Healthcare Domain

A PROJECT REPORT

SUBMITTED BY

ANMOL UPADHAYA

DEVOPS Course

SUBMITTED TO

Star Agile

APR & 2025

Certification Project

Medicure is a super specialty hospital based in New York, USA and provide world class treatment and surgery including Heart, Liver, Kidney transplants and first robotic surgery center. The chain is owned and managed by Global Health Limited. The Medicure would centrally like to manage all the doctor's and patient's data across the Medicure hospitals in various cities. They have developed an microservice, which offers these services. In order to reduce unnecessary maintenance cost and manual labor, they would like to automate their application build and deployment process using DevOps. They are fine to use any one of the (AWS, Azure, GCP) cloud platform as their primary cloud service provider. The company's primary goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers. They would like to use Kubernetes to manage their container deployments, scaling and descaling of containers etc. Initially, Medicure is facing multiple problems, because of involved complexity in application as well as Infrastructure.

Following are the problems:

✓ Building Complex builds is difficult

✓ Manual efforts to test various components/modules of the project

✓ Incremental builds are difficult to manage, test and deploy

✓ Creation of infrastructure and configure it manually is very time consuming

✓ Continuous manual monitoring the application is quite challenging.

Later, you need to implement Continuous Integration & Continuous Deployment using following tools:

- ✓ Git - For version control for tracking changes in the code files
- ✓ Jenkins - For continuous integration and continuous deployment
- ✓ Docker - For containerizing applications
- ✓ Ansible - Configuration management tools
- ✓ Selenium - For automating tests on the deployed web application
- ✓ Terraform - For creation of infrastructure.
- ✓ Kubernetes – for running containerized application in managed cluster.

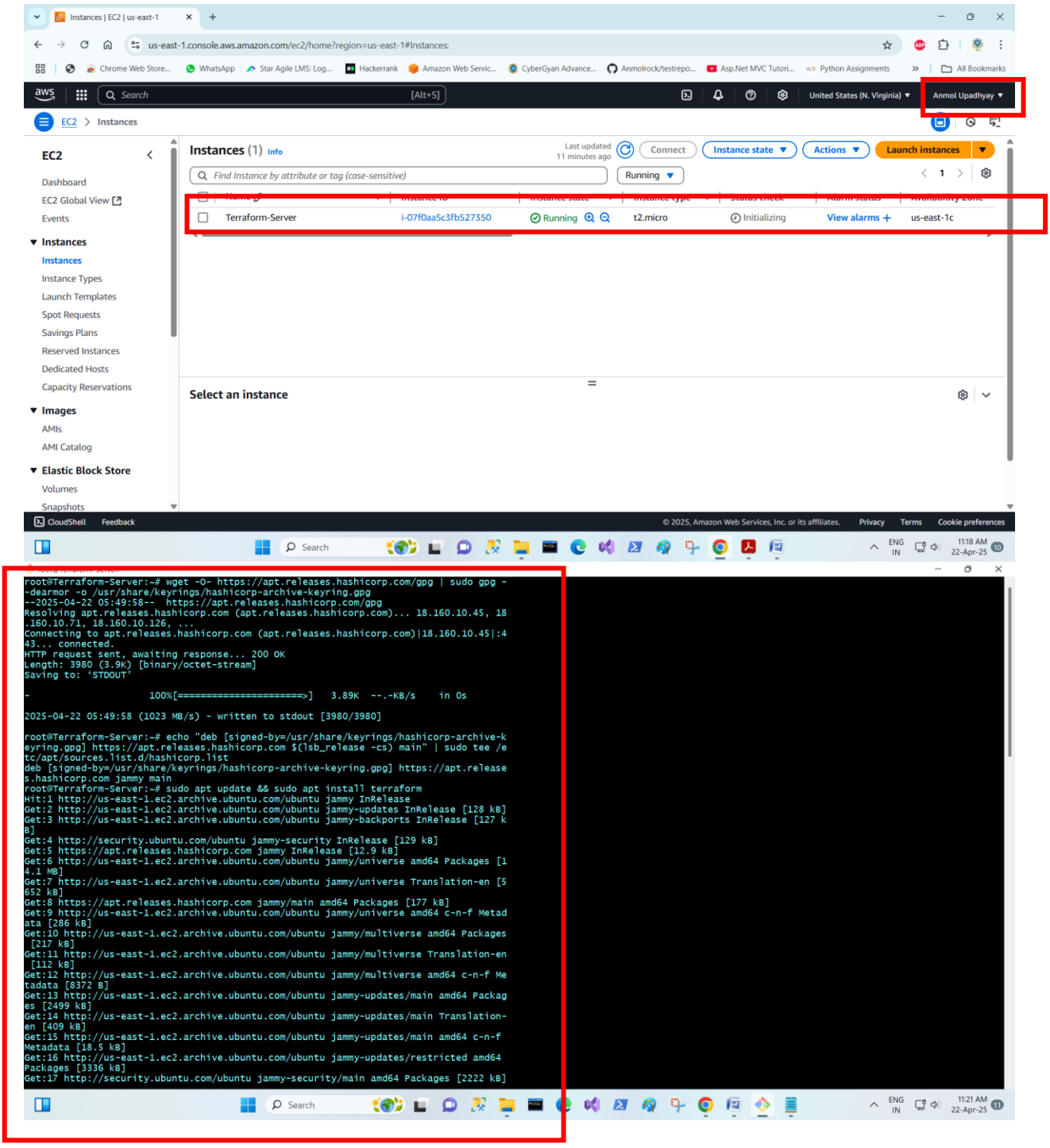
Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the Jenkins pipeline should be triggered and code should be checkout, compiled, tested, packaged and containerized. A new test-cluster should be provisioned and configured automatically with all the required software's and as soon as the cluster is healthy and available, the application must be deployed to the test-server automatically using Kubernetes.

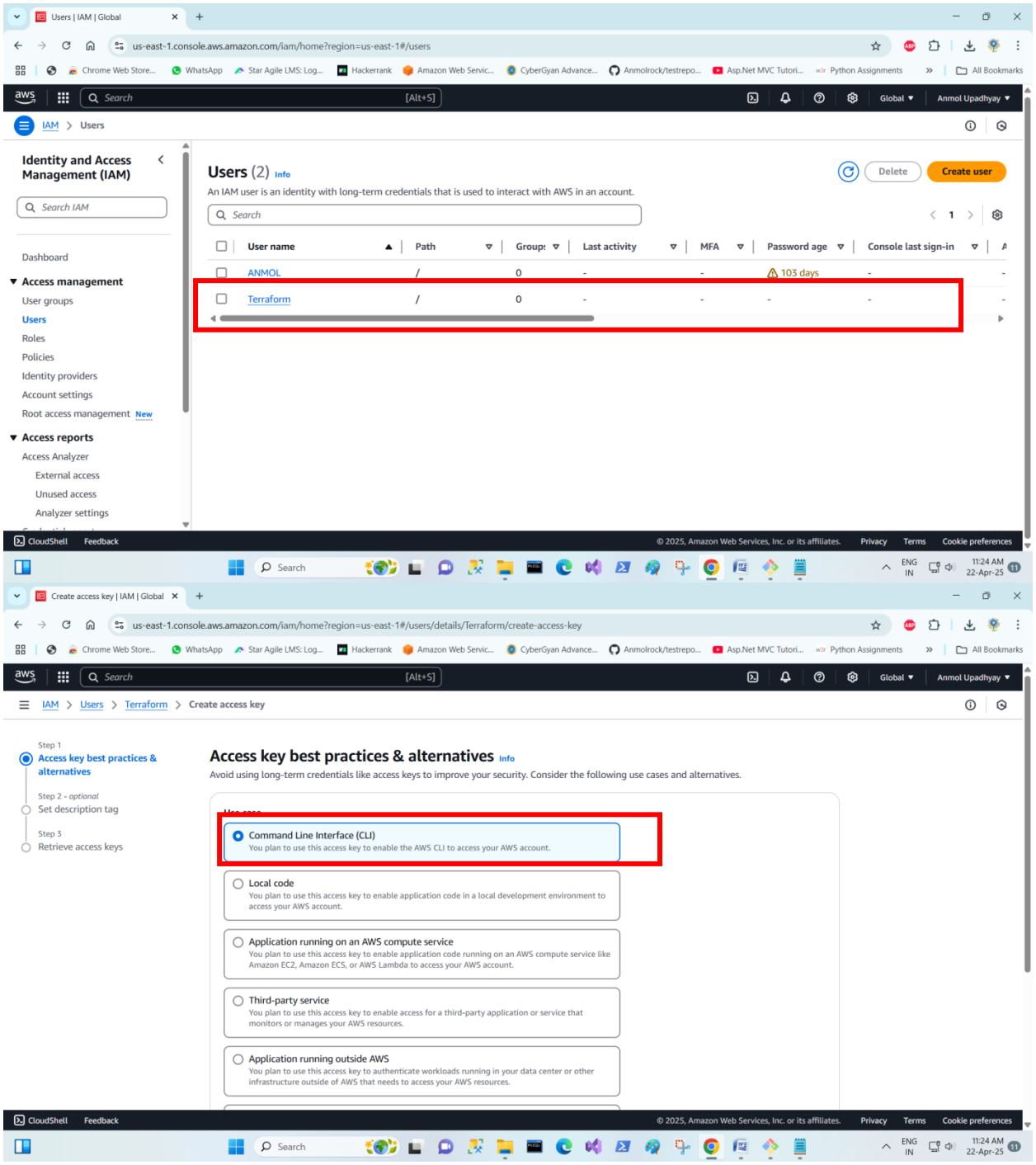
The deployment should then be tested using a test automation tool, and if the build is successful, it should be deployed to the prod server/cluster using Kubernetes. All this should happen automatically and should be triggered from a push to the GitHub master branch.

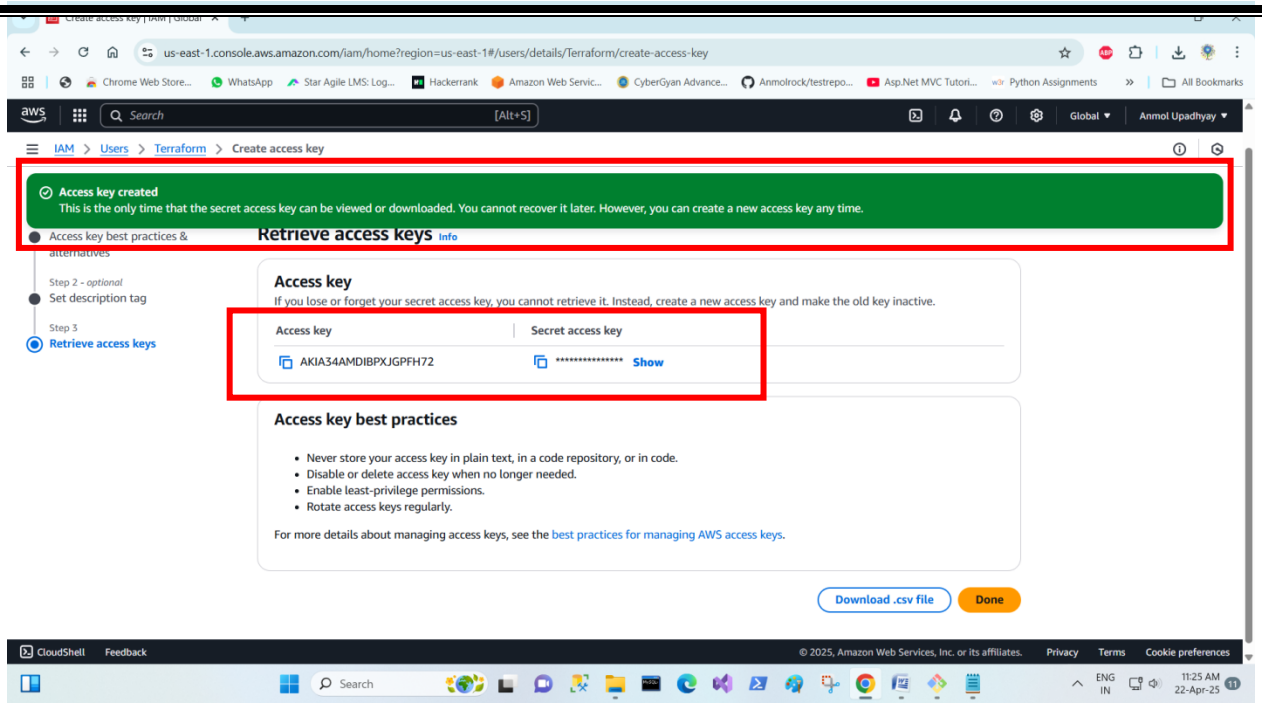
Kubernetes cluster must contain at least 2 servers and must be monitored continuously using Prometheus and dashboard must be visualized using Grafana.

Step1: Create a Terraform – Server to Deploy a DEV and TEST Server for Further Process.

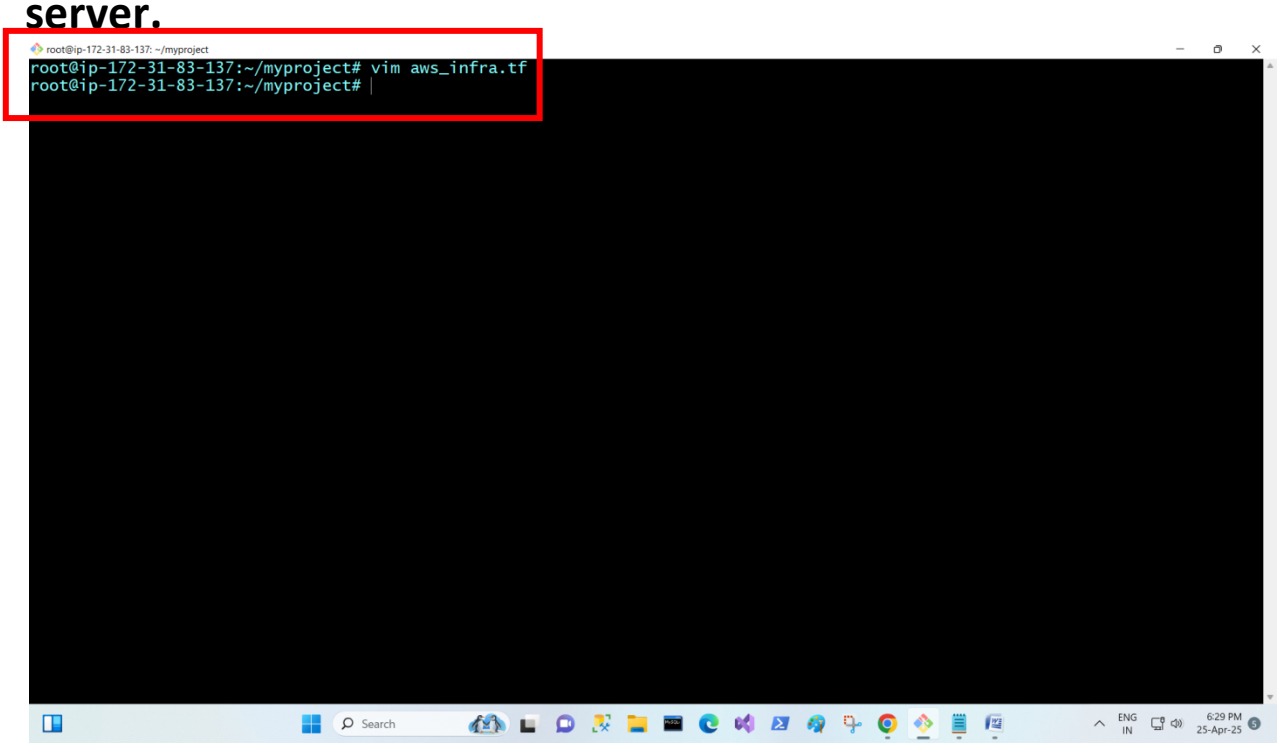


Step2: Create Access key and secret key in AWS for Terraform to connect





Step3: Write Terraform configuration file to create an 2 EC2 server.



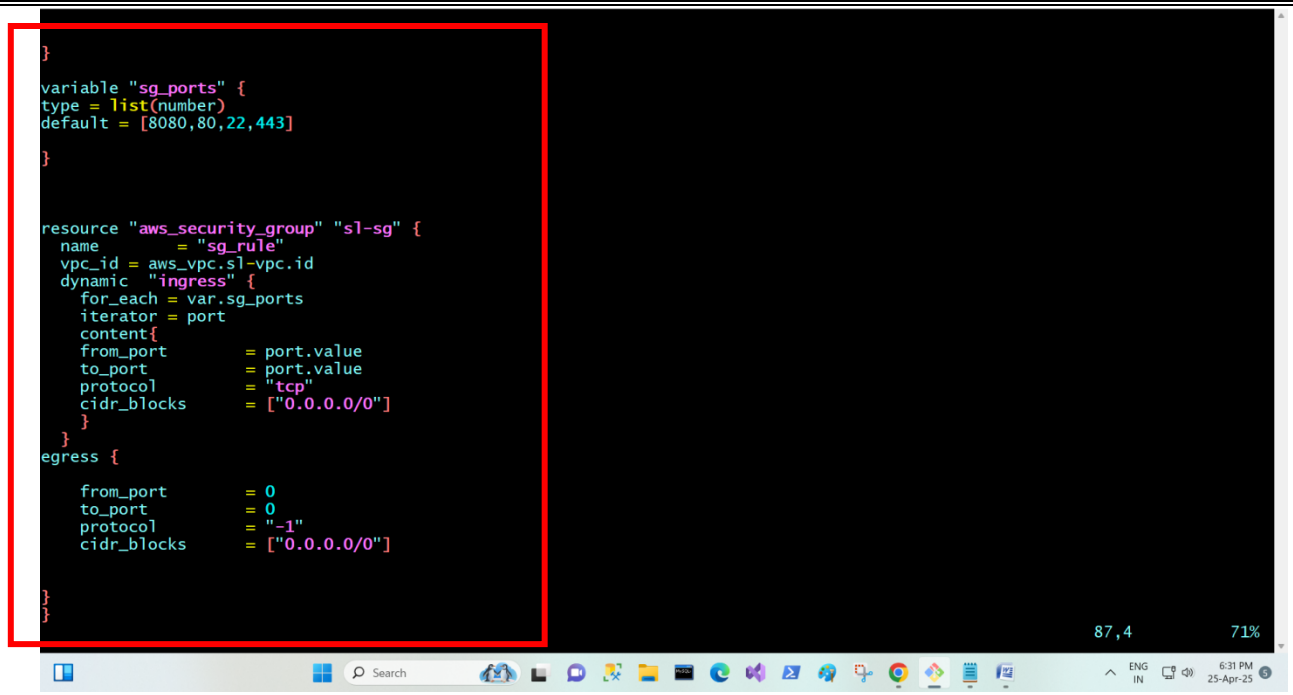
```
provider "aws" {

region = "us-east-1"


}


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
  tags = {
   Name = "sl-route-table"
}
```

24,4 Top

root@ip-172-31-83-137: ~/myproject

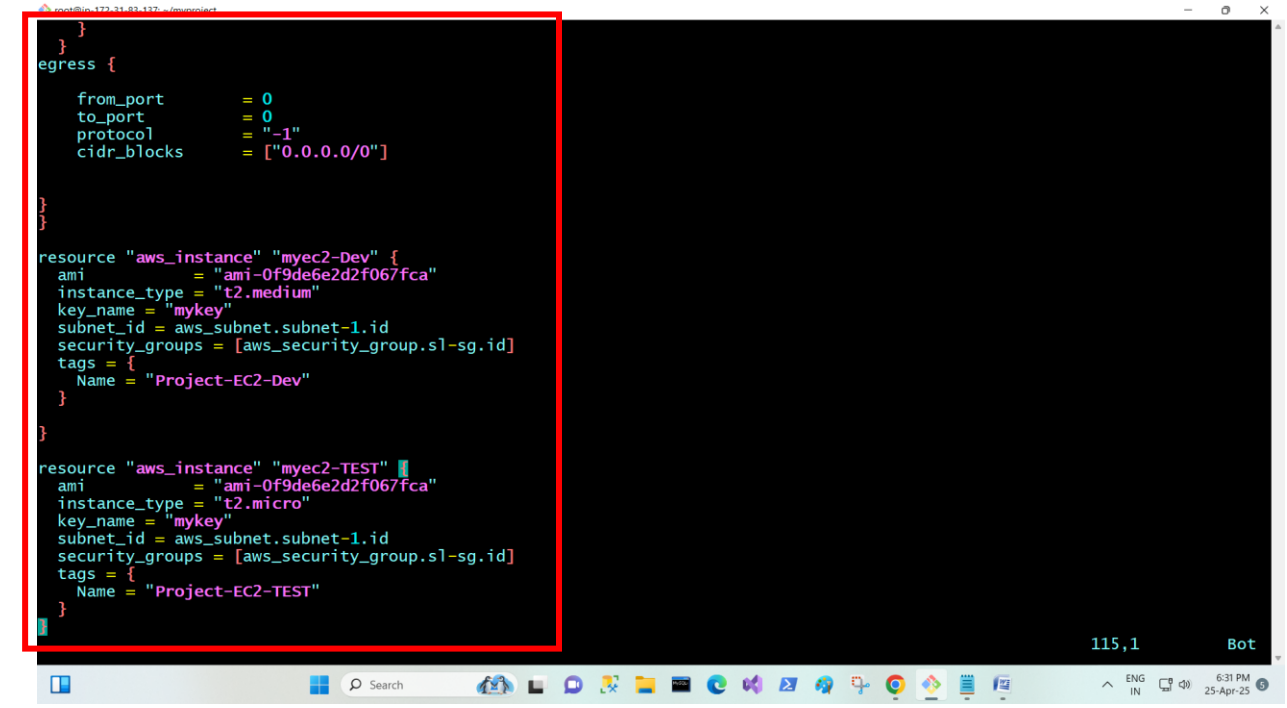
```
resource "aws_route_table" "sl-route-table"{
vpc_id = aws_vpc.sl-vpc.id
  tags = {
   Name = "sl-route-table"
}

}

resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.subnet-1.id
  route_table_id = aws_route_table.sl-route-table.id
}


resource "aws_internet_gateway" "gw" {
 vpc_id = aws_vpc.sl-vpc.id
 depends_on = [aws_vpc.sl-vpc]
   tags = {
   Name = "sl-gw"
}

}

resource "aws_route" "sl-route" {

route_table_id = aws_route_table.sl-route-table.id
destination_cidr_block = "0.0.0.0/0"
gateway_id = aws_internet_gateway.gw.id


}

variable "sg_ports" {
```

