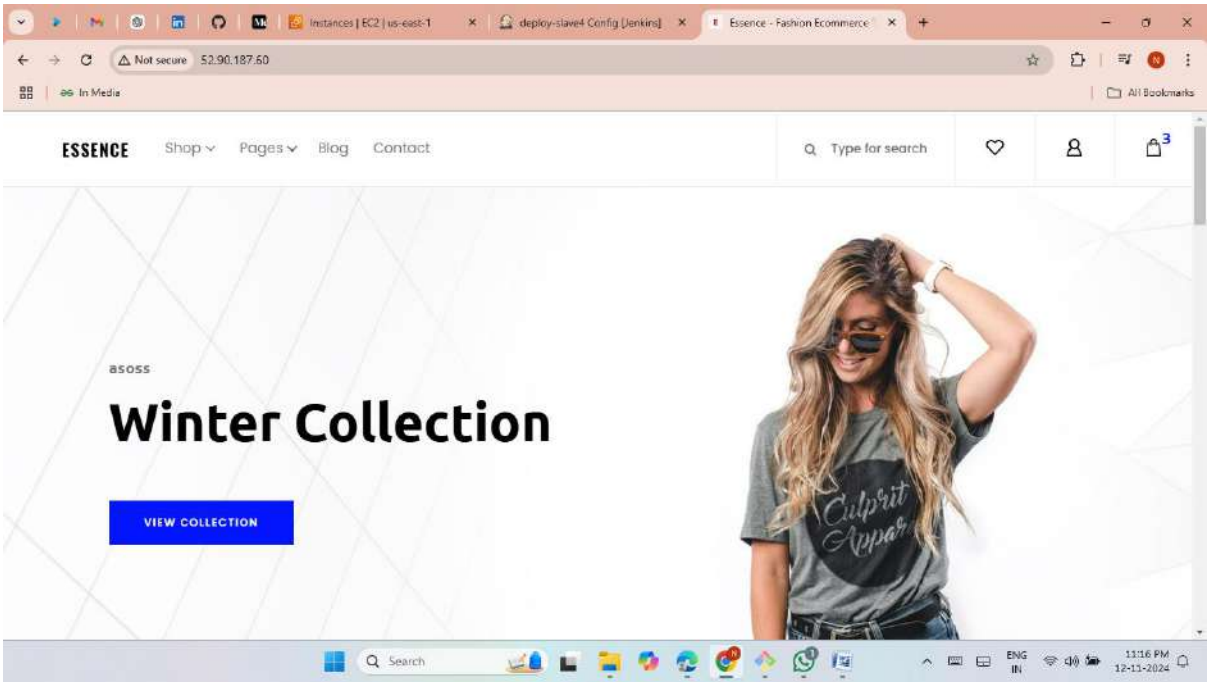
sudo mv /home/ec2-user/workspace/deploy-slave5/* /var/www/html



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



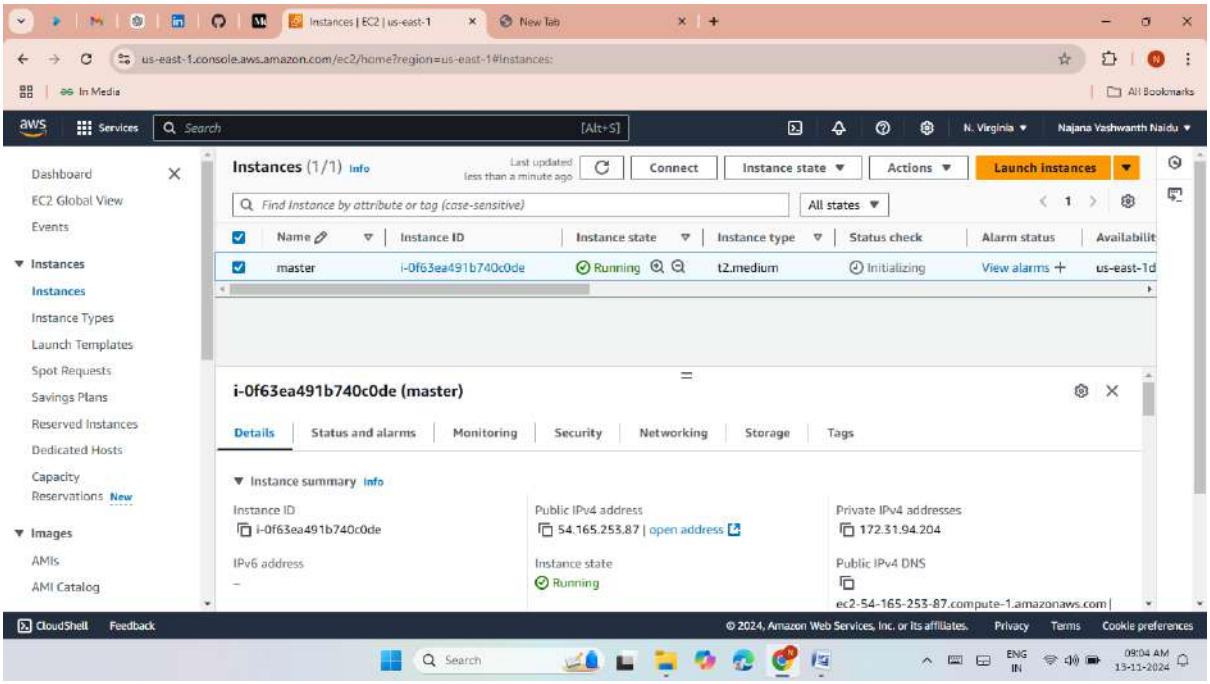
- Now copy the slave5 public ip and past it on Google browser.
- The ecomm web application was hosted successfully.



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.



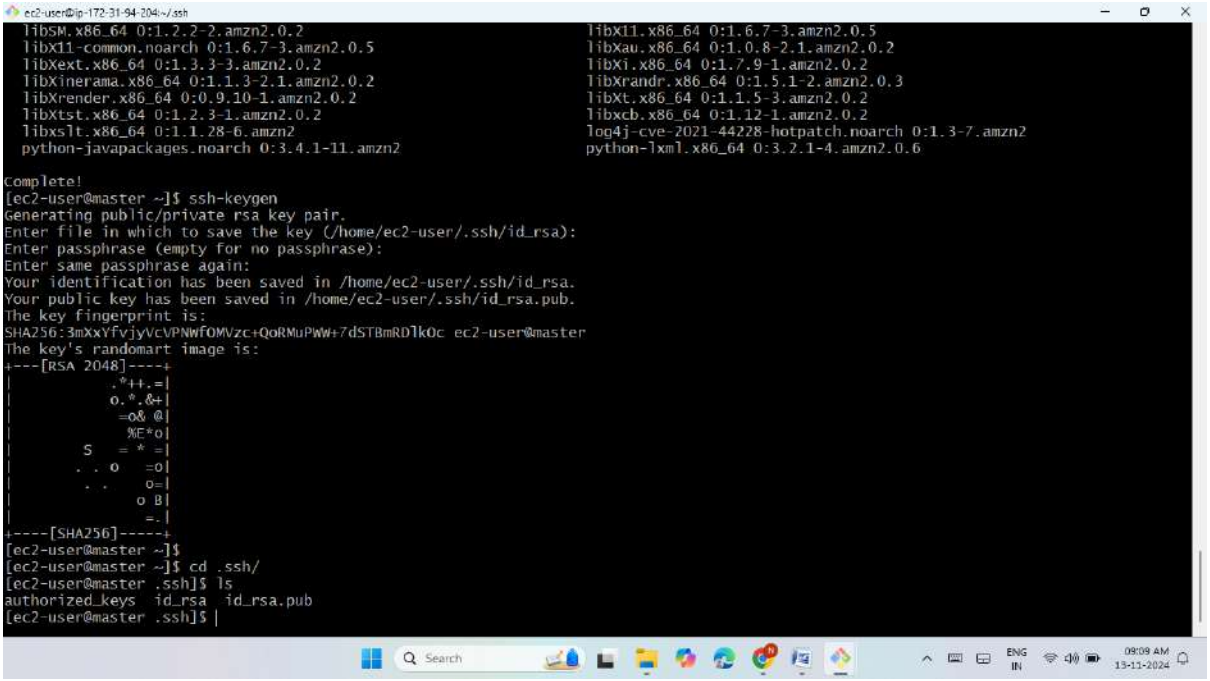
- Now install java and git.
- Now install Jenkins then start and enable the Jenkins.
- We can change the hostname with the commands

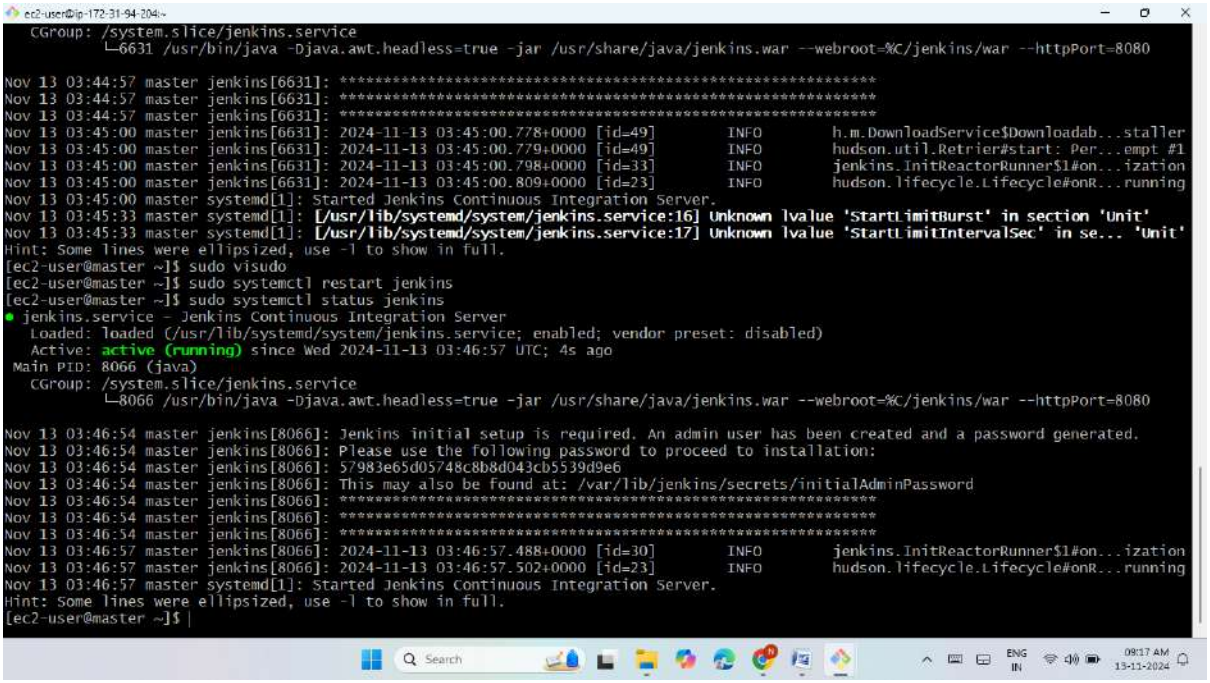
  Sudo hostname master

  Exec bash

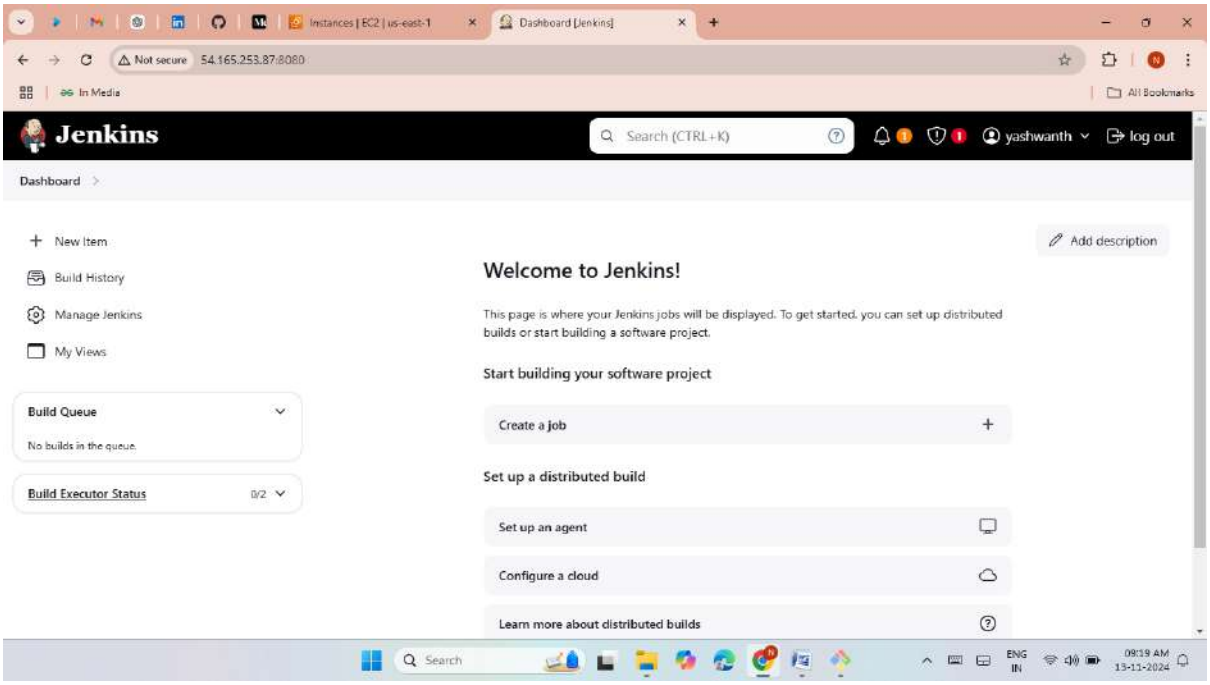
- Now connect to the Master server and then generate keys with the command.

  ssh-keygen

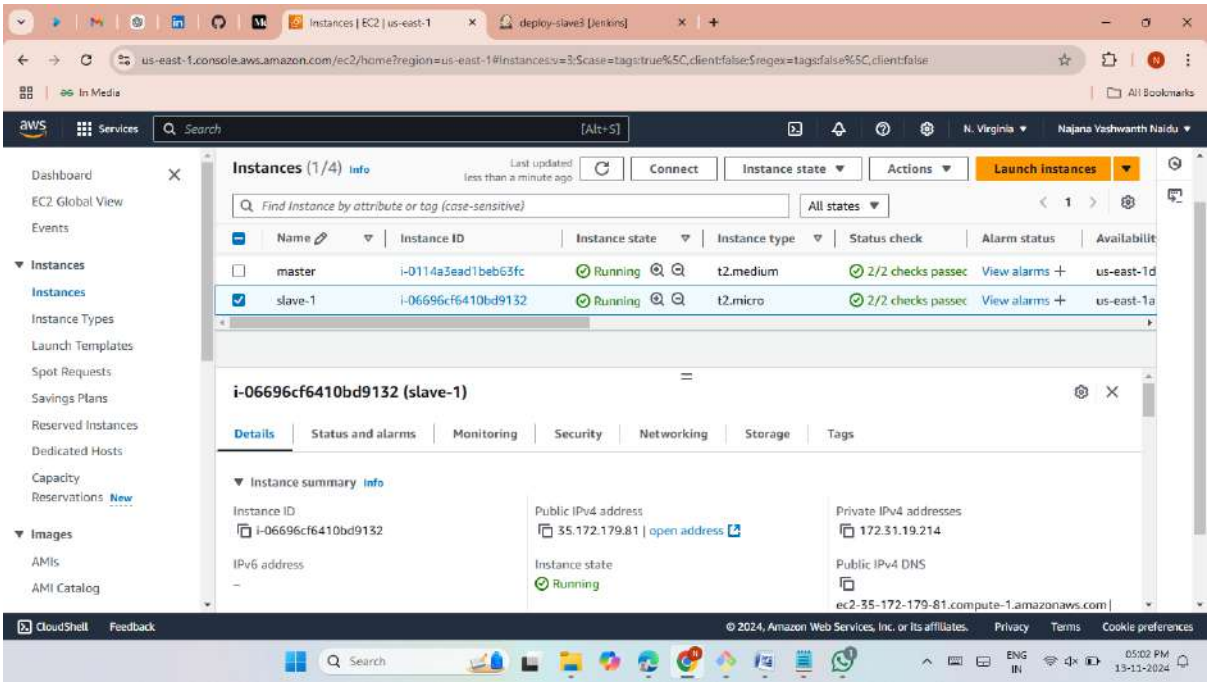

ec2-user@ip-172-31-94-204:~/.ssh
libSM.x86_64 0:1.2.2-2.amzn2.0.2
libX11.x86_64 0:1.6.7-3.amzn2.0.5
libX11-common.noarch 0:1.6.7-3.amzn2.0.5
libXau.x86_64 0:1.0.8-2.1.amzn2.0.2
libXext.x86_64 0:1.3.3-3.amzn2.0.2
libXi.x86_64 0:1.7.9-1.amzn2.0.2
libXinerama.x86_64 0:1.1.3-2.1.amzn2.0.2
libXrandr.x86_64 0:1.5.1-2.amzn2.0.3
libXrender.x86_64 0:0.9.10-1.amzn2.0.2
libXt.x86_64 0:1.1.5-3.amzn2.0.2
libXtst.x86_64 0:1.2.3-1.amzn2.0.2
libxcb.x86_64 0:1.12-1.amzn2.0.2
libxslt.x86_64 0:1.1.28-6.amzn2
log4j-cve-2021-44228-hotpatch.noarch 0:1.3-7.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:
SHA256:3mXxYfvjyVcVPNWfOMVzc+QoRMuPWW+7dSTBmRDlkOc ec2-user@master
The key's randomart image is:
+---[RSA 2048]----+
| .*++.=|
| o.*.&+|
| =o& @|
| %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]$
Search
ENG
IN
09:09 AM
13-11-2024


ec2-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$Downloadab...staller
Nov 13 03:45:00 master jenkins[6631]: 2024-11-13 03:45:00.779+0000 [id=49] INFO hudson.util.Retrier#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]: 57983e65d05748c8b8d043cb5539d9e6
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]: **
Nov 13 03:46:57 master jenkins[8066]: 2024-11-13 03:46:57.488+0000 [id=30] INFO jenkins.InitReactorRunner$1#on...ization
Nov 13 03:46:57 master jenkins[8066]: 2024-11-13 03:46:57.502+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onR...running
Nov 13 03:46:57 master systemd[1]: Started Jenkins Continuous Integration Server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@master ~]$
Search
ENG
IN
09:17 AM
13-11-2024

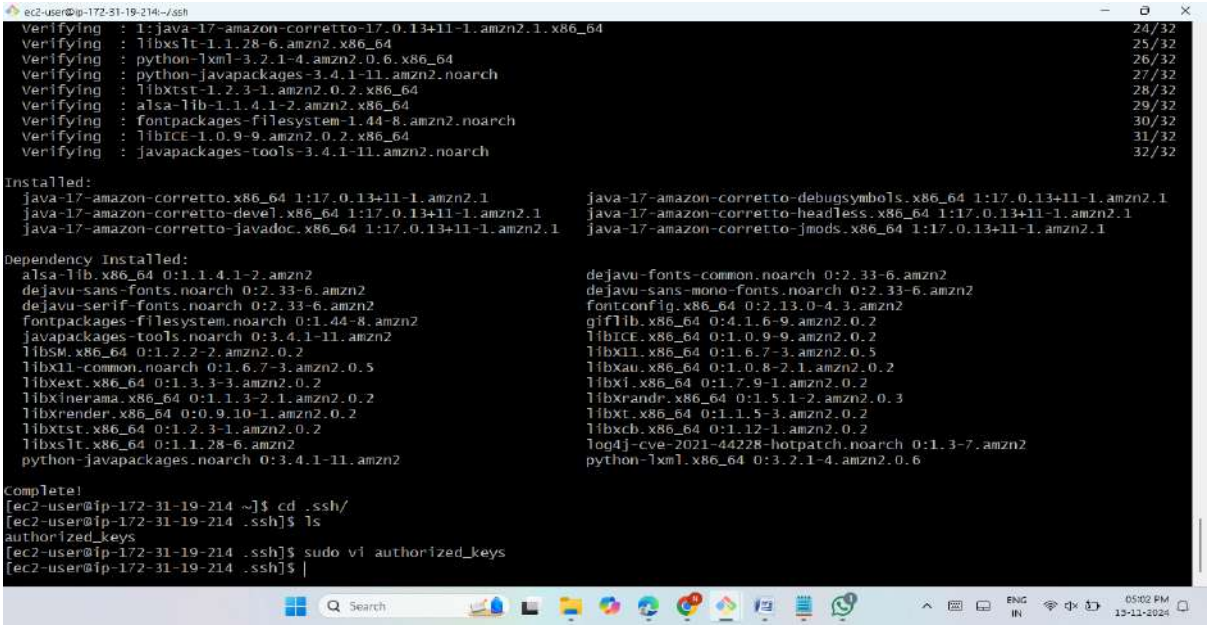
- Here the Jenkins was launched successfully.



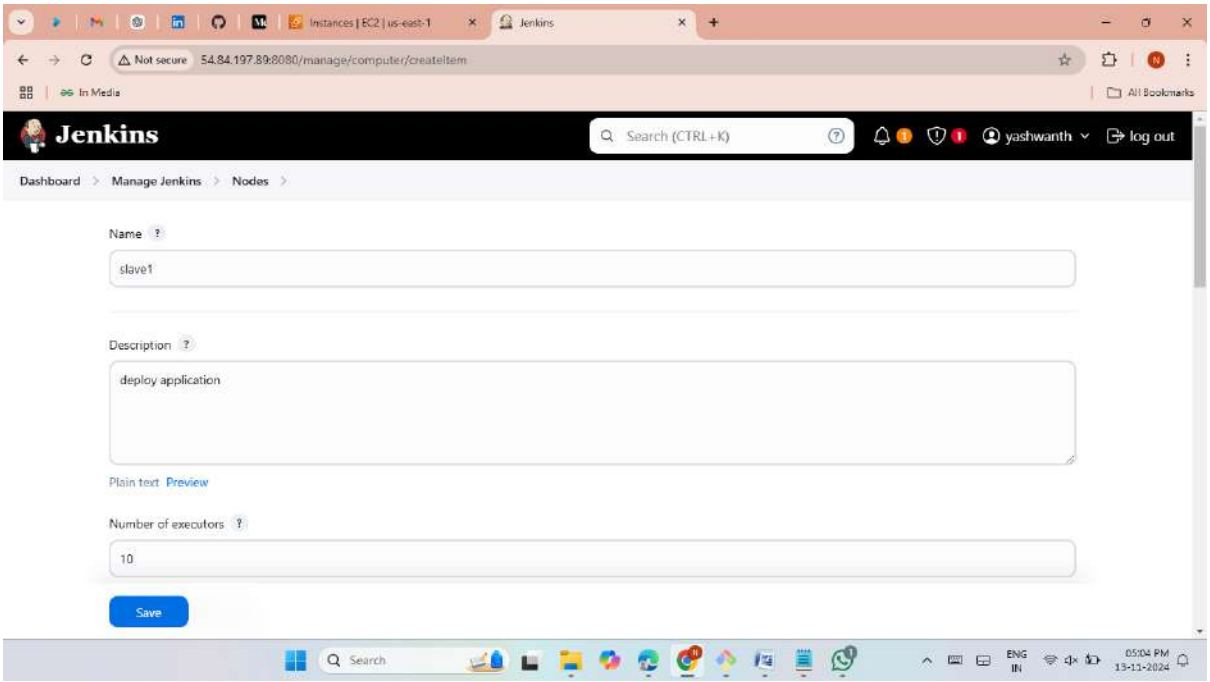
- Now launch an Ec2 instances and give name as slave1.

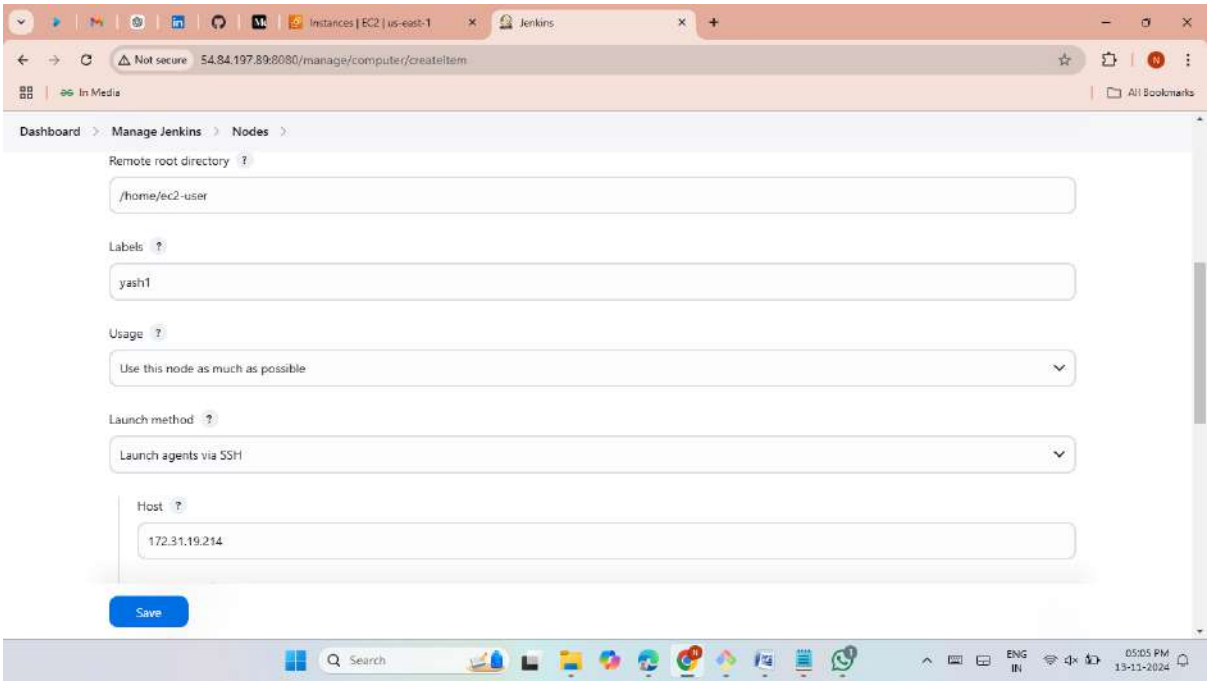


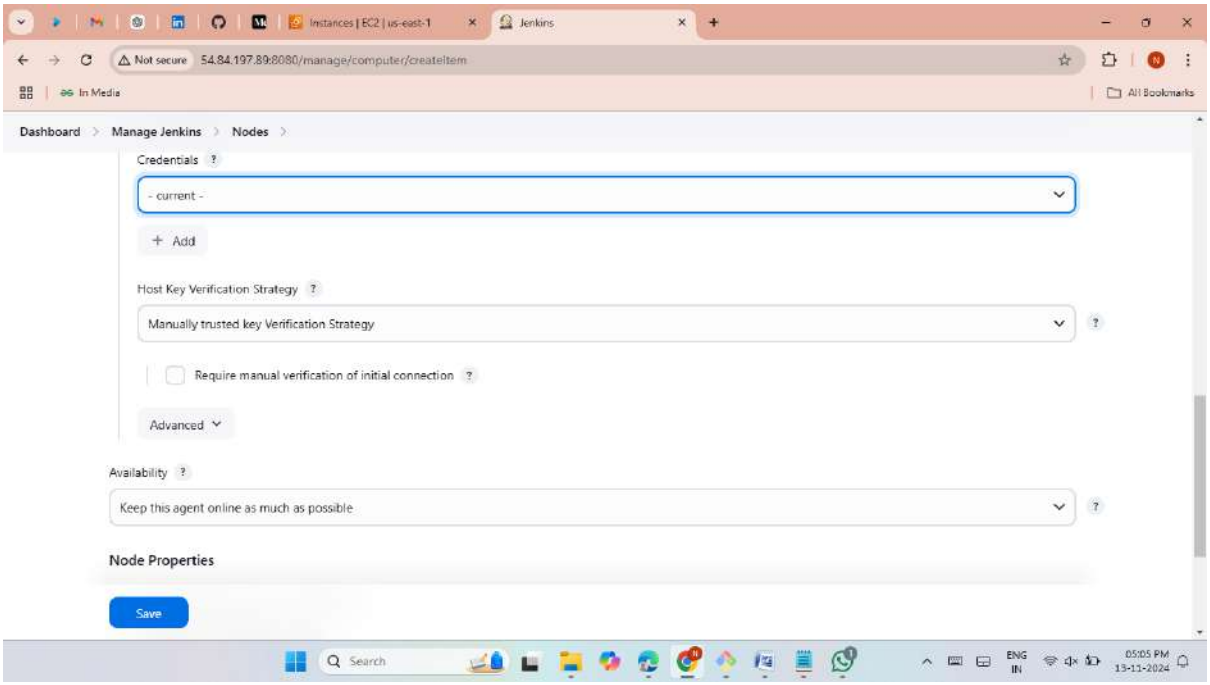
- Now connect Slave1 then install java.
- Now give private keys from master server.



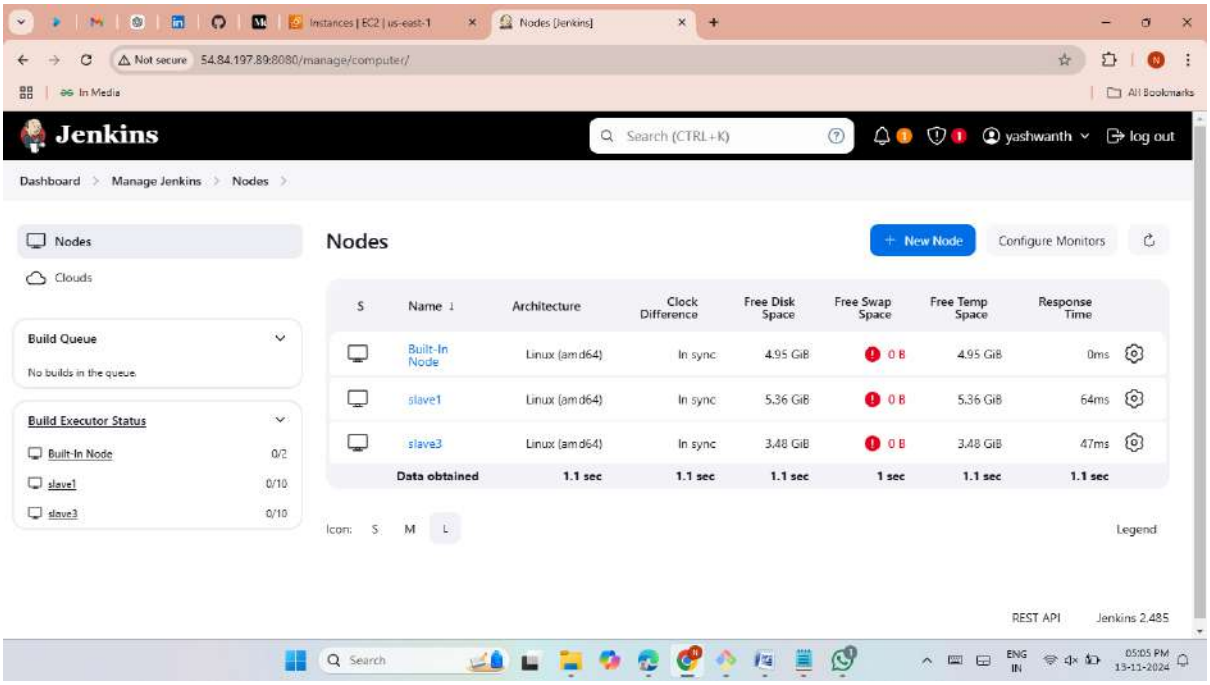
- Now create a node for slave1 server.
- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.



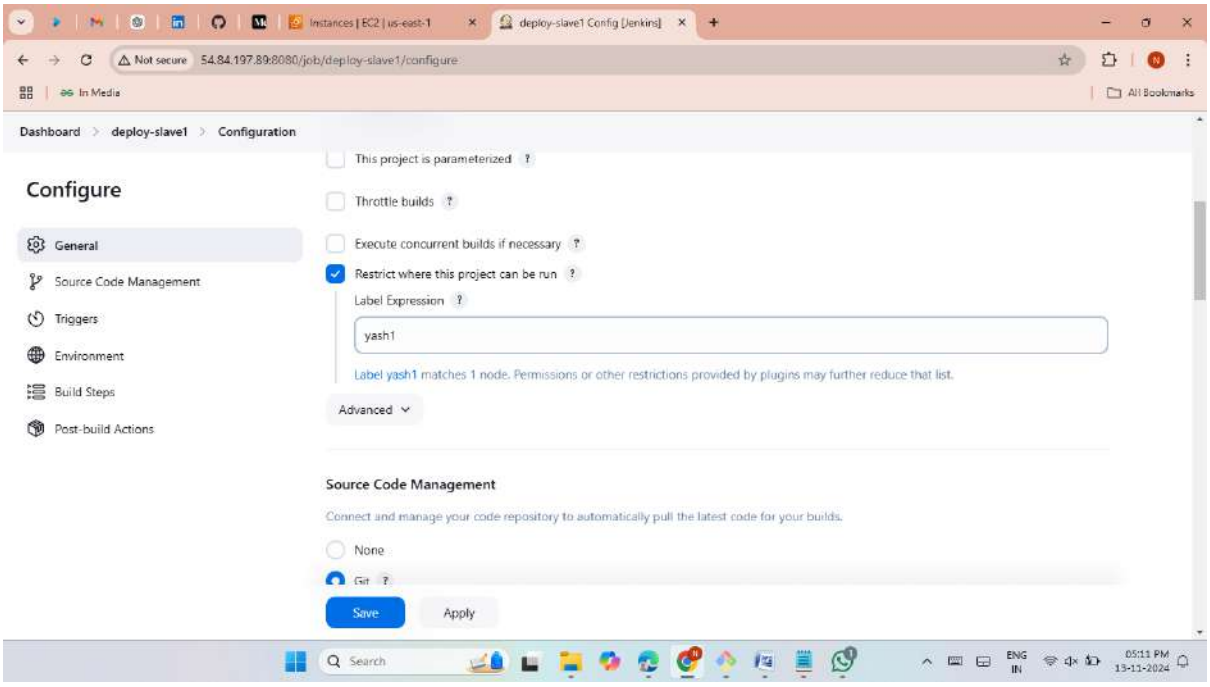

Instances | EC2 | us-east-1
Jenkins
Not secure 54.84.197.89:8080/manage/computer/createItem
Dashboard > Manage Jenkins > Nodes >
Remote root directory
/home/ec2-user
Labels
yash1
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
172.31.19.214
Save


Instances | EC2 | us-east-1
Jenkins
Not secure 54.84.197.89:8080/manage/computer/createItem
Dashboard > Manage Jenkins > Nodes >
Credentials
- current -
+ Add
Host Key Verification Strategy
Manually trusted key Verification Strategy
Require manual verification of initial connection
Advanced
Availability
Keep this agent online as much as possible
Node Properties
Save

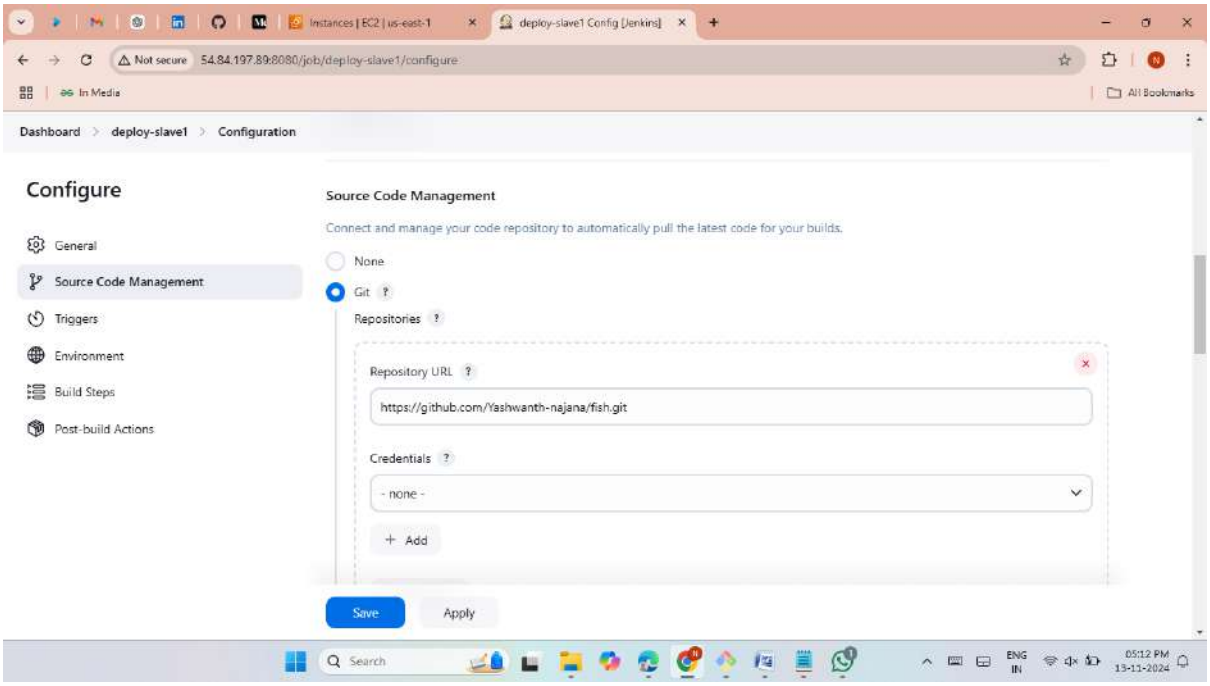
- Here the slave1 node is successfully created.



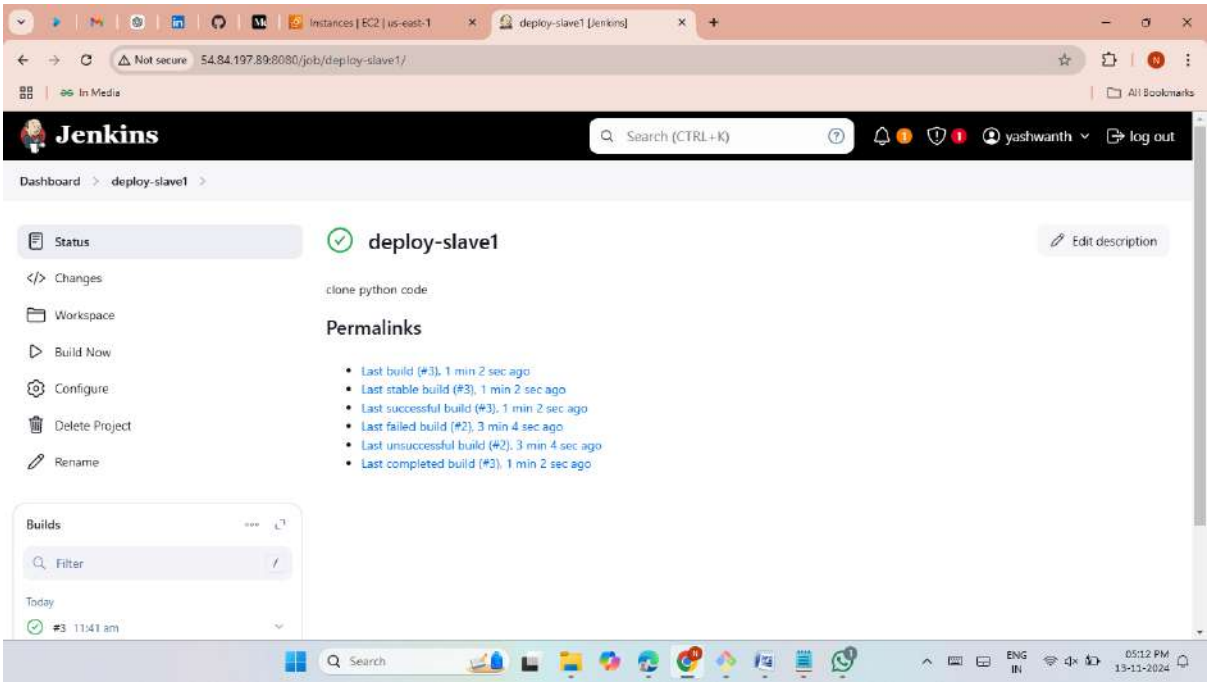
- Now create a job and give slave1 label.



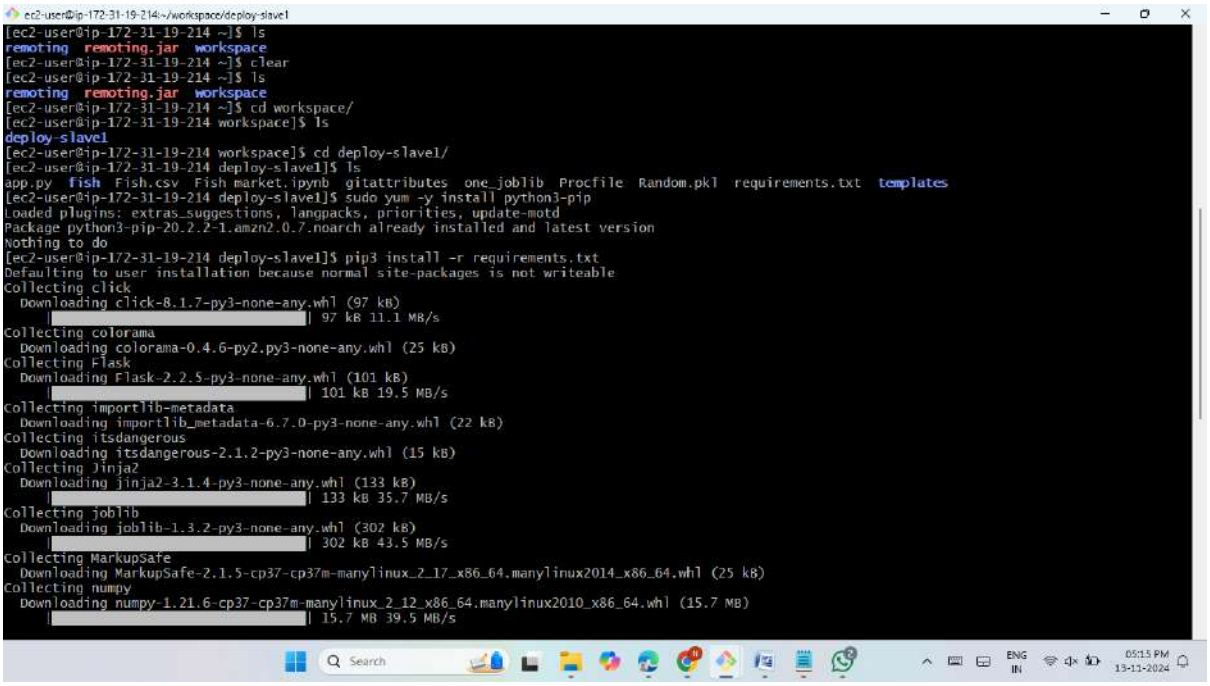
- Now clone the python application repo form github.
- Click on save.

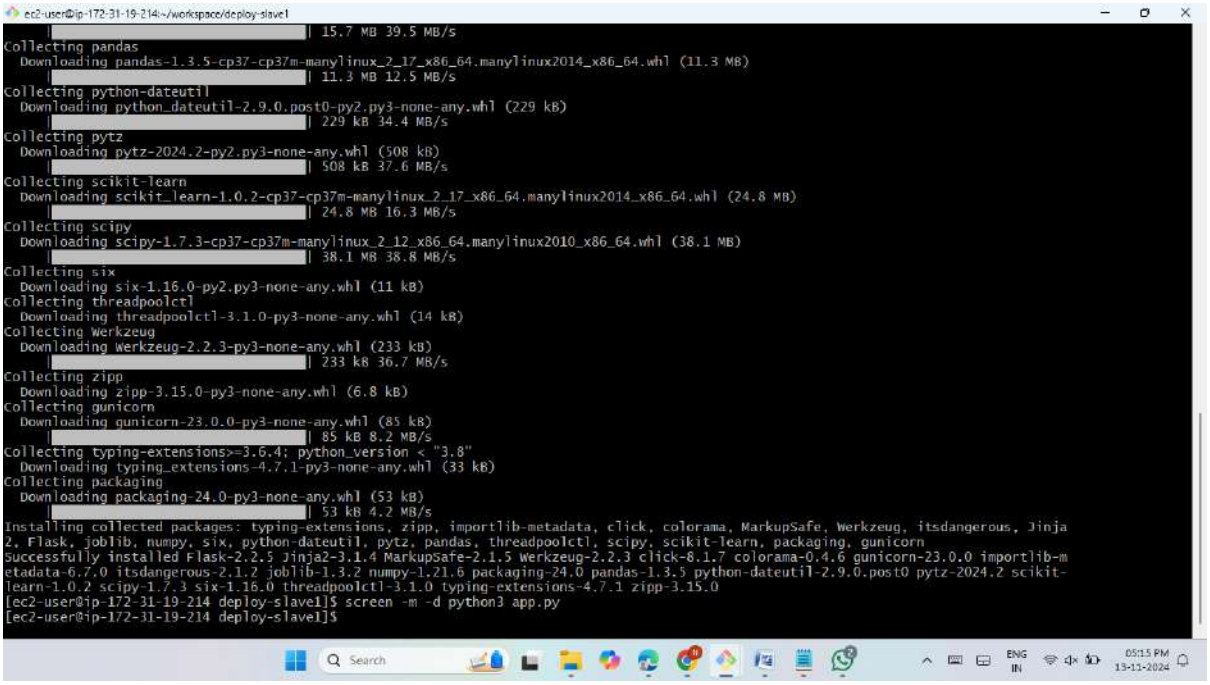


- Now click on build now.
- Here the job was success.

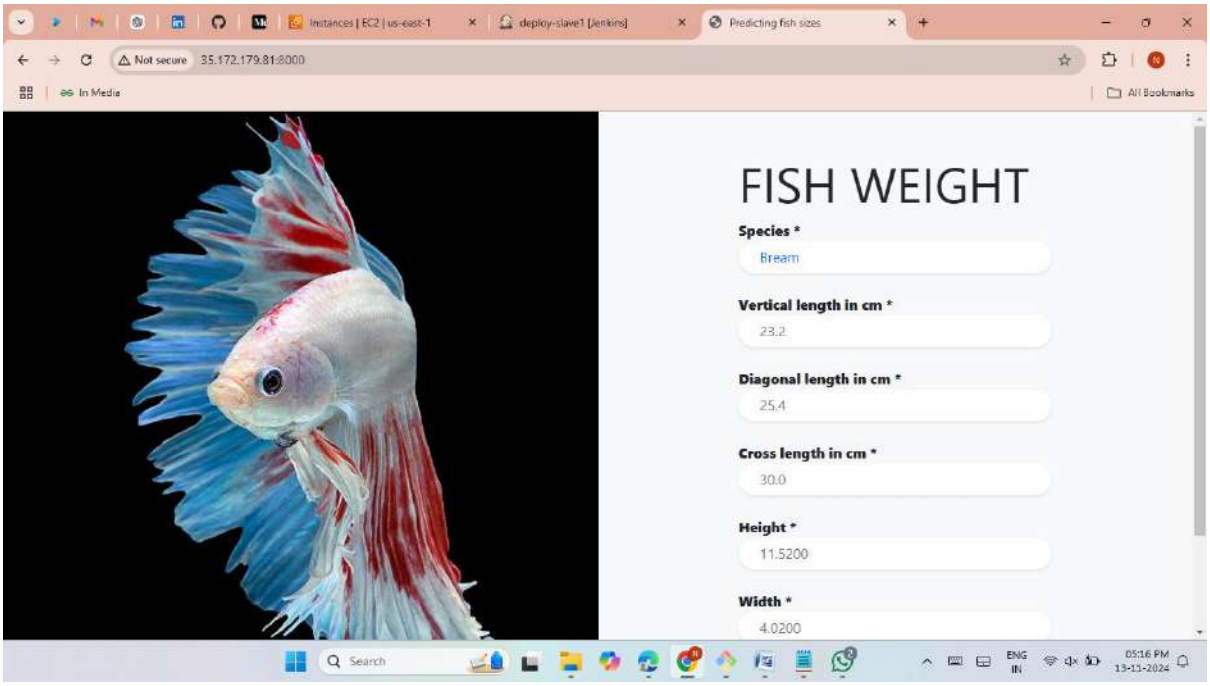


- Now run the python applications with the commands.
- Pip3 install -r requirements.txt
- Screen -m -d python3 app.py

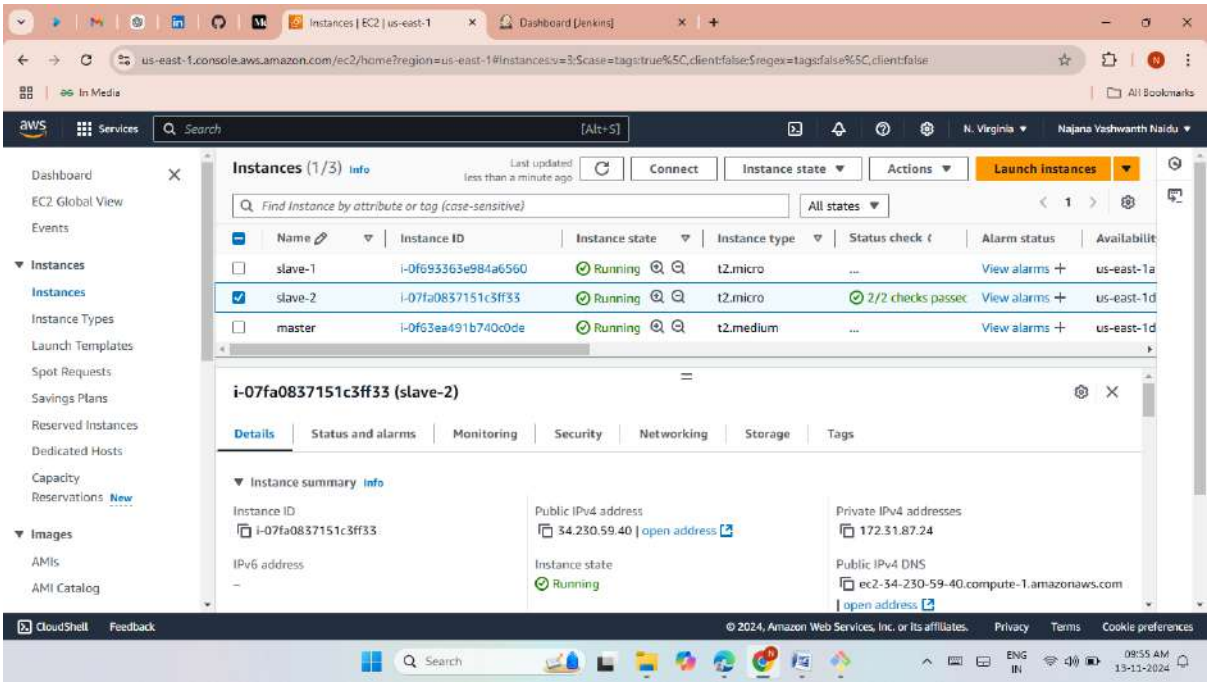




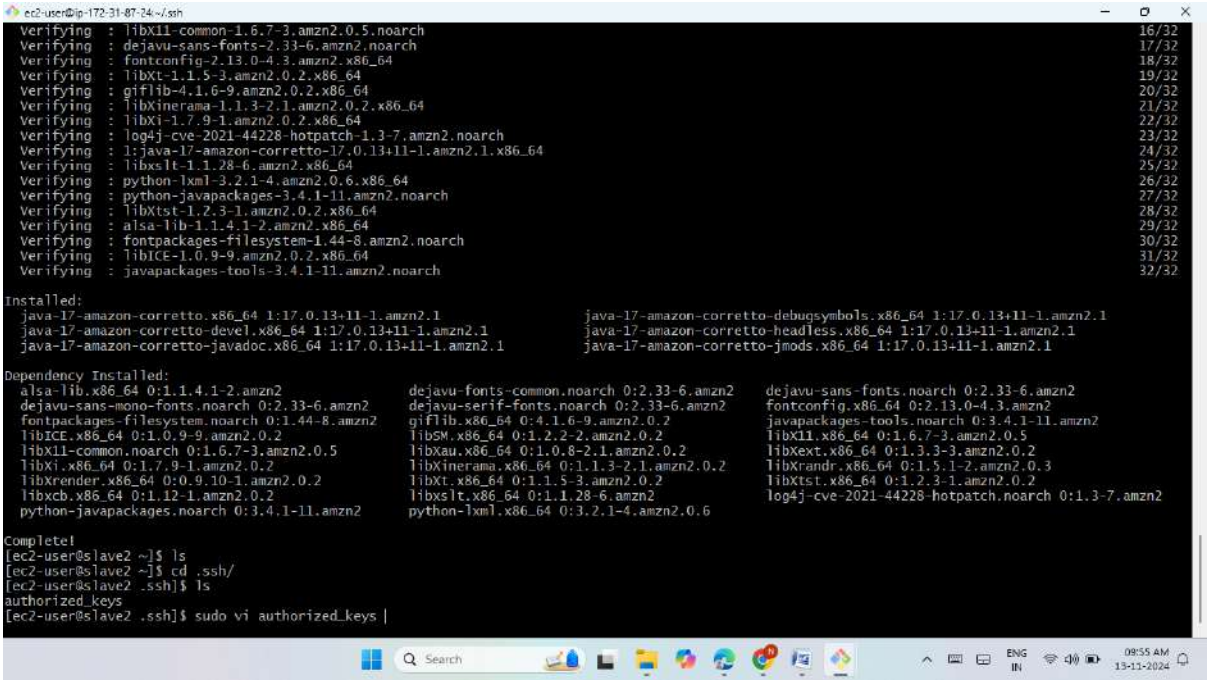
- Now copy the slave1 public ip and past it on google browser.
- The python application was hosted successfully.



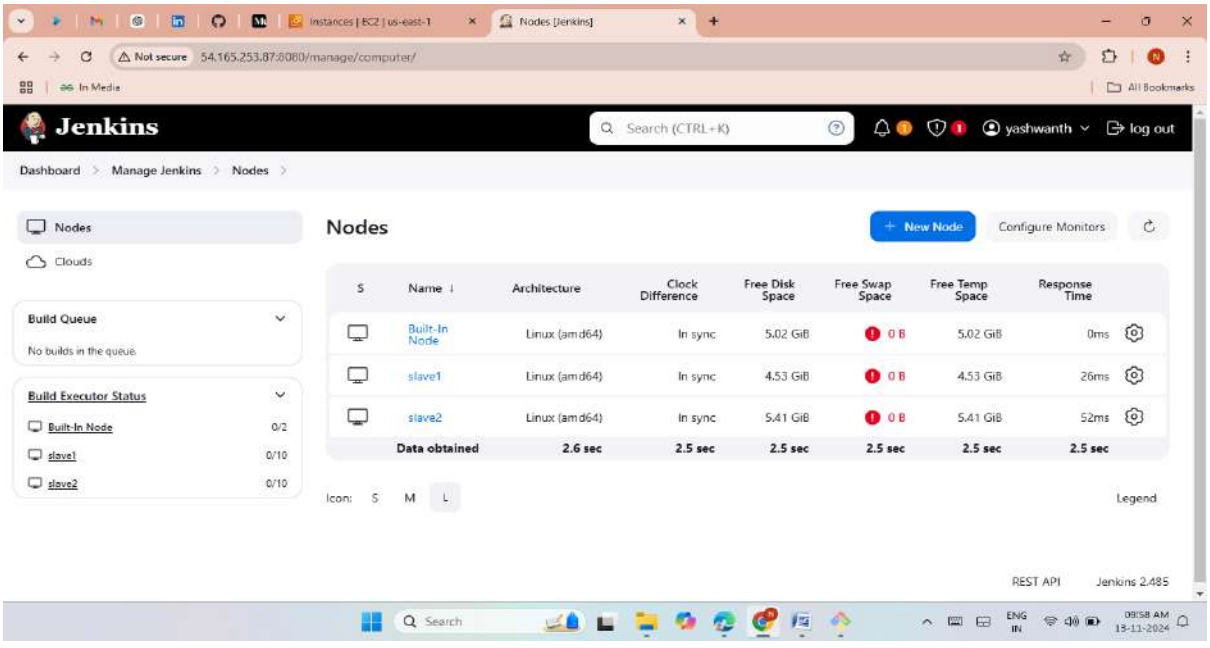
- Now launch an ec2 instances and give name as slave2.



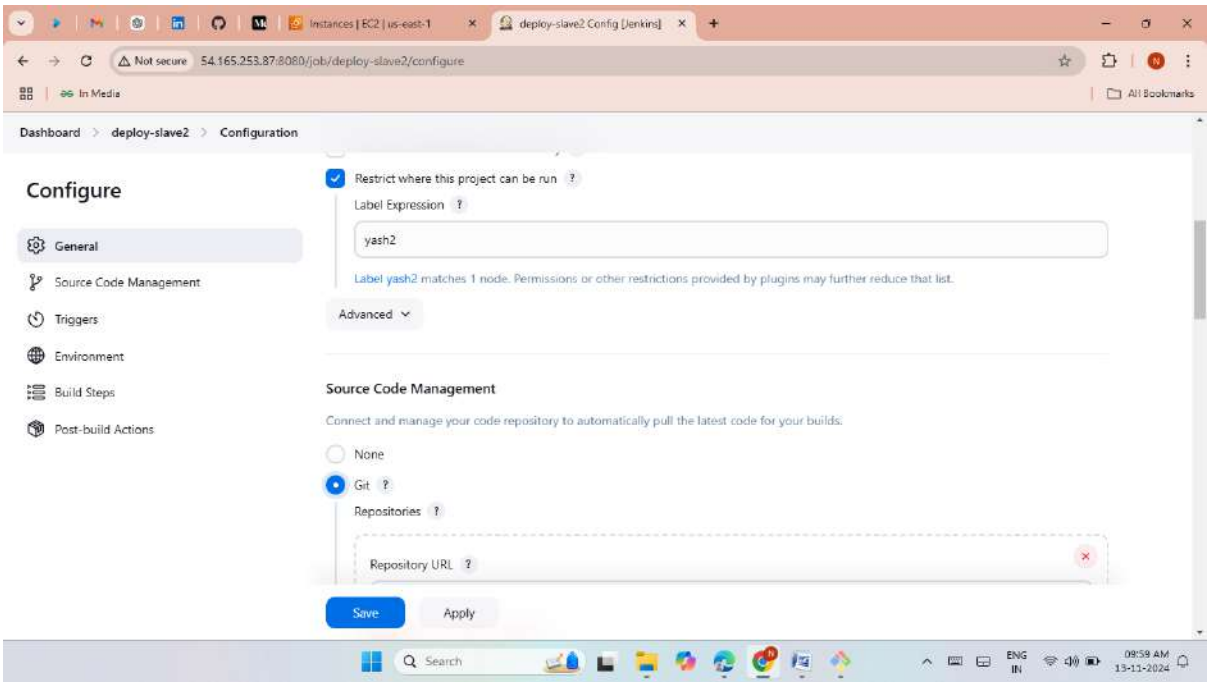
- Now connect Slave2 then install java.
- Now give private keys from master server.



- 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.



- 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

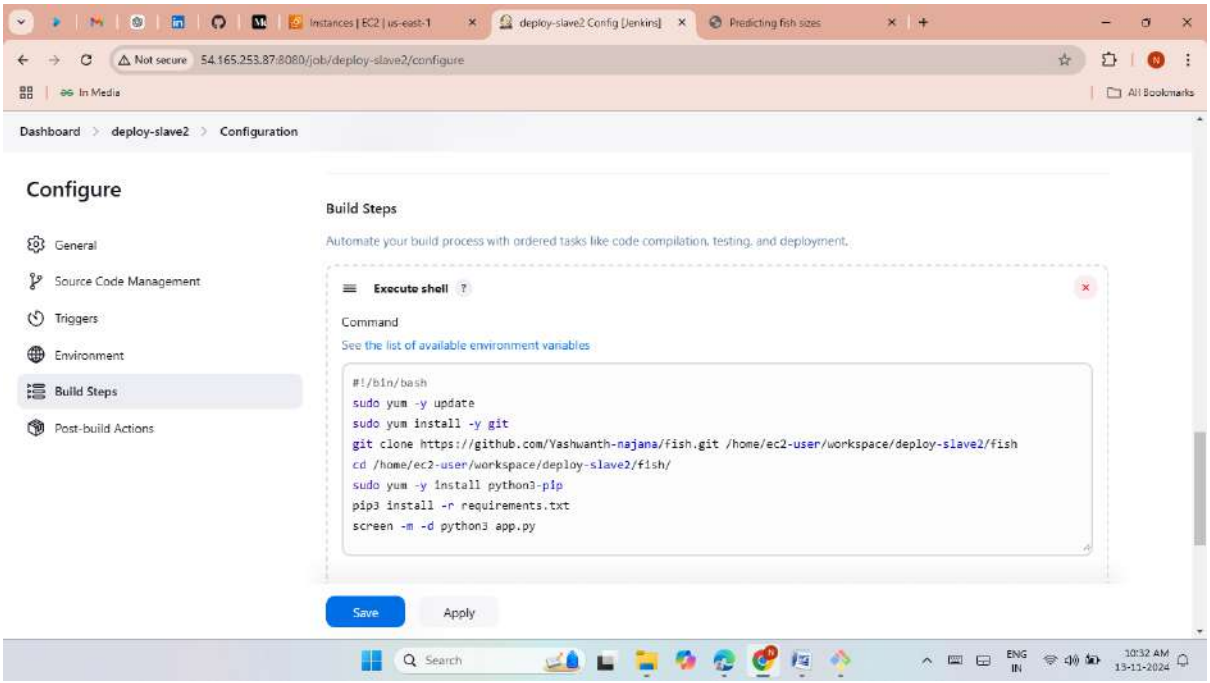
git clone https://github.com/Yashwanth-najana/fish.git /home/ec2-user/workspace/deploy-slave2/fish

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

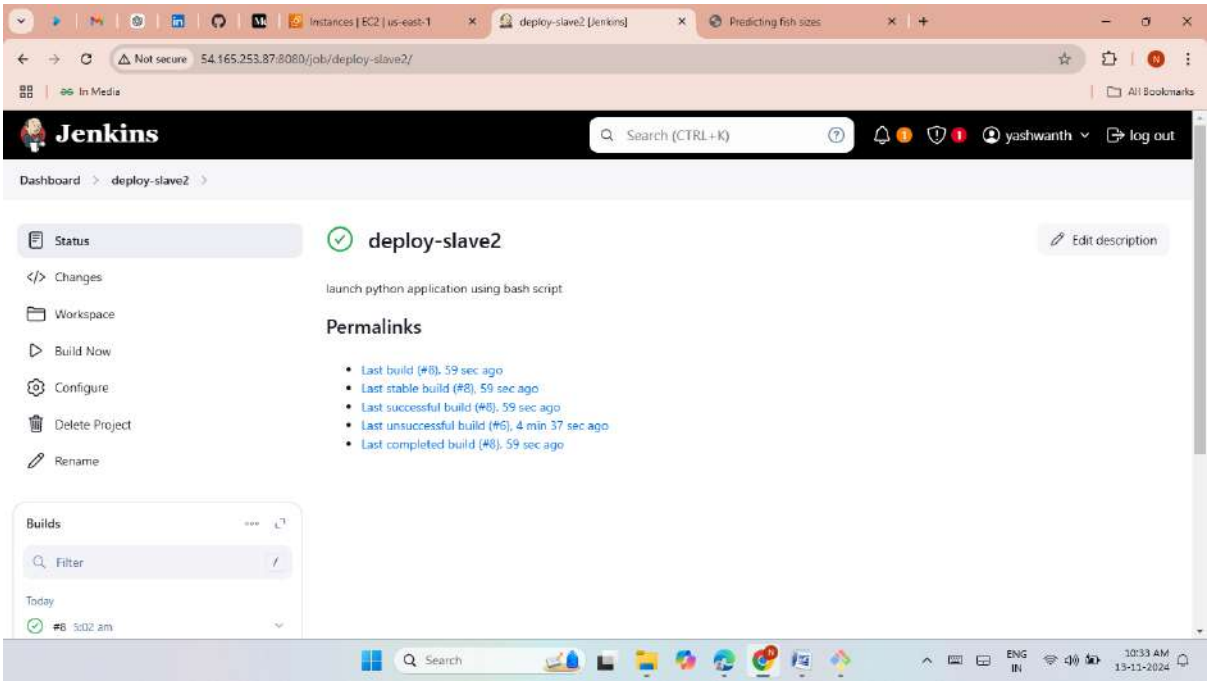
sudo yum -y install python3-pip

pip3 install -r requirements.txt

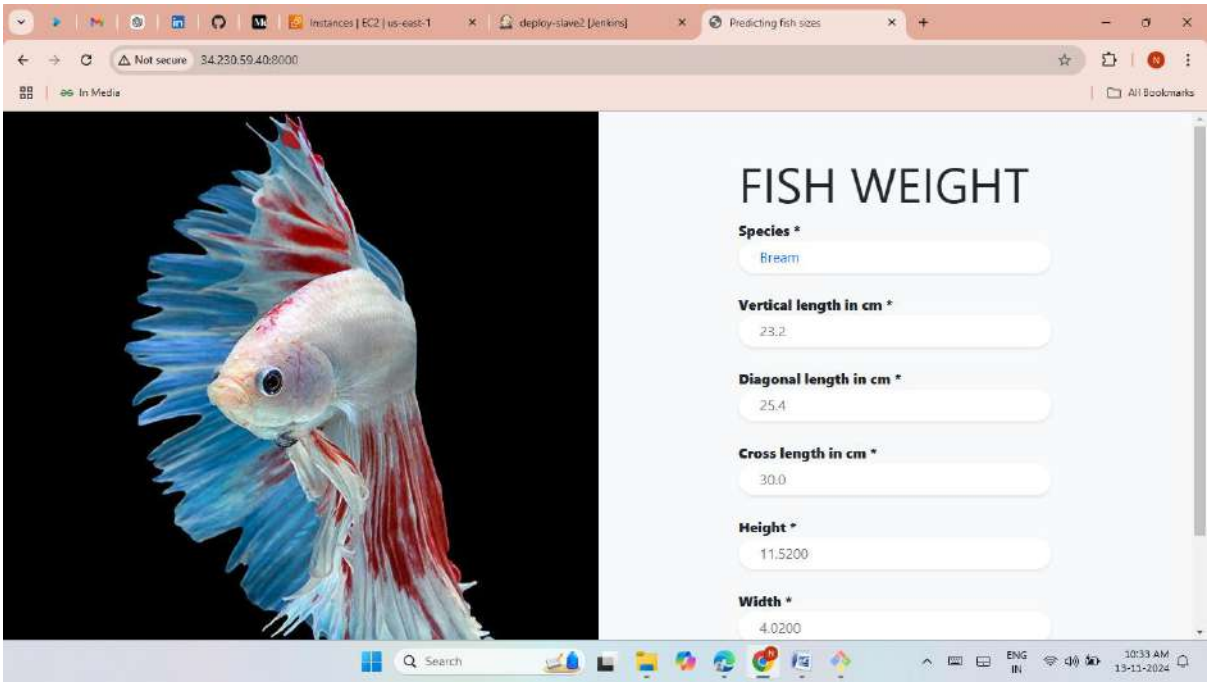
screen -m -d python3 app.py



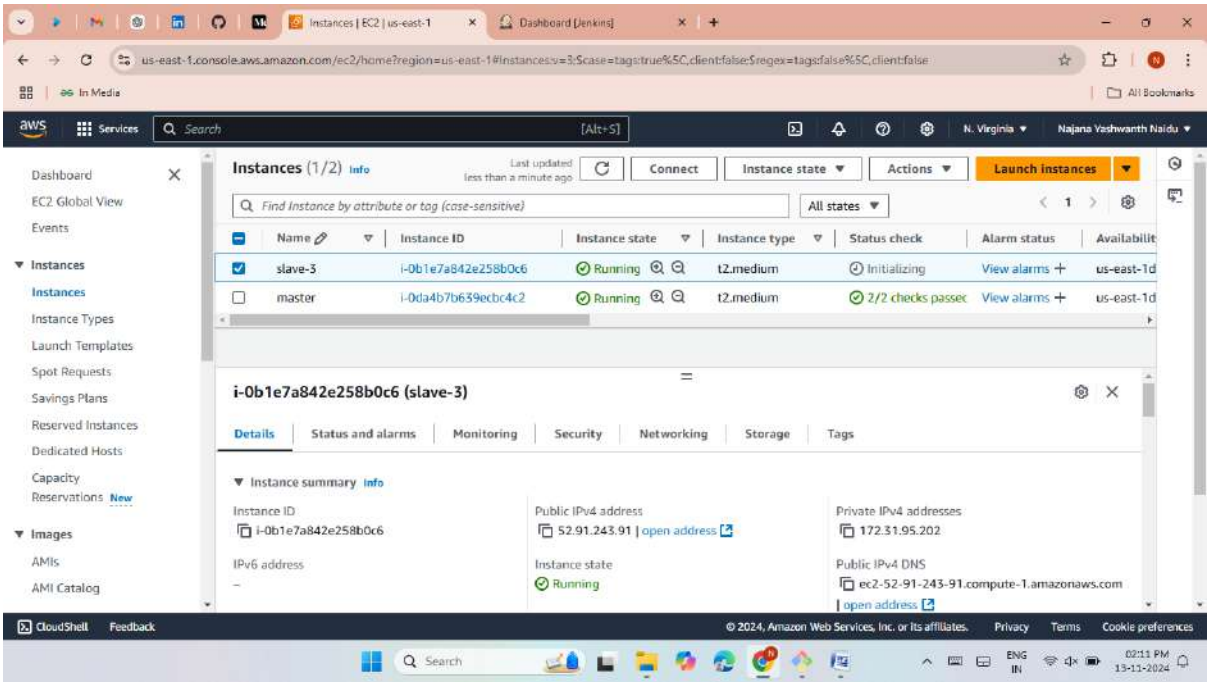
- Now click on build now.
- Here the job was success.



