Project -1: Certification Project – Finance Me

Banking and Finance Domain

Problems in traditional method implementation:

Building Complex Monolithic Application is difficult.

Manual efforts to test various components/modules of the project

Incremental builds are difficult to manage, test and deploy.

It was not possible to scale up individual modules independently.

Creation of infrastructure and configure it manually is very time consuming

Continuous manual monitoring the application is quite challenging.

In the GitHub repository we have the necessary codes for our project.

https://github.com/Sonal0409/banking-finance-Project1.git

To overcome the traditional method issues, using the above code creating a DevOps CICD pipeline which includes.

GIT and Github -> Maintain source code

Terraform to create a EC2 instance

Ansible -> Setup to install required packages

Jenkins -> Create Build Pipeline and integrate pollSCM

Docker -> Use docker to build images and push to dockerhub

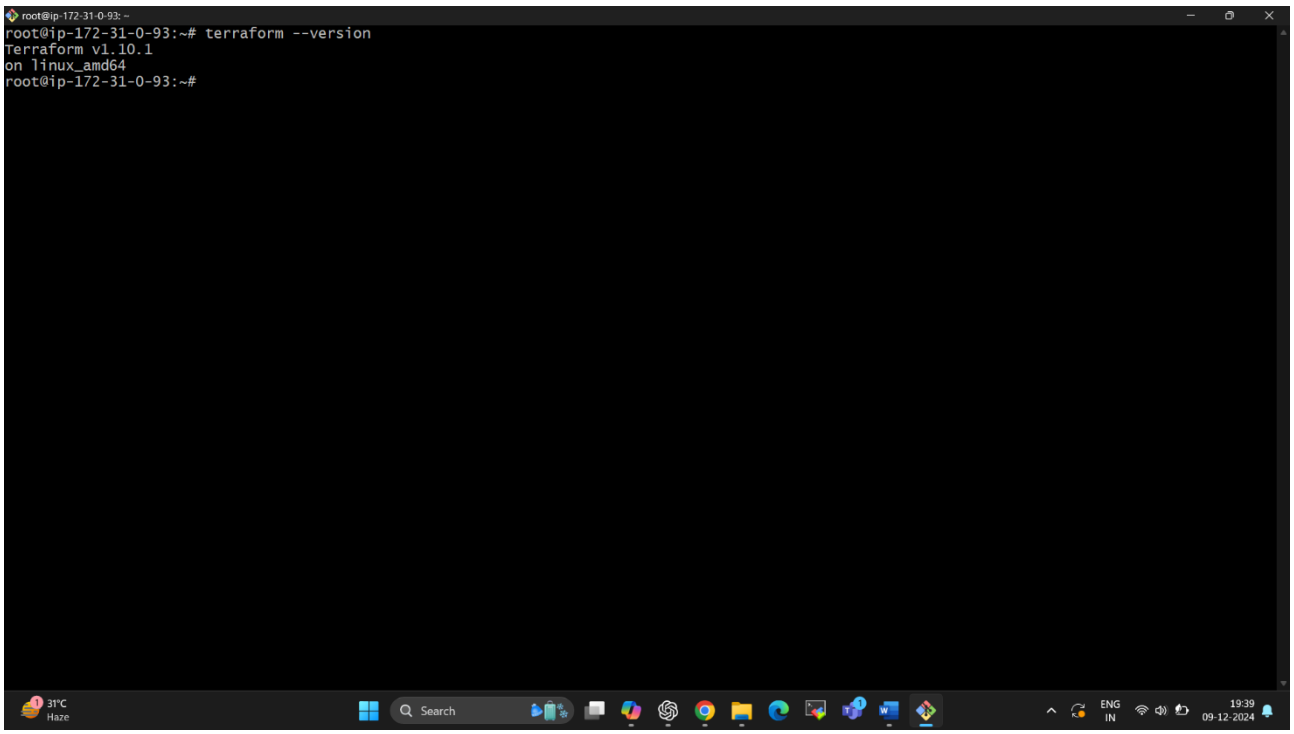
Prometheus and Grafana - monitor Jenkins pipeline

STEPS:

Create a VM and install Terraform on it

OS: ubuntu 22 → Instance type: t2.micro → Connecting to the instance →

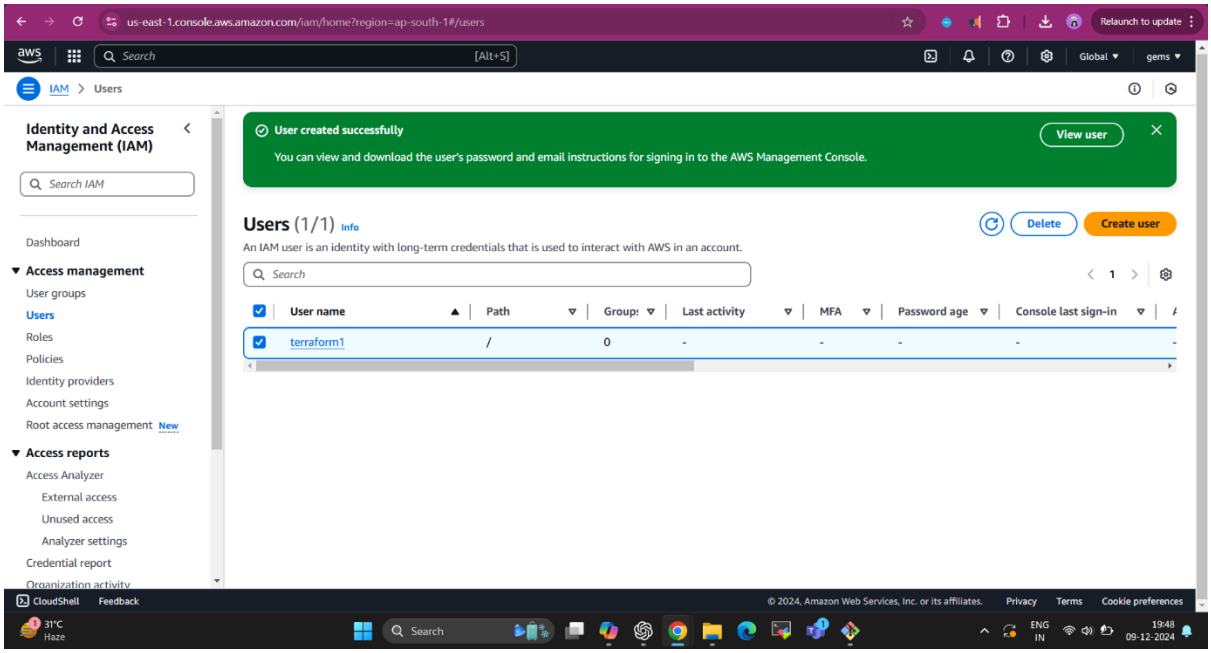
Then, install terraform on it**.**

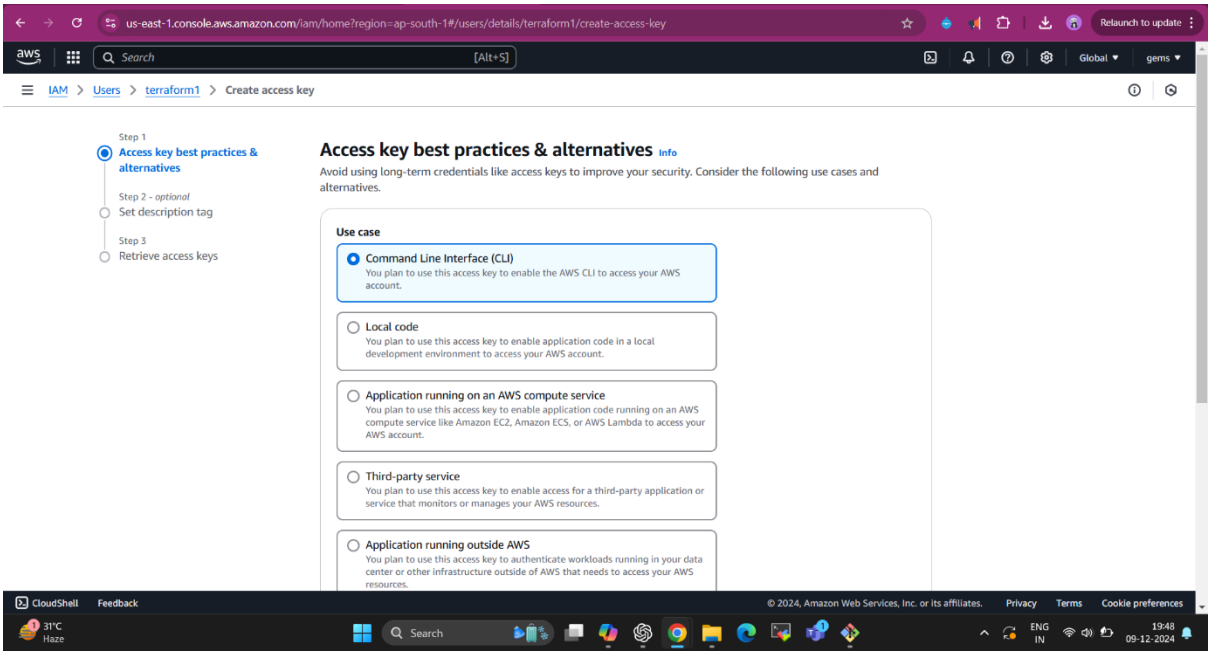


Creating Accesskey and secret key in AWS for terraform to connect:

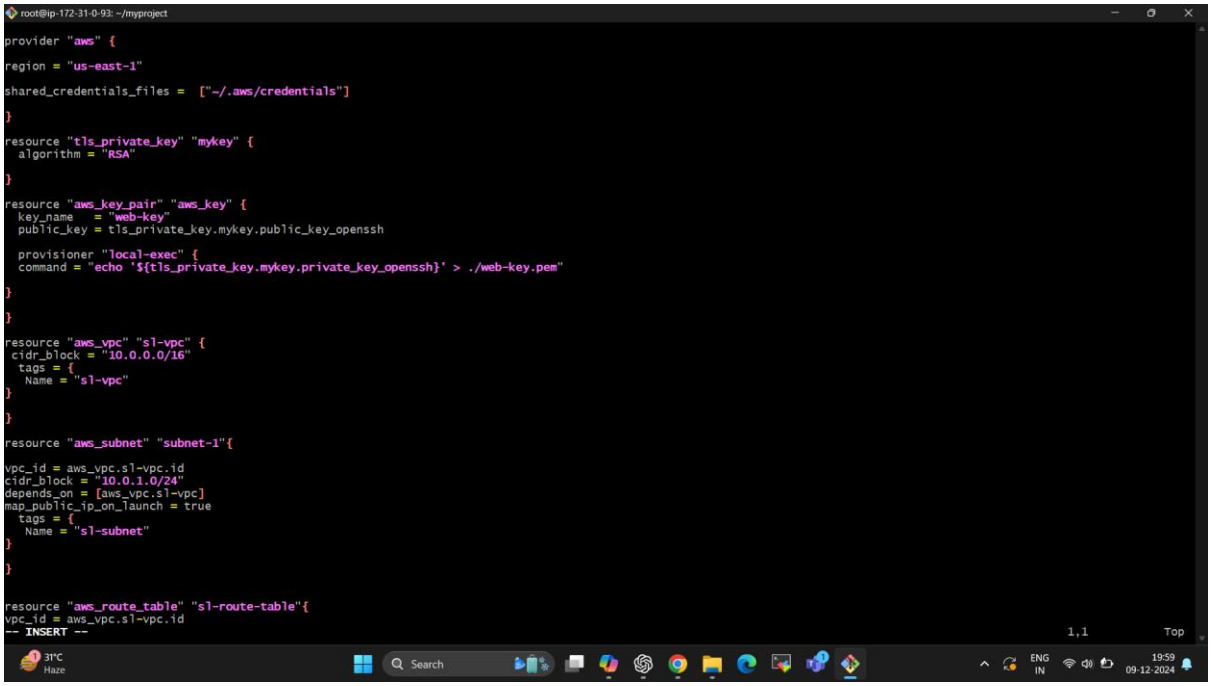
Once Accesskey and secret key is created in AWS – IAM

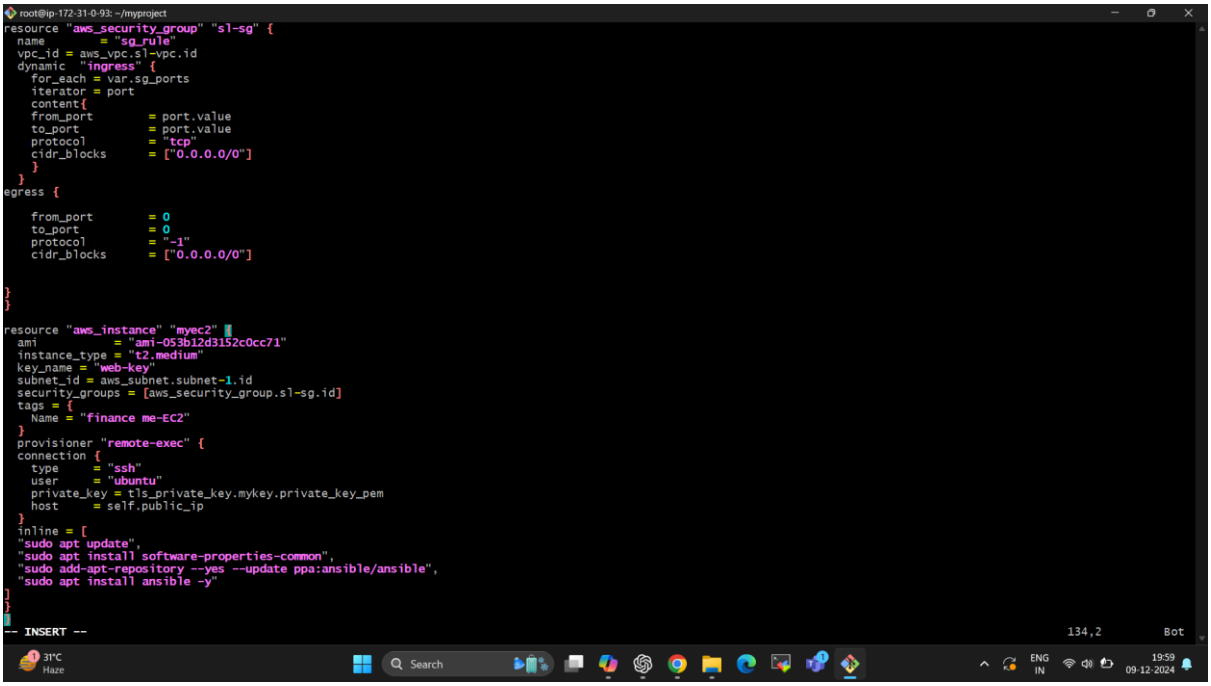
Go and configure it on the terraform VM by the command ‘aws configure’





Writing terraform configuration file to create an EC2 server and installing ansible on it.


root@ip-172-31-0-93: ~/myproject
provider "aws" {
region = "us-east-1"
shared_credentials_files = ["~/.aws/credentials"]
}
resource "tls_private_key" "mykey" {
algorithm = "RSA"
}
resource "aws_key_pair" "aws_key" {
key_name = "web-key"
public_key = tls_private_key.mykey.public_key_openssh
provisioner "local-exec" {
command = "echo '${tls_private_key.mykey.private_key_openssh}' > ./web-key.pem"
}
}
resource "aws_vpc" "sl-vpc" {
cidr_block = "10.0.0.0/16"
tags = {
Name = "sl-vpc"
}
}
resource "aws_subnet" "subnet-1"{
vpc_id = aws_vpc.sl-vpc.id
cidr_block = "10.0.1.0/24"
depends_on = [aws_vpc.sl-vpc]
map_public_ip_on_launch = true
tags = {
Name = "sl-subnet"
}
}
resource "aws_route_table" "sl-route-table"{
vpc_id = aws_vpc.sl-vpc.id
-- INSERT --
1,1
Top
31°C
Haze
Search
ENG
IN
19:59
09-12-2024


root@ip-172-31-0-93: ~/myproject
resource "aws_security_group" "sl-sg" {
name = "sg_rule"
vpc_id = aws_vpc.sl-vpc.id
dynamic "ingress" {
for_each = var.sg_ports
iterator = port
content{
from_port = port.value
to_port = port.value
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
resource "aws_instance" "myec2" {
ami = "ami-053b12d3152c0cc71"
instance_type = "t2.medium"
key_name = "web-key"
subnet_id = aws_subnet.subnet-1.id
security_groups = [aws_security_group.sl-sg.id]
tags = {
Name = "finance me-EC2"
}
provisioner "remote-exec" {
connection {
type = "ssh"
user = "ubuntu"
private_key = tls_private_key.mykey.private_key_pem
host = self.public_ip
}
inline = [
"sudo apt update",
"sudo apt install software-properties-common",
"sudo add-apt-repository --yes --update ppa:ansible/ansible",
"sudo apt install ansible -y"
]
}
-- INSERT --
134,2
Bot
31°C
Haze
Search
ENG
IN
19:59
09-12-2024

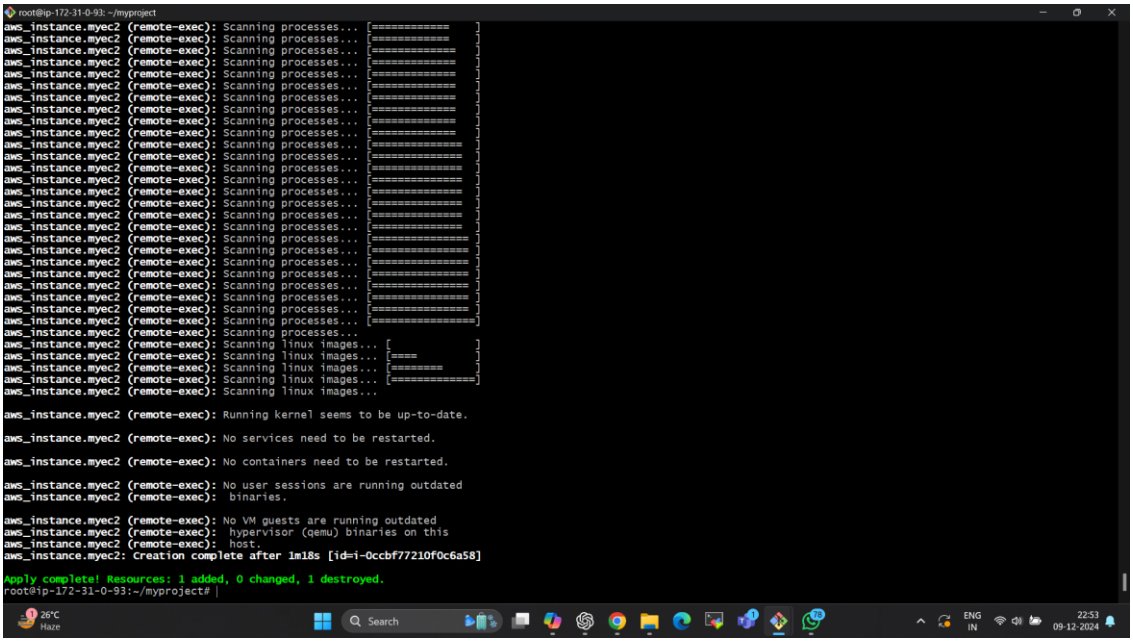
```
root@ip-172-31-0-93:~# mkdir myproject
root@ip-172-31-0-93:~# cd myproject
root@ip-172-31-0-93:~/myproject# vim terra1.tf
root@ip-172-31-0-93:~/myproject# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/tls...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/tls v4.0.6...
- Installed hashicorp/tls v4.0.6 (signed by HashiCorp)
- Installing hashicorp/aws v5.80.0...
- Installed hashicorp/aws v5.80.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@ip-172-31-0-93:~/myproject#
```

Terraform works at the directory level

Run terraform init command in the terraform directory to initialize

and connect to the aws instance.