57,4 33%



```
}

variable "sg_ports" {
type = list(number)
default = [8080,80,22,443]

}



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"]


}
}
```

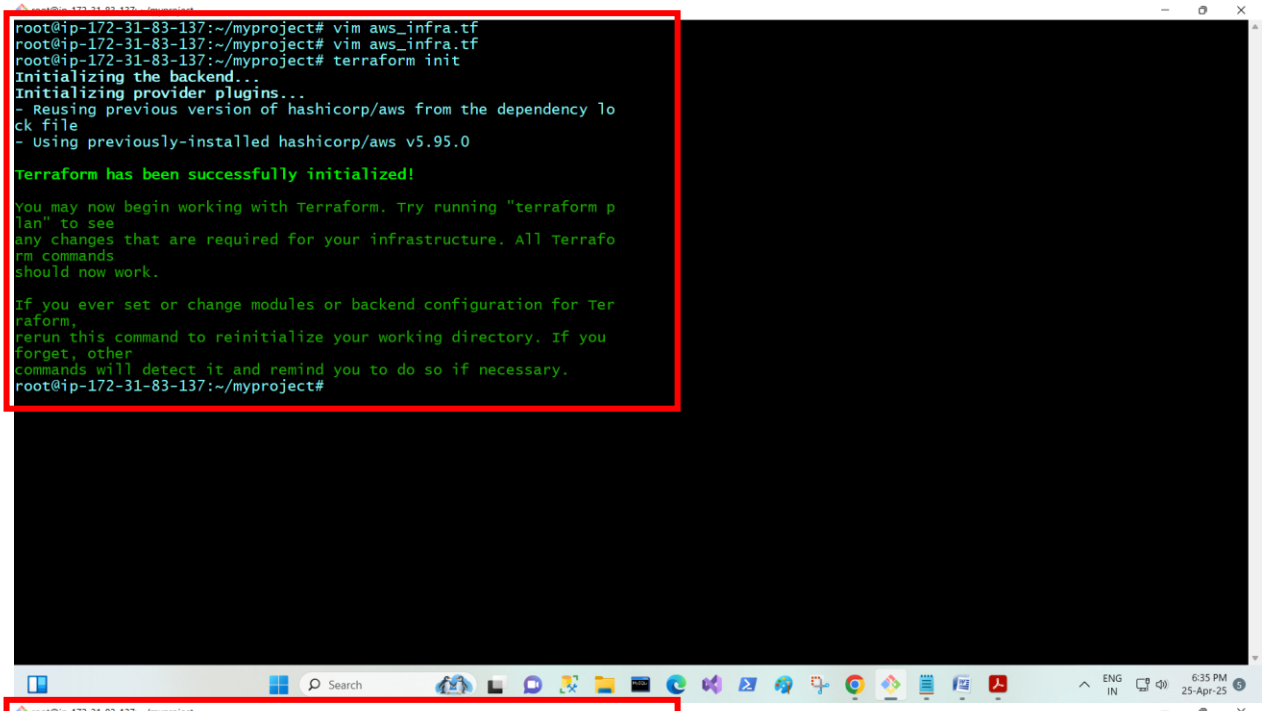
87,4 71%



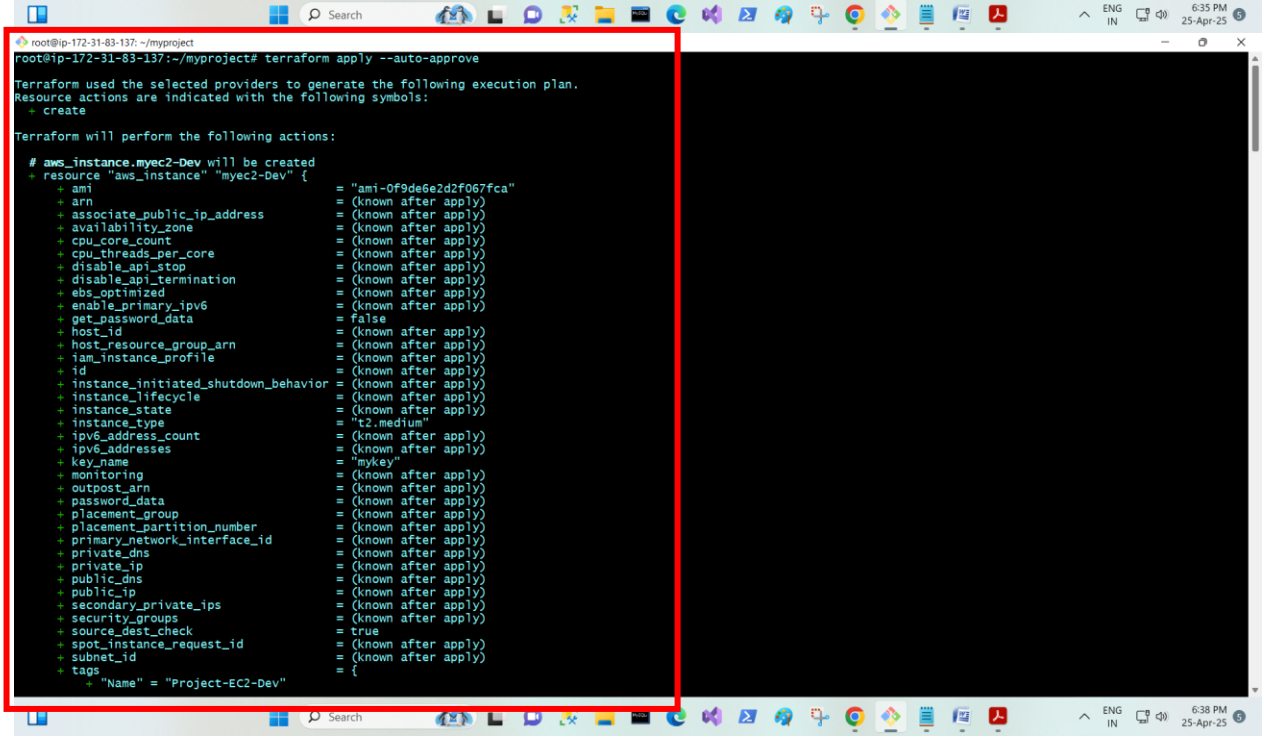
```
    }
  }
egress {

    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]


}
}

resource "aws_instance" "myec2-Dev" {
  ami           = "ami-0f9de6e2d2f067fca"
  instance_type = "t2.medium"
  key_name = "mykey"
  subnet_id = aws_subnet.subnet-1.id
  security_groups = [aws_security_group.sl-sg.id]
  tags = {
    Name = "Project-EC2-Dev"
  }

}

resource "aws_instance" "myec2-TEST" {
  ami           = "ami-0f9de6e2d2f067fca"
  instance_type = "t2.micro"
  key_name = "mykey"
  subnet_id = aws_subnet.subnet-1.id
  security_groups = [aws_security_group.sl-sg.id]
  tags = {
    Name = "Project-EC2-TEST"
  }
}
```

115,1 Bot

Step4: Then Execute “terraform init” and “terraform apply --auto-approve” Command

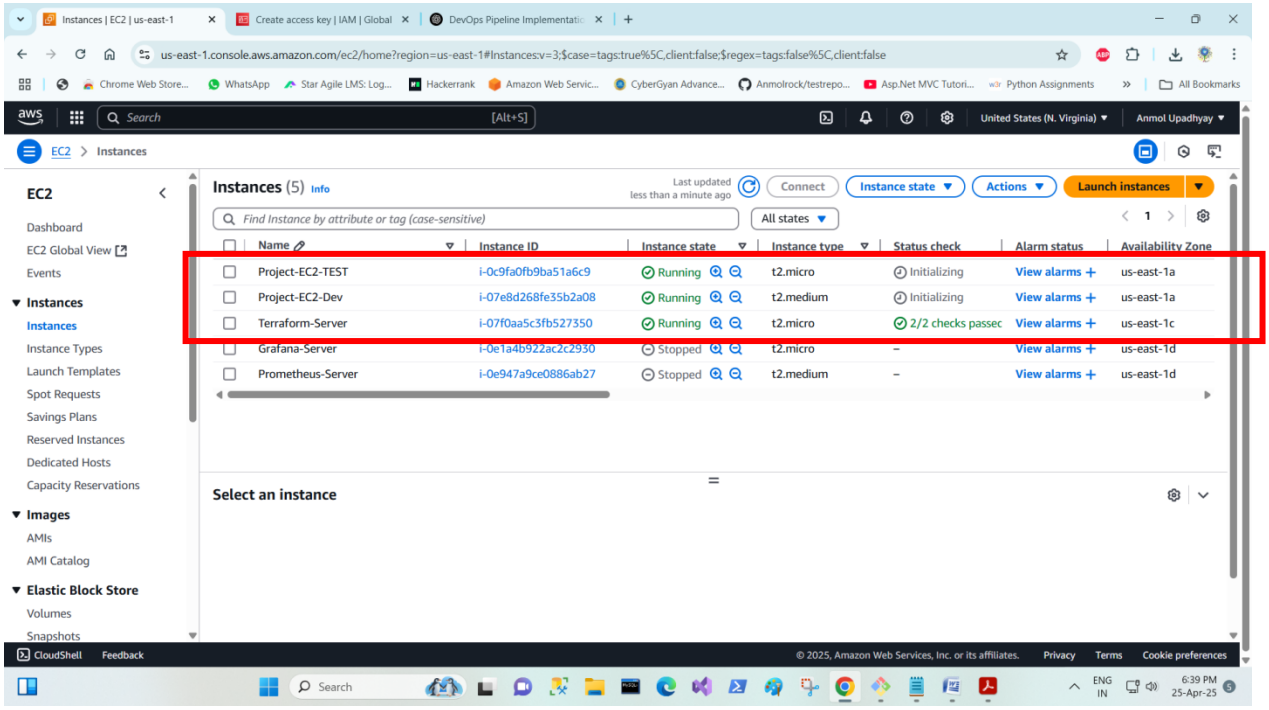


```
root@ip-172-31-83-137:~/myproject# vim aws_infra.tf
root@ip-172-31-83-137:~/myproject# vim aws_infra.tf
root@ip-172-31-83-137:~/myproject# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lo
ck file
- Using previously-installed hashicorp/aws v5.95.0

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Ter
raform,
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-172-31-83-137:~/myproject#
```

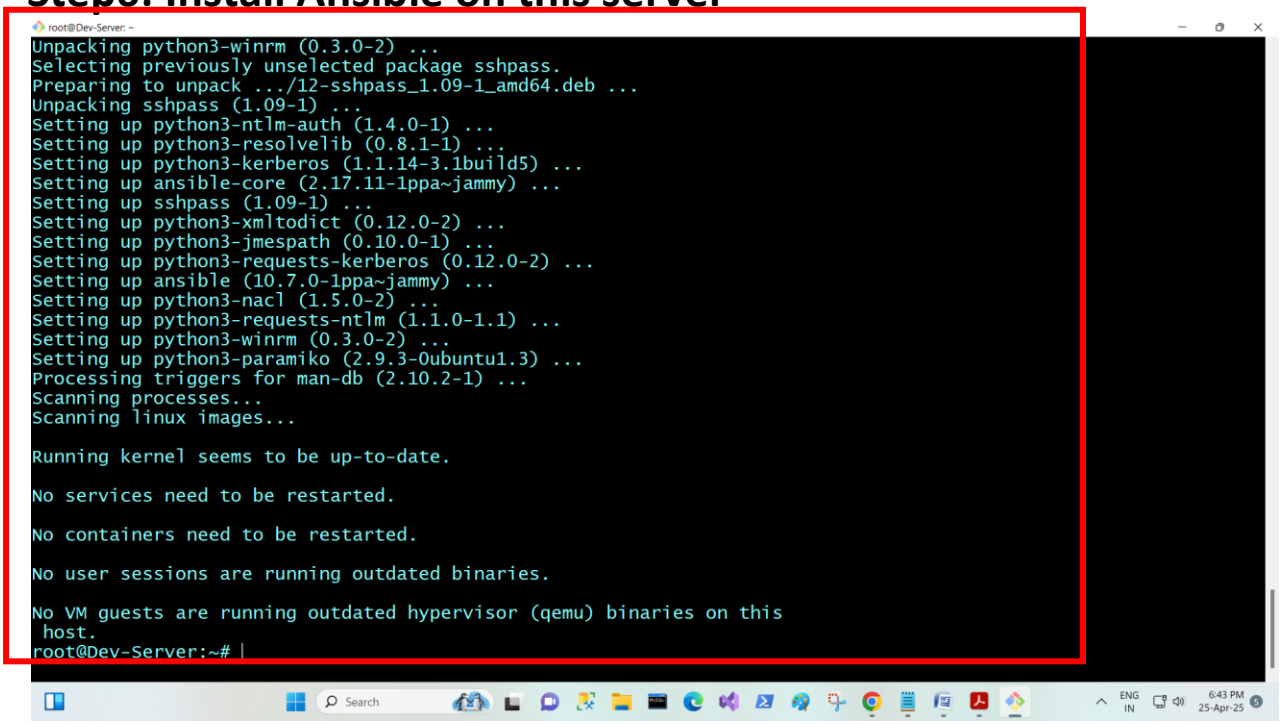


```
root@ip-172-31-83-137:~/myproject# terraform apply --auto-approve

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.myec2-Dev will be created
  + resource "aws_instance" "myec2-Dev" {
      + ami                                  = "ami-0f9de6e2d2f067fca"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.medium"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "mykey"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + spot_instance_request_id             = (known after apply)
      + subnet_id                            = (known after apply)
      + tags                                 = {
          + "Name" = "Project-EC2-Dev"
```

Step5: Now connect to the Dev Server which got created on AWS



Step6: Install Ansible on this server



```
root@Dev-Server:~# ansible --version
ansible [core 2.17.11]
  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 = 3.10.12 (main, Feb  4 2025, 14:57:36) [GCC 11.4.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
root@Dev-Server:~#
```

Step7: Run the playbook on the localhost

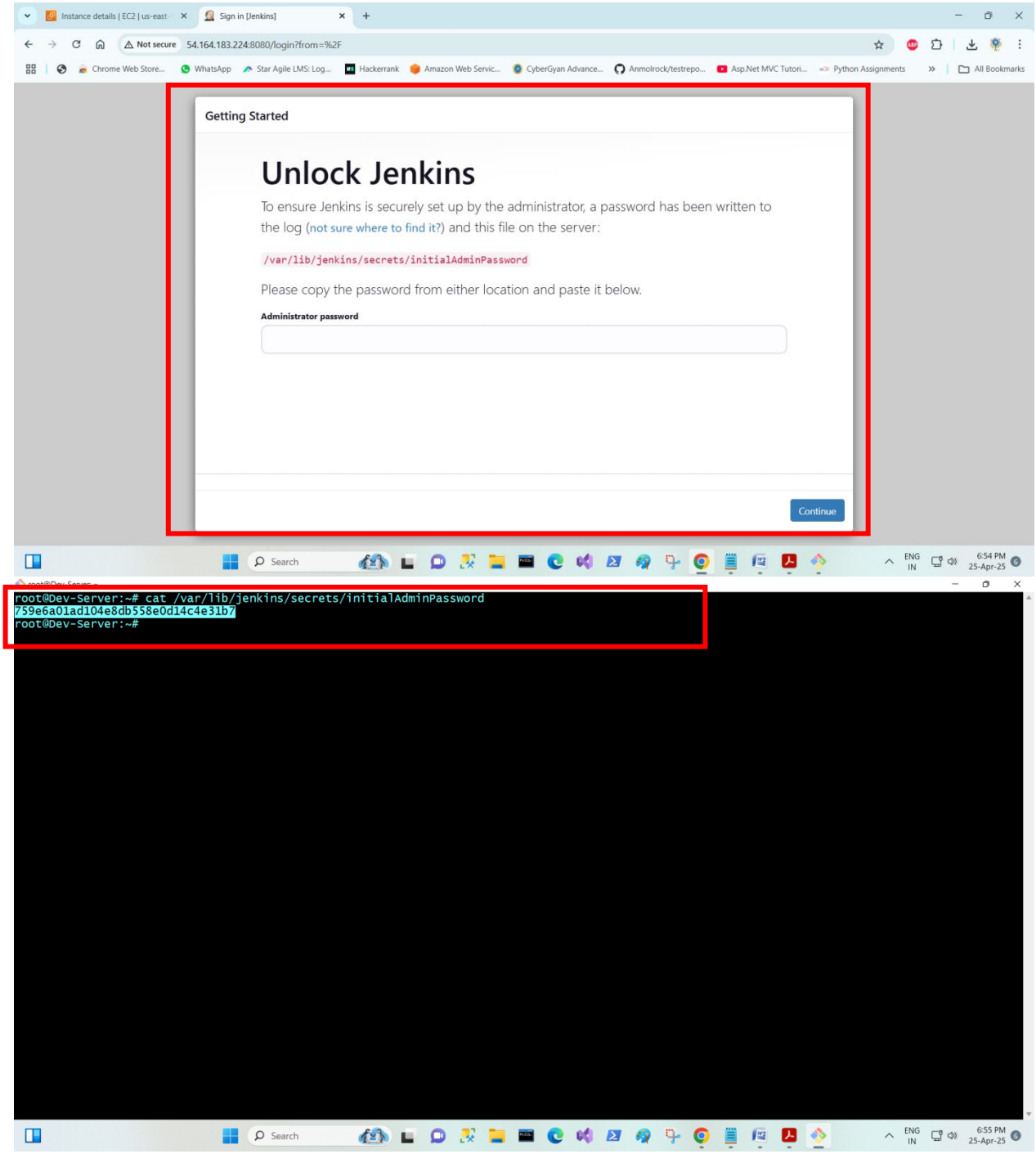
```
root@Dev-Server: ~
root@Dev-Server:~# vim playbook-install.yml
root@Dev-Server:~#
```

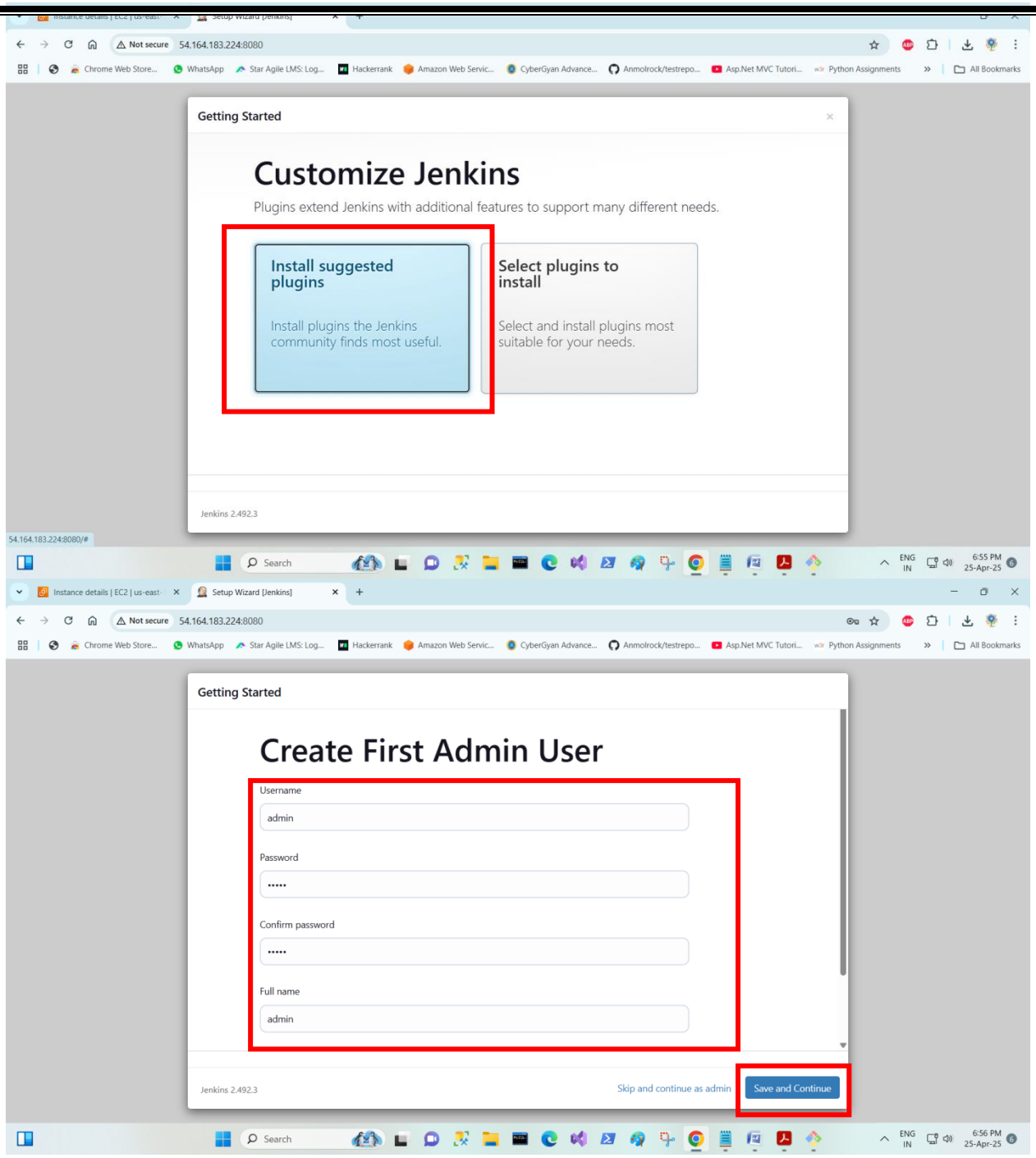
```yaml
- name: Install and set up devops tools
  hosts: localhost
  become: true
  tasks:
  - name: Update the apt repo
    command: apt-get update
  - name: Install multiple packages
    package: name={{item}} state=present
    loop:
     - git
     - docker.io
     - openjdk-17-jdk
  - name: install jenkins
    command: sudo apt-get install jenkins -y
  - name: start Jenkins and docker service
    service: name={{item}} state=started
    loop:
    - jenkins
    - docker
```

```
-- INSERT --                                             19,13         All
```

