

# **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.

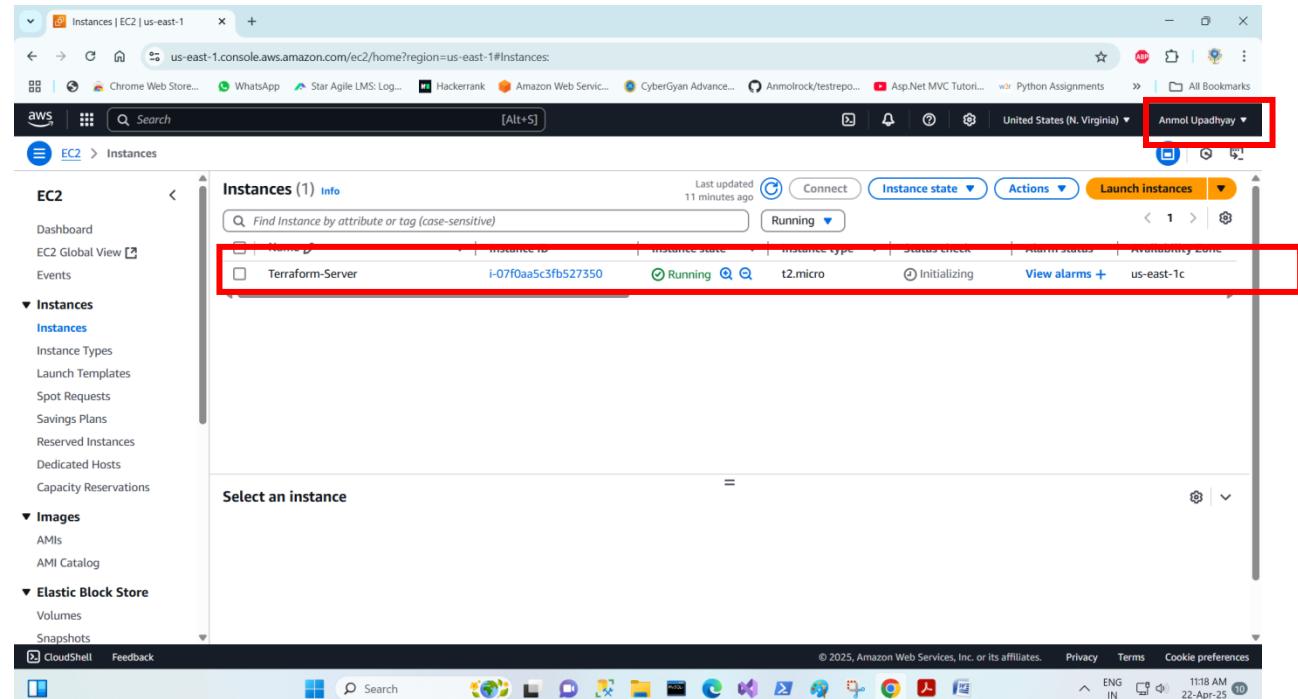
### **Business challenge/requirement**

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.

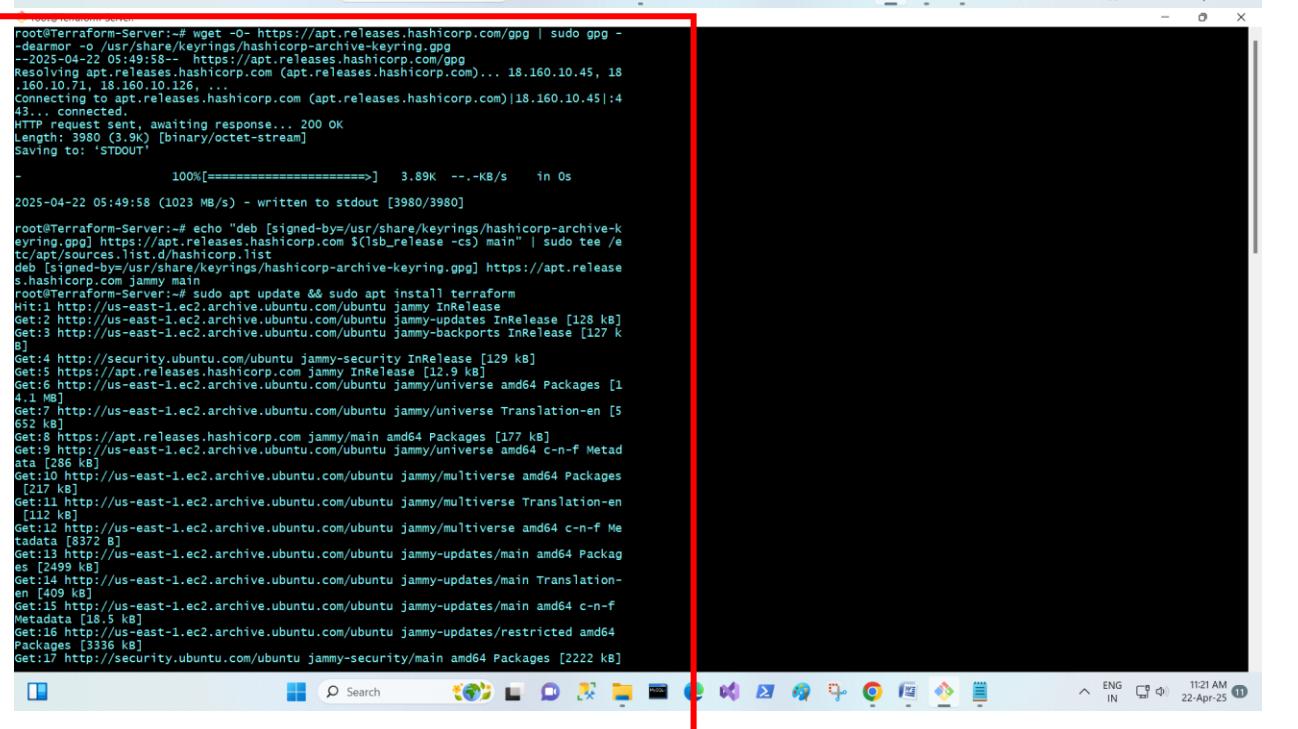
The deployment should then be tested using a test automation tool, and if the build is successful, it should be deployed to the prod server/cluster using Kubernetes. All this should happen automatically and should be triggered from a push to the GitHub master branch.