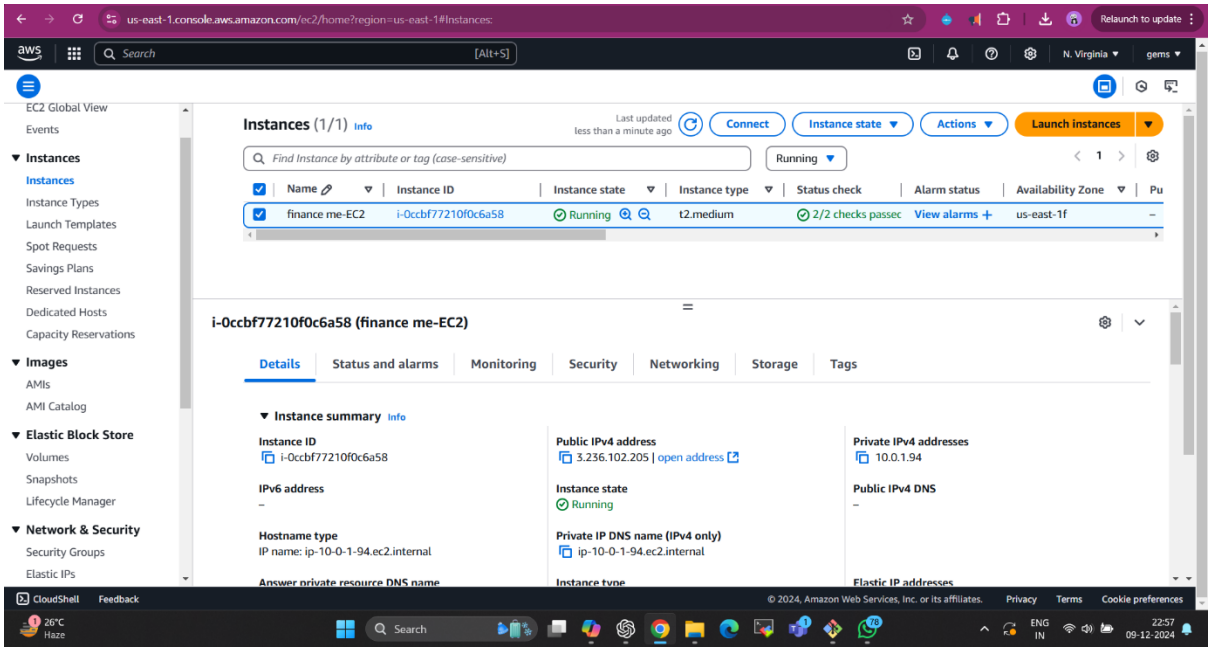
```
root@ip-172-31-0-93: ~/myproject
aws_instance.myec2 (remote-exec): Reading package lists... Done
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... Done
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... Done
aws_instance.myec2 (remote-exec): 40 packages can be upgraded. Run 'apt list --upgradable' to see them.
aws_instance.myec2 (remote-exec): Reading package lists... 0%
aws_instance.myec2 (remote-exec): Reading package lists... 100%
aws_instance.myec2 (remote-exec): Reading package lists... Done
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 0%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... 50%
aws_instance.myec2 (remote-exec): Building dependency tree... Done
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... 0%
aws_instance.myec2 (remote-exec): Reading state information... Done
aws_instance.myec2 (remote-exec): The following additional packages will be installed:
aws_instance.myec2 (remote-exec):   python3-software-properties
aws_instance.myec2 (remote-exec): The following packages will be upgraded:
aws_instance.myec2 (remote-exec):   python3-software-properties
aws_instance.myec2 (remote-exec):   software-properties-common
aws_instance.myec2 (remote-exec): 2 upgraded, 0 newly installed, 0 to remove and 38 not upgraded.
aws_instance.myec2 (remote-exec): Need to get 44.1 kB of archives.
aws_instance.myec2 (remote-exec): After this operation, 0 B of additional disk space will be used.
aws_instance.myec2 (remote-exec): Do you want to continue? [Y/n]
/
aws_instance.myec2: Still creating... [50s elapsed]
aws_instance.myec2: Still creating... [1m0s elapsed]
aws_instance.myec2: Still creating... [1m10s elapsed]
aws_instance.myec2: Still creating... [1m20s elapsed]
aws_instance.myec2: Still creating... [1m30s elapsed]
aws_instance.myec2: Still creating... [1m40s elapsed]
aws_instance.myec2: Still creating... [1m50s elapsed]
aws_instance.myec2: Still creating... [2m0s elapsed]
aws_instance.myec2: Still creating... [2m10s elapsed]
aws_instance.myec2: Still creating... [2m20s elapsed]
aws_instance.myec2: Still creating... [2m30s elapsed]
aws_instance.myec2: Still creating... [2m40s elapsed]
aws_instance.myec2: Still creating... [2m50s elapsed]
aws_instance.myec2: Still creating... [3m0s elapsed]
aws_instance.myec2: Still creating... [3m10s elapsed]
aws_instance.myec2: Still creating... [3m20s elapsed]
```

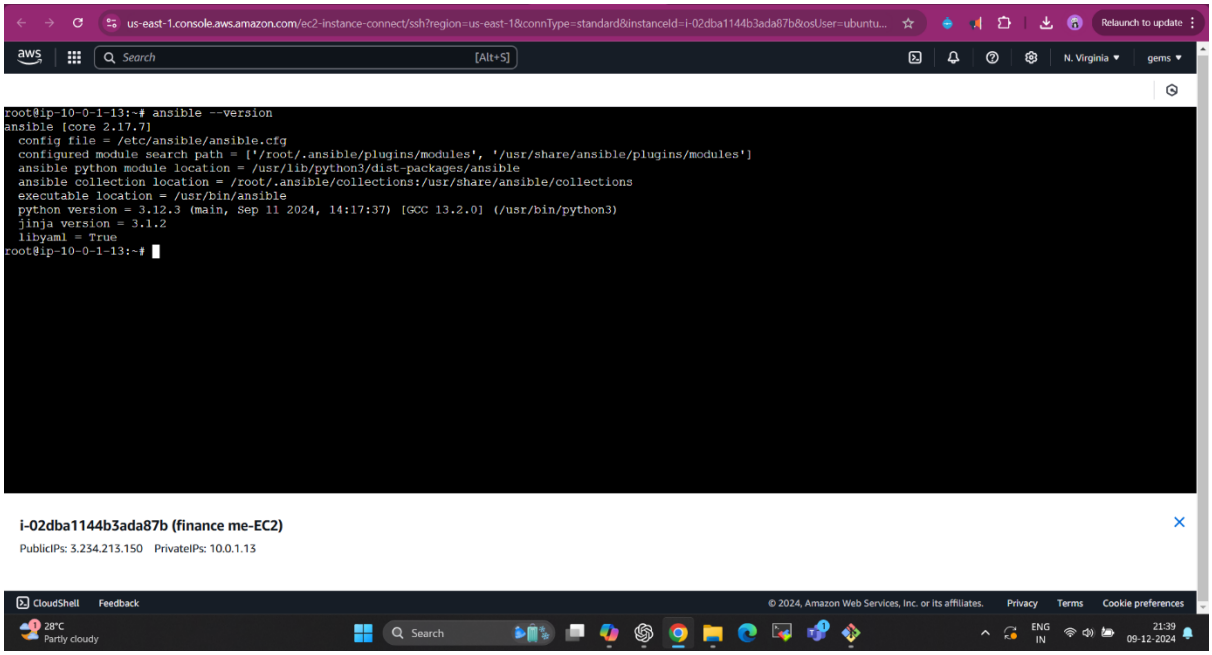


Install all the tools for CICD

Go the us-east-1 region:

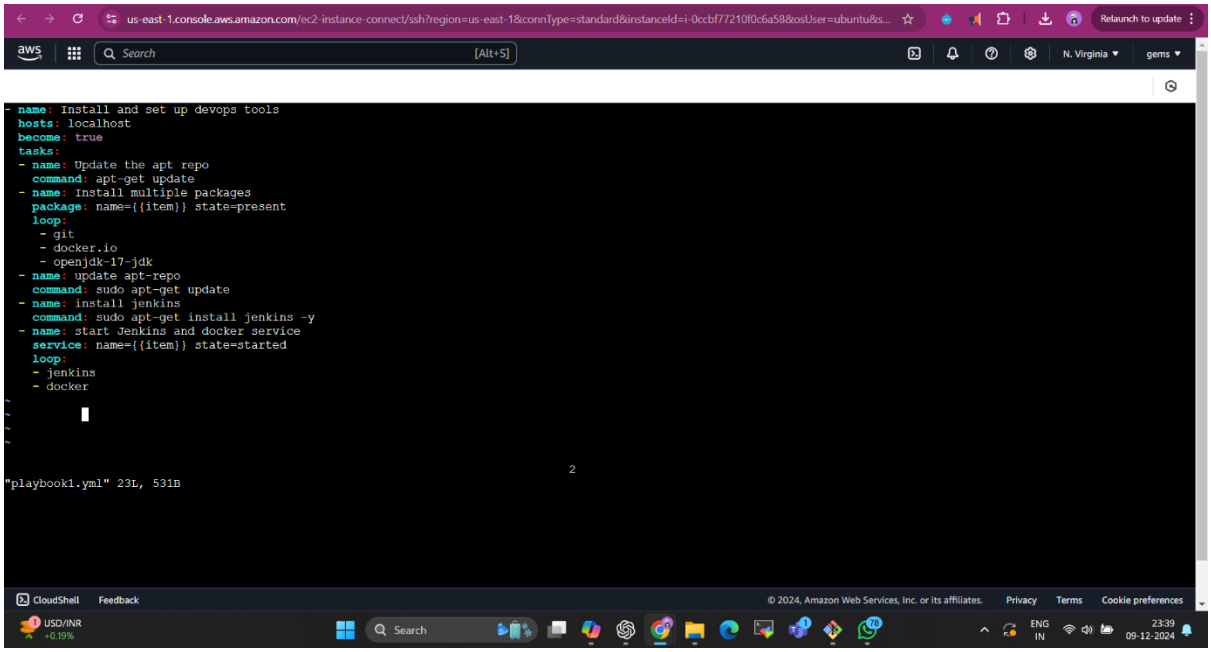
connect to the created t2. medium EC2 instance which has Ansible installed on it → Connect using EC2 instance connect → Add Jenkins key to the server → Make Jenkins apt repo.

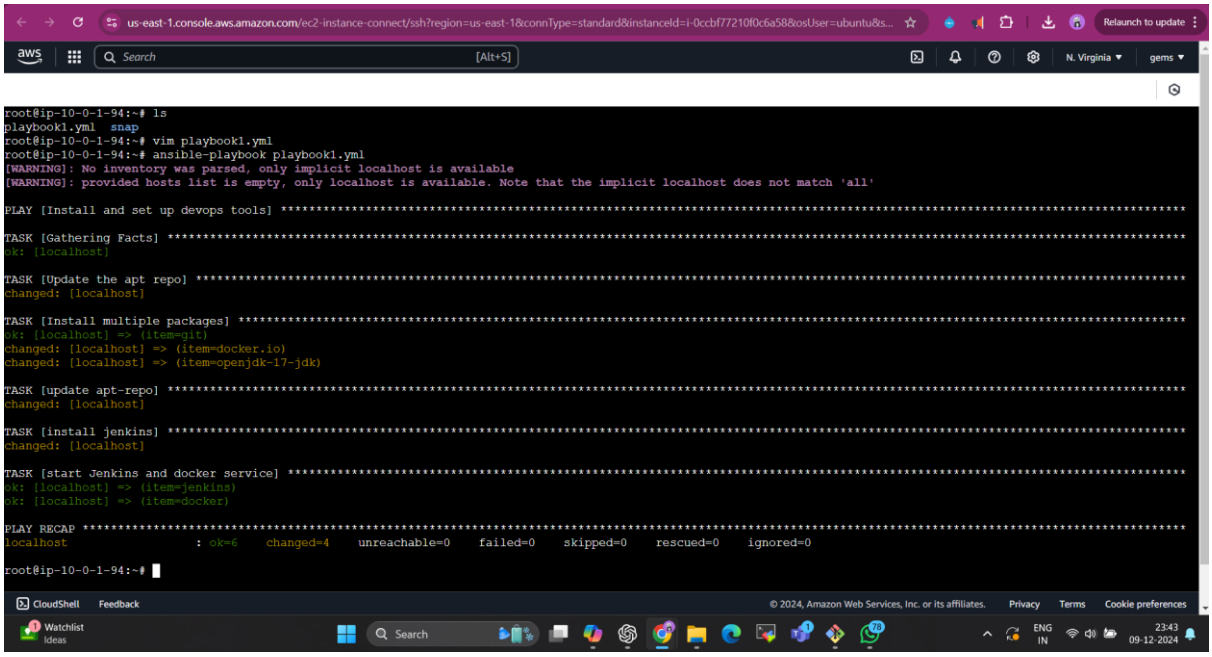




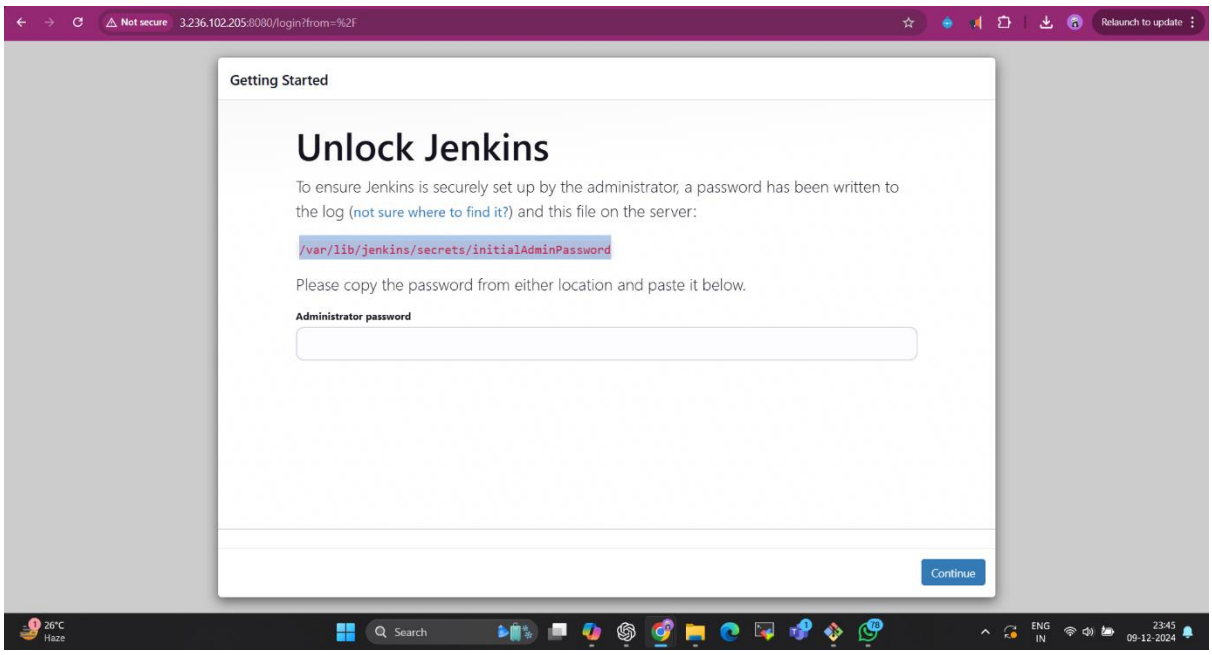
Creating an ansible playbook to setup required packages [openjdk,Jenkins,docker] in the local hosts server → Save the file and run it.

Starting the Jenkins and docker service.





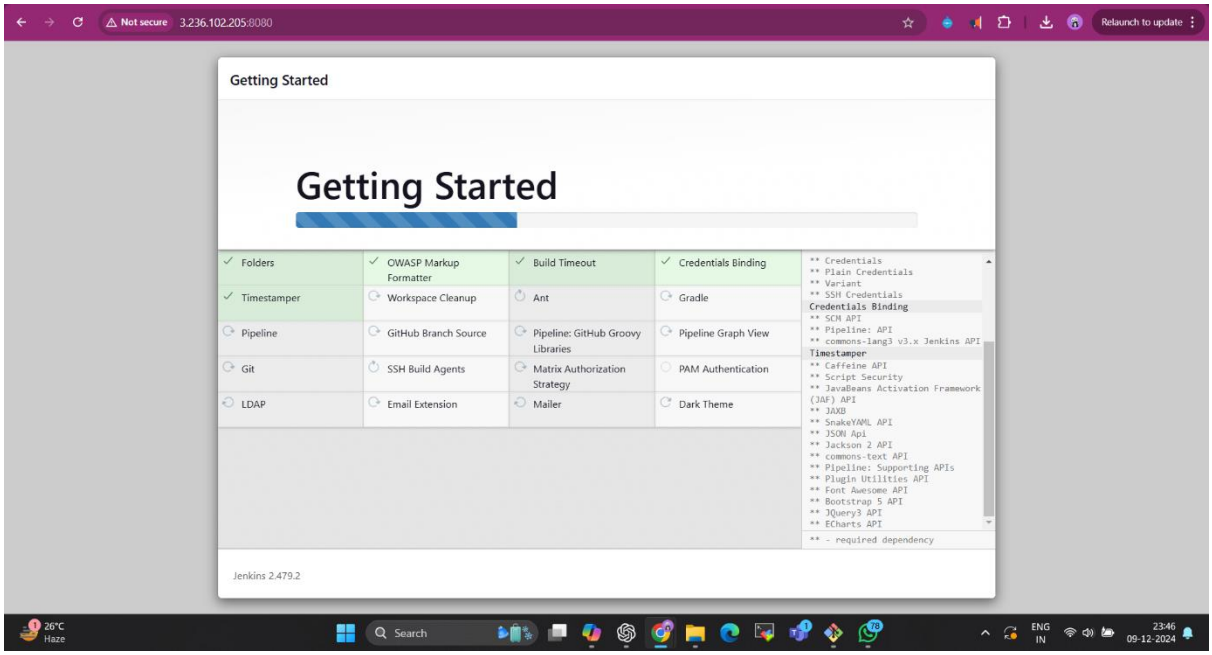
Accessing Jenkins server at the port no:8080



CICD pipeline:

Setup Jenkins dashboard and login to the Jenkins Dashboard.

Setup maven and ansible in tools section of Jenkins.