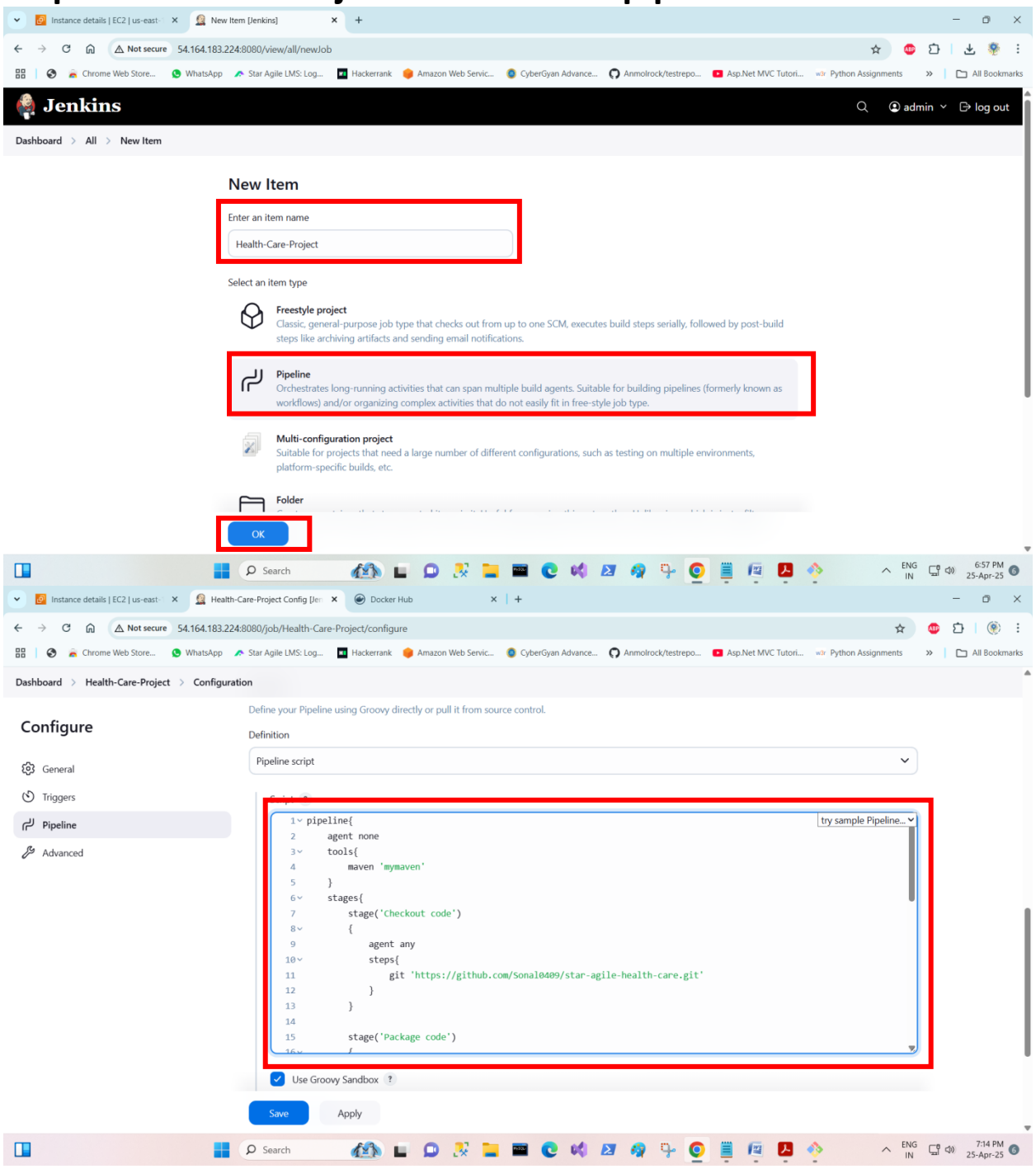
```
root@Dev-Server:~# vim playbook-install.yml
root@Dev-Server:~# ansible-playbook playbook-install.yml
[WARNING]: provided hosts list is empty, only localhost is availab
le. Note that
the implicit localhost does not match 'all'

PLAY [Install and set up devops tools] ***************************
**************

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

TASK [Update the apt repo] ***************************************
**************
changed: [localhost]

TASK [Install multiple packages] *******************************************************
ok: [localhost] => (item=git)
changed: [localhost] => (item=docker.io)
changed: [localhost] => (item=openjdk-17-jdk)

TASK [install jenkins] *****************************************************************
changed: [localhost]

TASK [start Jenkins and docker service] ************************************************
ok: [localhost] => (item=jenkins)
ok: [localhost] => (item=docker)

PLAY RECAP *****************************************************************************
localhost                  : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@Dev-Server:~#
```

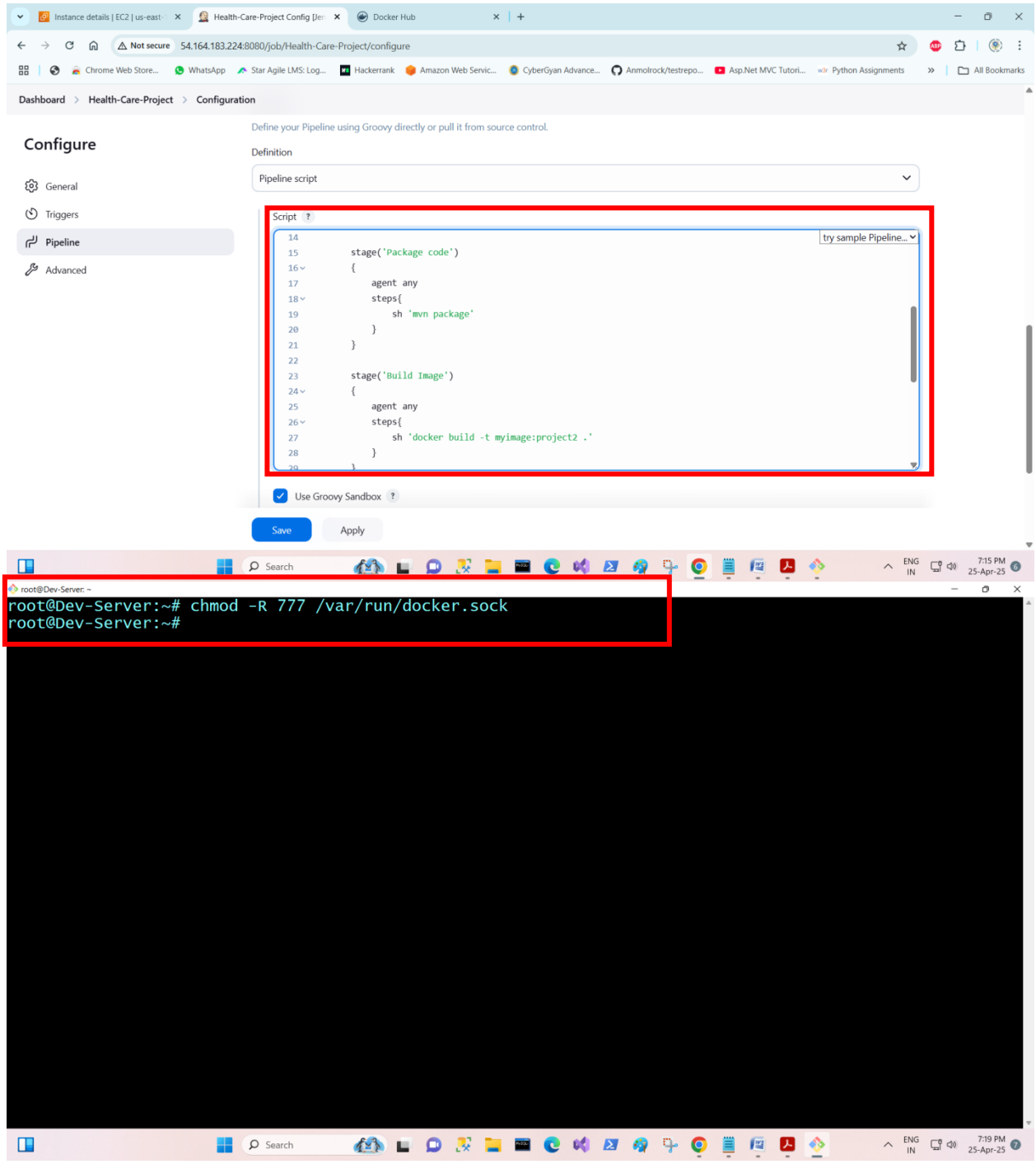
Step8: Now access the Jenkins Server -> Setup the Jenkins dashboard.

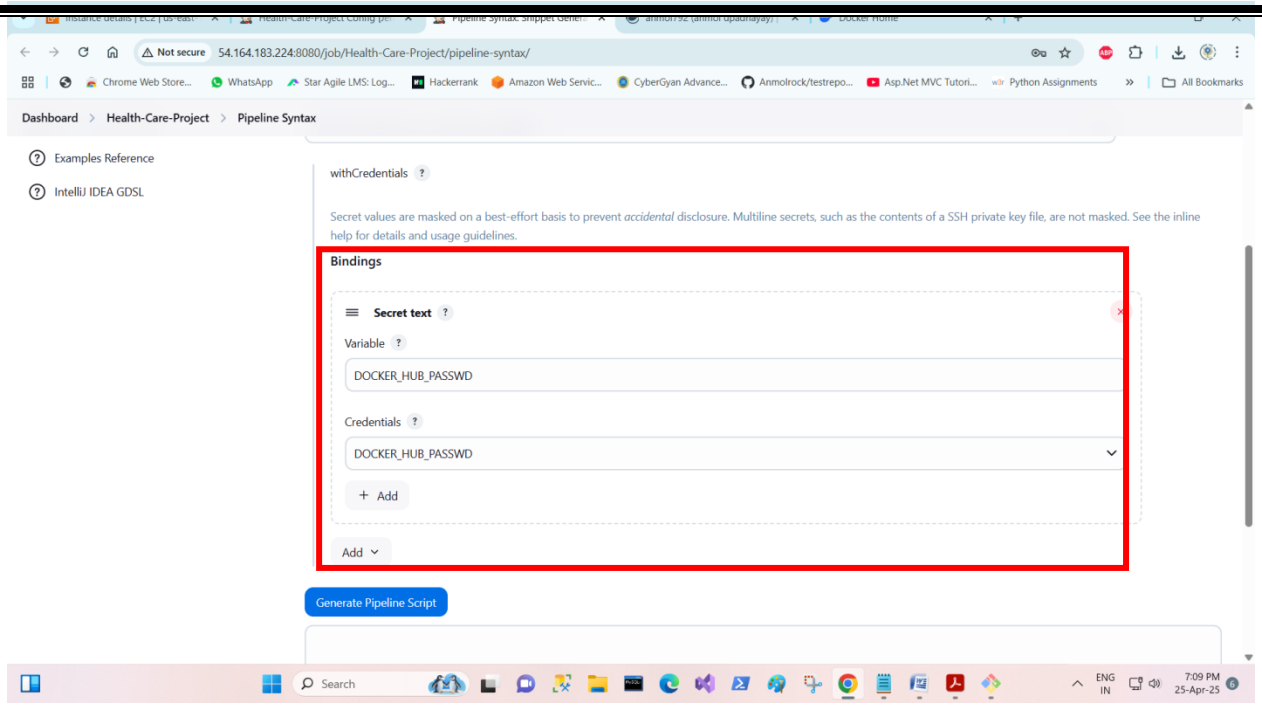


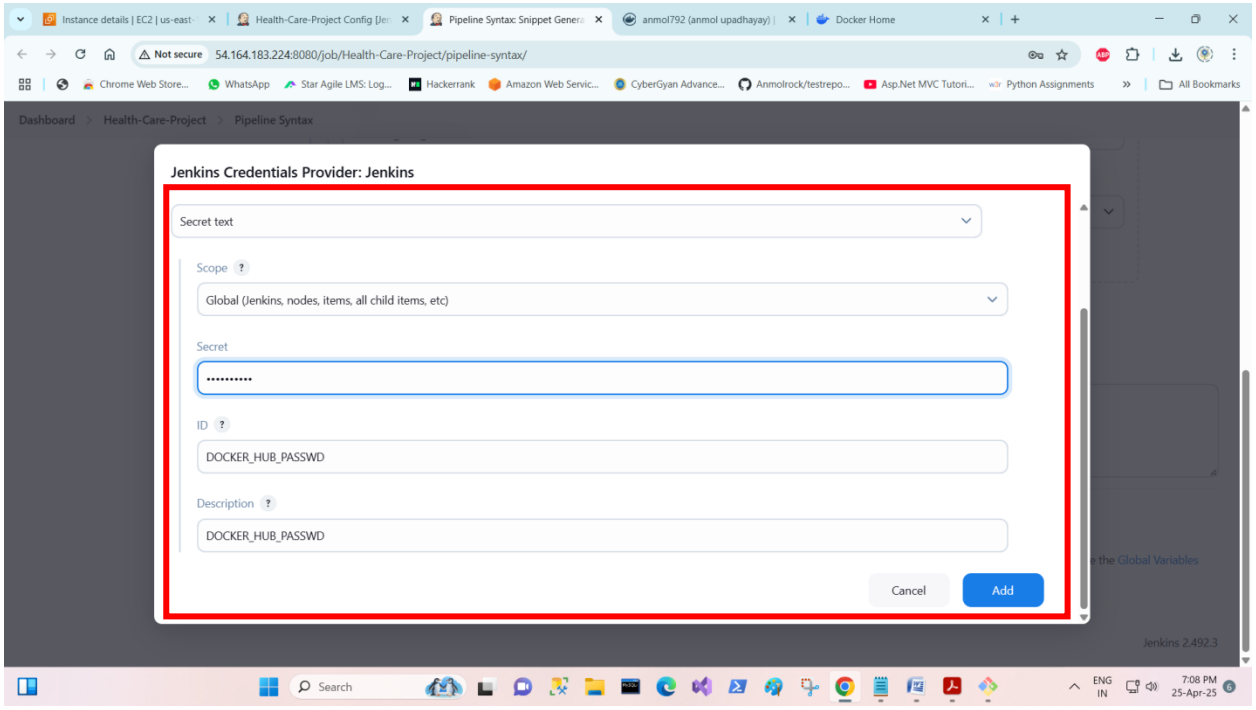

Not secure 54.164.183.224:8080
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.492.3
Not secure 54.164.183.224:8080
Getting Started
Create First Admin User
Username
admin
Password
Confirm password
Full name
admin
Jenkins 2.492.3
Skip and continue as admin
Save and Continue

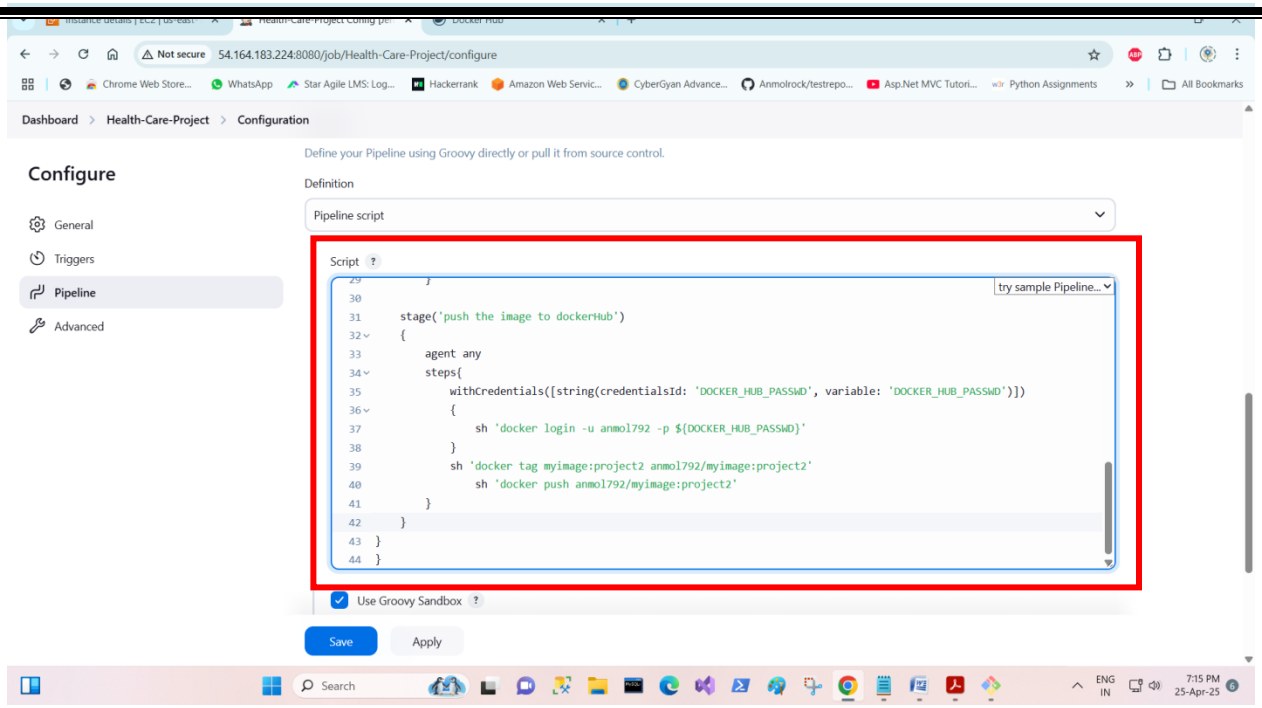
Step9: Create a new job and Create a pipeline