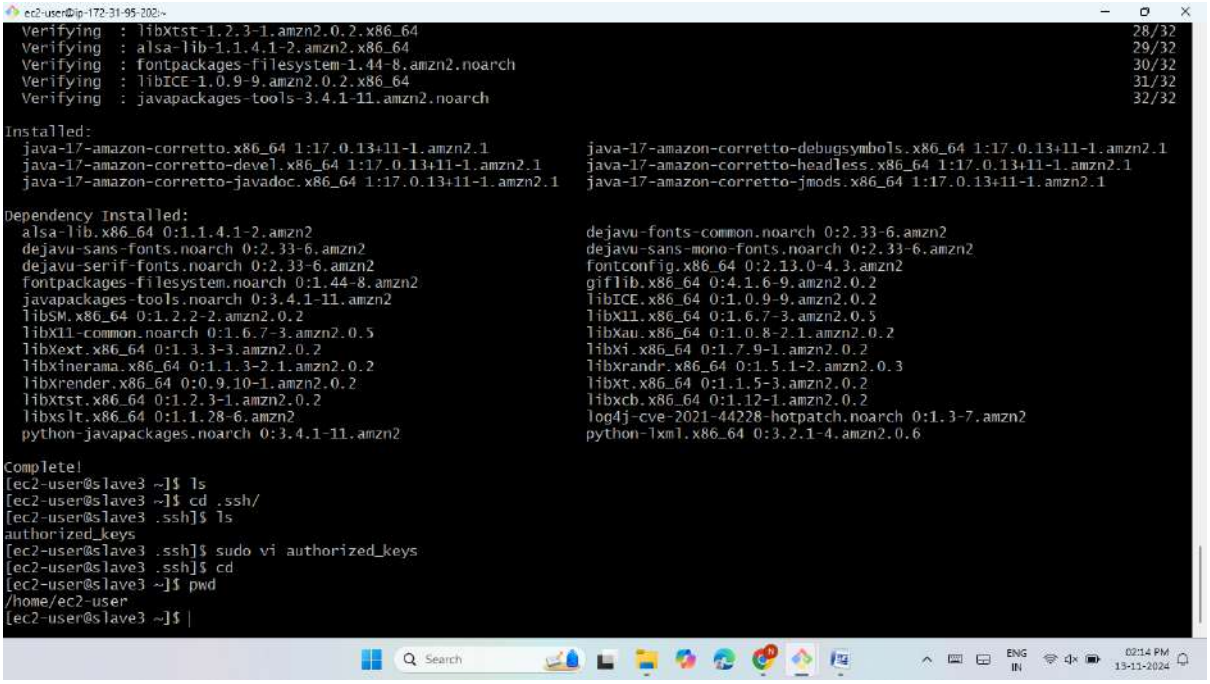
- Now copy the slave2 public ip and past it on google browser.
- The python web application was hosted successfully.



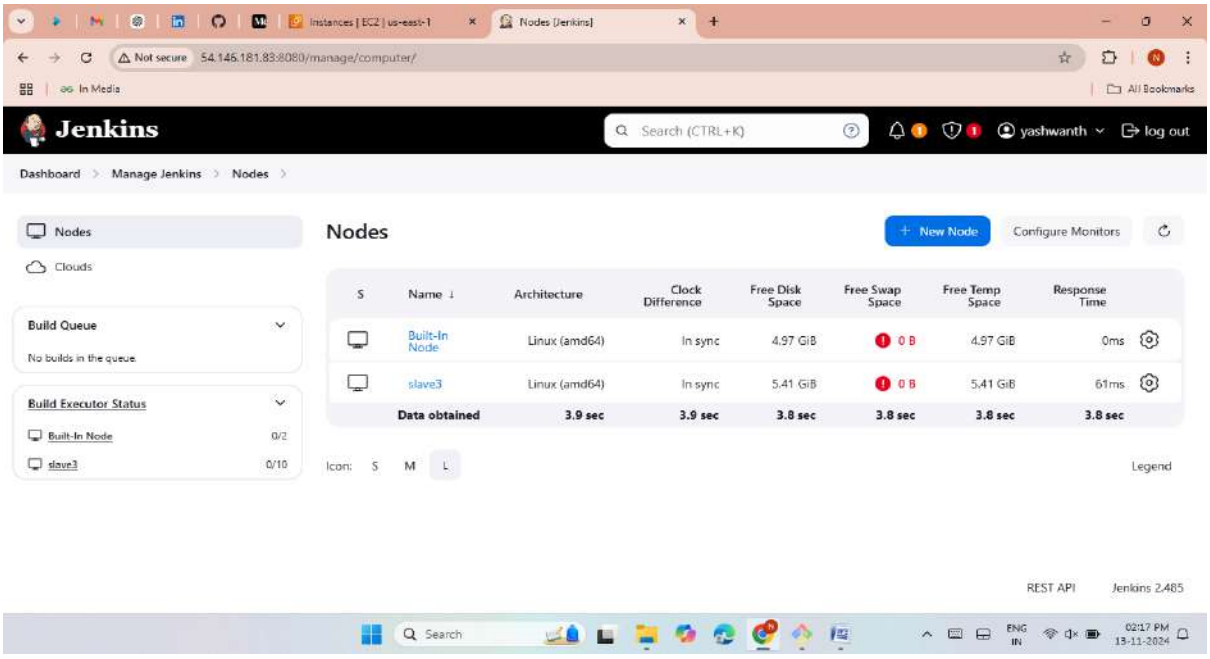
- Now launch an ec2 instances and give name as slave3.



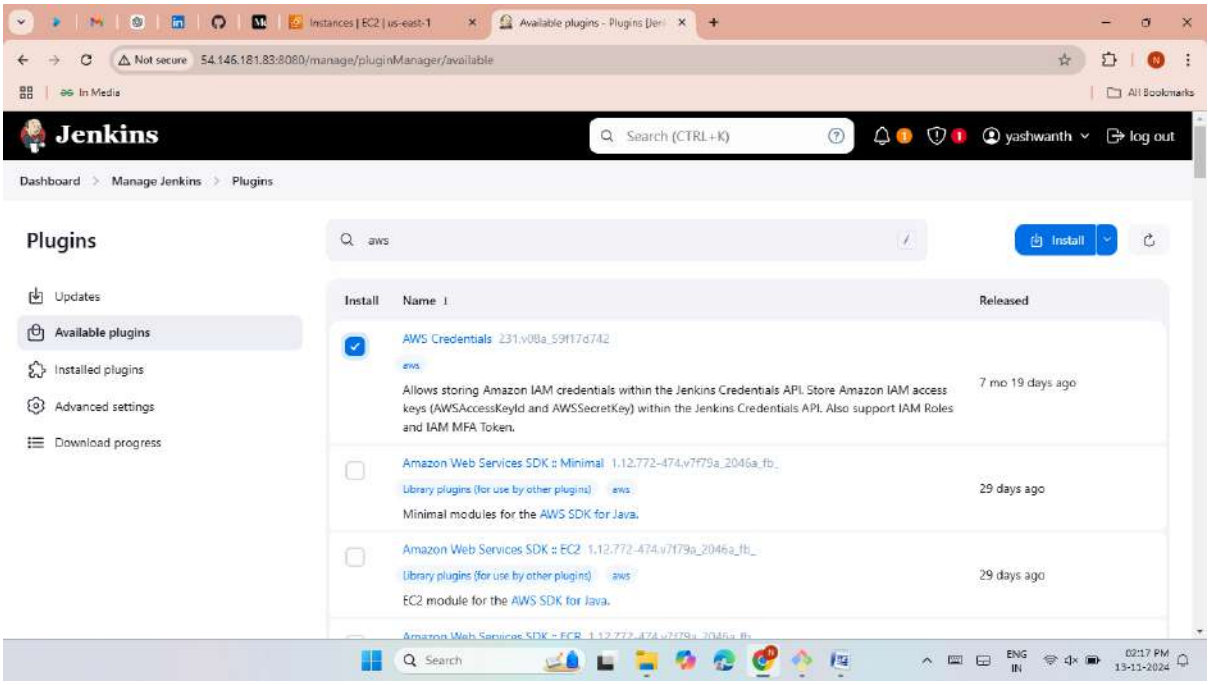
- Now connect Slave3 then install java.
- Now give private keys from master server.



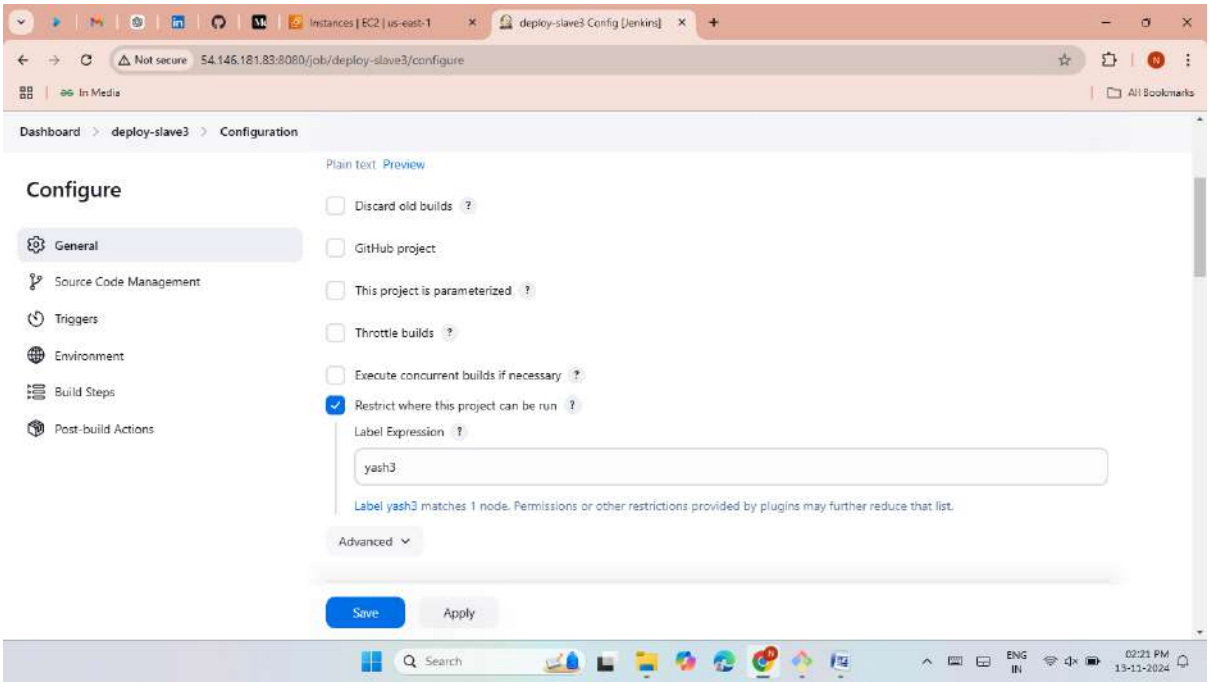
- Now create a node for slave3 server.
- Give details for slave node.
- Here we have to give slave3 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave3 node is successfully created.



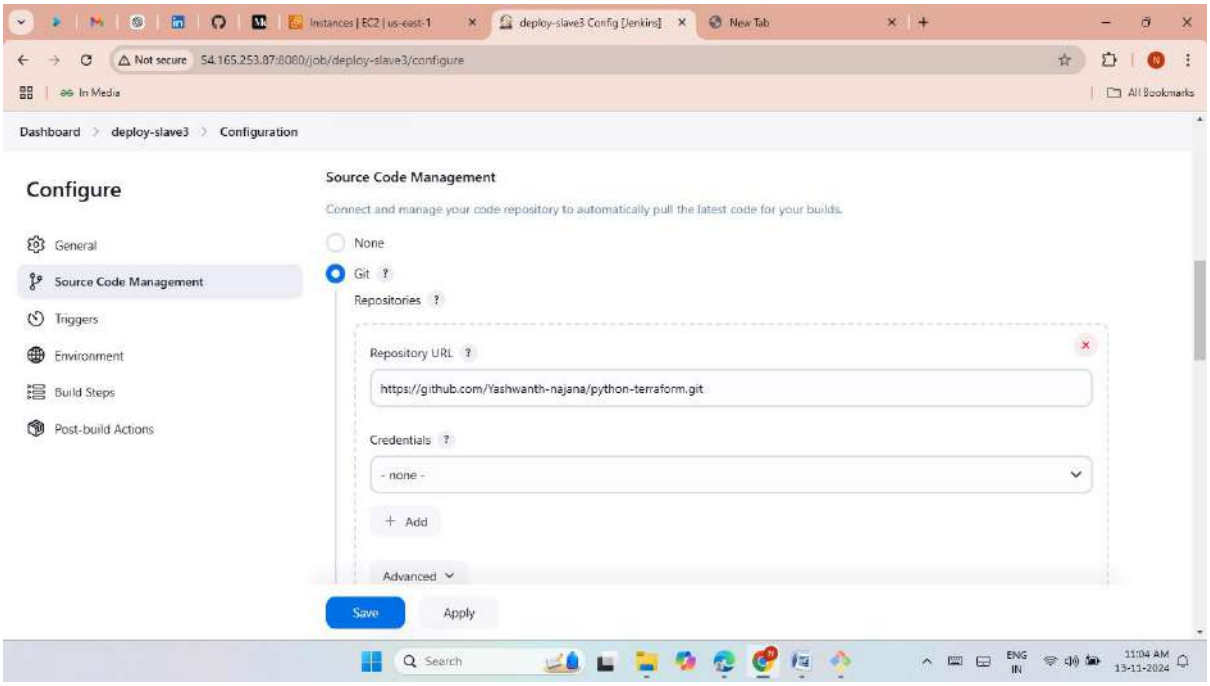
- Now install the AWS credentials plugin for give access key and secret key to the job.



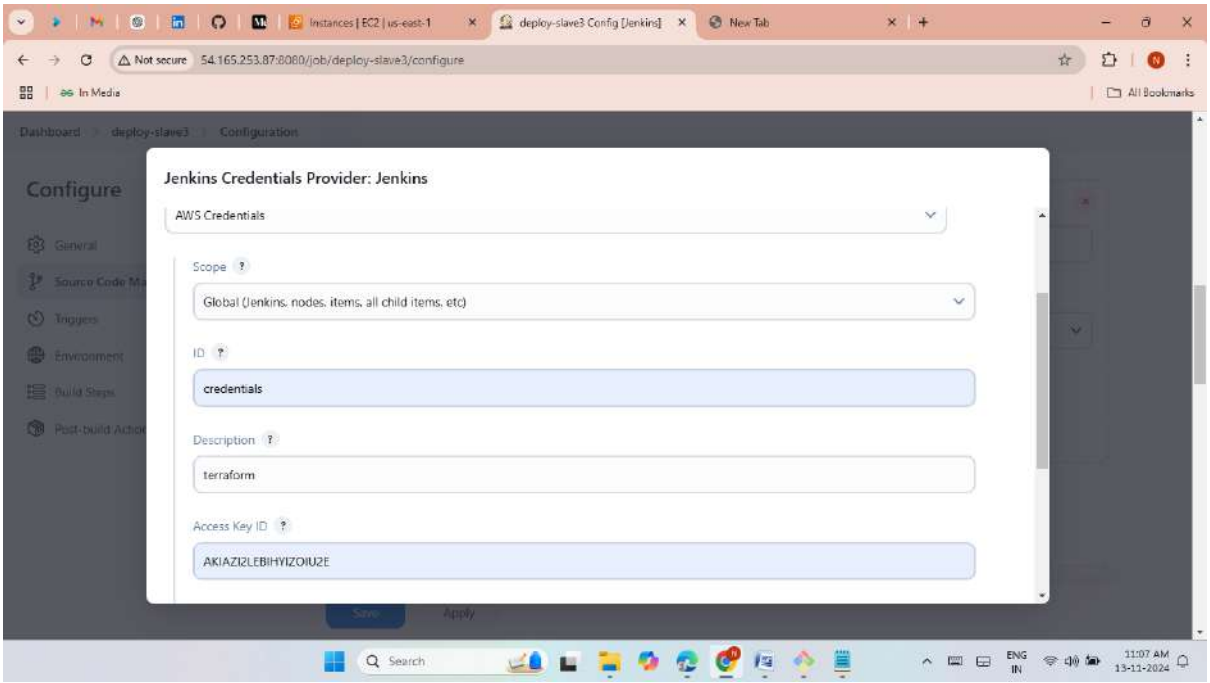
- Now create a job and give slave3 label.



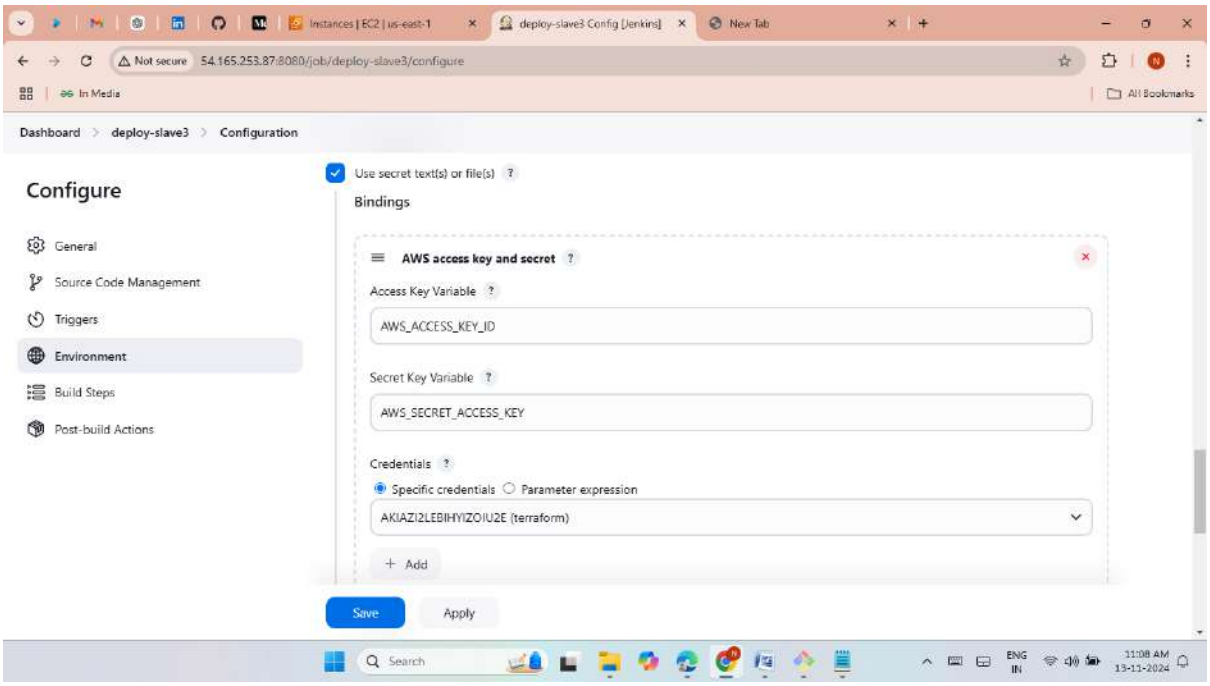
- Now clone the Terraform script for launch EC2 instance along with the python application then it creates VPC, Subnets, IGW, Route Table repo form github.



- 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

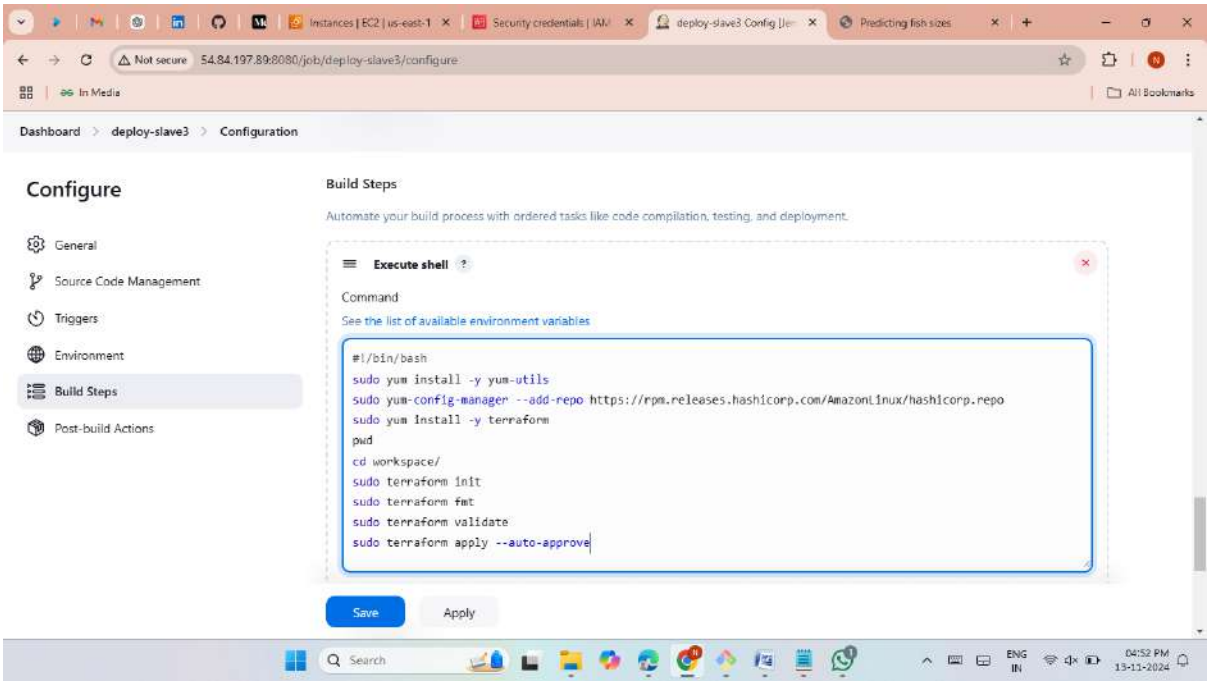
cd workspace/

sudo terraform init

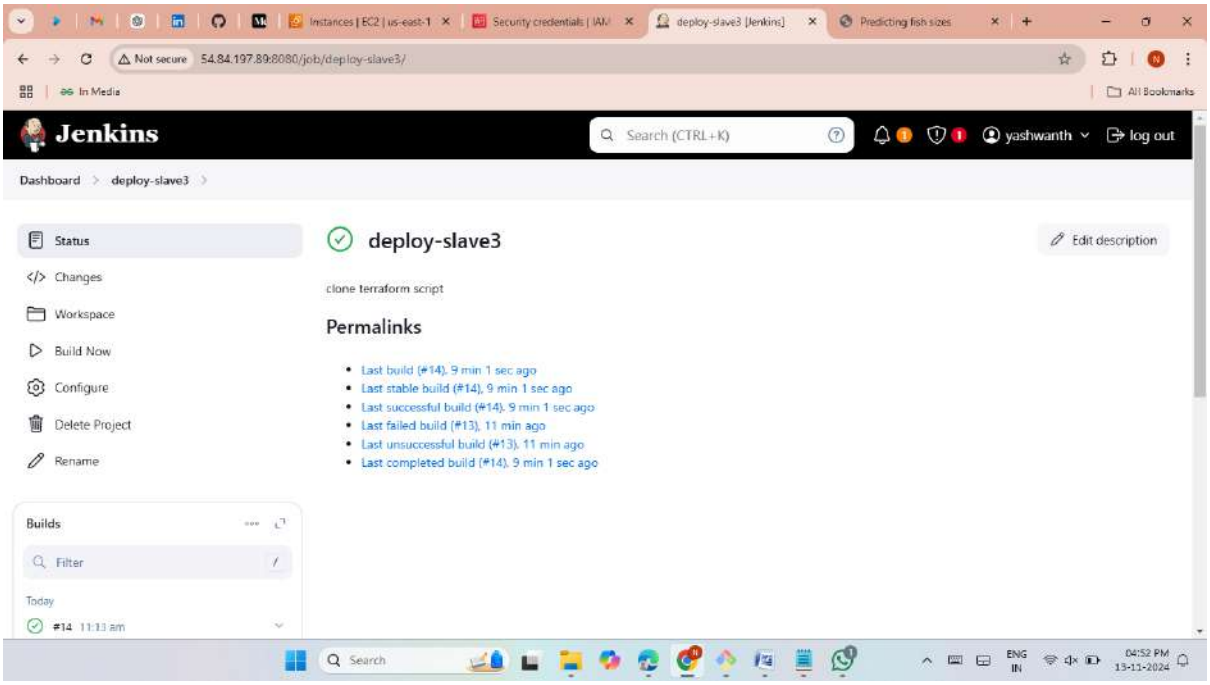
sudo terraform fmt

sudo terraform validate

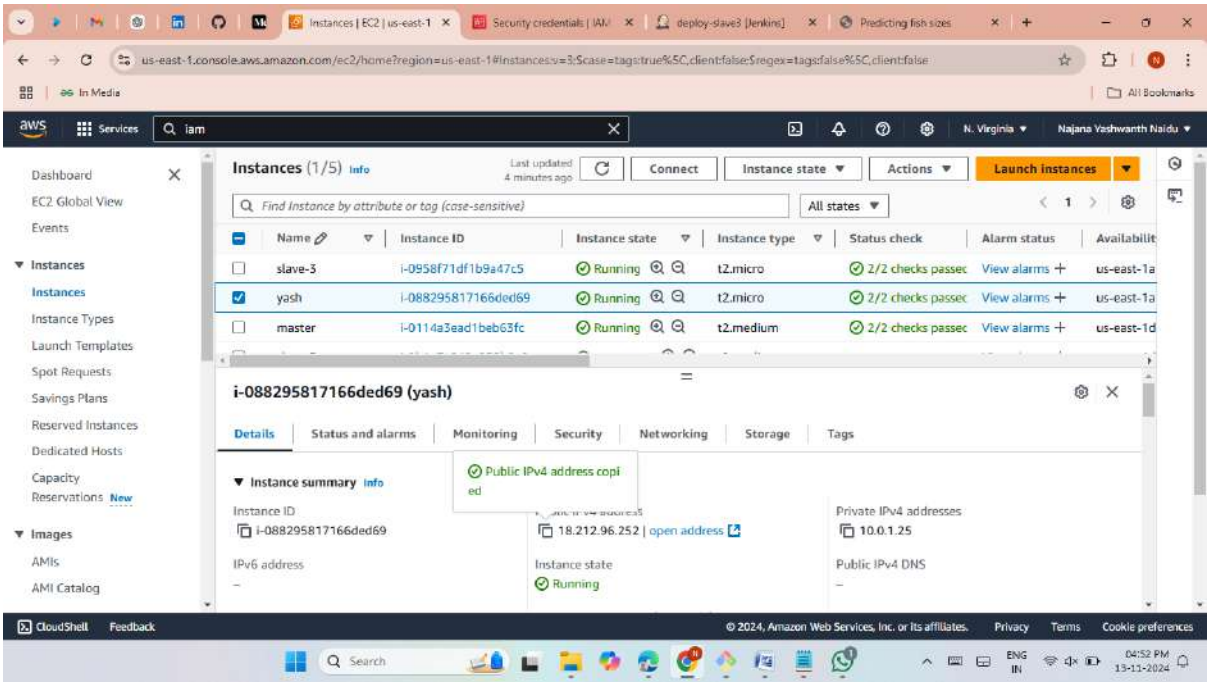
sudo terraform apply --auto-approve



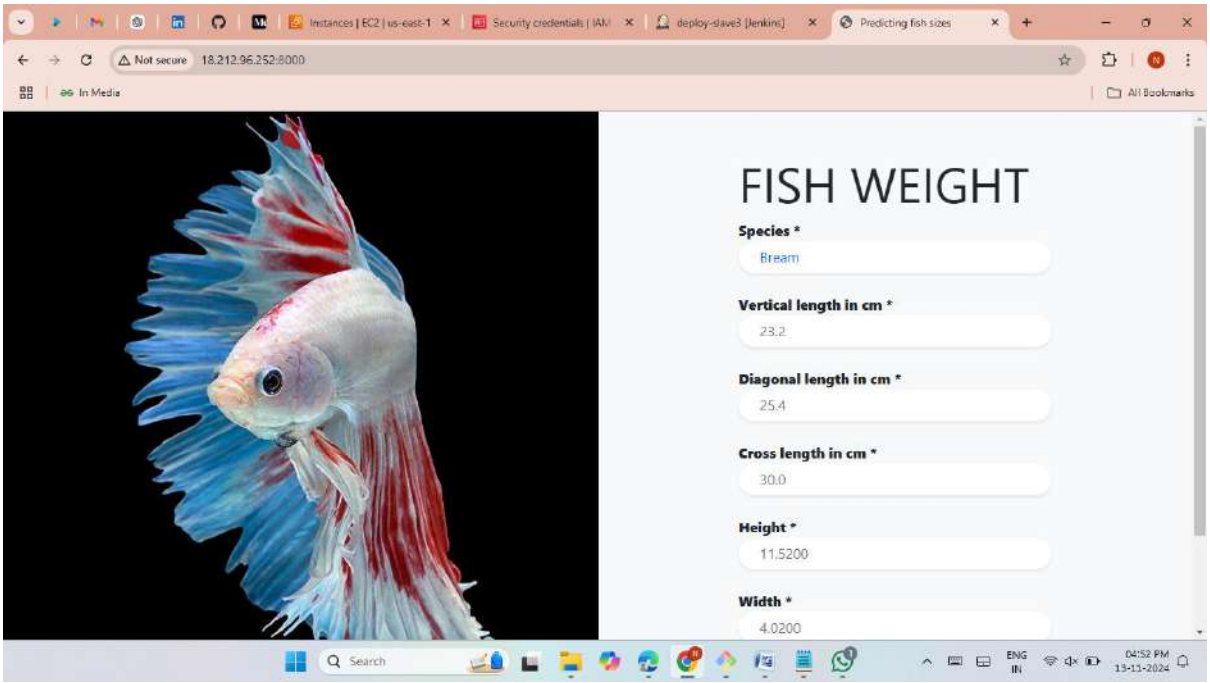
- Now click on build now.
- Here the job was success.



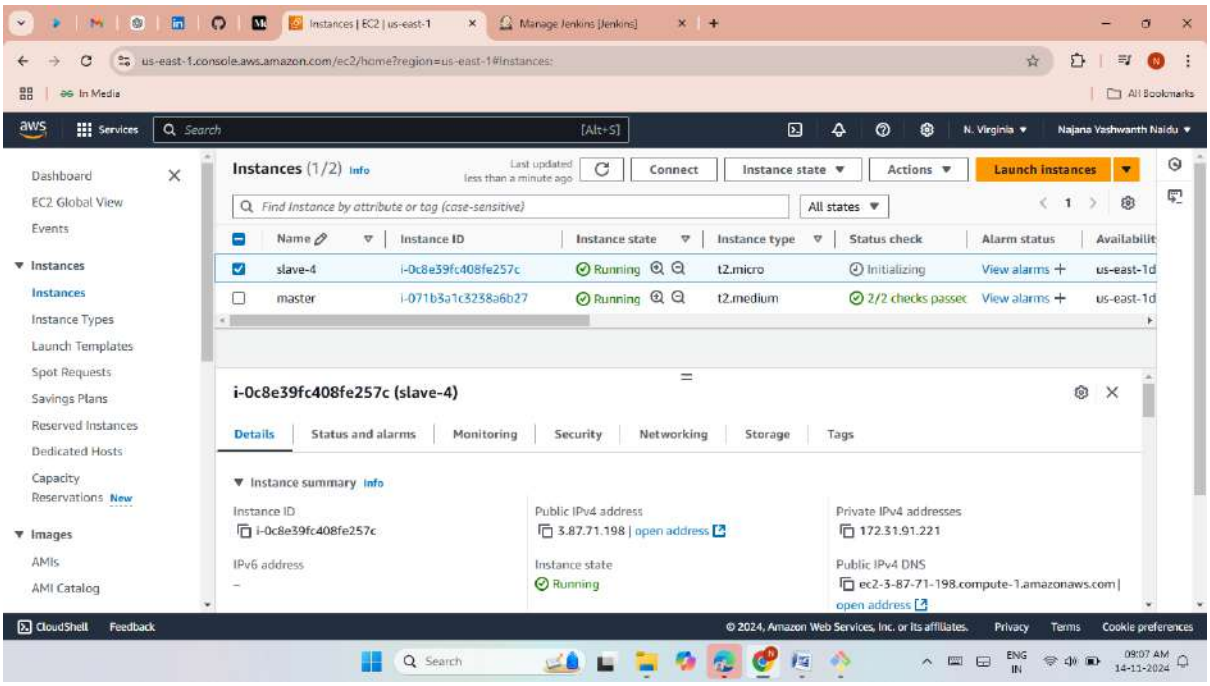
- Here the instance, VPC, Subnets, IGW, Route table was successfully created with help of Terraform script.



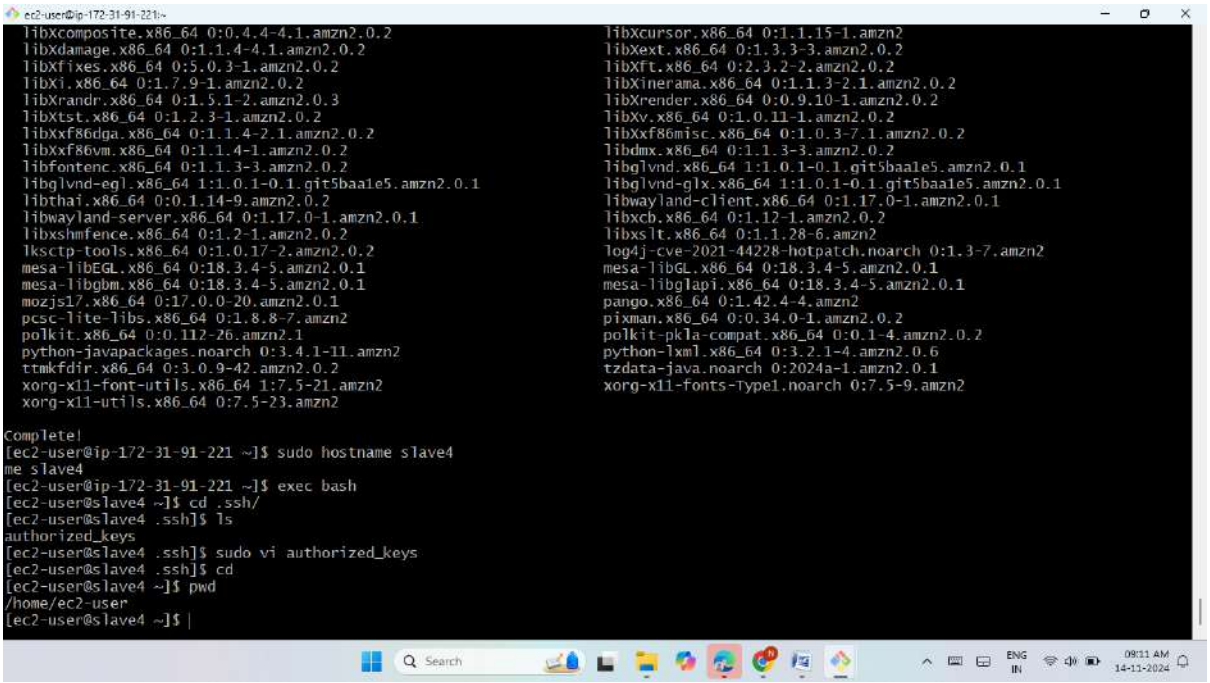
- Now copy the created instance public ip and past it on google browser.
- The python web application was hosted successfully.



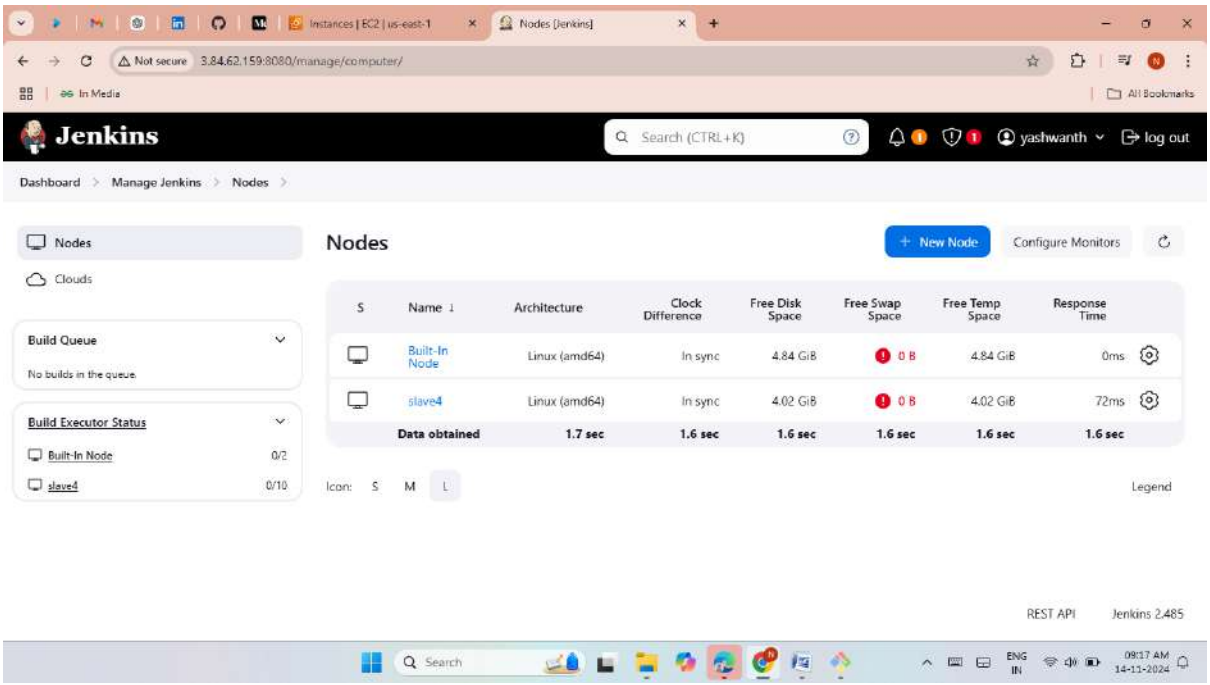
- Now launch an ec2 instances and give name as slave4.



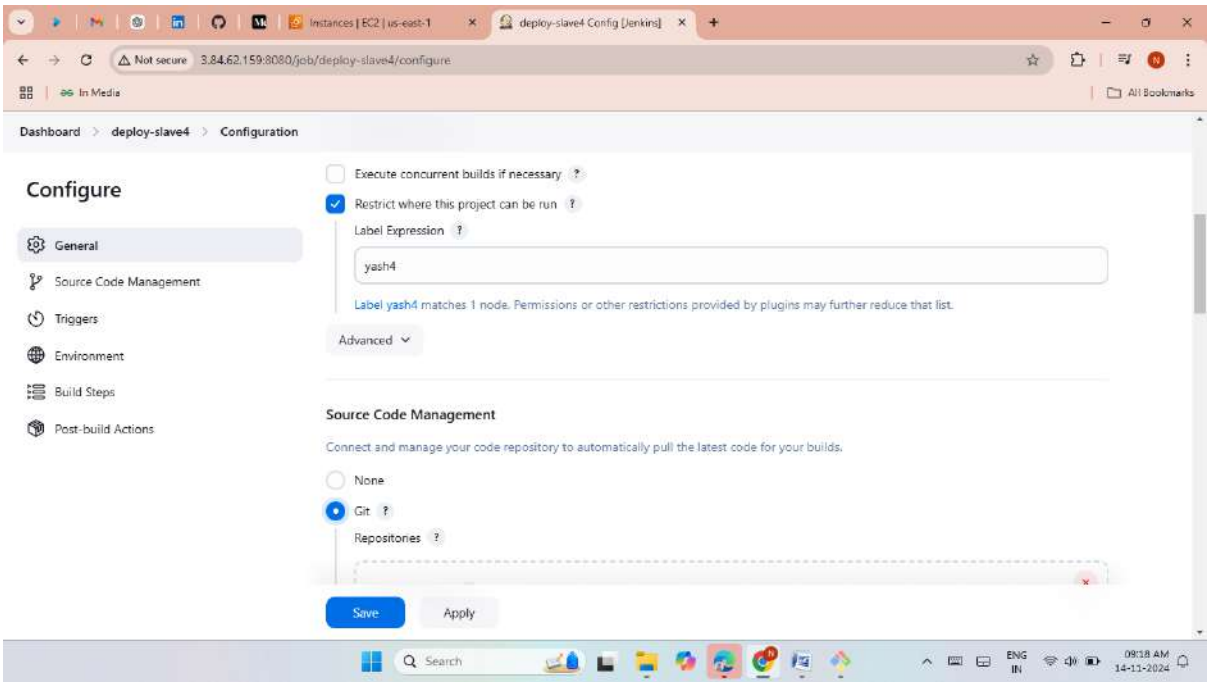
- Now connect Slave4 then install java.
- Now give private keys from master server.



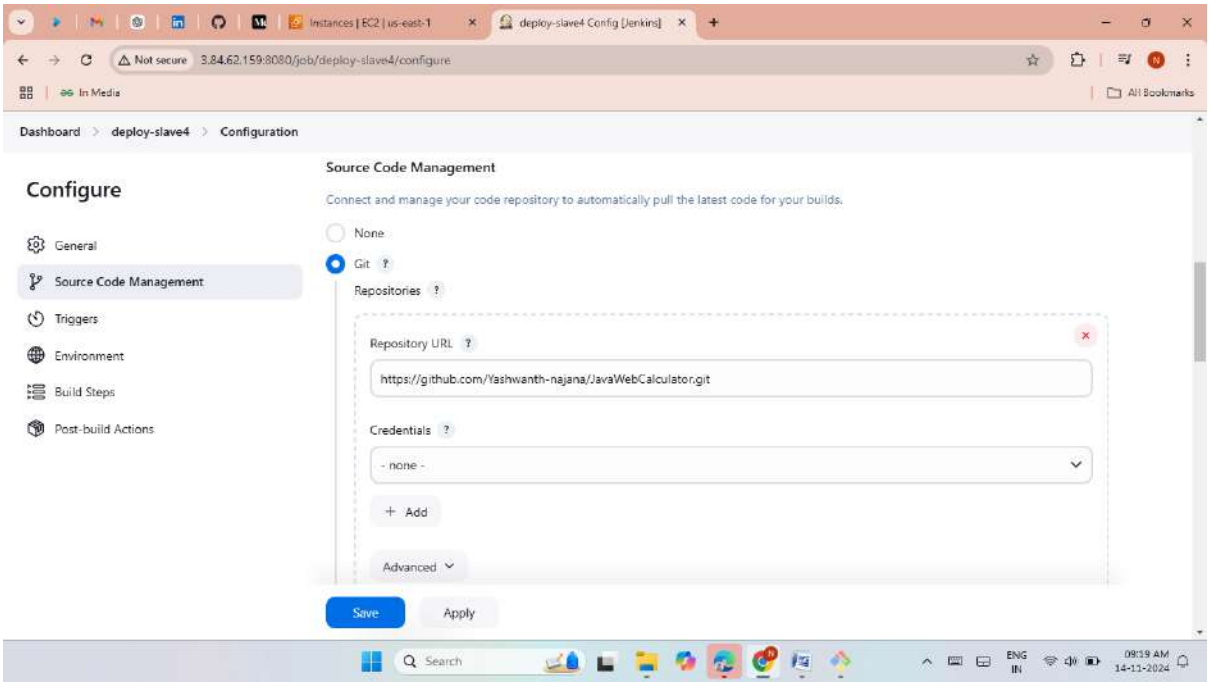
- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave4 node is successfully created.



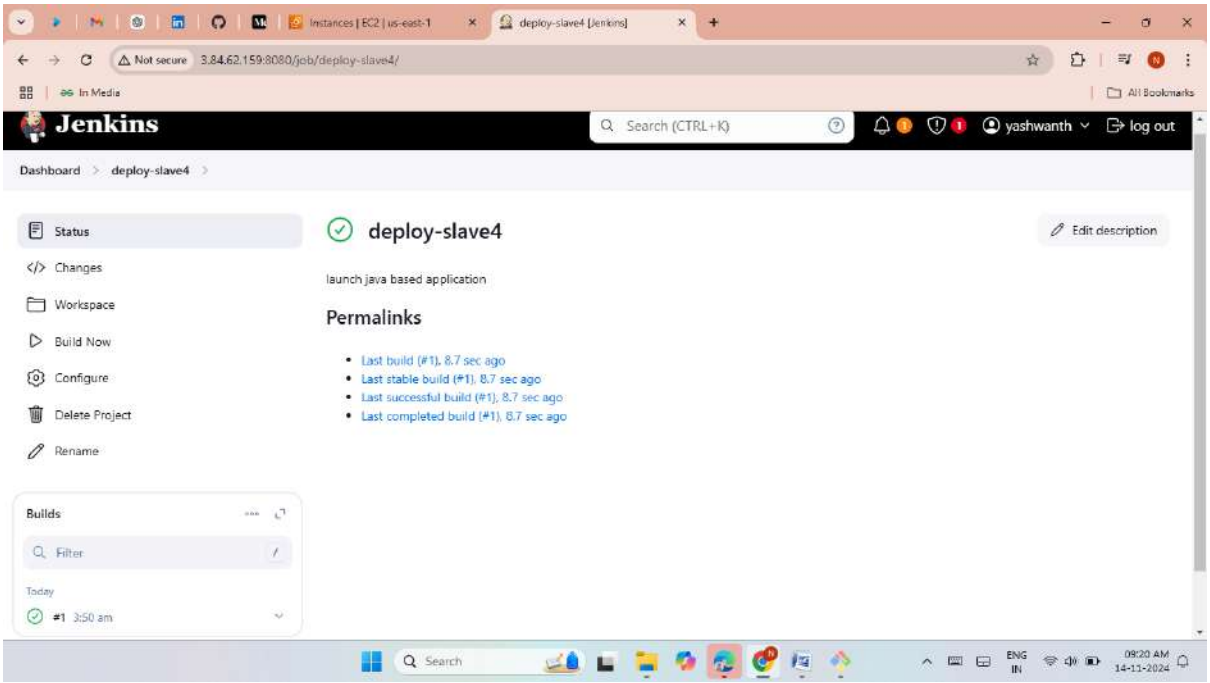
- Now create a job and give slave4 label.



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

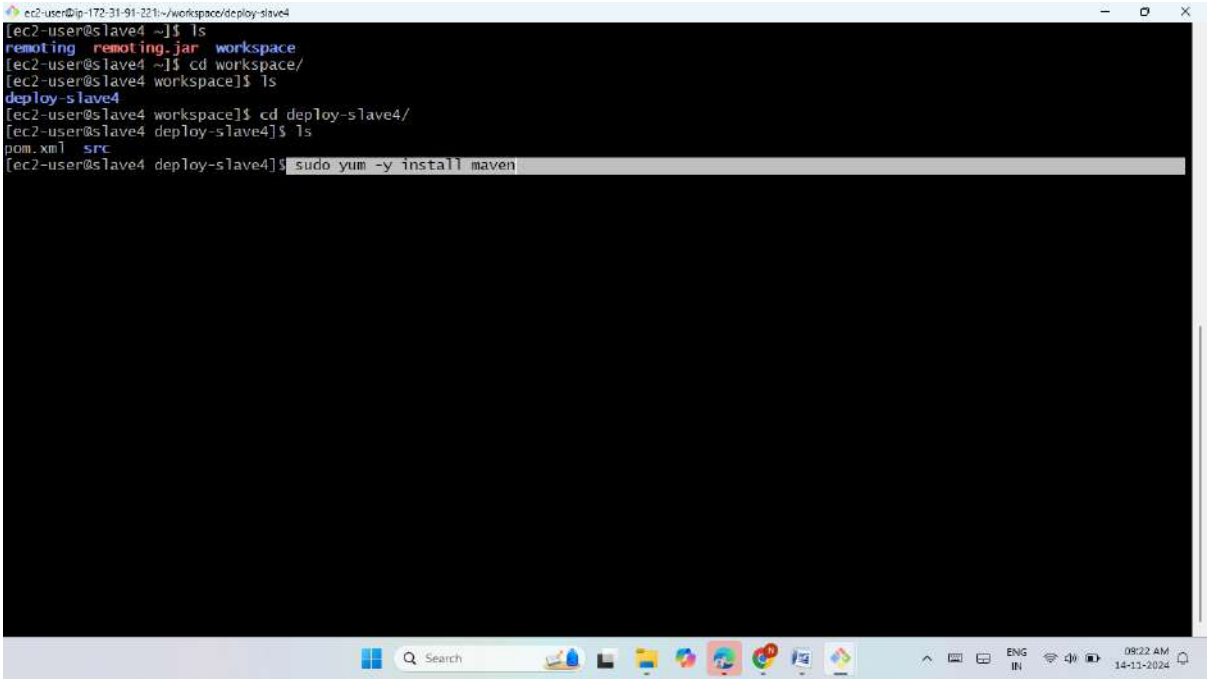


- Now click on build now.
- Here the job was success.

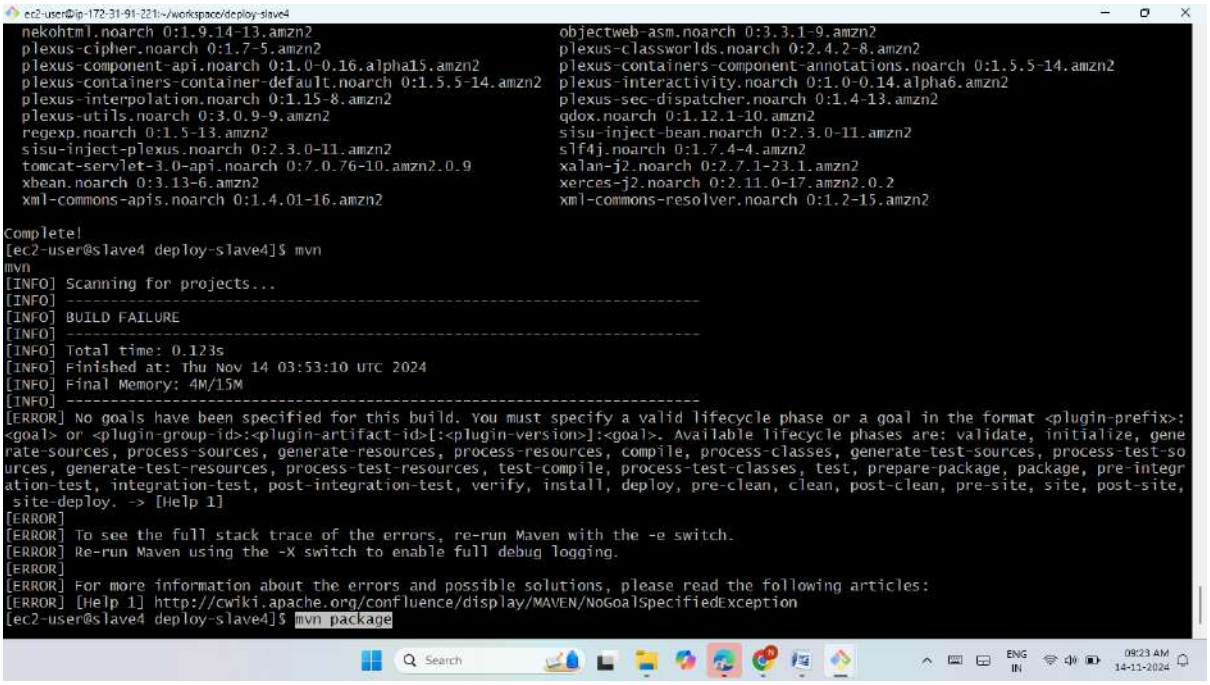


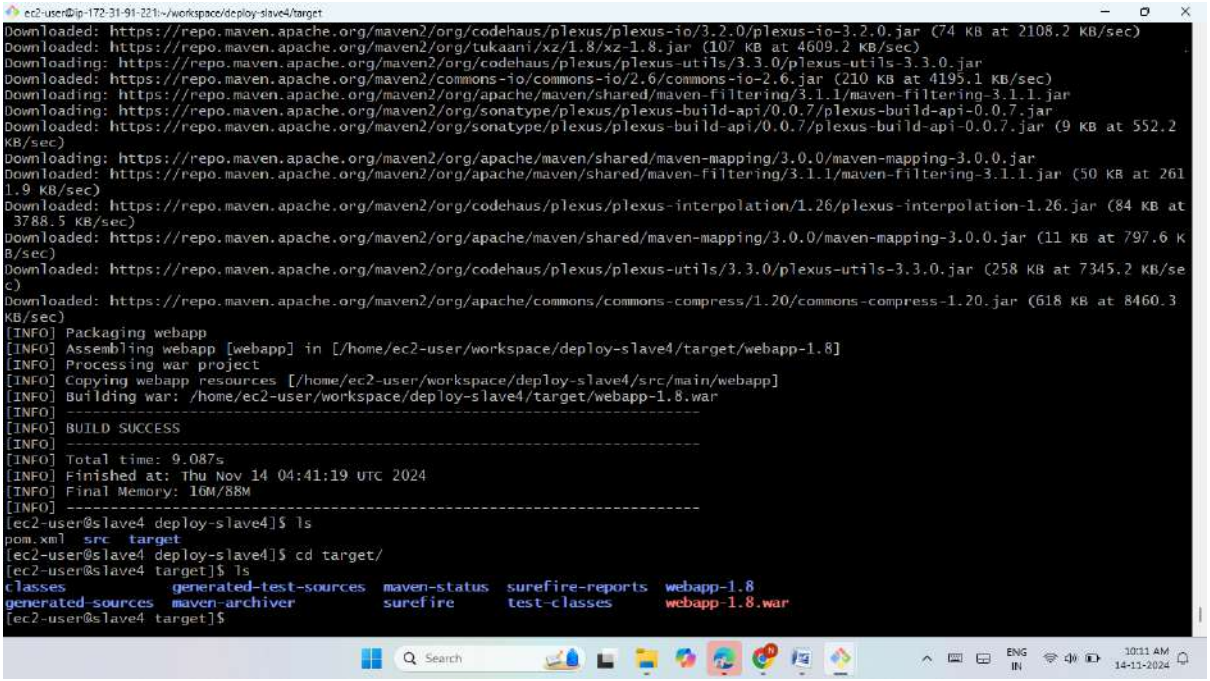
- Now launch java applications manually with help of some commands
- Install maven with the command.

sudo yum –y install maven



- Now package the application with the command
- Mvn package

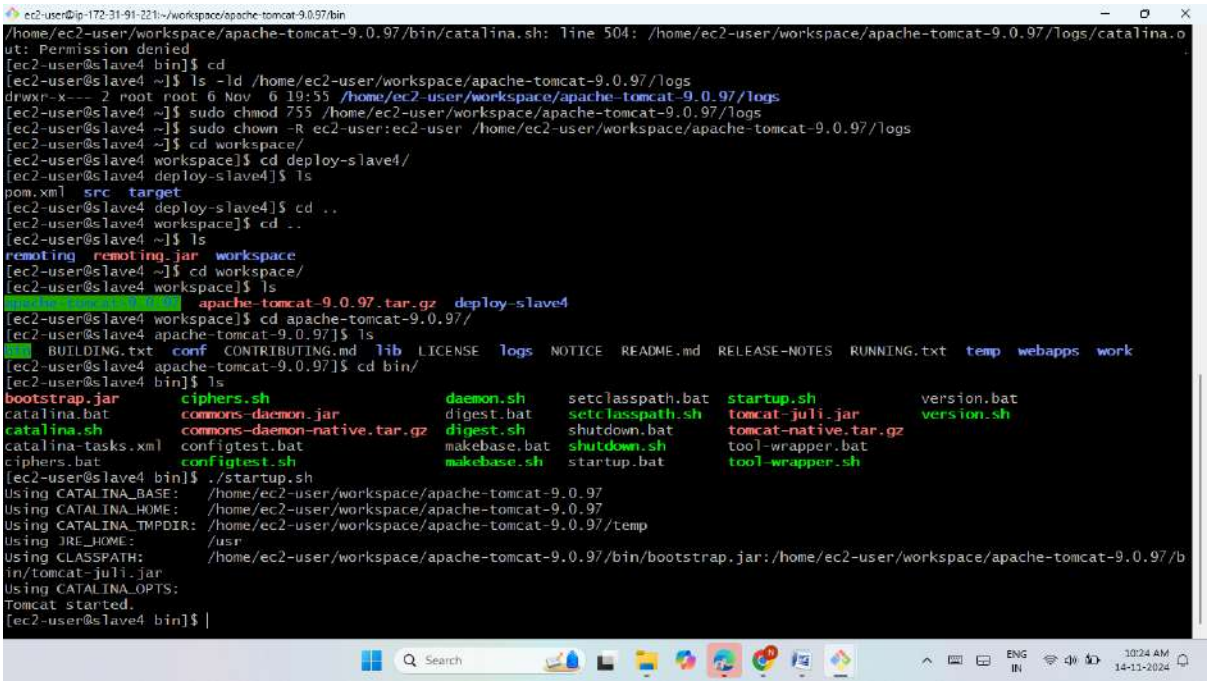




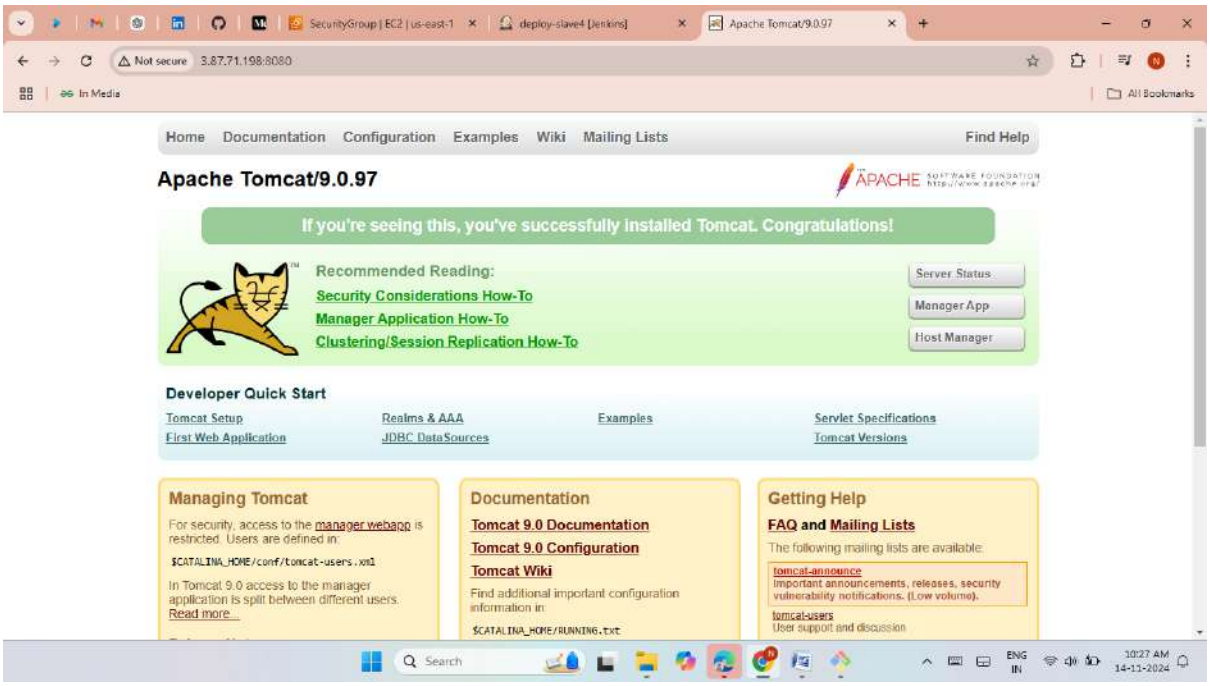
- Now install tomcat tar file form Google browser.
- Unzip the tomcat tar file with the command.

  Sudo tar –xvf <file name>
- Now start the tomcat with the command.

  ./startup.sh

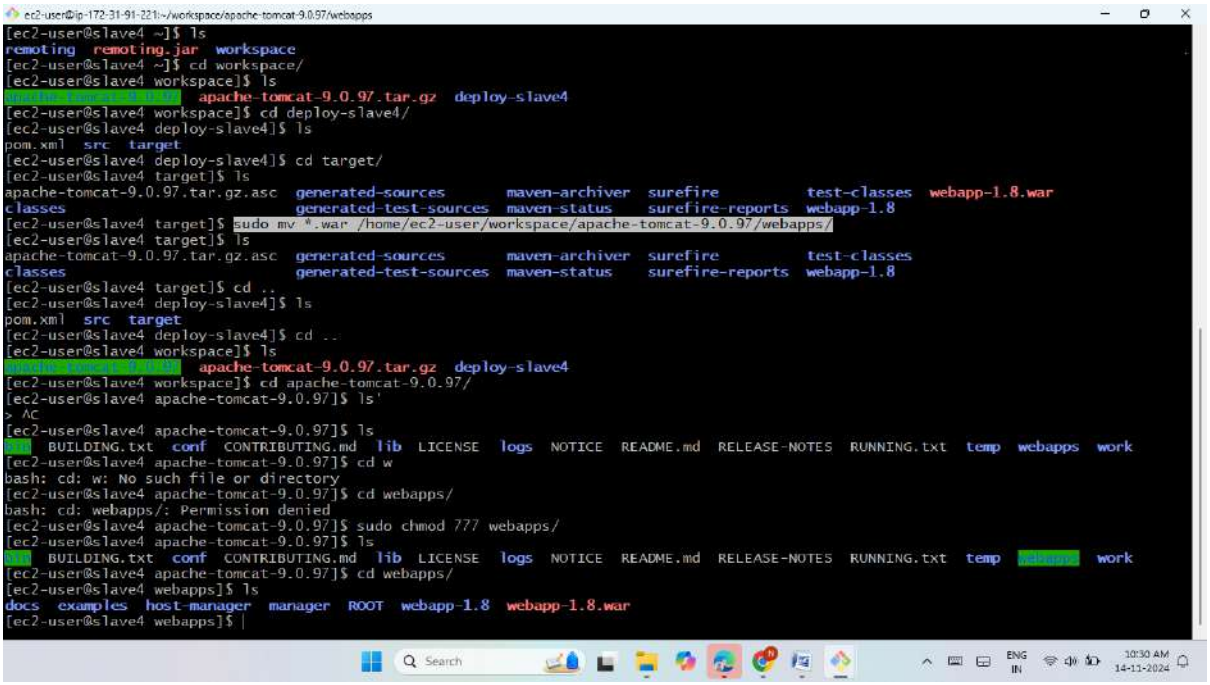


- Here the tomcat was hosted successfully.