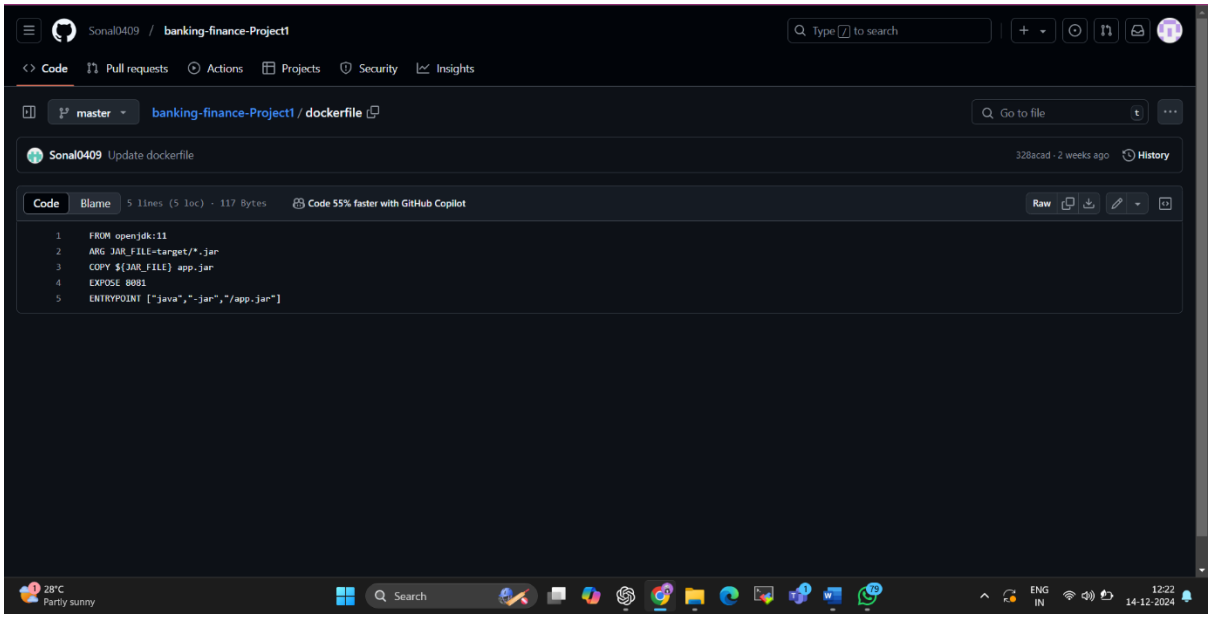


Creating a docker file to containerize the necessary packages to run the application and save it in the git hub repo.

Providing permission for Jenkins to run the docker commands.

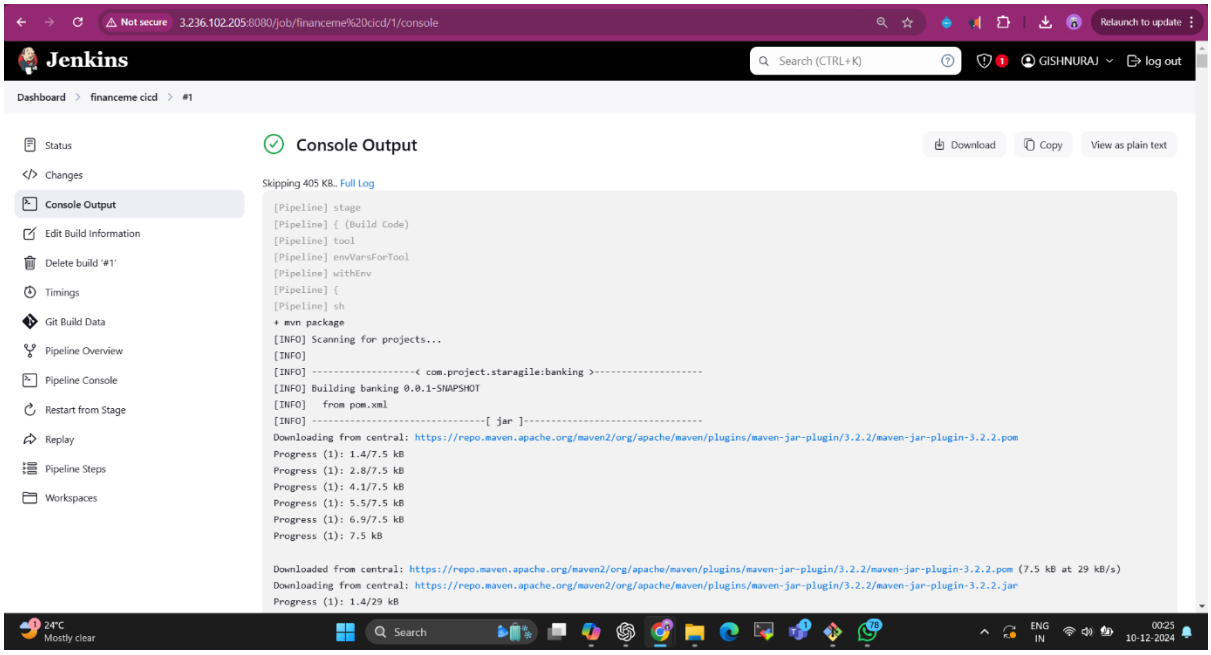
This command will allow Jenkins to run docker commands

chmod -R 777 /var/run/docker.sock



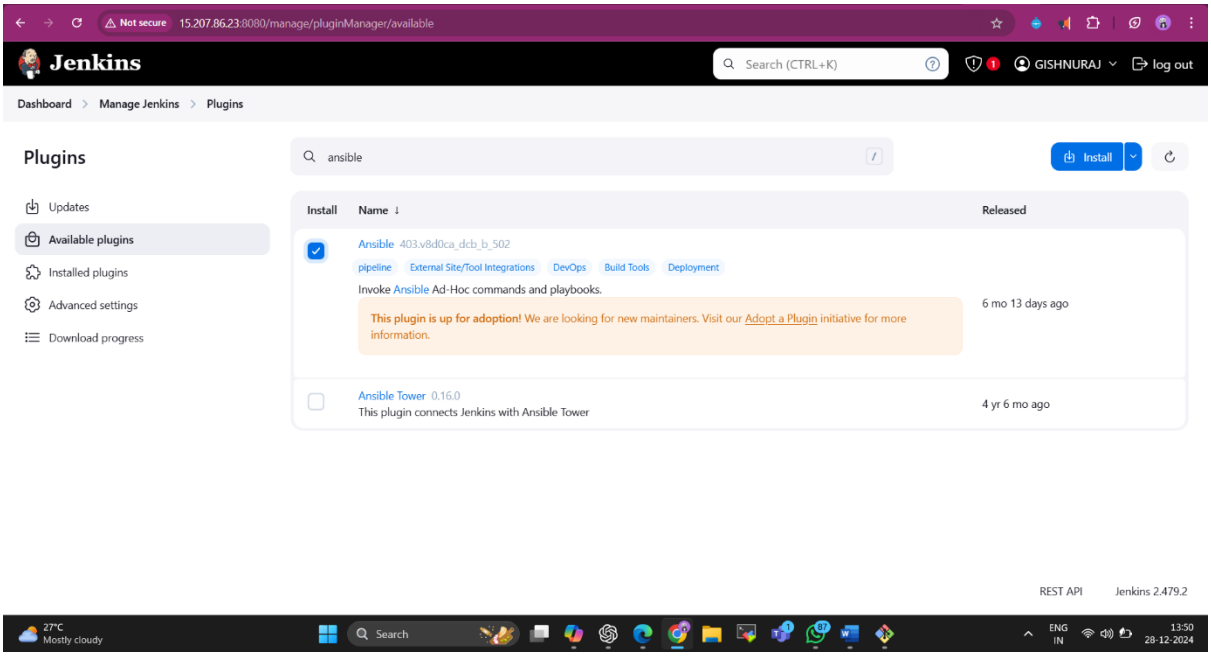
Creating a Pipeline job to

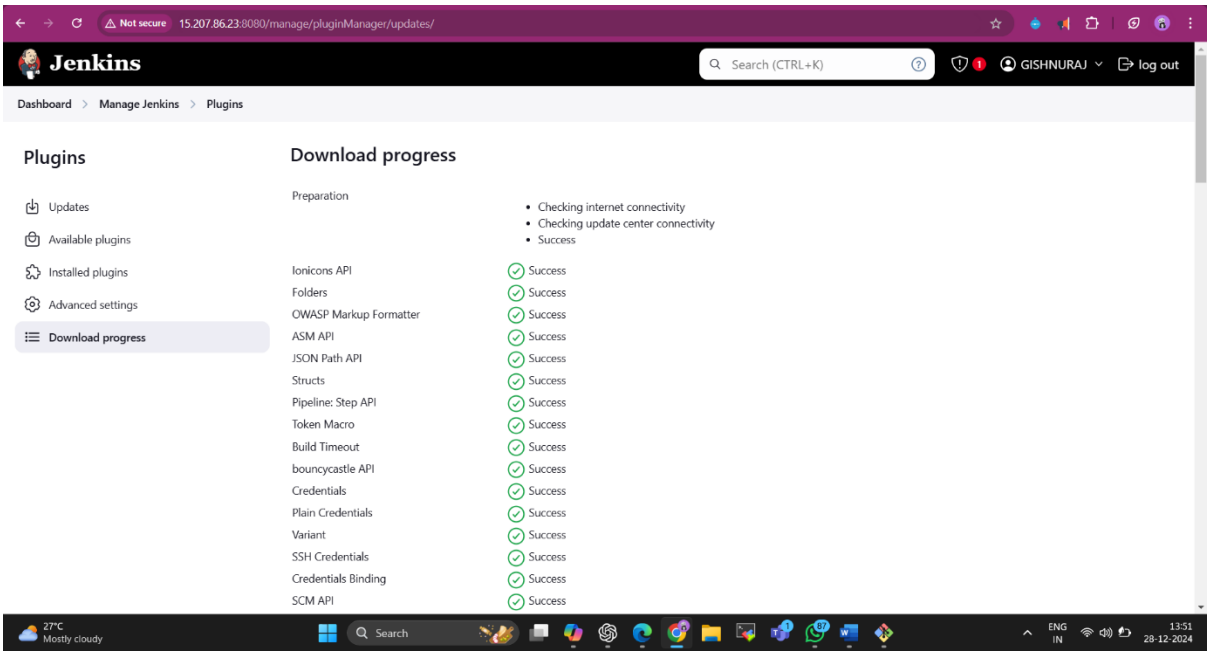
fetch code form github → test and build the code using maven commands → build docker image → creating an infrastructure (test server) using terraform → managing configurations using ansible playbook to deploy the containerized application.

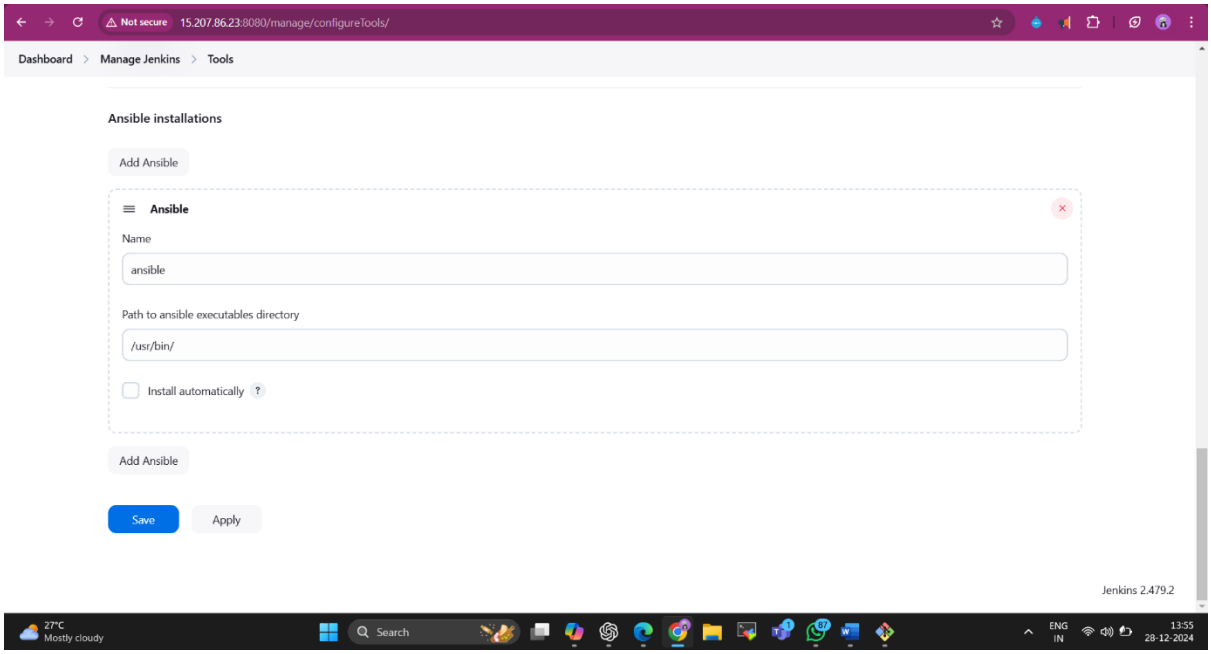


This cicd pipeline includes the various following actions:

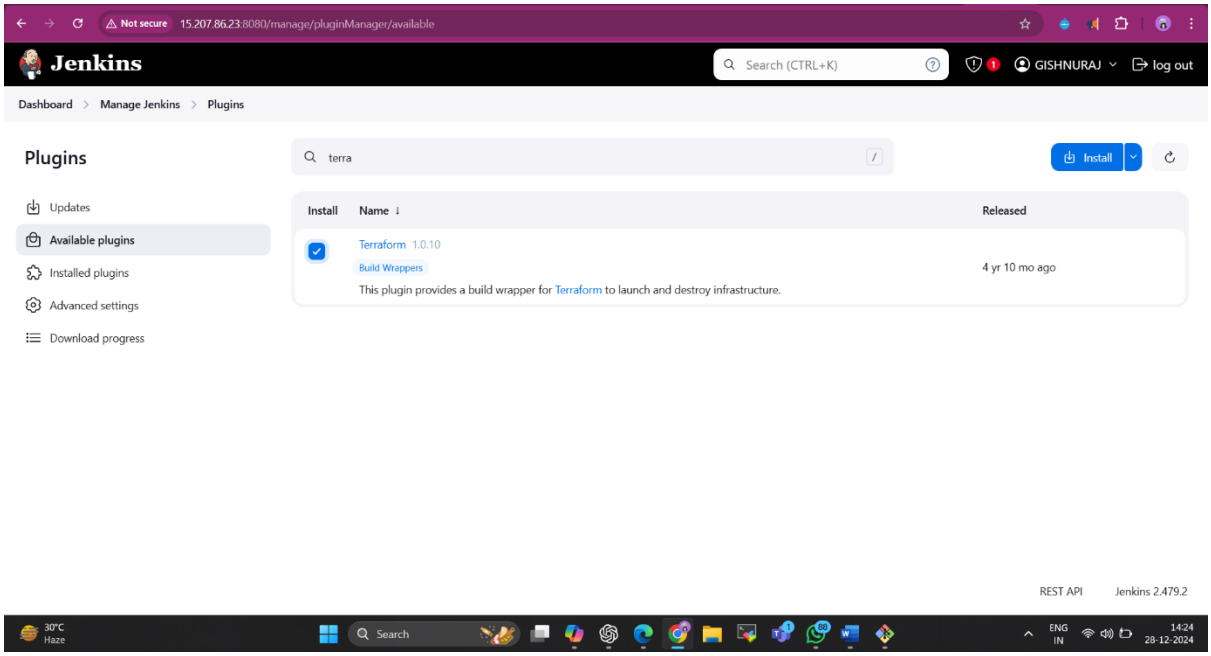
Installing necessary plugins (ansible,terraform)



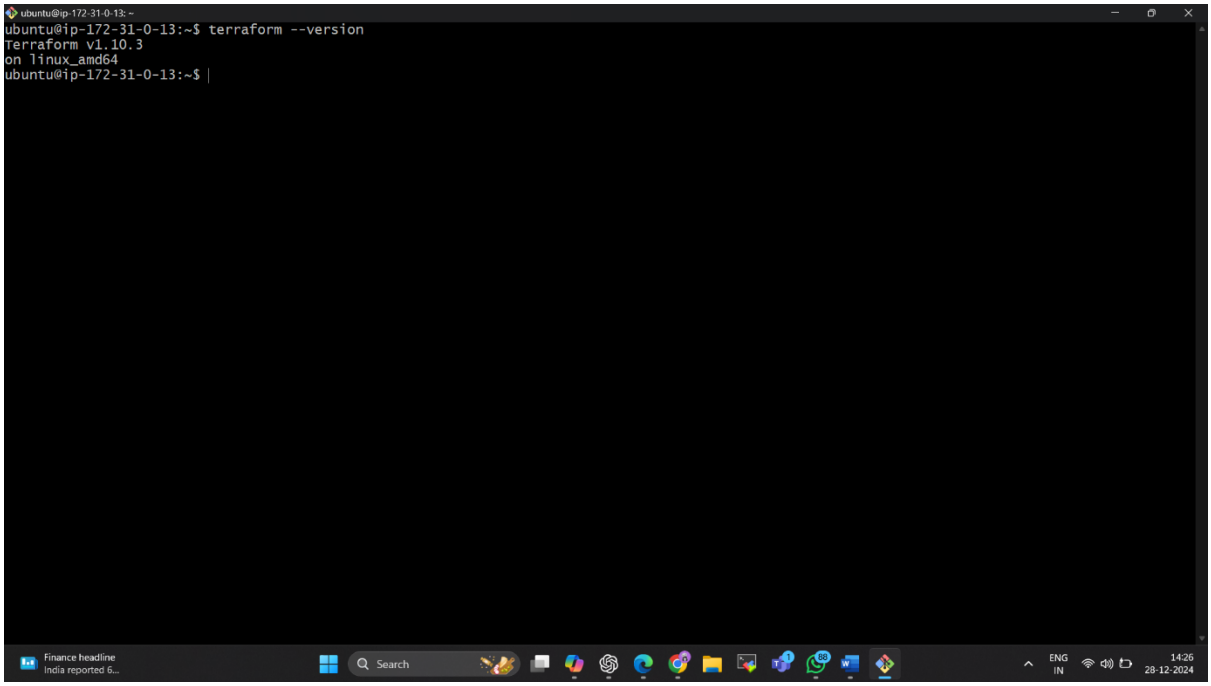




In the tools section configuring the ansible and the maven for the cicd pipeline.



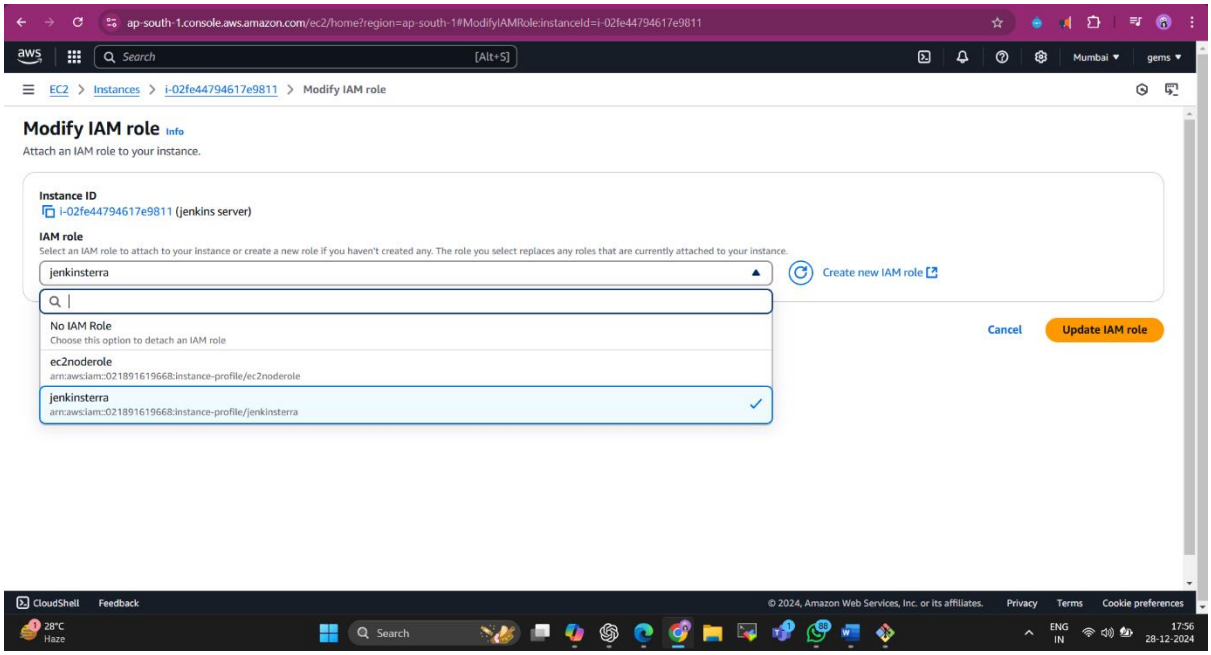
Installing terraform ,git, ansible, docker in the server.