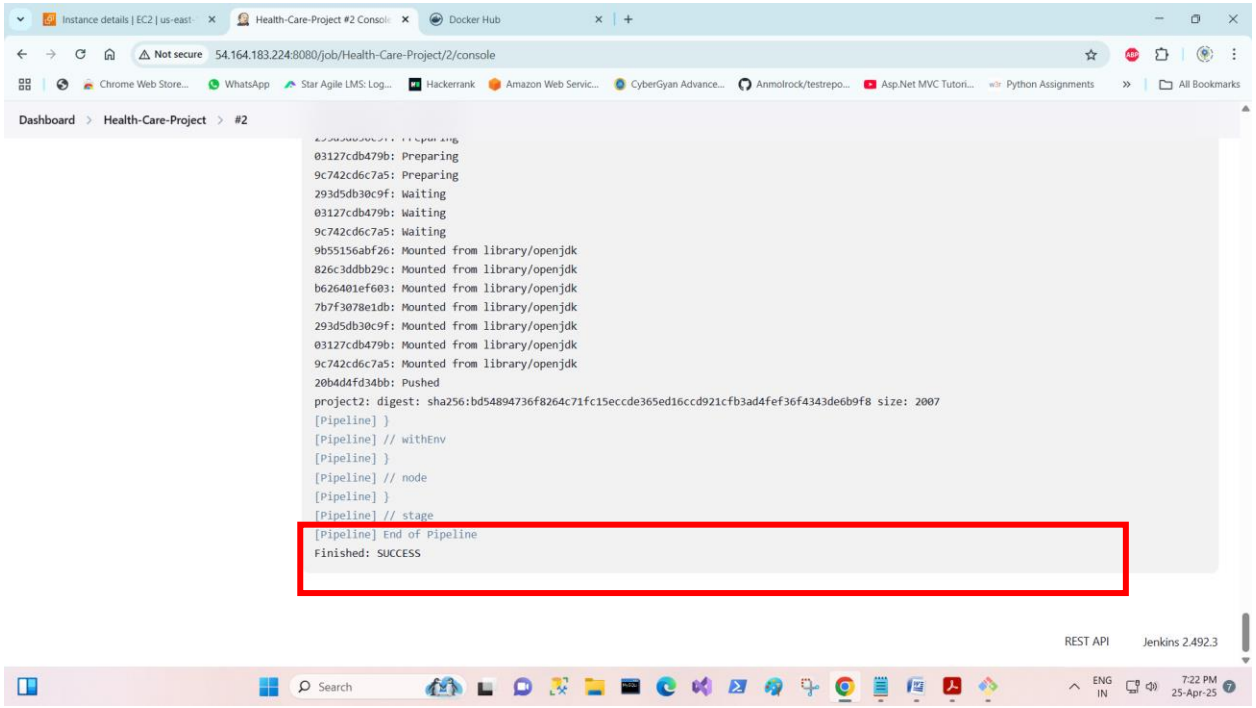

Instance details | EC2 | us-east-
Health-Care-Project Config [Jer
Docker Hub
Not secure
54.164.183.224:8080/job/Health-Care-Project/configure
Chrome Web Store...
WhatsApp
Star Agile LMS: Log...
Hackerrank
Amazon Web Servic...
CyberGyan Advance...
Anmolrock/testrepo...
Asp.Net MVC Tutori...
Python Assignments
All Bookmarks
Dashboard
Health-Care-Project
Configuration
Configure
General
Triggers
Pipeline
Advanced
Define your Pipeline using Groovy directly or pull it from source control.
Definition
Pipeline script
Script
try sample Pipeline...
14
15 stage('Package code')
16 {
17 agent any
18 steps{
19 sh 'mvn package'
20 }
21 }
22
23 stage('Build Image')
24 {
25 agent any
26 steps{
27 sh 'docker build -t myimage:project2 .'
28 }
29 }
Use Groovy Sandbox
Save
Apply
Search
ENG
IN
7:15 PM
25-Apr-25
root@Dev-Server: ~
root@Dev-Server:~# chmod -R 777 /var/run/docker.sock
root@Dev-Server:~#
Search
ENG
IN
7:19 PM
25-Apr-25

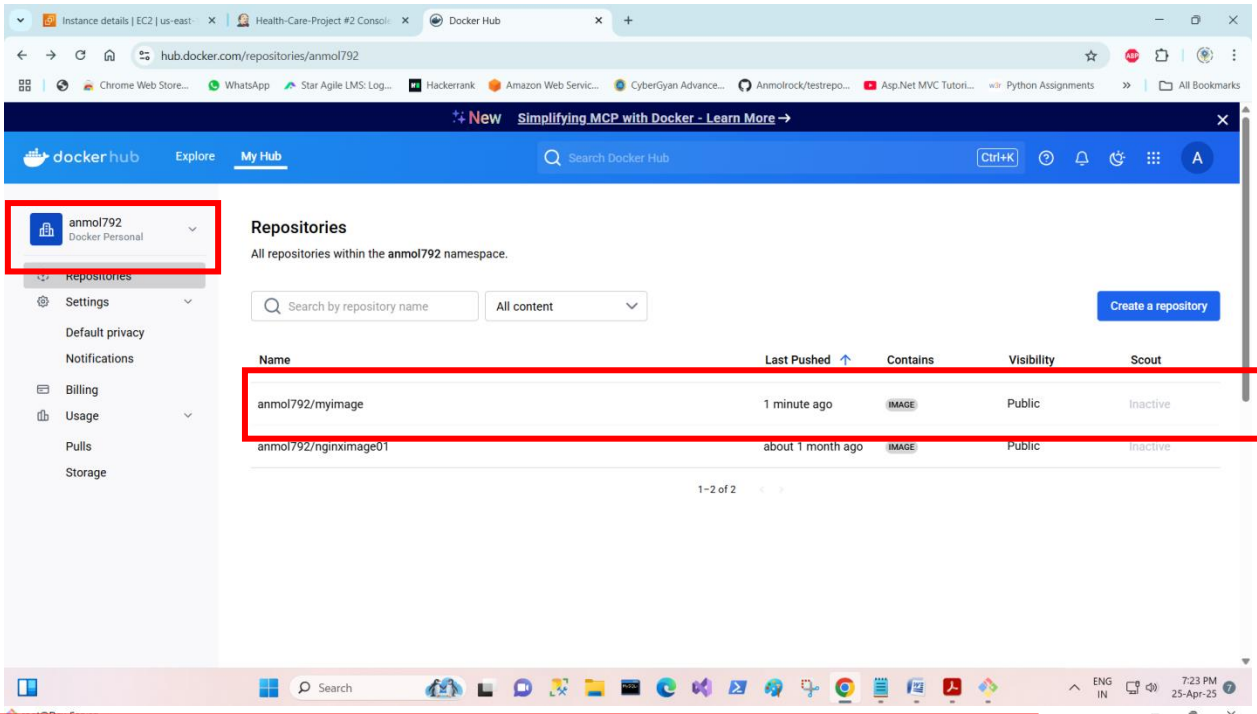

Not secure 54.164.183.224:8080/job/Health-Care-Project/pipeline-syntax/
Dashboard > Health-Care-Project > Pipeline Syntax
Examples Reference
IntelliJ IDEA GDSL
withCredentials
Secret values are masked on a best-effort basis to prevent accidental disclosure. Multiline secrets, such as the contents of a SSH private key file, are not masked. See the inline help for details and usage guidelines.
Bindings
Secret text
Variable
DOCKER_HUB_PASSWD
Credentials
DOCKER_HUB_PASSWD
+ Add
Add
Generate Pipeline Script

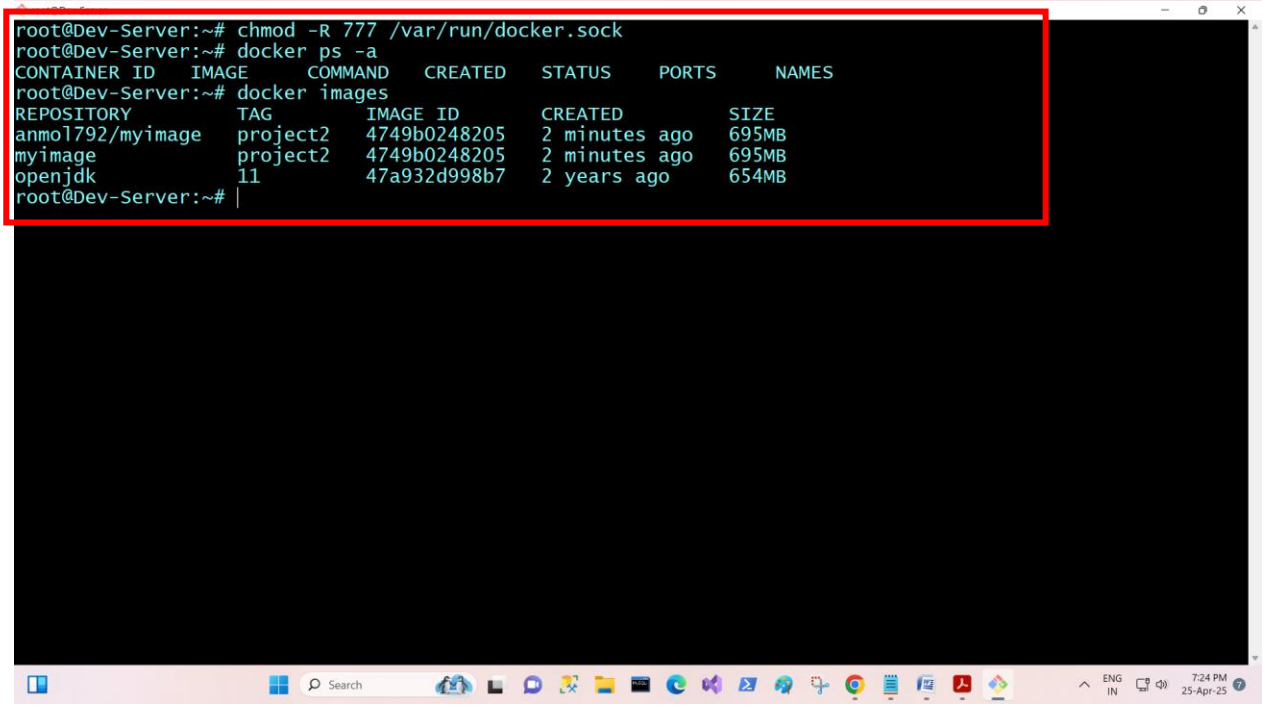

Not secure 54.164.183.224:8080/job/Health-Care-Project/pipeline-syntax/
Jenkins Credentials Provider: Jenkins
Secret text
Scope
Global (Jenkins, nodes, items, all child items, etc)
Secret
ID
DOCKER_HUB_PASSWD
Description
DOCKER_HUB_PASSWD
Cancel
Add



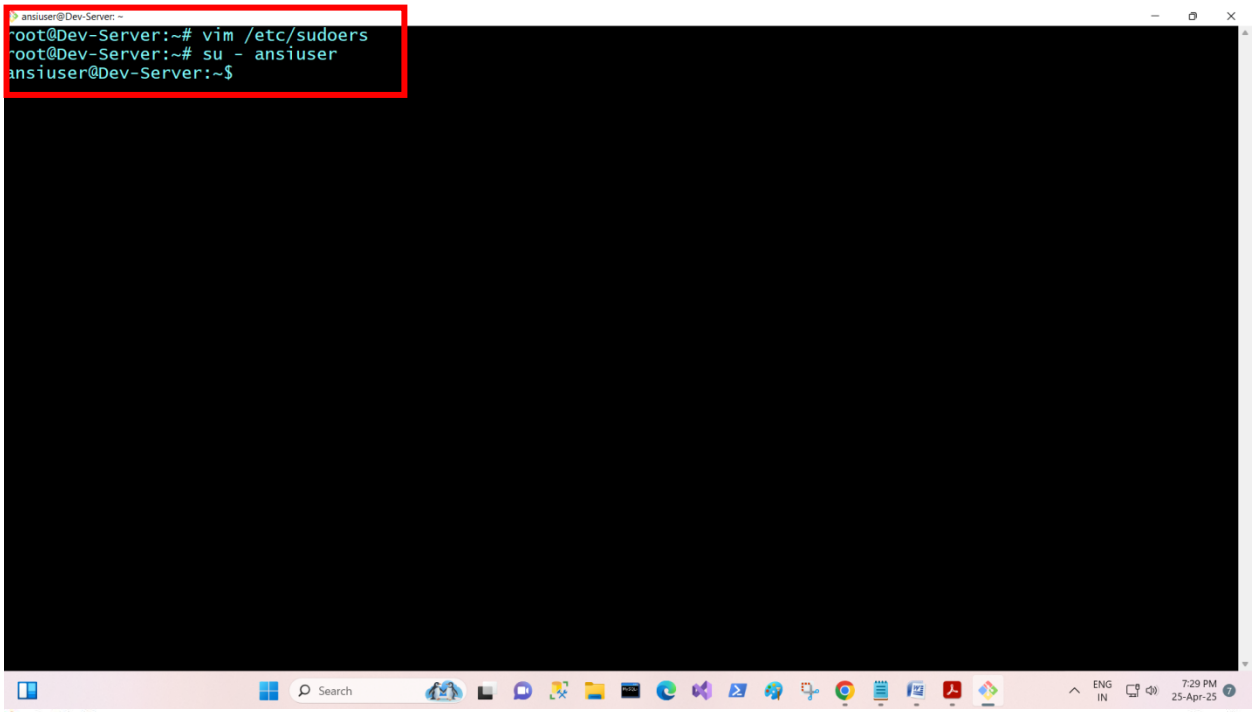
Step10: Save the job and run the pipeline

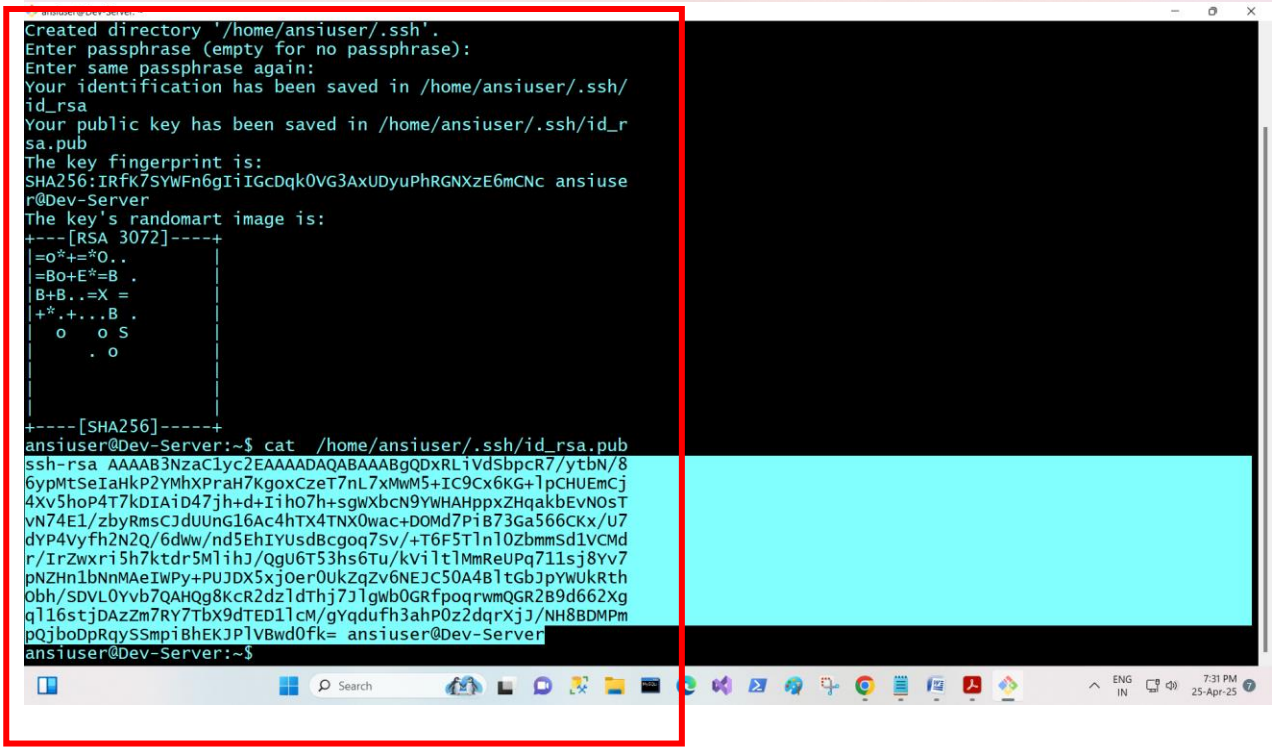


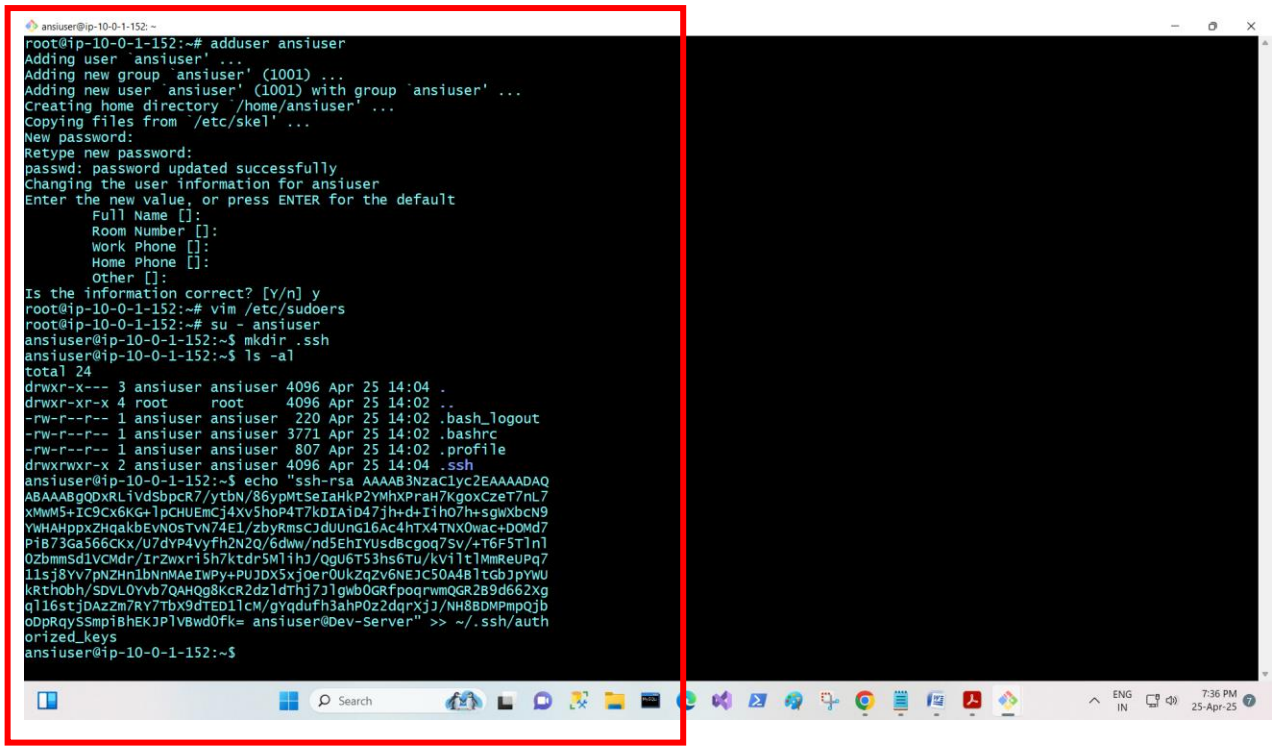

hub.docker.com/repositories/anmol792
New Simplifying MCP with Docker - Learn More →
dockerhub
Explore
My Hub
Search Docker Hub
Ctrl+K
anmol792
Docker Personal
Repositories
Settings
Default privacy
Notifications
Billing
Usage
Pulls
Storage
Repositories
All repositories within the anmol792 namespace.
Search by repository name
All content
Create a repository
Name
Last Pushed
Contains
Visibility
Scout
anmol792/myimage
1 minute ago
IMAGE
Public
Inactive
anmol792/nginximage01
about 1 month ago
IMAGE
Public
Inactive
1–2 of 2


root@Dev-Server:~# chmod -R 777 /var/run/docker.sock
root@Dev-Server:~# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@Dev-Server:~# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
anmol792/myimage project2 4749b0248205 2 minutes ago 695MB
myimage project2 4749b0248205 2 minutes ago 695MB
openjdk 11 47a932d998b7 2 years ago 654MB
root@Dev-Server:~#

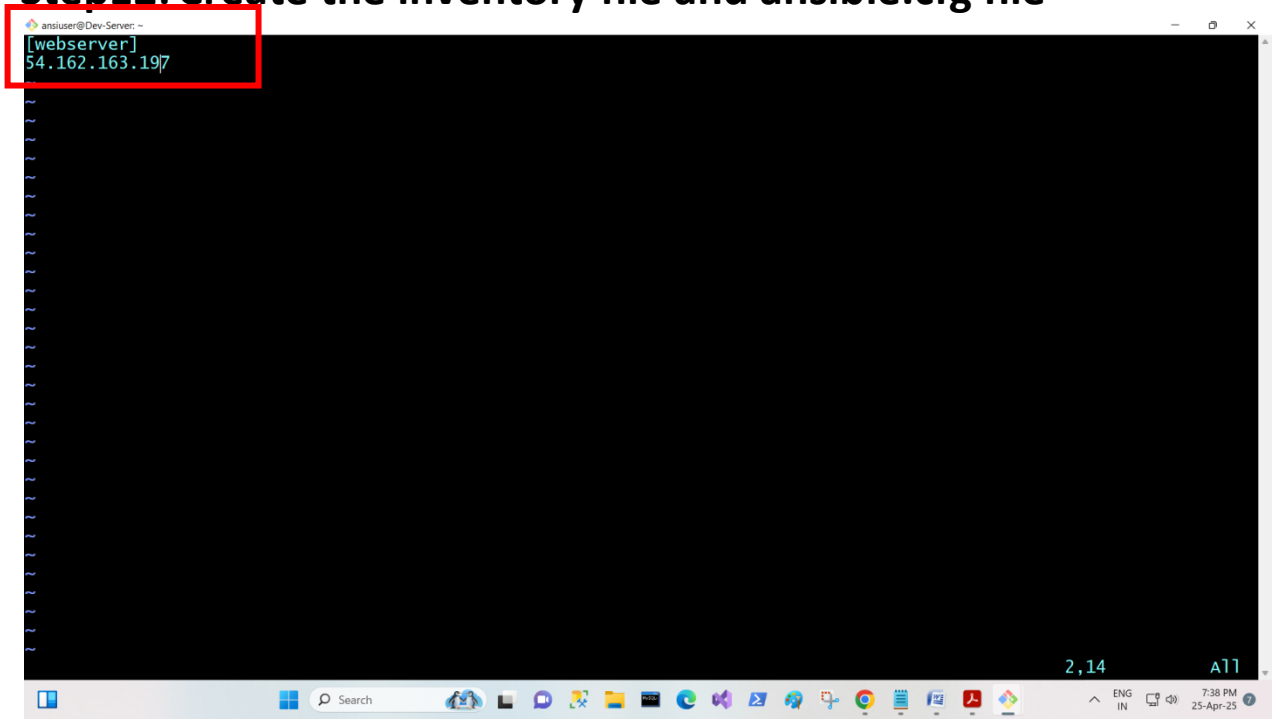
Step11: Go to Ansible Controller --> copy ssh keys of root user on the test server