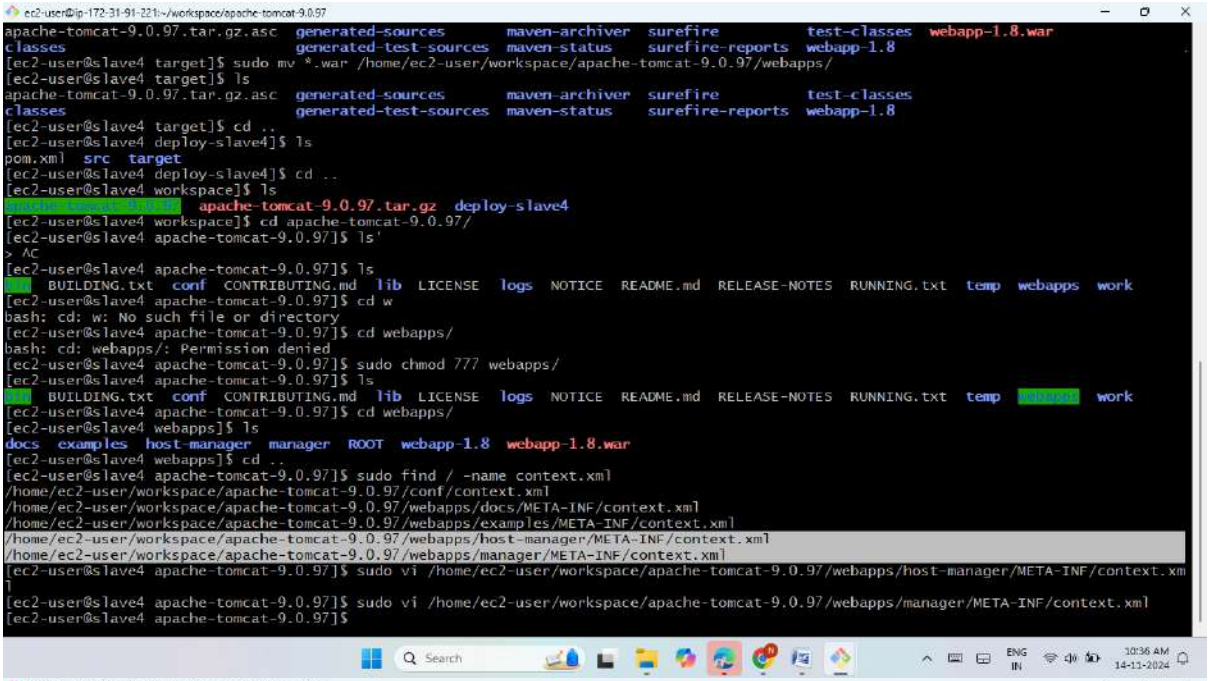


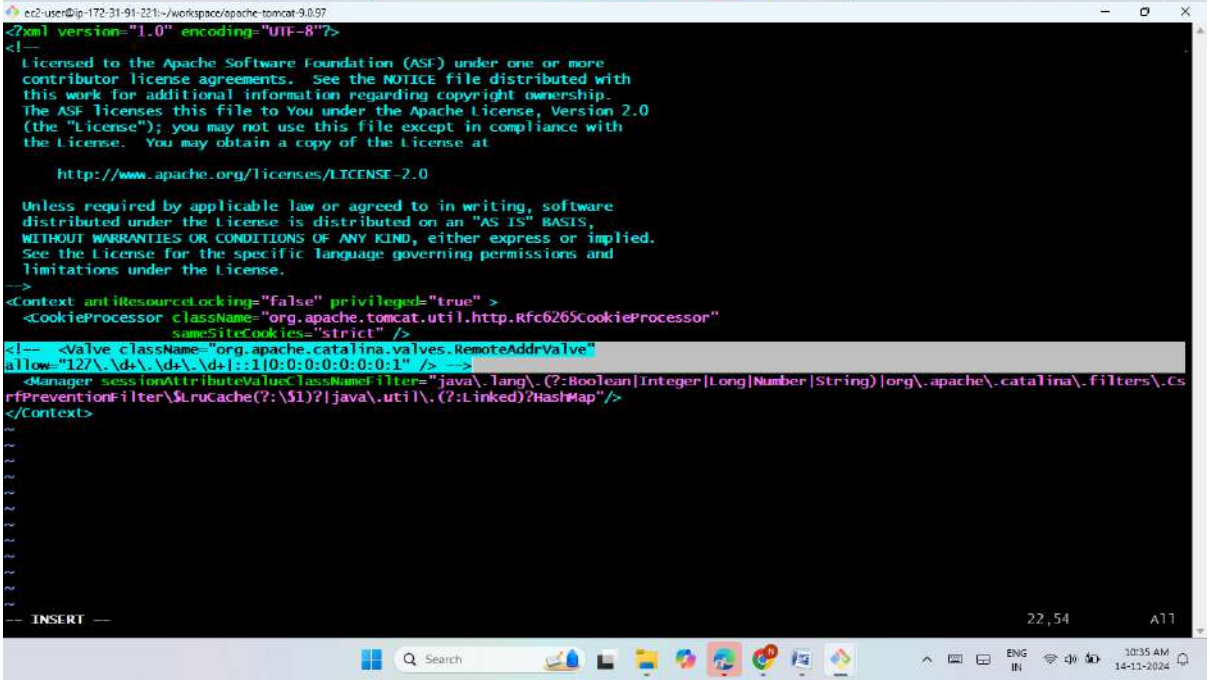
- Now copy the .war files form target to webapps.

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

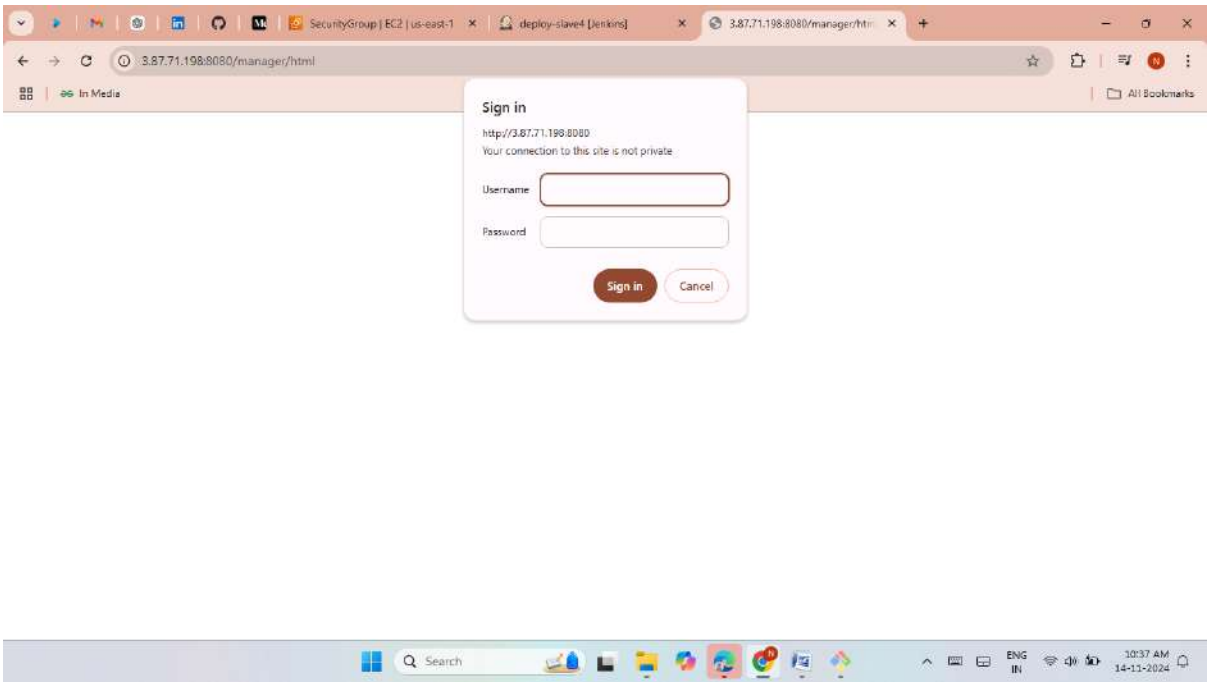


- Now go to host-manager and manager context file then remove some permission.

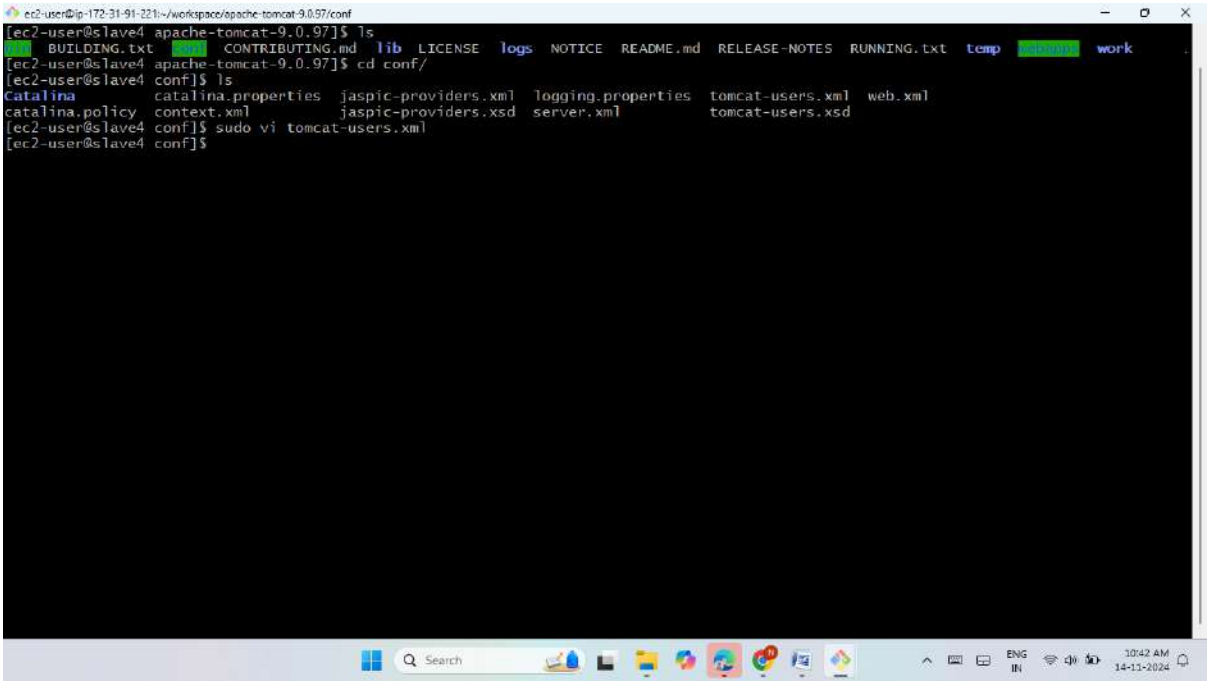


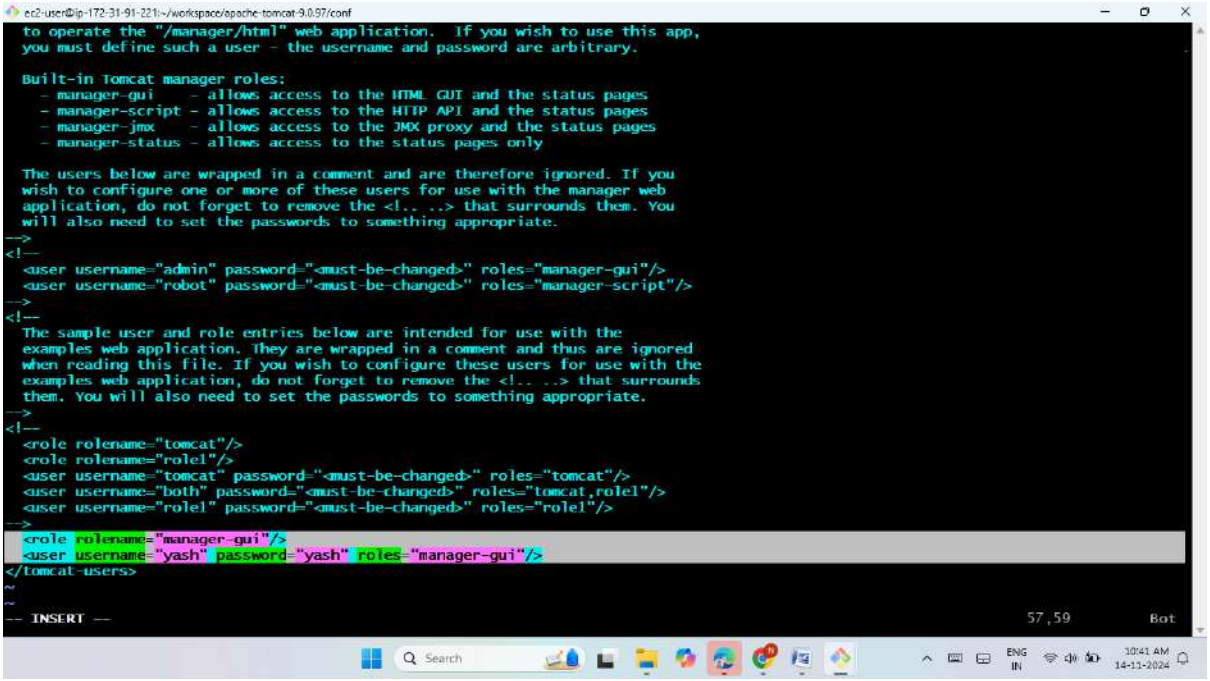


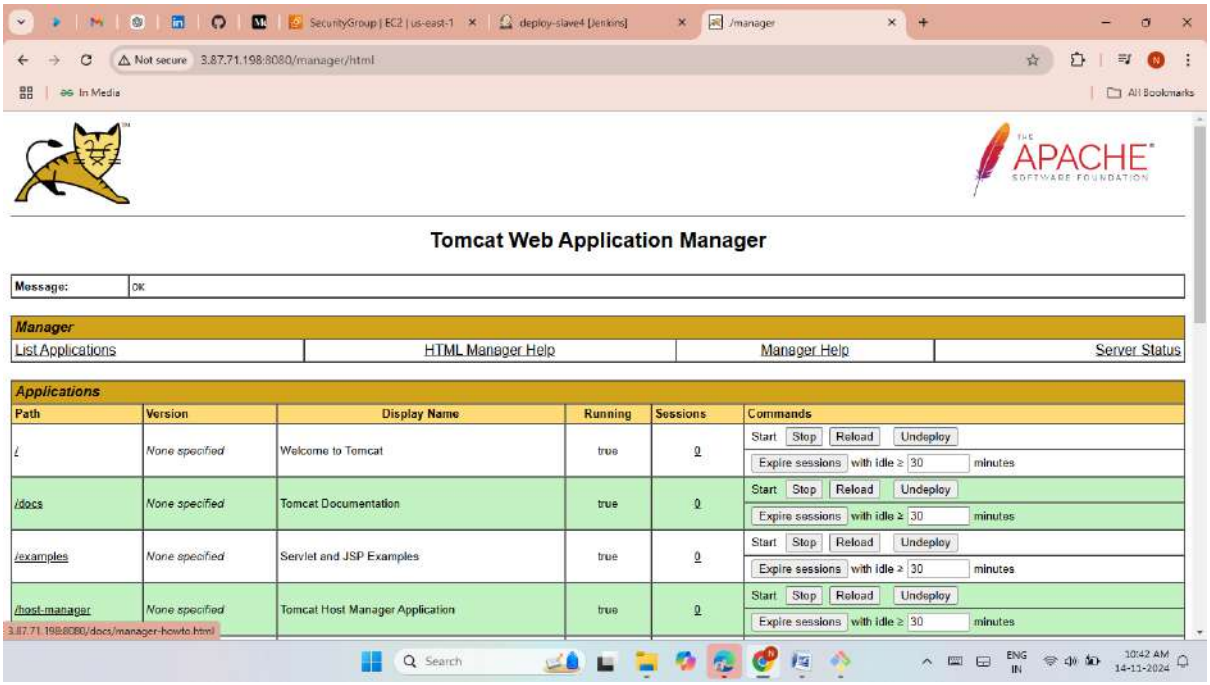
- Click on manager apps in tomcat the pages was changed.



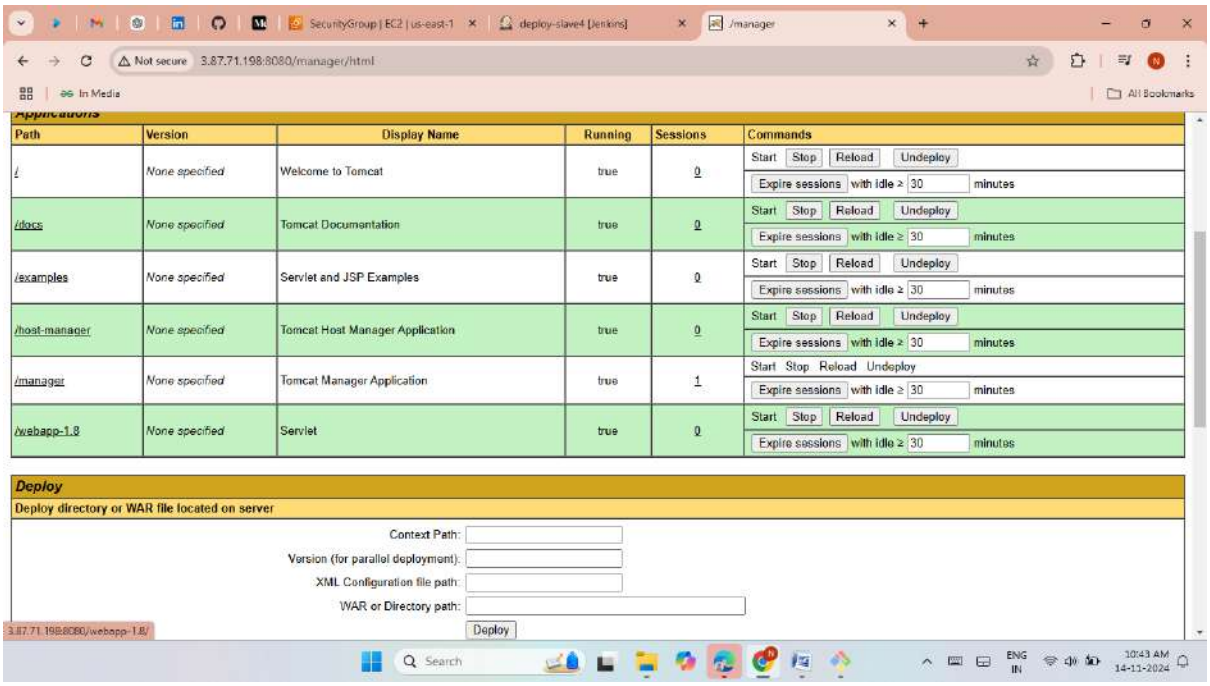
- Now give role name, username and password in tomcat-users.xml file.



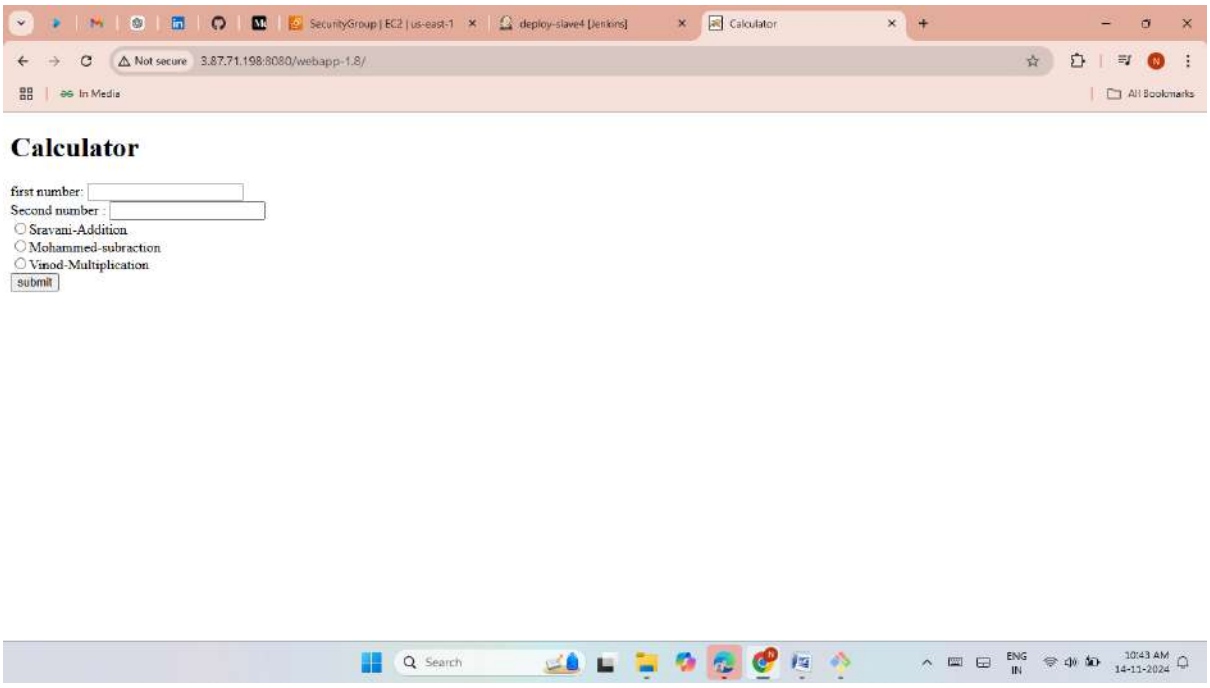

ec2-user@ip-172-31-91-221:~/workspace/apache-tomcat-9.0.97/conf
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary.
Built-in Tomcat manager roles:
- manager-gui - allows access to the HTML GUI and the status 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="<must-be-changed>" roles="manager-gui"/>
<user username="robot" password="<must-be-changed>" 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="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<role rolename="manager-gui"/>
<user username="yash" password="yash" roles="manager-gui"/>
</tomcat-users>
-- INSERT --
57,59
Bot
Search
ENG
IN
10:41 AM
14-11-2024

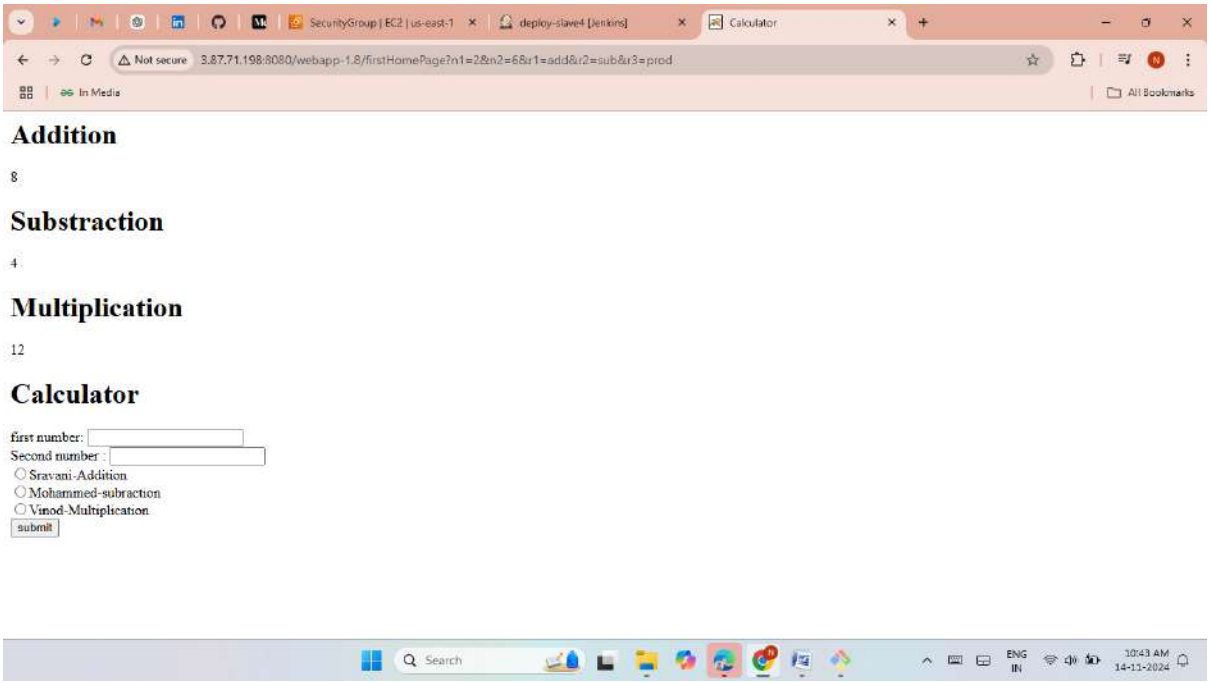

SecurityGroup | EC2 | us-east-1
deploy-slave4 [Jenkins]
/manager
Not secure 3.87.71.198:8080/manager/html
In Media
All Bookmarks
APACHE
SOFTWARE FOUNDATION
Tomcat Web Application Manager
Message: OK
Manager
List Applications
HTML Manager Help
Manager Help
Server Status
Applications
Path
Version
Display 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
3.87.71.198:8080/docs/manager-howto.html
Search
ENG
IN
10:42 AM
14-11-2024

- Now click on webapps-1.8.

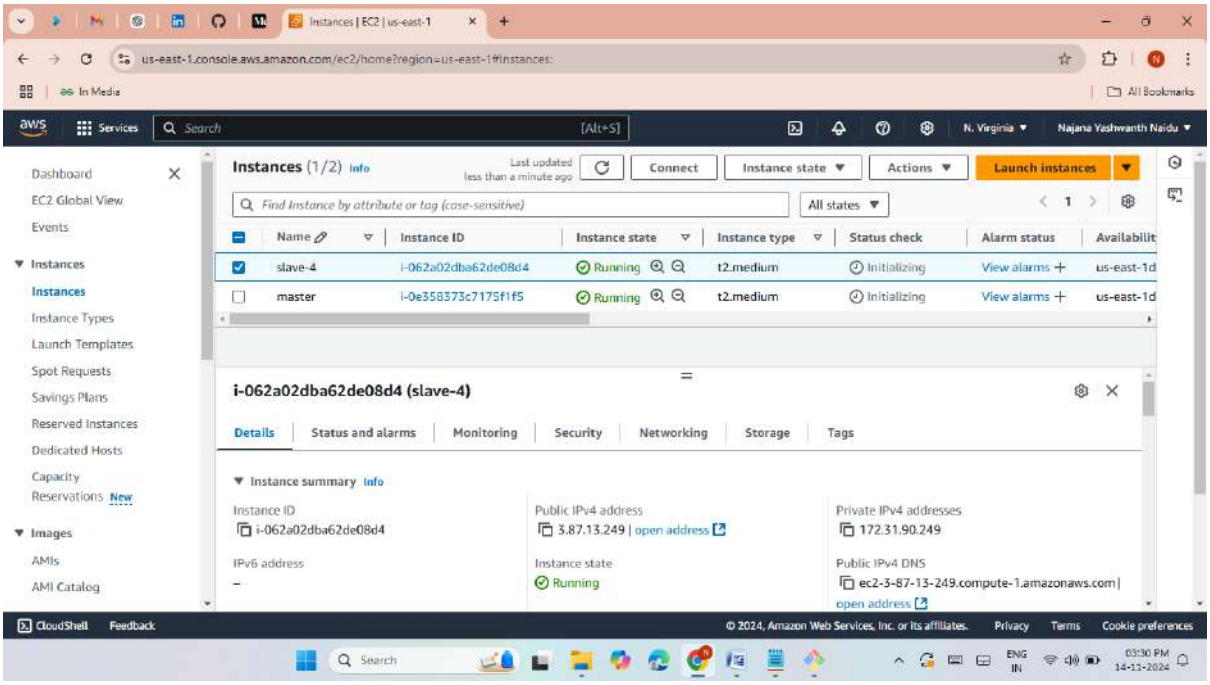


- The java web calculator was hosted successfully.

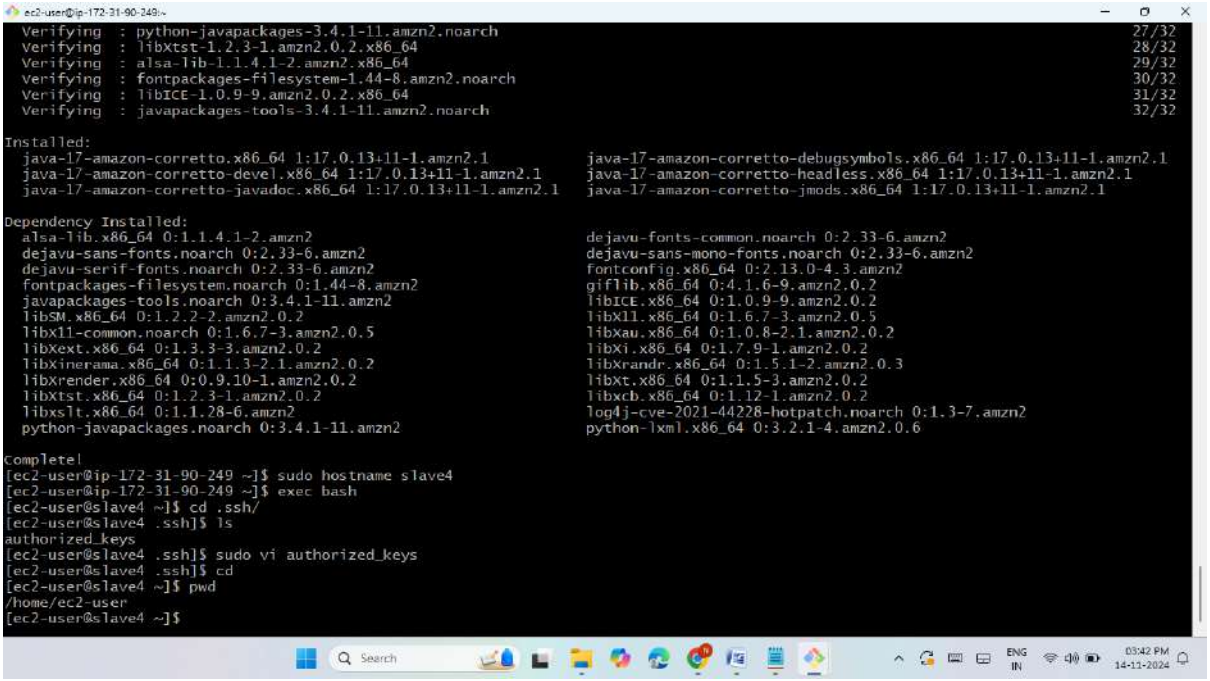




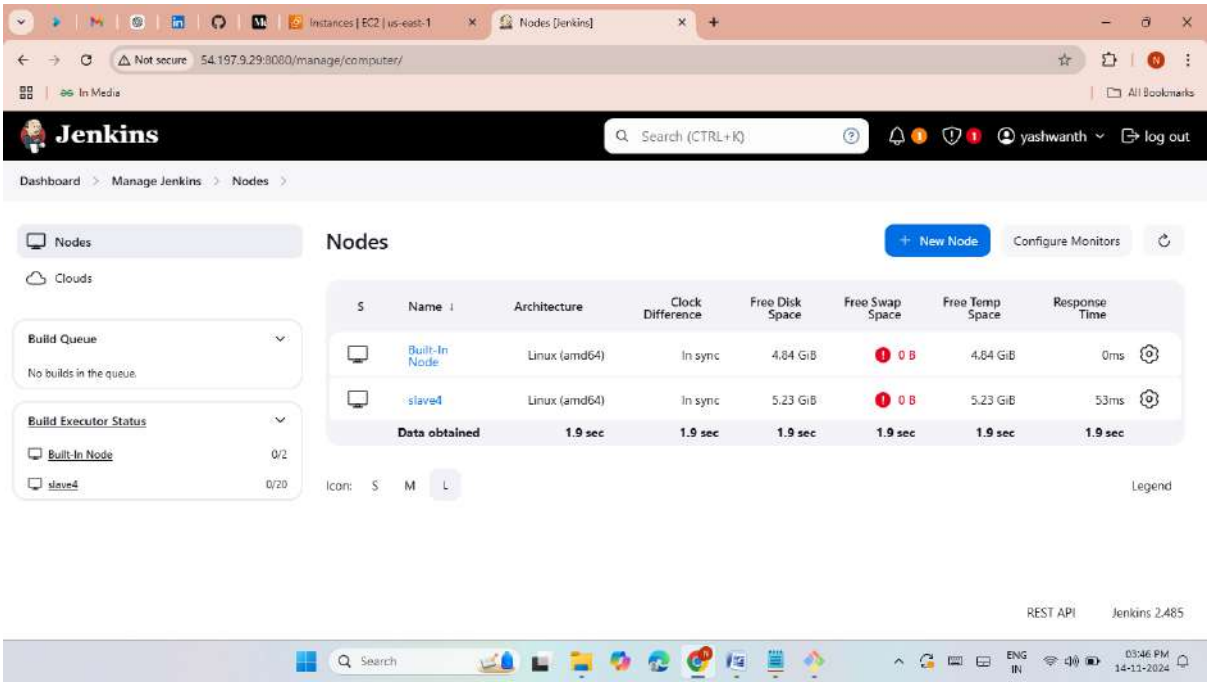
- Here I have done another method in slave4.
- Now launch an ec2 instances and give name as slave4.



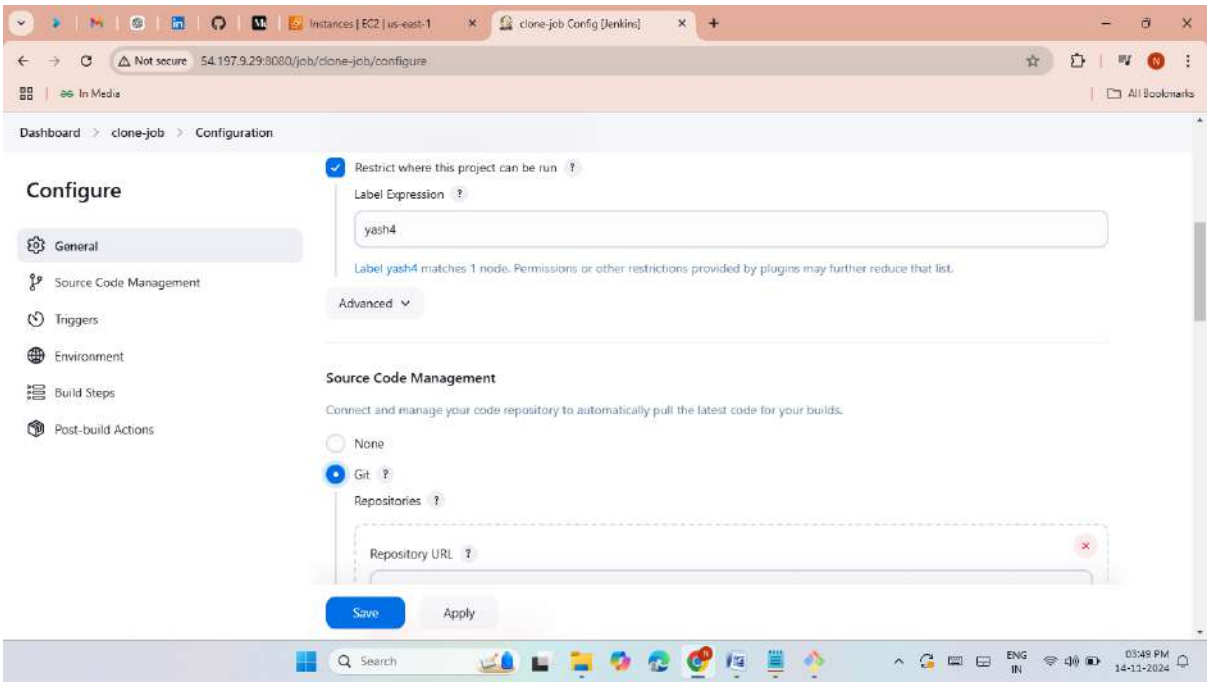
- Now connect Slave4 then install java.
- Now give private keys from master server.



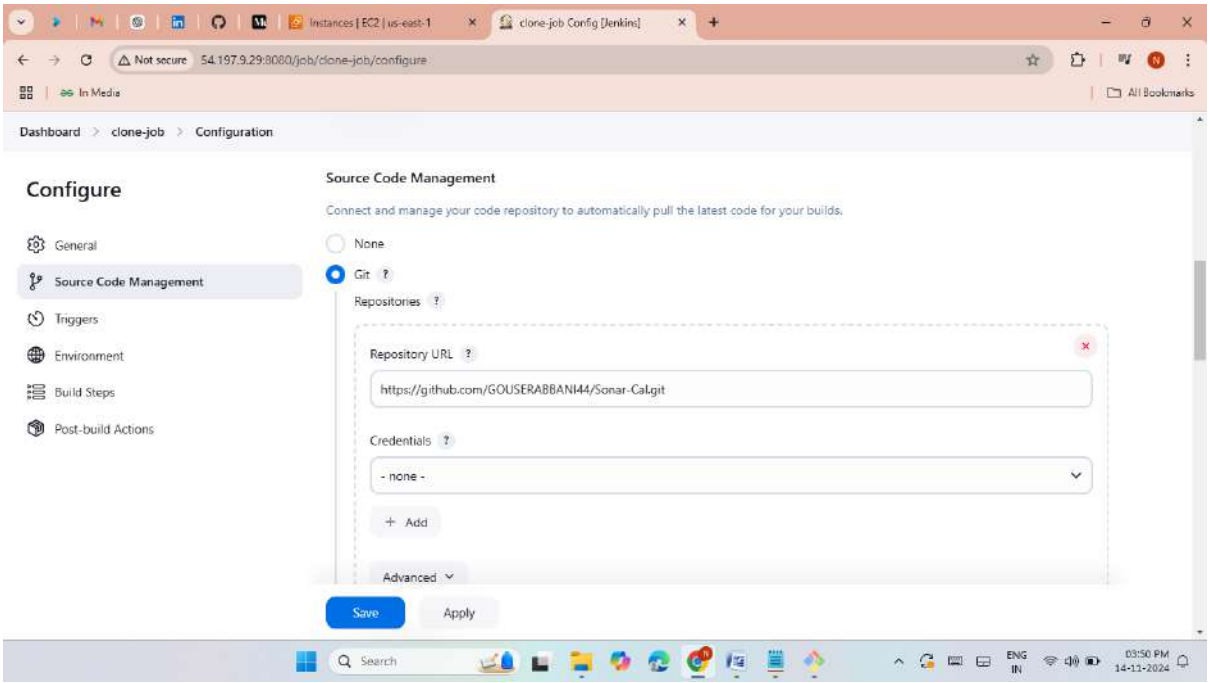
- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave4 node is successfully created.



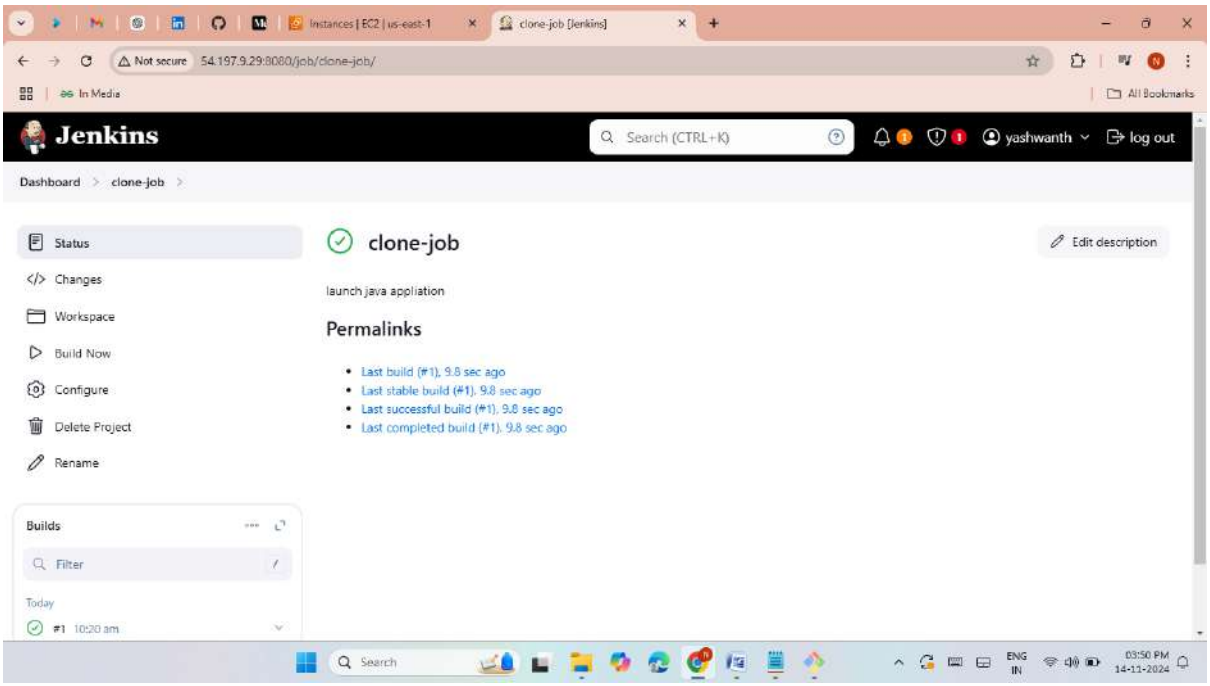
- Now create a cone-job and give slave4 label.



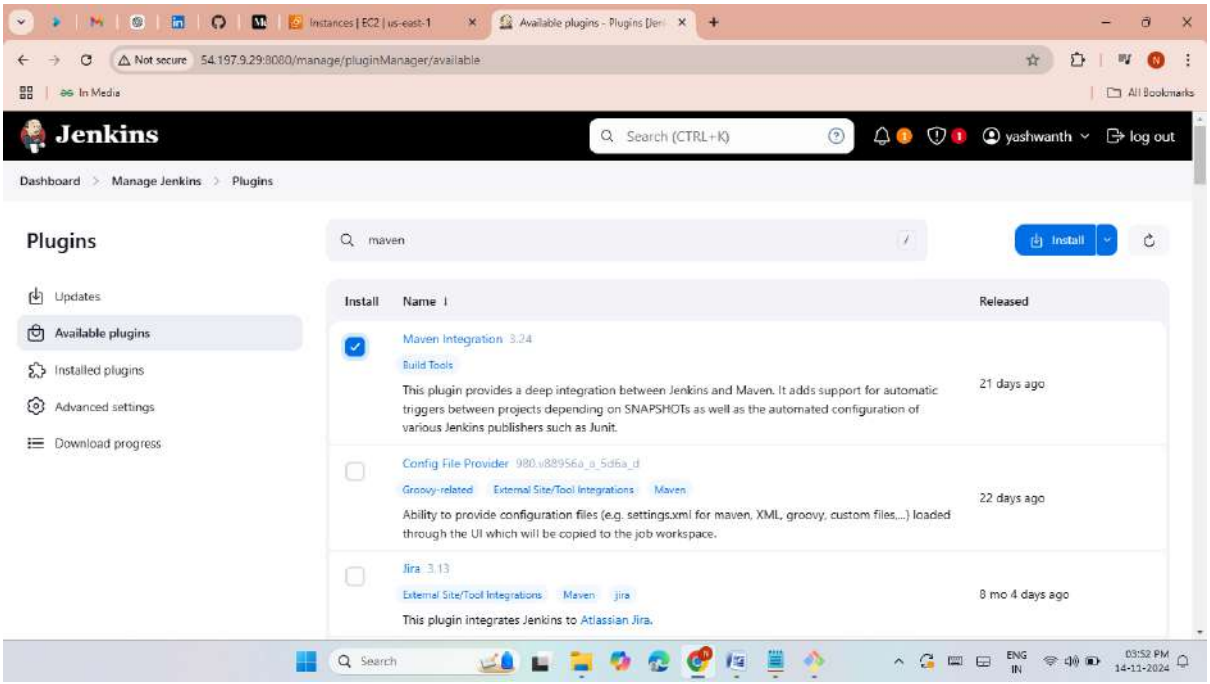
- Now clone the java application repo form github.
- Now Click on save.



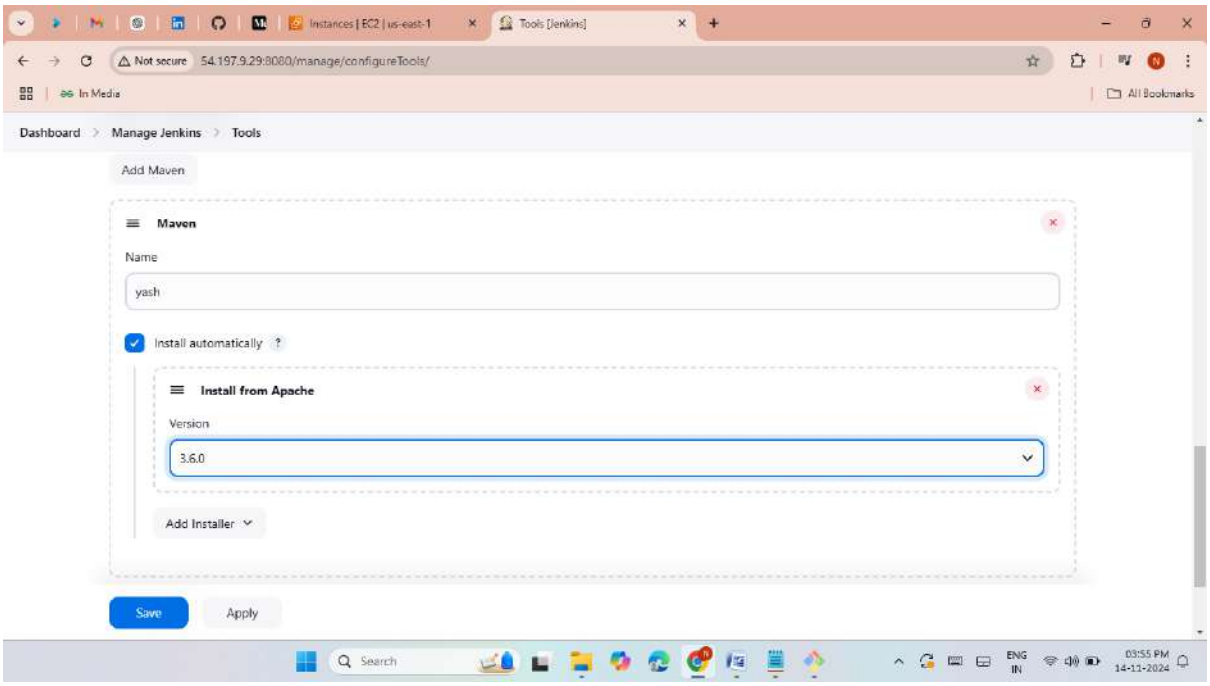
- Now click on build now.
- Here the job was success.



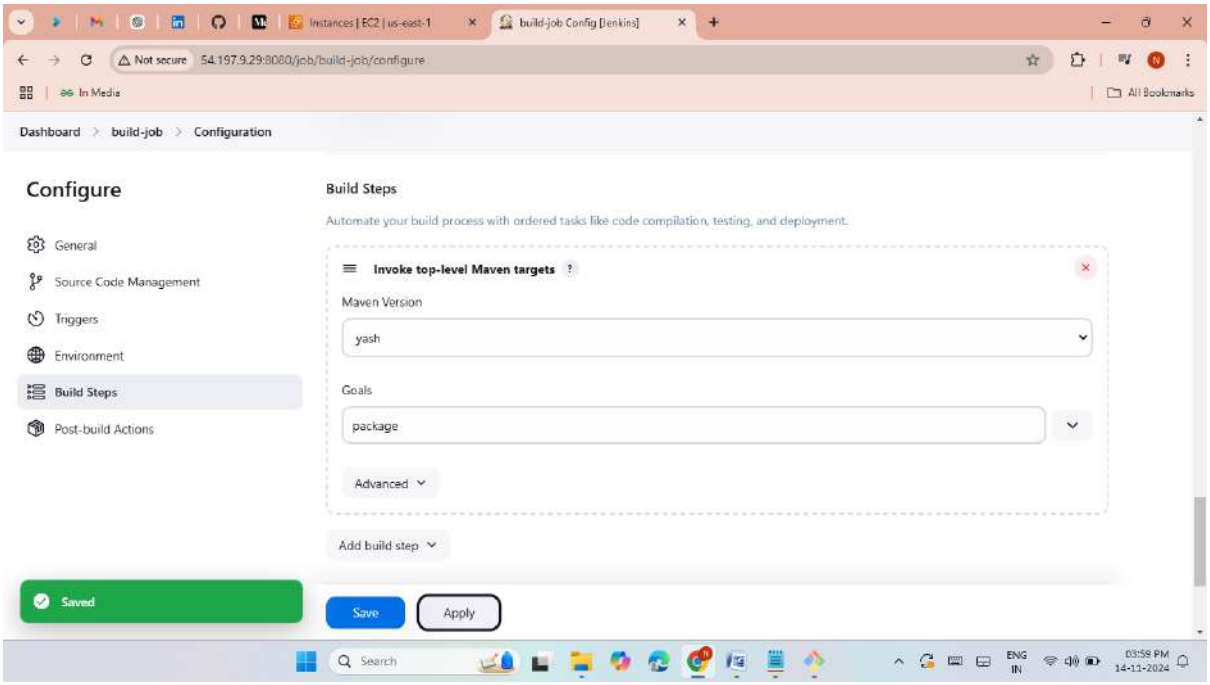
- Now install maven integration plugin for package the code.



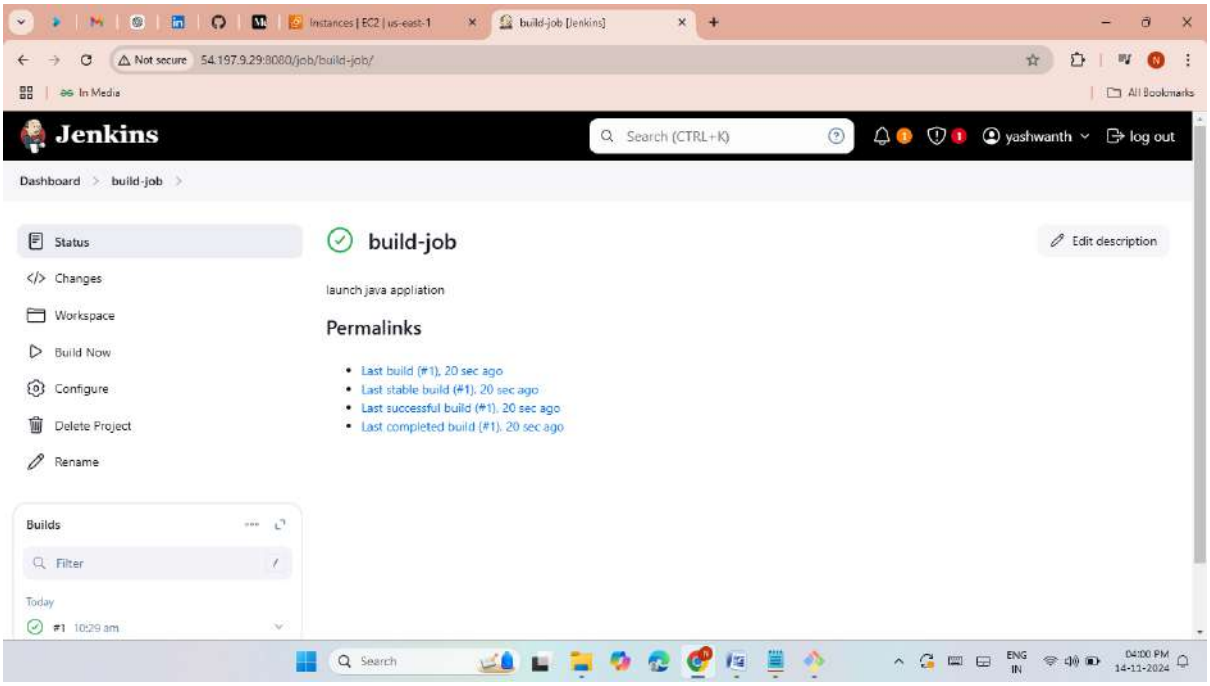
- Now go to tools then add maven along with the version.



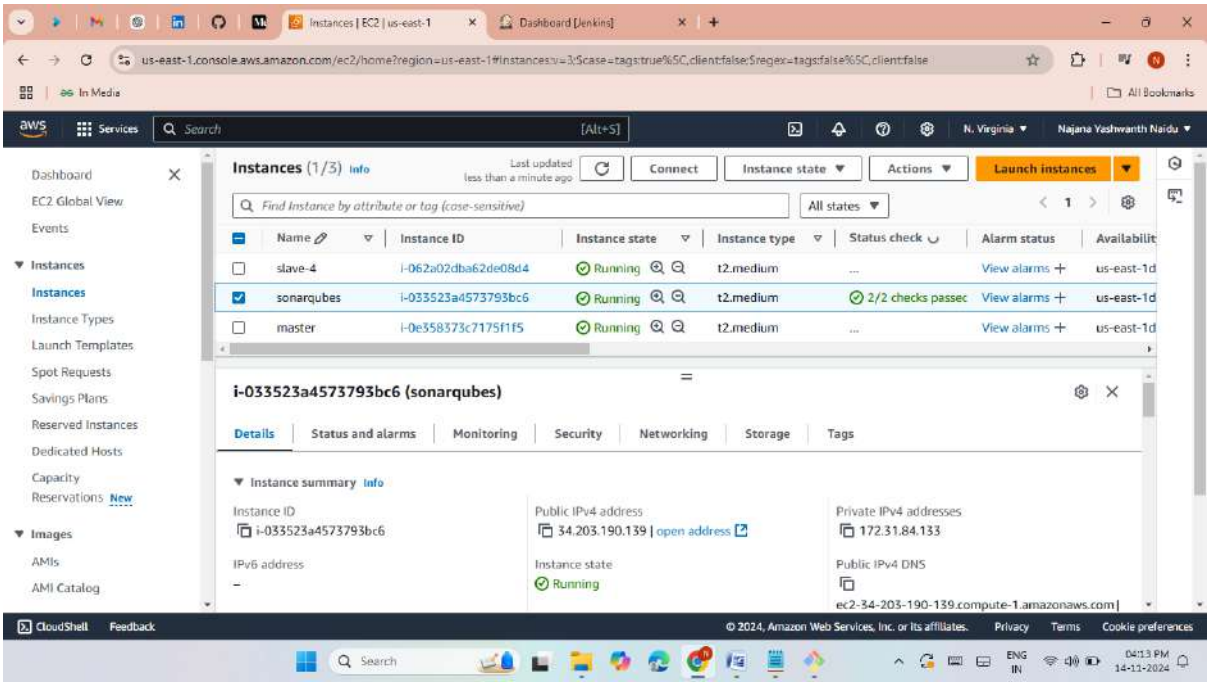
- Now create a build-job and copy the clone job created information.
- In build step add the created maven version and set the goles.
- Now click on apply.



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



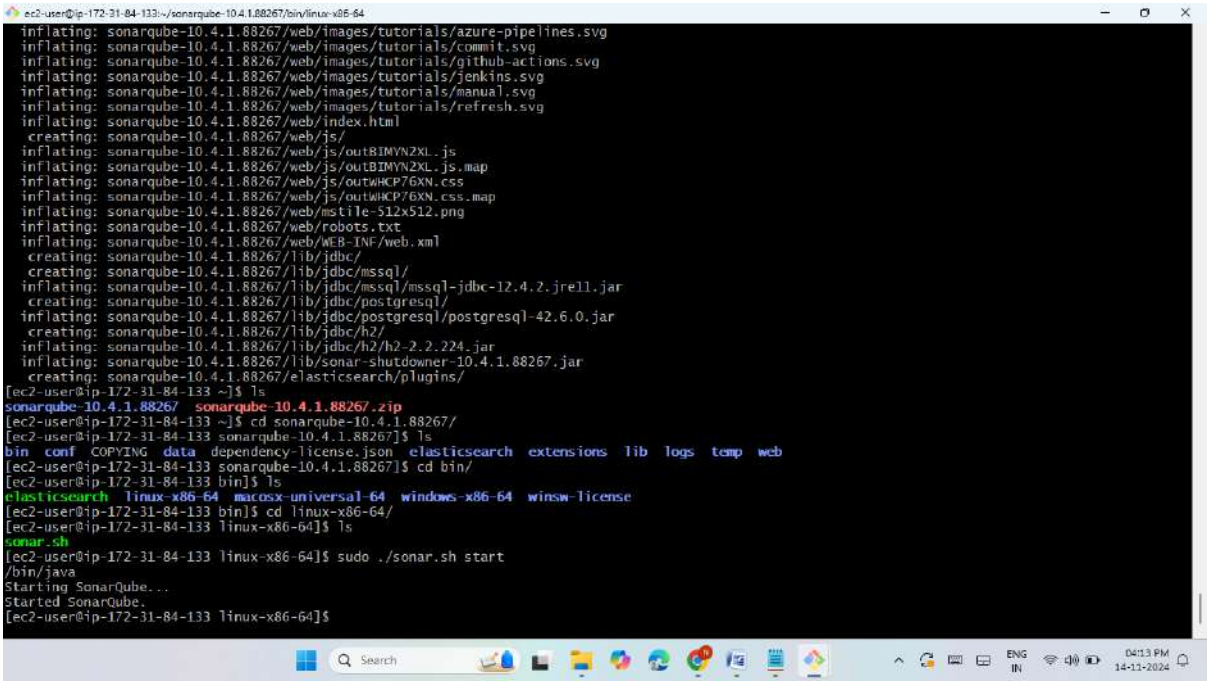
- Launch an another EC2 instance for install sonarqubes.
- In this step we can install sonarqubes in the same server.



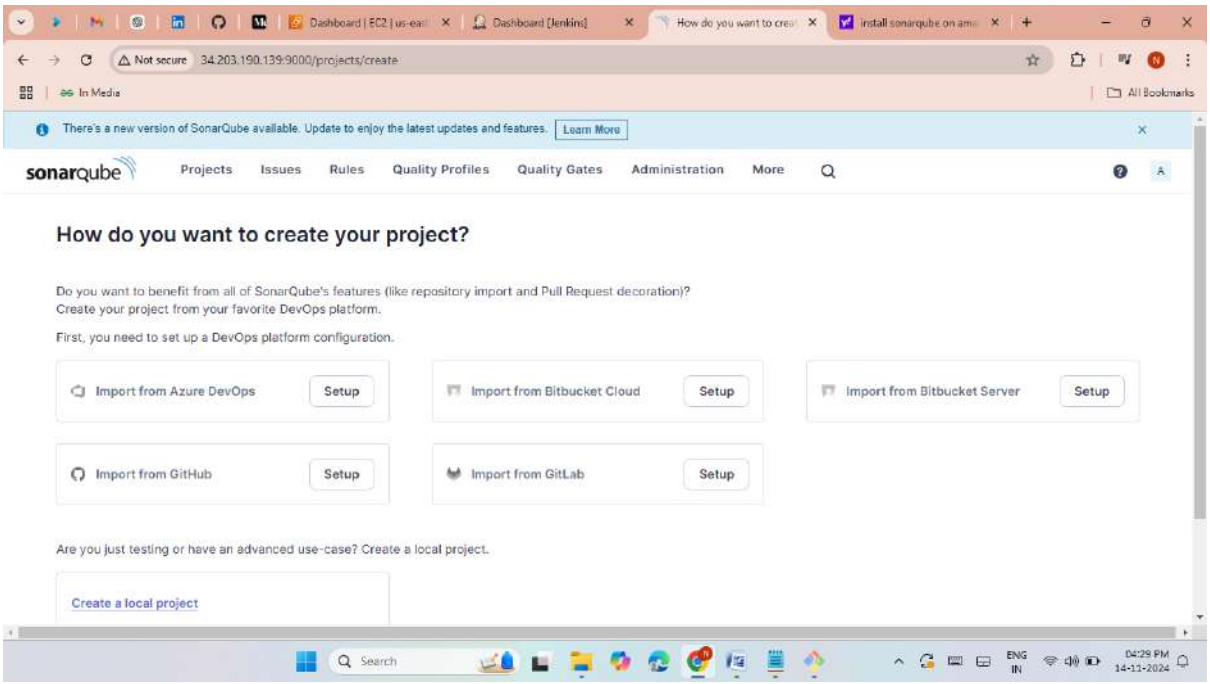
- Now connect the created server and install sonarqube zip file form Google.
- Unzip the file with the command

  Sudo unzip <file>
- Now go to sonarqubes→bin→linux-x86-64
- Run the sonarqube with the command

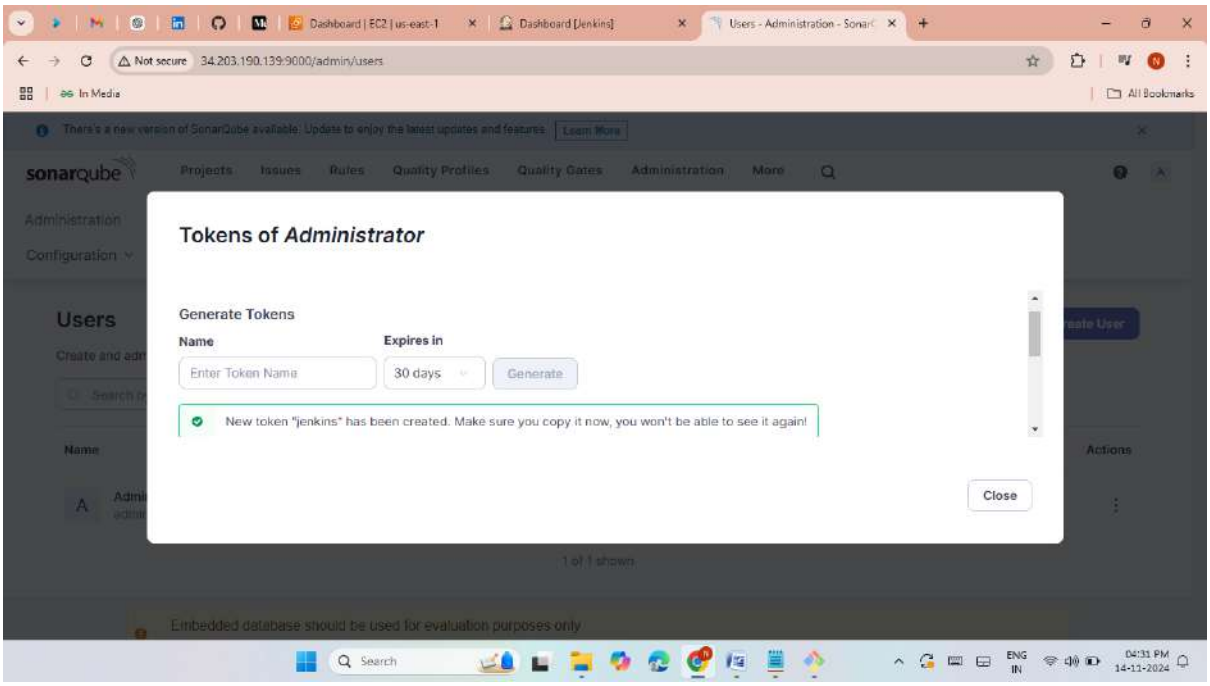
  Sudo ./sonar.sh start



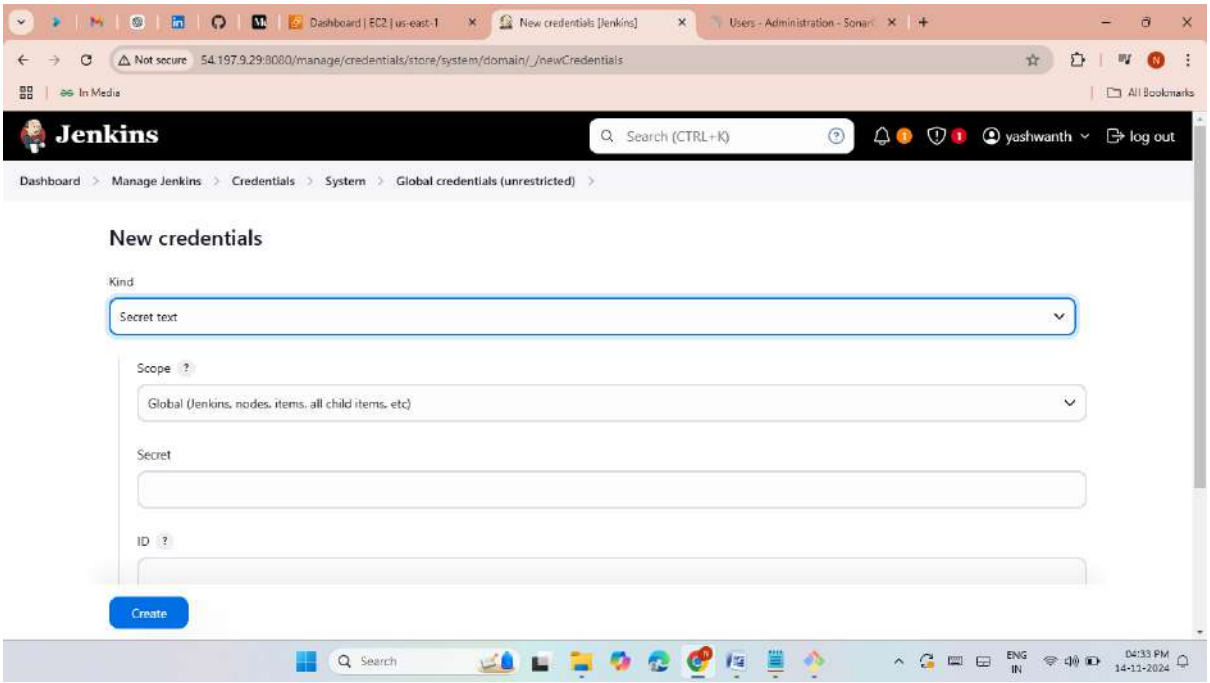
- Now launch the sonarqube with the ip address along with the port.