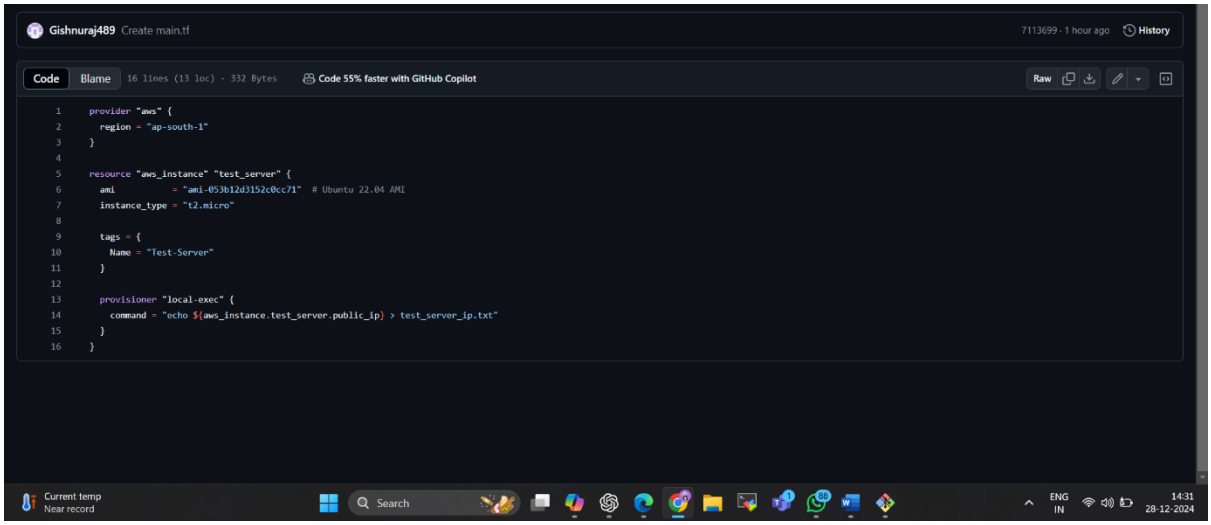


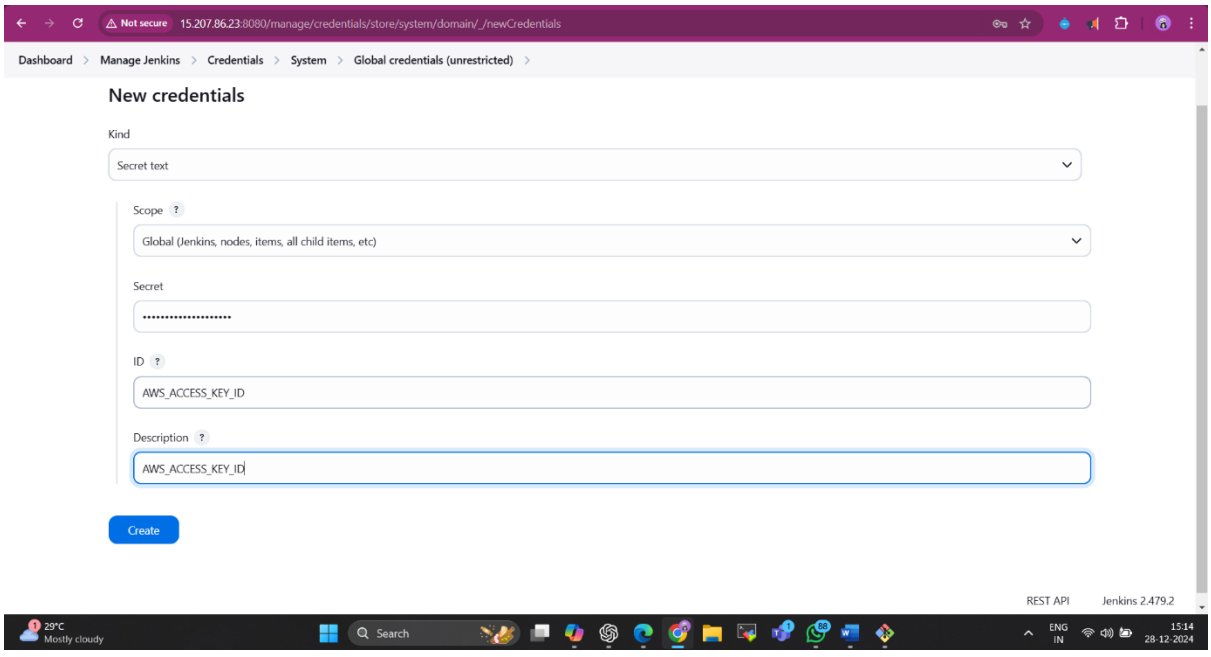
Writing a terraform conguration file to create the test server infrastructure .

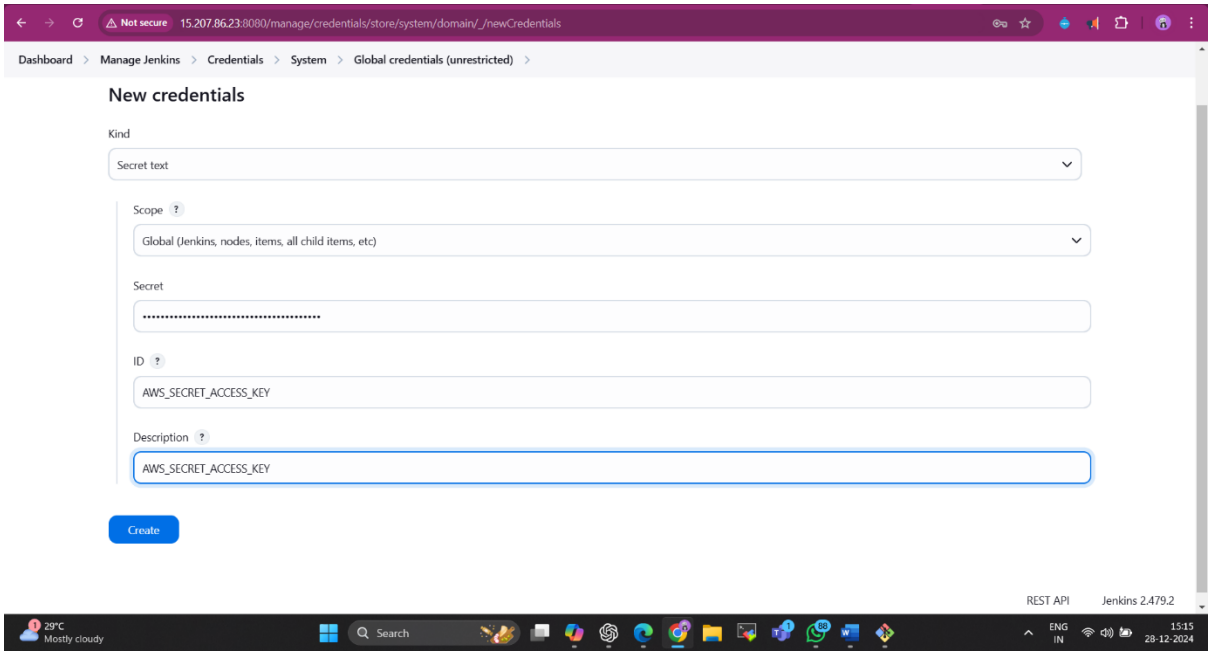
In jenkins server adding the AWS credentials to make changes in the aws console.

Also adding the IAM role to provide access to the aws account.




Gishnuraj489 Create main.tf
7113699 · 1 hour ago
History
Code
Blame
16 lines (13 loc) · 332 Bytes
Code 55% faster with GitHub Copilot
Raw
provider "aws" {
region = "ap-south-1"
}
resource "aws_instance" "test_server" {
ami = "ami-053b12d3152c0cc71" # Ubuntu 22.04 AMI
instance_type = "t2.micro"
tags = {
Name = "Test-Server"
}
provisioner "local-exec" {
command = "echo ${aws_instance.test_server.public_ip} > test_server_ip.txt"
}
}
Current temp
Near record
Search
ENG
IN
14:31
28-12-2024

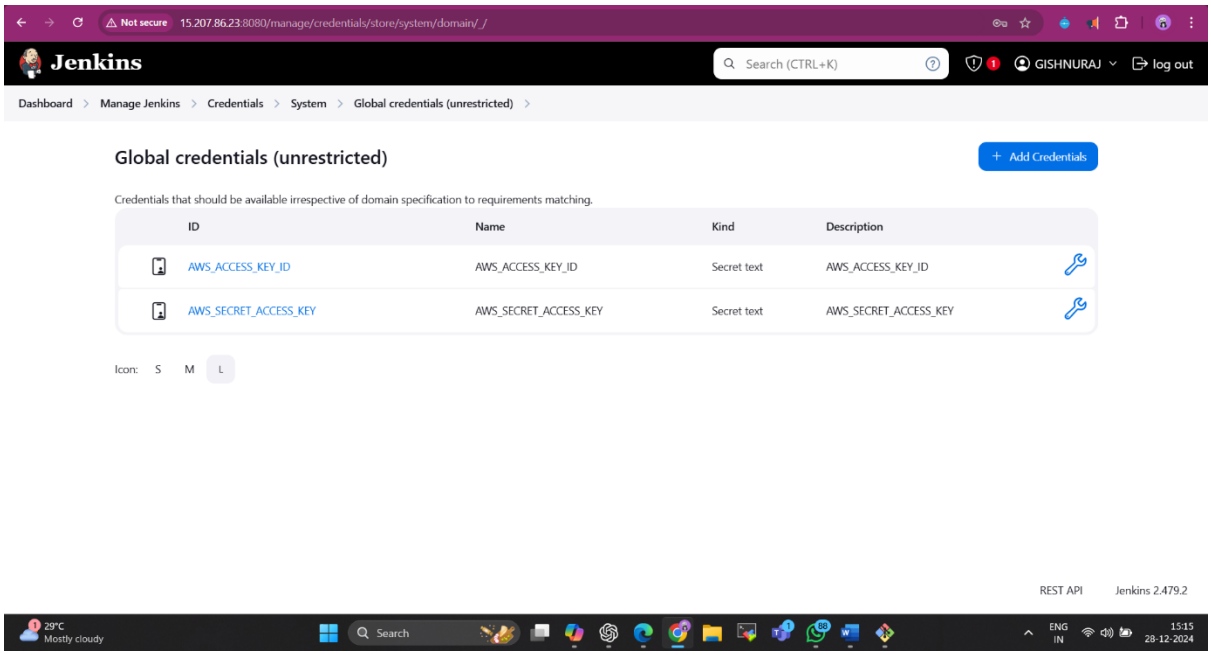

Not secure
15.207.86.23:8080/manage/credentials/store/system/domain/_/newCredentials
Dashboard
Manage Jenkins
Credentials
System
Global credentials (unrestricted)
New credentials
Kind
Secret text
Scope
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID
AWS_ACCESS_KEY_ID
Description
AWS_ACCESS_KEY_ID
Create
REST API
Jenkins 2.479.2
29°C
Mostly cloudy
Search
ENG
IN
15:14
28-12-2024

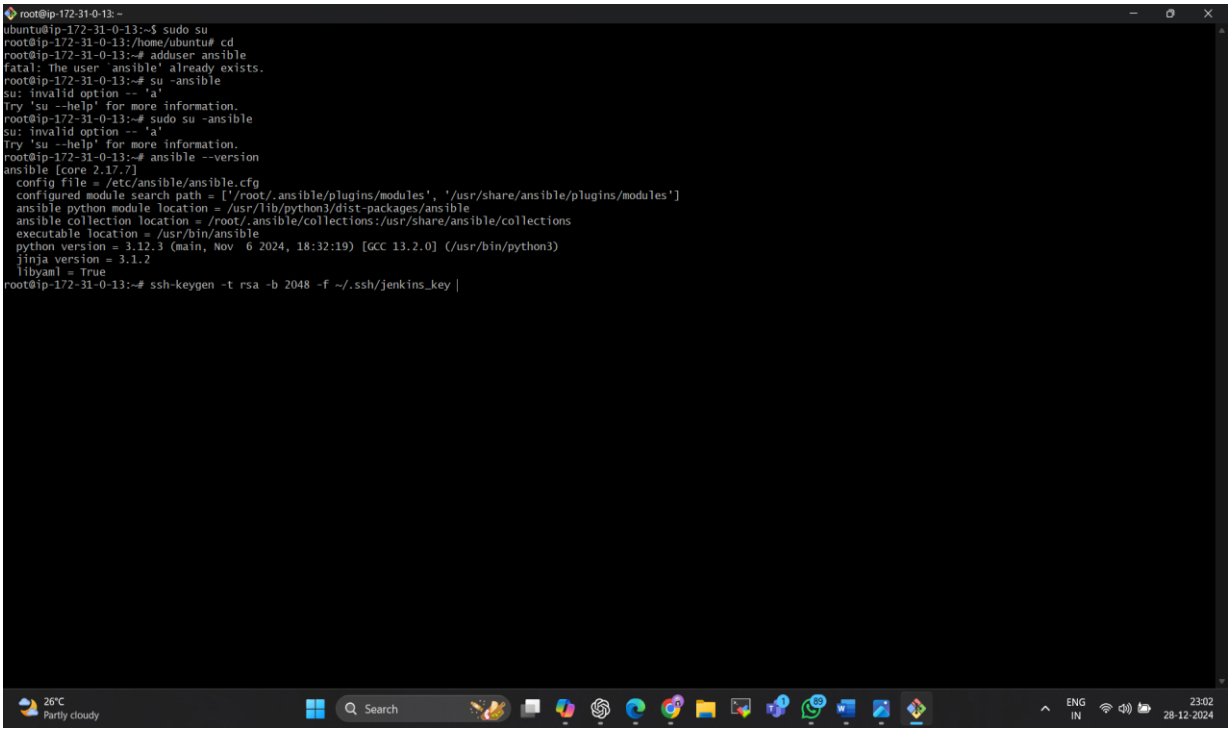


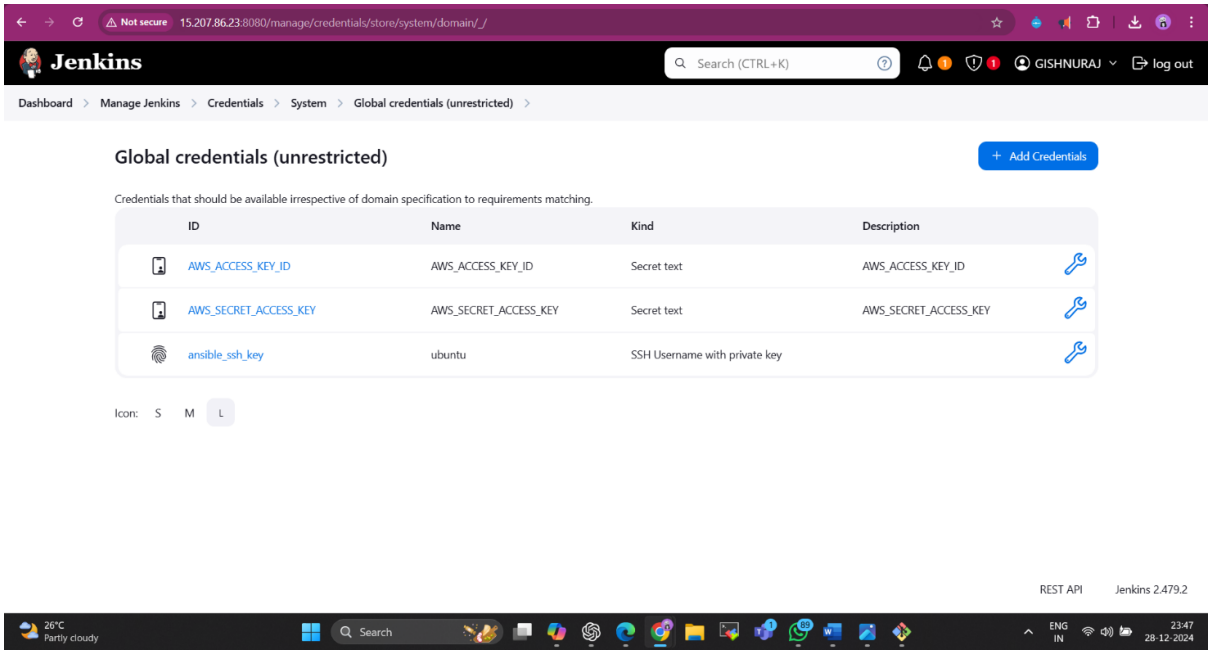
Writing the ansible playbook to make configuration changes in the test server (deploying an application).

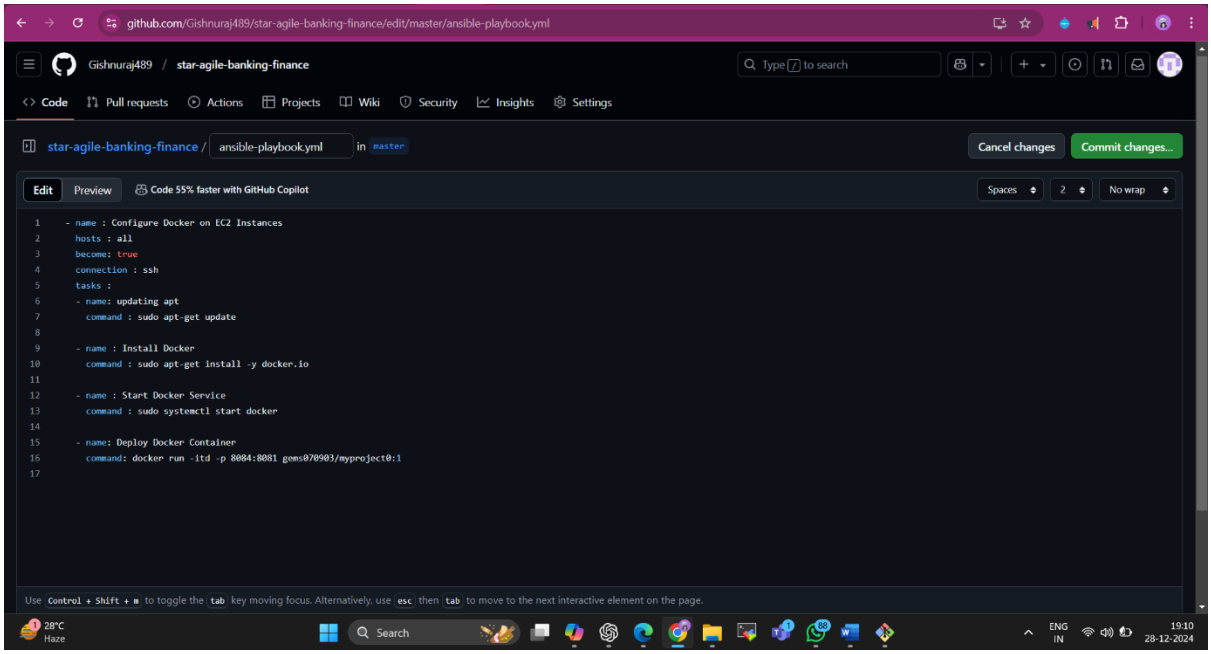
Write and save the playbook in the github repository (playbook , inventory).