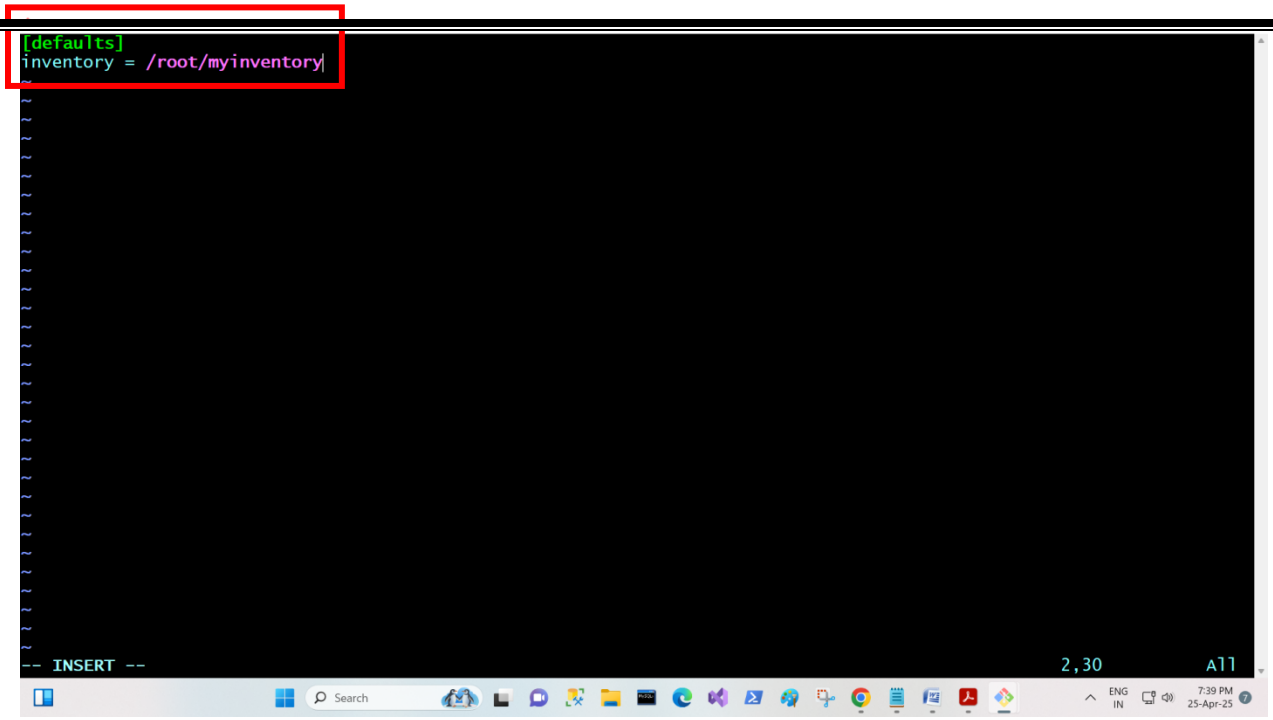


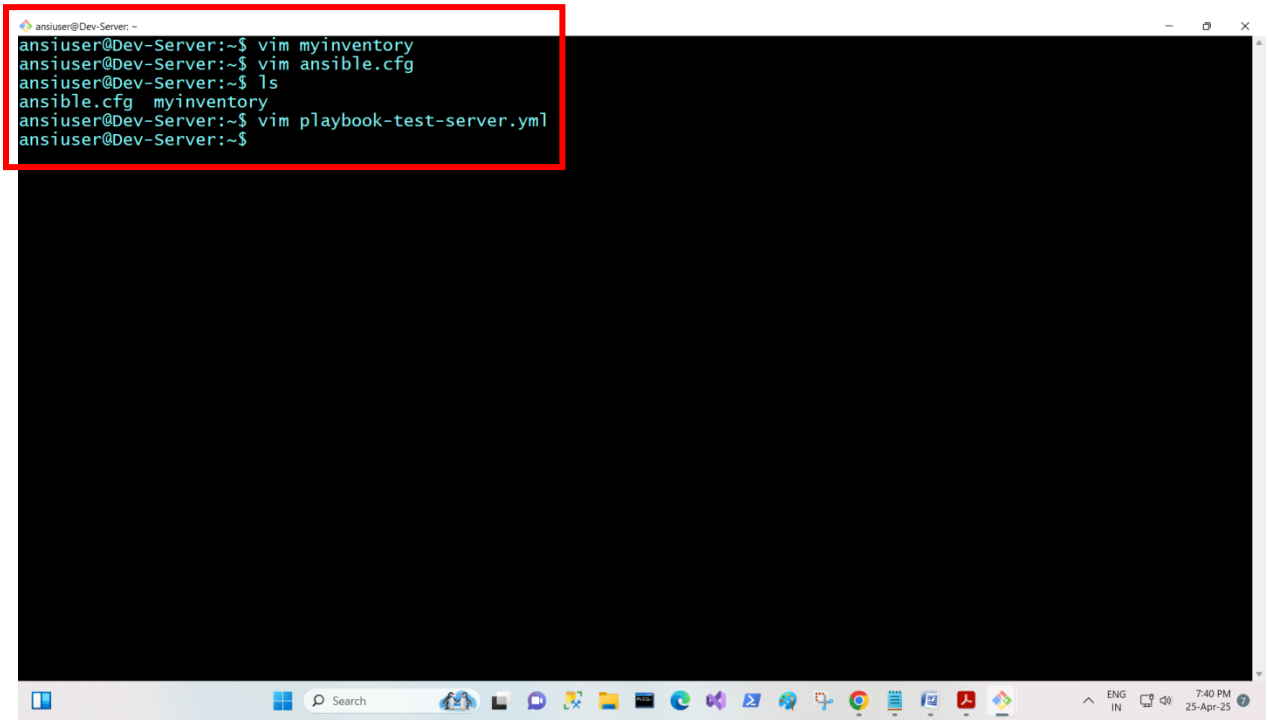




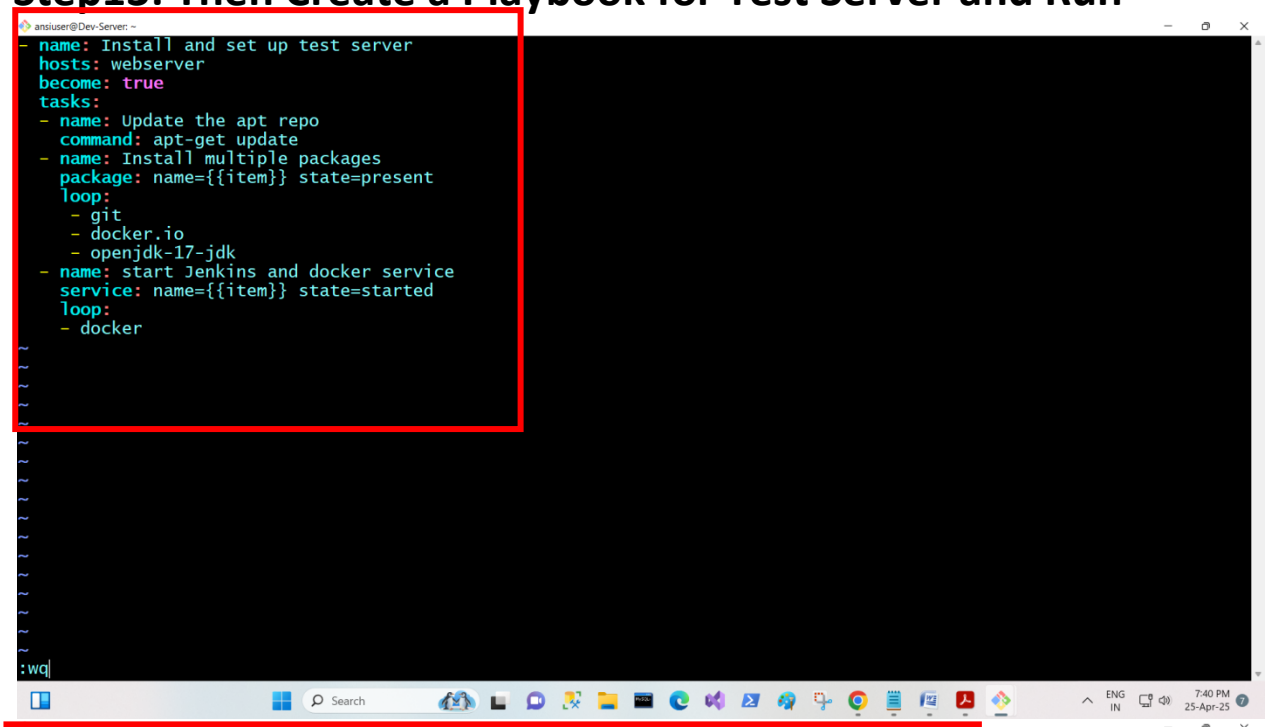
Step12: Create the inventory file and ansible.cfg file



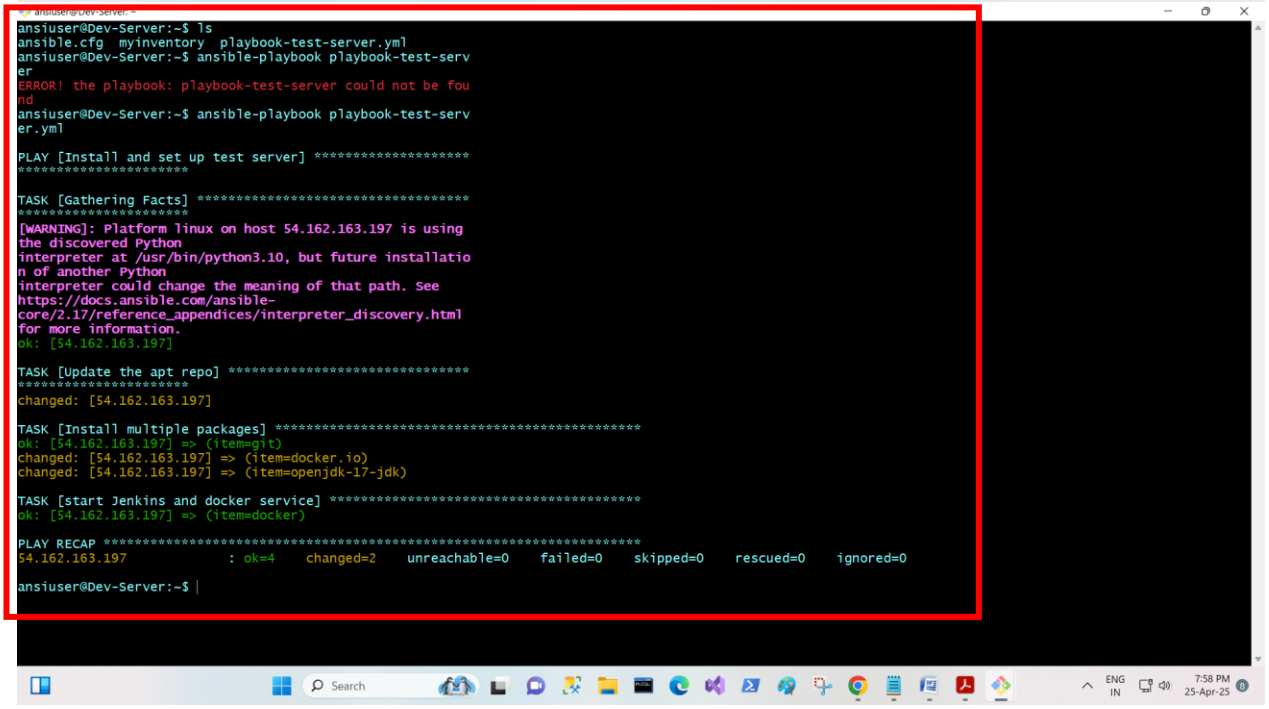

[defaults]
inventory = /root/myinventory
-- INSERT --
2,30
All


ansiuser@Dev-Server: ~
ansiuser@Dev-Server:~$ vim myinventory
ansiuser@Dev-Server:~$ vim ansible.cfg
ansiuser@Dev-Server:~$ ls
ansible.cfg myinventory
ansiuser@Dev-Server:~$ vim playbook-test-server.yml
ansiuser@Dev-Server:~$

Step13: Then Create a Playbook for Test Server and Run

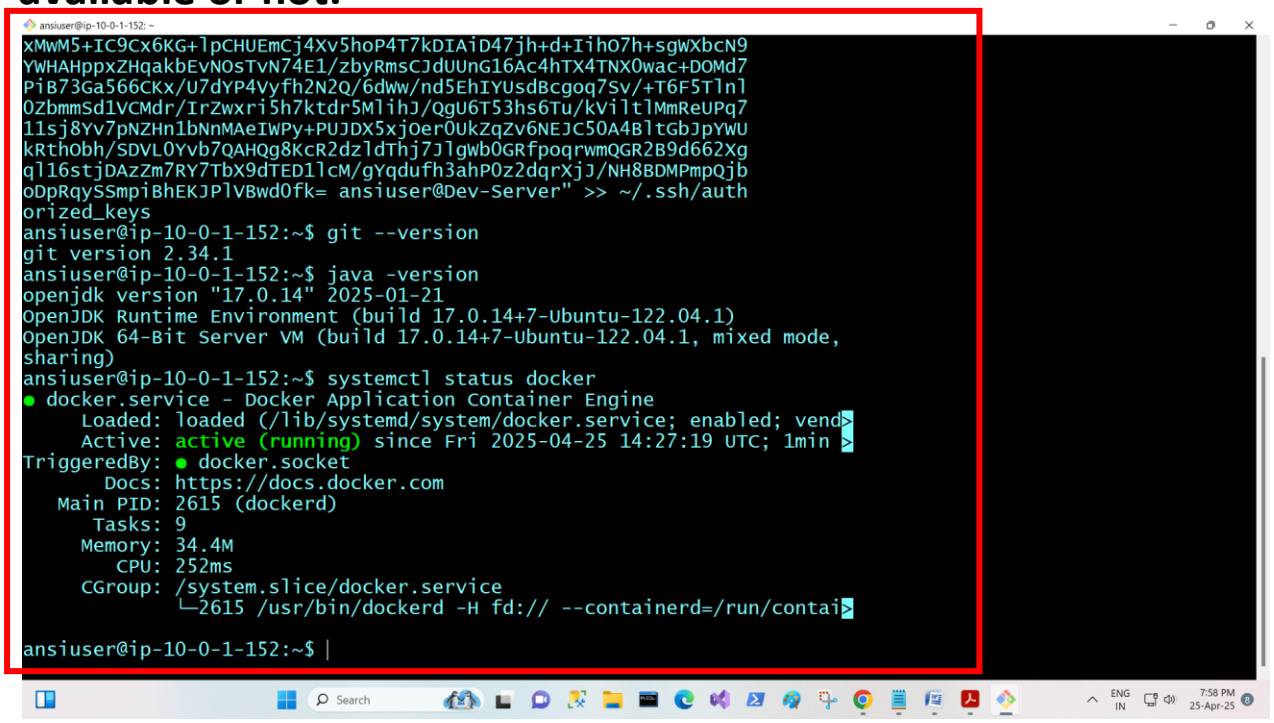


```
- name: Install and set up test server
  hosts: webserver
  become: true
  tasks:
  - name: Update the apt repo
    command: apt-get update
  - name: Install multiple packages
    package: name={{item}} state=present
    loop:
     - git
     - docker.io
     - openjdk-17-jdk
  - name: start Jenkins and docker service
    service: name={{item}} state=started
    loop:
    - docker
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

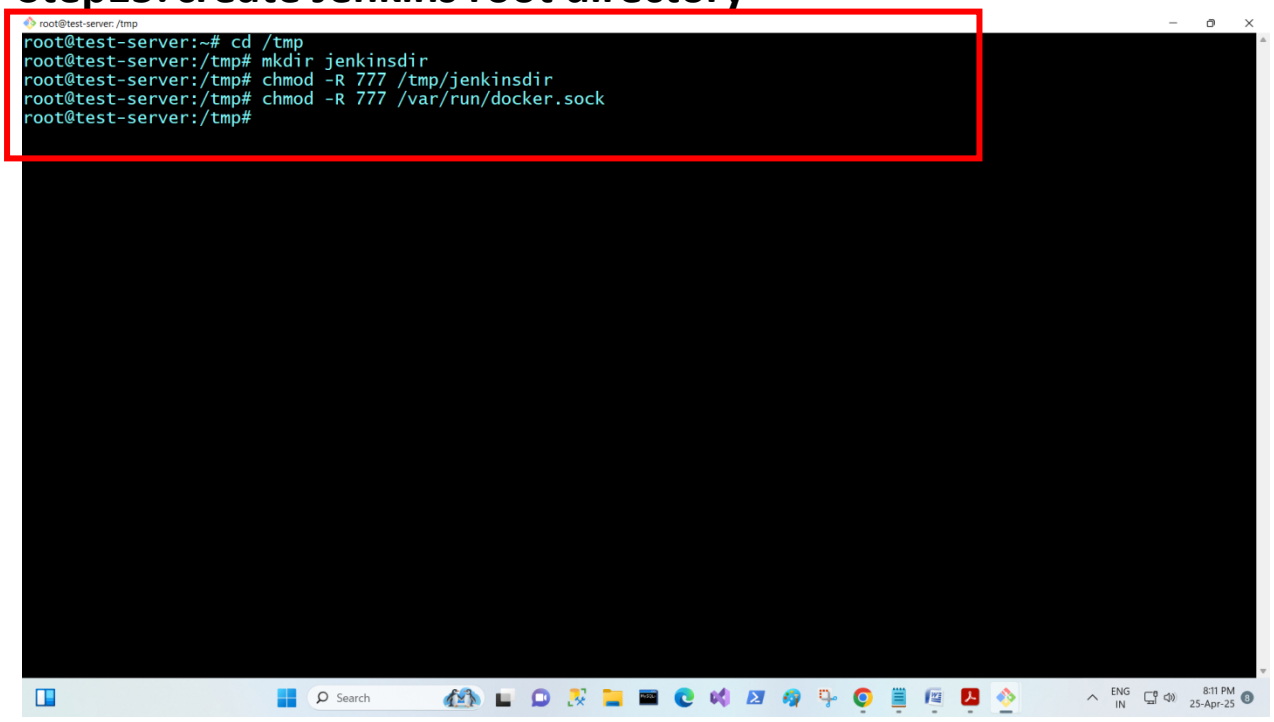


```
ansiuser@Dev-Server:~$ ls
ansible.cfg  myinventory  playbook-test-server.yml
ansiuser@Dev-Server:~$ ansible-playbook playbook-test-serv
er
ERROR! the playbook: playbook-test-server could not be fou
nd
ansiuser@Dev-Server:~$ ansible-playbook playbook-test-serv
er.yml

PLAY [Install and set up test server] ********************
**********************

TASK [Gathering Facts] ***********************************
**********************
[WARNING]: Platform linux on host 54.162.163.197 is using
the discovered Python
interpreter at /usr/bin/python3.10, but future installatio
n of another Python
interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html
for more information.
ok: [54.162.163.197]

TASK [Update the apt repo] *******************************
**********************
changed: [54.162.163.197]

TASK [Install multiple packages] *************************************************
ok: [54.162.163.197] => (item=git)
changed: [54.162.163.197] => (item=docker.io)
changed: [54.162.163.197] => (item=openjdk-17-jdk)

TASK [start Jenkins and docker service] ******************************************
ok: [54.162.163.197] => (item=docker)