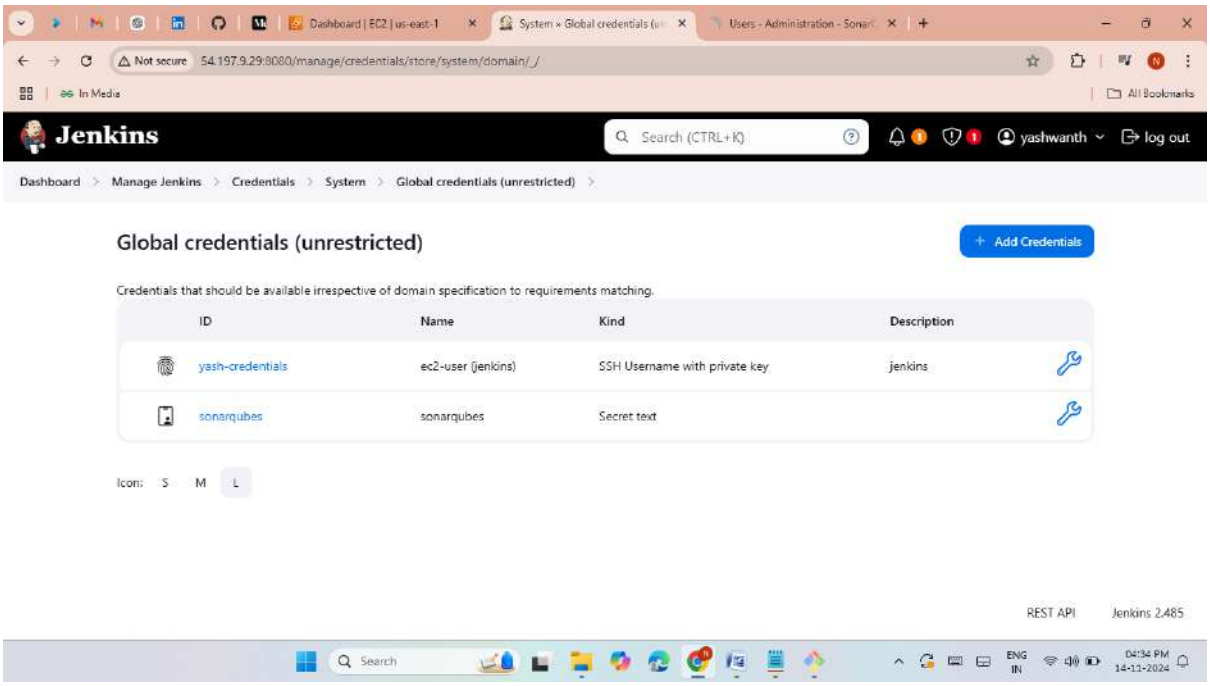
- Now generate token for access it on Jenkins.



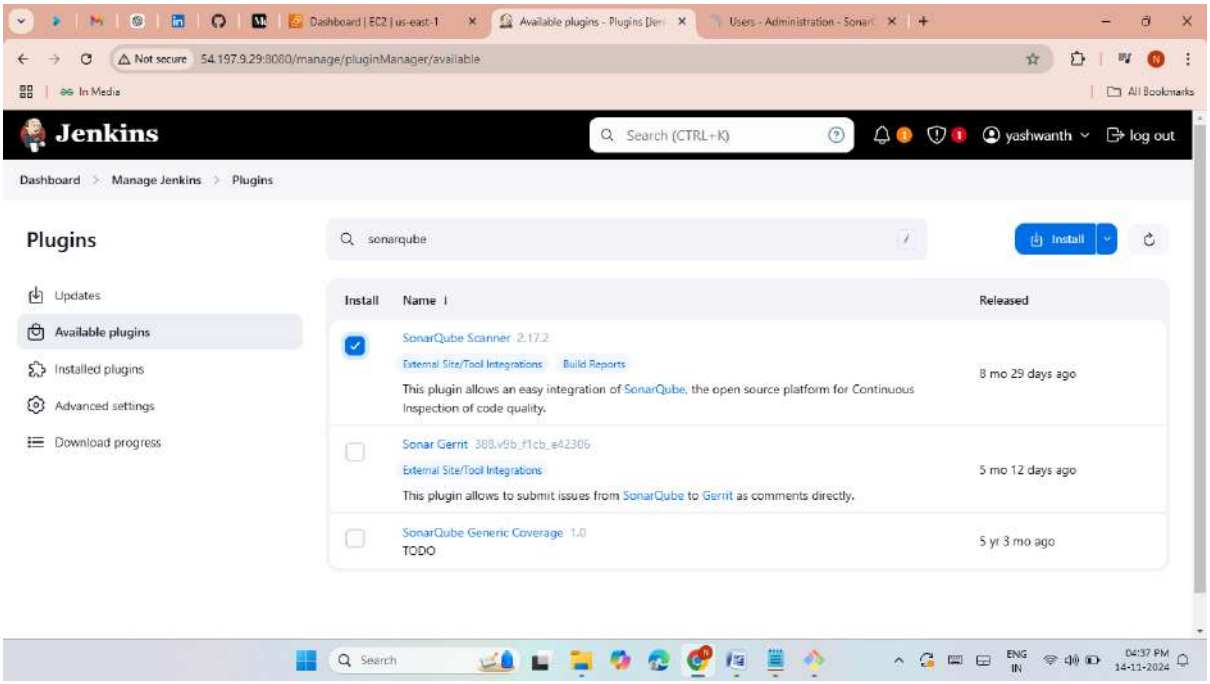
- Now create a credentials in Jenkins.
- Select secret text
- Past the sonarqubes token into the secret.



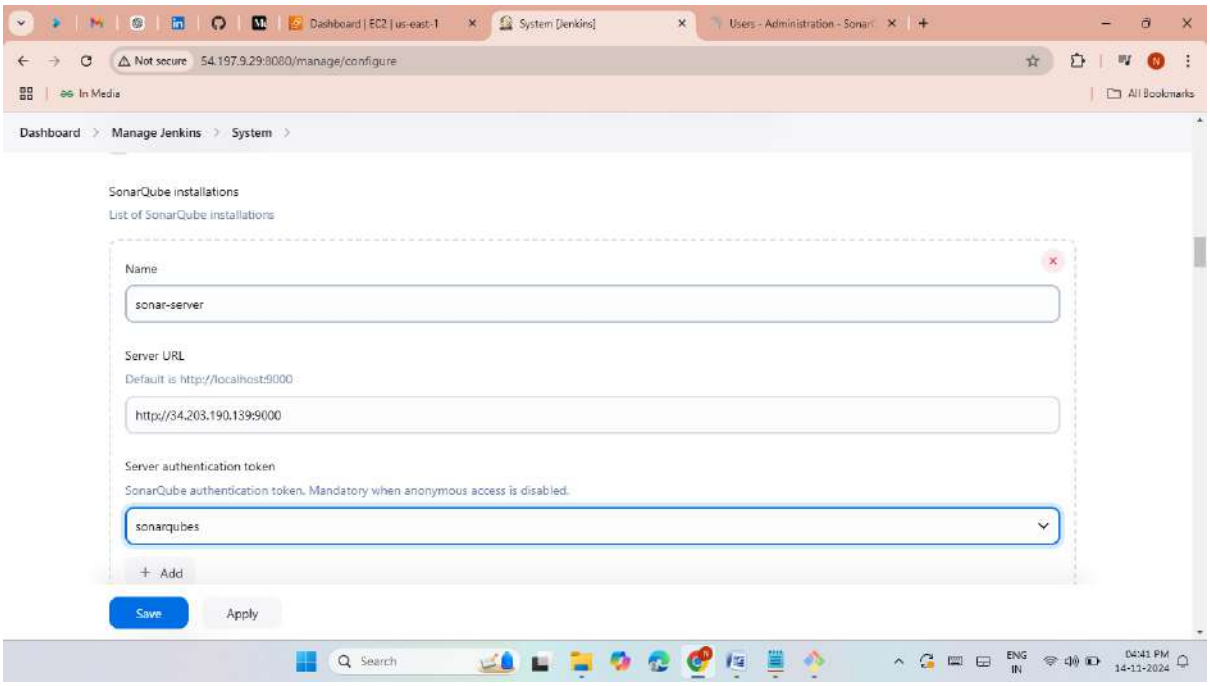
- Here the sonarqubes credentials was created.



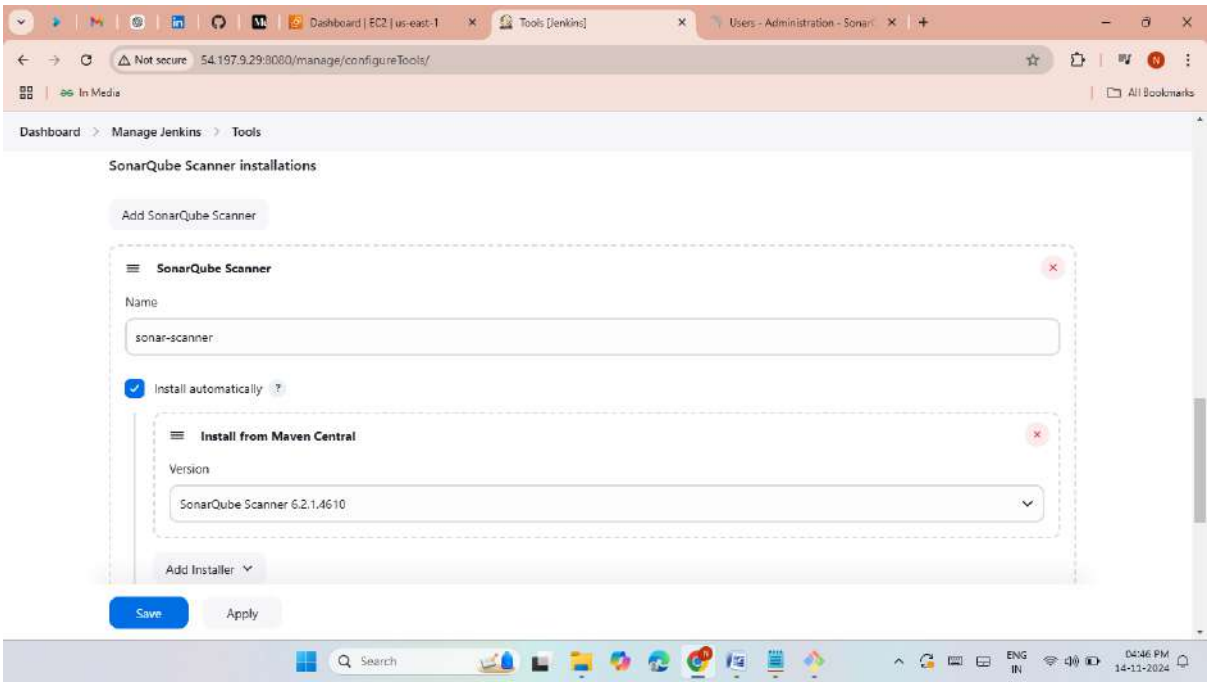
- Now install sonarqube scanner plugin.



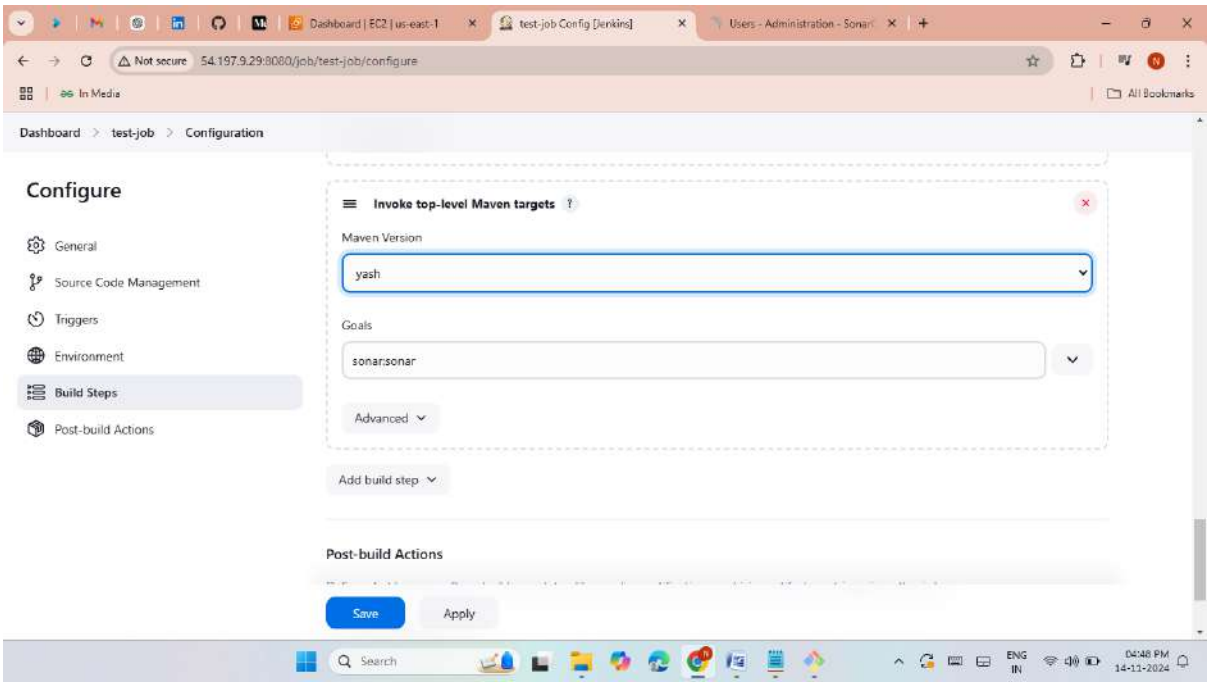
- Now go to system then add sonarqube tokes and server url.



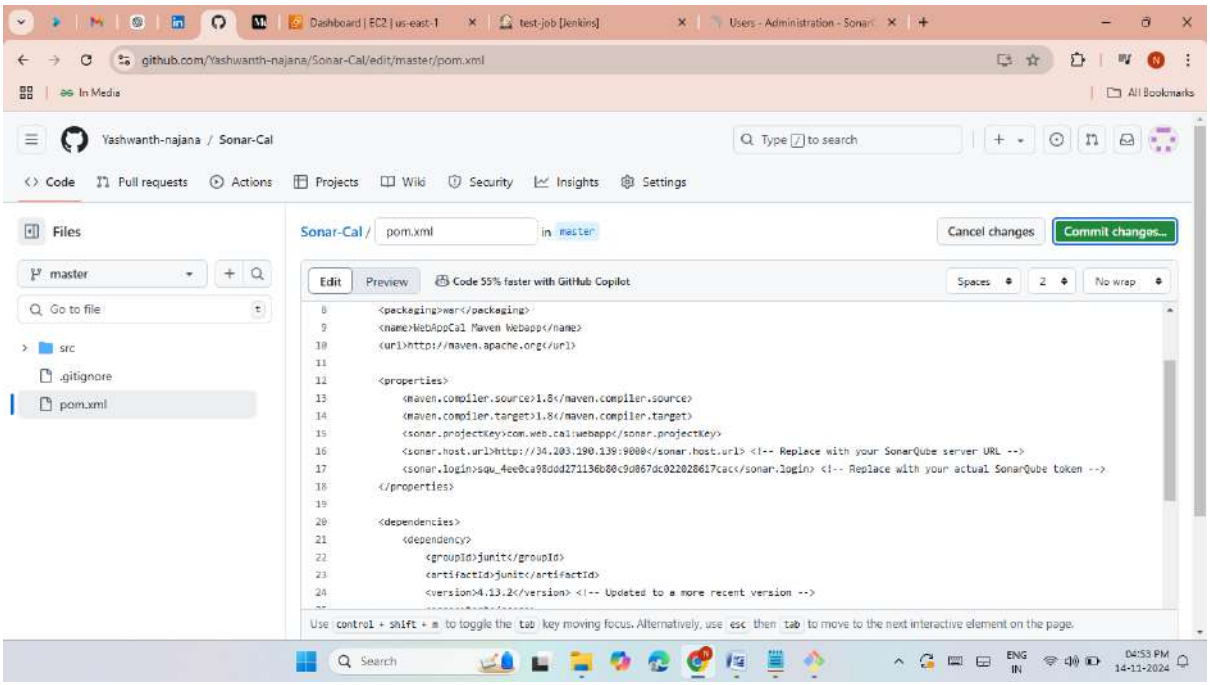
- Now go to tools then add sonarqube scanner.



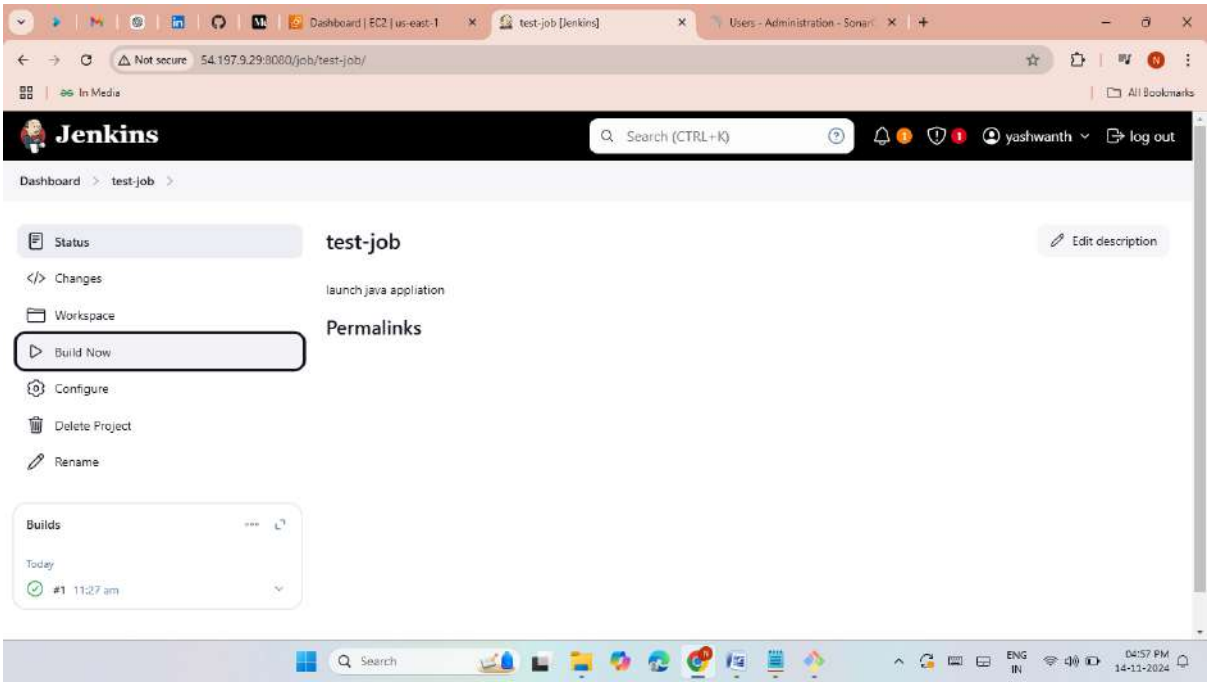
- Now create test-job then add maven version and gole for sonarqube in build step.



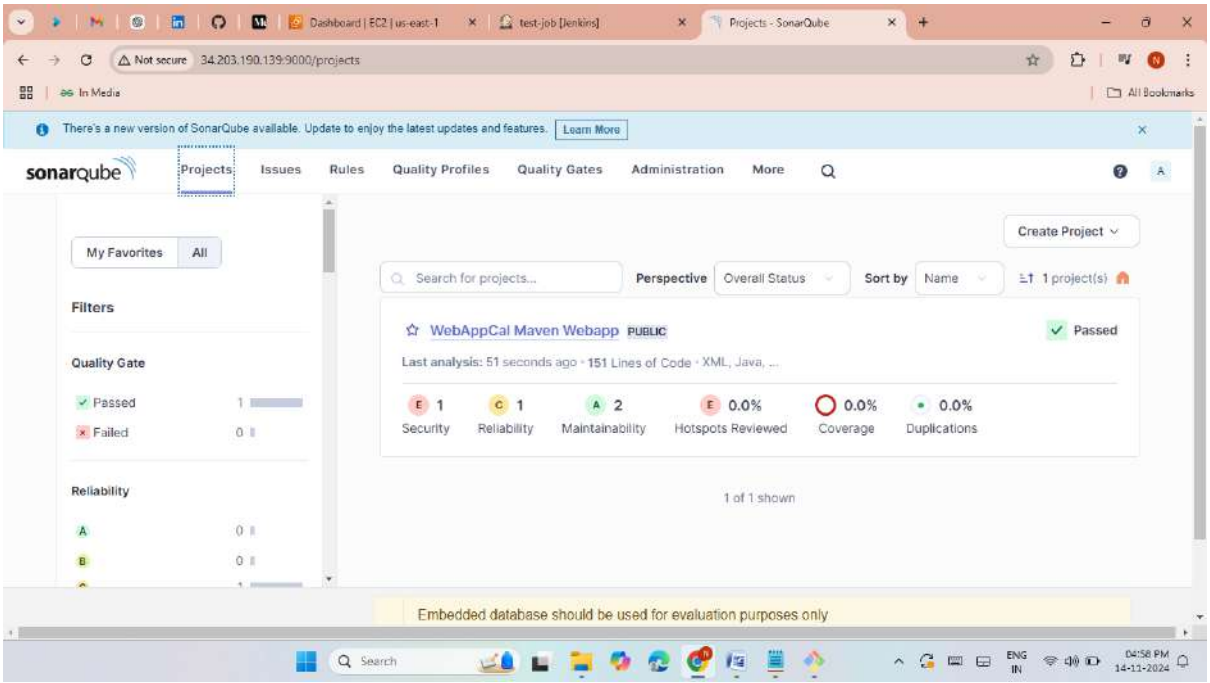
- Go to pom.xml file in github then give sonarqube ip address and sonarqube secret key.



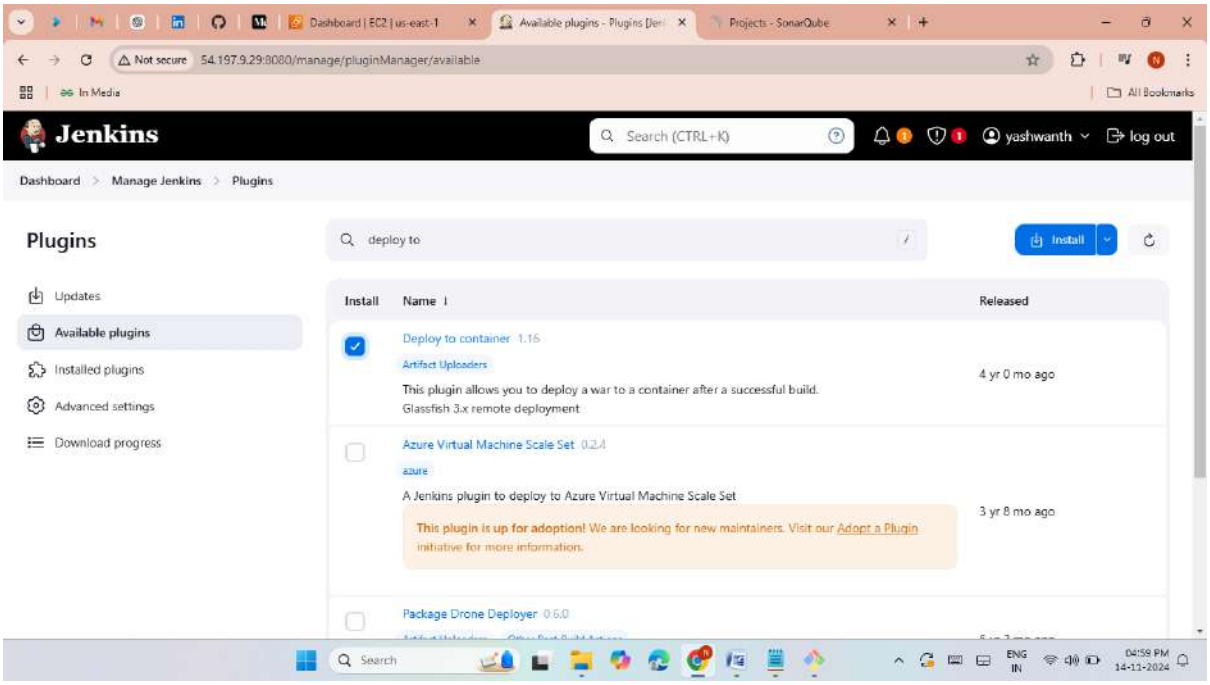
- Now click on build now.
- The test job was success.



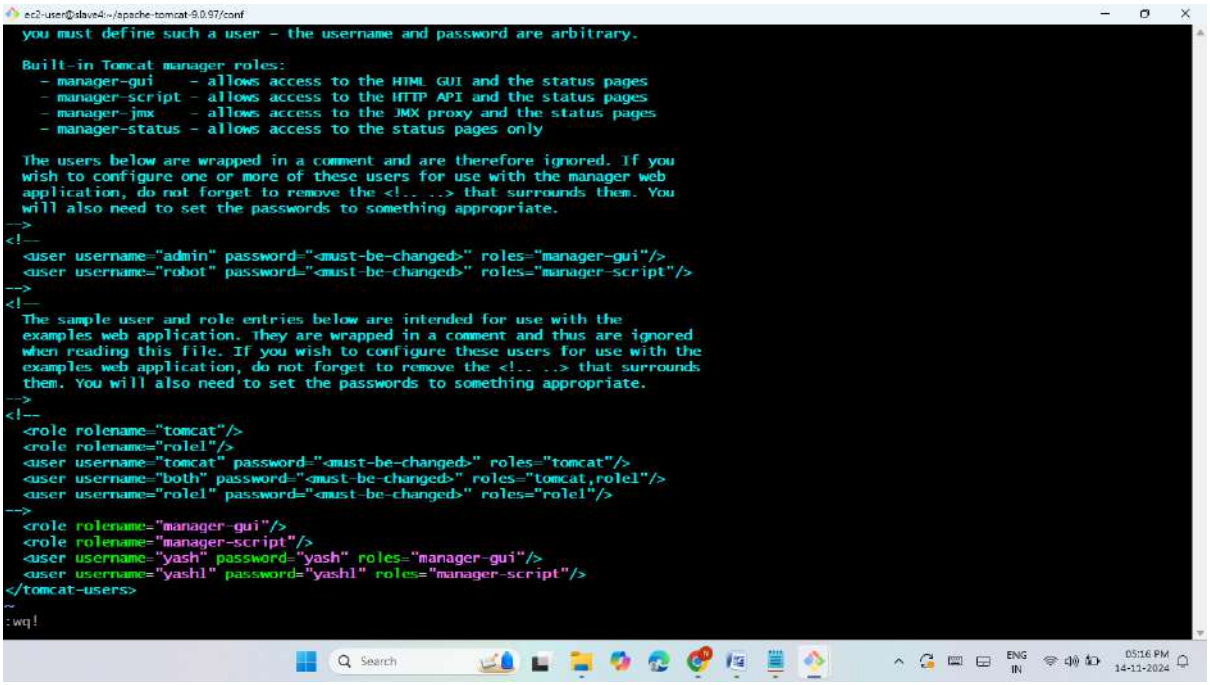
- Here we can see the result in sonarqubes.



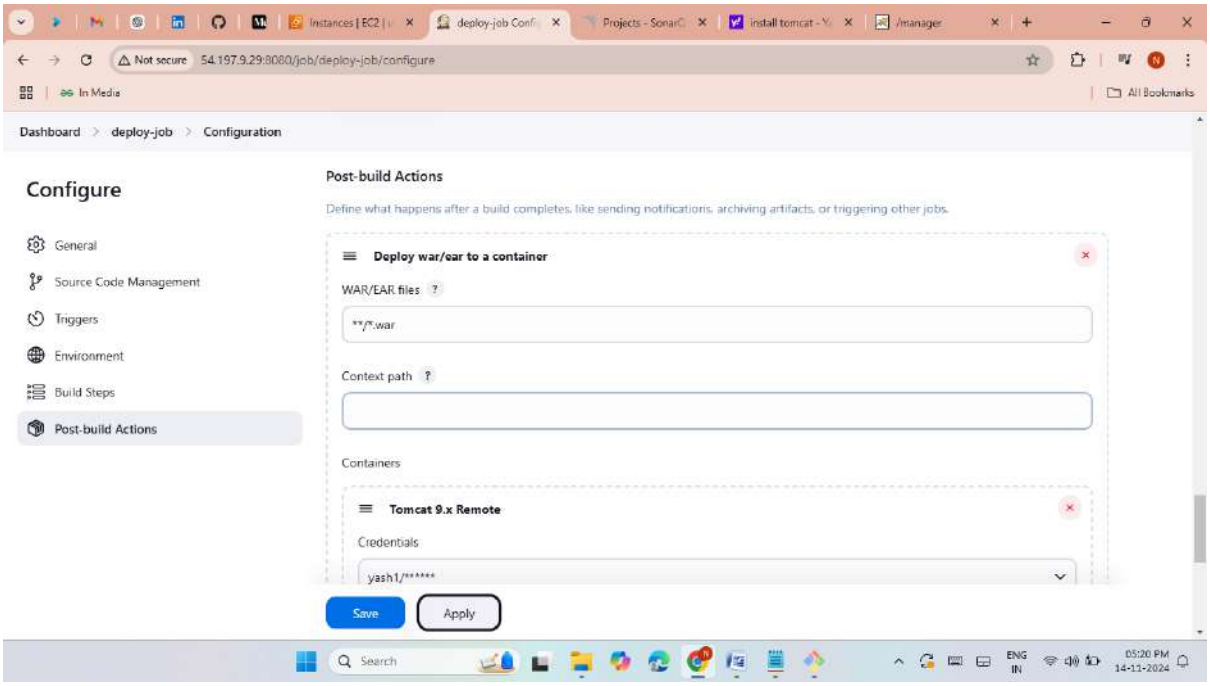
- Now install deploy to container plugin.

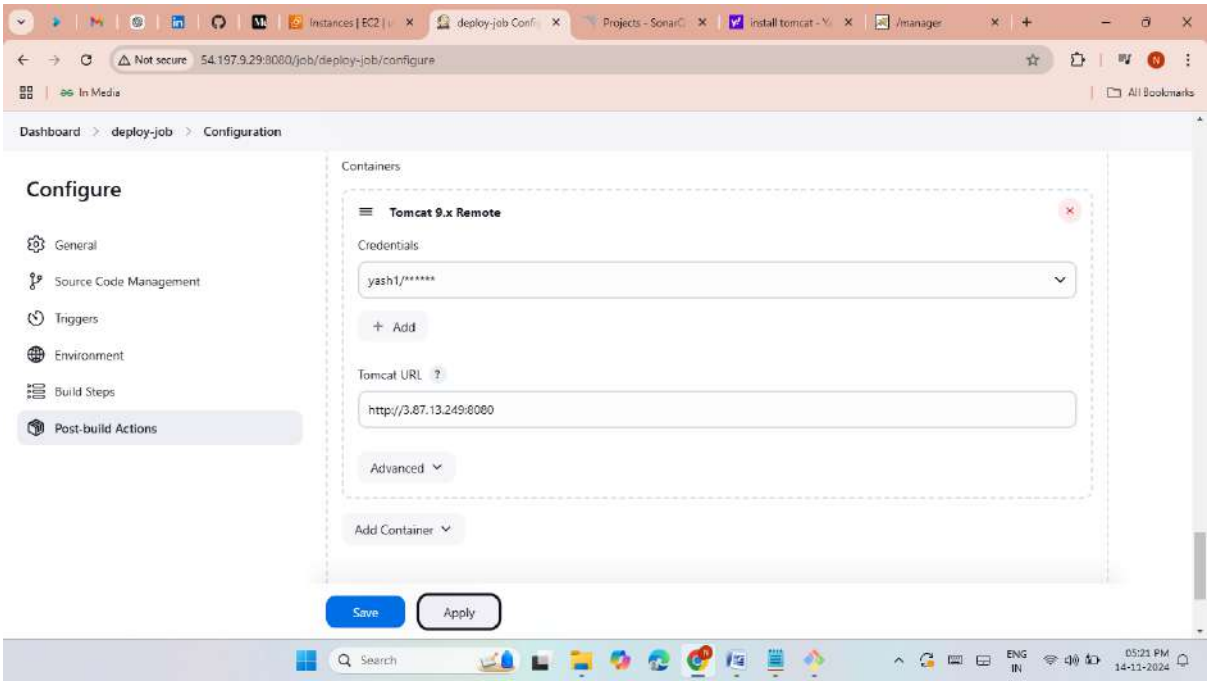


- Install tomcat in slave server then start the tomcat.
- Give roles and passwords in tomcat-user.xml
- Remove permissions in context.xml file.

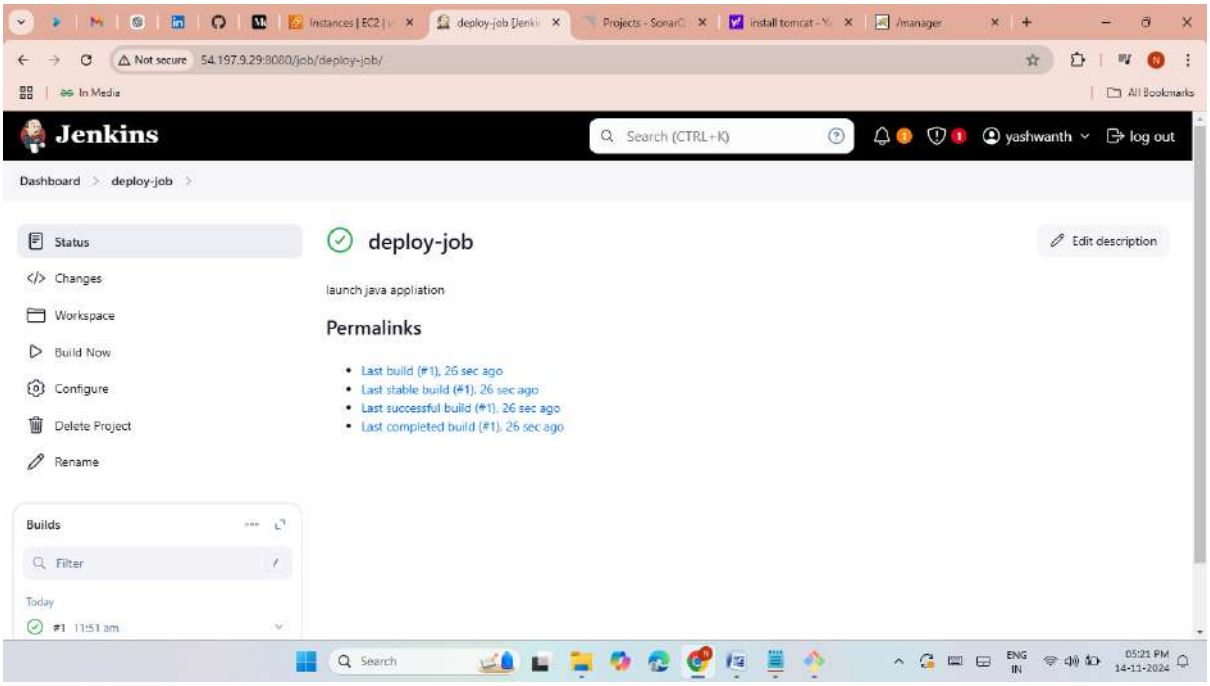


- Now create a deploy-job then add post-build action.
- Copy .war files and select tomcat version and add credentials (manager-script).
- Now add tomcat URL then click on save.

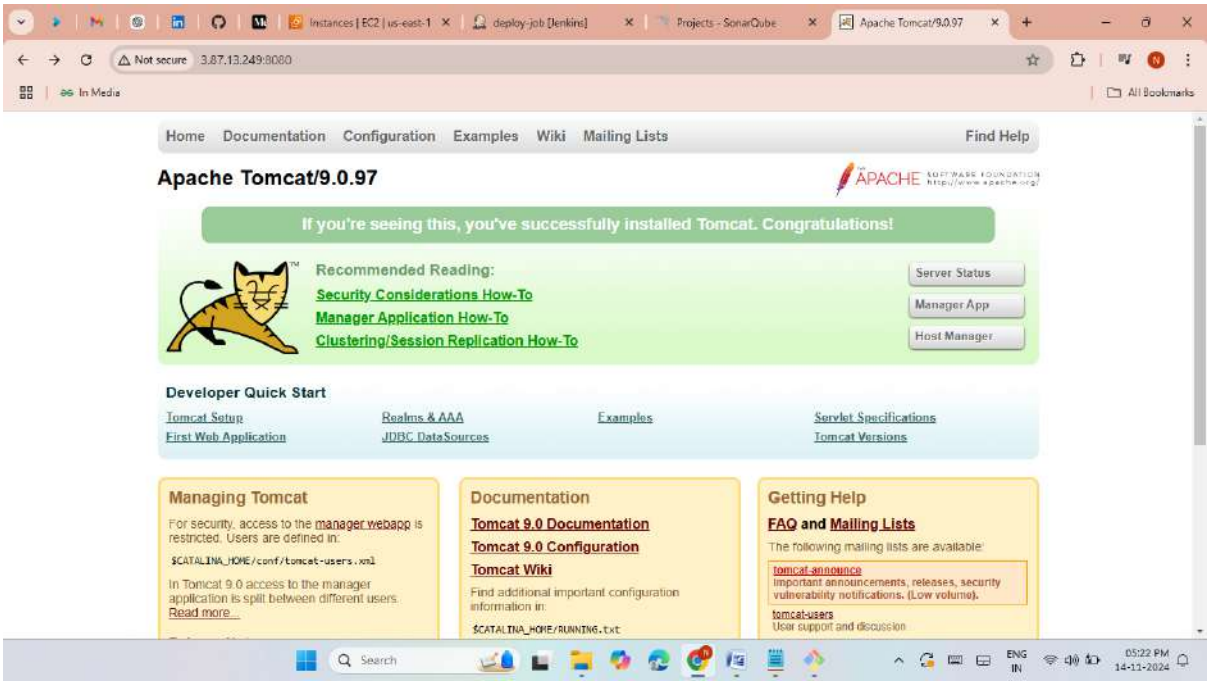




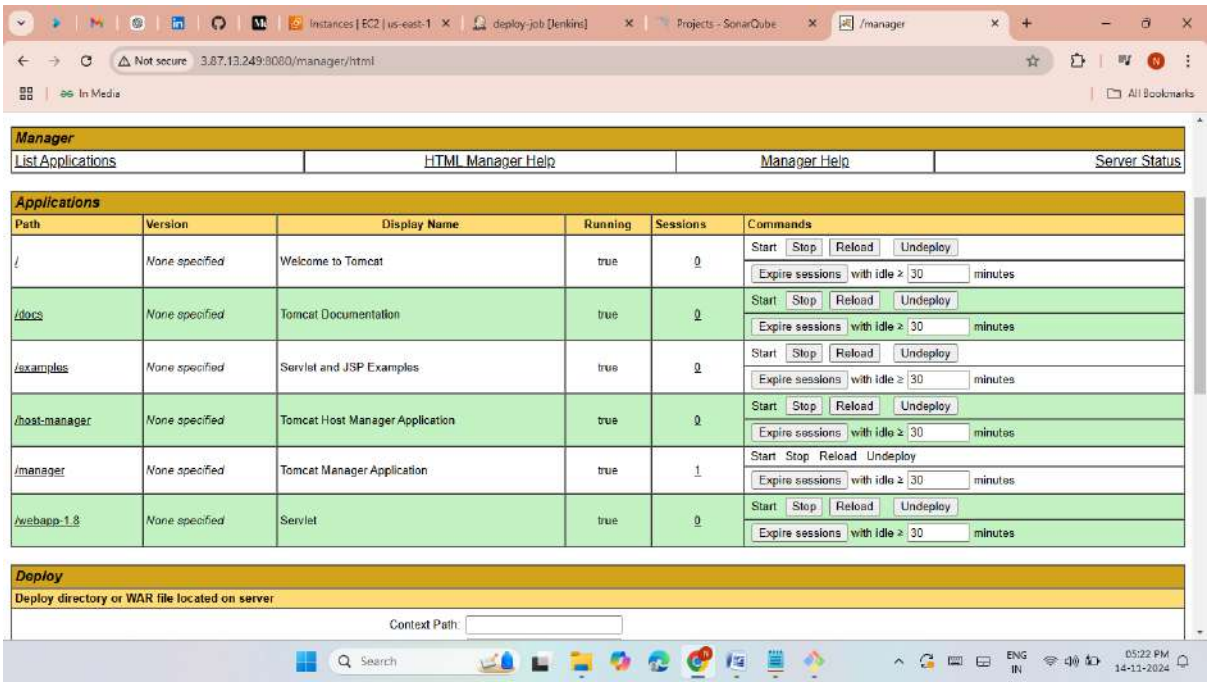
- Now click on build now.
- Here the job was success.



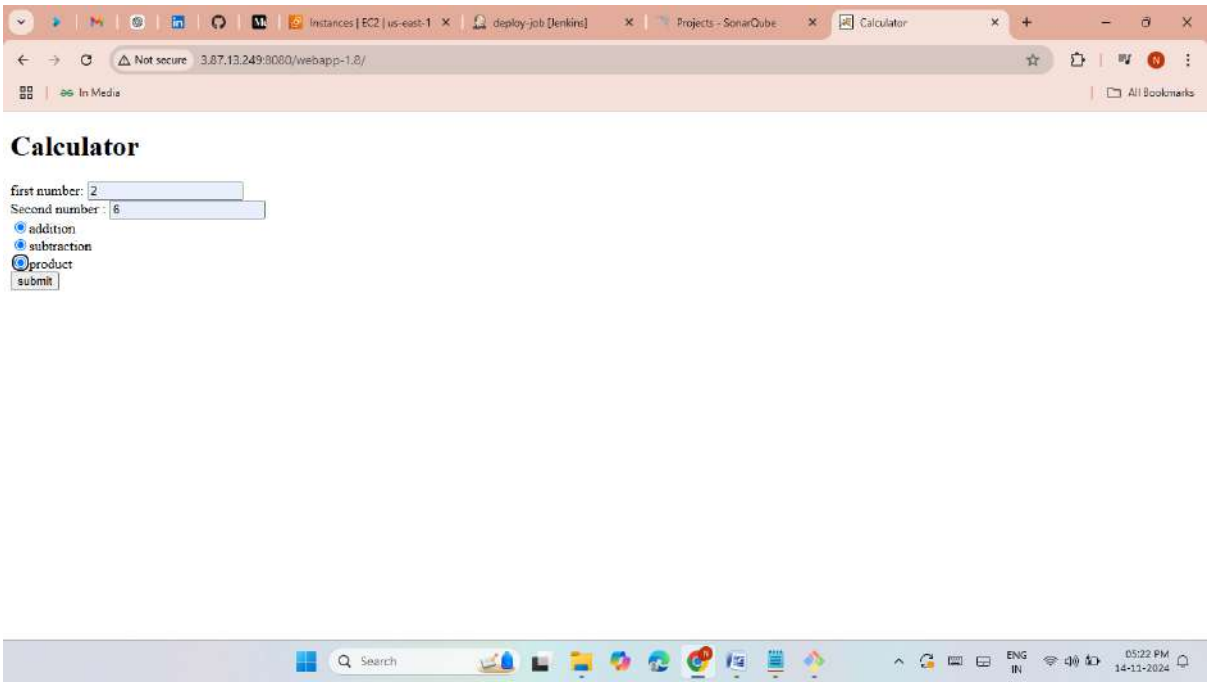
- Now copy the slave4 public ip and past it on google browser.
- The tomcat was hosted successfully.
- Click on manager app.

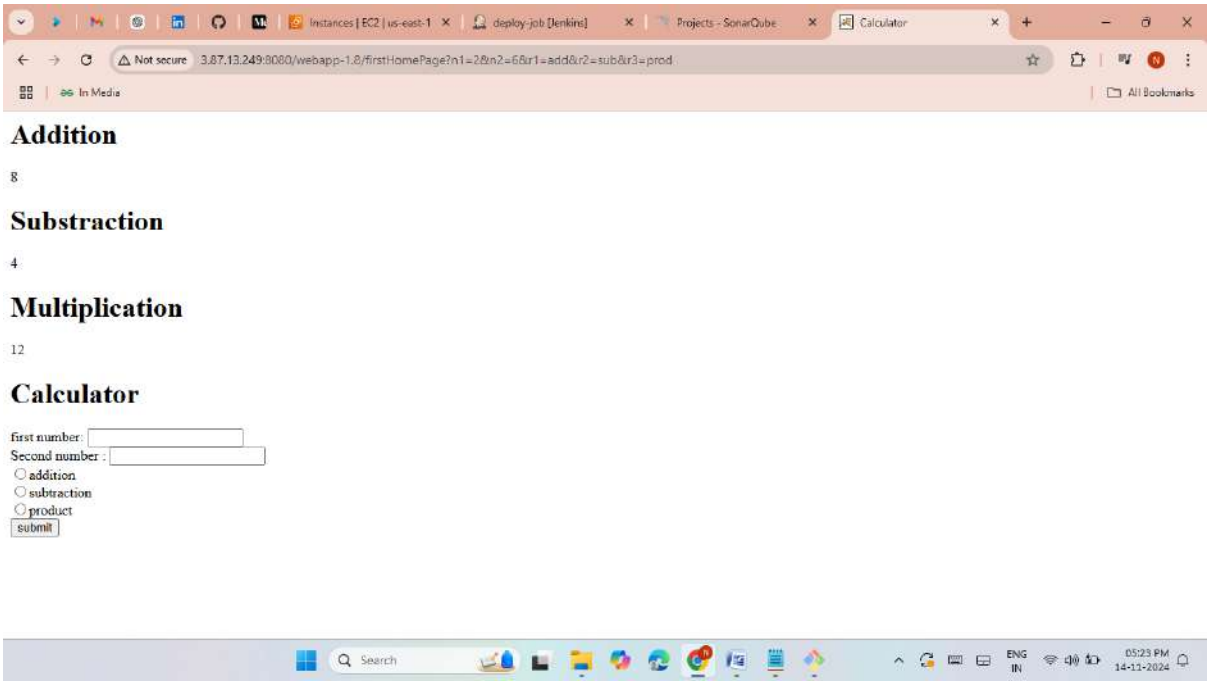


- Now click on web-apps.1.8.

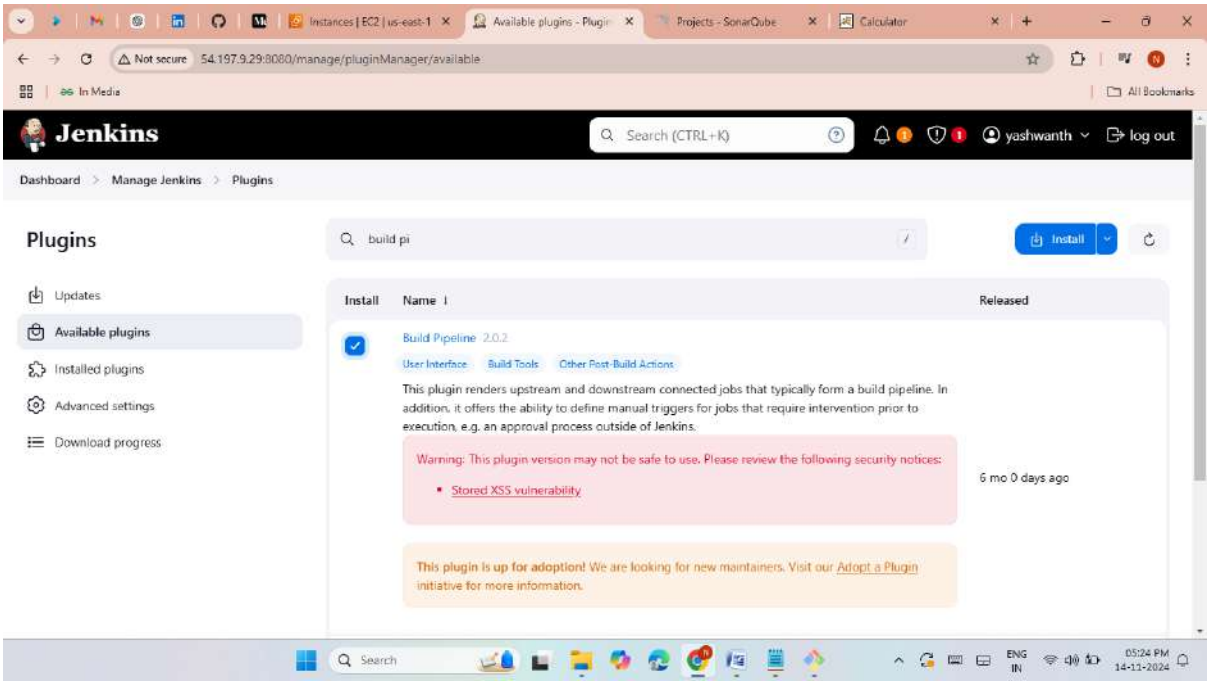


- Here the java web calculator was hosted successfully.

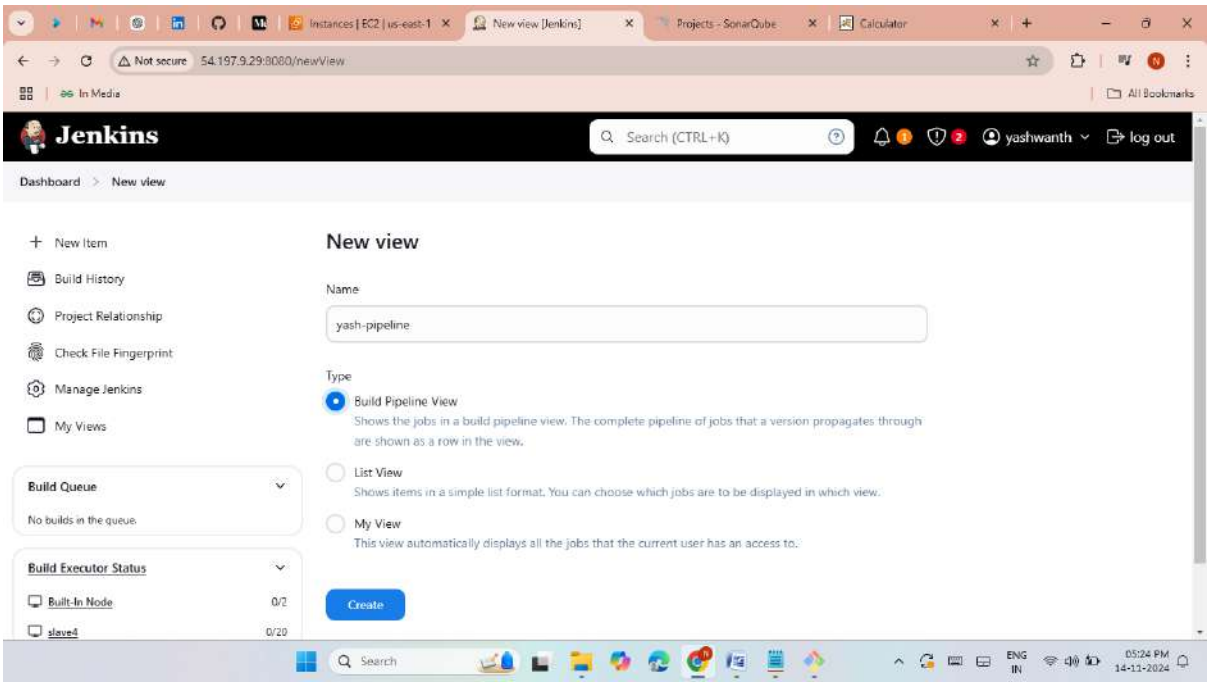




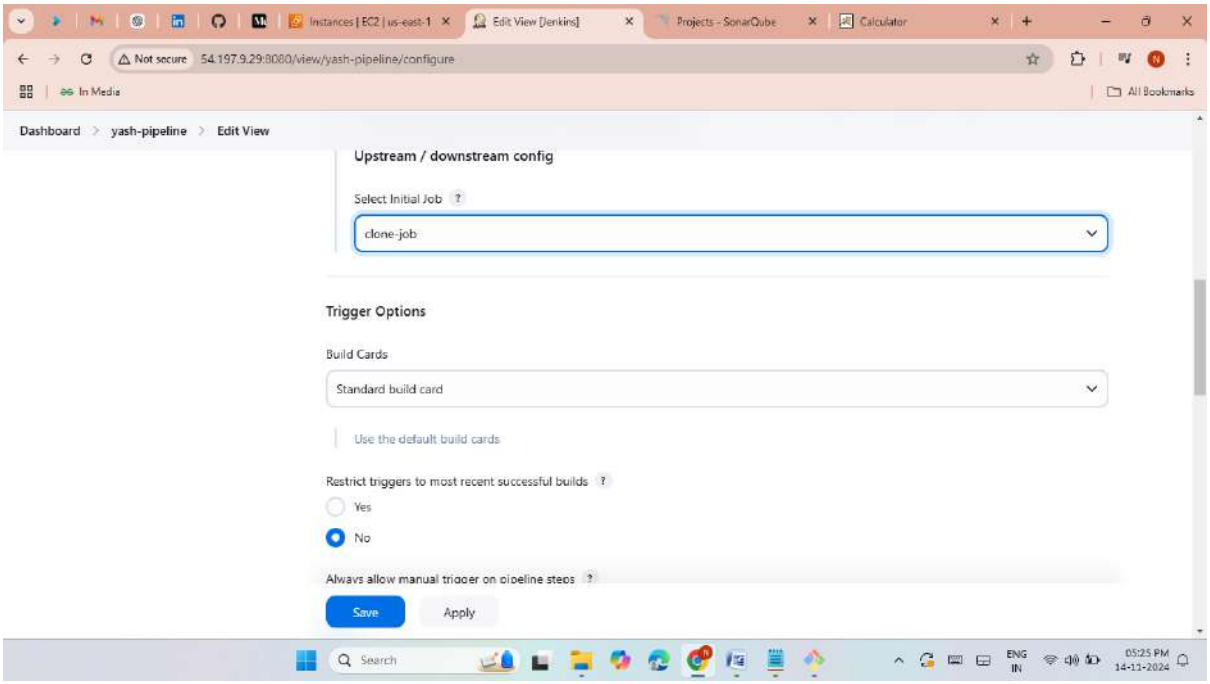
- Now install build pipeline plugin to create pipeline.



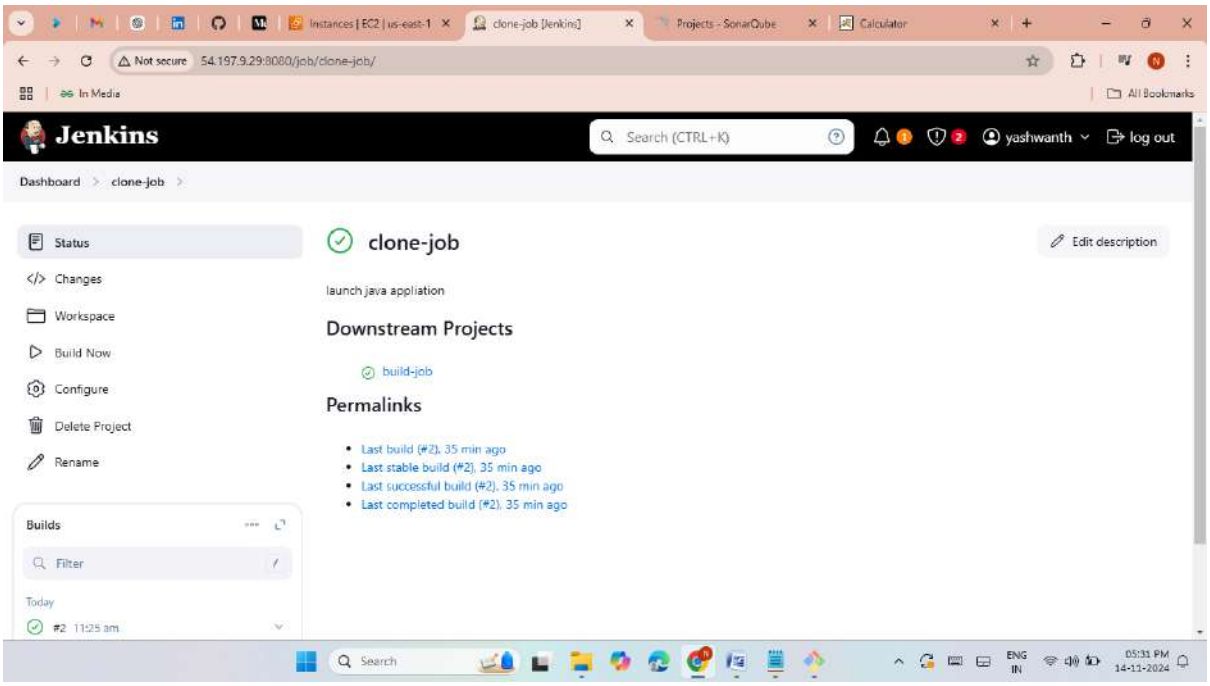
- Create a pipeline for all the jobs.



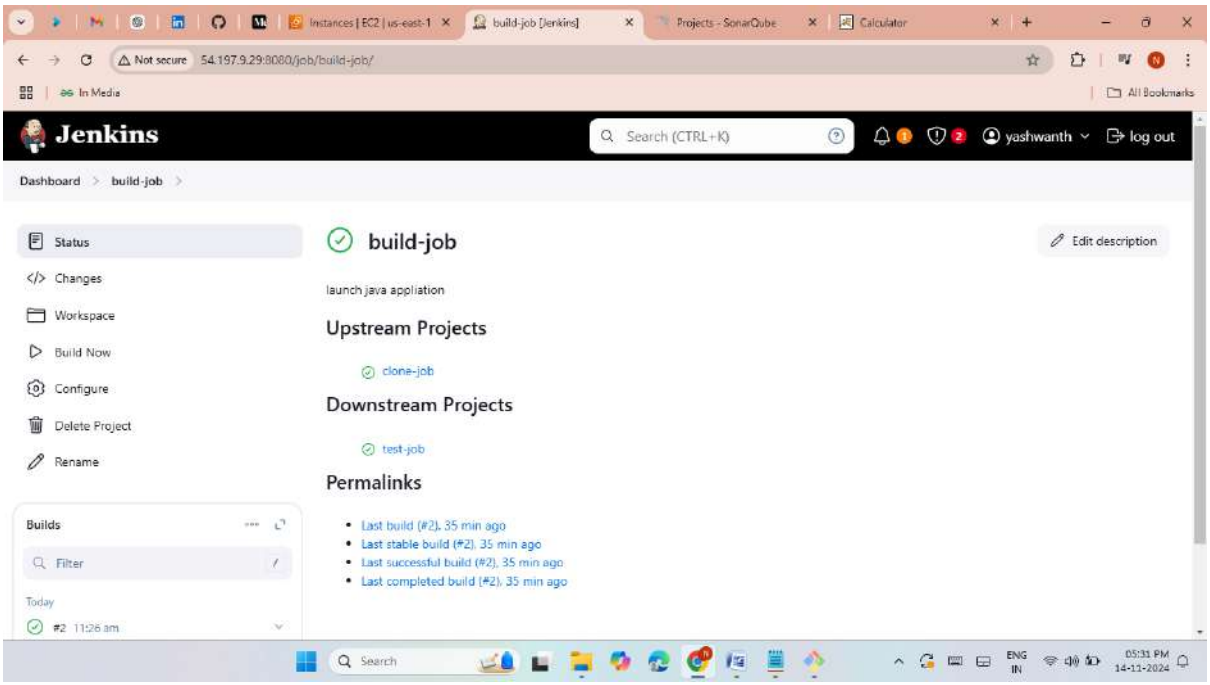
- Select initial job.



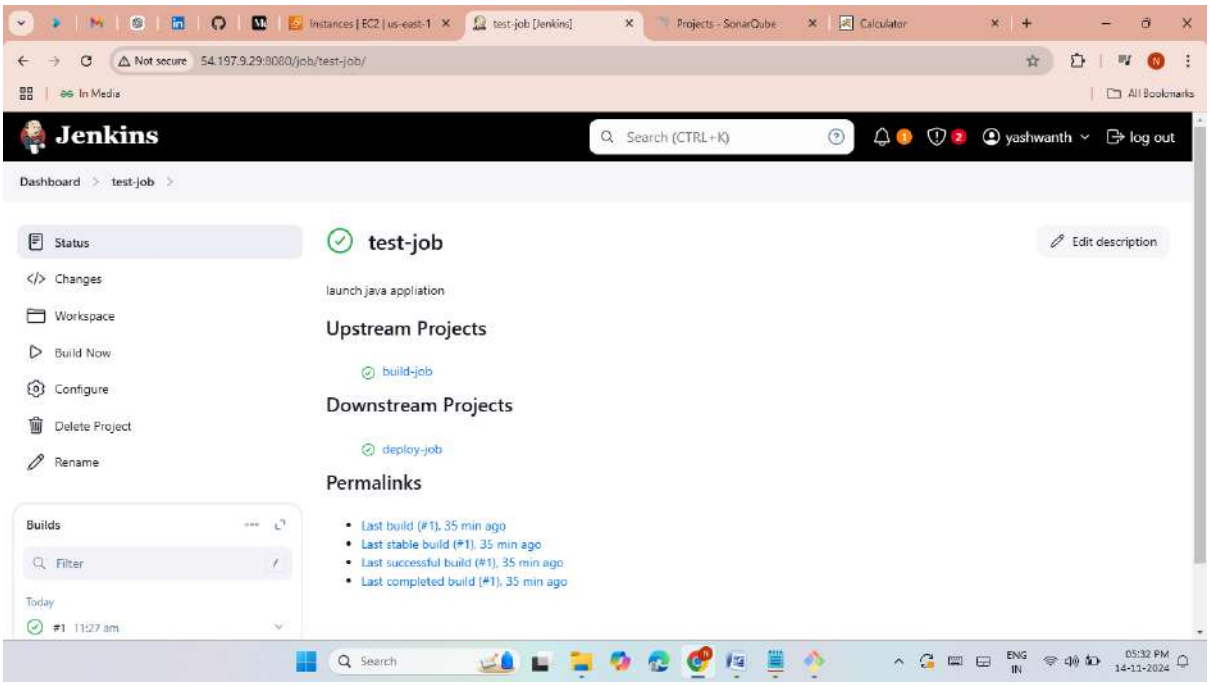
- Go to clone job add downstream project.



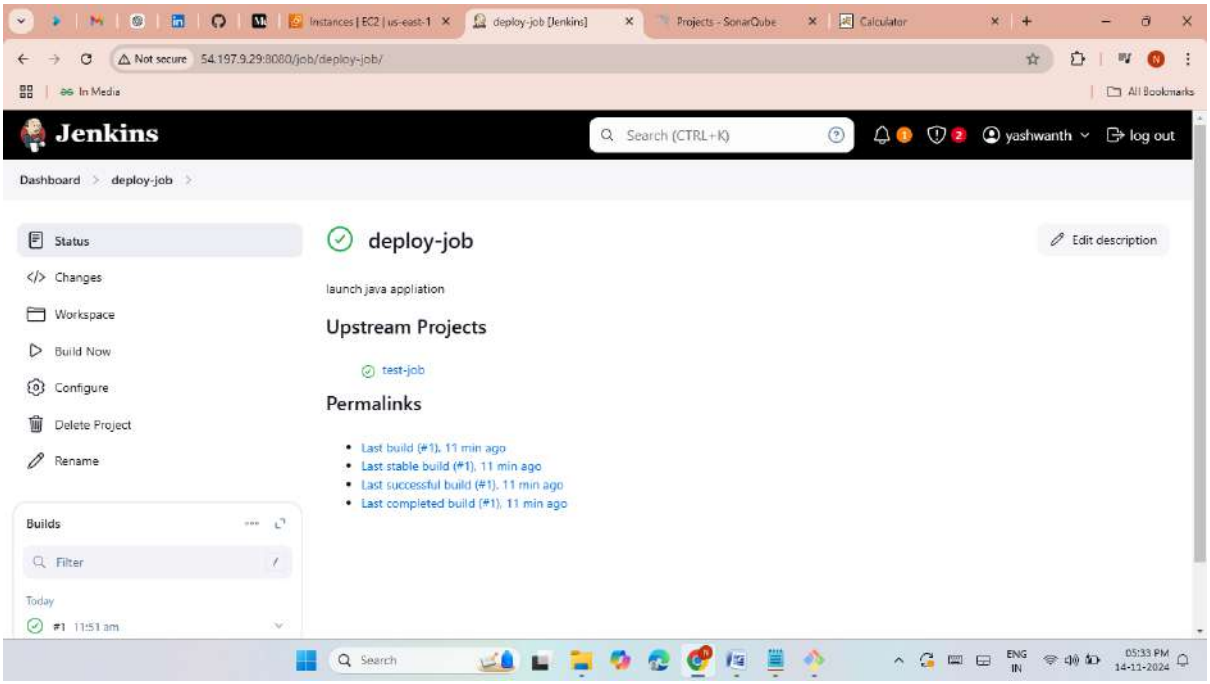
- Go to build job add downstream project and upstream project.