Also connecting to the test server and make configuration changes can be through ssh keys. These keys will be stored as a credential in the Jenkins server to make use of it.

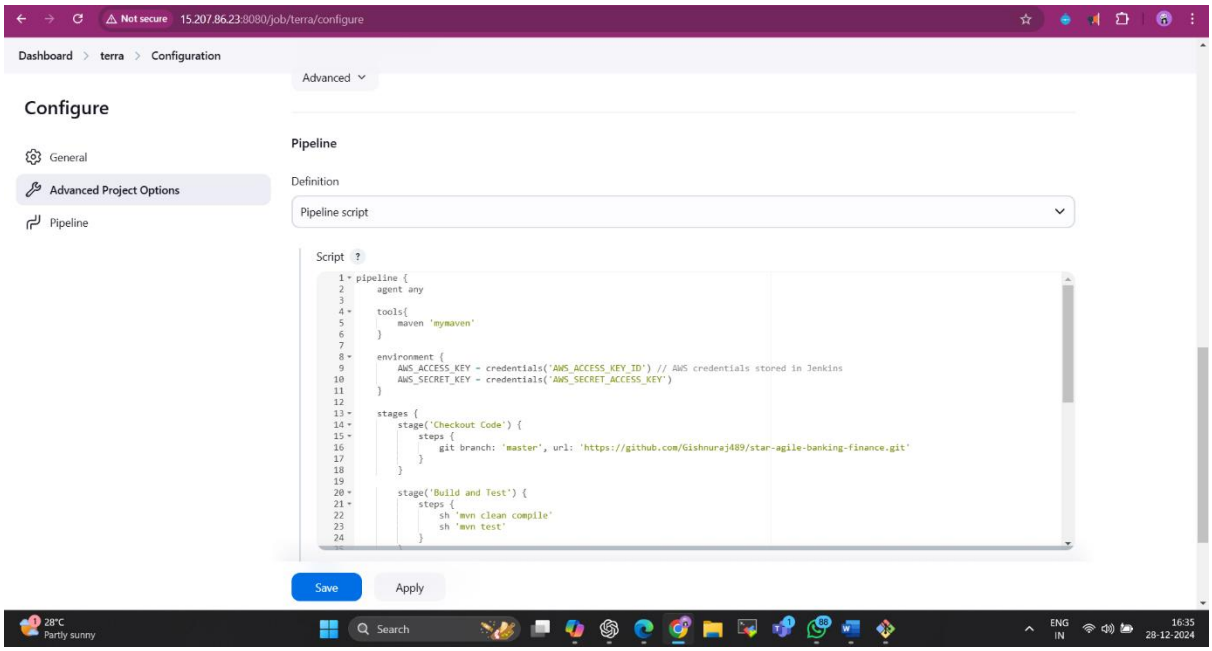

Not secure 15.207.86.23:8080/manage/credentials/store/system/domain/_/
Jenkins
Search (CTRL+K)
GISHNURAJ
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
AWS_ACCESS_KEY_ID
AWS_ACCESS_KEY_ID
Secret text
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SECRET_ACCESS_KEY
Secret text
AWS_SECRET_ACCESS_KEY
Icon: S M L
REST API
Jenkins 2.479.2
29°C
Mostly cloudy
Search
ENG
IN
15:15
28-12-2024

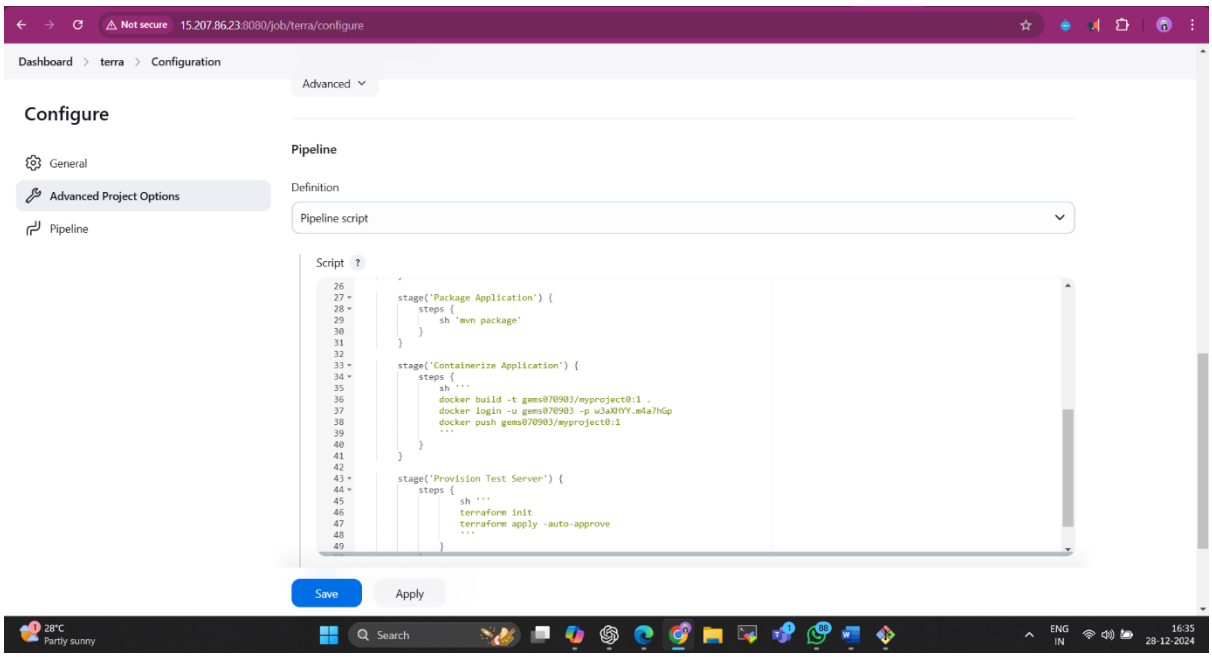

root@ip-172-31-0-13: ~
ubuntu@ip-172-31-0-13:~$ sudo su
root@ip-172-31-0-13:/home/ubuntu# cd
root@ip-172-31-0-13:~# adduser ansible
fatal: The user `ansible' already exists.
root@ip-172-31-0-13:~# su -ansible
su: invalid option -- 'a'
Try 'su --help' for more information.
root@ip-172-31-0-13:~# sudo su -ansible
su: invalid option -- 'a'
Try 'su --help' for more information.
root@ip-172-31-0-13:~# ansible --version
ansible [core 2.17.7]
config file = /etc/ansible/ansible.cfg
configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python3/dist-packages/ansible
ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
executable location = /usr/bin/ansible
python version = 3.12.3 (main, Nov 6 2024, 18:32:19) [GCC 13.2.0] (/usr/bin/python3)
jinja version = 3.1.2
libyaml = True
root@ip-172-31-0-13:~# ssh-keygen -t rsa -b 2048 -f ~/.ssh/jenkins_key
26°C
Partly cloudy
Search
ENG
IN
23:02
28-12-2024

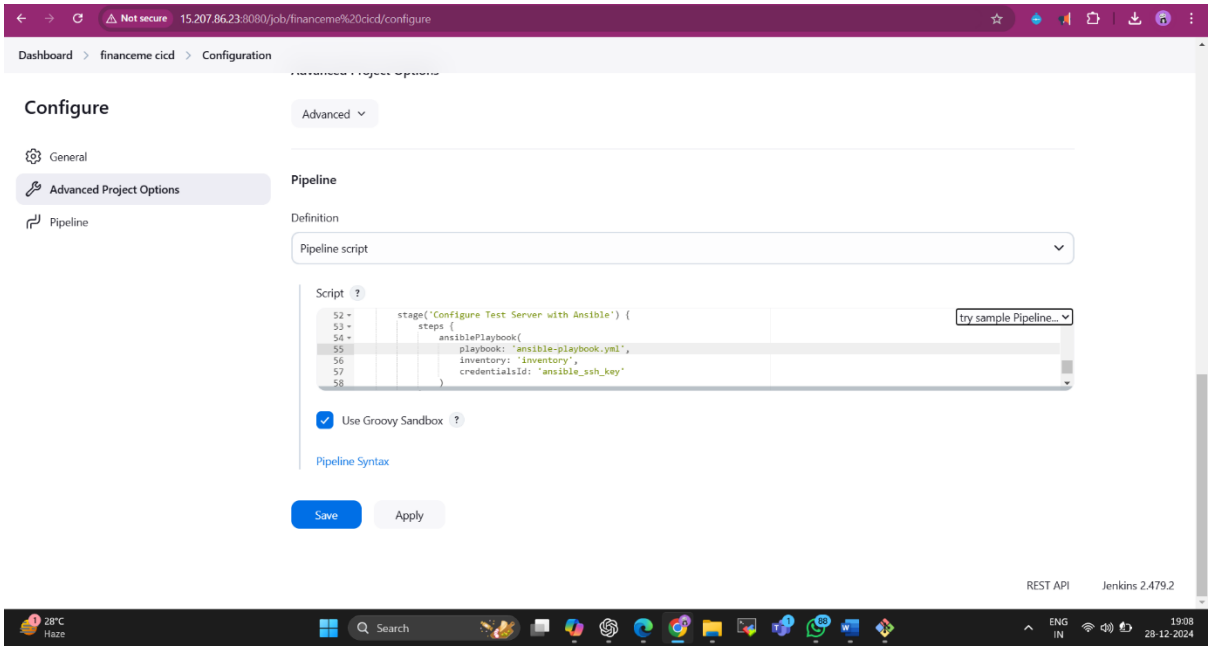

Not secure 15.207.86.23:8080/manage/credentials/store/system/domain/_/
Jenkins
Search (CTRL+K)
GISHNURAJ
log out
Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Global credentials (unrestricted)
+ Add Credentials
Credentials that should be available irrespective of domain specification to requirements matching.
ID
Name
Kind
Description
AWS_ACCESS_KEY_ID
AWS_ACCESS_KEY_ID
Secret text
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
AWS_SECRET_ACCESS_KEY
Secret text
AWS_SECRET_ACCESS_KEY
ansible_ssh_key
ubuntu
SSH Username with private key
Icon: S M L
REST API
Jenkins 2.479.2
26°C
Partly cloudy
Search
ENG
IN
23:47
28-12-2024

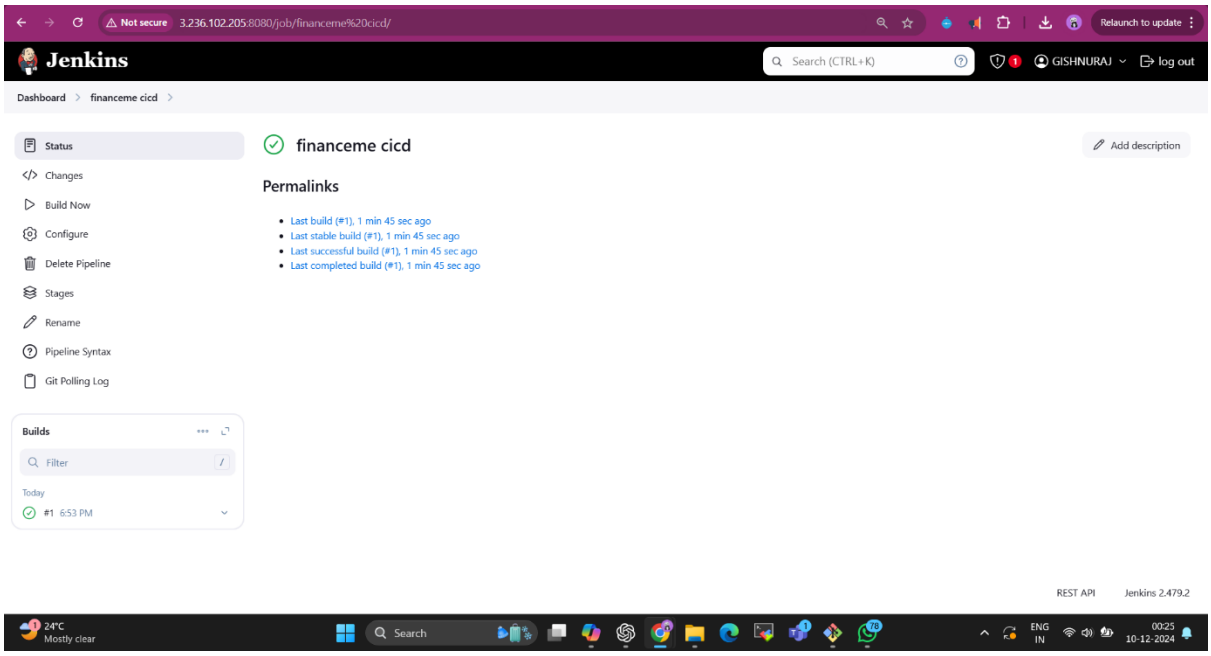

github.com/Gishnuraj489/star-agile-banking-finance/edit/master/ansible-playbook.yml
Gishnuraj489 / star-agile-banking-finance
Type / to search
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
star-agile-banking-finance / ansible-playbook.yml in master
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
No wrap
- name : Configure Docker on EC2 Instances
hosts : all
become: true
connection : ssh
tasks :
- name: updating apt
command : sudo apt-get update
- name : Install Docker
command : sudo apt-get install -y docker.io
- name : Start Docker Service
command : sudo systemctl start docker
- name: Deploy Docker Container
command: docker run -itd -p 8084:8081 gems070903/myproject0:1
Use Control + Shift + m to toggle the tab key moving focus. Alternatively, use esc then tab to move to the next interactive element on the page.
28°C
Haze
Search
ENG
IN
19:10
28-12-2024

Finally, integrating terraform and ansible in the pipeline scripts.

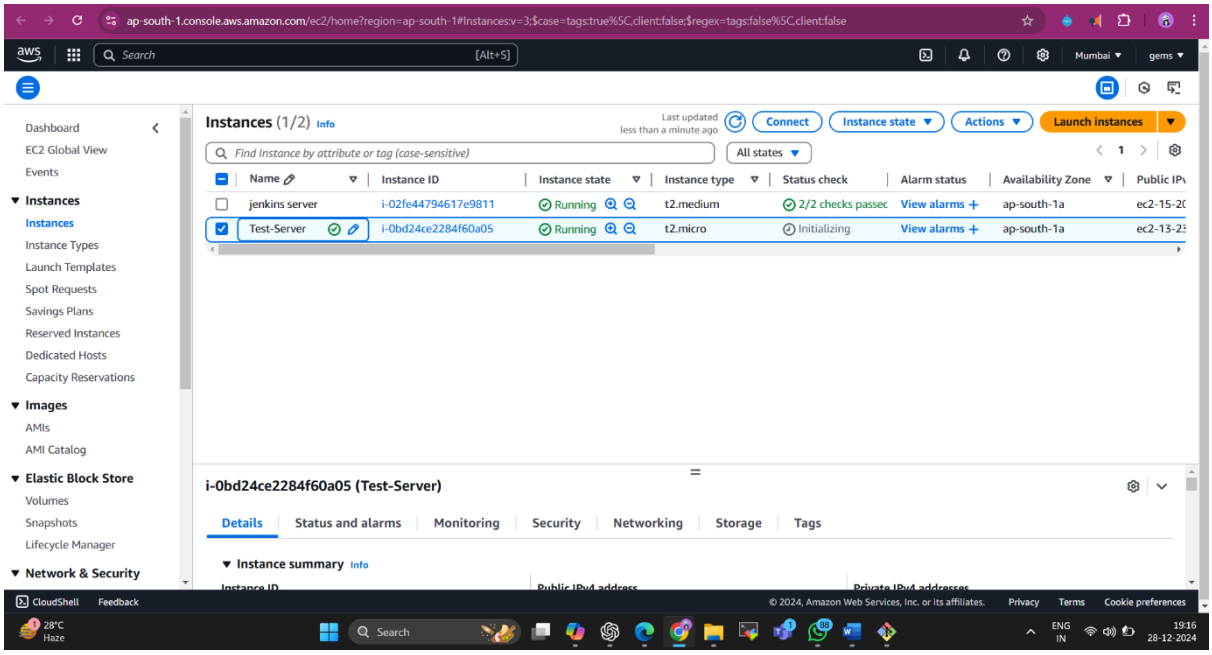


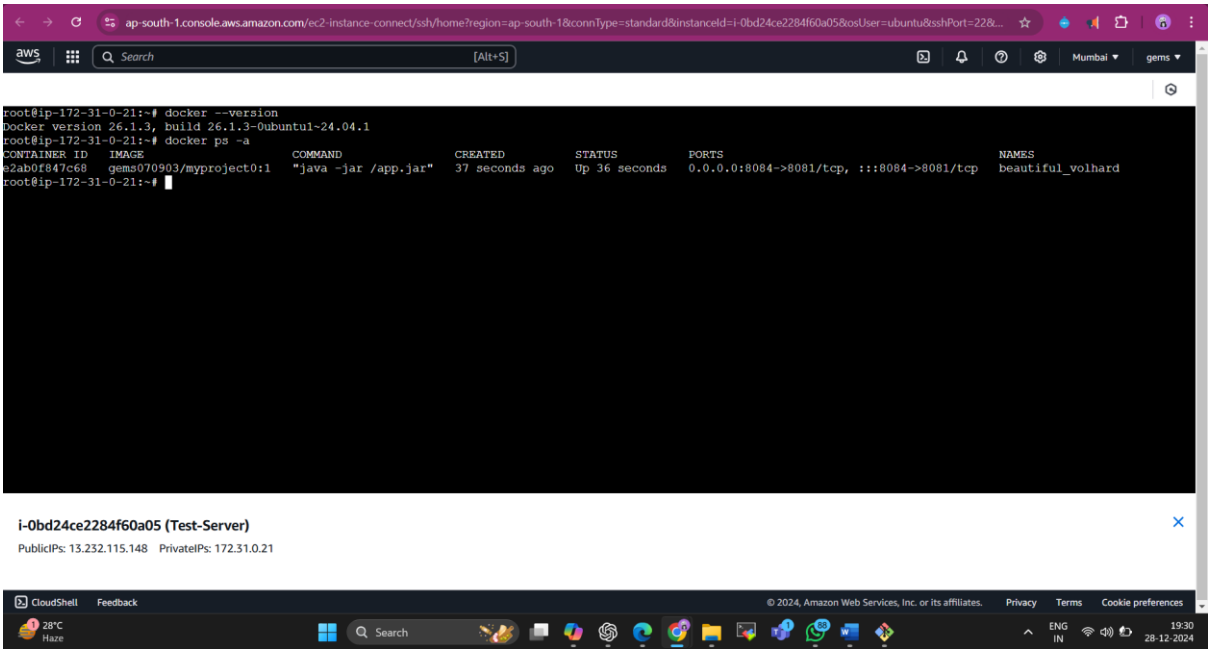


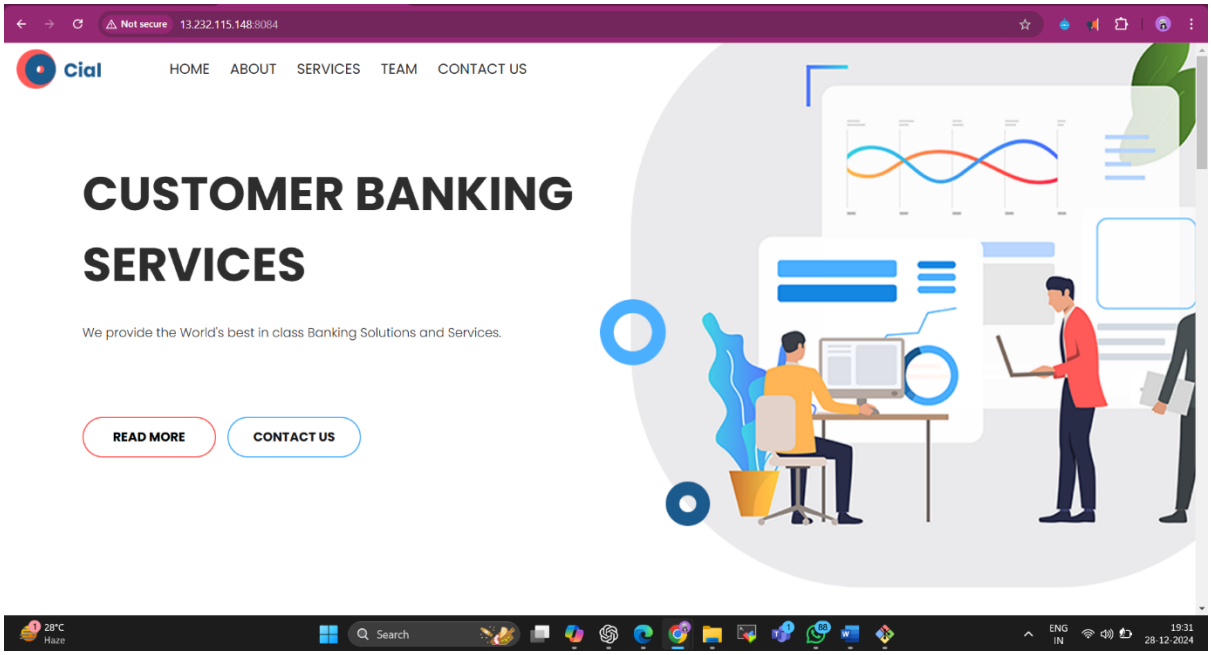

Not secure 15.207.86.23:8080/job/financeme%20cicd/configure
Dashboard > financeme cicd > Configuration
Configure
General
Advanced Project Options
Pipeline
Advanced
Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
stage('Configure Test Server with Ansible') {
steps {
ansiblePlaybook(
playbook: 'ansible-playbook.yml',
inventory: 'inventory',
credentialsId: 'ansible_ssh_key'
)
Use Groovy Sandbox
Pipeline Syntax
Save
Apply
REST API
Jenkins 2.479.2
28°C
Haze
Search
ENG
IN
19:08
28-12-2024