PLAY RECAP ***********************************************************************
54.162.163.197             : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ansiuser@Dev-Server:~$
```

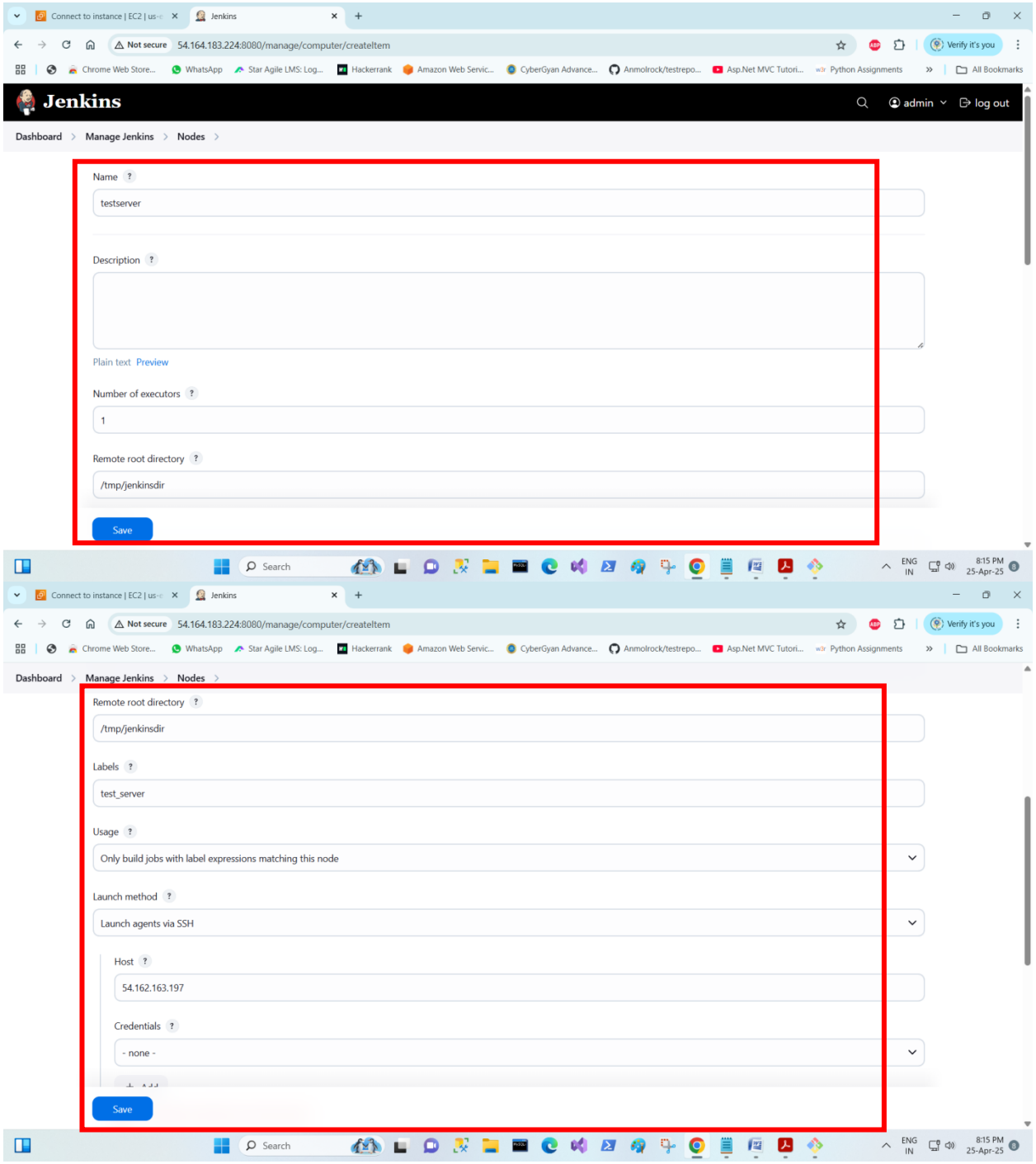
Step14: Go to the “Test Server” then check git, docker, Java is available or not.

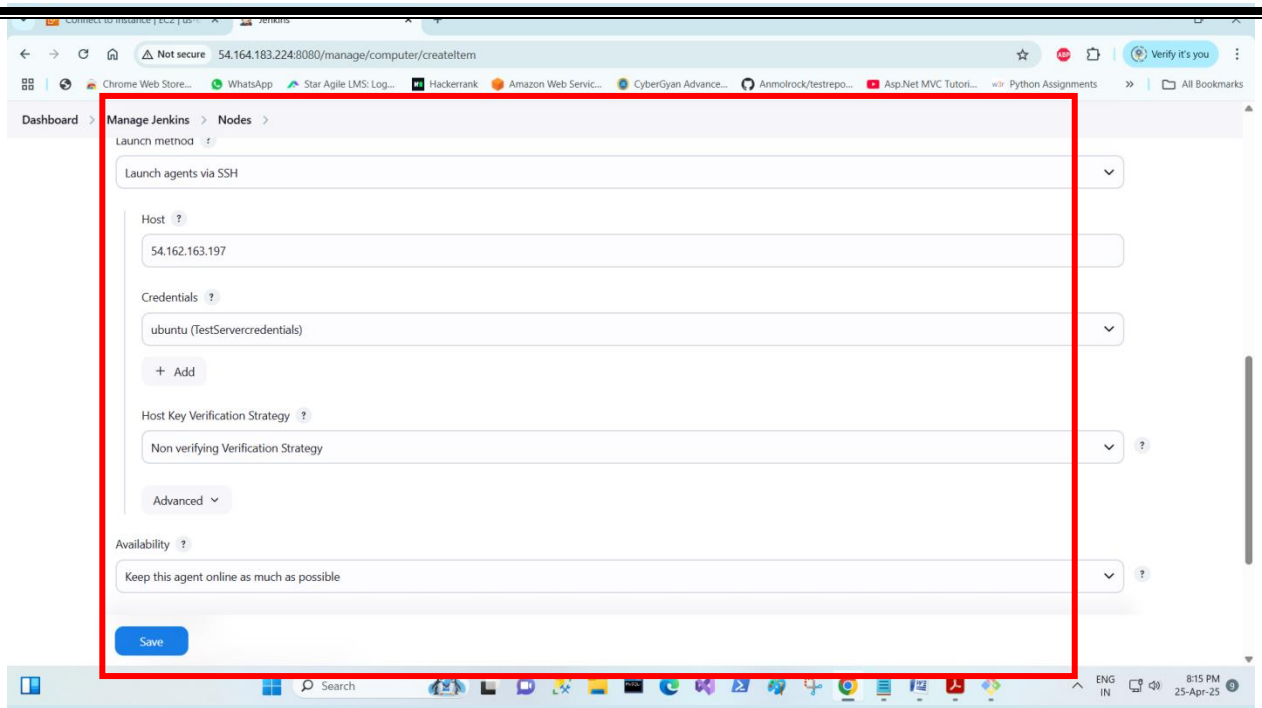


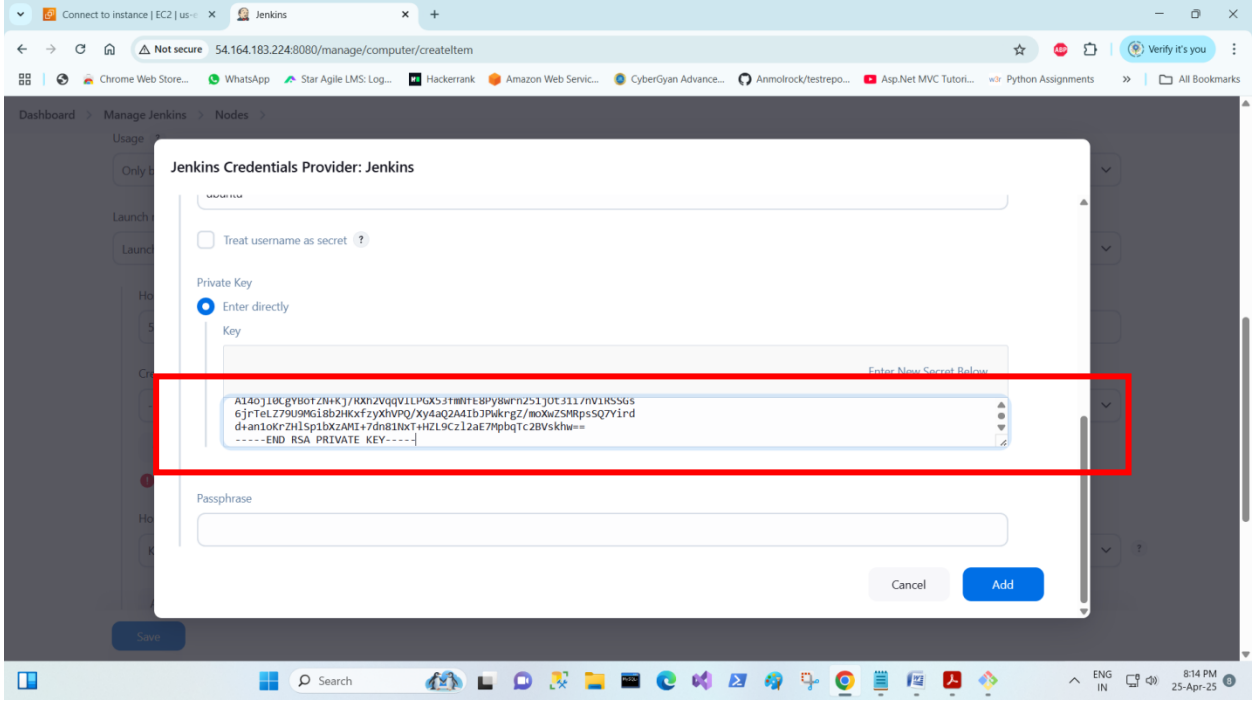
Step15: create Jenkins root directory

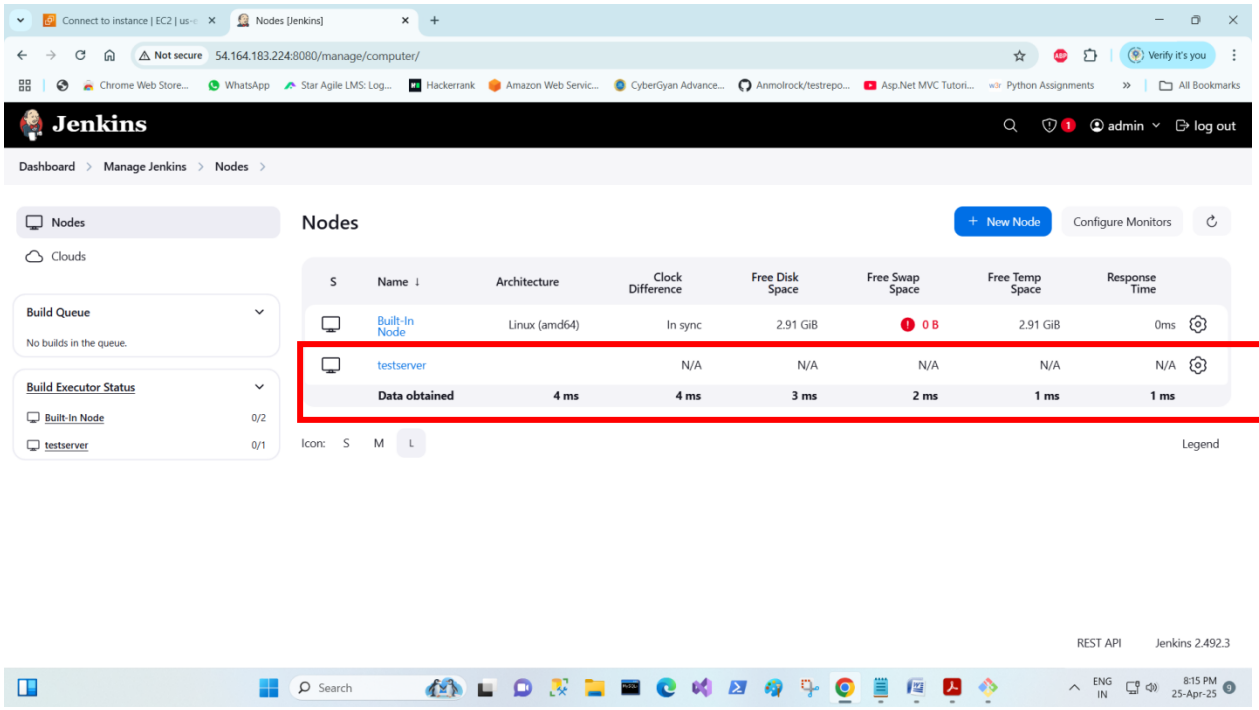


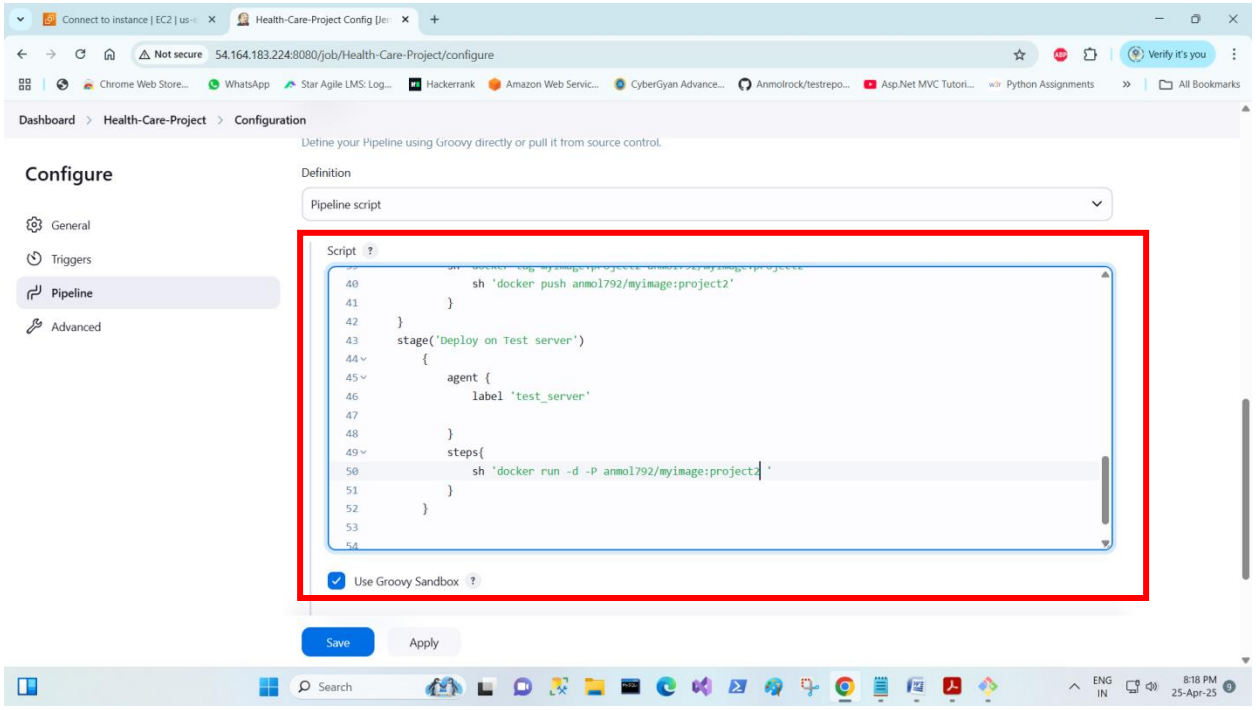
Step16: Connect test server to Jenkins as agent and Create a node on Jenkins server and continue the pipeline

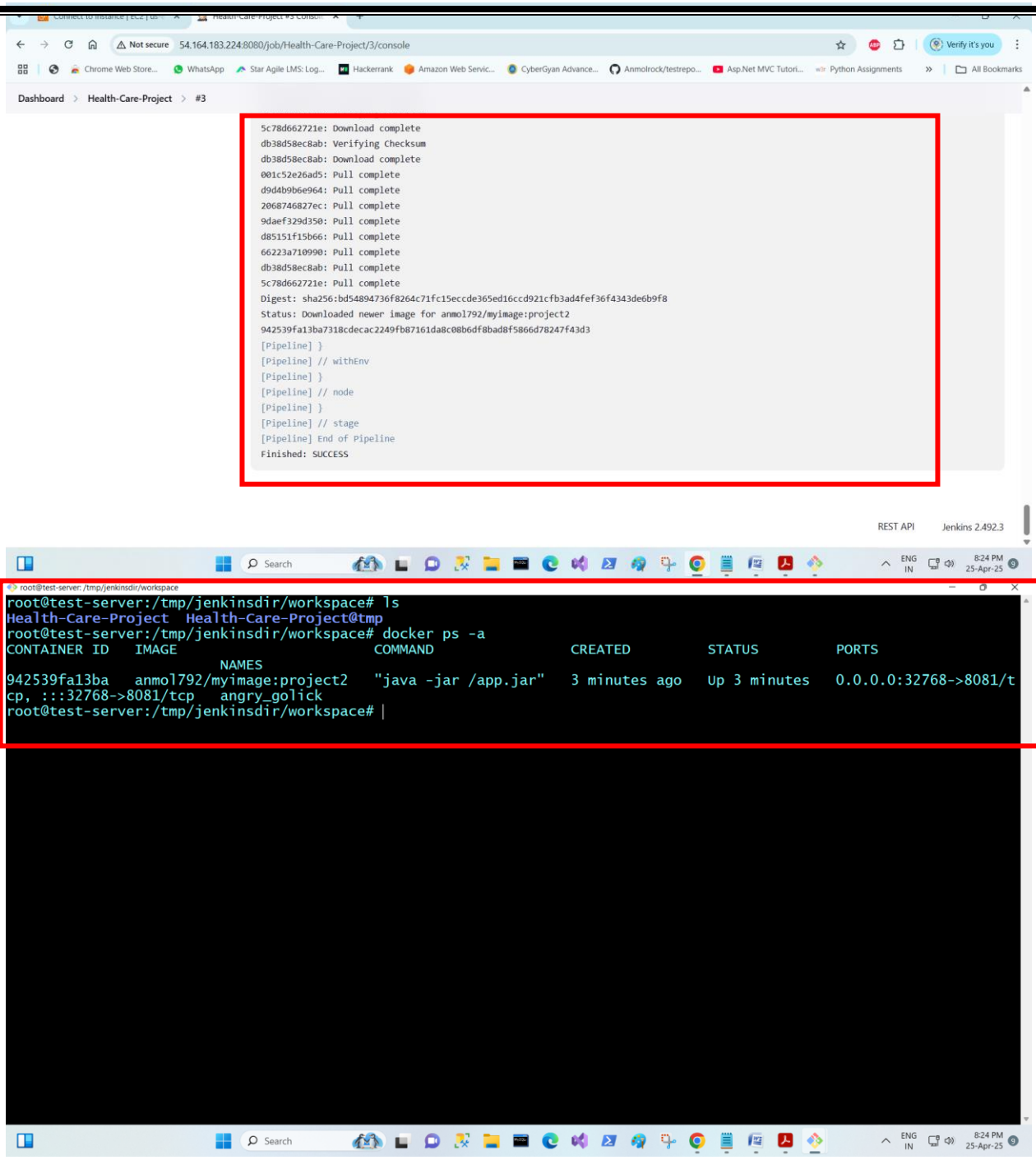



Not secure 54.164.183.224:8080/manage/computer/createItem
Dashboard > Manage Jenkins > Nodes >
Launch method
Launch agents via SSH
Host
54.162.163.197
Credentials
ubuntu (TestServercredentials)
+ Add
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced
Availability
Keep this agent online as much as possible
Save

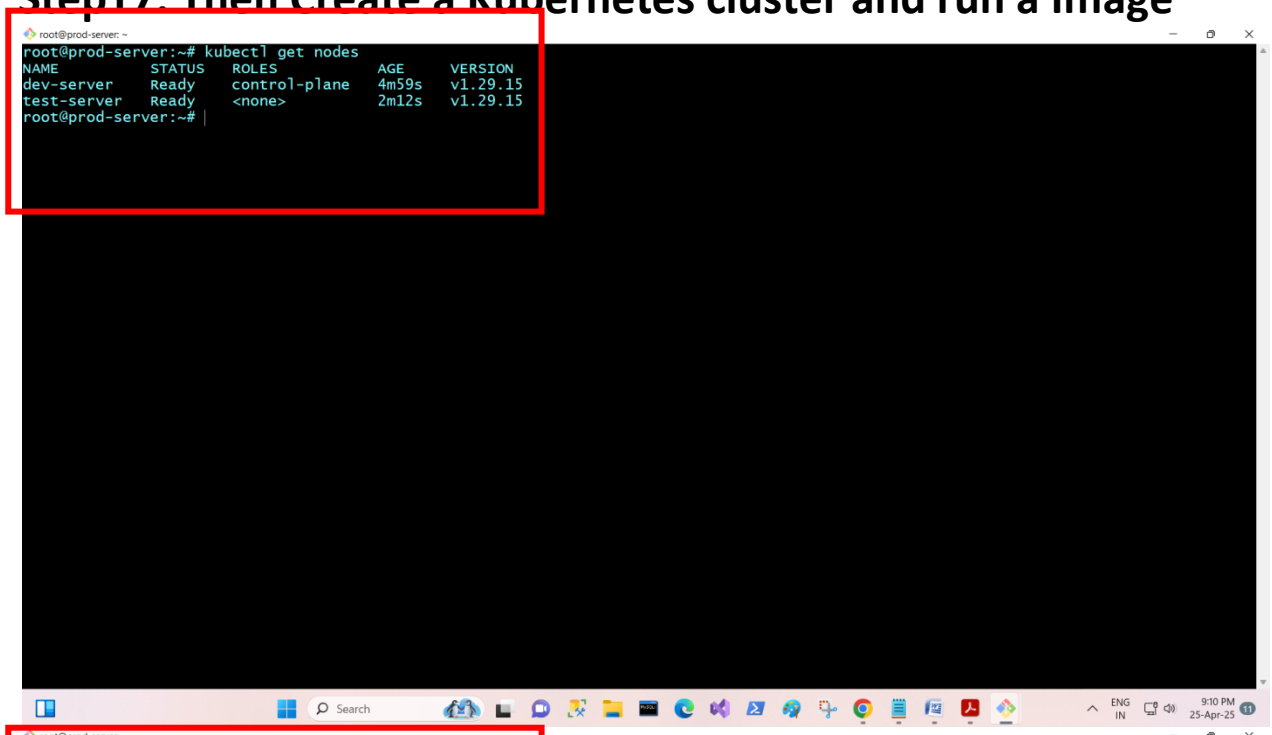

Not secure 54.164.183.224:8080/manage/computer/createItem
Jenkins Credentials Provider: Jenkins
Treat username as secret
Private Key
Enter directly
Key
Enter New Secret Below
-----END RSA PRIVATE KEY-----
Passphrase
Cancel
Add

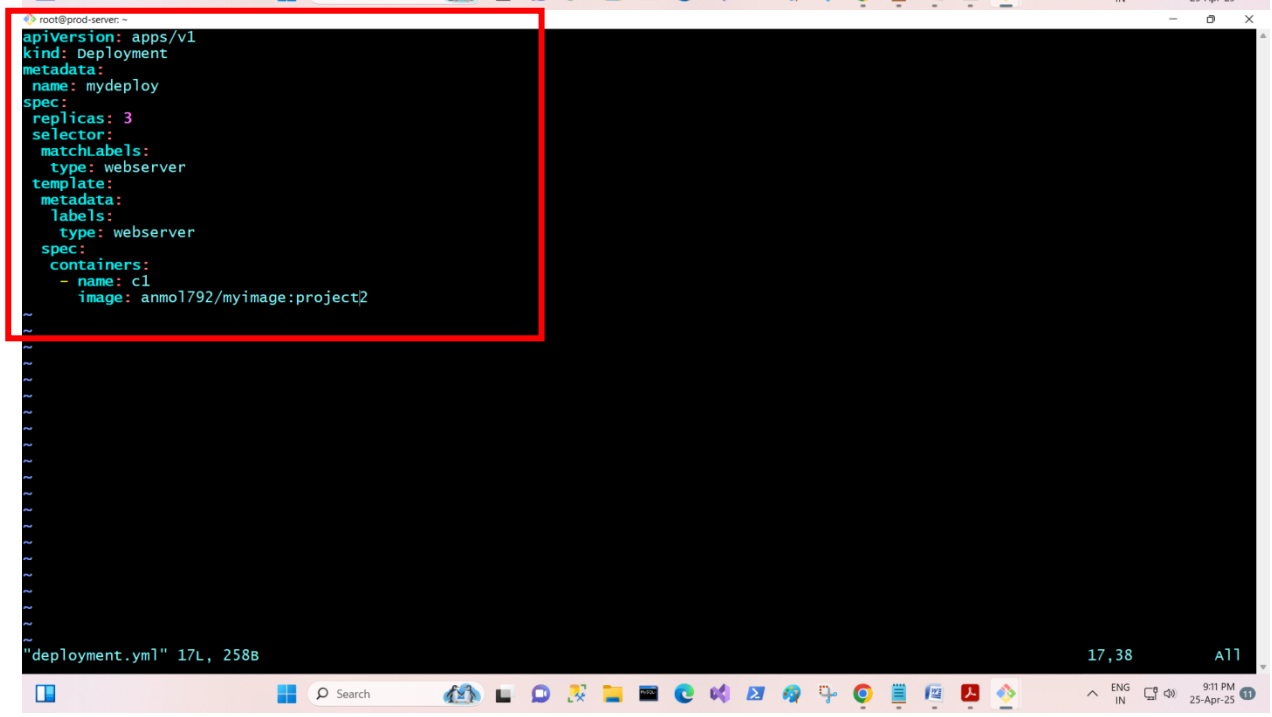

Nodes [Jenkins]
54.164.183.224:8080/manage/computer/
Jenkins
admin
log out
Dashboard > Manage Jenkins > Nodes
Nodes
Clouds
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node 0/2
testserver 0/1
Nodes
New Node
Configure Monitors
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
Built-In Node
Linux (amd64)
In sync
2.91 GiB
0 B
2.91 GiB
0ms
testserver
N/A
N/A
N/A
N/A
N/A
Data obtained
4 ms
4 ms
3 ms
2 ms
1 ms
1 ms
Icon: S M L
Legend
REST API
Jenkins 2.492.3


Health-Care-Project Config [Jen
54.164.183.224:8080/job/Health-Care-Project/configure
Dashboard > Health-Care-Project > Configuration
Configure
General
Triggers
Pipeline
Advanced
Define your Pipeline using Groovy directly or pull it from source control.
Definition
Pipeline script
Script
sh 'docker push anmol792/myimage:project2'
}
}
stage('Deploy on Test server')
{
agent {
label 'test_server'
}
steps{
sh 'docker run -d -P anmol792/myimage:project2'
}
}
Use Groovy Sandbox
Save
Apply

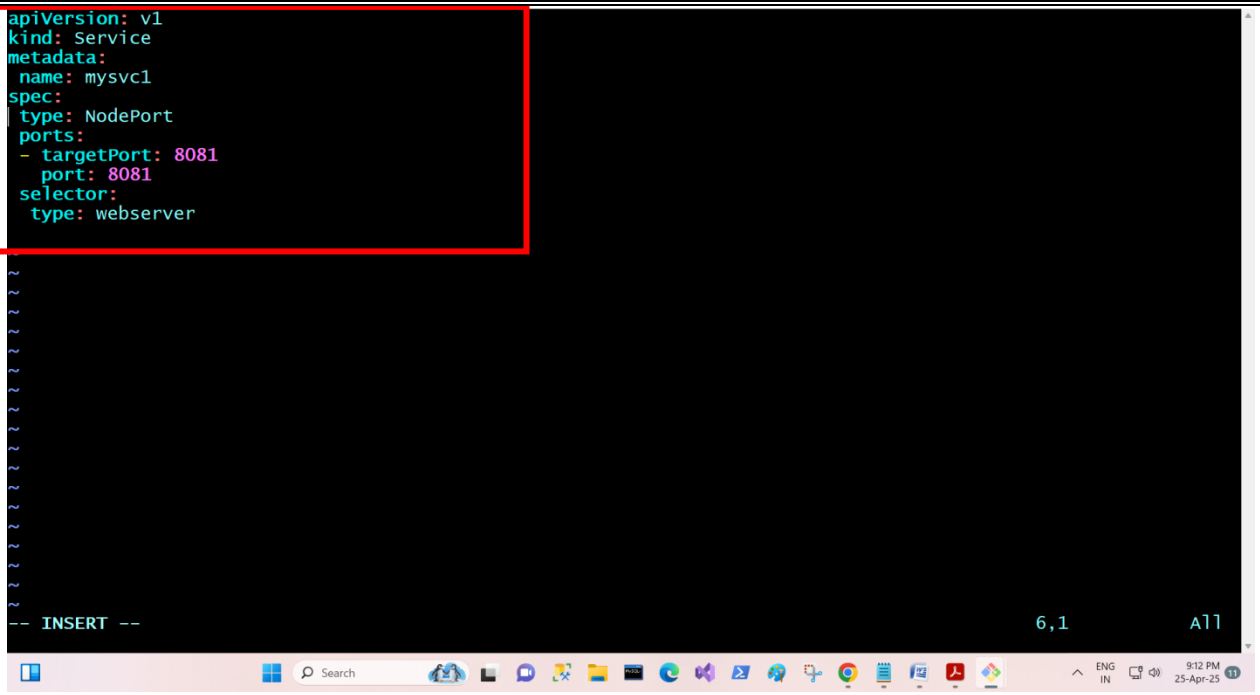

Not secure 54.164.183.224:8080/job/Health-Care-Project/3/console
Dashboard > Health-Care-Project > #3
5c78d662721e: Download complete
db38d58ec8ab: Verifying Checksum
db38d58ec8ab: Download complete
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
66223a710990: Pull complete
db38d58ec8ab: Pull complete
5c78d662721e: Pull complete
Digest: sha256:bd54894736f8264c71fc15eccde365ed16ccd921cfb3ad4fef36f4343de6b9f8
Status: Downloaded newer image for anmol792/myimage:project2
942539fa13ba7318cdecac2249fb87161da8c08b6df8bad8f5866d78247f43d3
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
REST API
Jenkins 2.492.3
root@test-server: /tmp/jenkinsdir/workspace
root@test-server:/tmp/jenkinsdir/workspace# ls
Health-Care-Project Health-Care-Project@tmp
root@test-server:/tmp/jenkinsdir/workspace# docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
942539fa13ba anmol792/myimage:project2 "java -jar /app.jar" 3 minutes ago Up 3 minutes 0.0.0.0:32768->8081/t
cp, :::32768->8081/tcp angry_golick
root@test-server:/tmp/jenkinsdir/workspace#

Step17: Then Create a Kubernetes cluster and run a Image

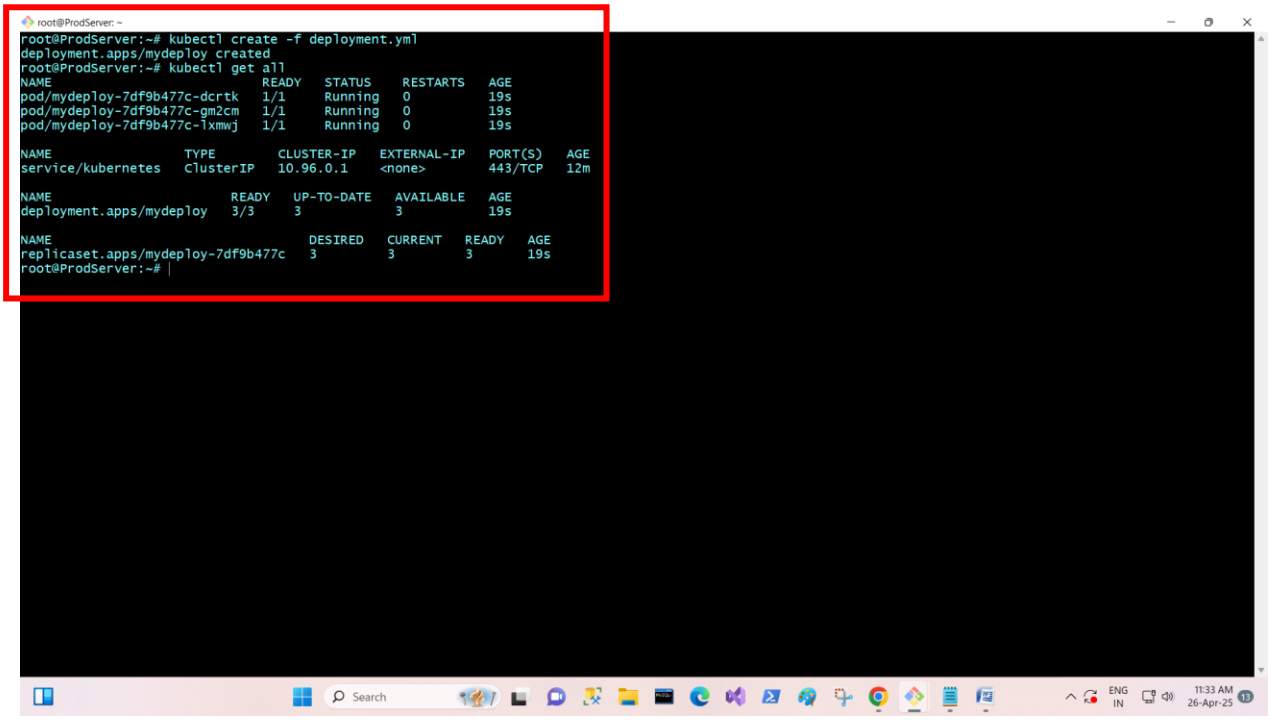


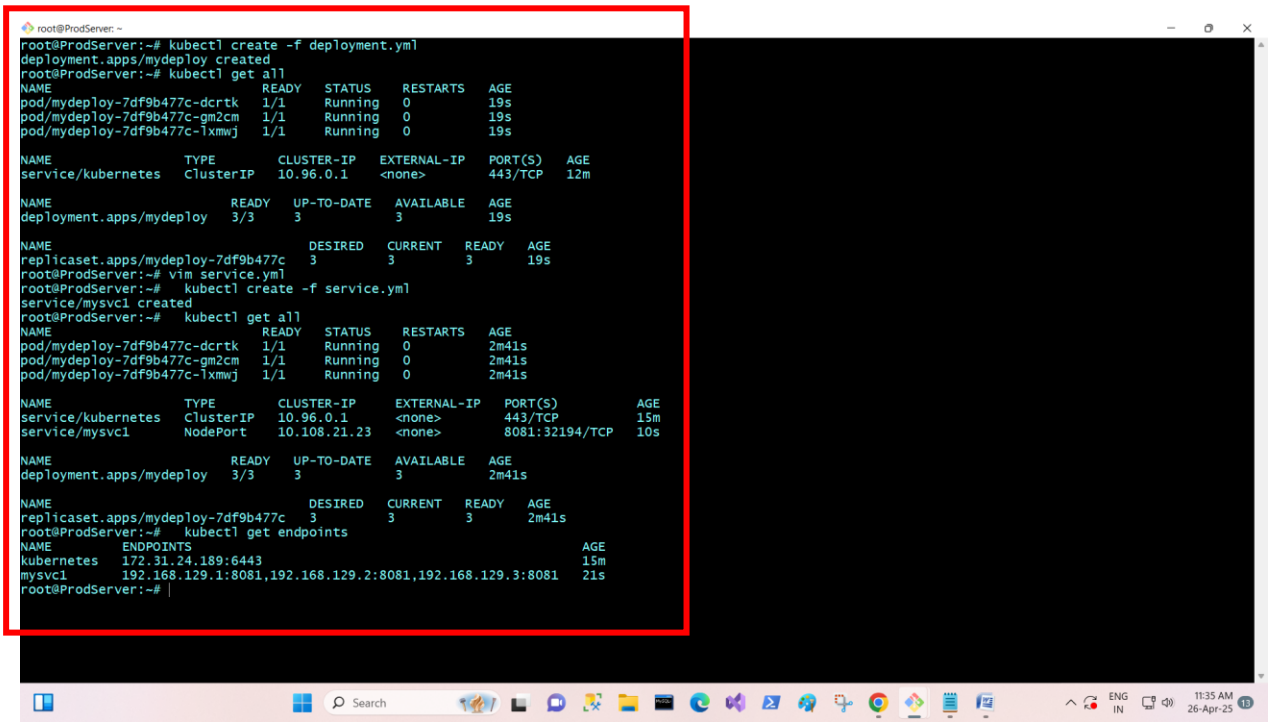