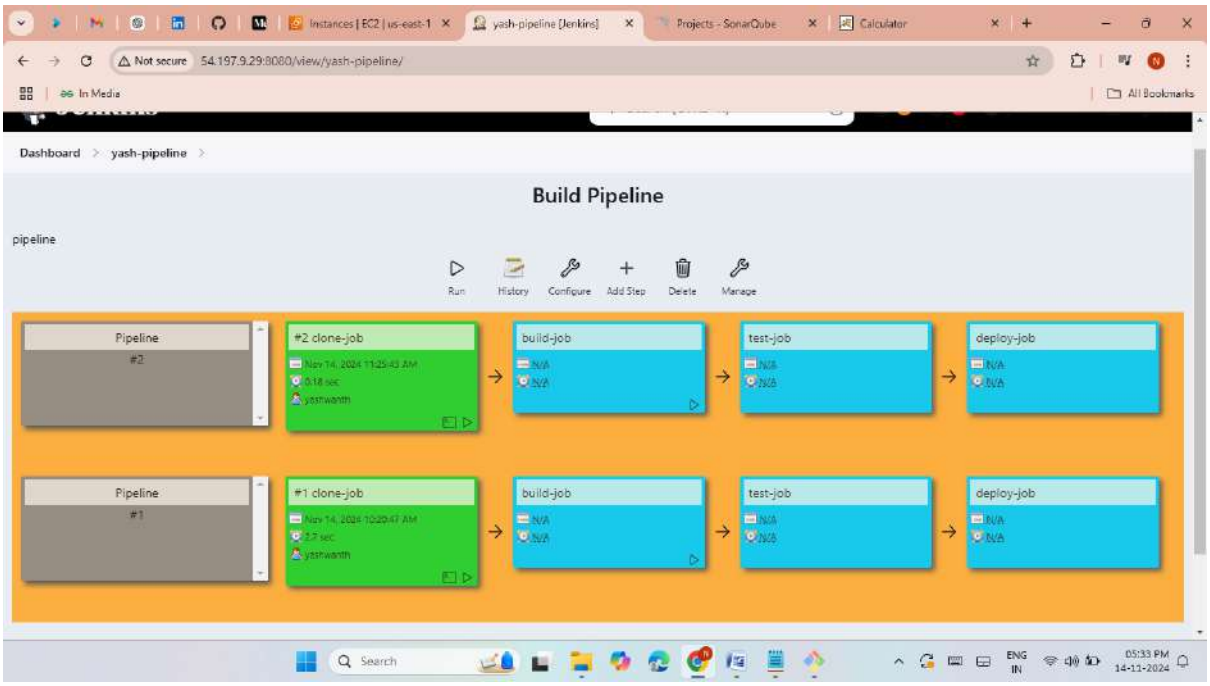
- Go to test job add downstream project and upstream project.



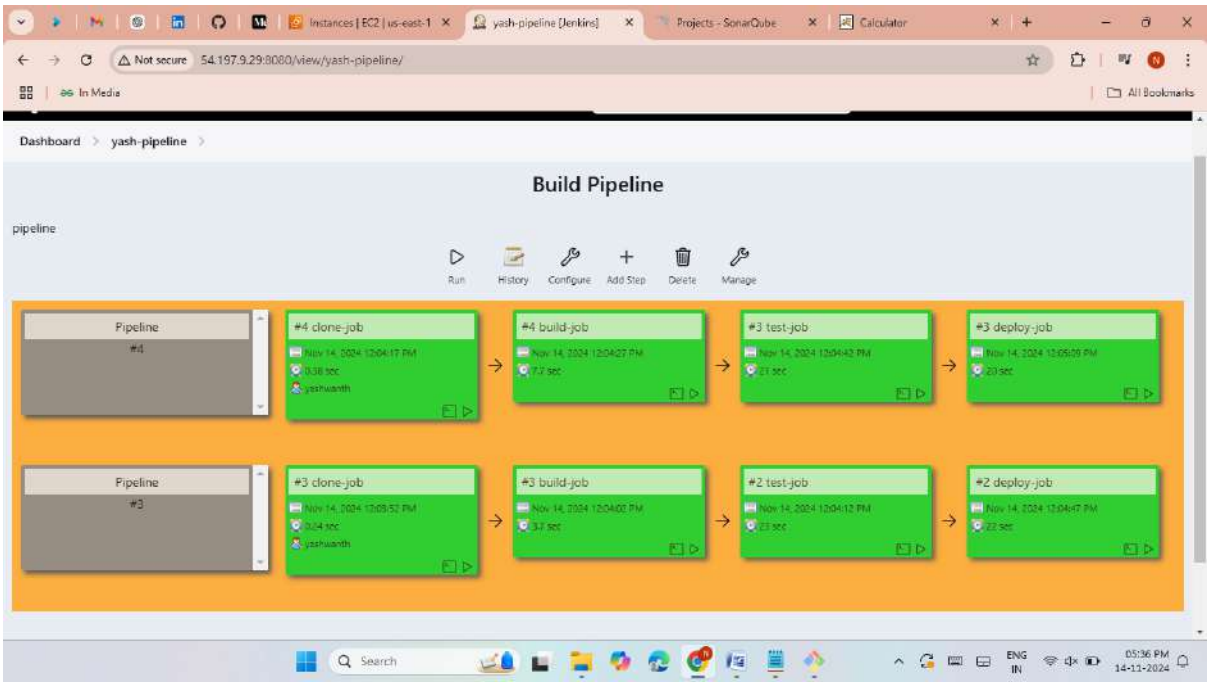
- Go to build job add upstream project.



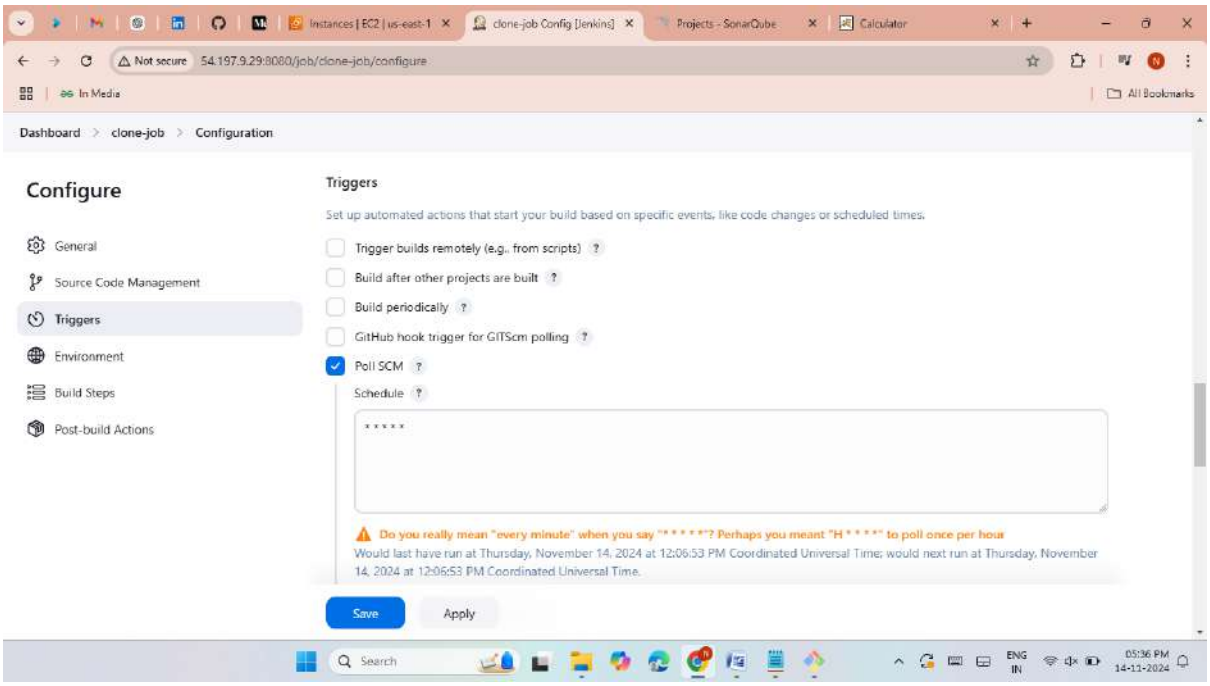
- Here the pipeline was successfully created.



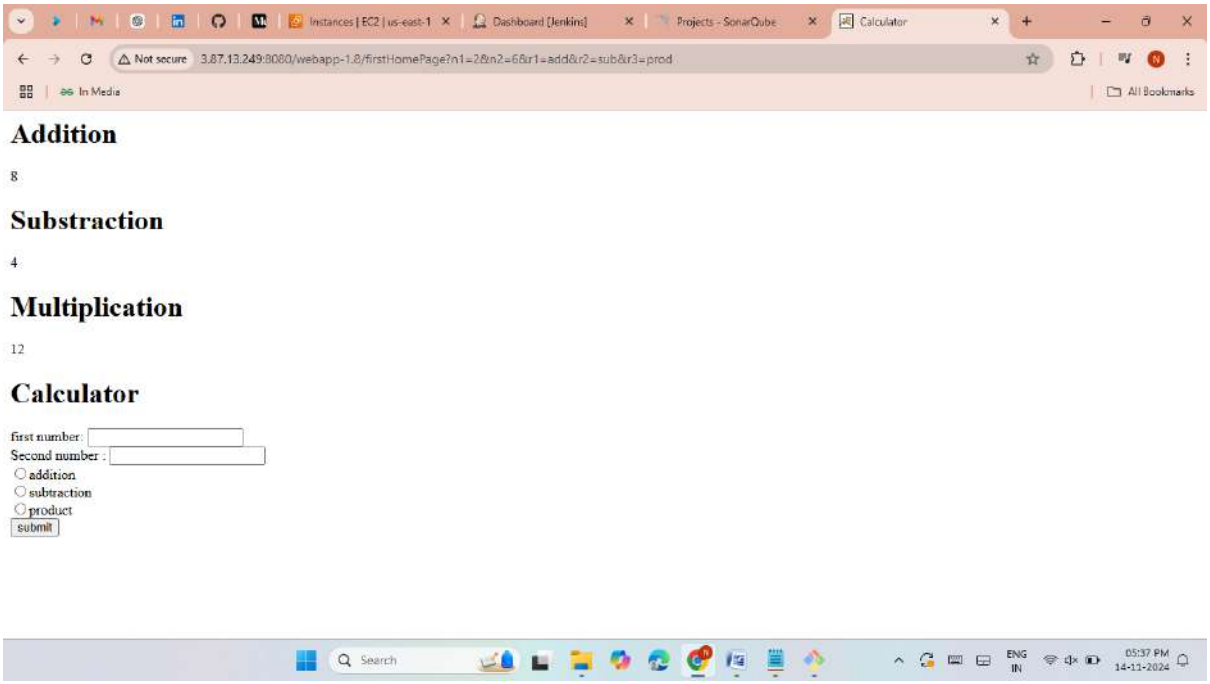
- Click on run the jobs was run onebyone.



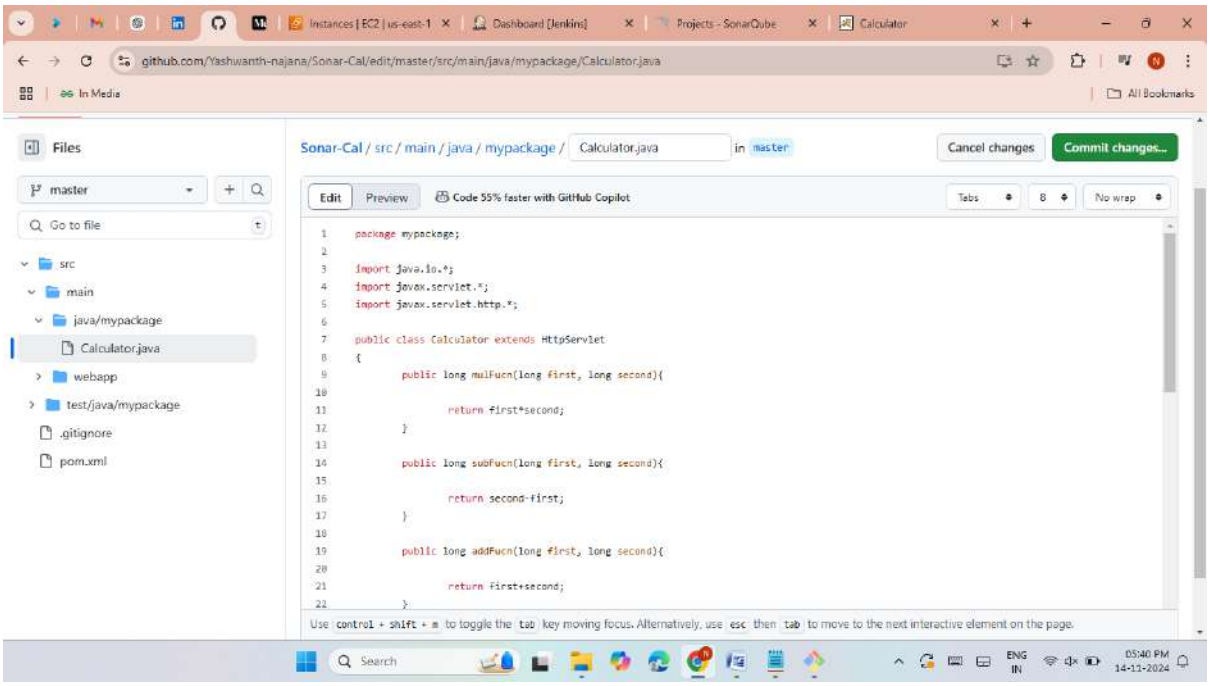
- Now add poll SCM.



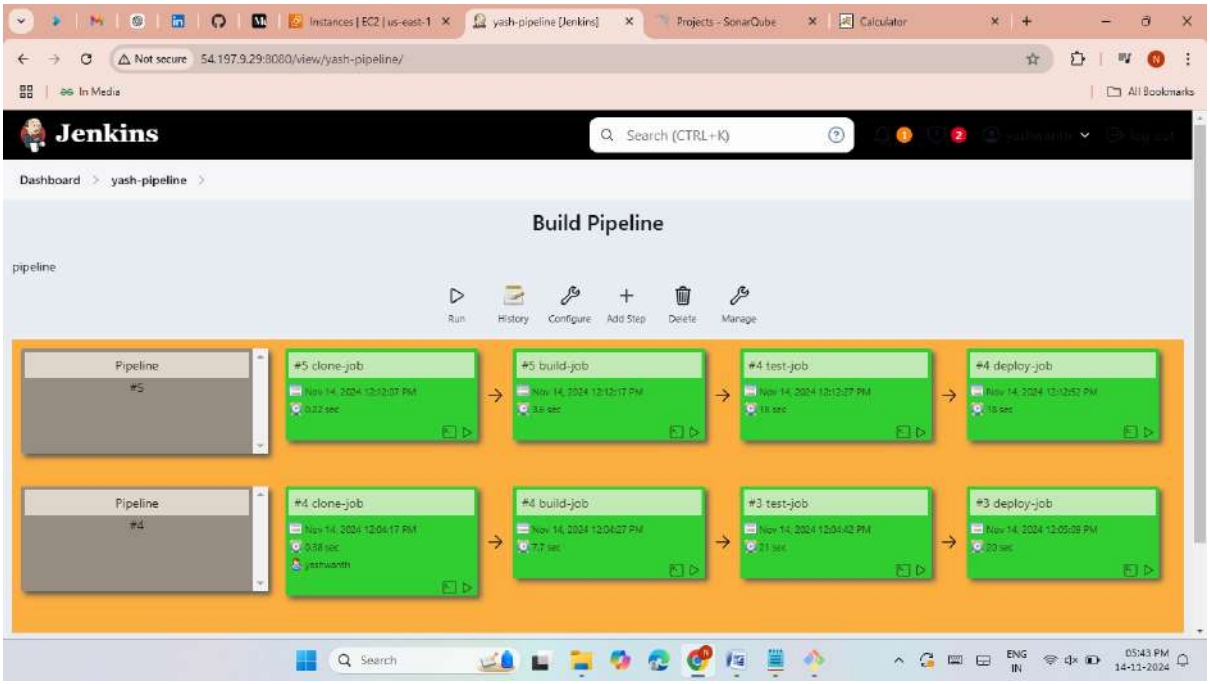
- Here the main output.



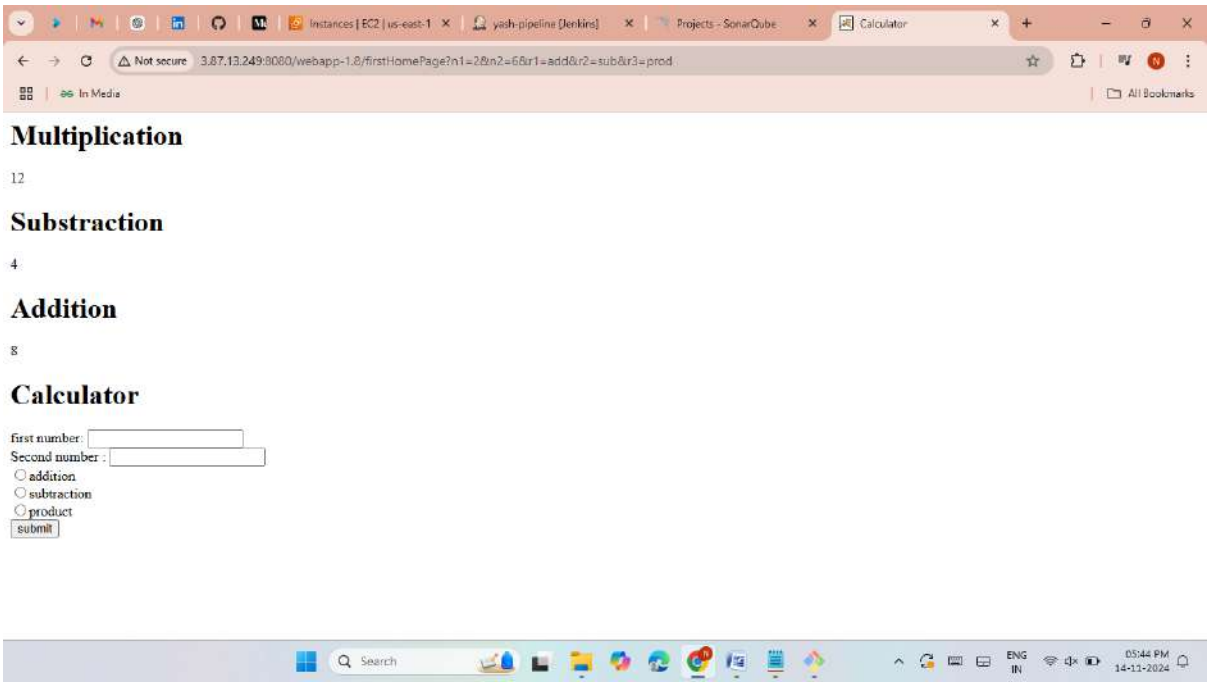
- I have changed the code in github.



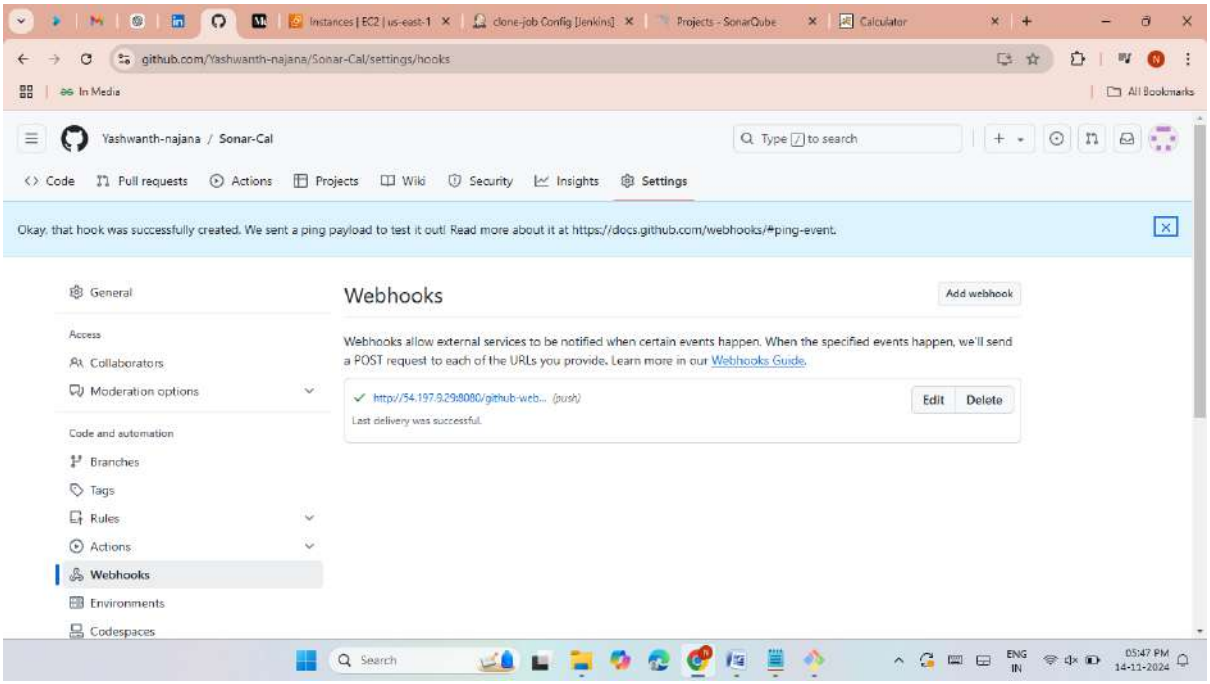
- After 1 min the job was run automactally because of poll scm.

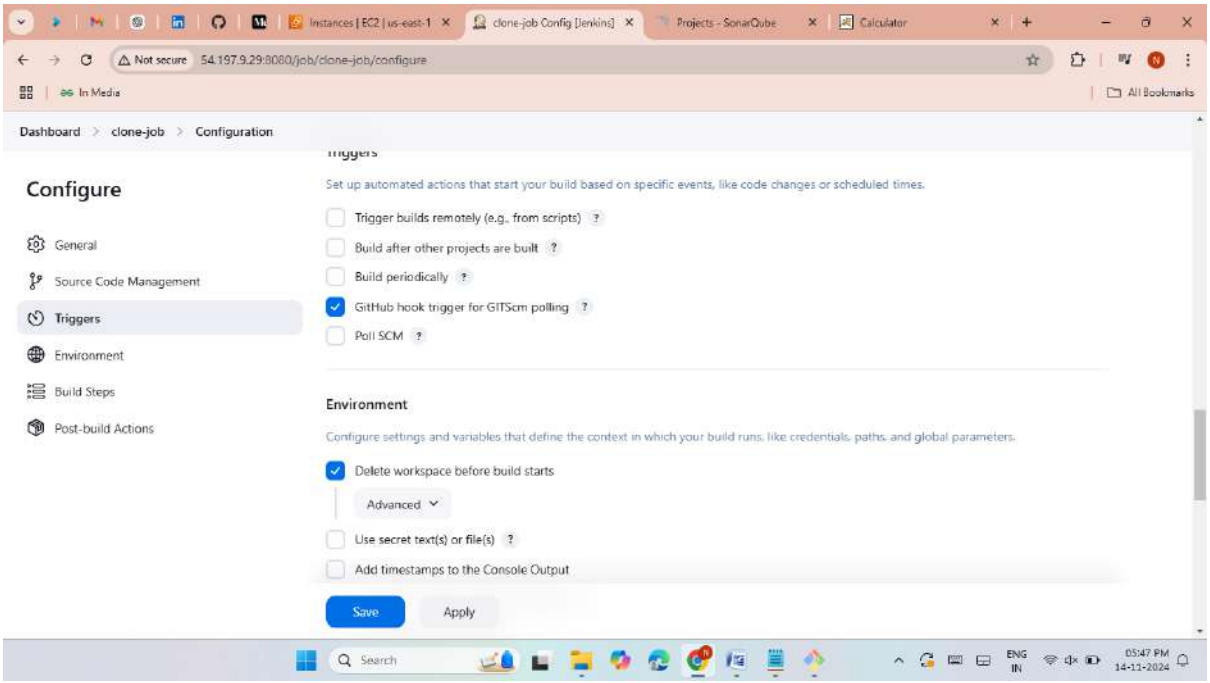


- Here the java calculator was changed successfully.

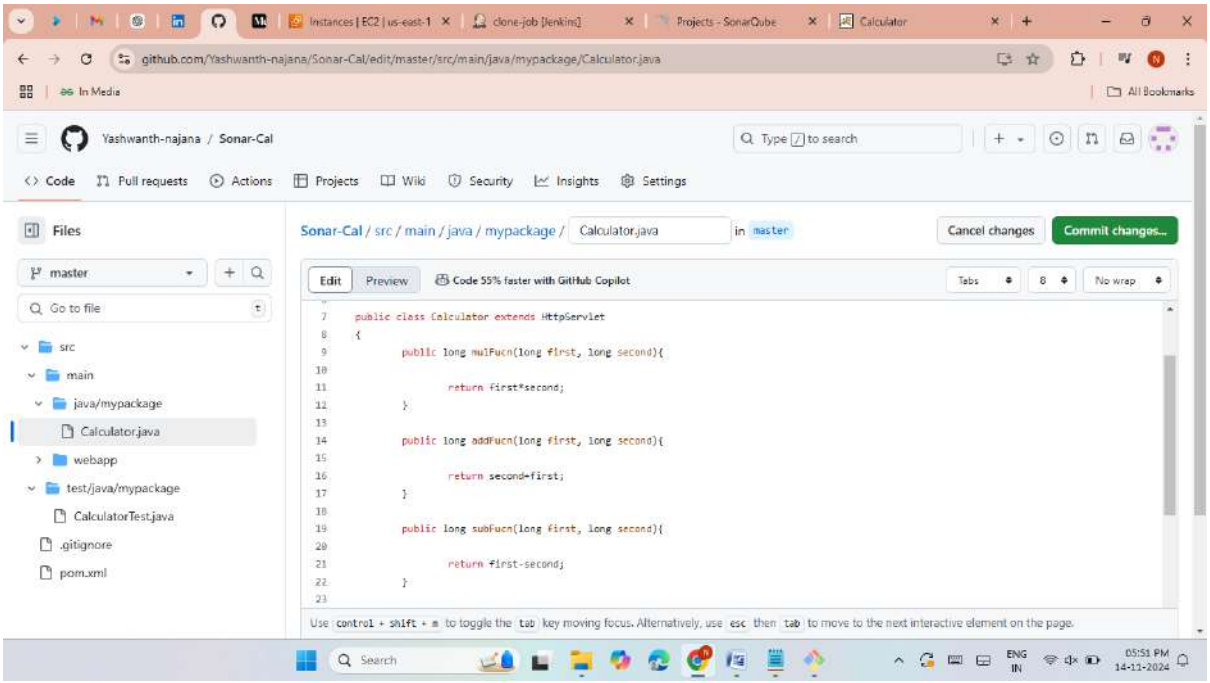


- Now I have added webhooks.

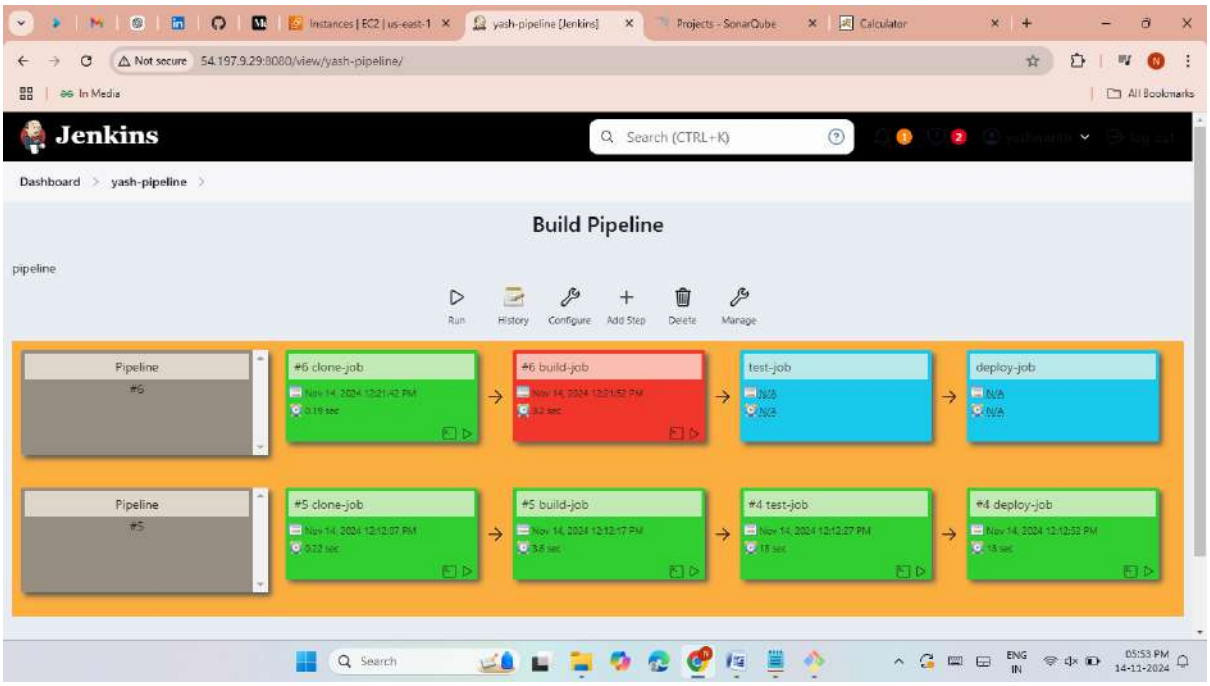




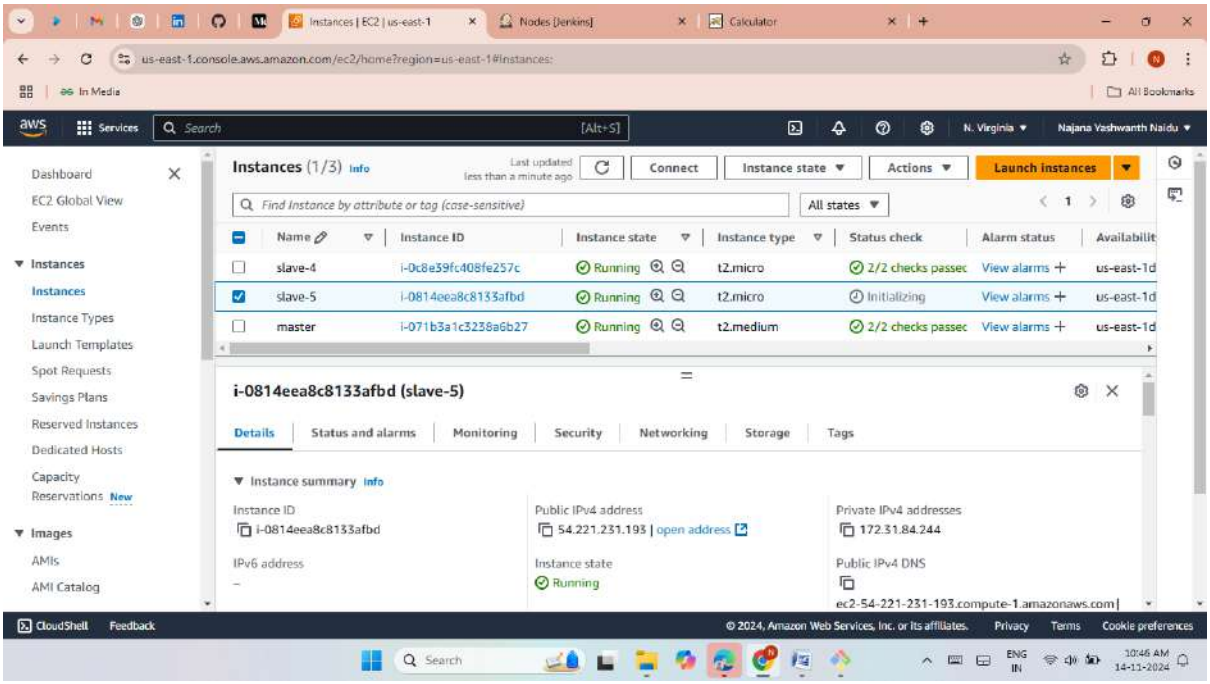
- Now change the code in github.



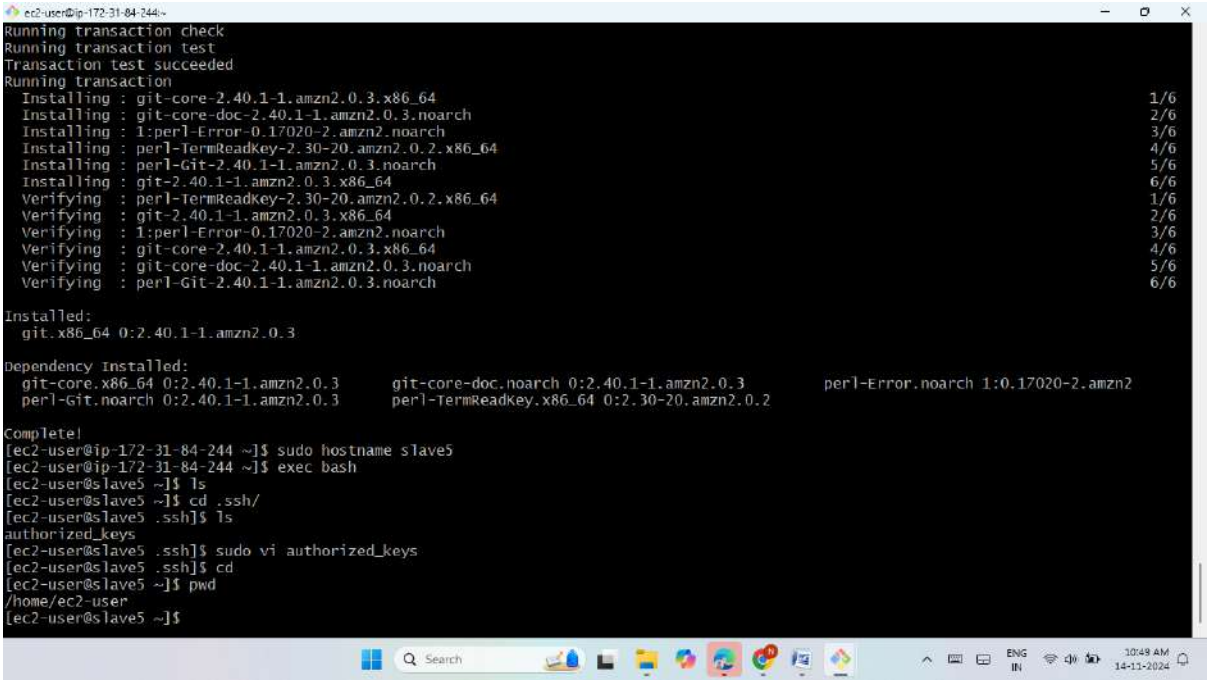
- Here the jobs was changed with in seconds because of webhooks.



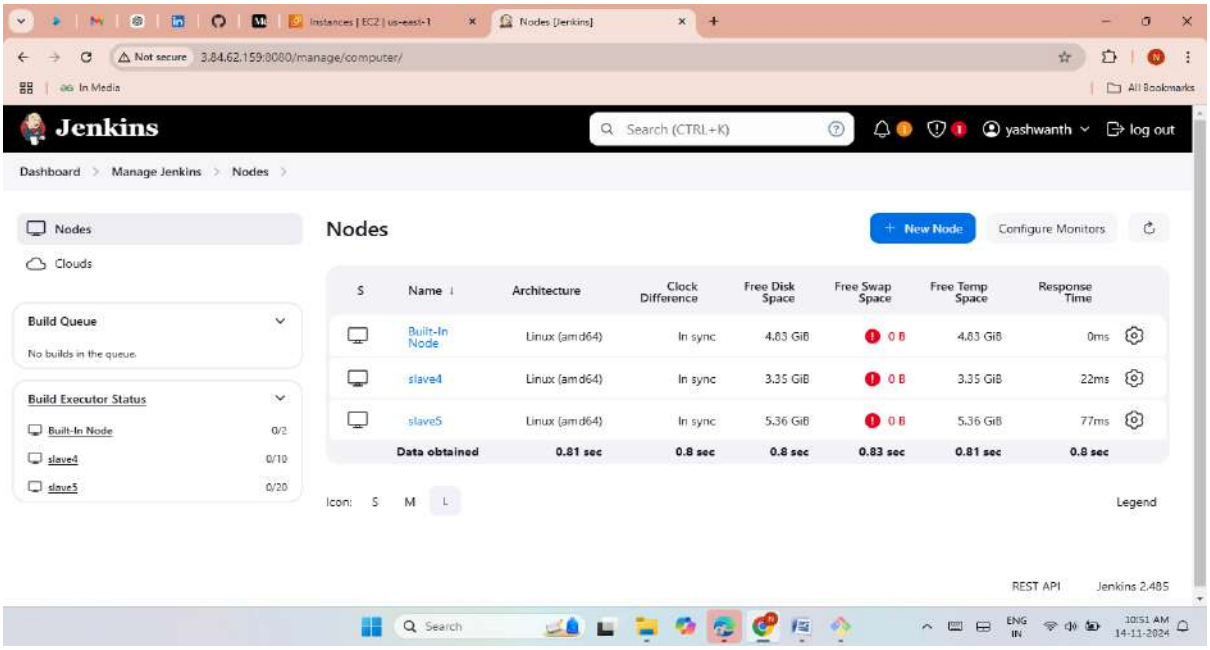
- Now launch an ec2 instances and give name as slave5.



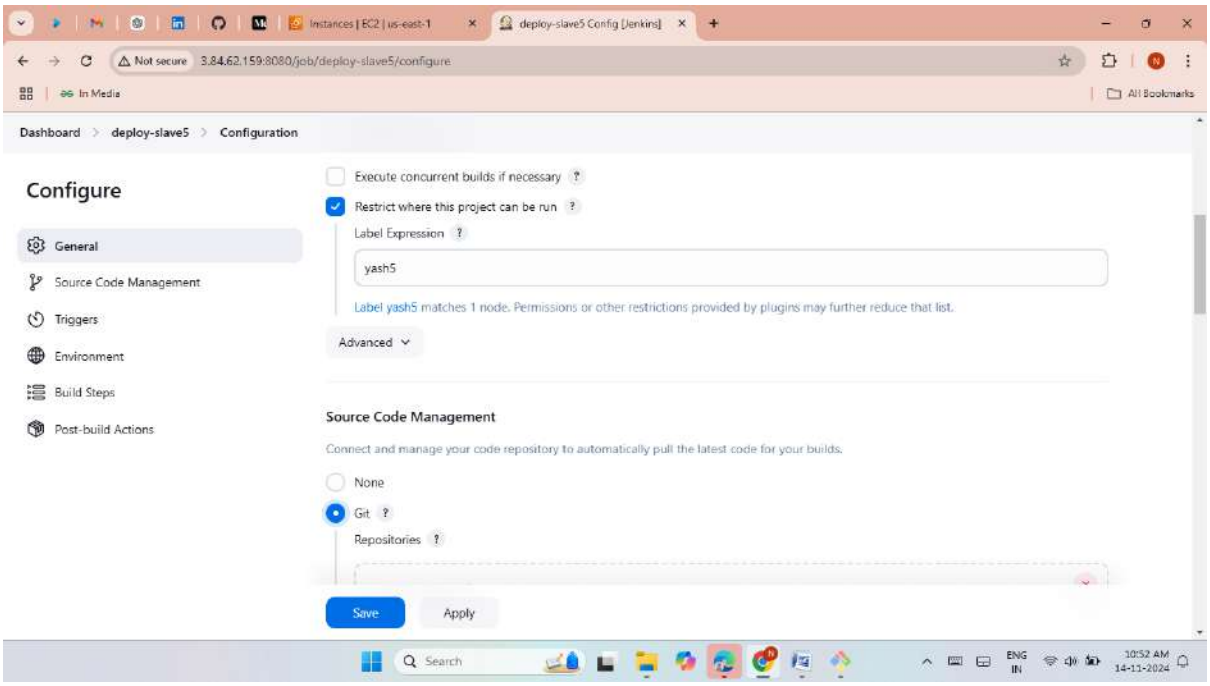
- Now connect Slave5 then install java.
- Now give private keys from master server.



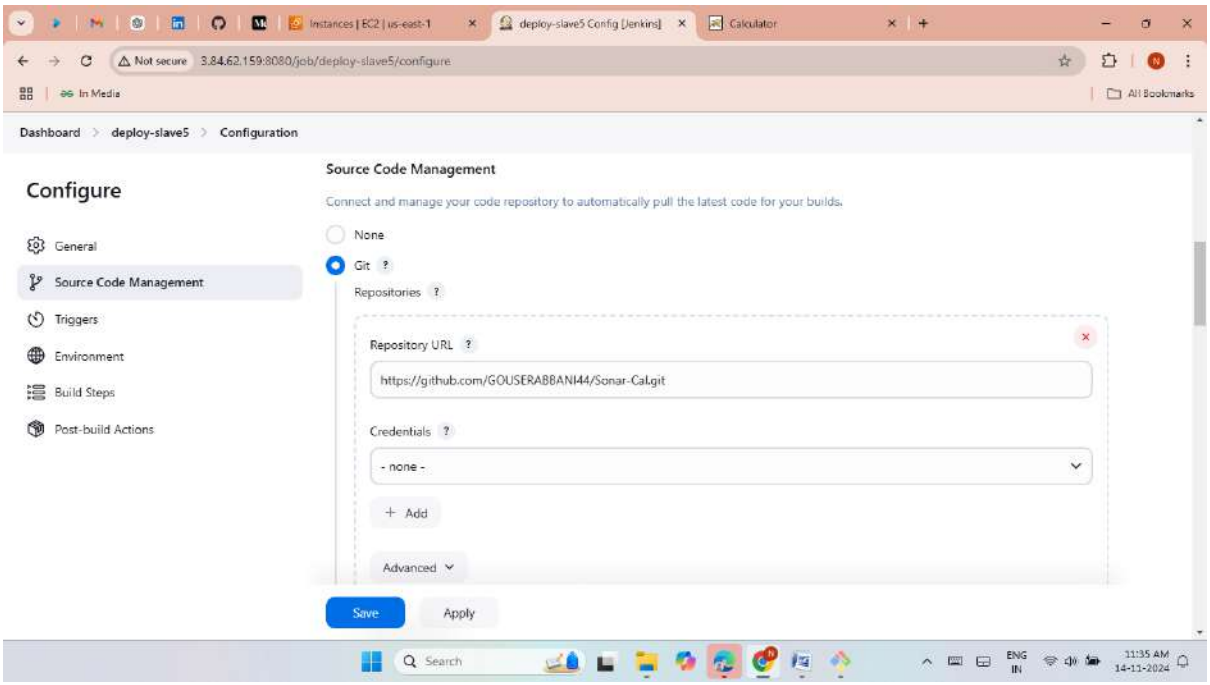
- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- Now need to create credentials because already created in before slave.
- Now click on save.
- Here the slave5 node is successfully created.



- Now create a job and give slave5 label.



- Now clone the java web calculator repo form 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

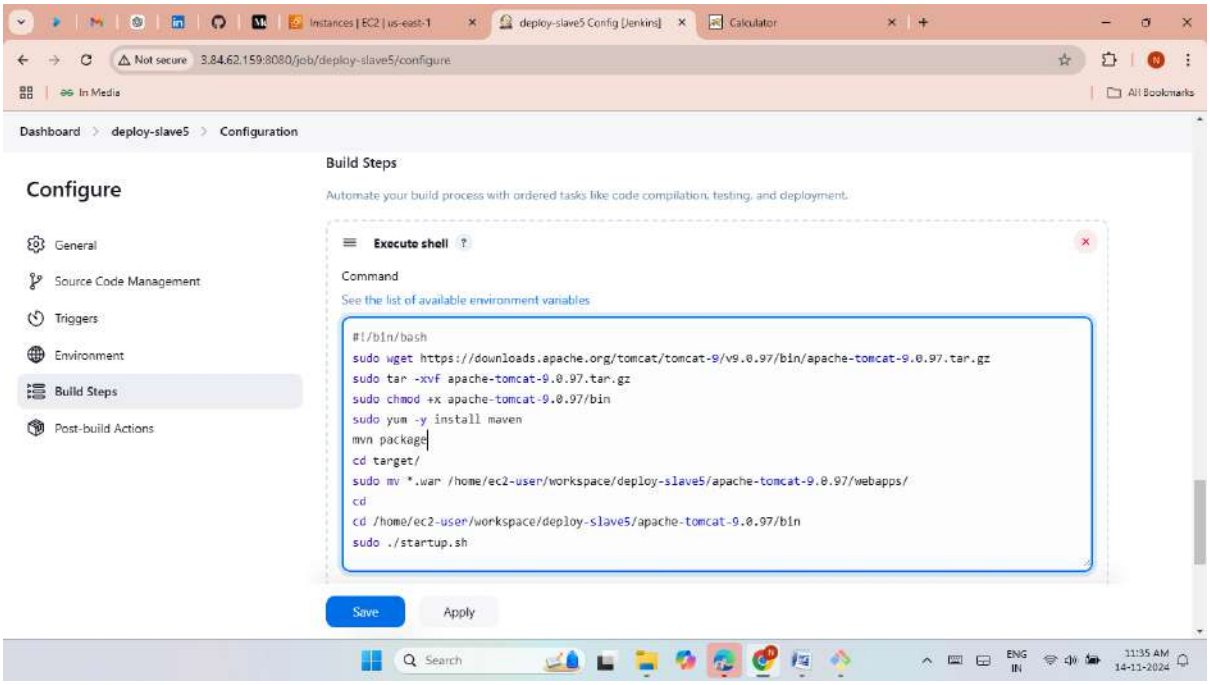
cd target/

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

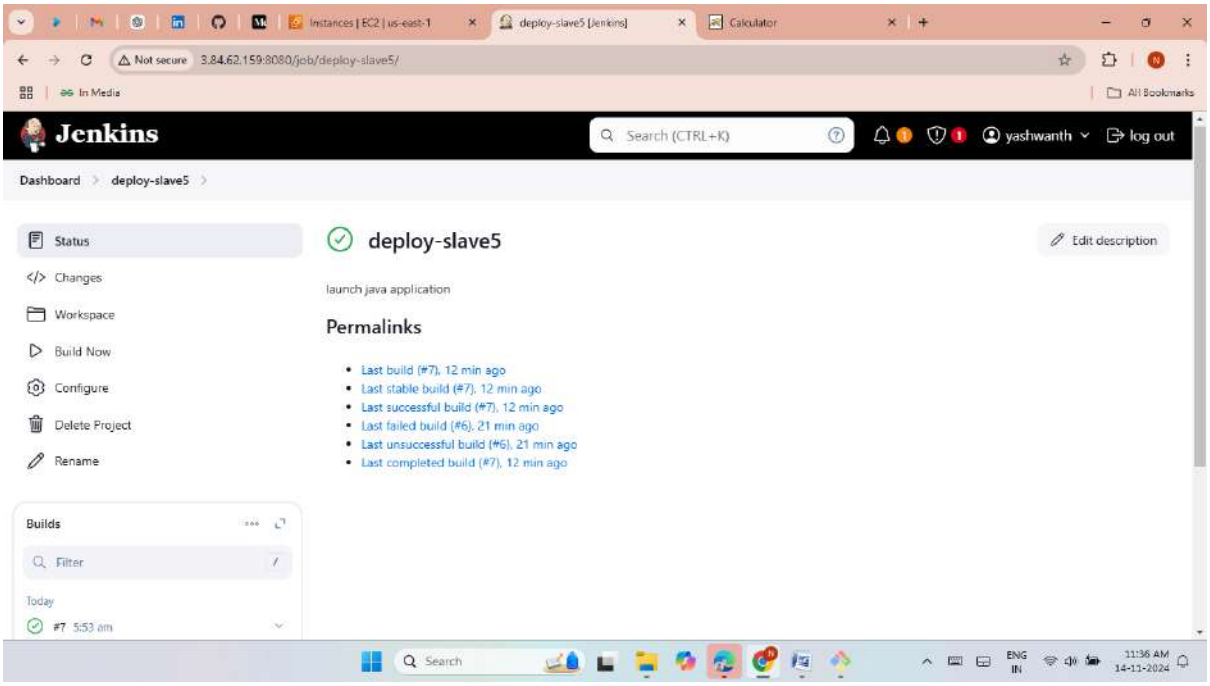
cd

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

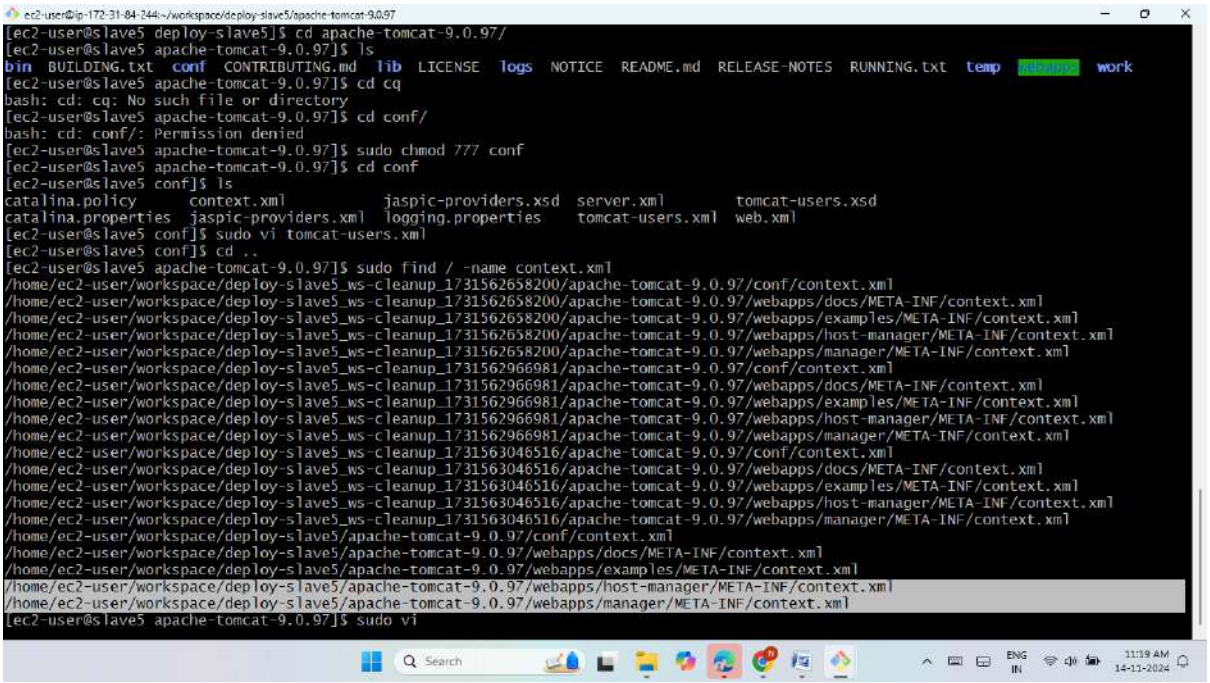
sudo ./startup.sh

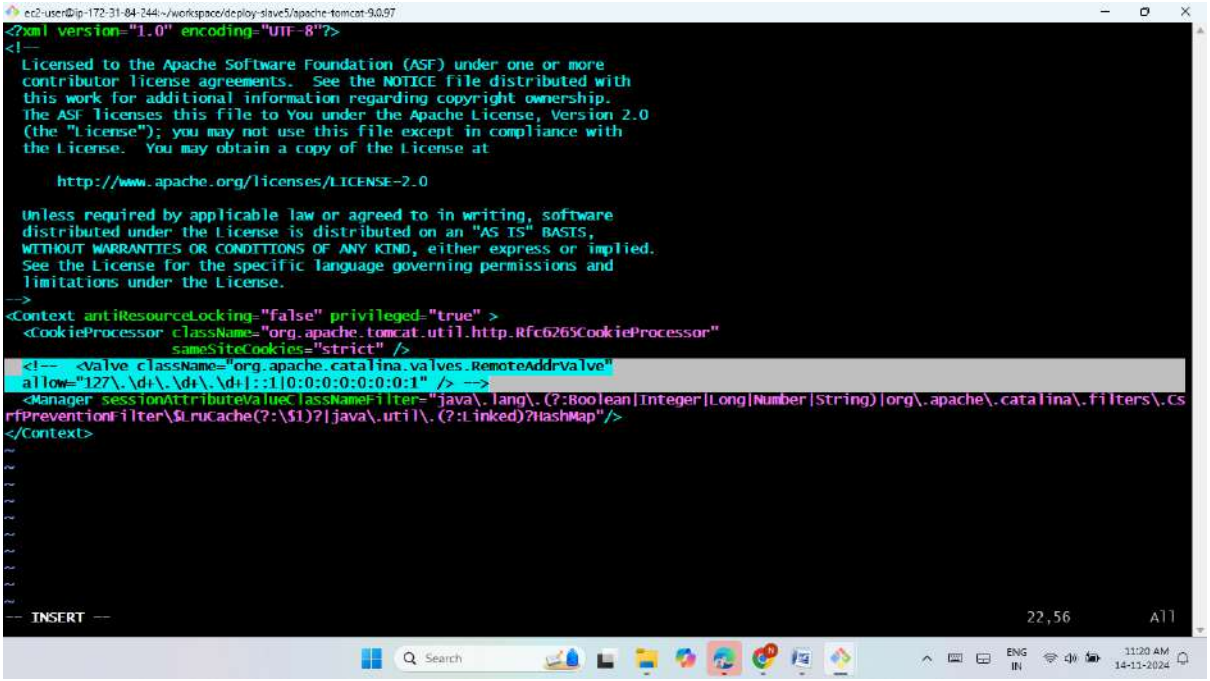


- Now click on build now.
- Here the job was success.

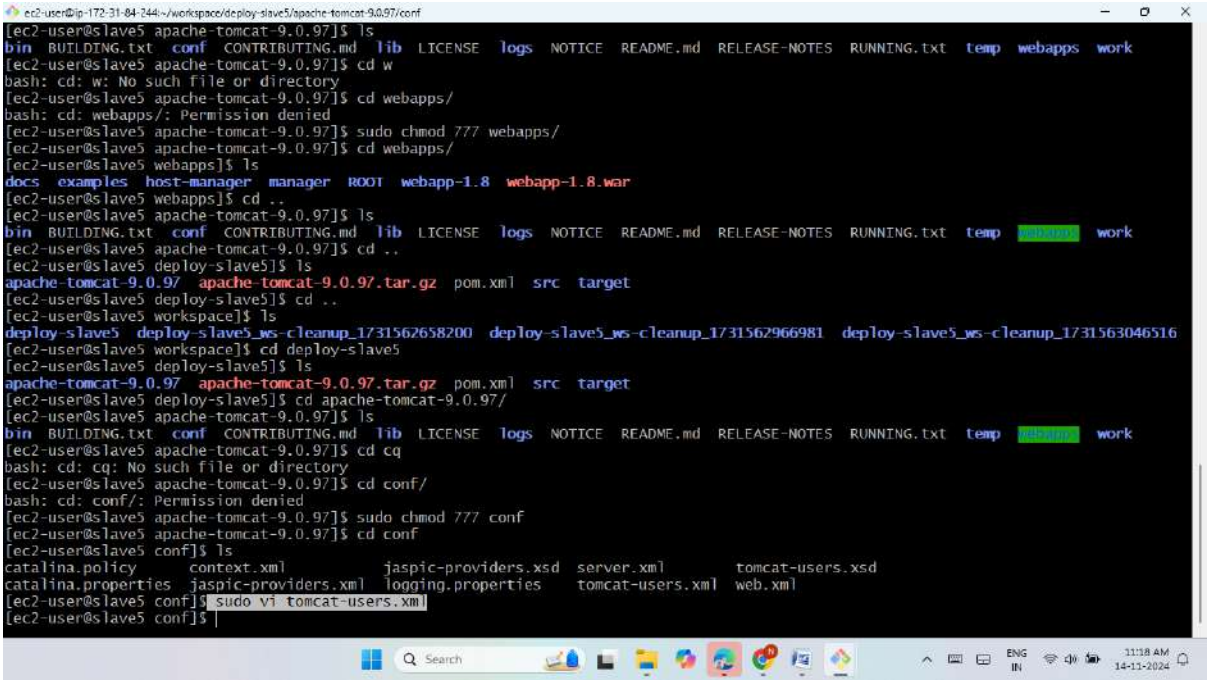


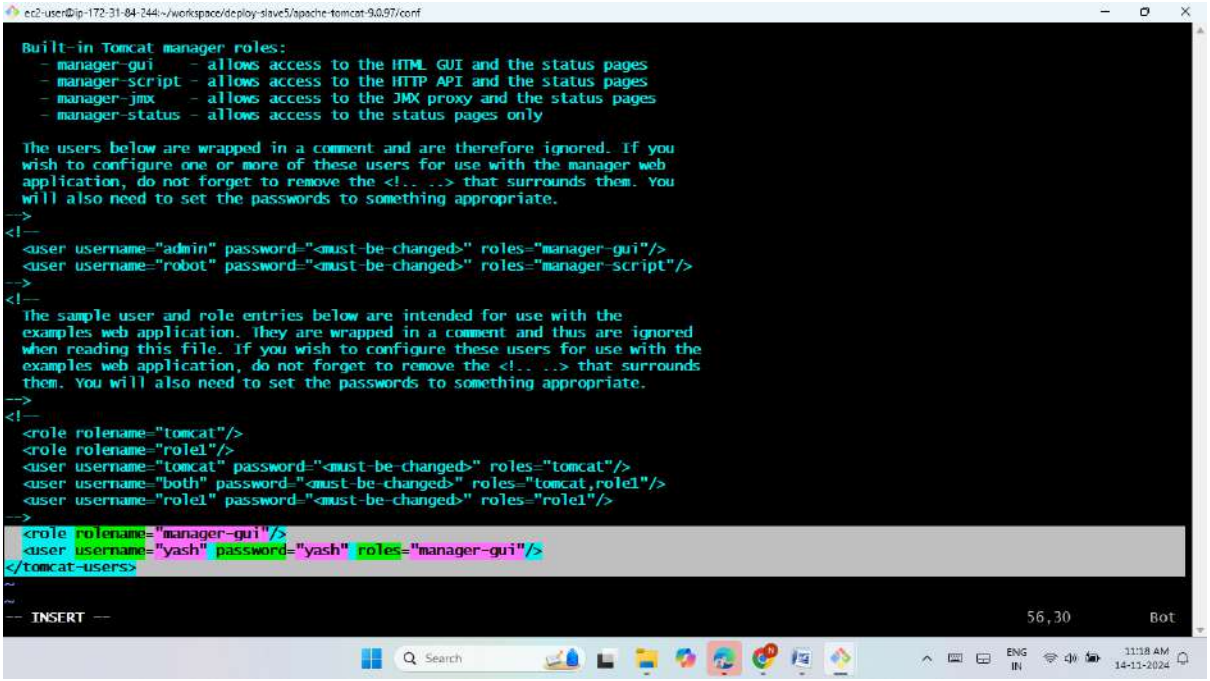
- Now remove some permissions in host-manager and manger context.xml files.



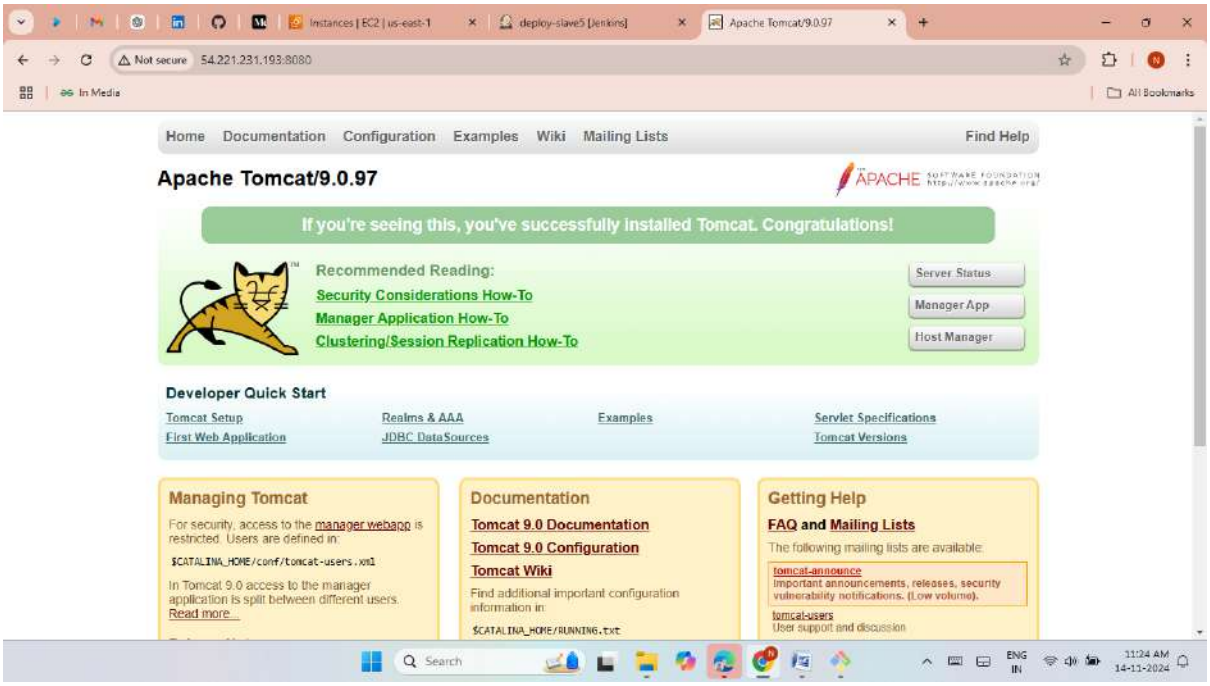


- Now create manager-gui role along with username and password.

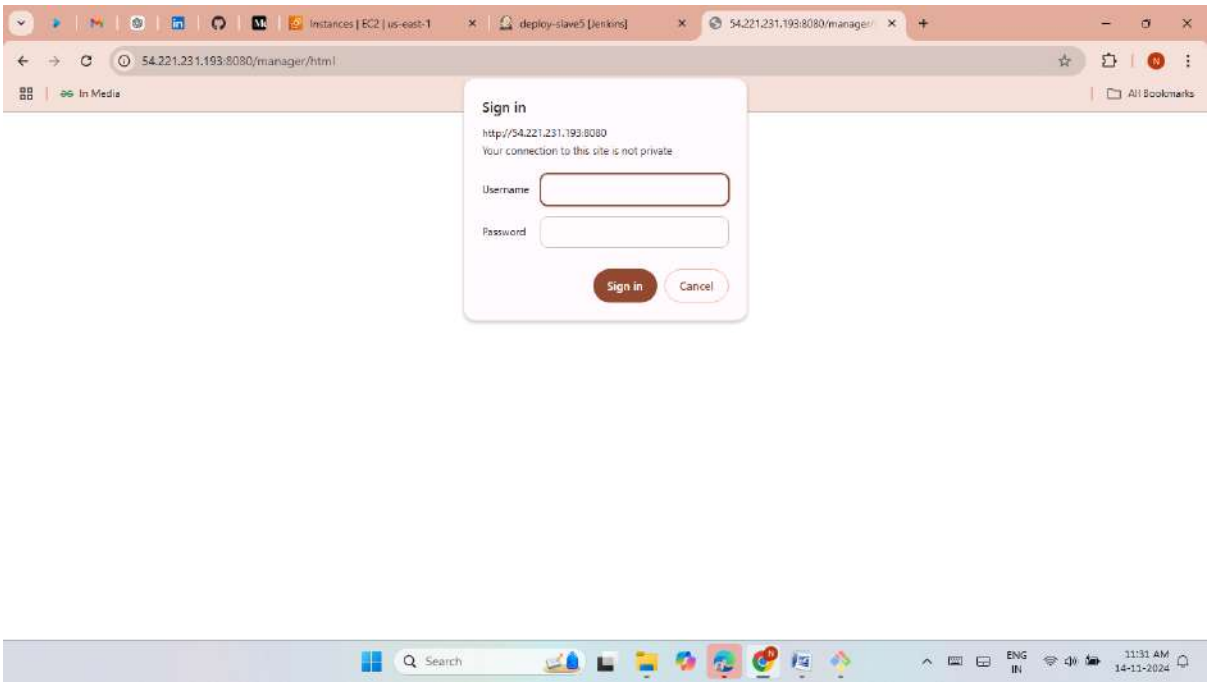




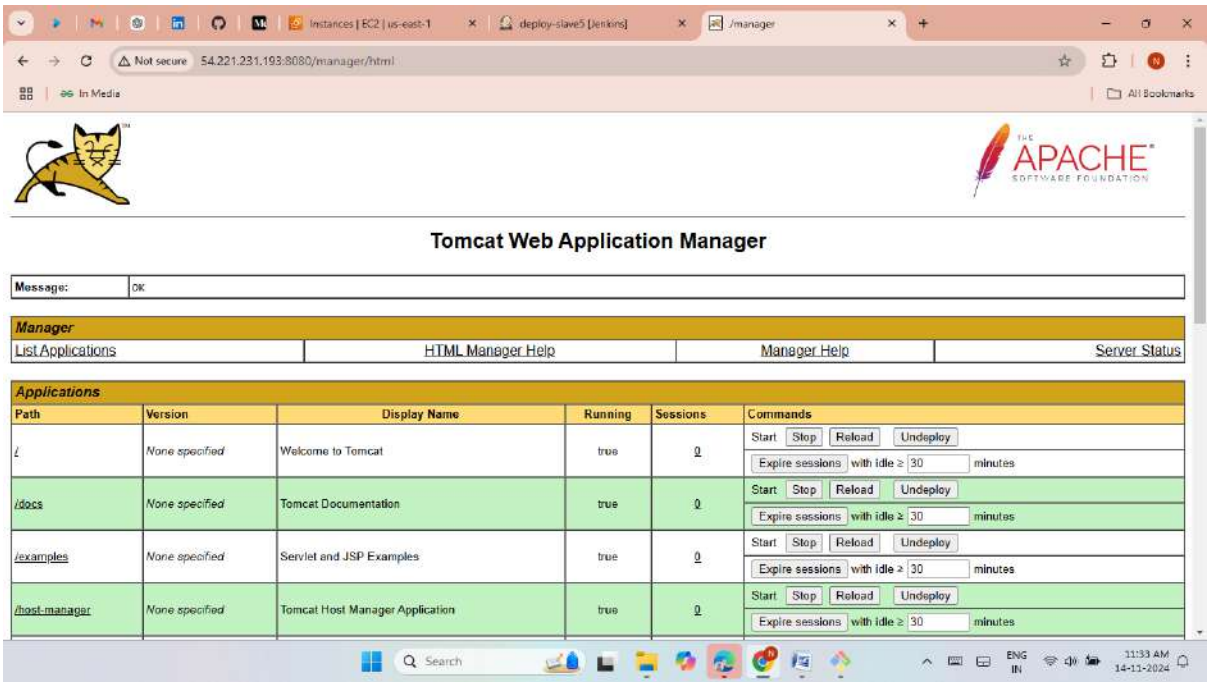
- Now copy the slave5 public ip and past it on google browser.
- The tomcat web server was hosted successfully.
- Click on manager app.