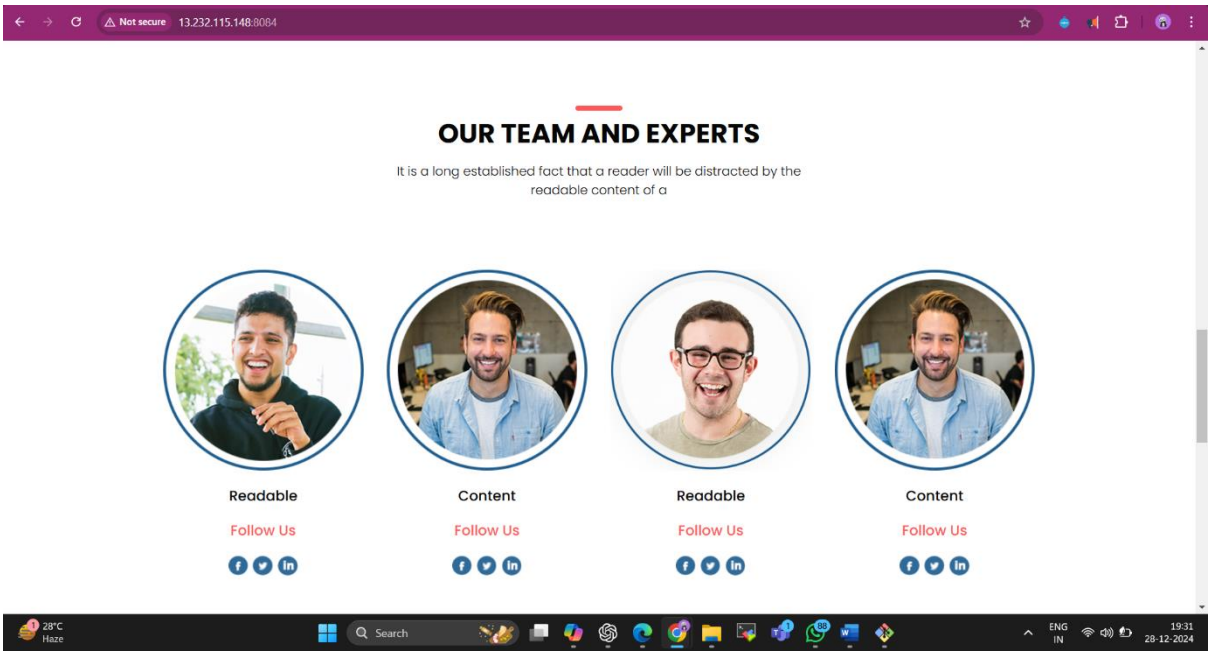

Not secure 3.236.102.205:8080/job/financeme%20cicd/
Relaunch to update
Jenkins
Search (CTRL+K)
GISHNURAJ
log out
Dashboard > financeme cicd
Status
Changes
Build Now
Configure
Delete Pipeline
Stages
Rename
Pipeline Syntax
Git Polling Log
financeme cicd
Add description
Permalinks
Last build (#1), 1 min 45 sec ago
Last stable build (#1), 1 min 45 sec ago
Last successful build (#1), 1 min 45 sec ago
Last completed build (#1), 1 min 45 sec ago
Builds
Filter
Today
#1 6:53 PM
REST API
Jenkins 2.479.2
24°C
Mostly clear
Search
ENG
IN
00:25
10-12-2024

Thus the test server is created ,then using ansible →docker is installed and then the application is deployed in it.

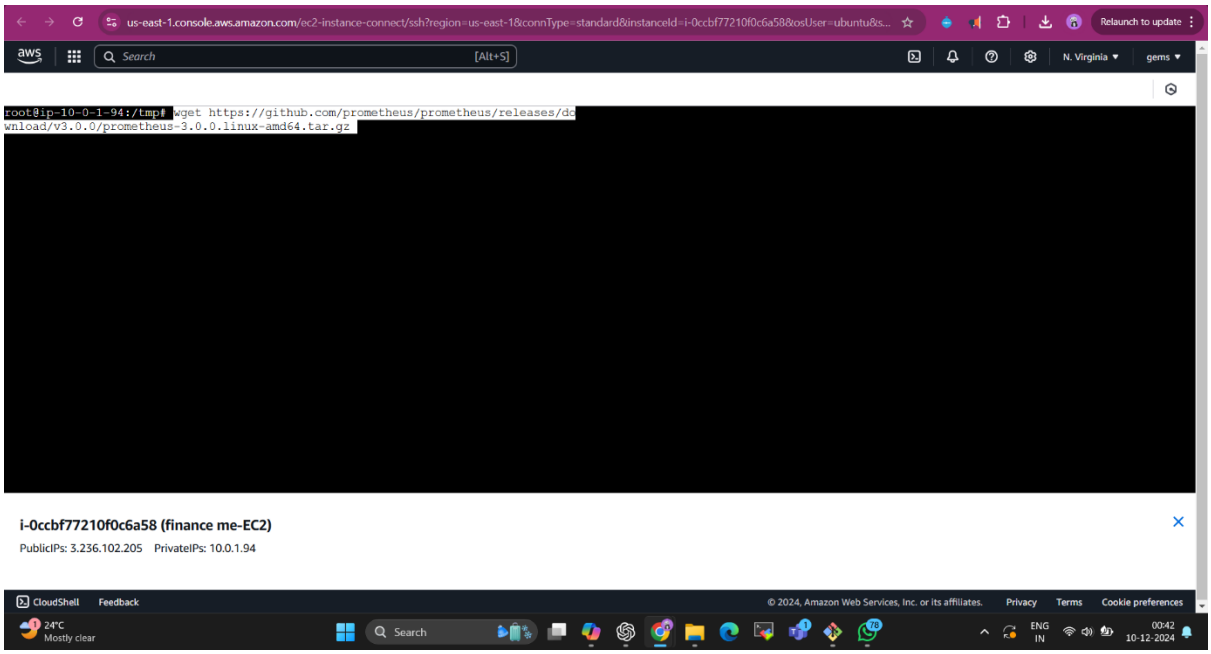





Not secure 13.232.115.148:8084
Cial
HOME ABOUT SERVICES TEAM CONTACT US
CUSTOMER BANKING SERVICES
We provide the World's best in class Banking Solutions and Services.
READ MORE
CONTACT US
28°C Haze
Search
ENG IN
19:31 28-12-2024


Not secure 13.232.115.148:8084
OUR TEAM AND EXPERTS
It is a long established fact that a reader will be distracted by the readable content of a
Readable
Follow Us
Content
Follow Us
Readable
Follow Us
Content
Follow Us
28°C Haze
Search
ENG IN
19:31 28-12-2024

Then, install the Prometheus metrics plugin for the continuous monitoring of the Jenkins job. → then restart the Jenkins server.



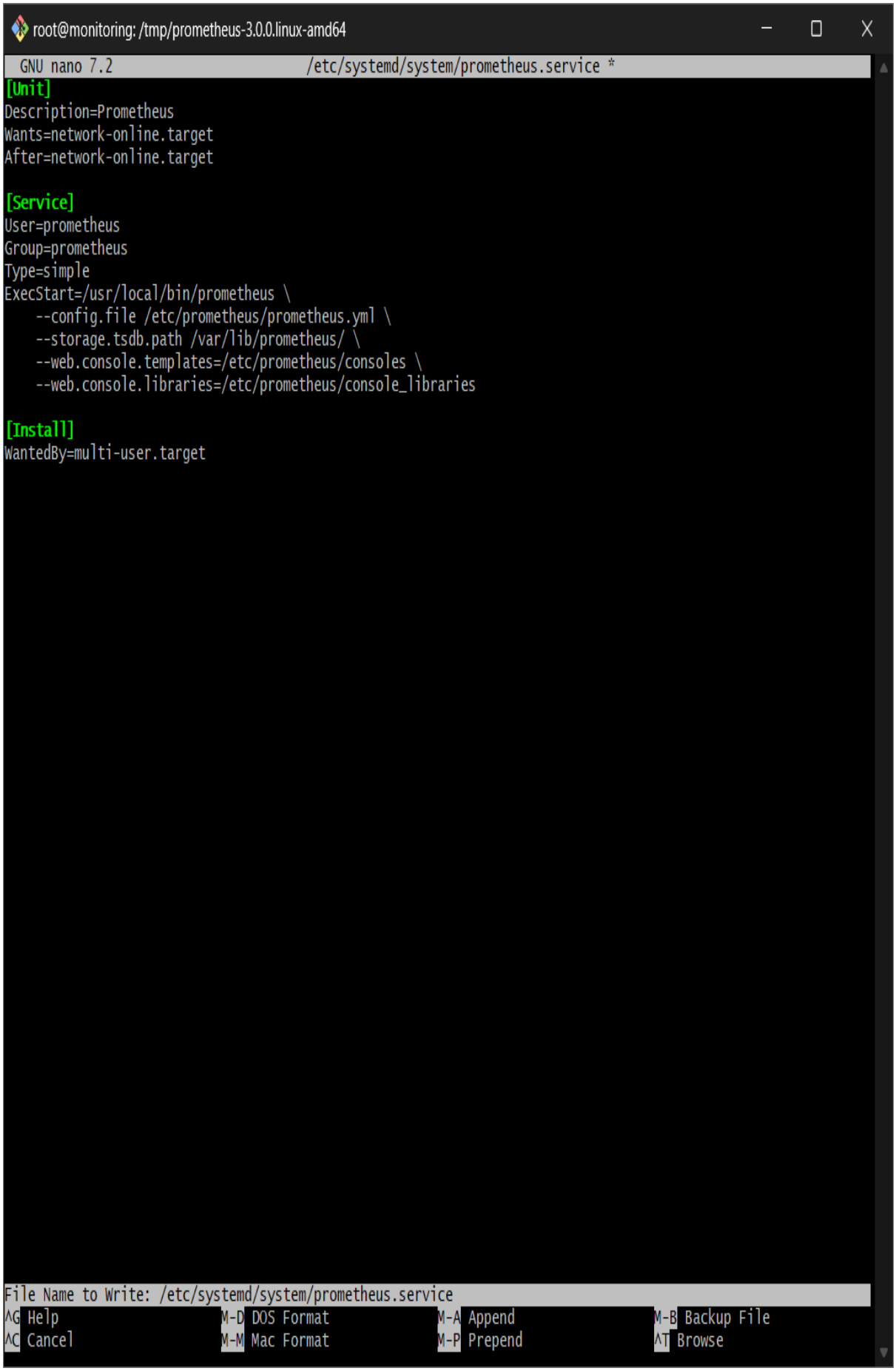
Monitoring of this Jenkins job using Prometheus and Grafana:

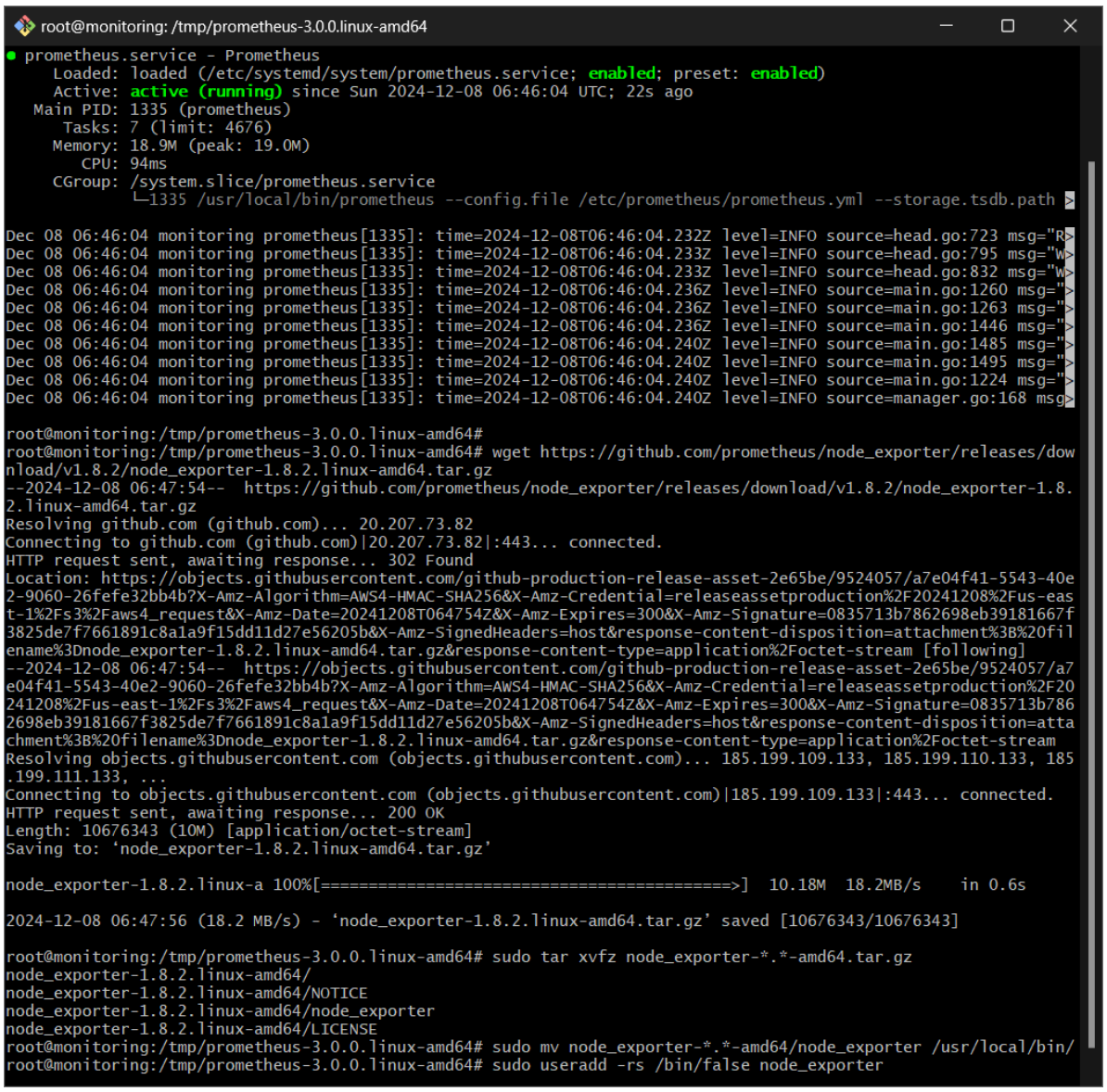
Installing and configuring Prometheus and Grafana as monitoring and visualization tools.