```
apiVersion: v1
kind: Service
metadata:
 name: mysvc1
spec:
 type: NodePort
 ports:
 - targetPort: 8081
   port: 8081
 selector:
  type: webserver
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    6,1           All
```

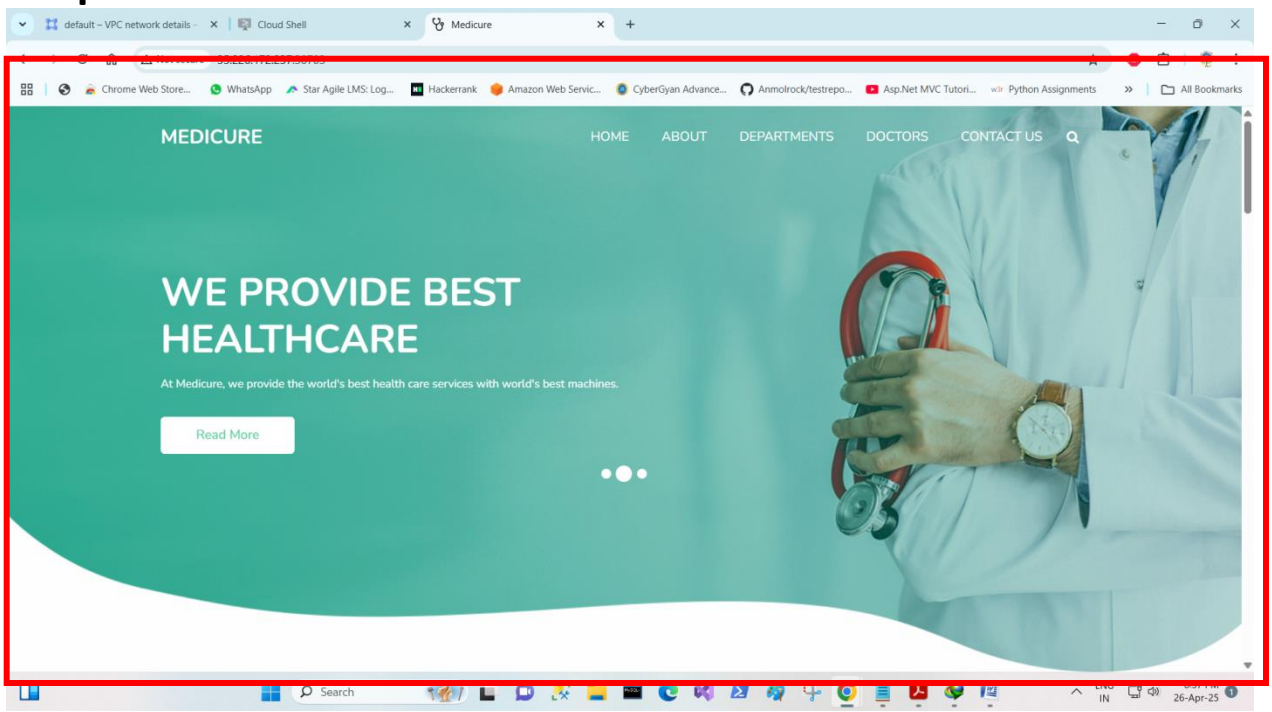


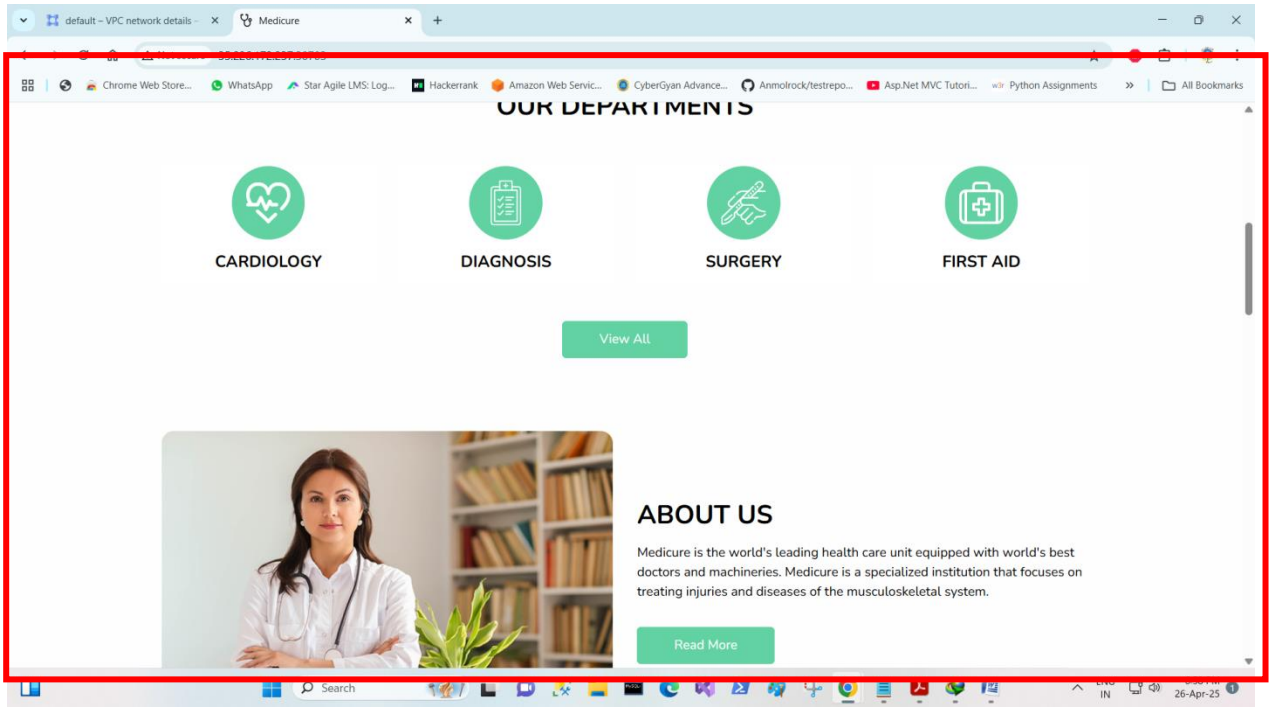
```
root@ProdServer:~# kubectl create -f deployment.yml
deployment.apps/mydeploy created
root@ProdServer:~# kubectl get all
NAME                            READY   STATUS    RESTARTS   AGE
pod/mydeploy-7df9b477c-dcrtk    1/1     Running   0          19s
pod/mydeploy-7df9b477c-gm2cm    1/1     Running   0          19s
pod/mydeploy-7df9b477c-lxmwj    1/1     Running   0          19s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   12m