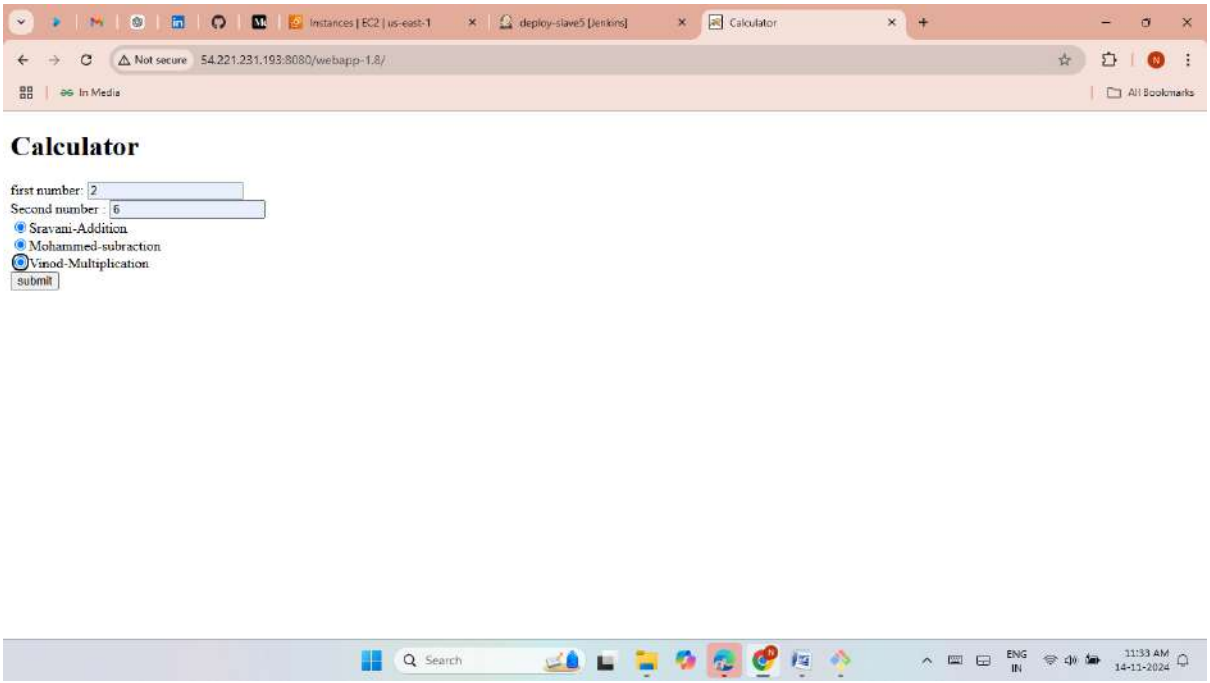
- Give manager-gui username and password.

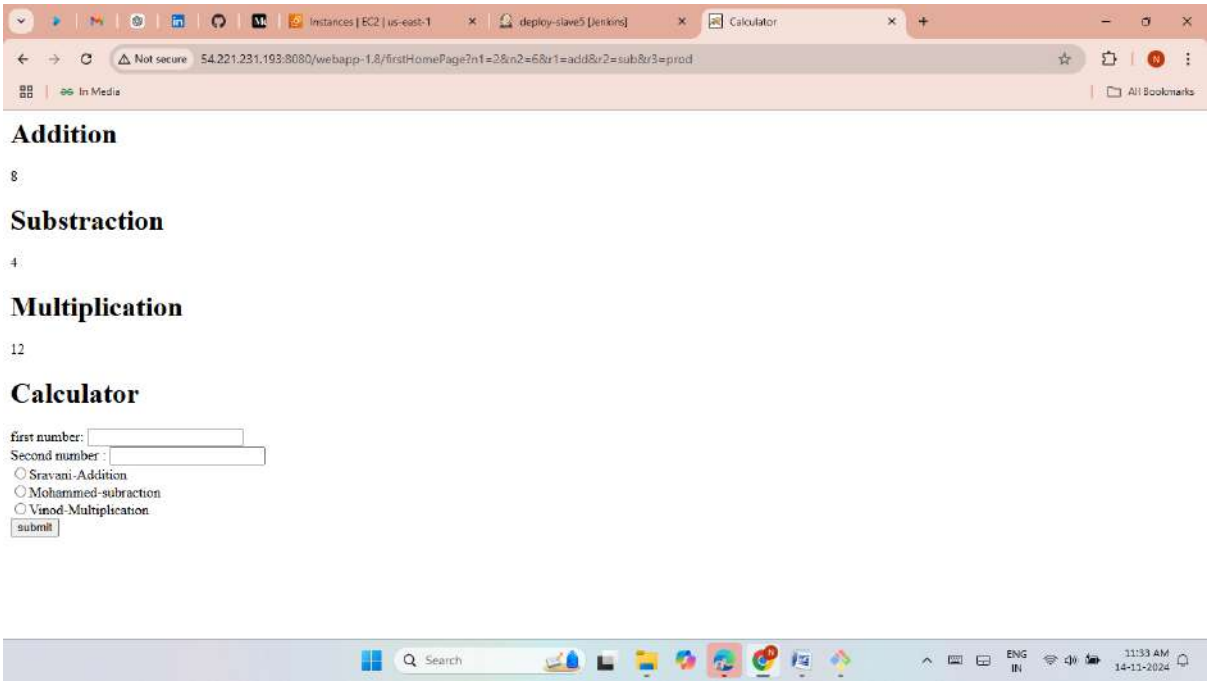


- Now click on webapps-1.8.



- Here the java web calculator was hosted successfully.

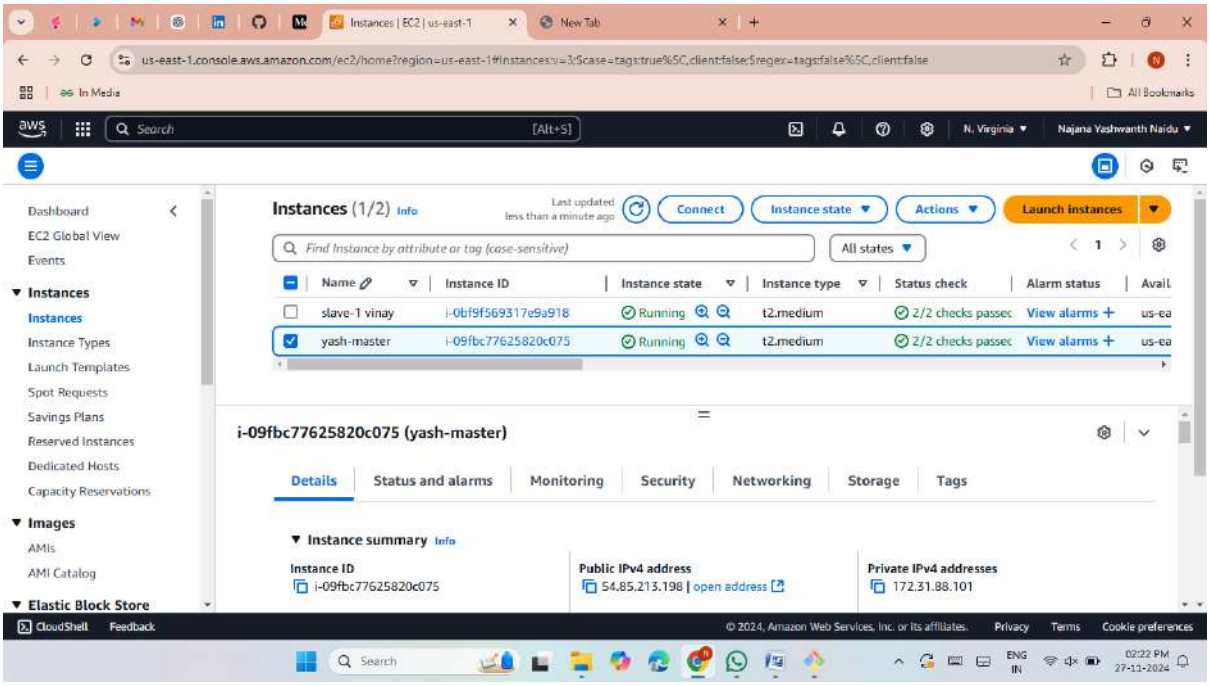

Instances | EC2 | us-east-1
deploy-slave5 [Jenkins]
Calculator
Not secure
54.221.231.193:8080/webapp-1.8/
In Media
All Bookmarks
Calculator
first number: 2
Second number : 6
Sravani-Addition
Mohammed-subraction
Vinod-Multiplication
submit
Search
ENG
IN
11:33 AM
14-11-2024


Instances | EC2 | us-east-1
deploy-slave5 [Jenkins]
Calculator
Not secure
54.221.231.193:8080/webapp-1.8/firstHomePage?n1=2&n2=6&r1=add&r2=sub&r3=prod
In Media
All Bookmarks
Addition
8
Substraction
4
Multiplication
12
Calculator
first number:
Second number :
Sravani-Addition
Mohammed-subraction
Vinod-Multiplication
submit
Search
ENG
IN
11:33 AM
14-11-2024

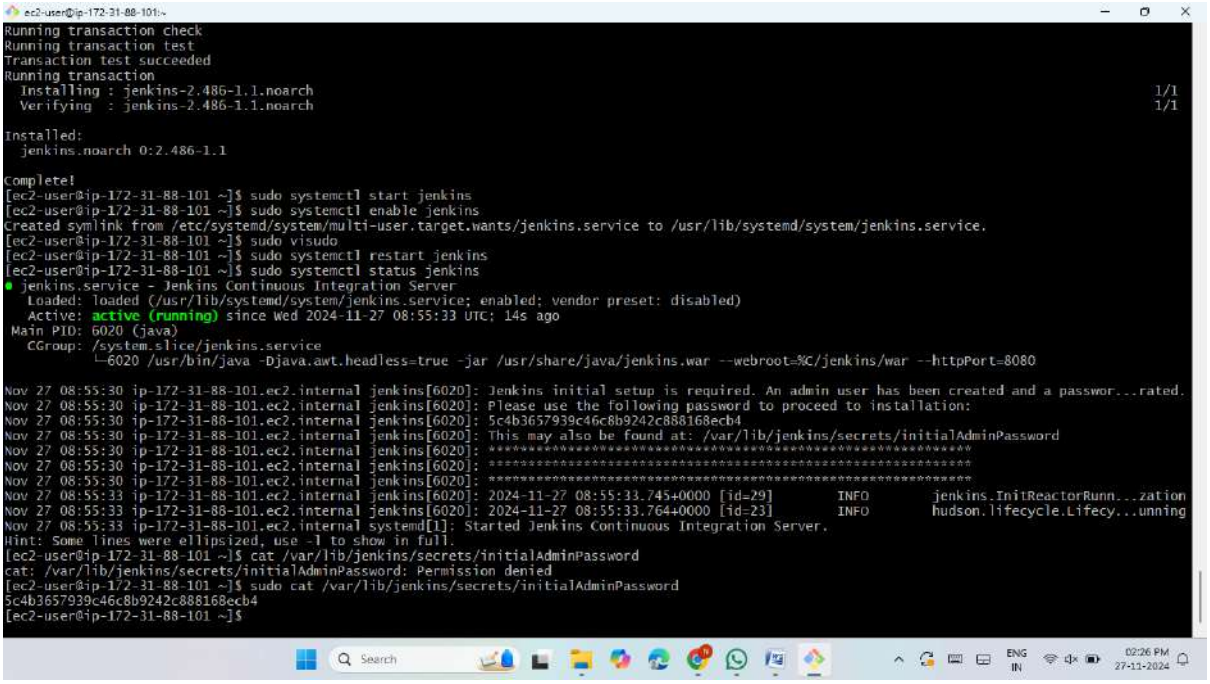
Method-3

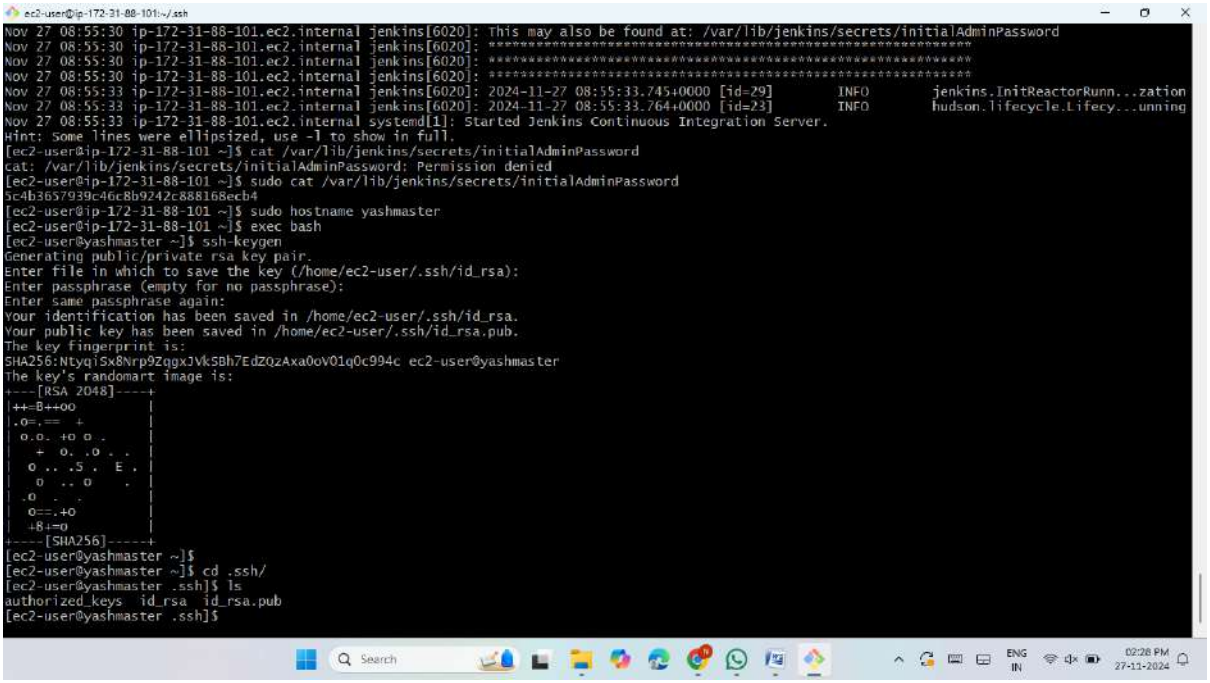
Jostna Create one master and Slave-5 and Yaswanth creates 2nd slave node and Vinay creates 3rd slave and Ramsai creates 4th slave Rajesh creates slave-1 Same AWS account and Jostna deploy word press with bash script(slave-5) and yashwanth deploy word press with Docker Compose file (slave-2), and Vinay 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.

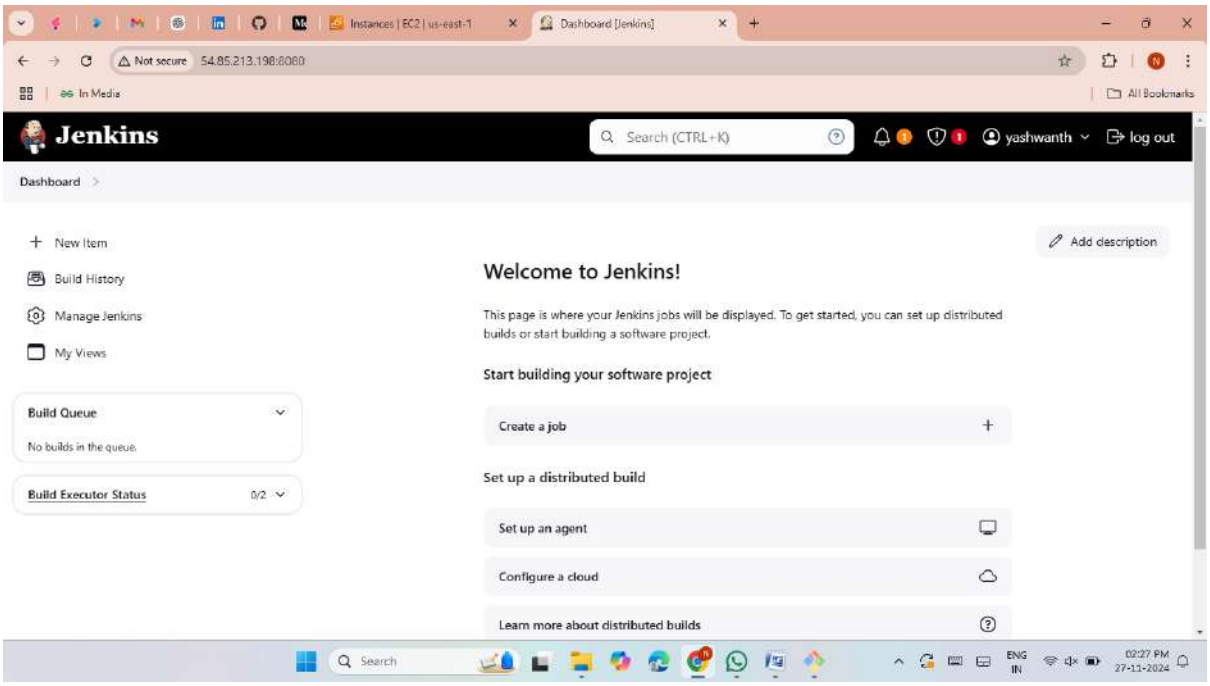


- Now connect master then install java.
- Install Jenkins then start and enable.
- Generate keys with the command ssh-keygen.

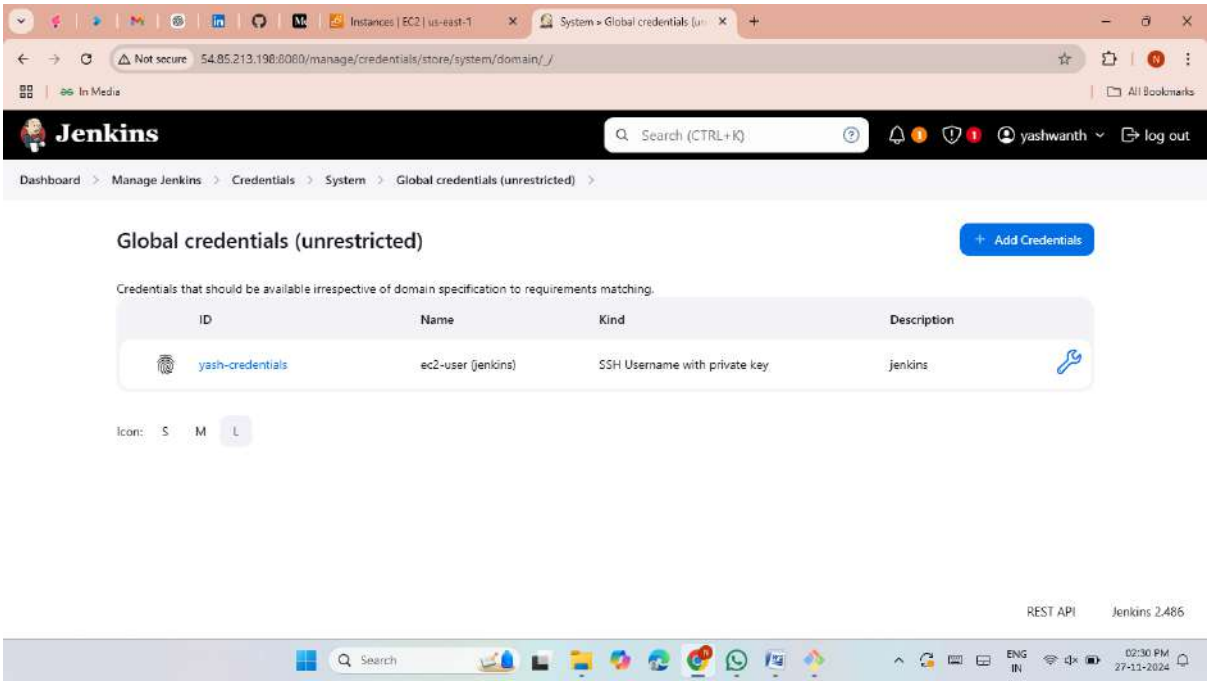




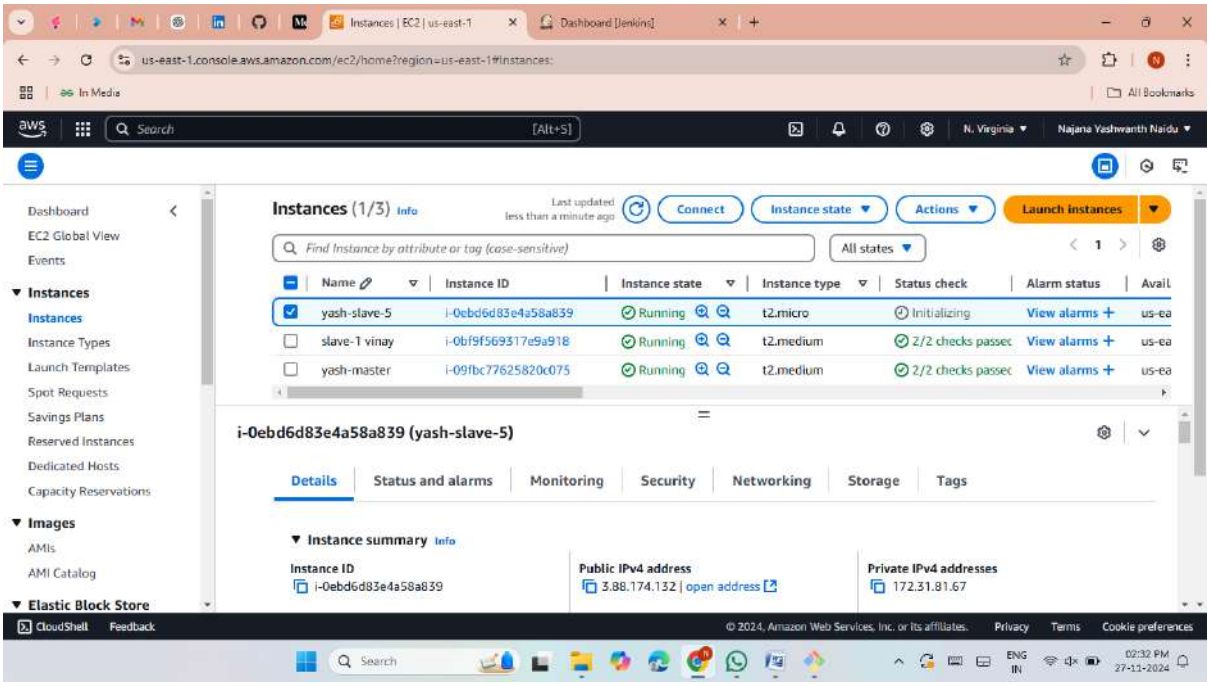
- Here the Jenkins was hosted successfully.



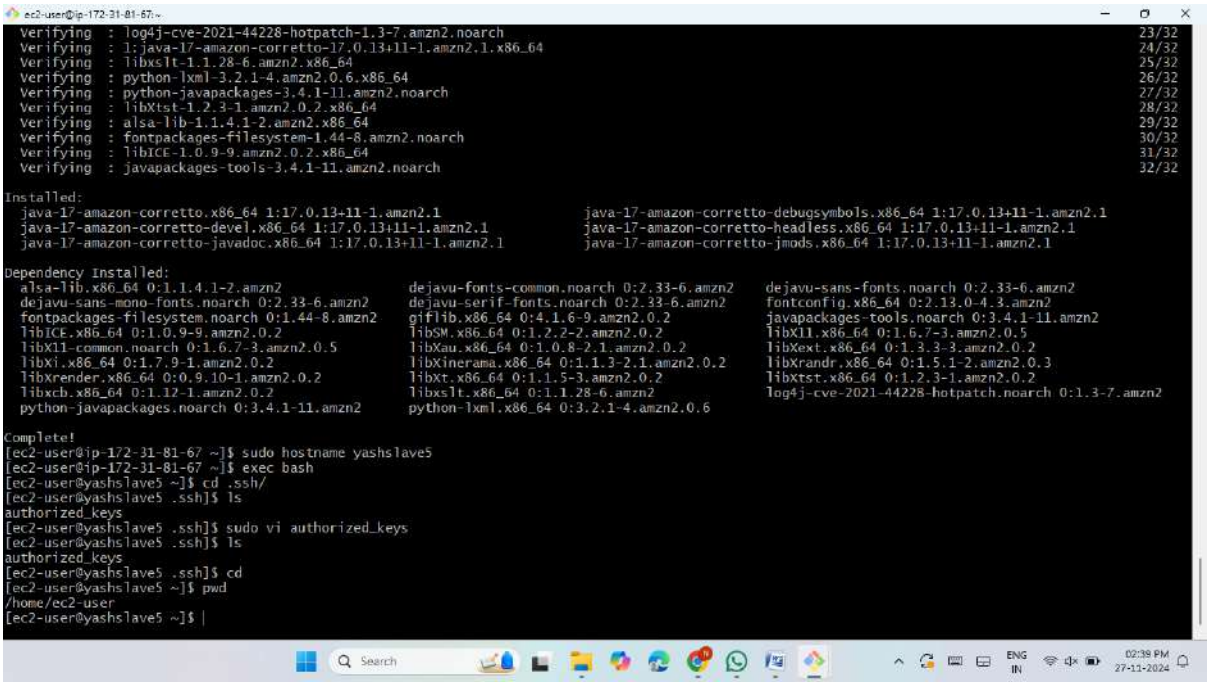
- Now create a credentials with private key form master.



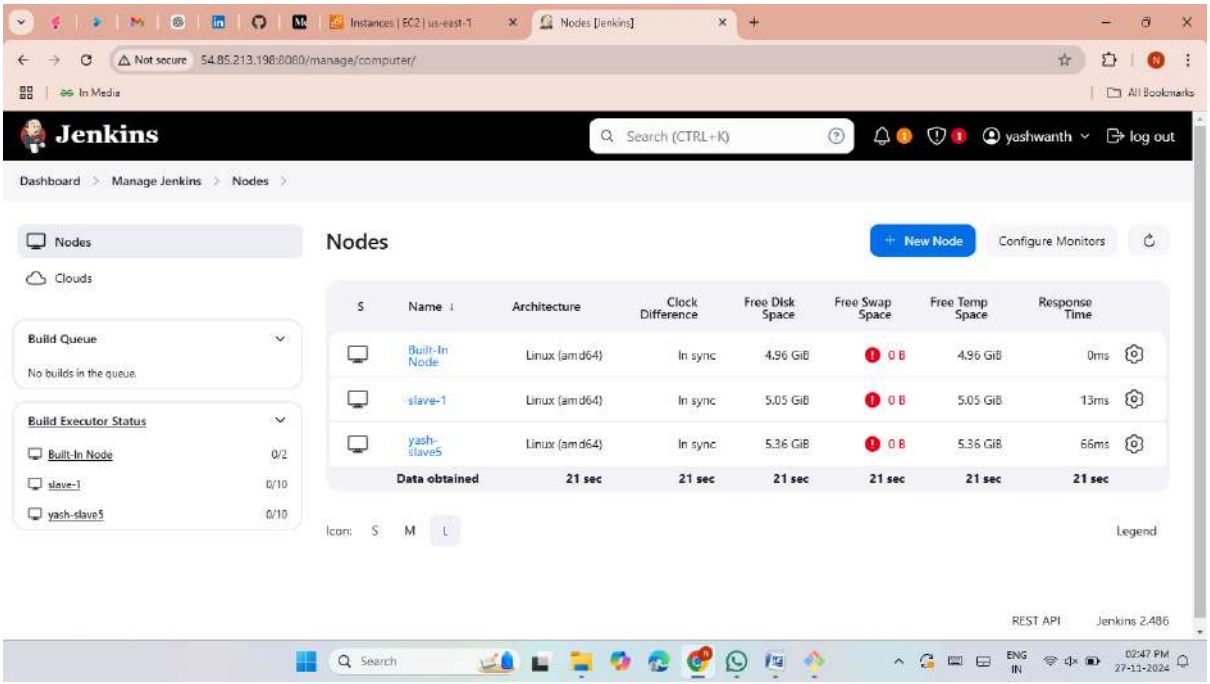
- Here I have worked as a slave-5 in a team project.
- Launch an ec2 instance along with required ports.



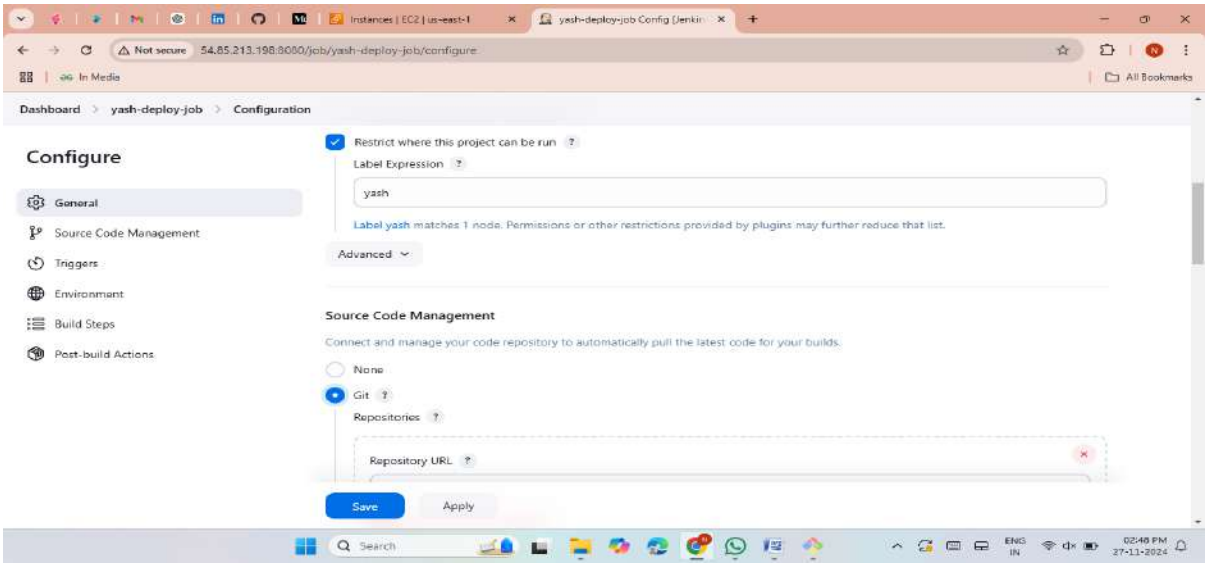
- Now connect Slave5 then install java.
- Now give private keys from master server.

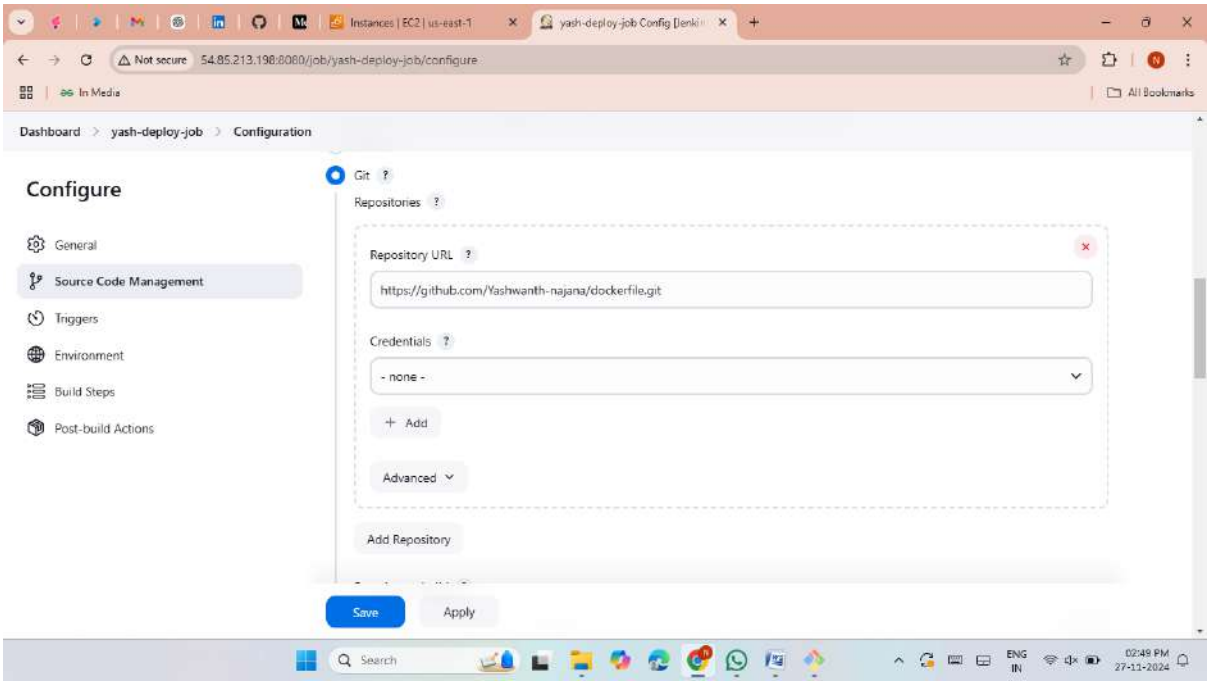


- Now create a node for slave5 server.
- Give details for slave node.
- Here we have to give slave5 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here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



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





- 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

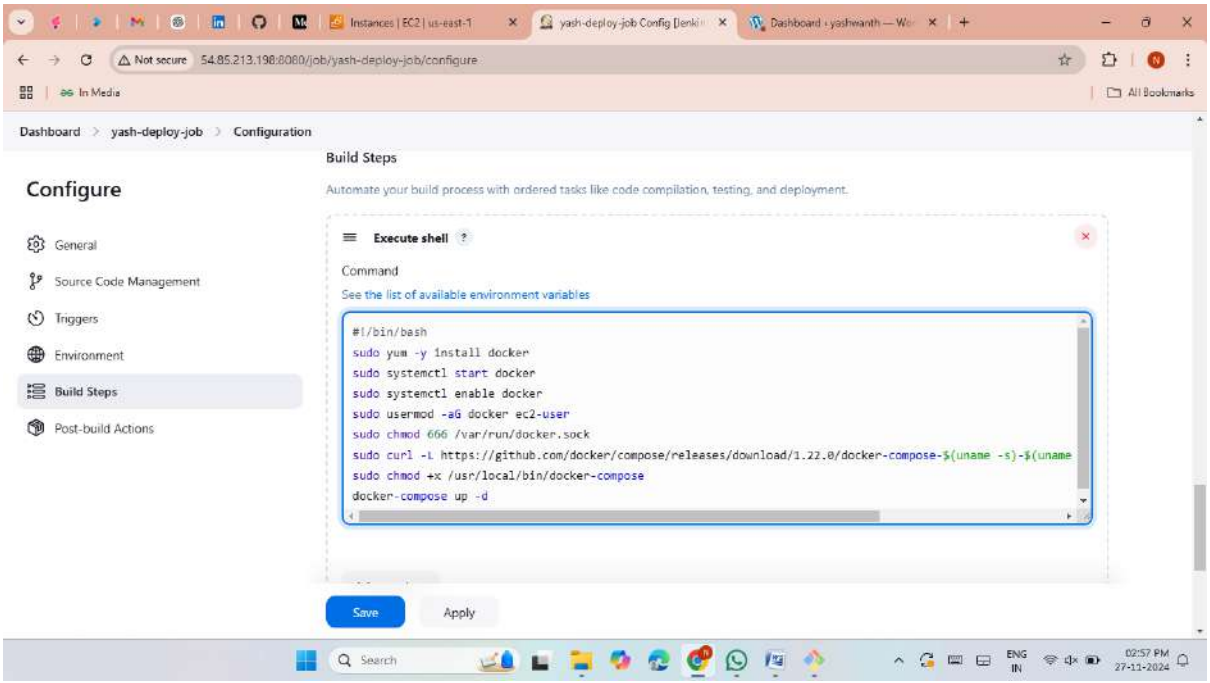
sudo usermod -aG docker ec2-user

sudo chmod 666 /var/run/docker.sock

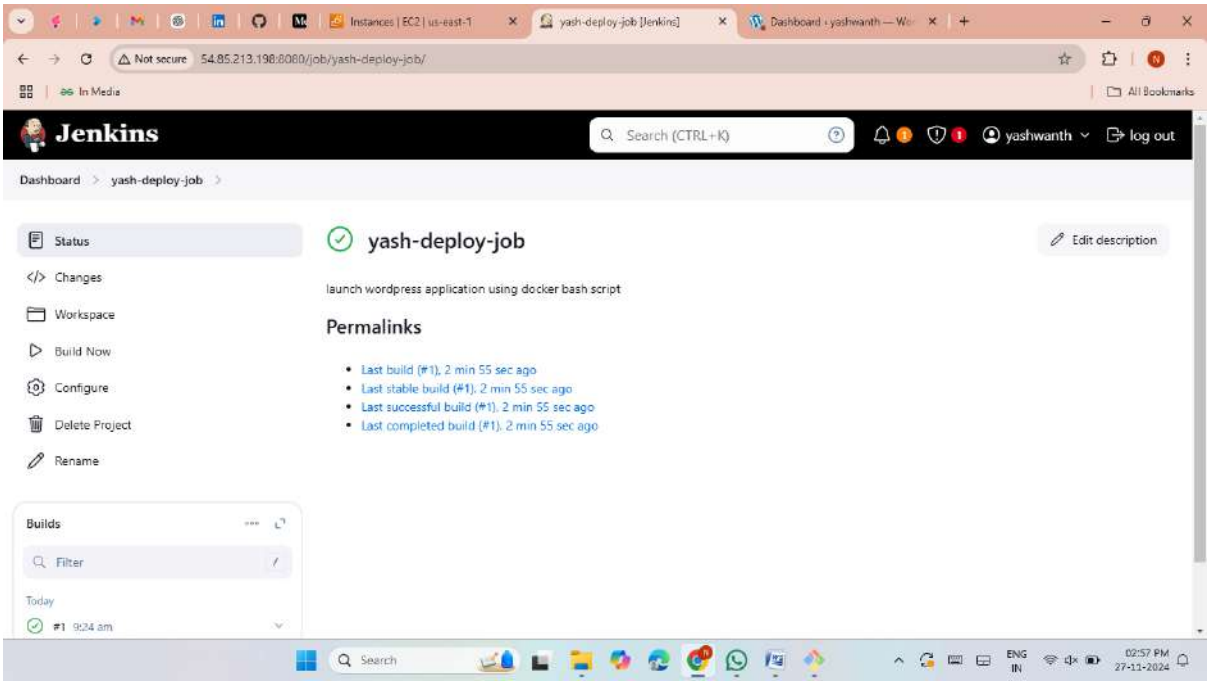
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

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

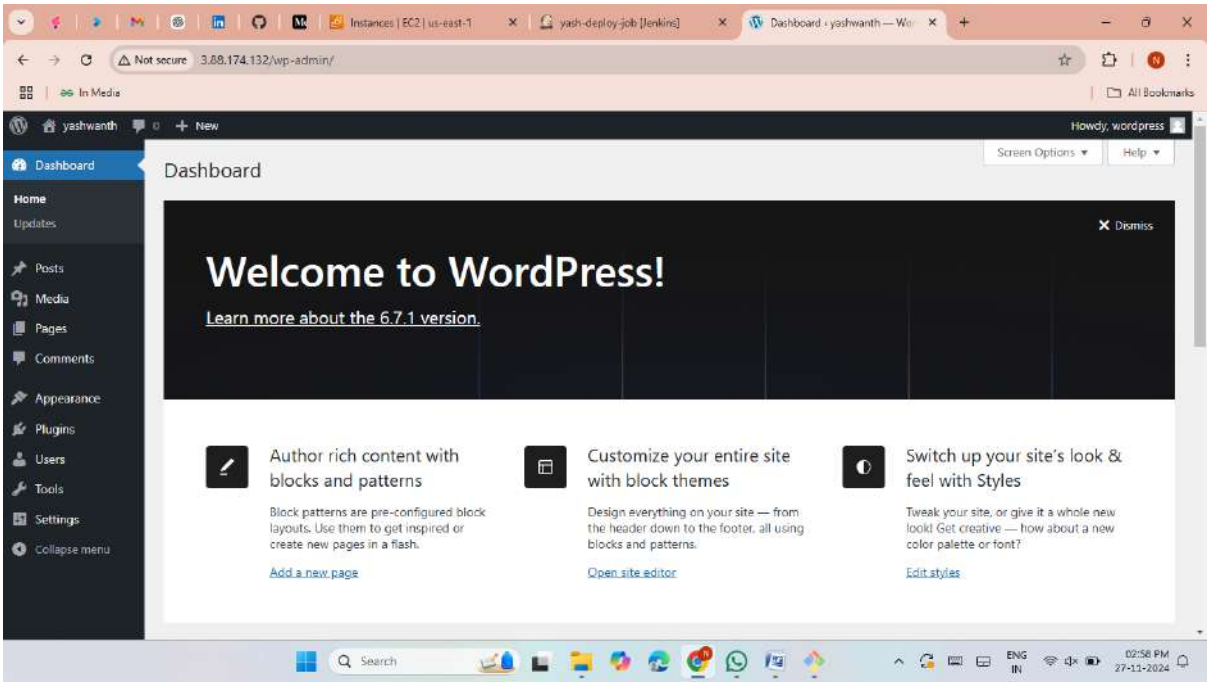
docker-compose up –d



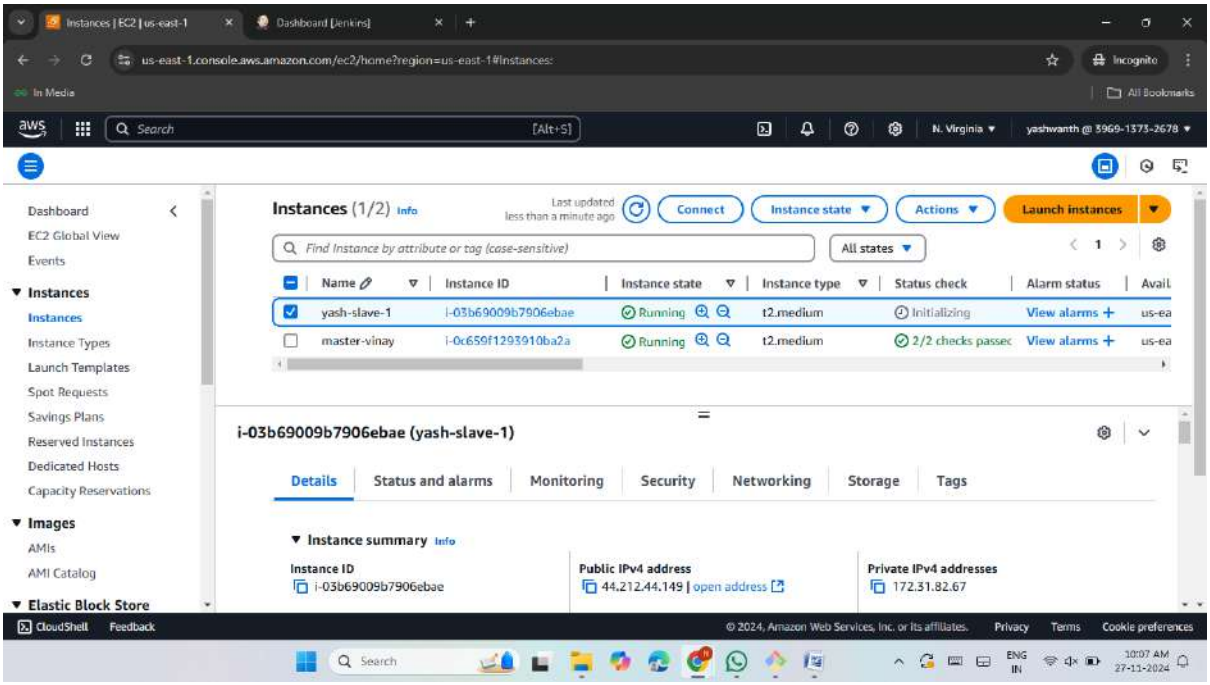
- Now click on build now.
- Here the job was success.



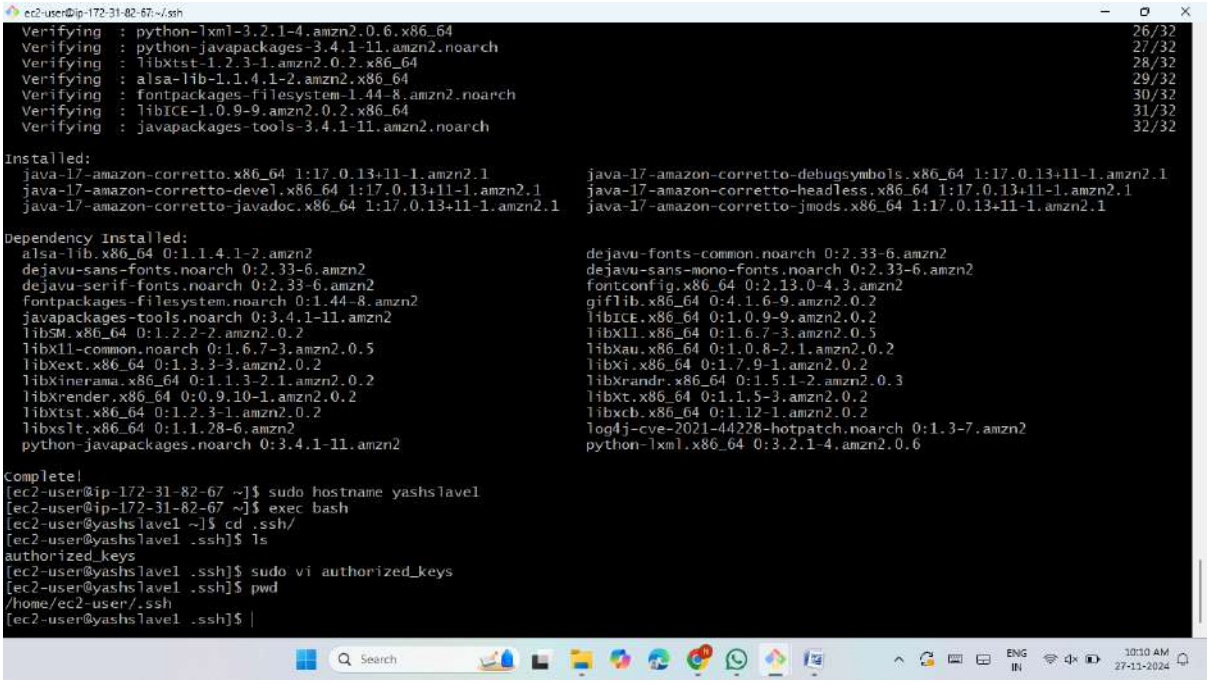
- Now copy the slave5 public ip and past it on google browser.
- The wordpress was hosted successfully.



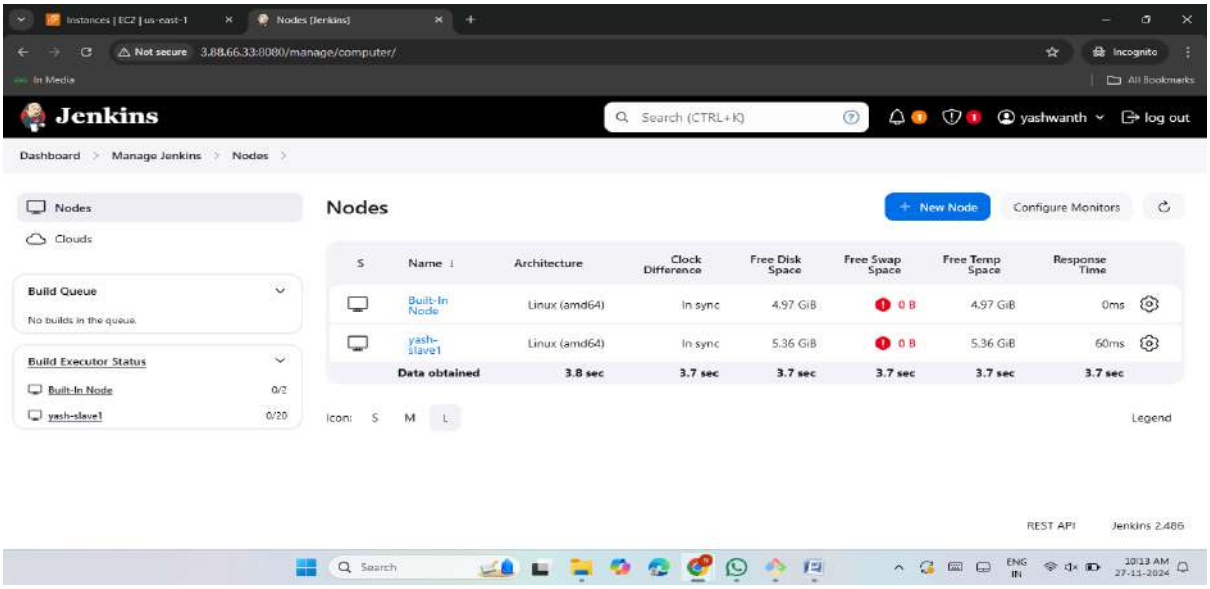
- Here I have worked as a slave-1 in a team project.
- Launch an ec2 instance along with required ports.



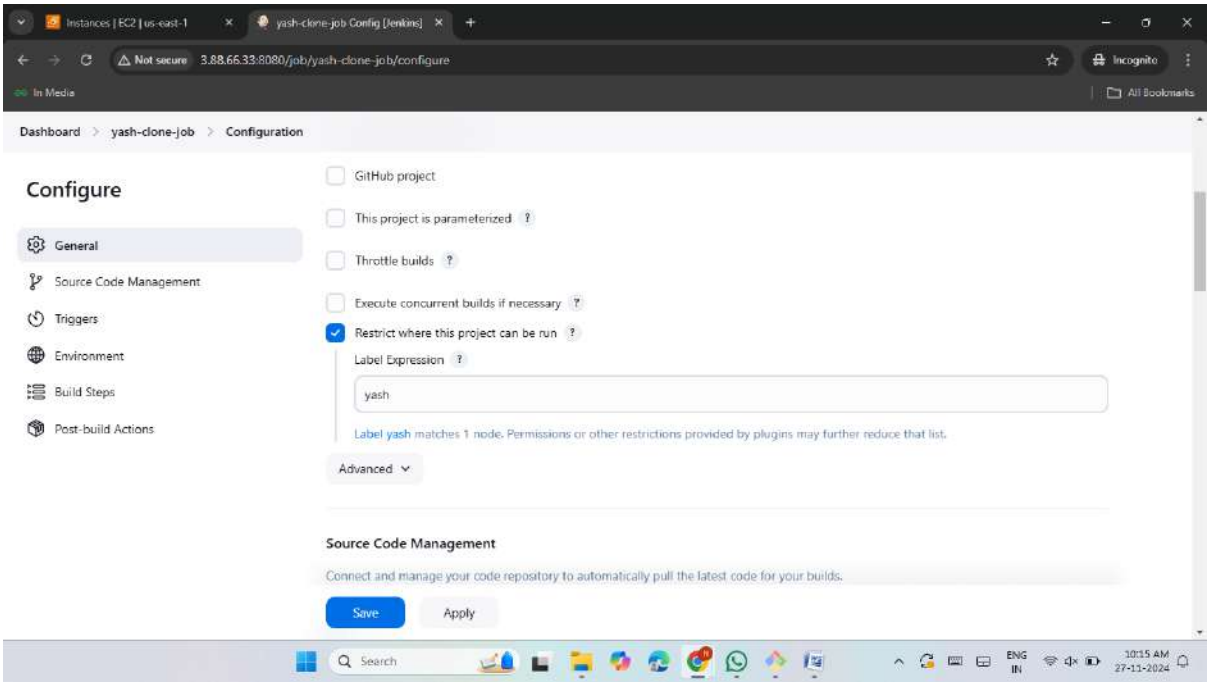
- Now connect Slave1 then install java.
- Now give private keys from master server.

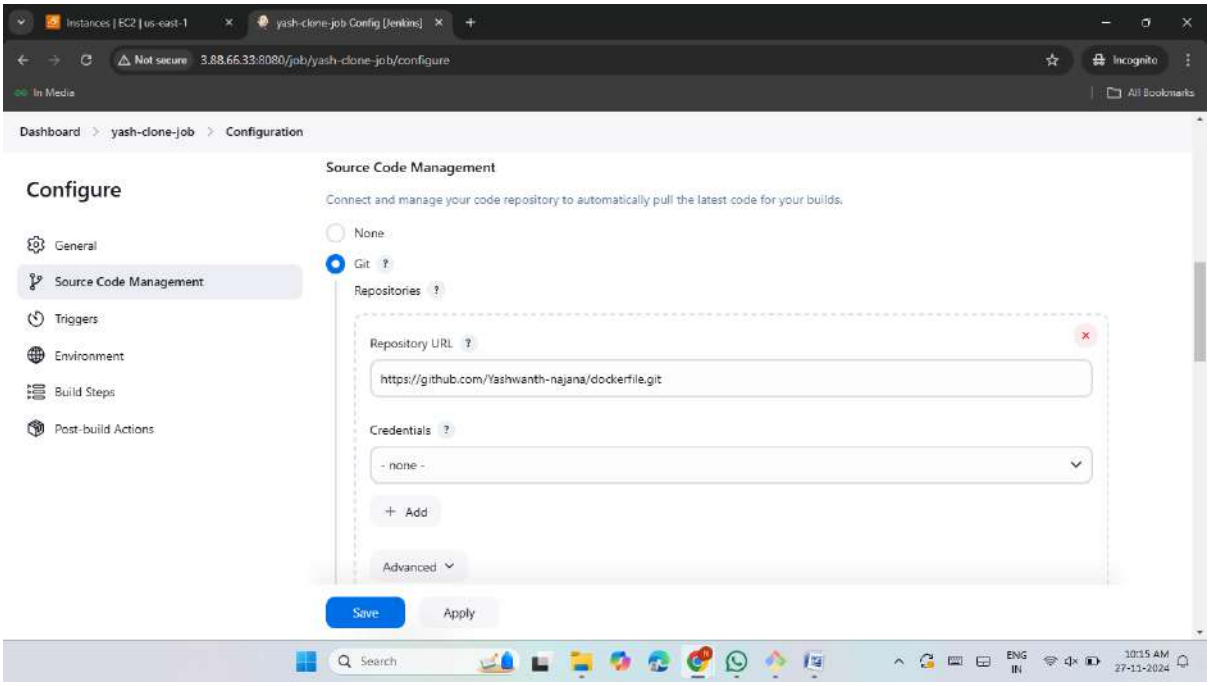


- Now create a node for slave1 server.
- Give details for slave node.
- Here we have to give slave1 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here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

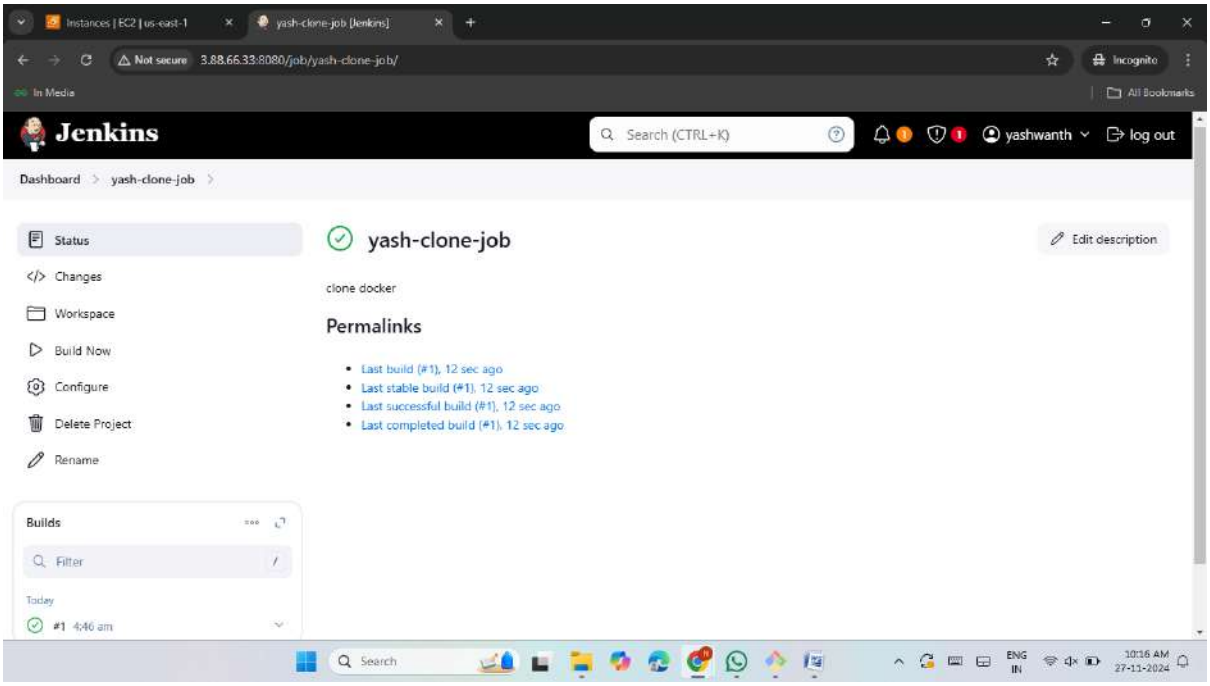


- Now create a job and give slave1 label.
- Now clone the wordpress application repo form github.
- Now Click on save.

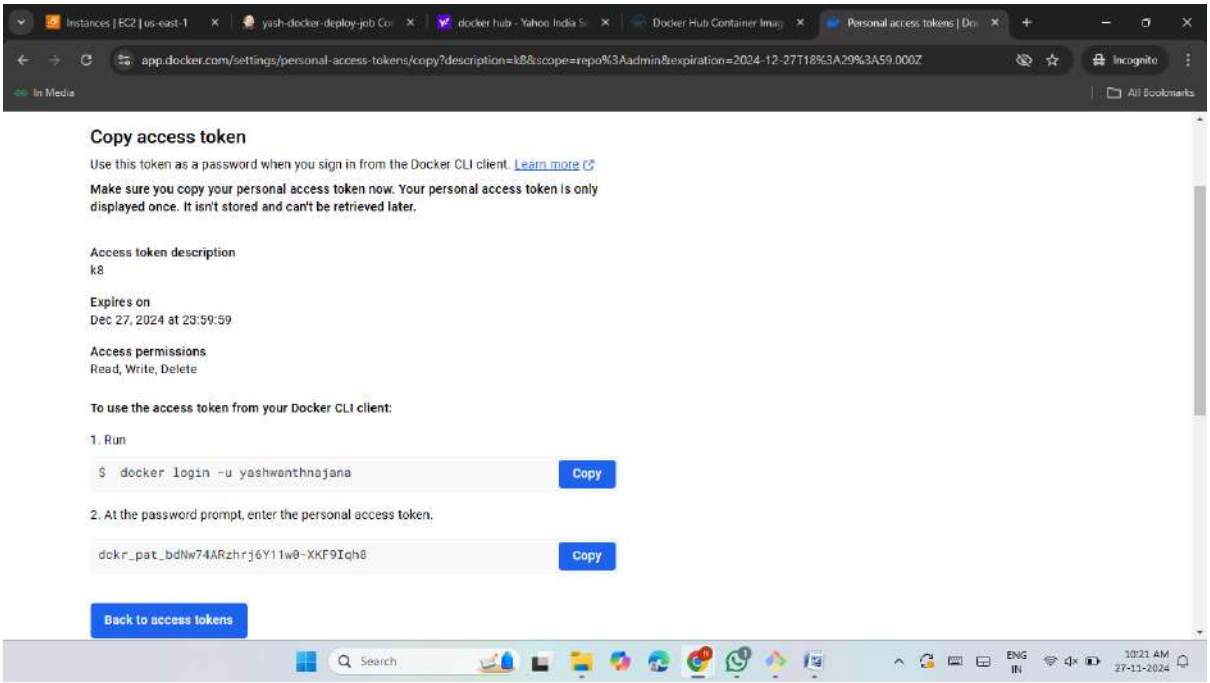




- Now click on build now.
- Here the job was success.



- 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

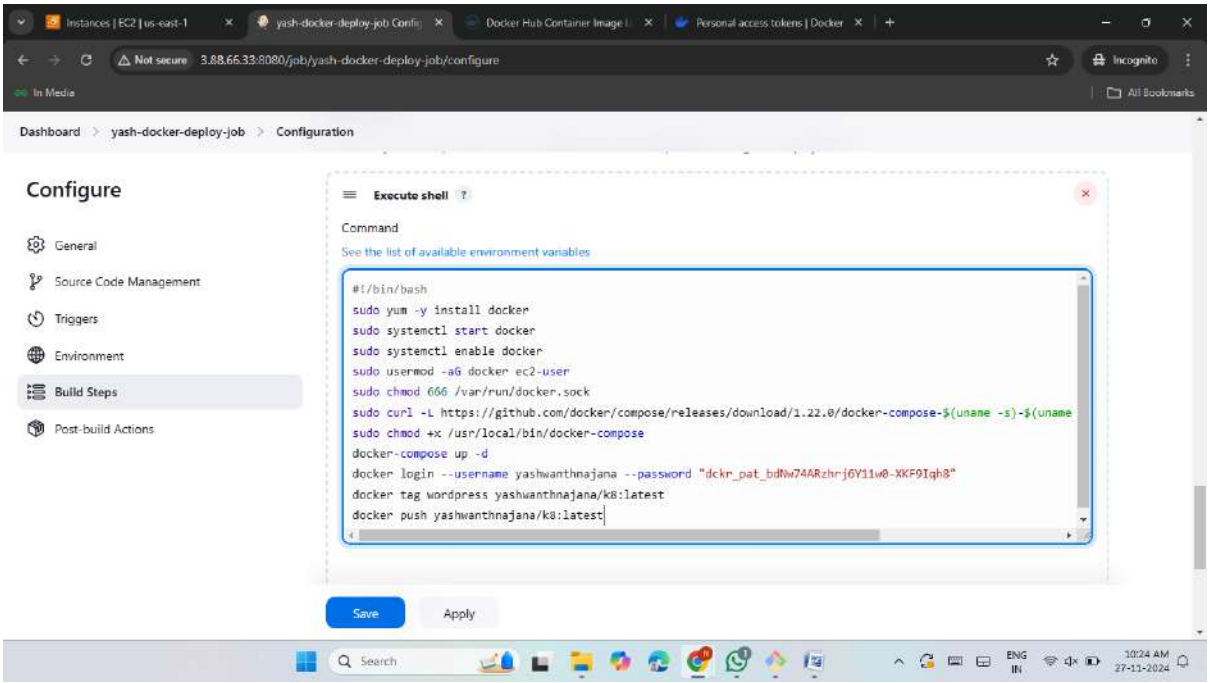
sudo chmod +x /usr/local/bin/docker-compose

docker-compose up -d

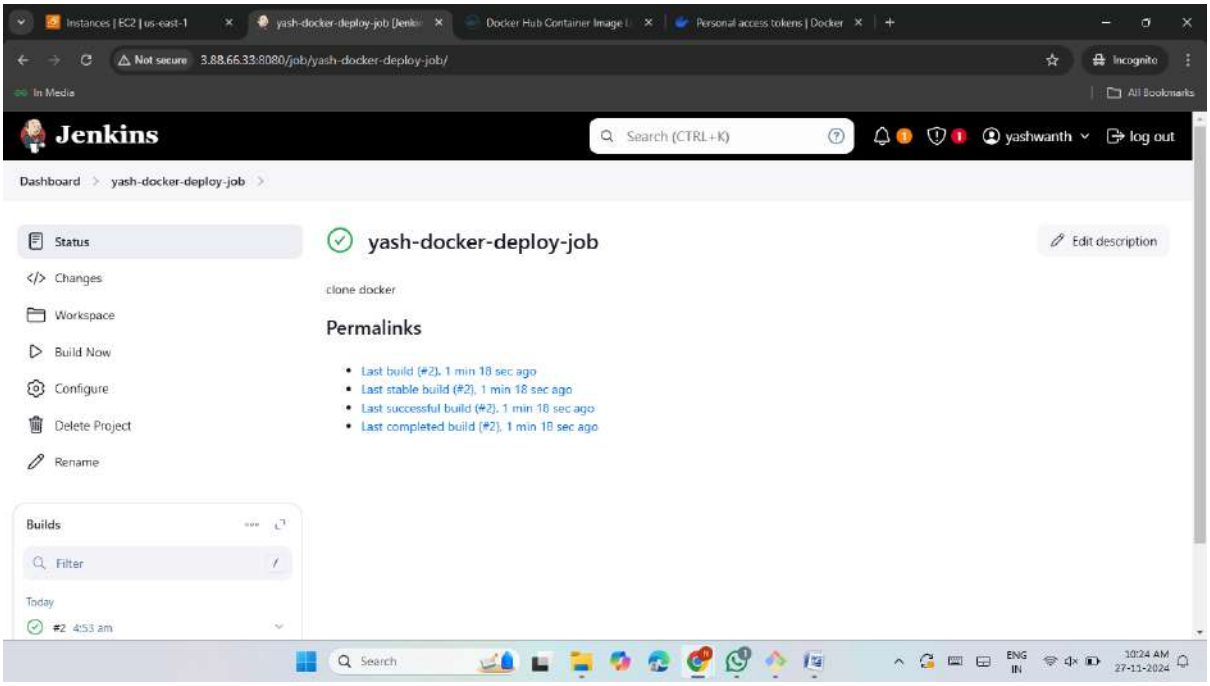
docker login --username yashwanthnajana --password "dckr_pat_bdNw74ARzhrj6Y11w0-XKF9Iqh8"

docker tag wordpress yashwanthnajana/k8:latest

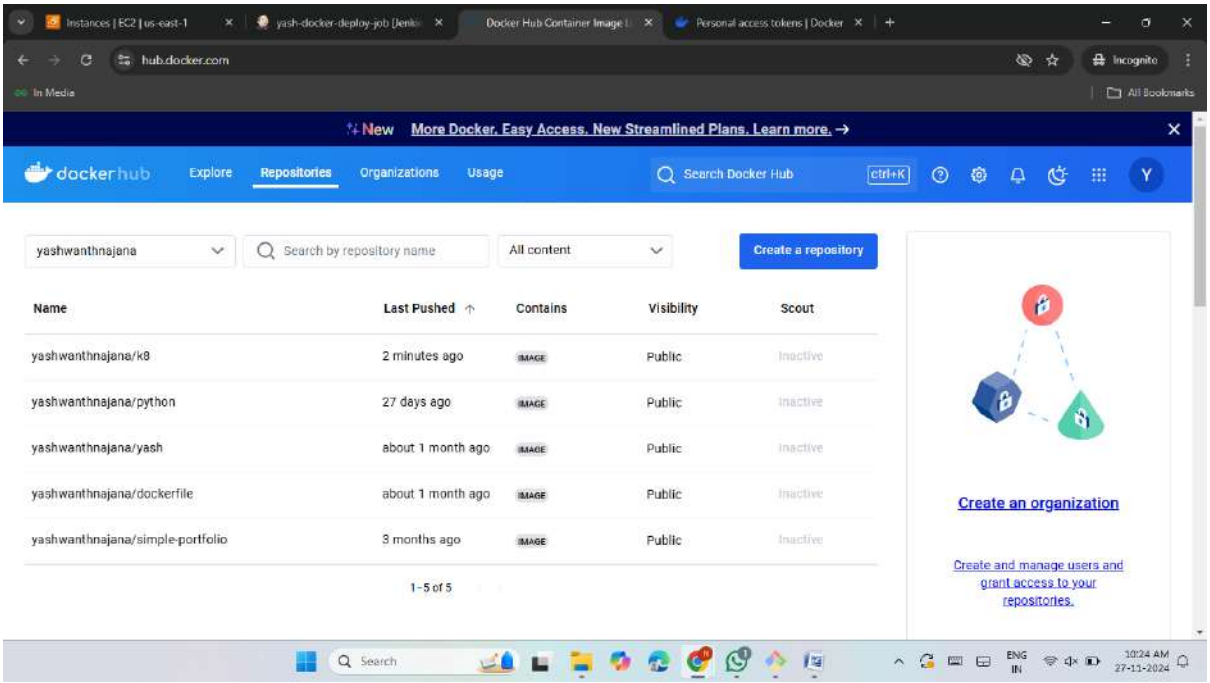
docker push yashwanthnajana/k8:latest



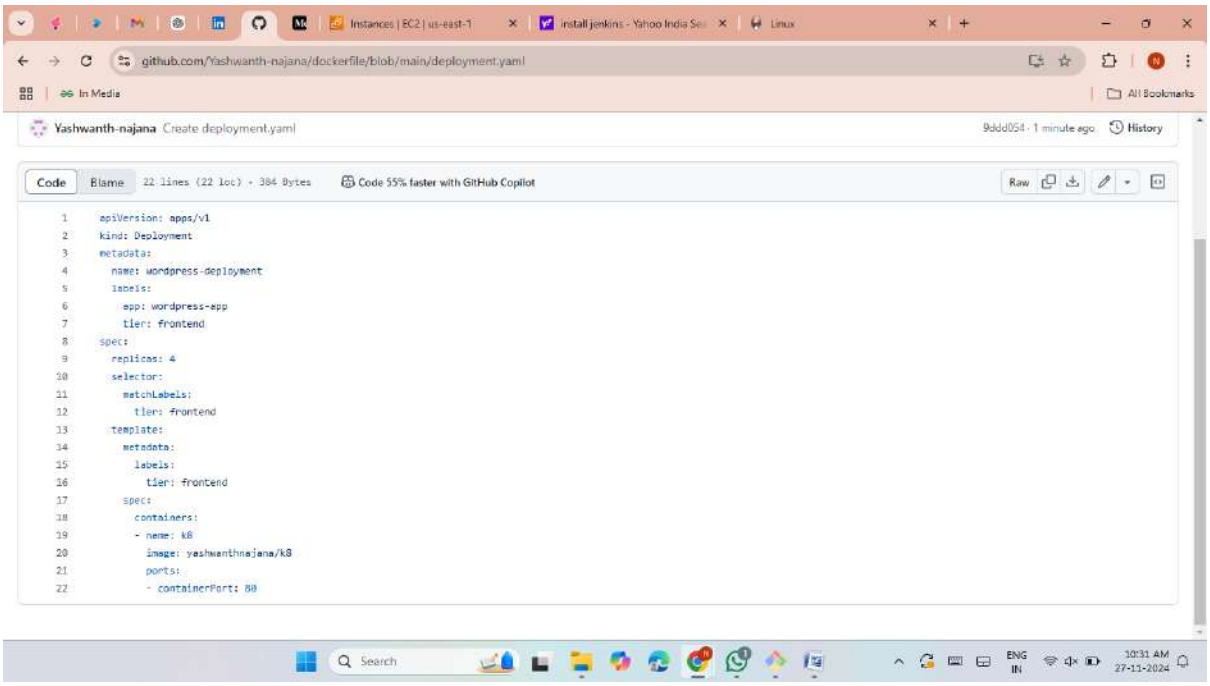
- Now click on build now.
- Here the job was success.