STEPS:

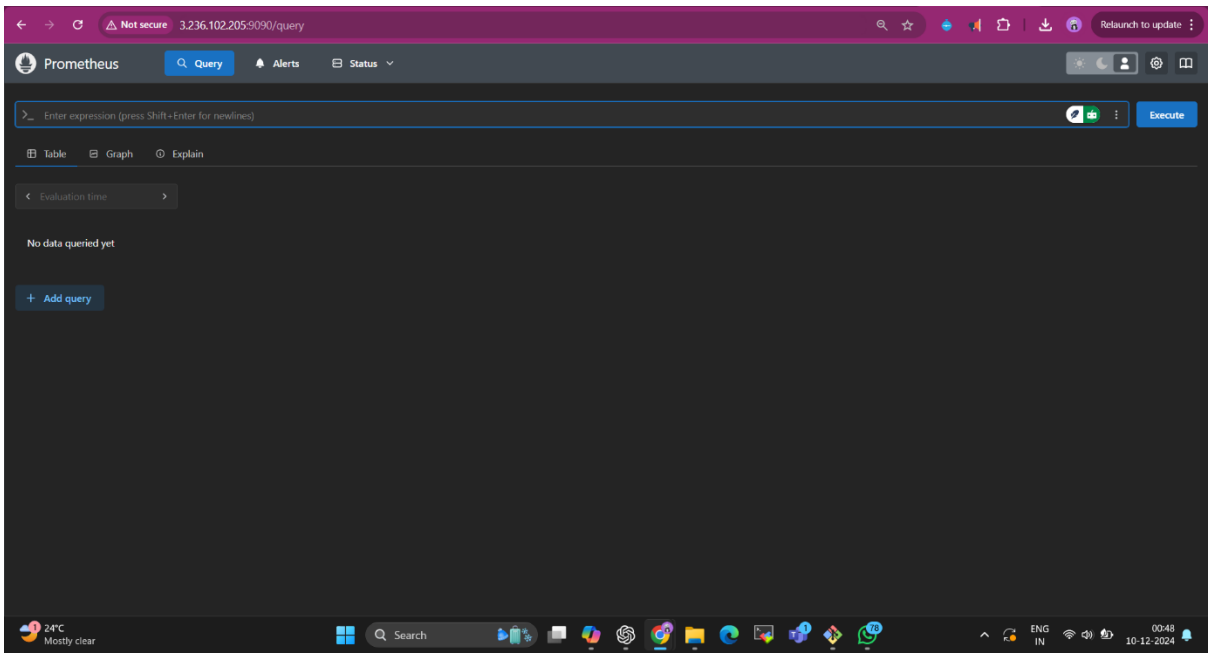
Create a system user for Prometheus using below commands → Create the directories in which we will be storing our configuration files and libraries

Move the configuration file and set the owner to the Prometheus → Move the binaries and set the owner → Creating the Prometheus service file in the directory name**: /etc/systemd/system/prometheus.service**

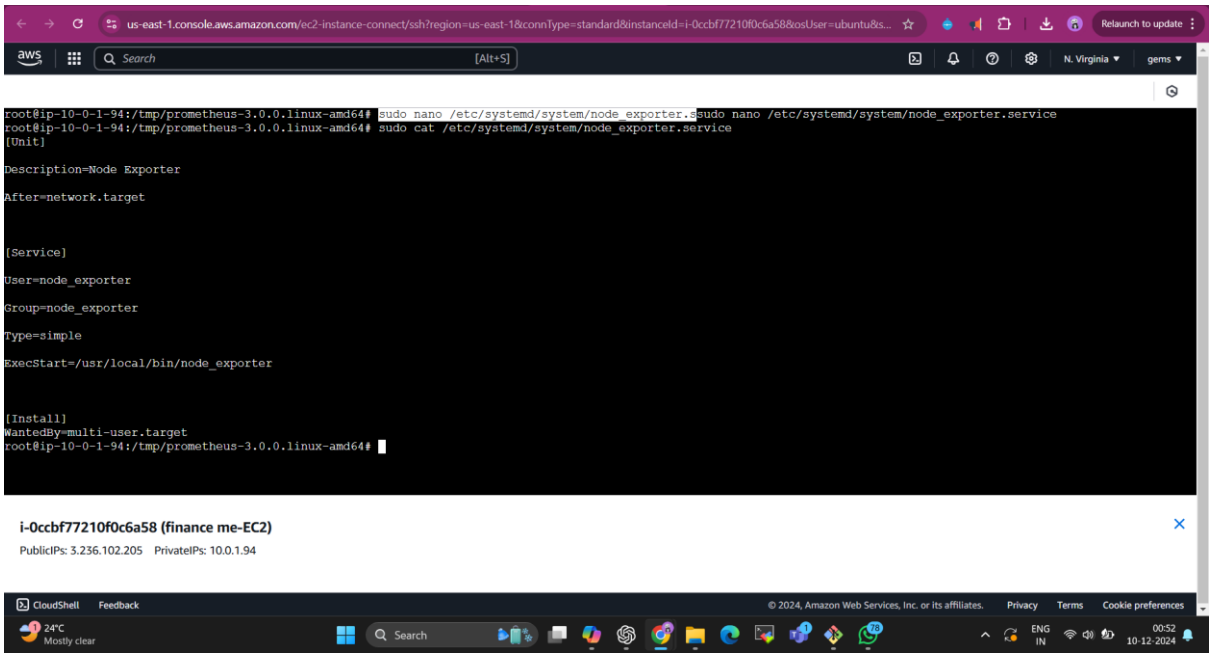

root@monitoring: /tmp/prometheus-3.0.0.linux-amd64
GNU nano 7.2 /etc/systemd/system/prometheus.service *
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
[Install]
WantedBy=multi-user.target
File Name to Write: /etc/systemd/system/prometheus.service
^G Help M-D DOS Format M-A Append M-B Backup File
^C Cancel M-M Mac Format M-P Prepend ^T Browse



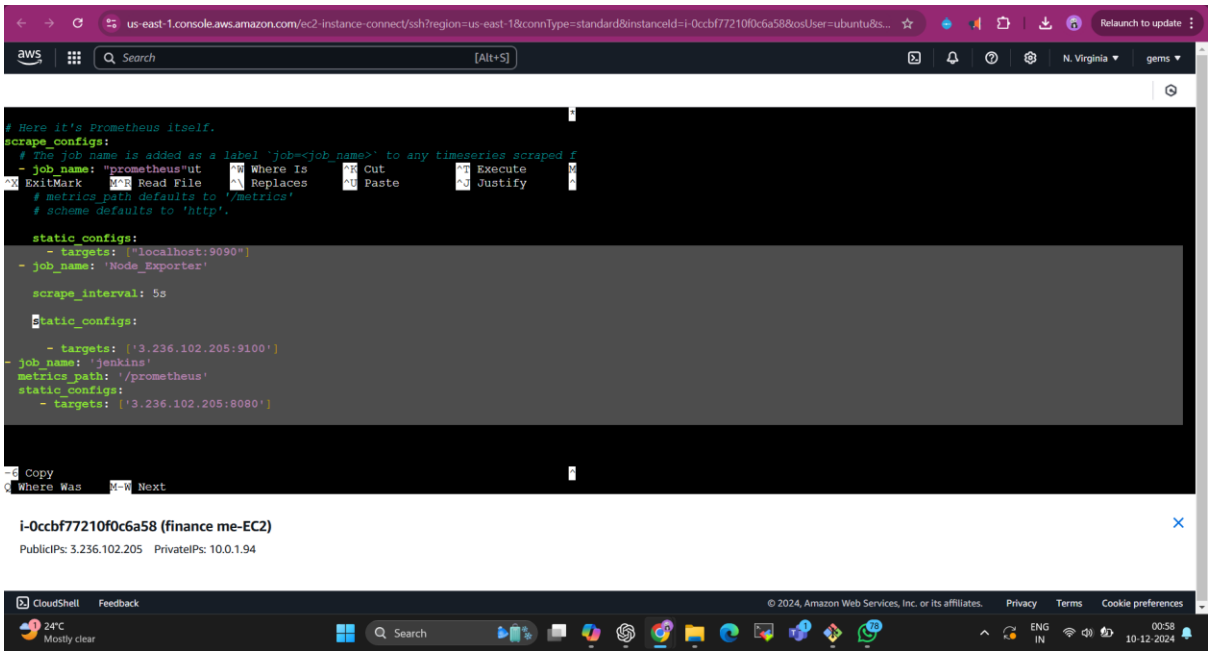
Reload systemd → Start and enable Prometheus service → Now accessing Prometheus in your browser in the PORT NO:9090.



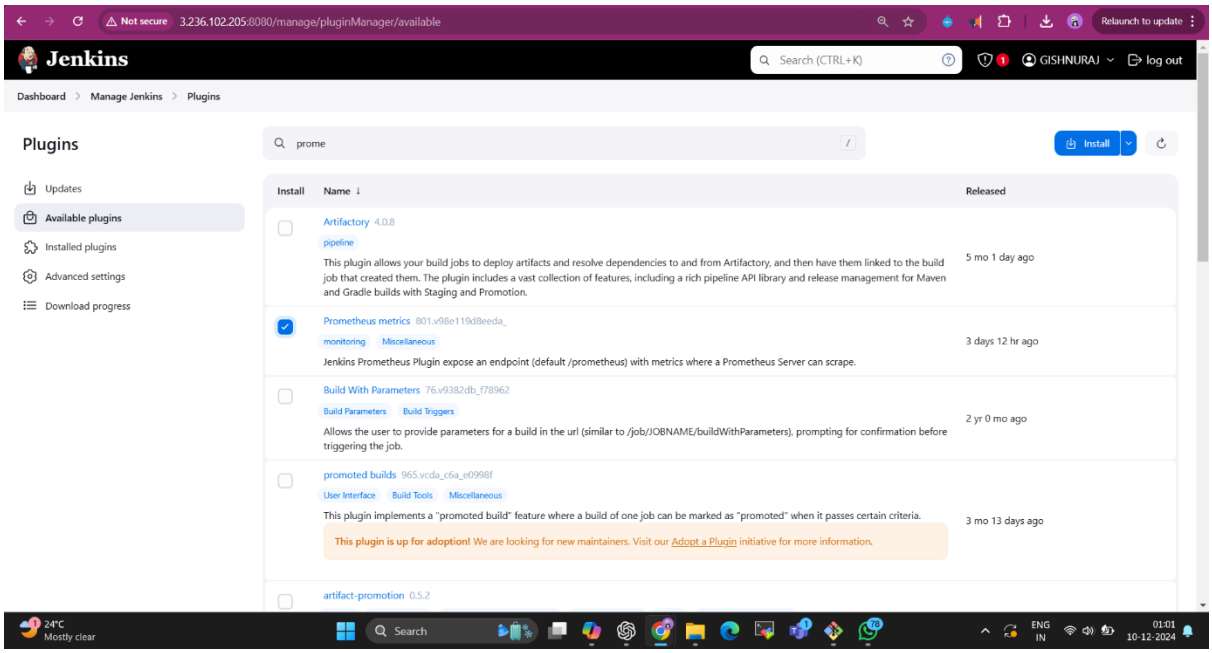
Installing Node Exporter on Ubuntu 22.04 LTS → Move the binary file of node exporter to /usr/local/bin location → Create a node_exporter user to run the node exporter service →Create a Custom Node Exporter Service in the name: **/etc/systemd/system/node_exporter.service.**



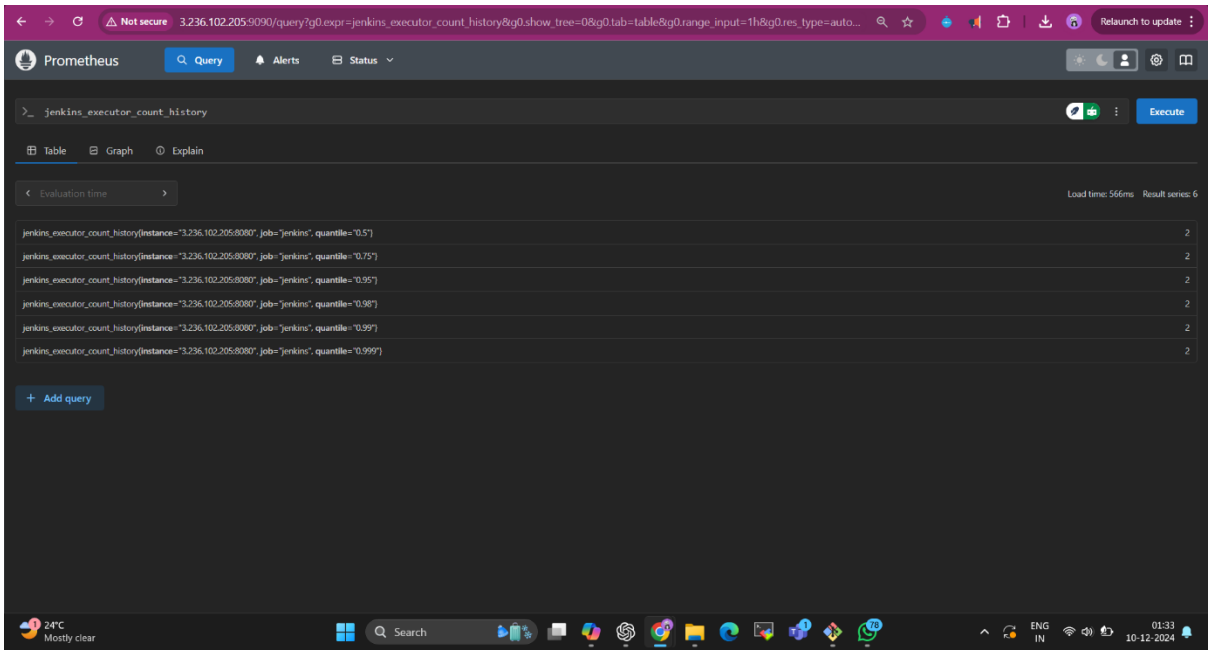
Reload the system → Start, enable node exporter → update our with jobs [node exporter, jenkin jobs] configuration file:
/etc/prometheus/prometheus.yml

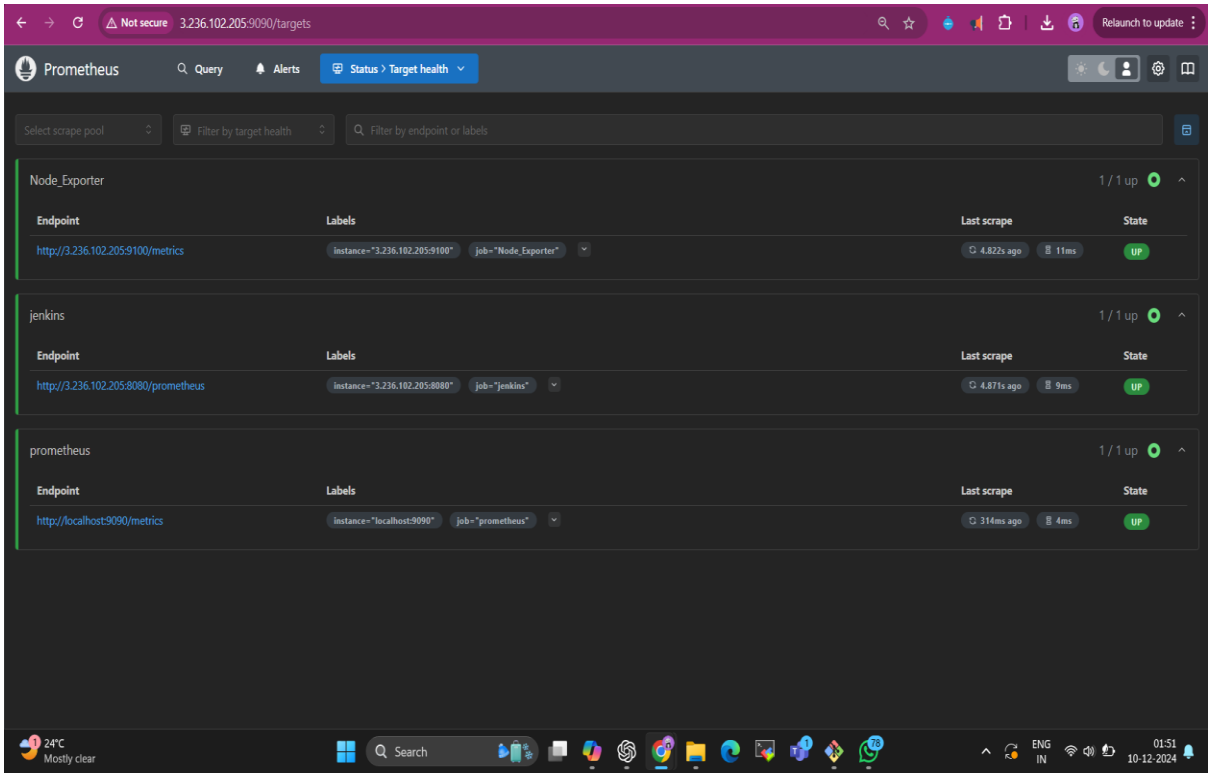


installing the Prometheus metric plugin so that the jenkin jobs can be monitored continuously.



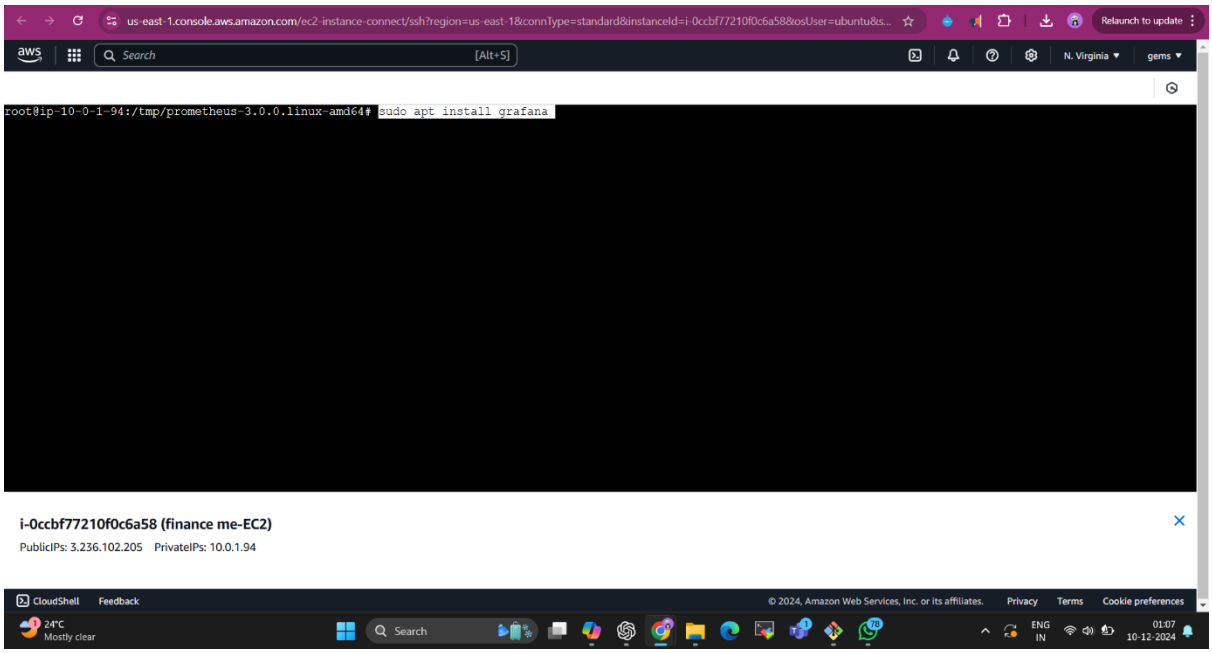
[Note: After changes any configuration file we need to restart our Prometheus]





Installing Grafana on Ubuntu 22.04 LTS with necessary packages.

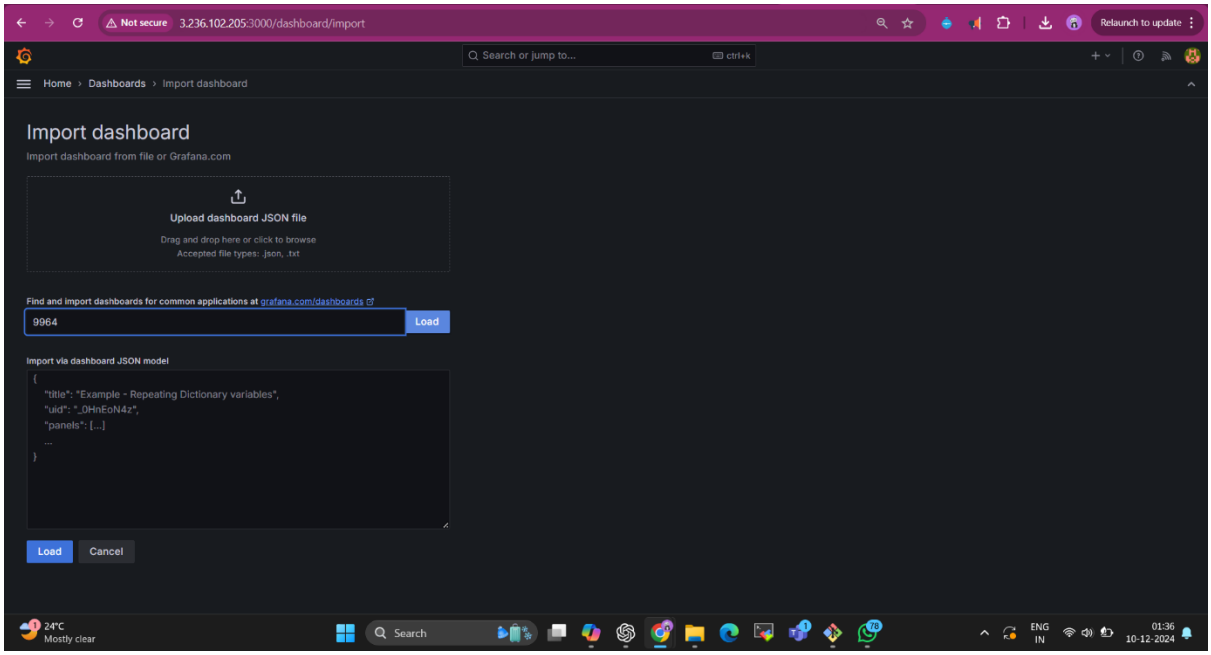
Once Grafana is installed, use systemctl command to start the Grafana server → we can access Grafana server in the port no:3000.