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy    3/3     3            3           19s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeploy-7df9b477c    3         3         3       19s
root@ProdServer:~#
```



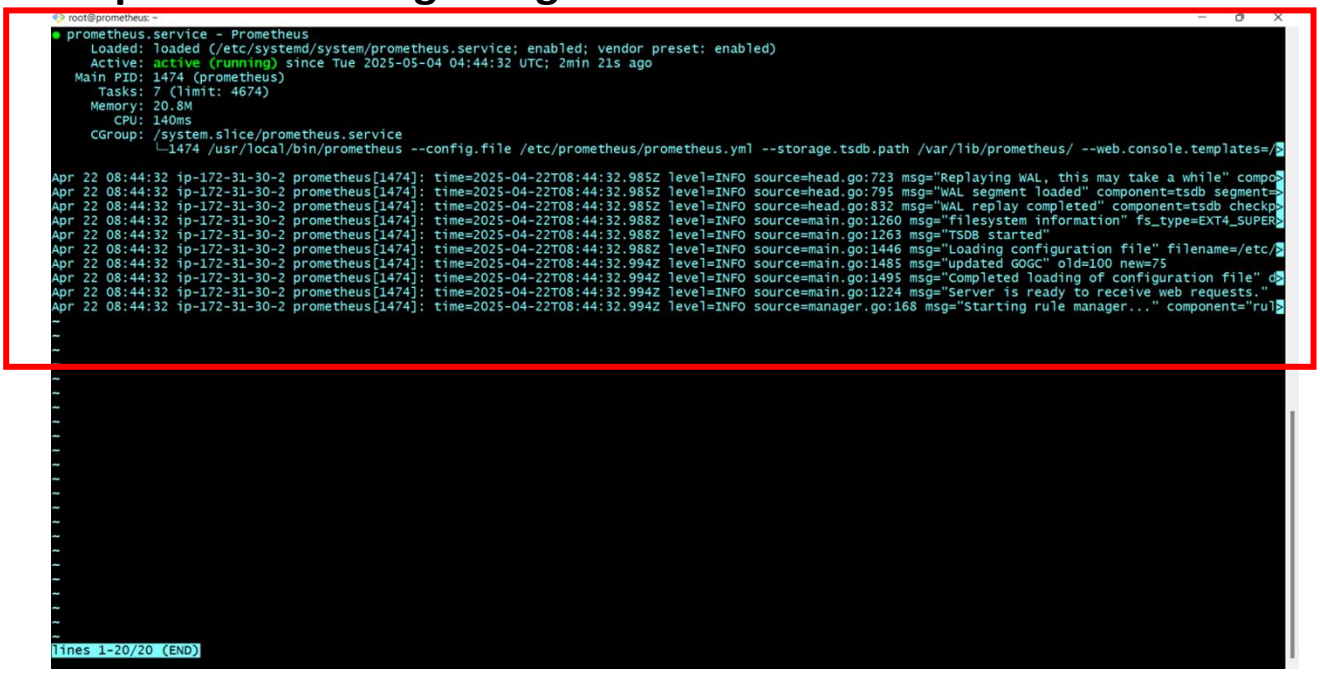

root@ProdServer:~# kubectl create -f deployment.yml
deployment.apps/mydeploy created
root@ProdServer:~# kubectl get all
NAME READY STATUS RESTARTS AGE
pod/mydeploy-7df9b477c-dcrtk 1/1 Running 0 19s
pod/mydeploy-7df9b477c-gm2cm 1/1 Running 0 19s
pod/mydeploy-7df9b477c-lxmwj 1/1 Running 0 19s
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 12m
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/mydeploy 3/3 3 3 19s
NAME DESIRED CURRENT READY AGE
replicaset.apps/mydeploy-7df9b477c 3 3 3 19s
root@ProdServer:~# vim service.yml
root@ProdServer:~# kubectl create -f service.yml
service/mysvc1 created
root@ProdServer:~# kubectl get all
NAME READY STATUS RESTARTS AGE
pod/mydeploy-7df9b477c-dcrtk 1/1 Running 0 2m41s
pod/mydeploy-7df9b477c-gm2cm 1/1 Running 0 2m41s
pod/mydeploy-7df9b477c-lxmwj 1/1 Running 0 2m41s
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 15m
service/mysvc1 NodePort 10.108.21.23 <none> 8081:32194/TCP 10s
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/mydeploy 3/3 3 3 2m41s
NAME DESIRED CURRENT READY AGE
replicaset.apps/mydeploy-7df9b477c 3 3 3 2m41s
root@ProdServer:~# kubectl get endpoints
NAME ENDPOINTS AGE
kubernetes 172.31.24.189:6443 15m
mysvc1 192.168.129.1:8081,192.168.129.2:8081,192.168.129.3:8081 21s
root@ProdServer:~#

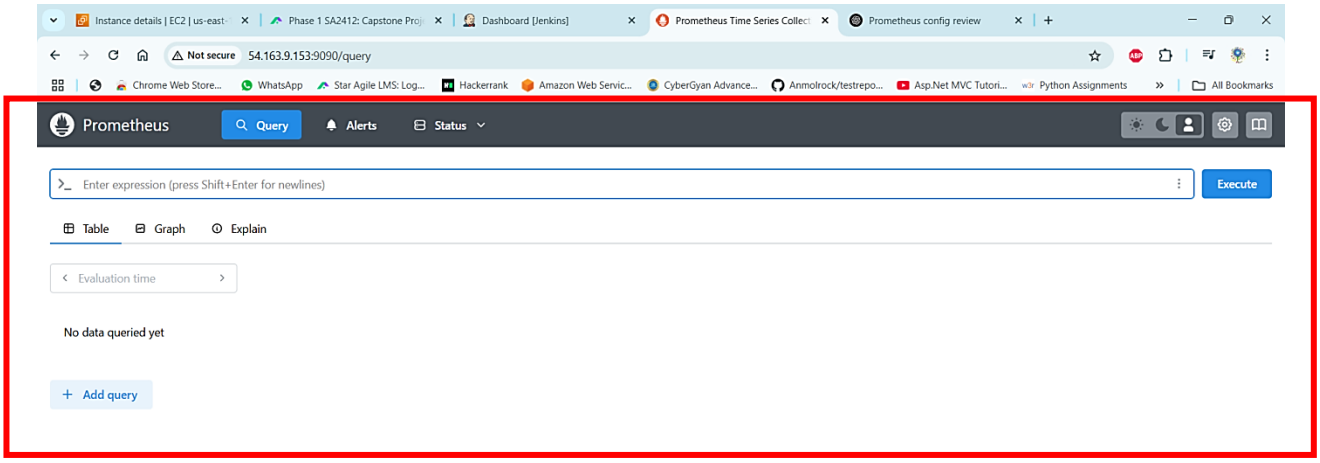
Output➔

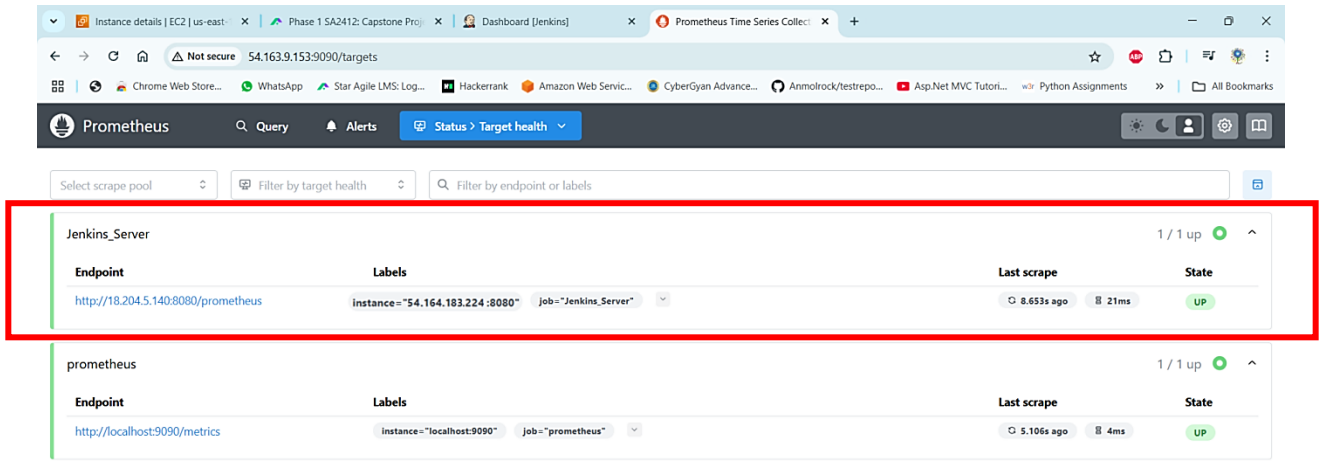

MEDICURE
HOME
ABOUT
DEPARTMENTS
DOCTORS
CONTACT US
WE PROVIDE BEST HEALTHCARE
At Medicure, we provide the world's best health care services with world's best machines.
Read More

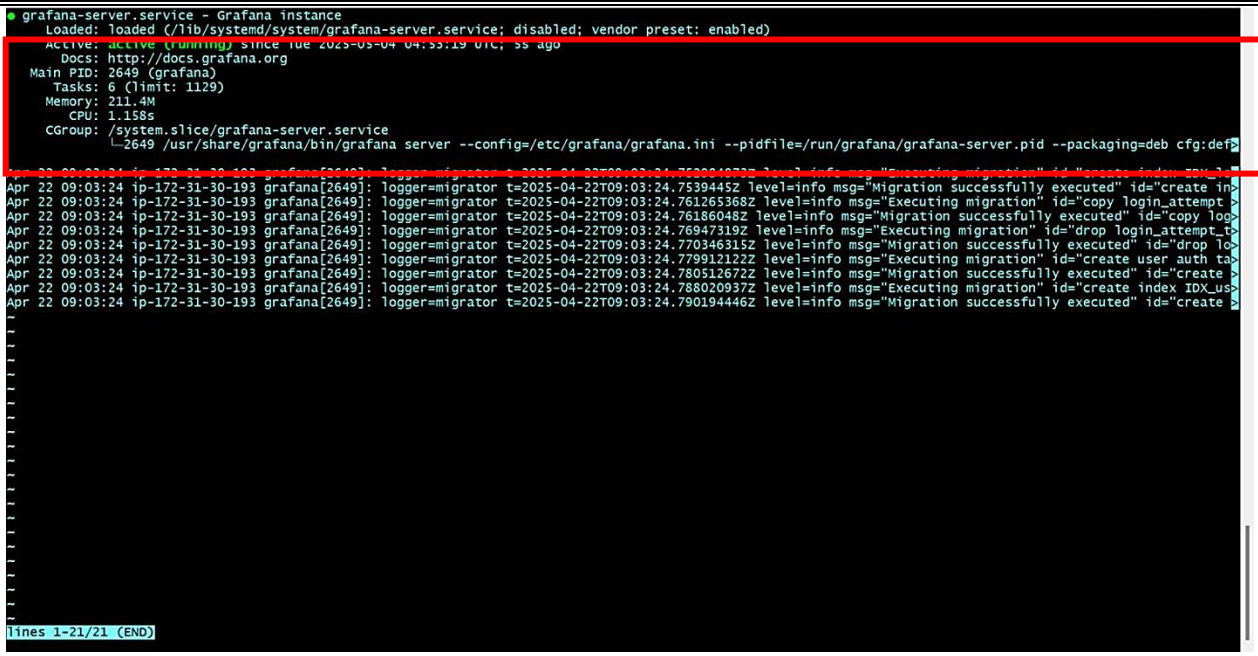


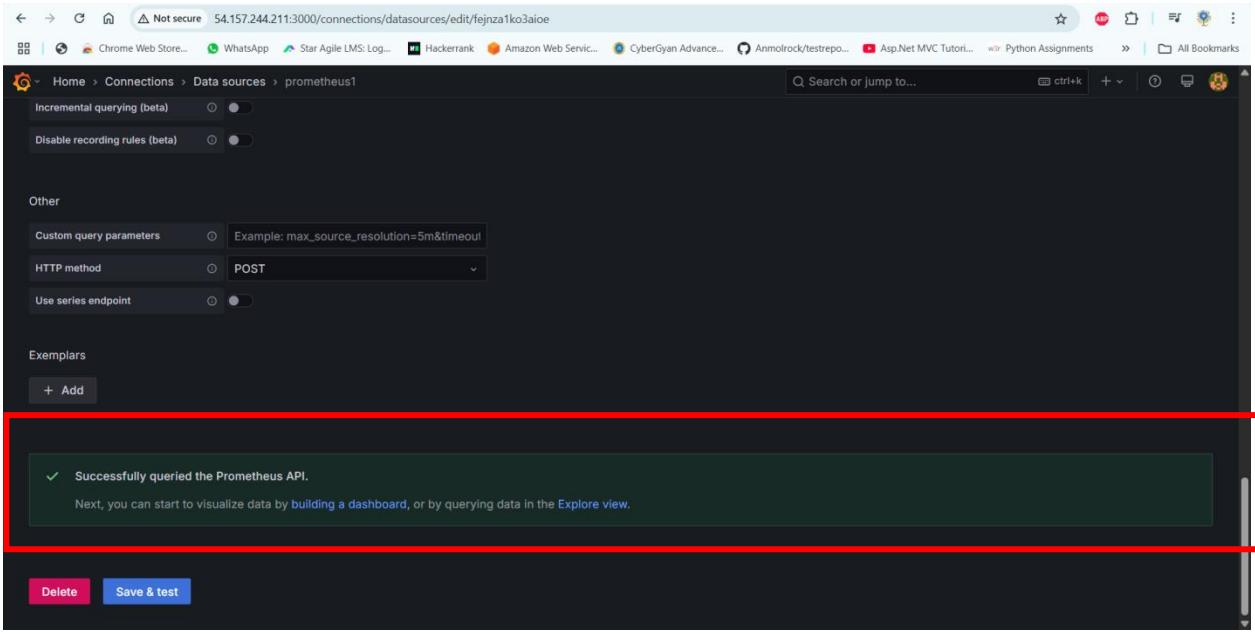
Step18: Monitoring Using Prometheus and Grafana

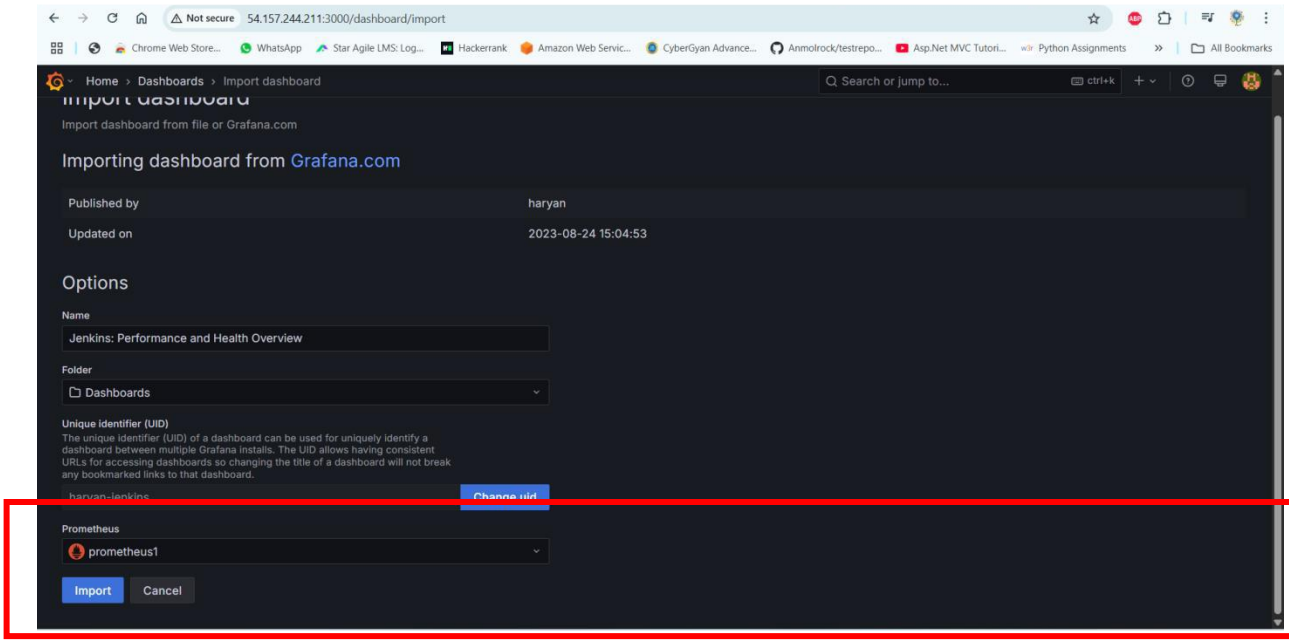


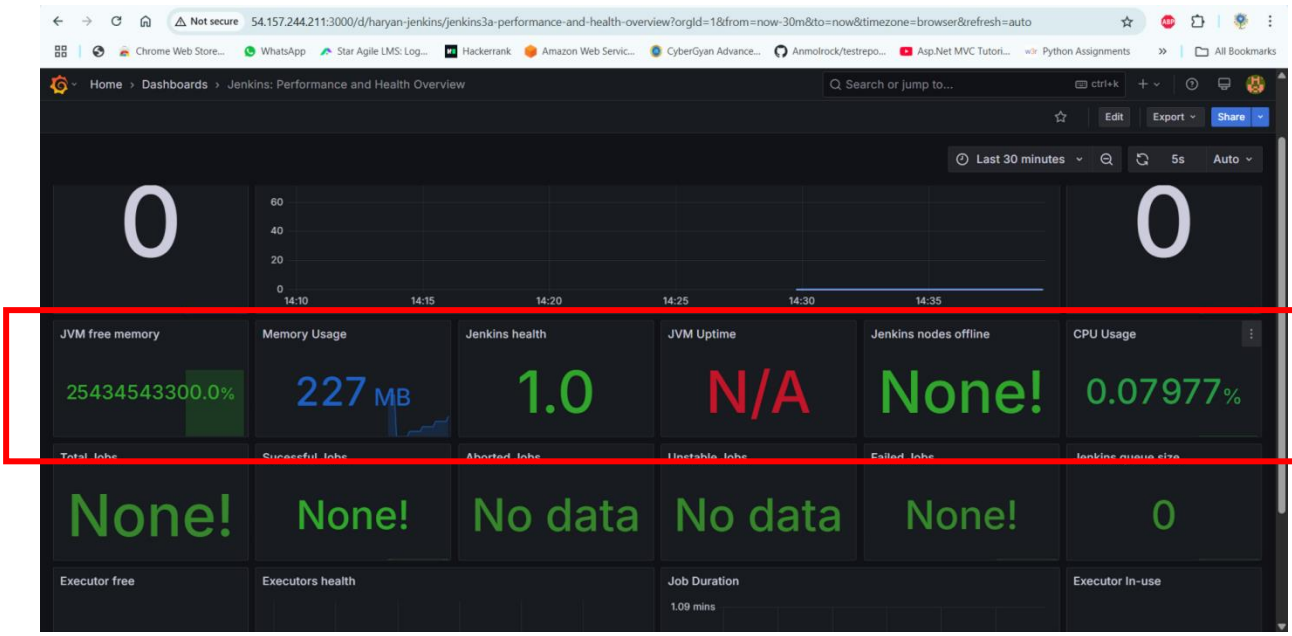

Not secure 54.163.9.153:9090/query
Prometheus
Query
Alerts
Status
Enter expression (press Shift+Enter for newlines)
Execute
Table
Graph
Explain
Evaluation time
No data queried yet
Add query


Not secure 54.163.9.153:9090/targets
Prometheus
Query
Alerts
Status > Target health
Select scrape pool
Filter by target health
Filter by endpoint or labels
Jenkins_Server
1 / 1 up
Endpoint
Labels
Last scrape
State
http://18.204.5.140:8080/prometheus
instance="54.164.183.224:8080"
job="Jenkins_Server"
8.653s ago
21ms
UP
prometheus
1 / 1 up
Endpoint
Labels
Last scrape
State
http://localhost:9090/metrics
instance="localhost:9090"
job="prometheus"
5.106s ago
4ms
UP


grafana-server.service - Grafana instance
Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
Active: active (running) since Tue 2025-05-04 04:55:19 UTC; 5s ago
Docs: http://docs.grafana.org
Main PID: 2649 (grafana)
Tasks: 6 (limit: 1129)
Memory: 211.4M
CPU: 1.158s
CGroup: /system.slice/grafana-server.service
└─2649 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:def
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.7539445Z level=info msg="Migration successfully executed" id="create in
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.761265368Z level=info msg="Executing migration" id="copy login_attempt
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.76186048Z level=info msg="Migration successfully executed" id="copy log
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.76947319Z level=info msg="Executing migration" id="drop login_attempt_t
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.770346315Z level=info msg="Migration successfully executed" id="drop lo
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.779912122Z level=info msg="Executing migration" id="create user auth ta
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.780512672Z level=info msg="Migration successfully executed" id="create
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.788020937Z level=info msg="Executing migration" id="create index IDX_us
Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.790194446Z level=info msg="Migration successfully executed" id="create
lines 1-21/21 (END)


Not secure 54.157.244.211:3000/connections/datasources/edit/fejnza1ko3aioe
Home › Connections › Data sources › prometheus1
Search or jump to...
Incremental querying (beta)
Disable recording rules (beta)
Other
Custom query parameters
Example: max_source_resolution=5m&timeout
HTTP method
POST
Use series endpoint
Exemplars
+ Add
Successfully queried the Prometheus API.
Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.
Delete
Save & test


Not secure 54.157.244.211:3000/dashboard/import
Home › Dashboards › Import dashboard
Search or jump to... ctrl+k
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
prometheus1
Import
Cancel


Not secure 54.157.244.211:3000/d/haryan-jenkins/jenkins3a-performance-and-health-overview?orgId=1&from=now-30m&to=now&timezone=browser&refresh=auto
Home › Dashboards › Jenkins: Performance and Health Overview
Search or jump to... ctrl+k
Edit
Export
Share
Last 30 minutes
5s
Auto
0
60
40
20
0
14:10
14:15
14:20
14:25
14:30
14:35
0
JVM free memory
25434543300.0%
Memory Usage
227 MB
Jenkins health
1.0
JVM Uptime
N/A
Jenkins nodes offline
None!
CPU Usage
0.07977%
None!
None!
No data
No data
None!
0
Executor free
Executors health
Job Duration
1.09 mins
Executor In-use