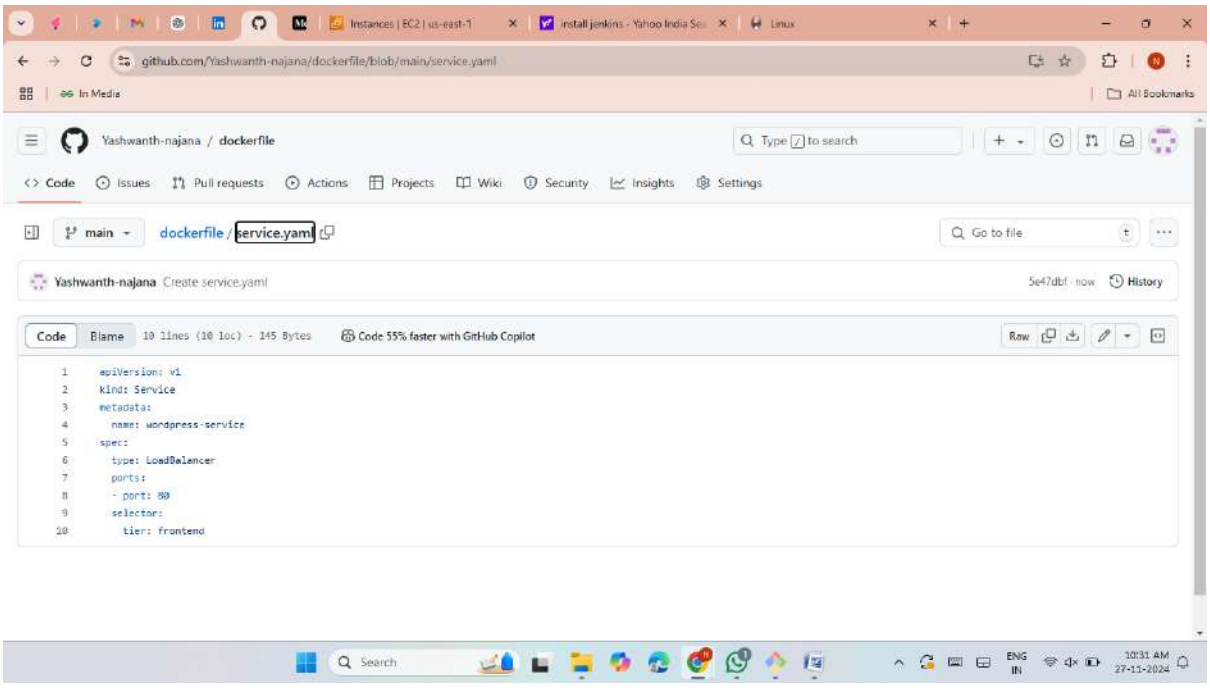
- Here the images is successfully pushed into dockerhub.



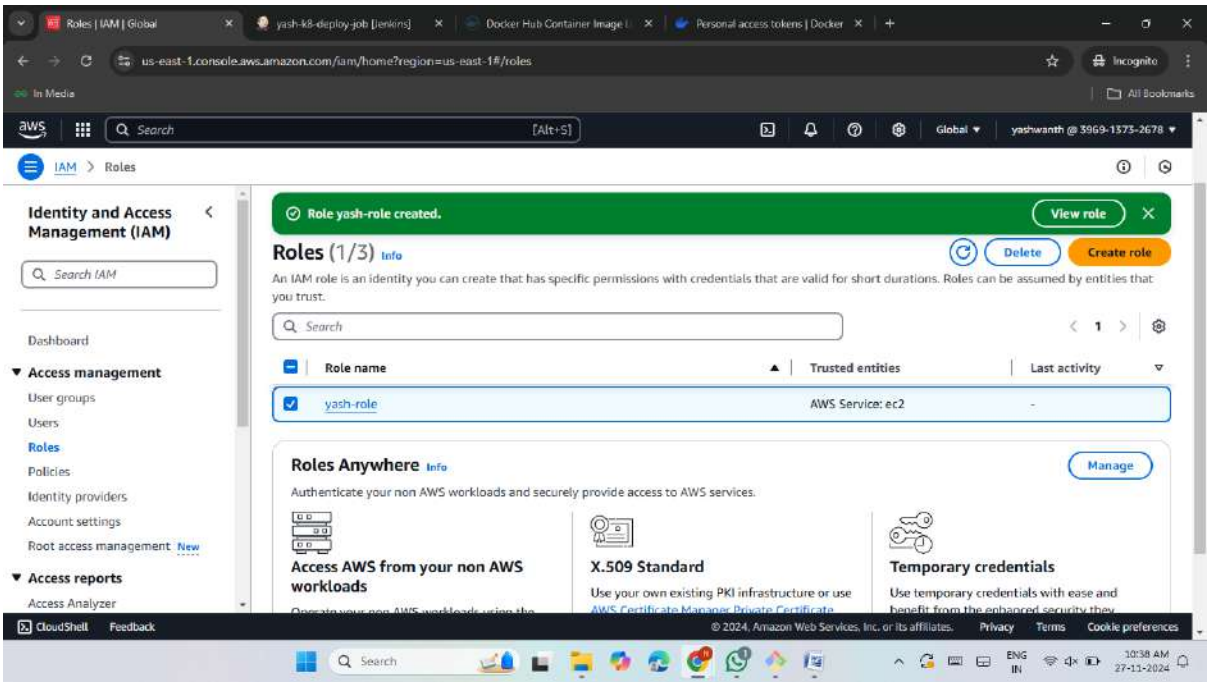
- Create a deployment.yaml file into github.

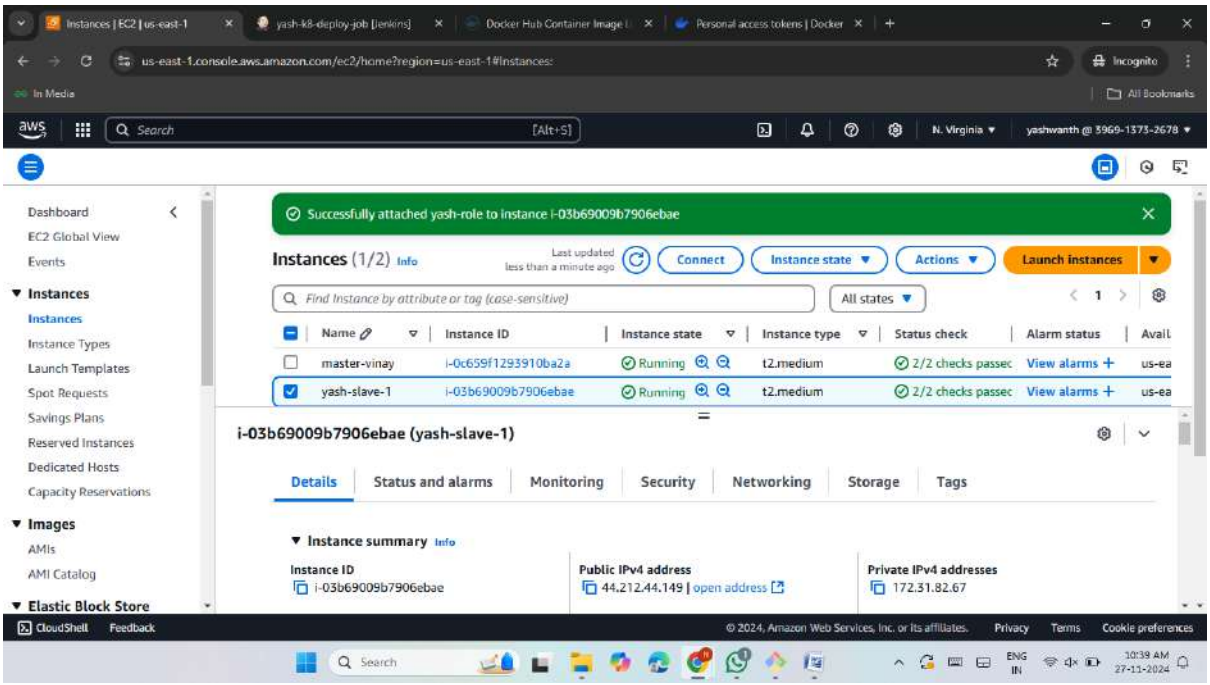


- Now create service.yaml file in github.



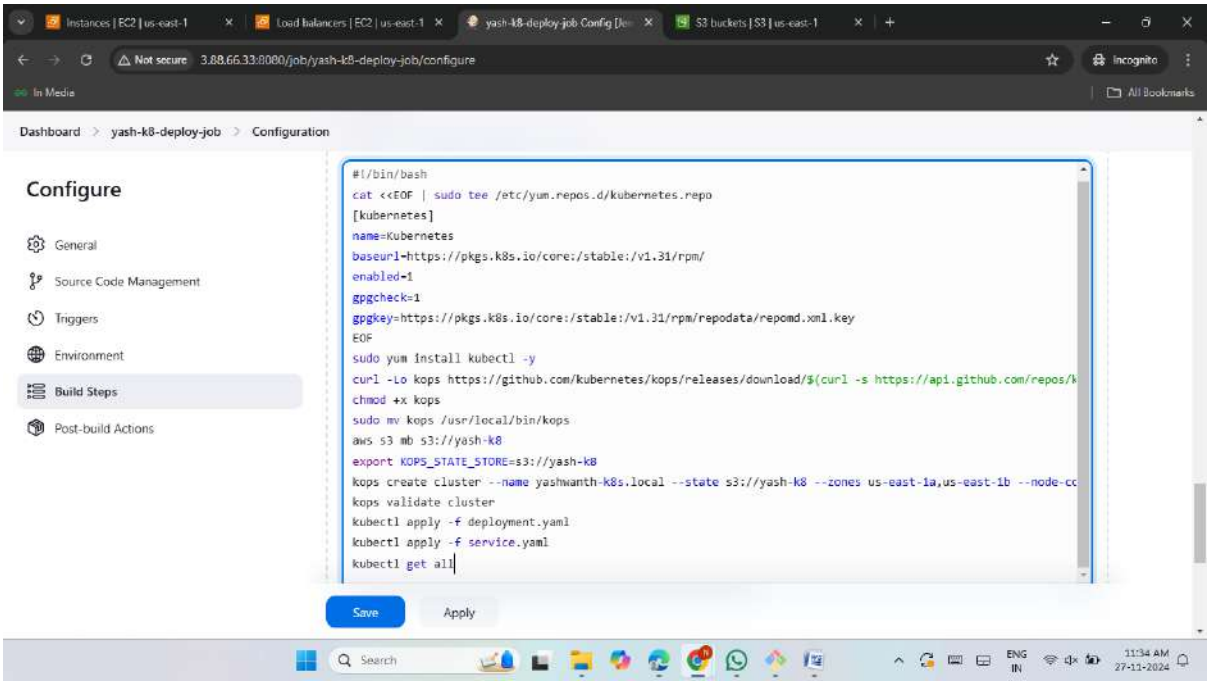
- Now create role with admin full permissions then attach to slave server.



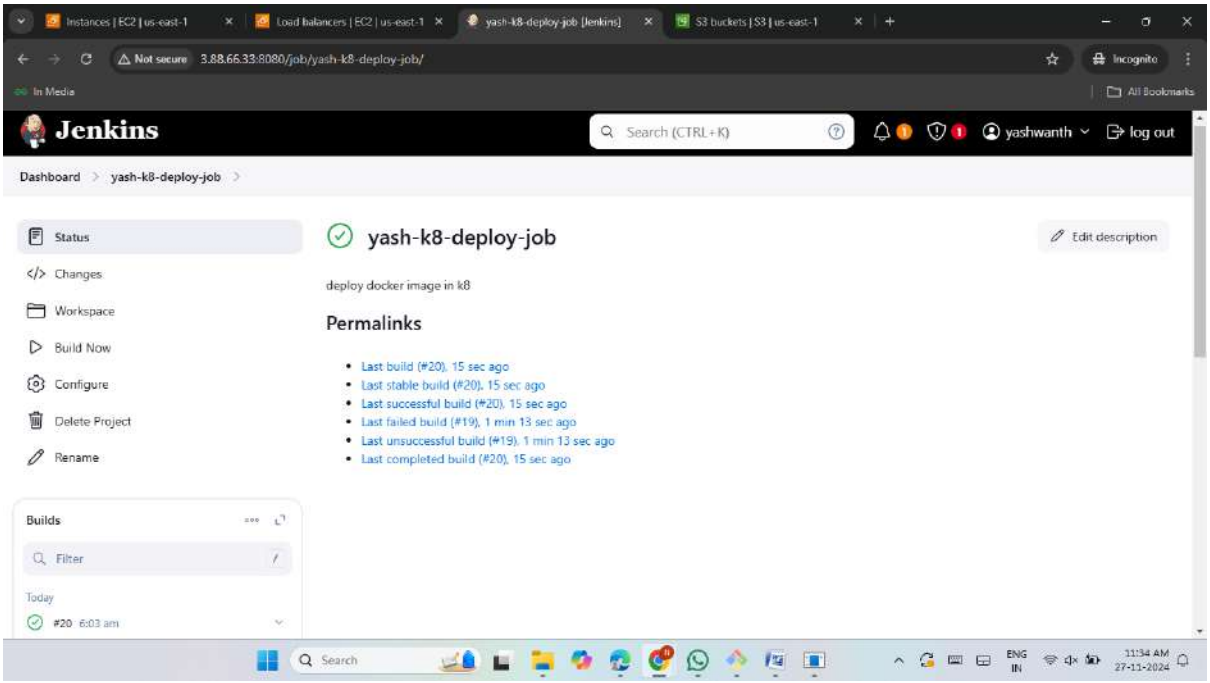


- Now write the following command to install kops and kubectl.
- Run the deployment and service files.

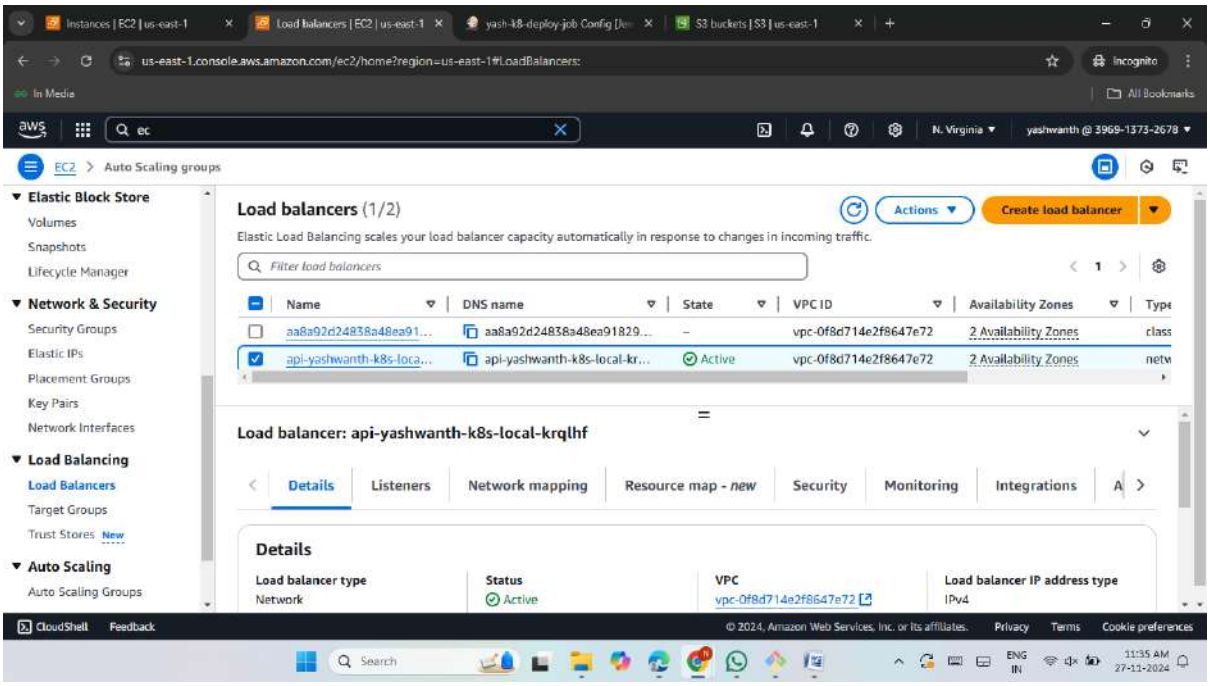
```
#!/bin/bash

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/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
```



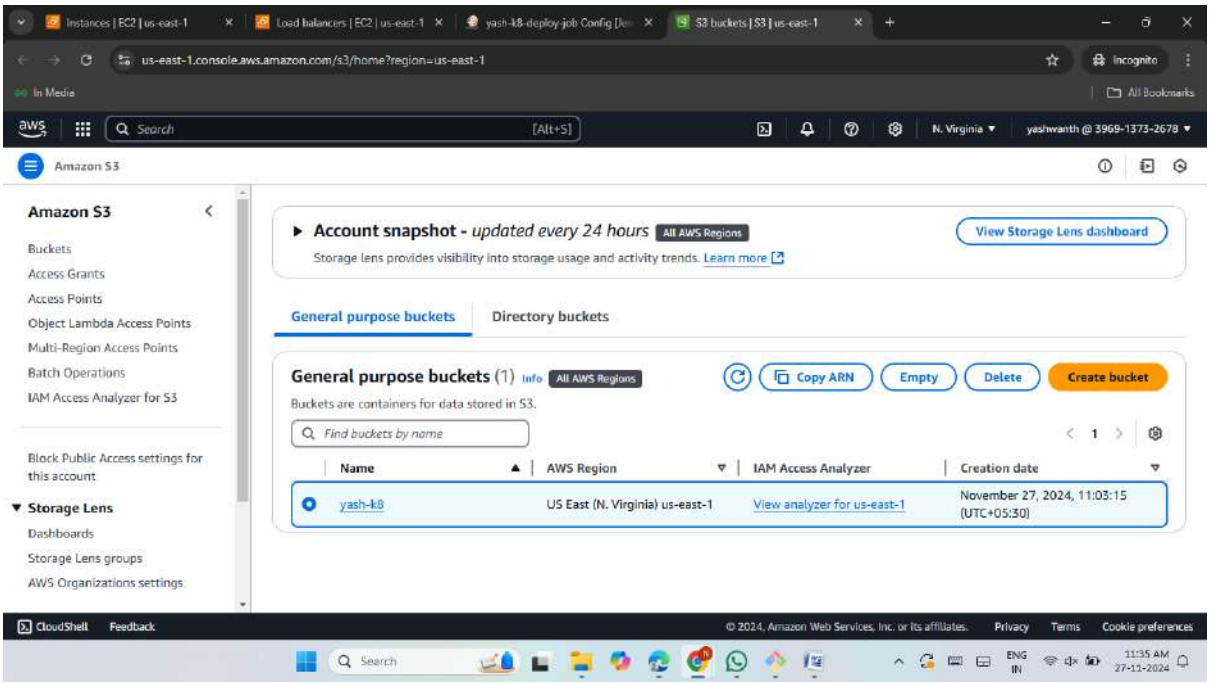
- Here the job was success.



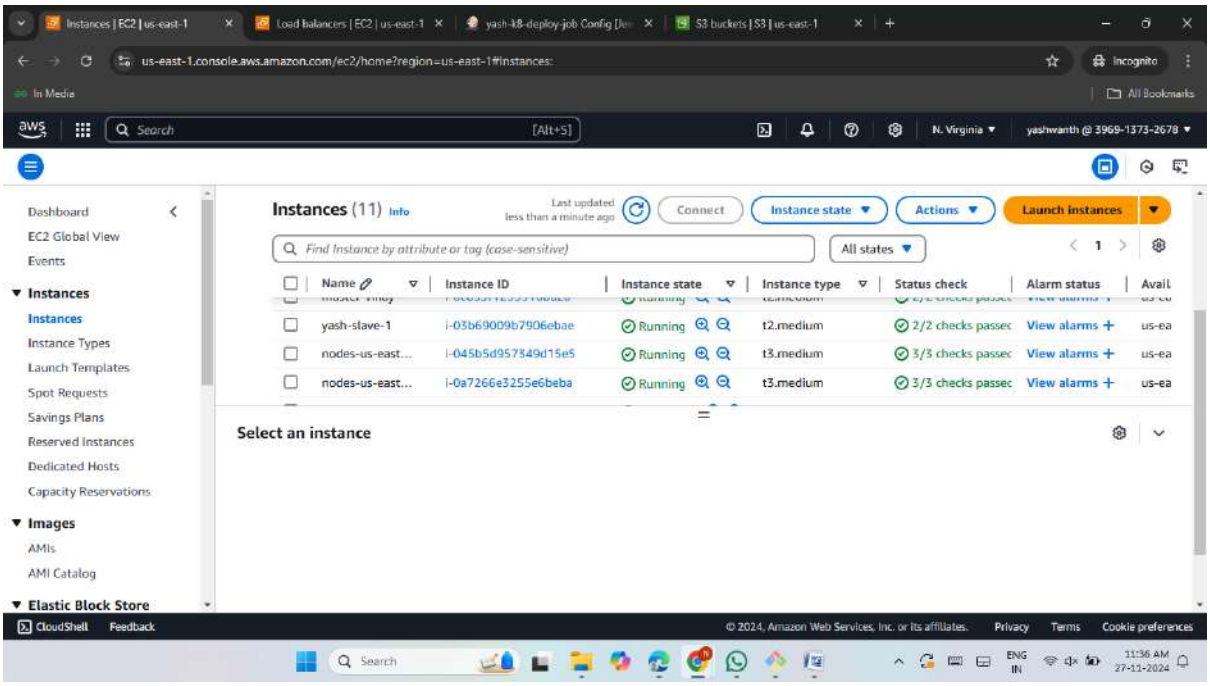
- Here the loadbalancer was successfully created.



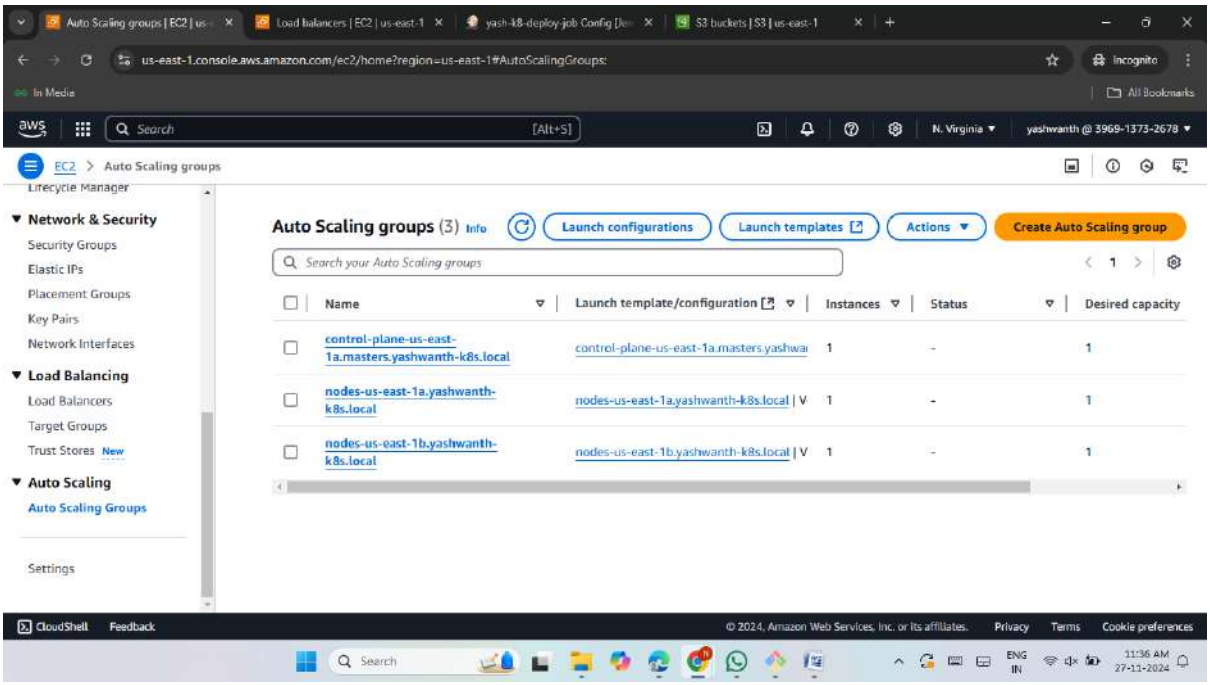
- Here the s3 was created.



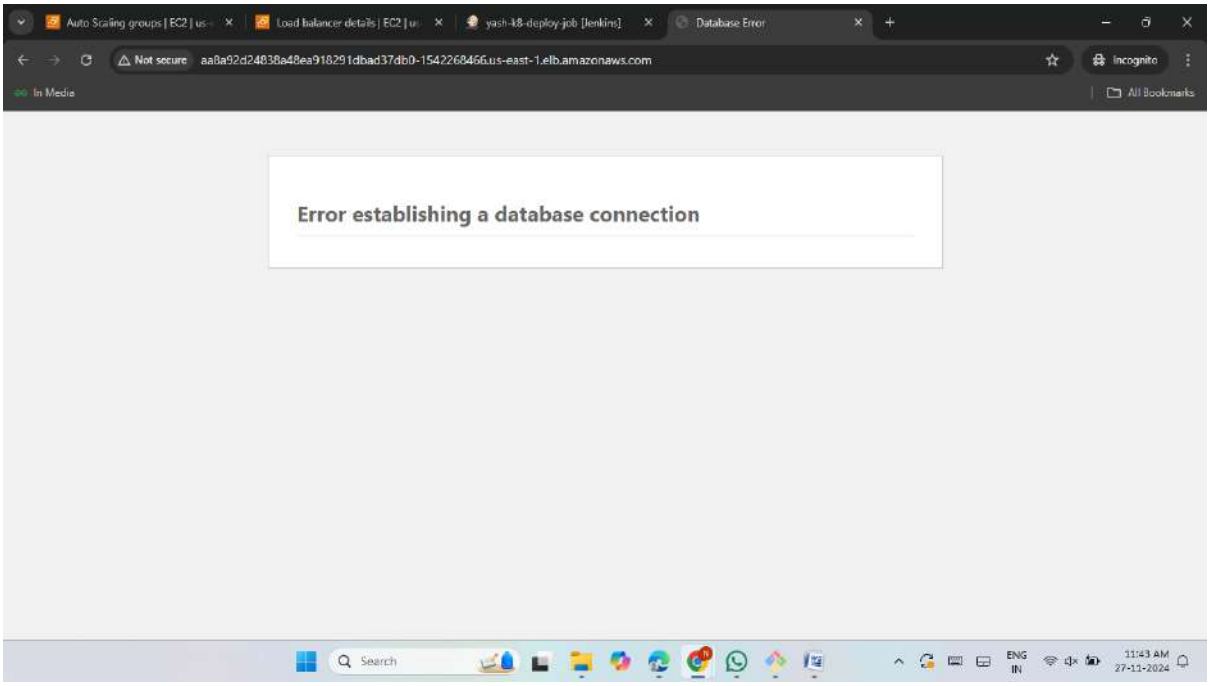
- Here the required instances also created.



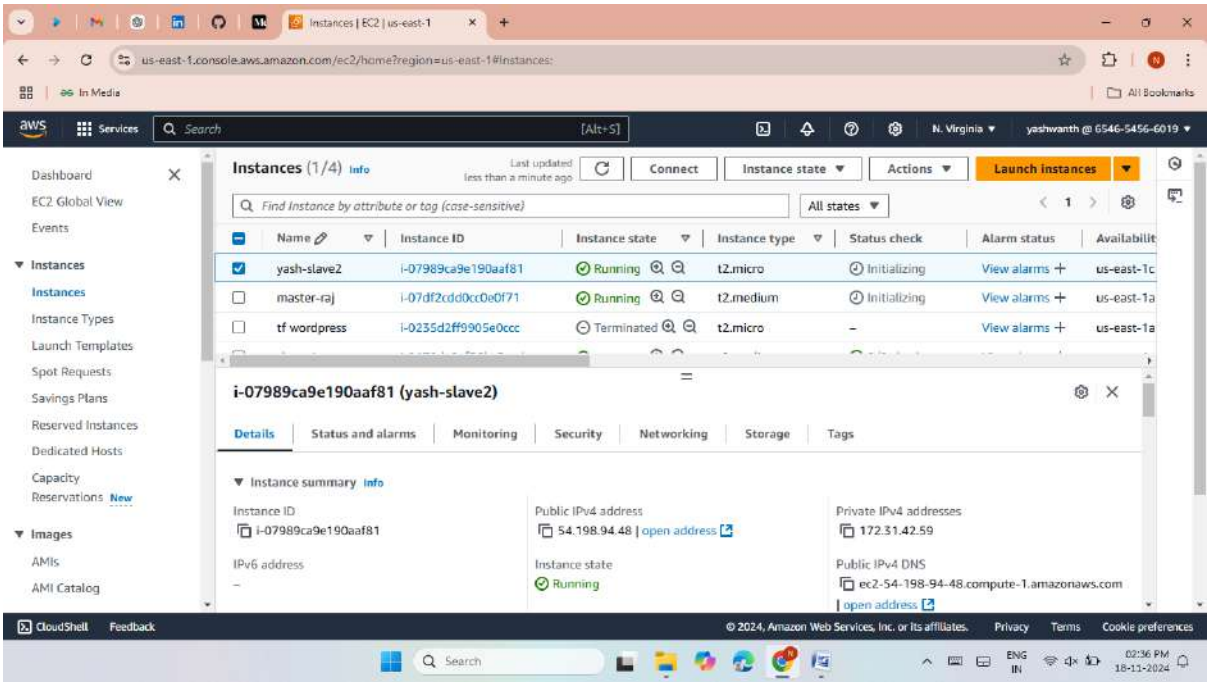
- Here the auto scaling was created.



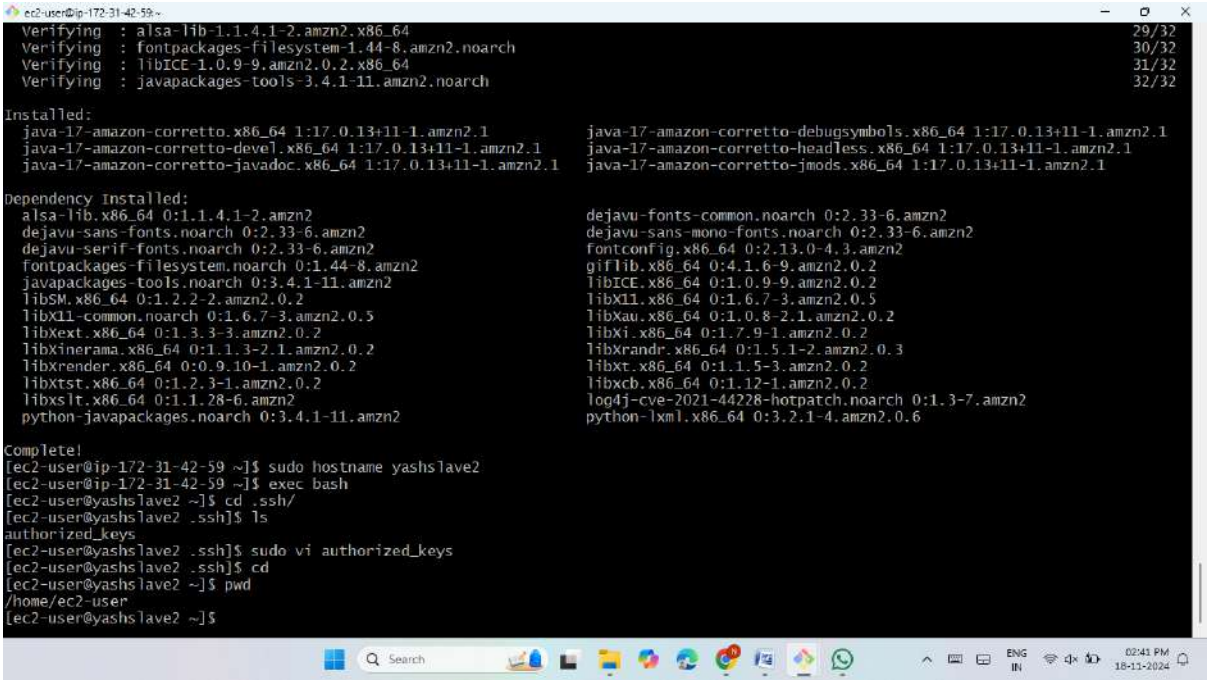
- Now copy the slave1 public ip and past it on google browser.
- The wordpress was hosted successfully.



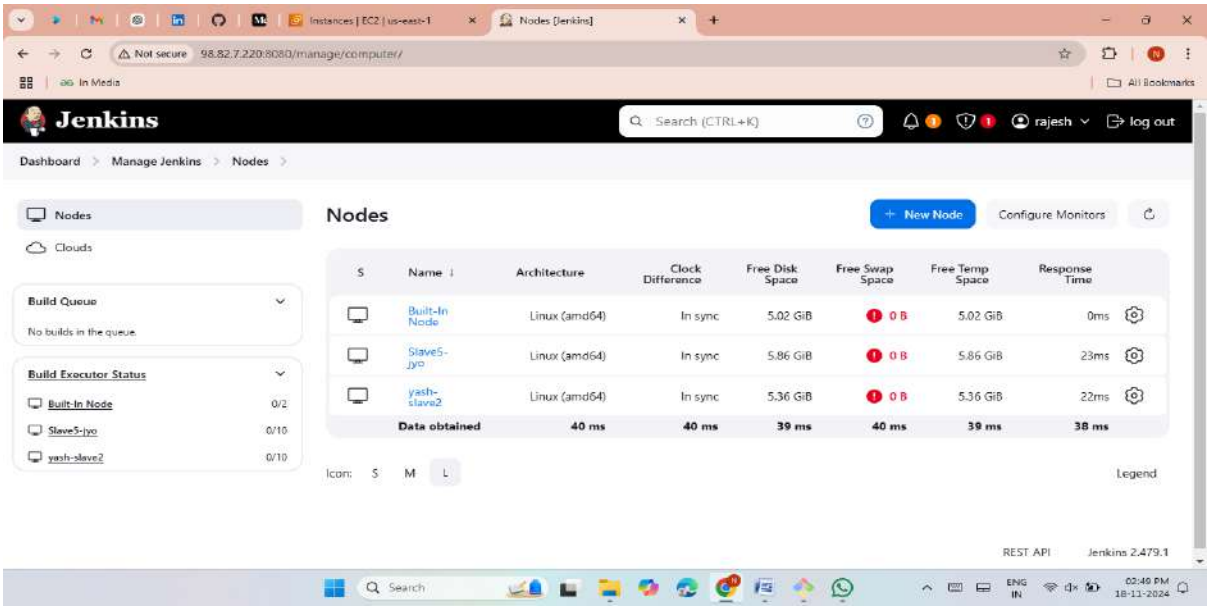
- Here I have worked as a slave-2 in a team project.
- Launch an ec2 instance along with required ports.



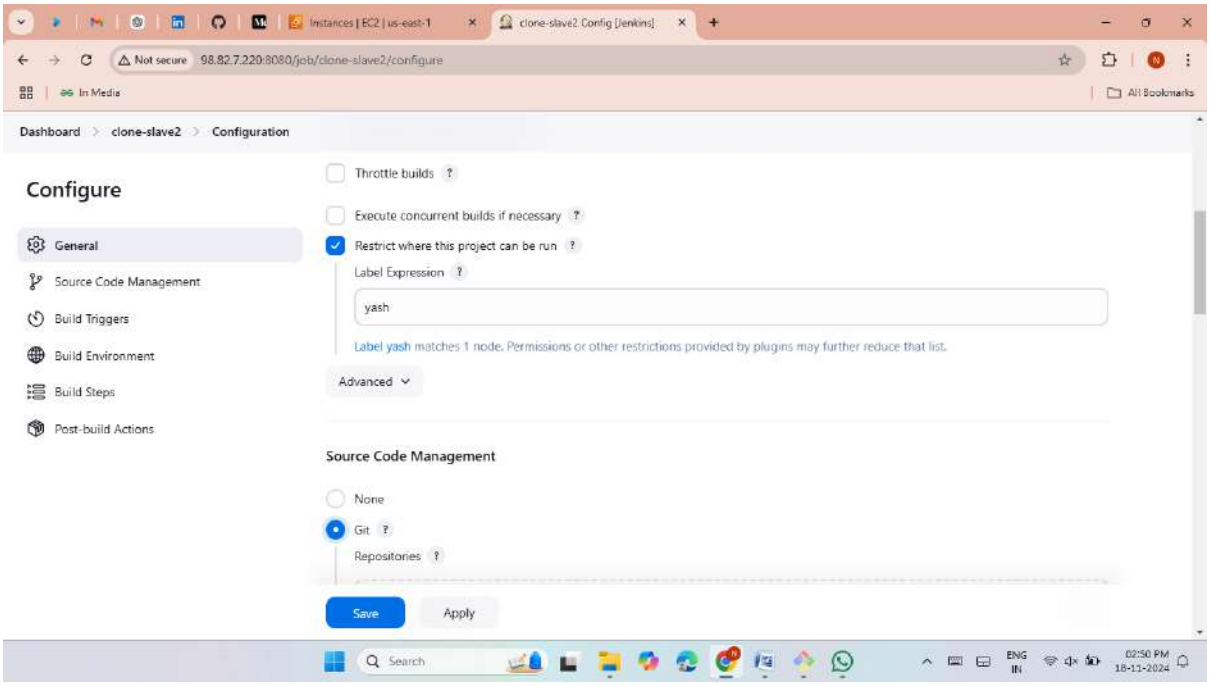
- Now connect Slave2 then install java.
- Now give private keys from master server.

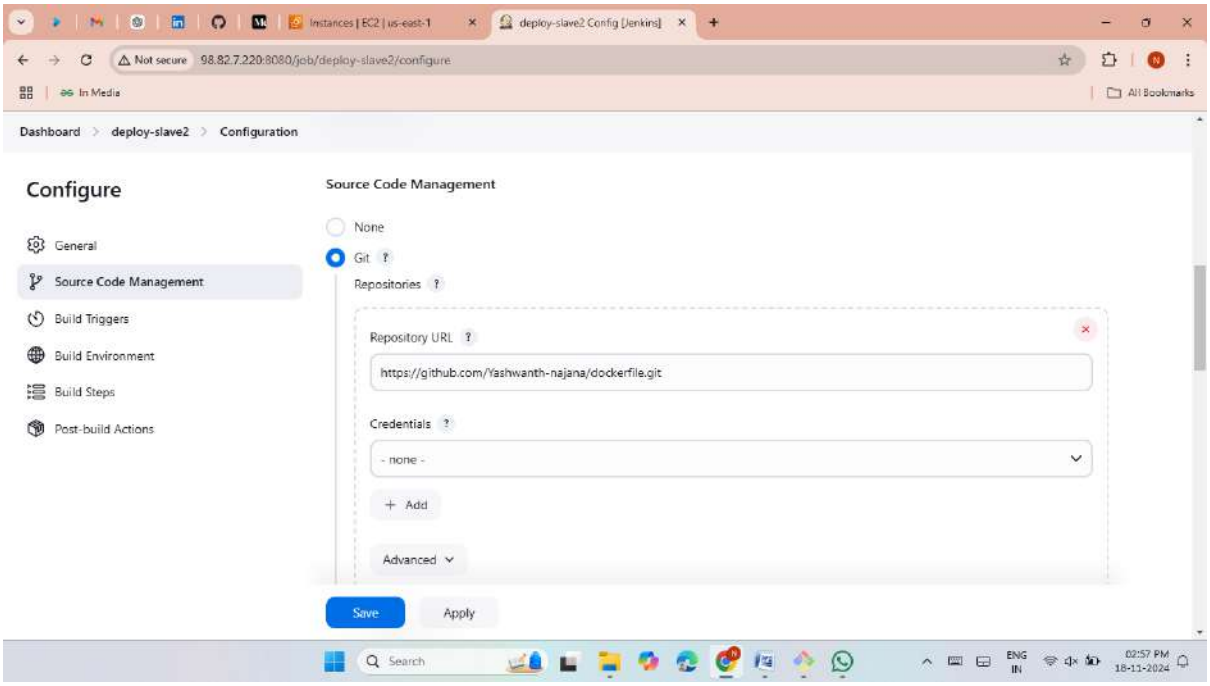


- Now create a node for slave2 server.
- Give details for slave node.
- Here we have to give slave2 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here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

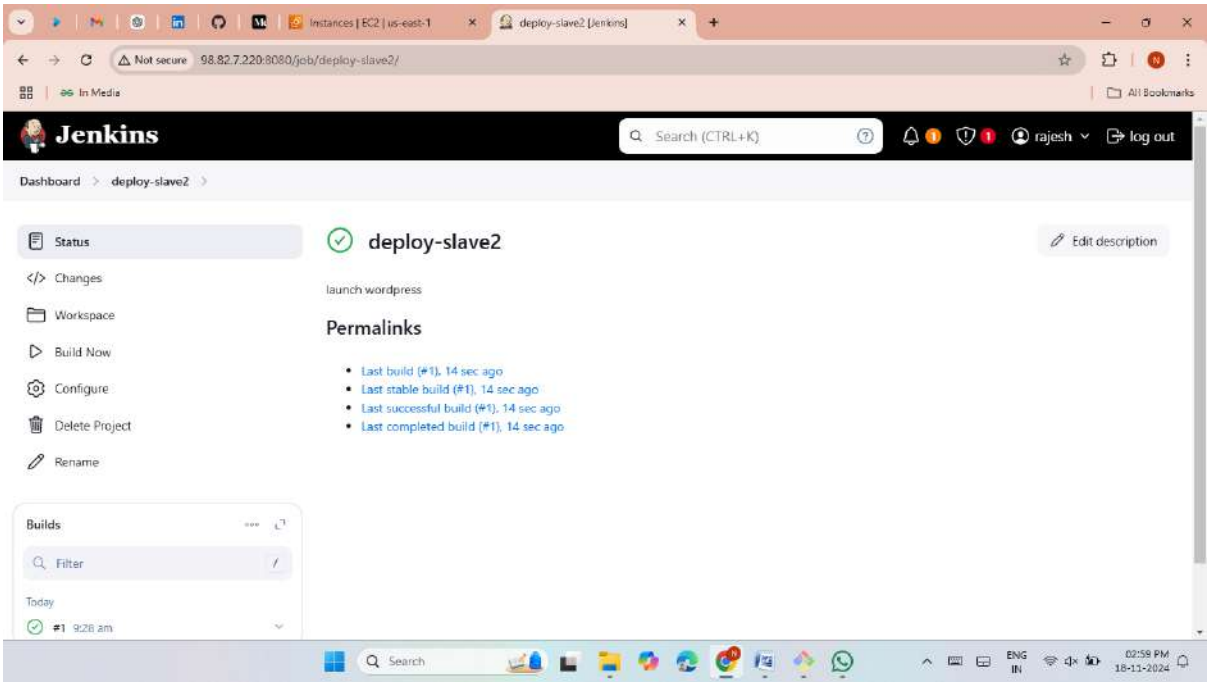


- Now create a job and give slave2 label.
- Now clone the wordpress application repo form github.
- Now Click on save.





- 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

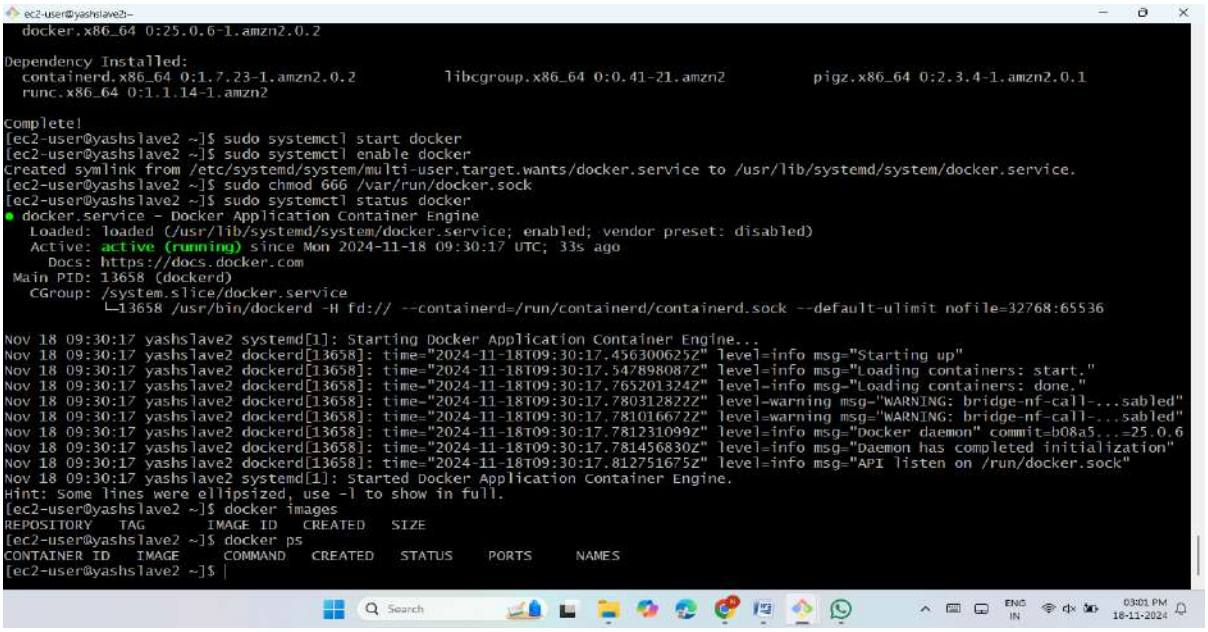
sudo systemctl enable docker

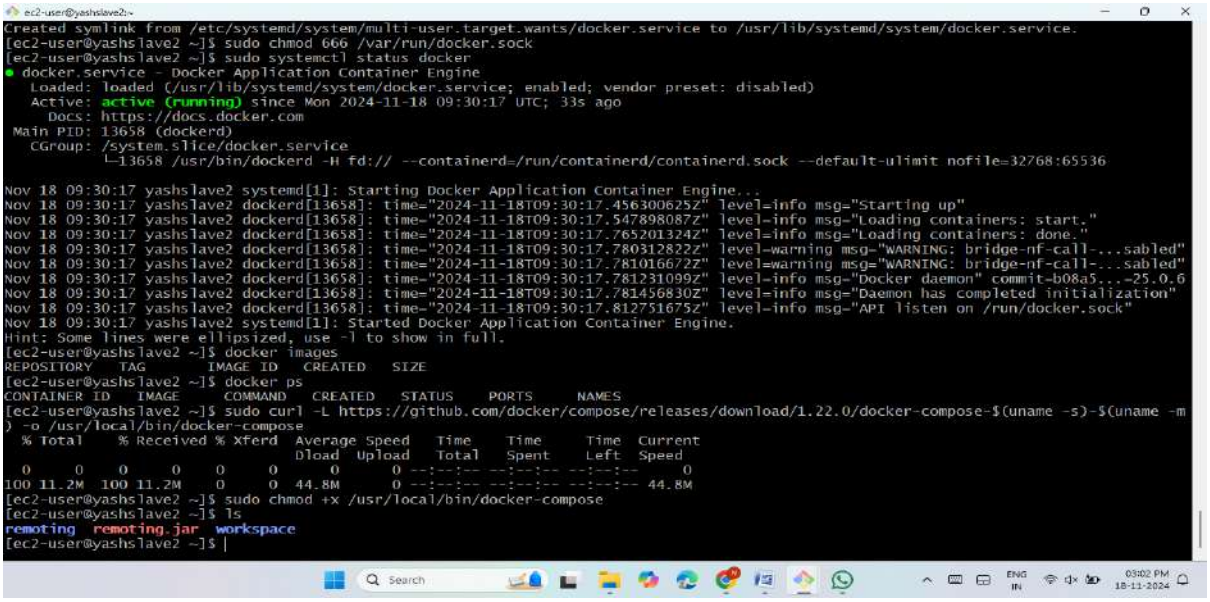
sudo chmod 666 /var/run/docker.sock

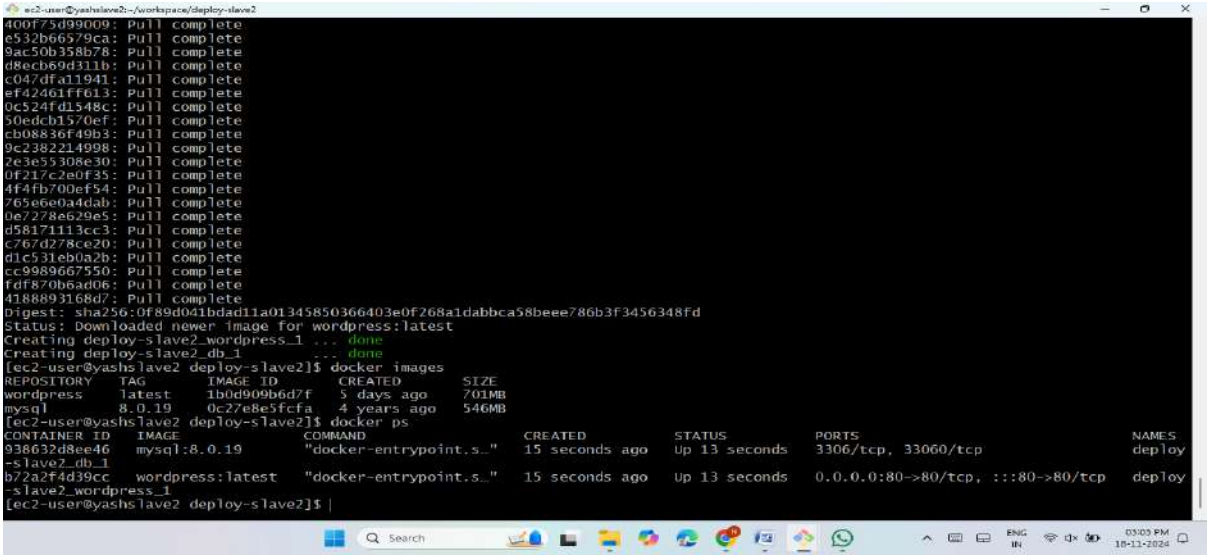
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

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

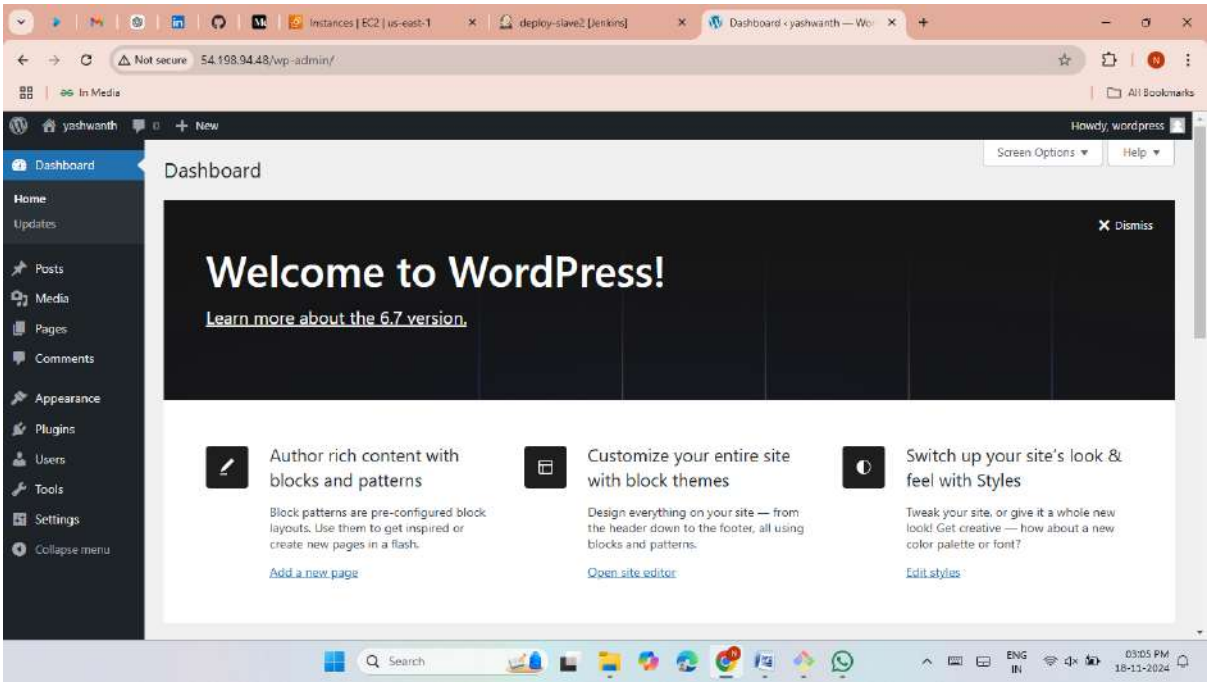
docker-compose up –d


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 ~]$

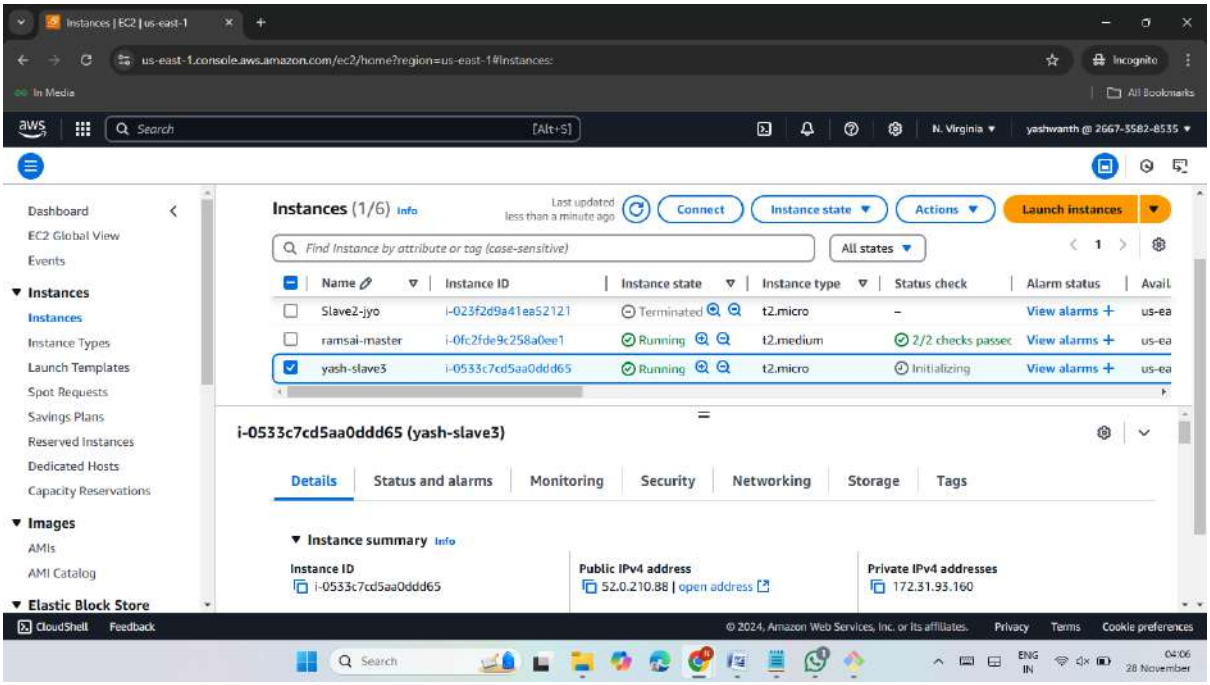

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
[ec2-user@yashslave2 ~]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@yashslave2 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
Loaded: 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 ~]$ 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 Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0
100 11.2M 100 11.2M 0 0 44.8M 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 ~]$


400f75d99009: Pull complete
e532b66579ca: Pull complete
9ac50b358b78: Pull complete
d8ecb69d311b: Pull complete
c047dfa11941: Pull complete
ef42461ff613: Pull complete
0c524fd1548c: Pull complete
50edcb1570ef: Pull complete
cb08836f49b3: Pull complete
9c2382214998: Pull complete
2e3e55308e30: Pull complete
0f217c2e0f35: Pull complete
4f4fb700ef54: Pull complete
765e6e0a4dab: Pull complete
0e7278e629e5: Pull complete
d58171113cc3: Pull complete
c767d278ce20: Pull complete
d1c531eb0a2b: Pull complete
cc9989667550: Pull complete
fdf870b6ad06: Pull complete
4188893168d7: Pull complete
Digest: sha256:0f89d041bdad11a01345850366403e0f268a1dabbca58beee786b3f3456348fd
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 1b0d909b6d7f 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 deploy-slave2_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-slave2_wordpress_1
[ec2-user@yashslave2 deploy-slave2]$

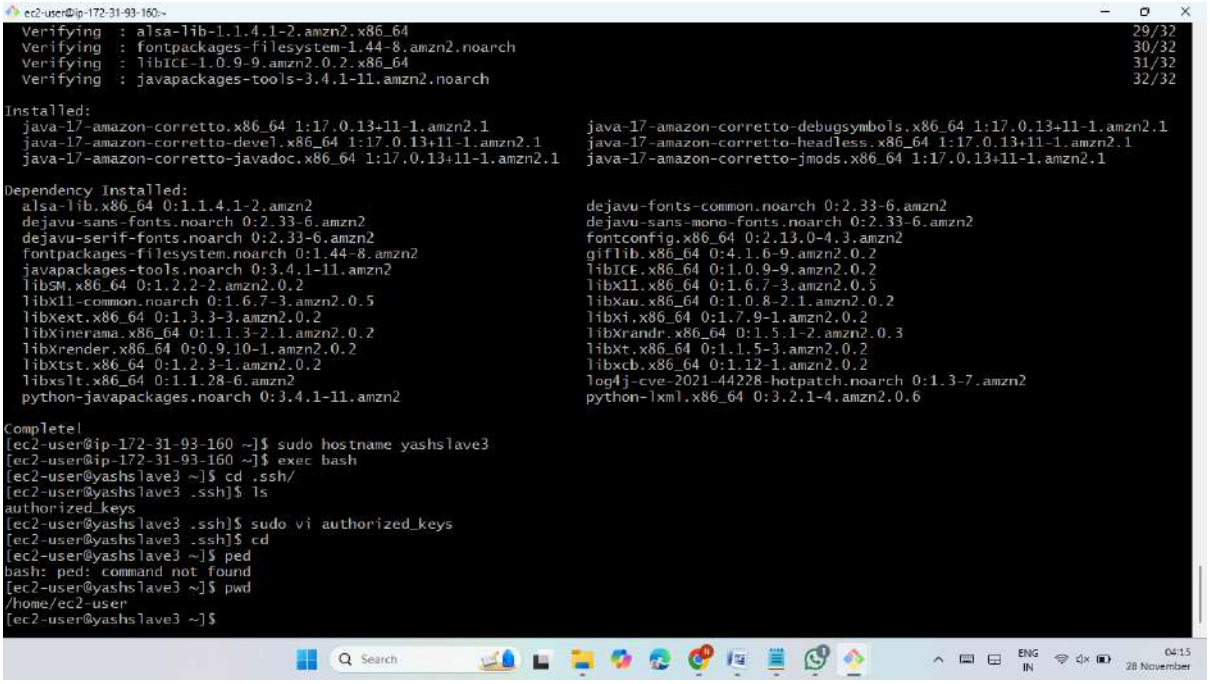
- Now copy the slave2 public ip and past it on google browser.
- The wordpress was hosted successfully.



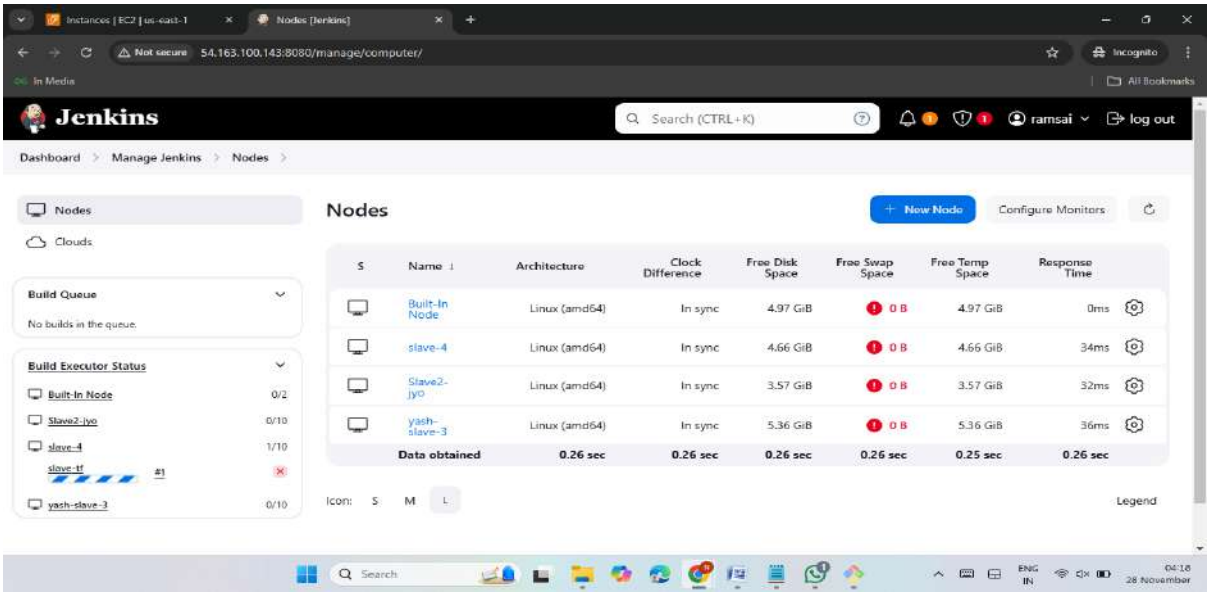
- Here I have worked as a slave-3 in a team project.
- Launch an ec2 instance along with required ports.