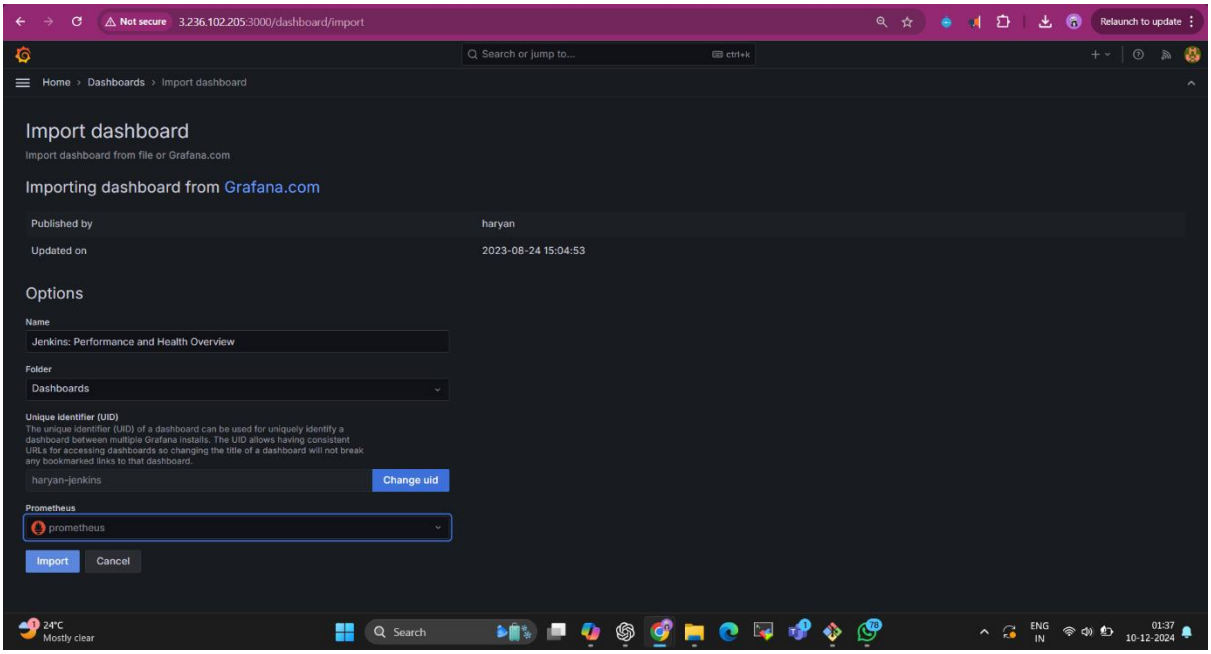


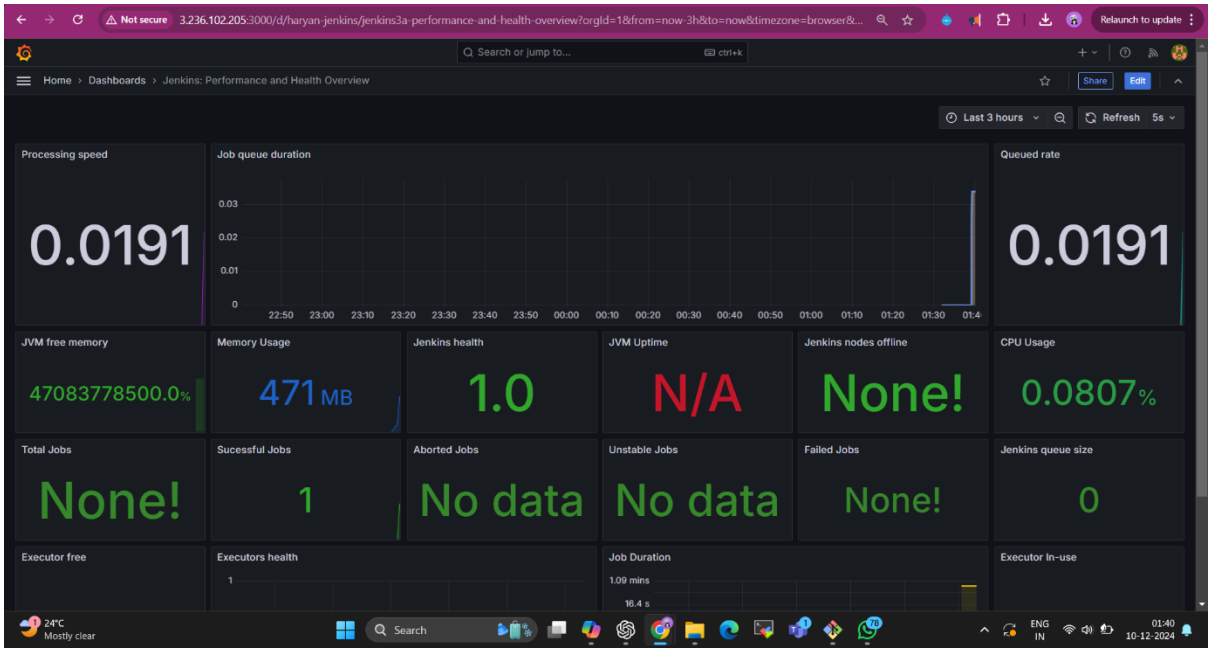
Clicking Add data source and select Prometheus → For the URL, enter http://3.236.102.205:9090 and click Save and test .

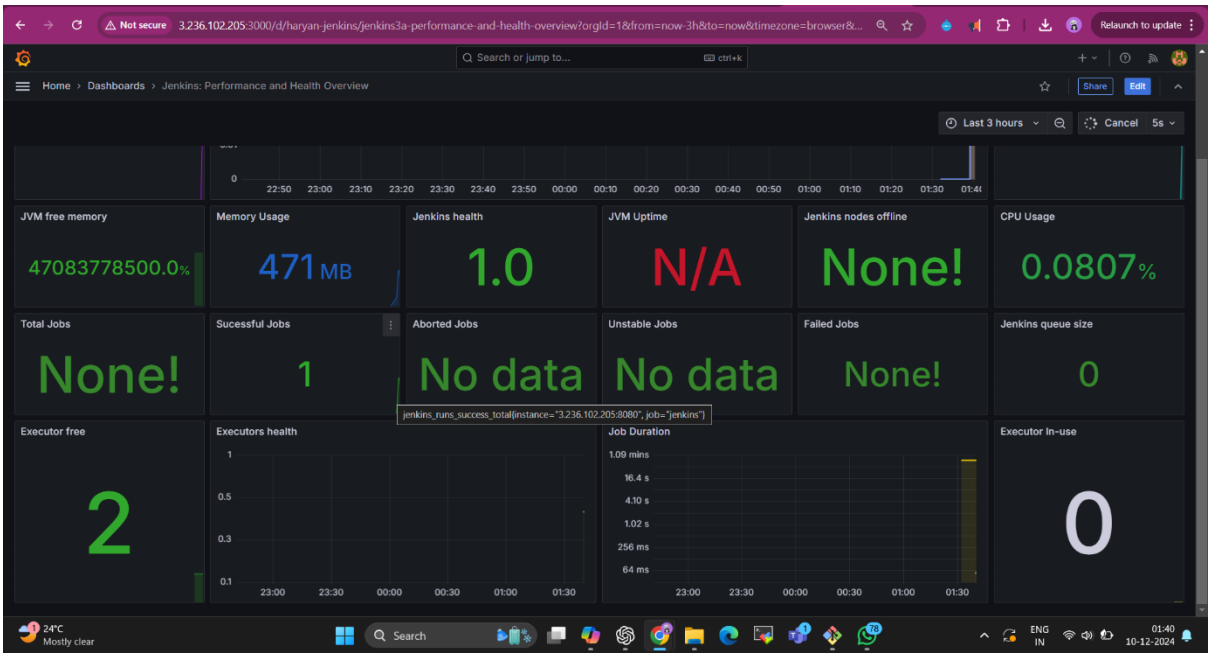
Finally, adding Dashboard for a better view in Grafana.

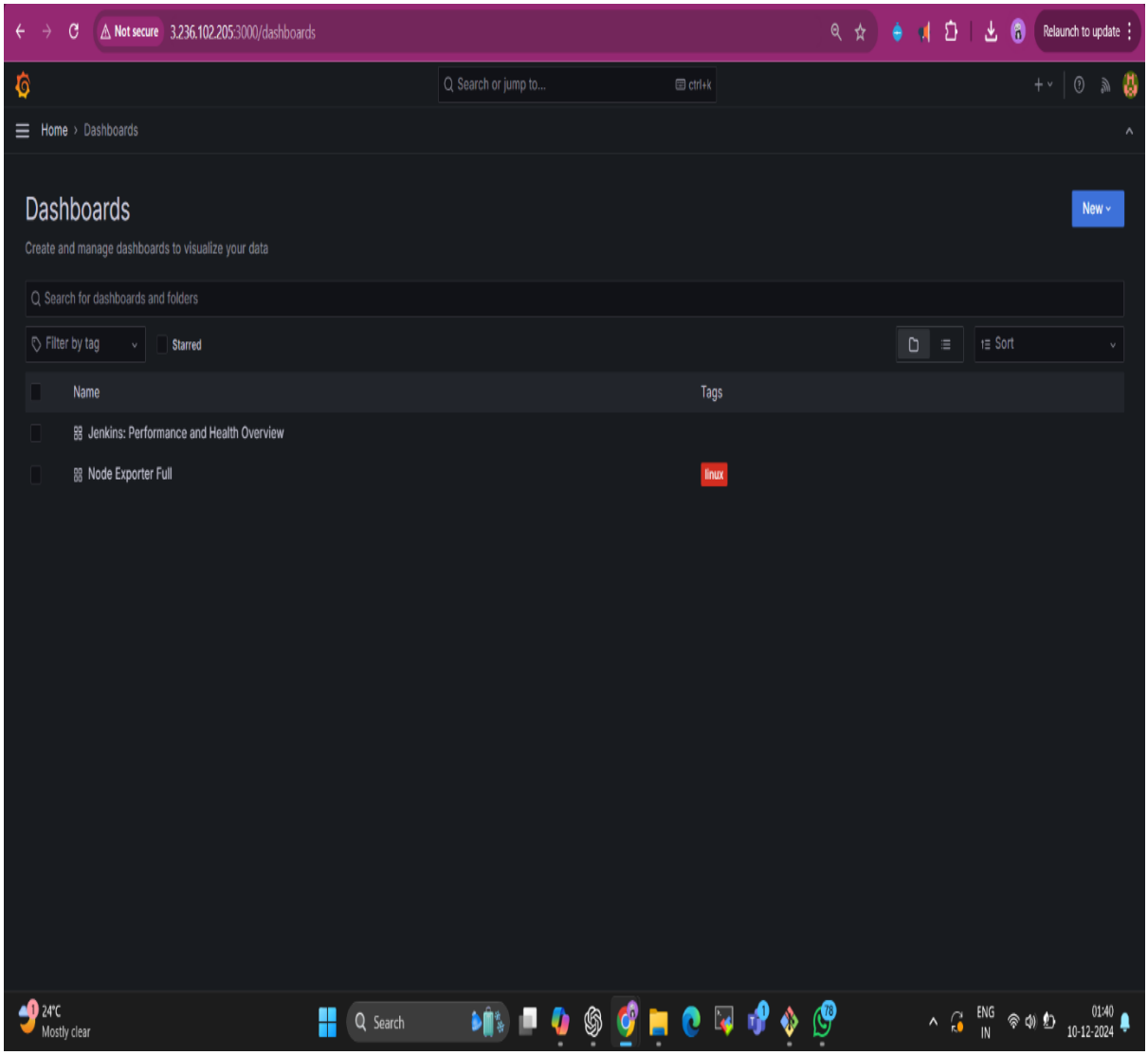
Click On Dashboard → Import Dashboard then paste the code 9964 and click on load to see the Jenkins and its jobs performance metrices.


Not secure 3.236.102.205:3000/dashboard/import
Relaunch to update
Search or jump to... ctrl+k
Home › Dashboards › Import dashboard
Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt
Find and import dashboards for common applications at grafana.com/dashboards
9964
Load
Import via dashboard JSON model
{
"title": "Example - Repeating Dictionary variables",
"uid": "_0HnEoN4z",
"panels": [...]
...
}
Load
Cancel
24°C
Mostly clear
Search
ENG
IN
01:36
10-12-2024


Not secure 3.236.102.205:3000/dashboard/import
Relaunch to update
Search or jump to... ctrl+k
Home › Dashboards › Import dashboard
Import dashboard
Import dashboard from file or Grafana.com
Importing dashboard from Grafana.com
Published by
haryan
Updated on
2023-08-24 15:04:53
Options
Name
Jenkins: Performance and Health Overview
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
haryan-jenkins
Change uid
Prometheus
prometheus
Import
Cancel
24°C
Mostly clear
Search
ENG
IN
01:37
10-12-2024


Not secure 3.236.102.205:3000/d/haryan-jenkins/jenkins3a-performance-and-health-overview?orgId=1&from=now-3h&to=now&timezone=browser&...
Relaunch to update
Search or jump to... ctrl+k
Home › Dashboards › Jenkins: Performance and Health Overview
Share
Edit
Last 3 hours
Refresh 5s
Processing speed
0.0191
Job queue duration
0.03
0.02
0.01
0
22:50 23:00 23:10 23:20 23:30 23:40 23:50 00:00 00:10 00:20 00:30 00:40 00:50 01:00 01:10 01:20 01:30
Queued rate
0.0191
JVM free memory
47083778500.0%
Memory Usage
471 MB
Jenkins health
1.0
JVM Uptime
N/A
Jenkins nodes offline
None!
CPU Usage
0.0807%
Total Jobs
None!
Sucessful Jobs
1
Aborted Jobs
No data
Unstable Jobs
No data
Failed Jobs
None!
Jenkins queue size
0
Executor free
Executors health
1
Job Duration
1.09 mins
16.4 s
Executor In-use
24°C
Mostly clear
Search
ENG
IN
01:40
10-12-2024

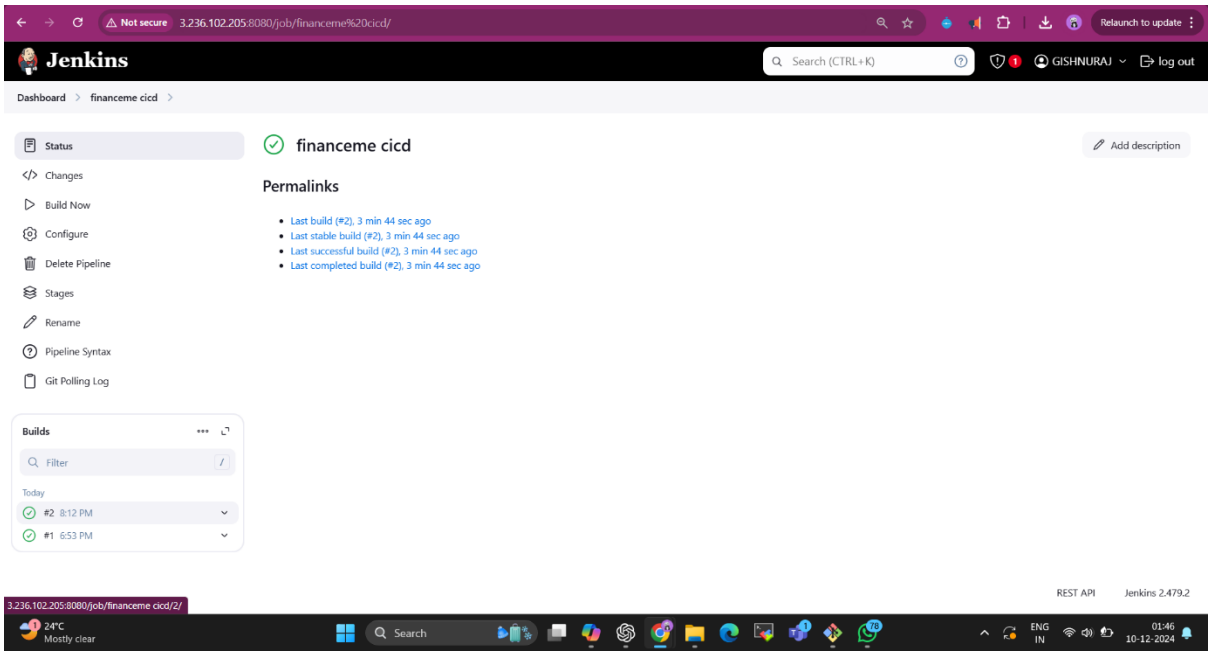

Not secure 3.236.102.205:3000/d/haryan-jenkins/jenkins3a-performance-and-health-overview?orgId=1&from=now-3h&to=now&timezone=browser&...
Relaunch to update
Search or jump to... ctrl+k
Home › Dashboards › Jenkins: Performance and Health Overview
Share
Edit
Last 3 hours
Cancel 5s
0
22:50 23:00 23:10 23:20 23:30 23:40 23:50 00:00 00:10 00:20 00:30 00:40 00:50 01:00 01:10 01:20 01:30
JVM free memory
47083778500.0%
Memory Usage
471 MB
Jenkins health
1.0
JVM Uptime
N/A
Jenkins nodes offline
None!
CPU Usage
0.0807%
Total Jobs
None!
Sucessful Jobs
1
Aborted Jobs
No data
Unstable Jobs
No data
Failed Jobs
None!
Jenkins queue size
0
jenkins_runs_success_total{instance="3.236.102.205:8080", job="jenkins"}
Executor free
2
Executors health
1
0.5
0.3
0.1
23:00 23:30 00:00 00:30 01:00 01:30
Job Duration
1.09 mins
16.4 s
4.10 s
1.02 s
256 ms
64 ms
23:00 23:30 00:00 00:30 01:00 01:30
Executor In-use
0
24°C
Mostly clear
Search
ENG
IN
01:40
10-12-2024


Not secure 3.236.102.205:3000/dashboards
Relaunch to update
Search or jump to... ctrl+k
Home › Dashboards
Dashboards
Create and manage dashboards to visualize your data
New
Search for dashboards and folders
Filter by tag
Starred
Sort
Name
Tags
Jenkins: Performance and Health Overview
Node Exporter Full
linux
24°C
Mostly clear
Search
ENG
IN
01:40
10-12-2024

CONCLUSION:

As we built the CICD pipeline, if there are any changes committed in the Git hub repo it immediately triggers the pipeline, then the updated code (application) gets deployed through docker containers. And, we can monitor the Jenkins job (pipeline) continuously through Prometheus & visualize the data in Grafana.

Whenever there is a commit in the repo, it automatically triggers the pipeline(#2build).



In the previous Jenkins dash board you can see the successful job count is “1”.

Thus, the continuous monitoring tool updated the successful job count as “2”.