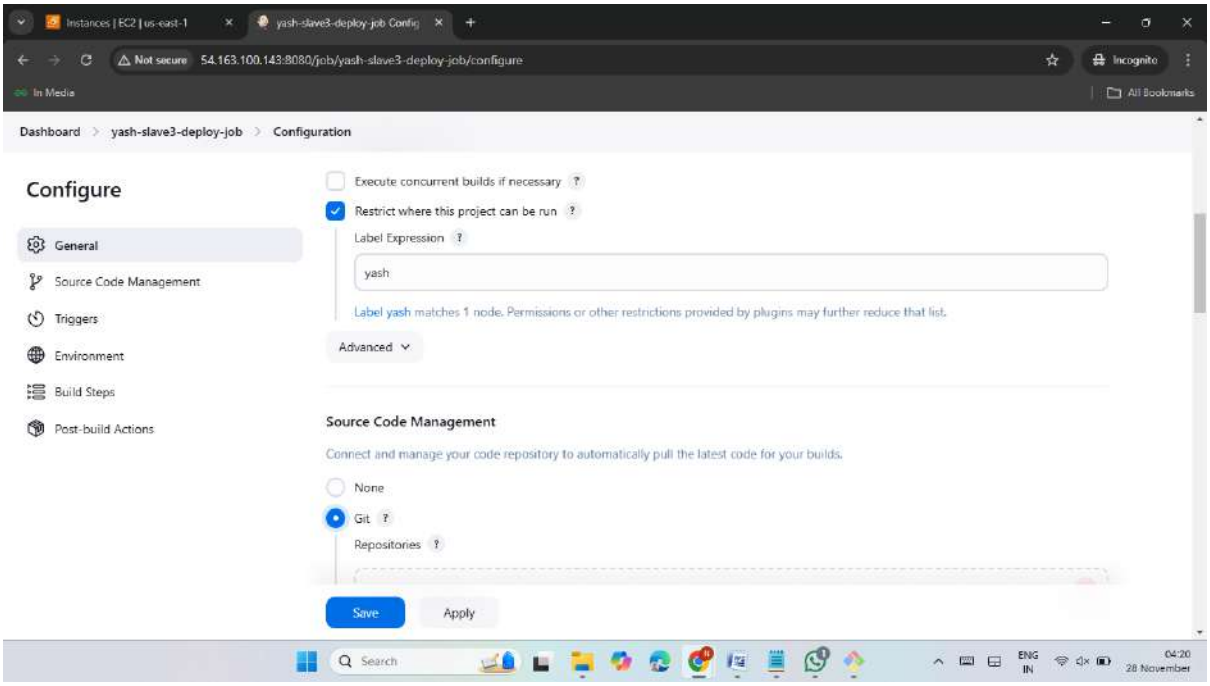
- Now connect Slave3 then install java.
- Now give private keys from master server.

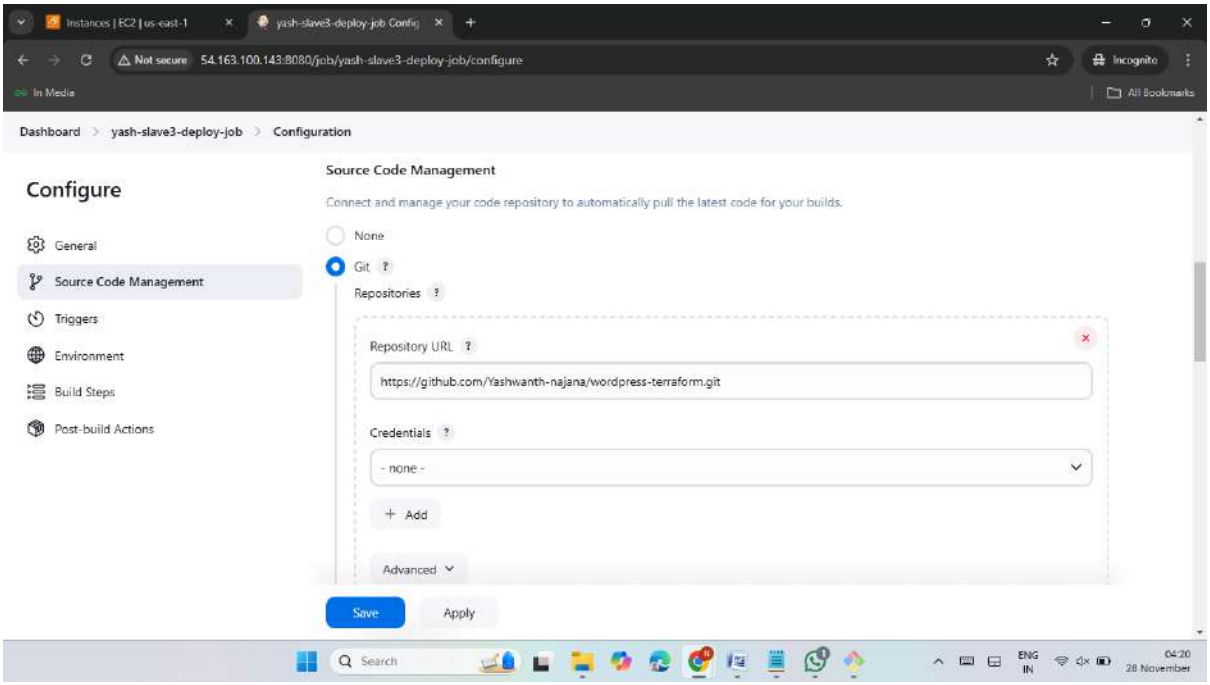


- Now create a node for slave3 server.
- Give details for slave node.
- Here we have to give slave3 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here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

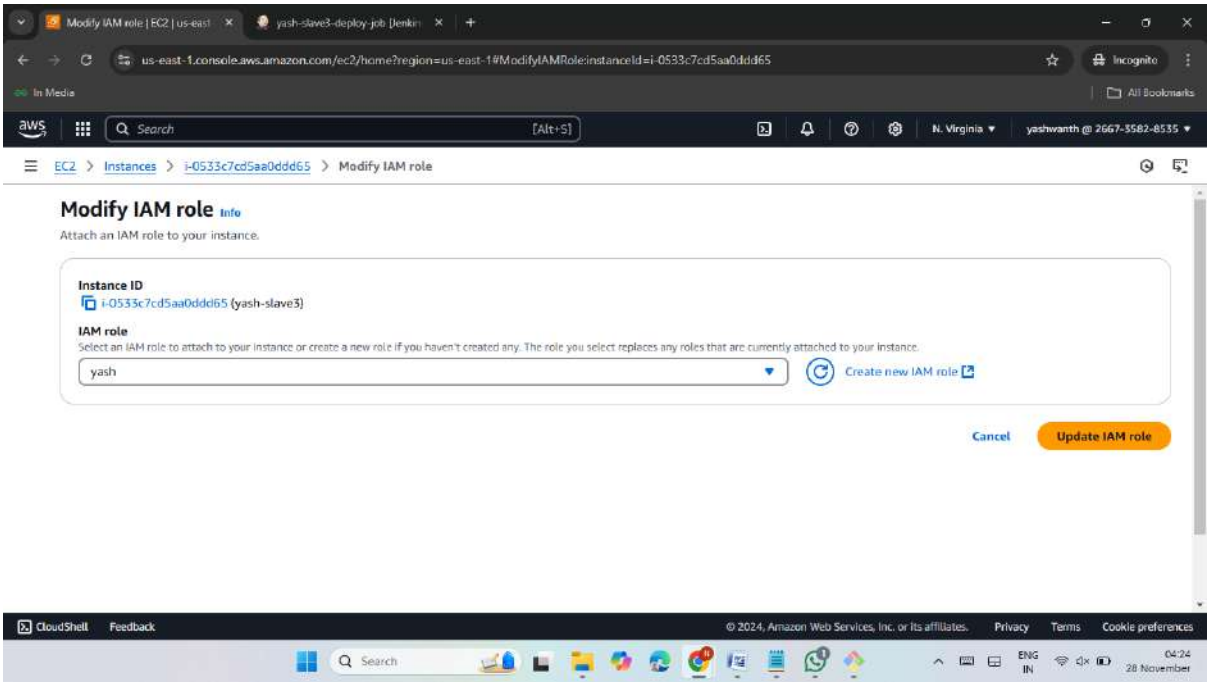


- Now create a job and give slave3 label.
- Now clone the wordpress application repo form github.
- Now Click on save.





- 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

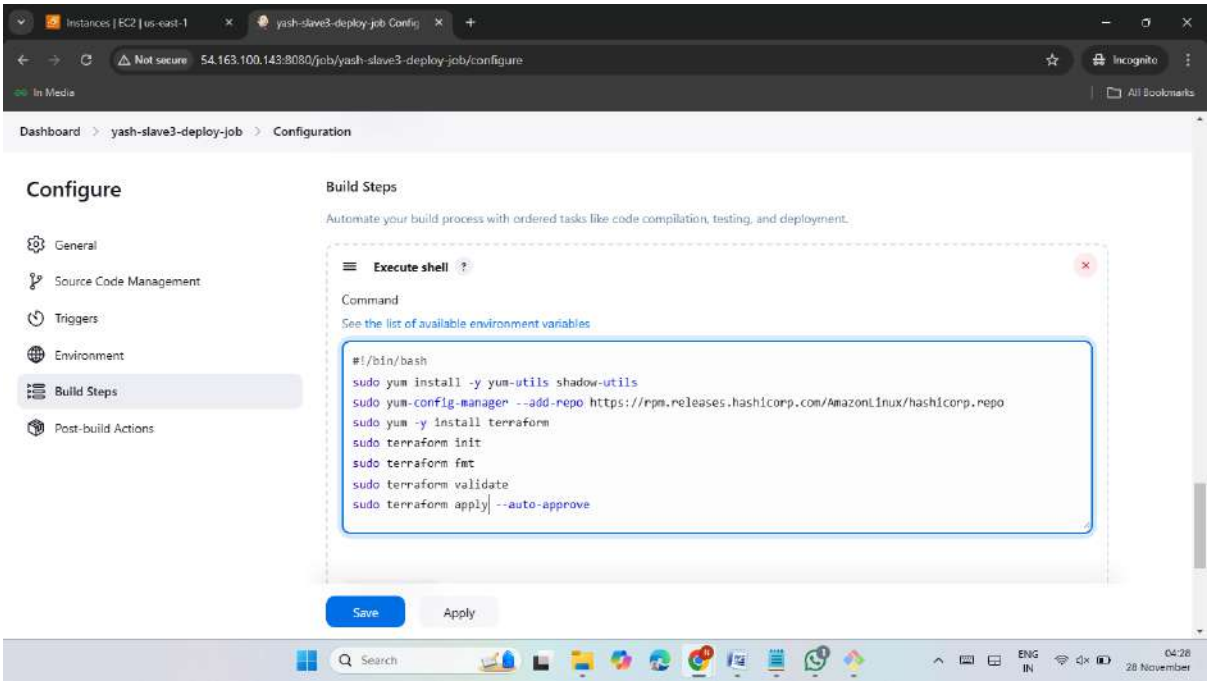
sudo yum -y install terraform

sudo terraform init

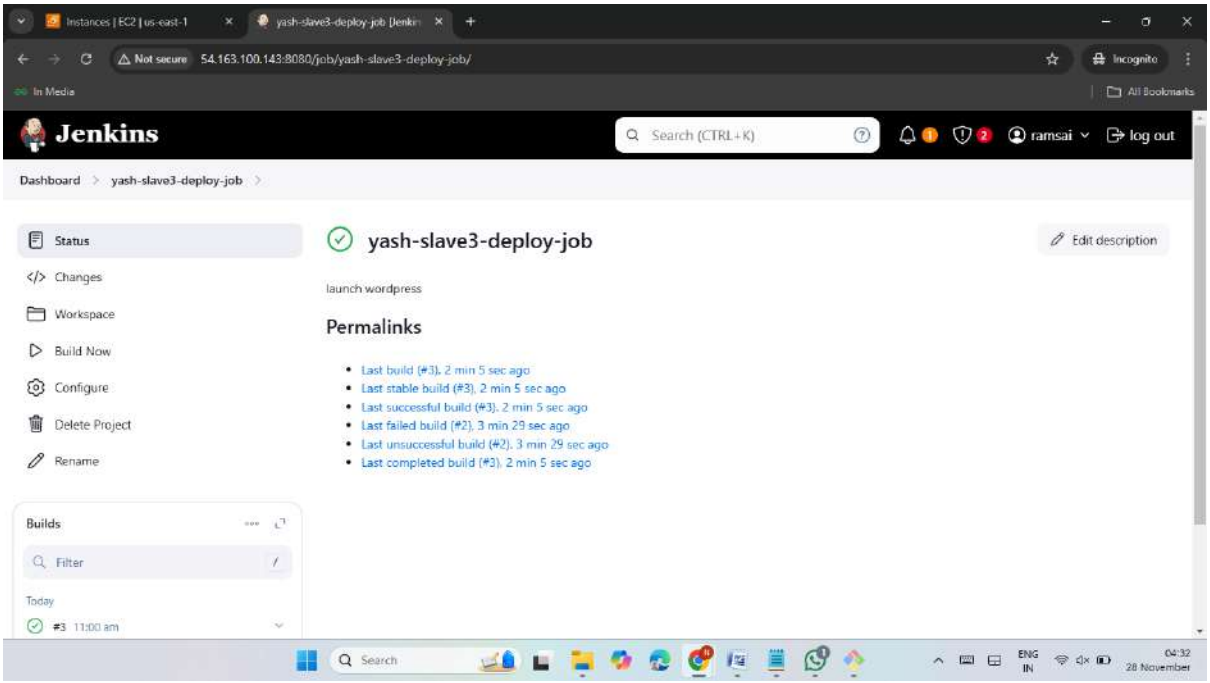
sudo terraform fmt

sudo terraform validate

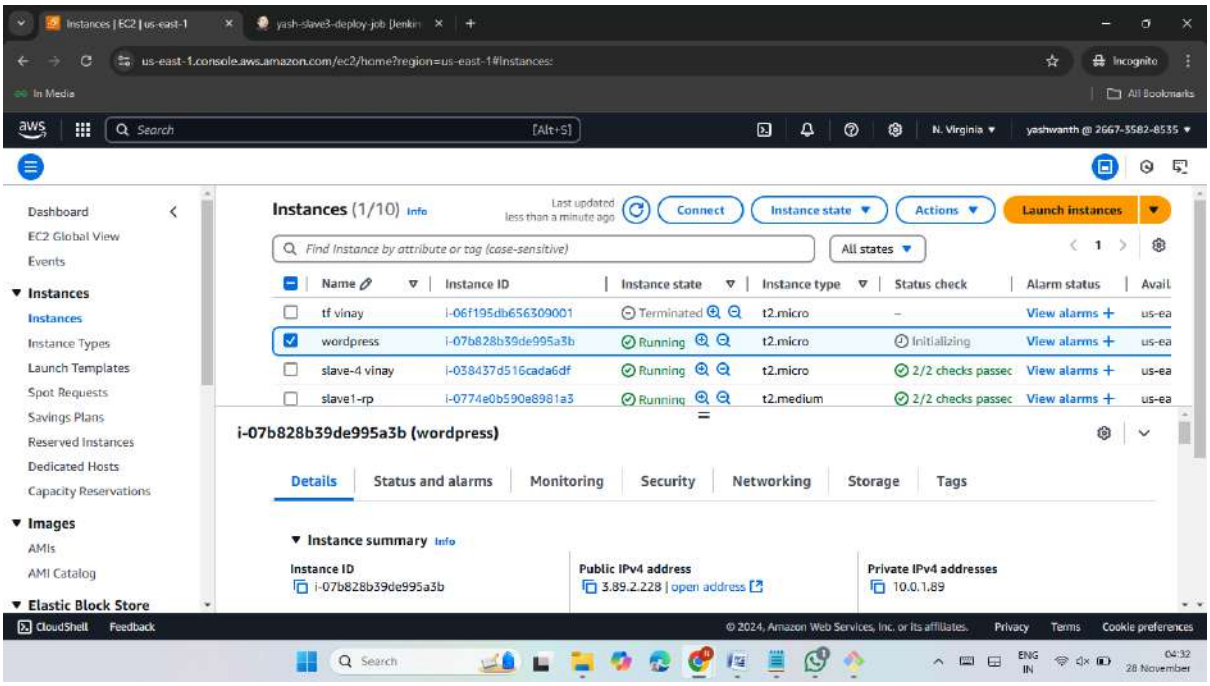
sudo terraform apply --auto-approve



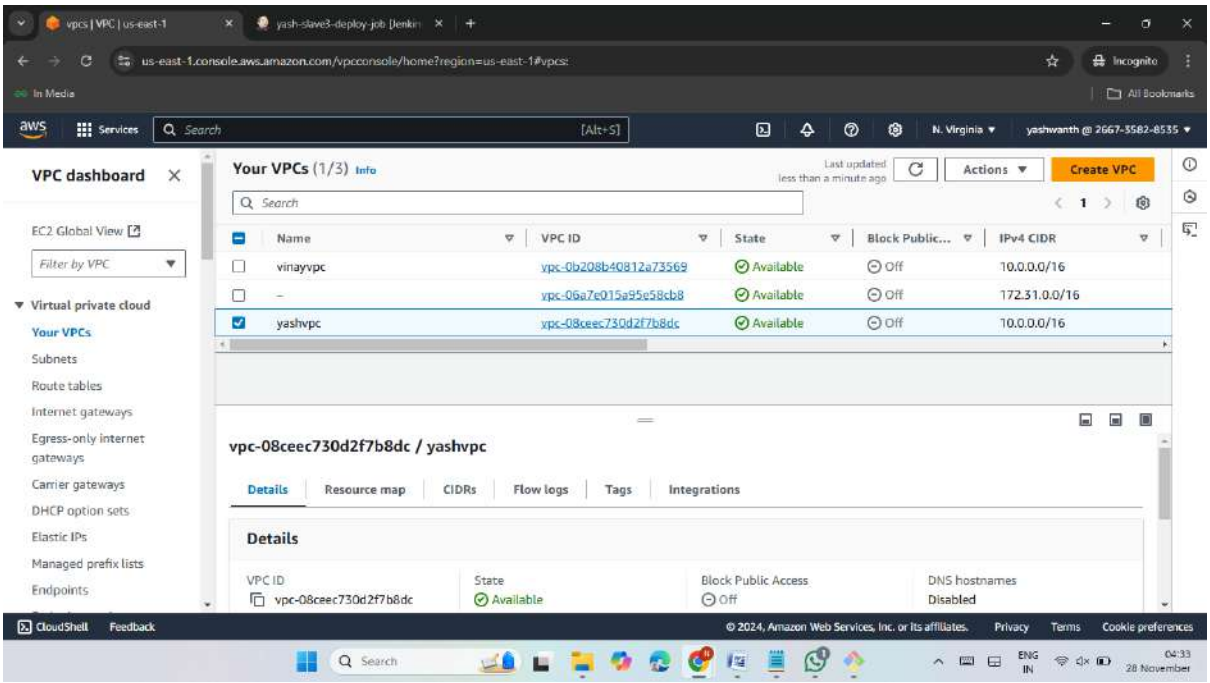
- Now click on build now.
- Here the job was success.



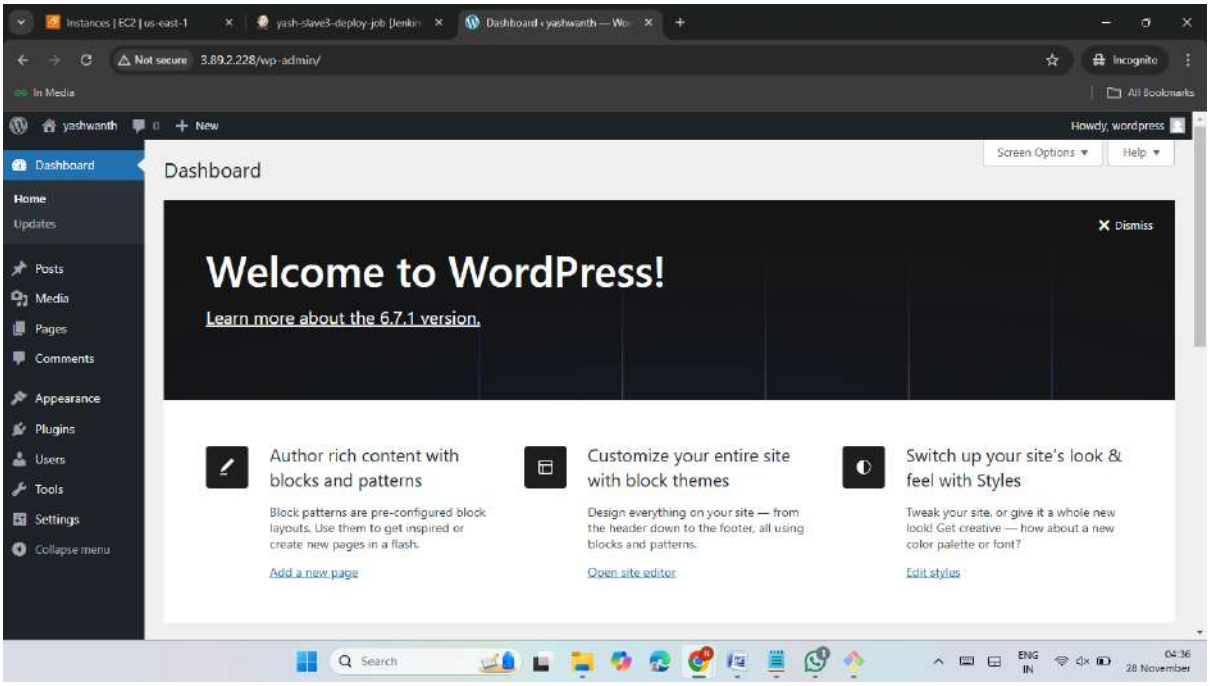
- Here the instance was launched.



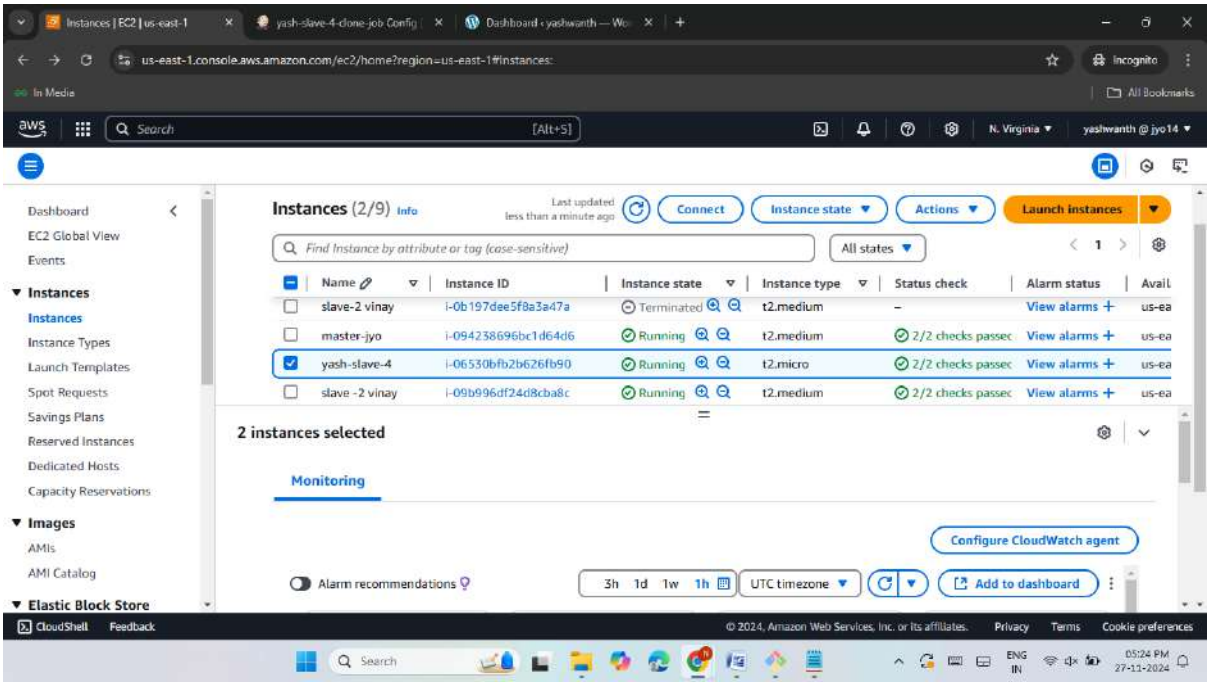
- Here the VPC was launched.



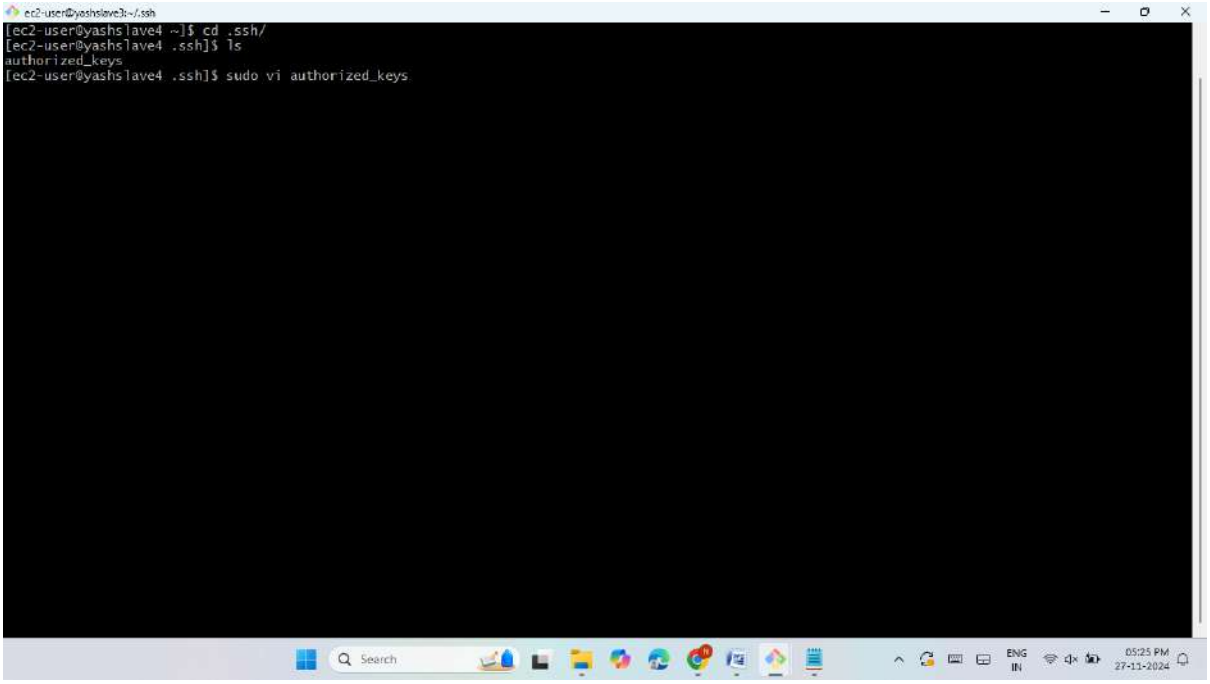
- Now copy the created instance public ip and past it on Google browser.
- The wordpress was hosted successfully.



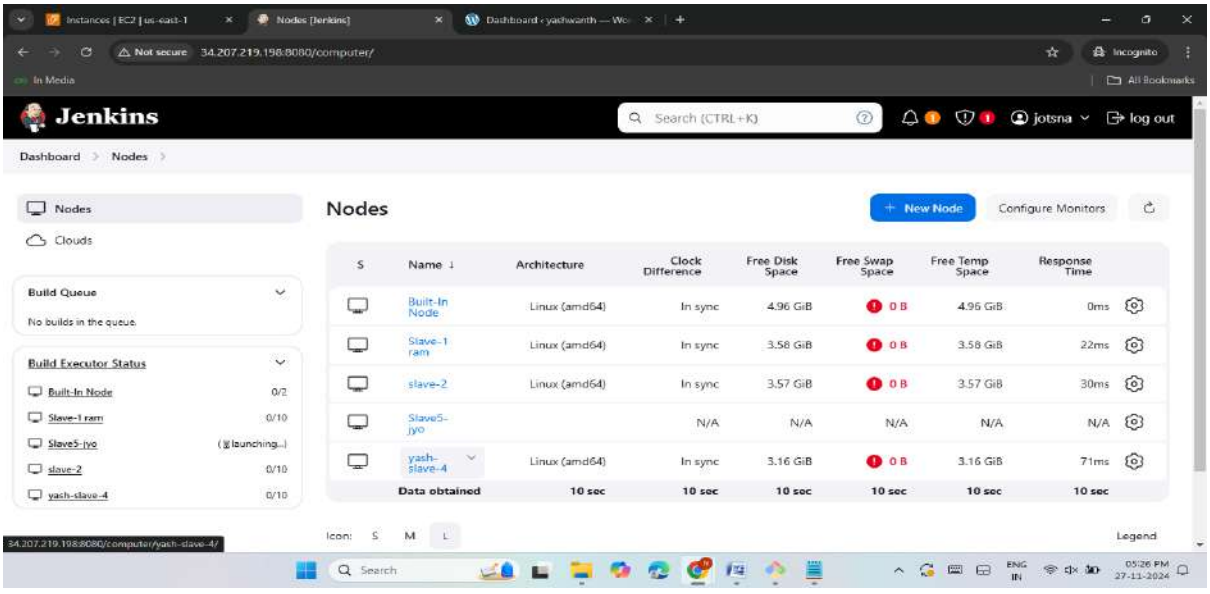
- Here I have worked as a slave-4 in a team project.
- Launch an ec2 instance along with required ports.



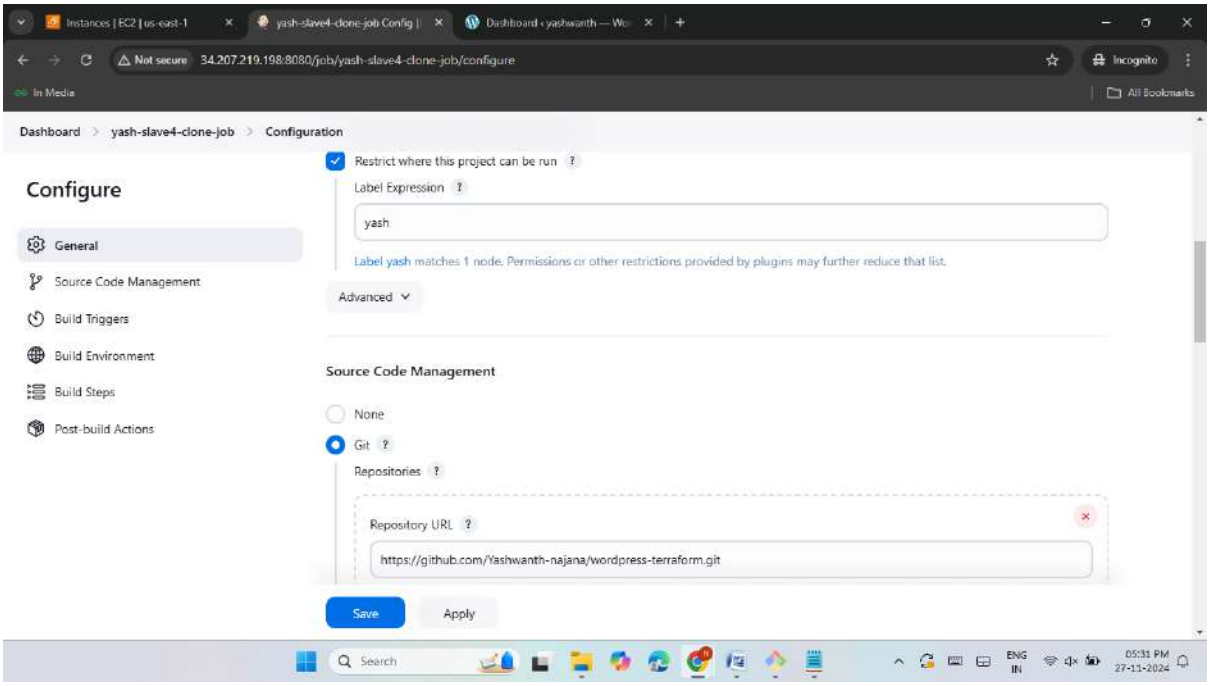
- Now connect Slave4 then install java.
- Now give private keys from master server.



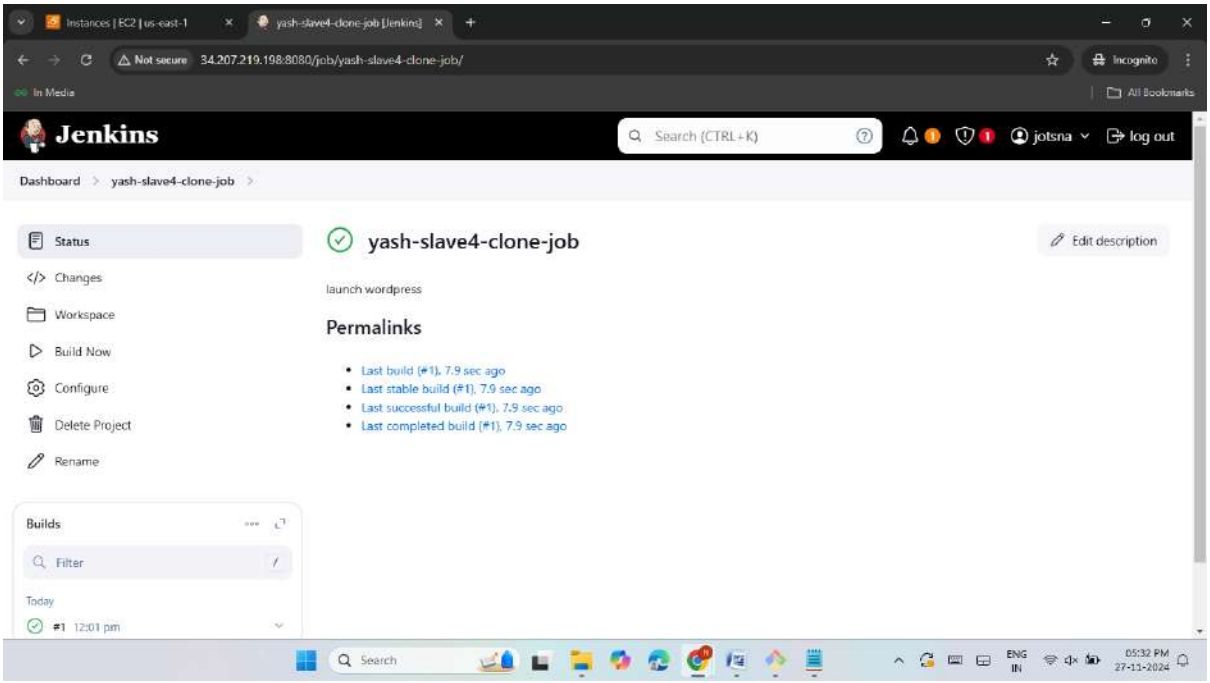
- Now create a node for slave4 server.
- Give details for slave node.
- Here we have to give slave4 path of root directory and private ip.
- We have to create credentials with ssh-username with private password.
- In the credentials we have to give master server private keys.
- Now click on save.
- Here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



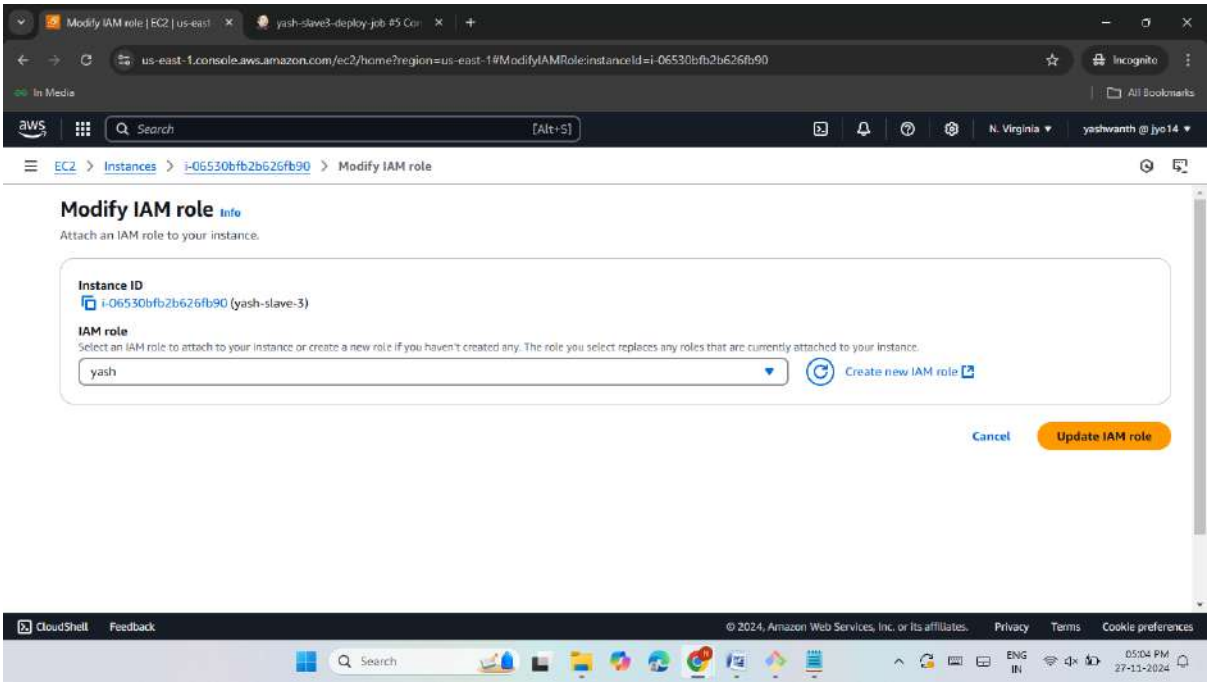
- Now create a job and give slave4 label.
- Now clone the wordpress application repo form github.
- Now Click on save.



- 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 antother 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

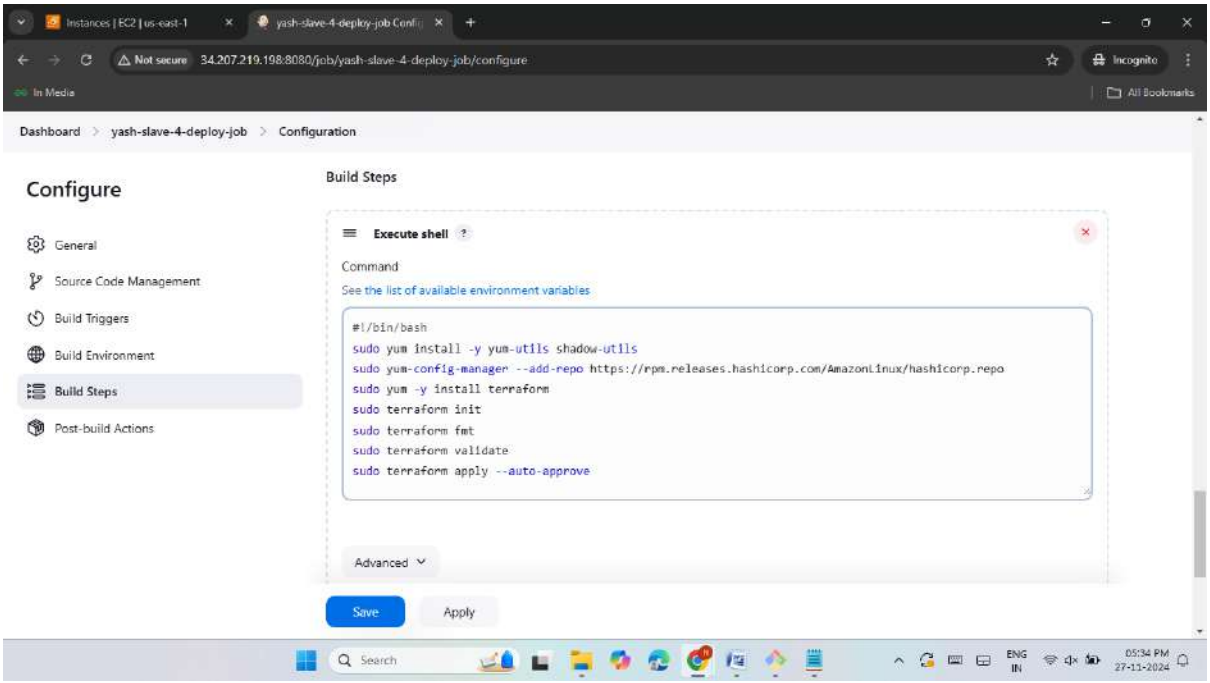
sudo yum -y install terraform

sudo terraform init

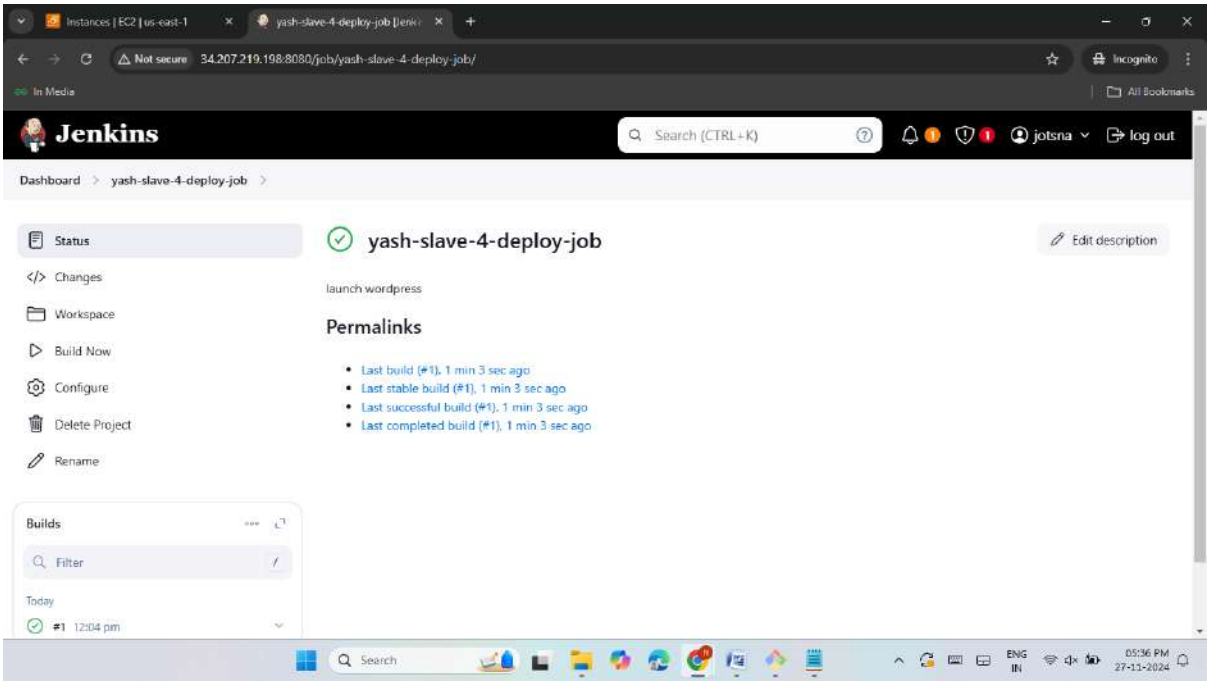
sudo terraform fmt

sudo terraform validate

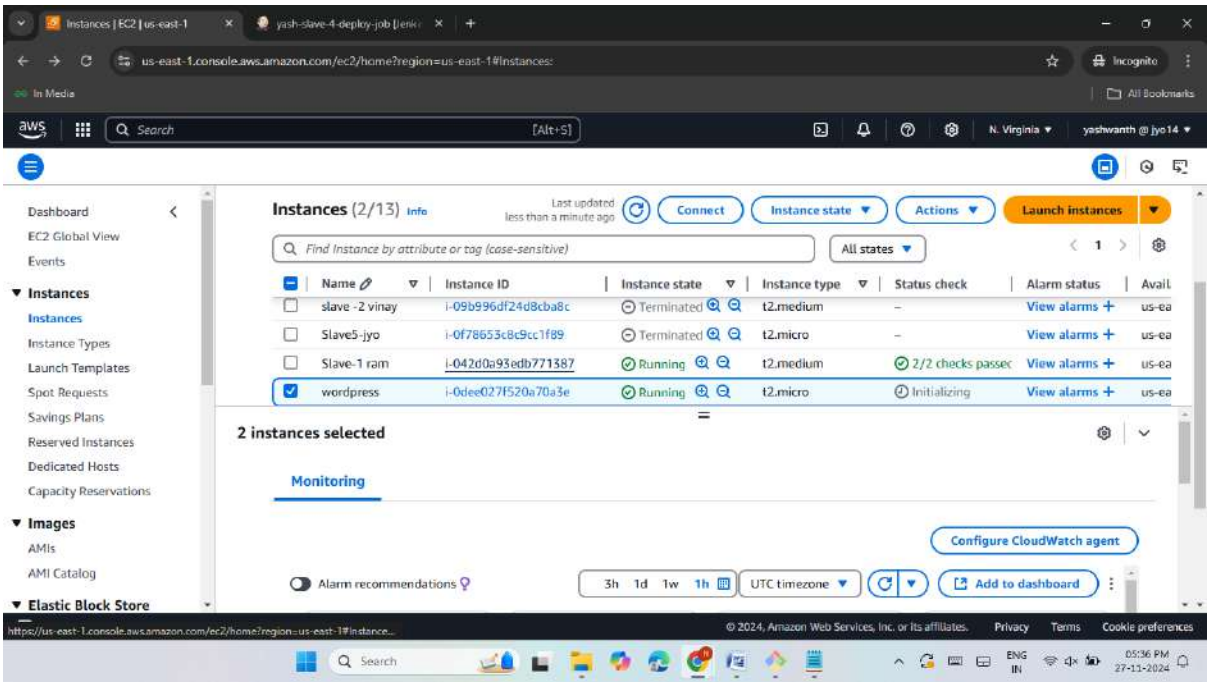
sudo terraform apply --auto-approve



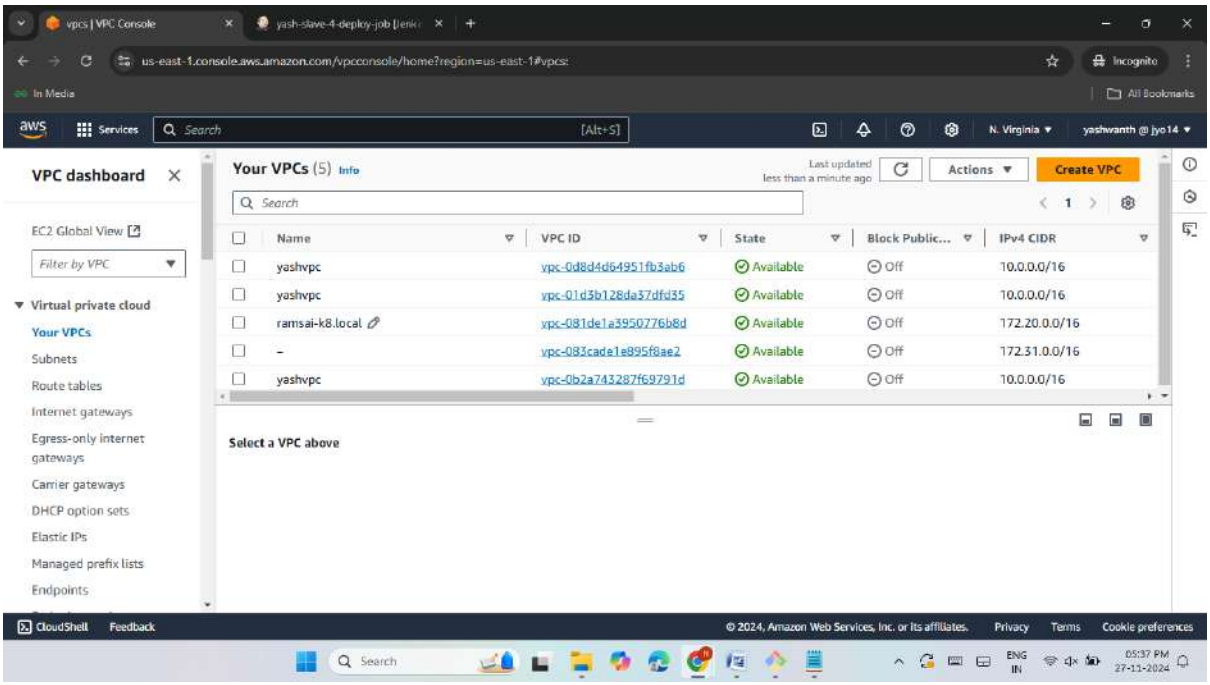
- Now click on build now.
- Here the job was success.



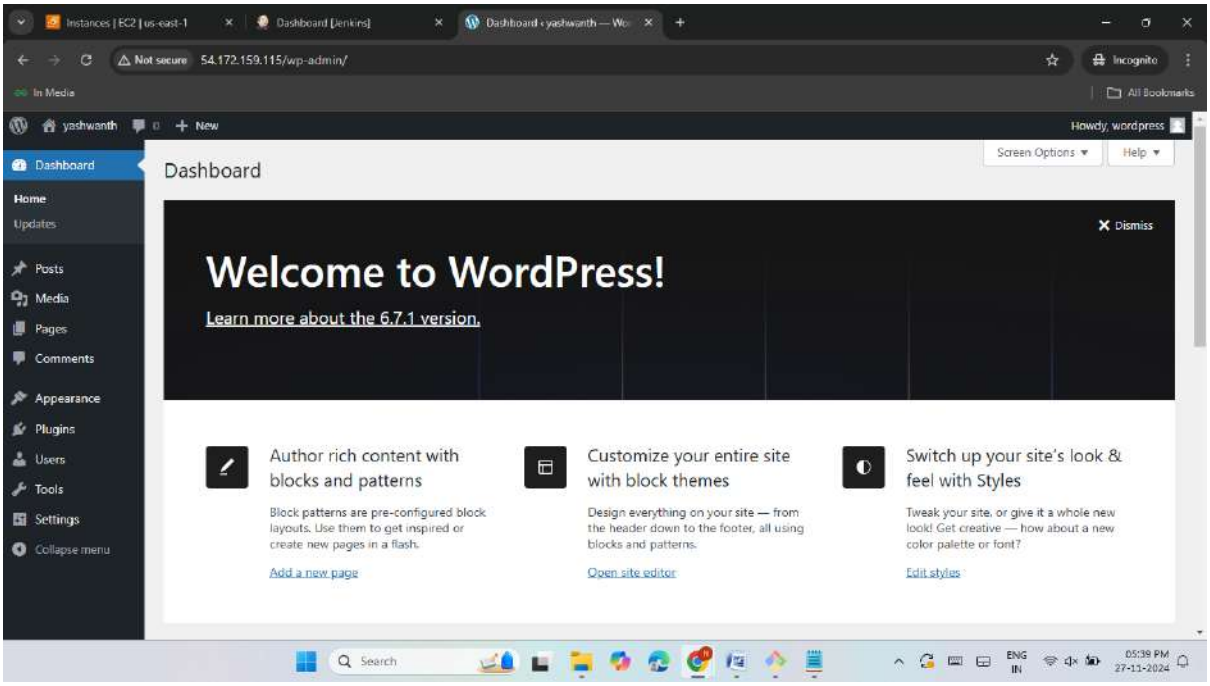
- Here the instance was launched.



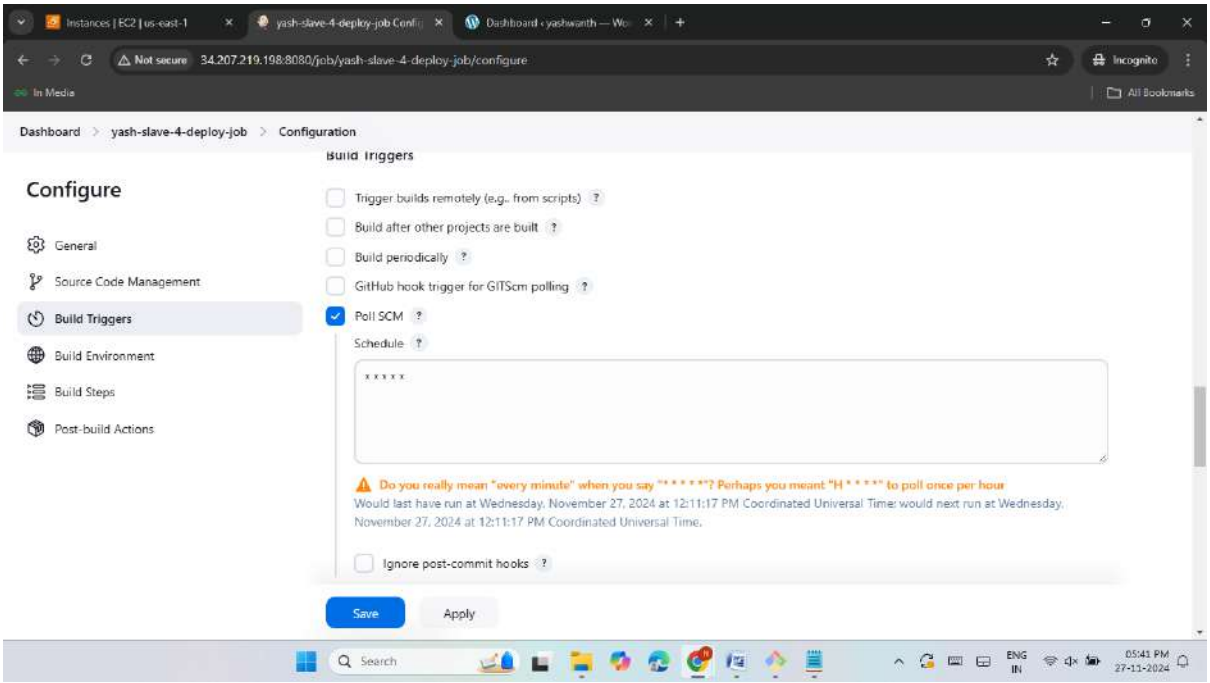
- Here the VPC was launched.



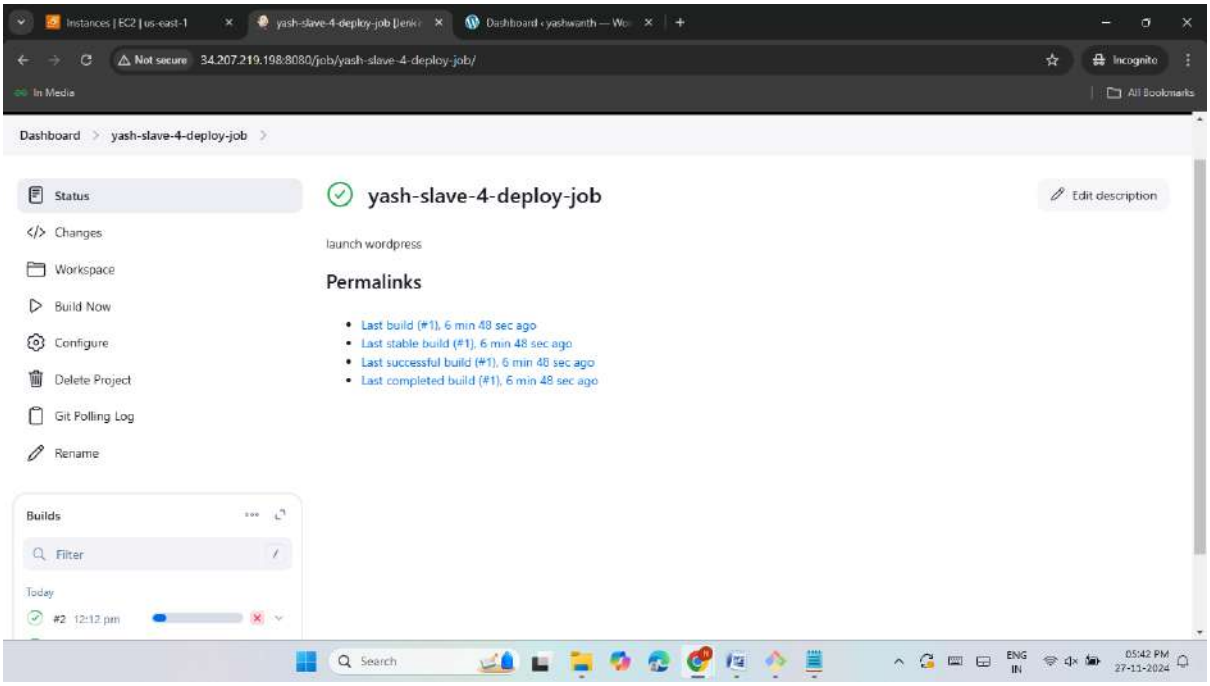
- Now copy the created instance public ip and past it on google browser.
- The wordpress was hosted successfully.



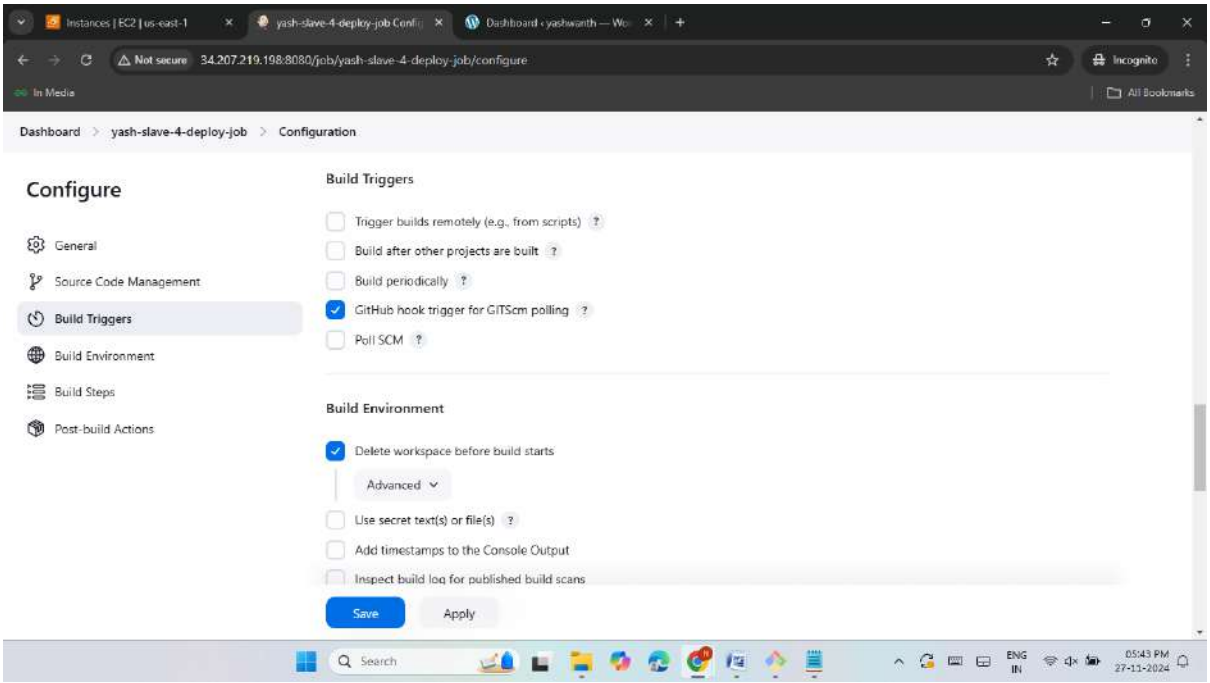
- Now add Poll SCM.



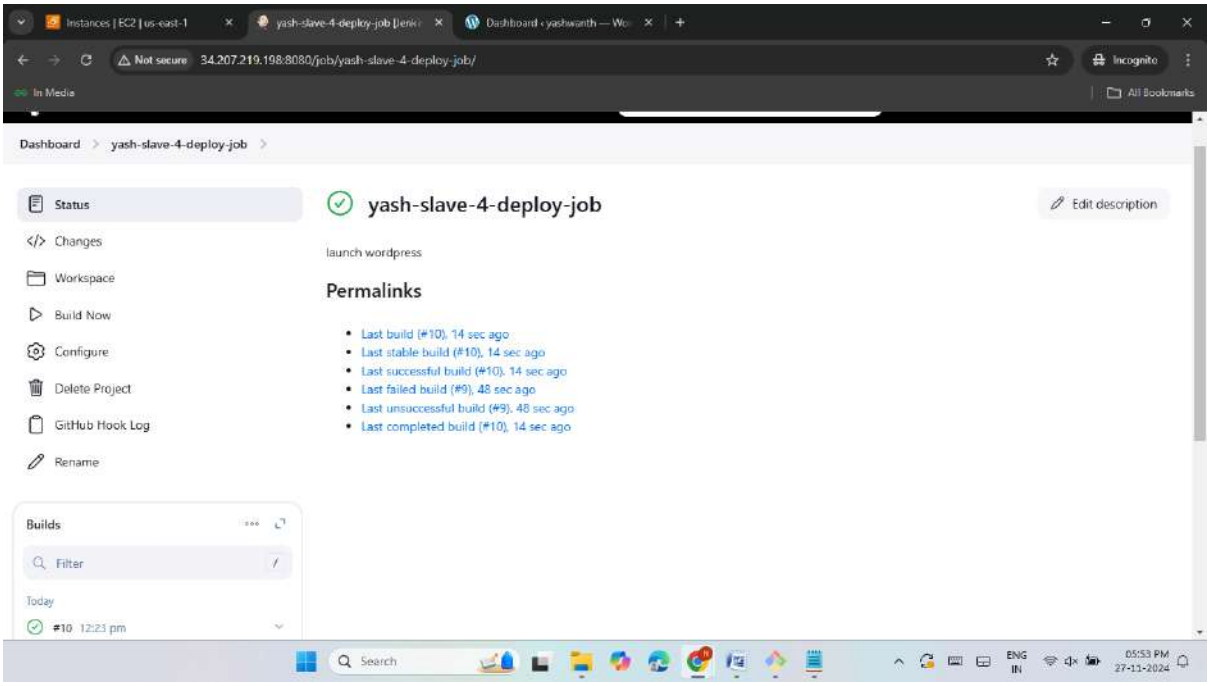
- When we change the code in github. The jobs was autometacally runs after one min because of Poll SCM.



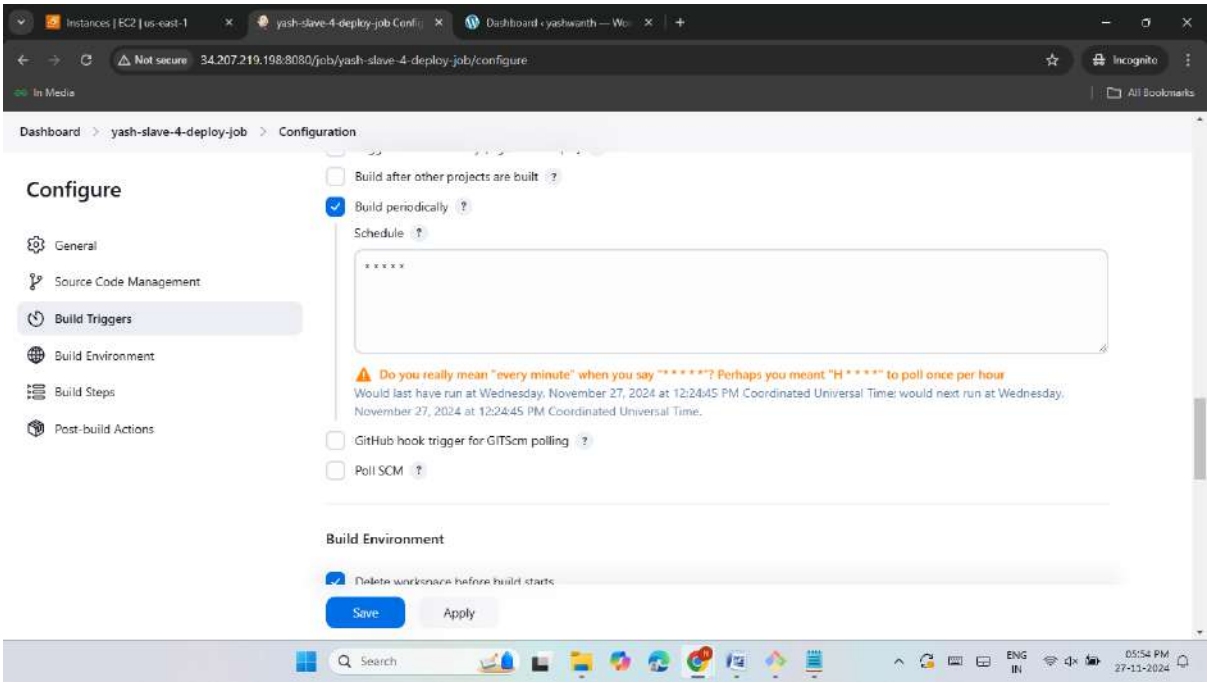
- Create webhooks in github then add into the Jenkins job.



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



- Now add build periodically in Jenkins job.



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