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.



The screenshot shows the AWS EC2 Instances page. A single instance named "Terraform-Server" is listed as "Running". The user "Anmol Upadhyay" is highlighted in the top right corner. The instance details table is also highlighted with a red box.



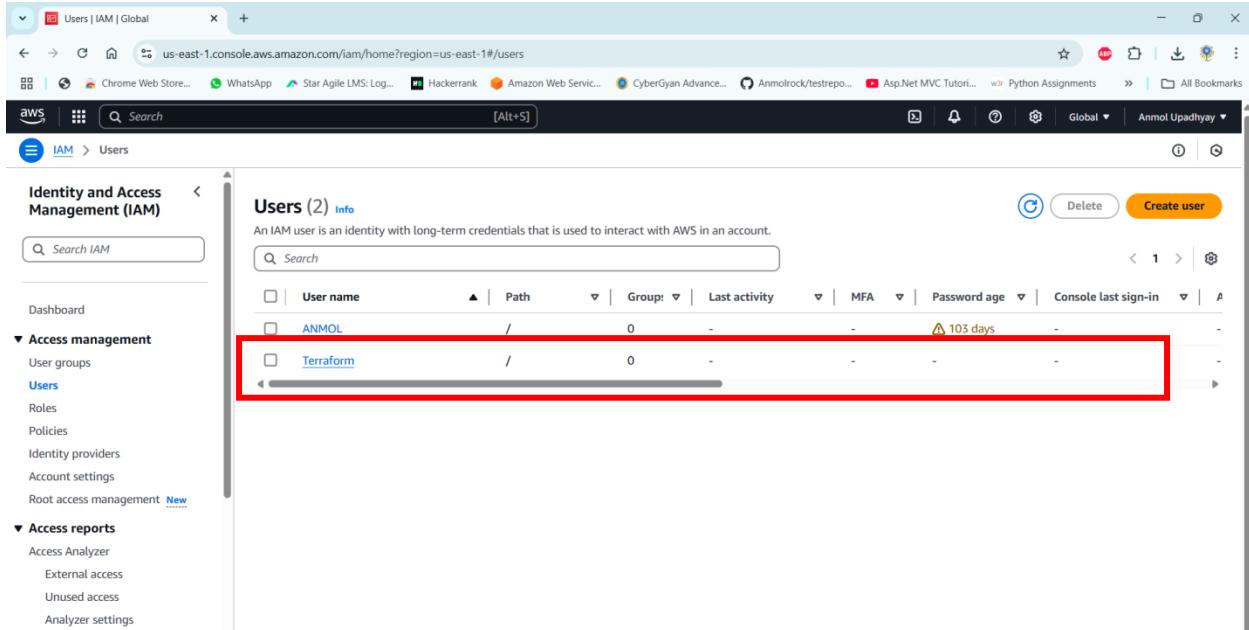
The terminal window shows the following commands being run:

```
root@terraform-Server:~# wget -O https://apt.releases.hashicorp.com/gpg | sudo gpg -darmor > /usr/share/keyrings/hashicorp-archive-keyring.gpg
--2025-04-22 05:49:58-- https://apt.releases.hashicorp.com/gpg
Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.160.10.45, 18.160.10.71, 18.160.10.126...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.160.10.45|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

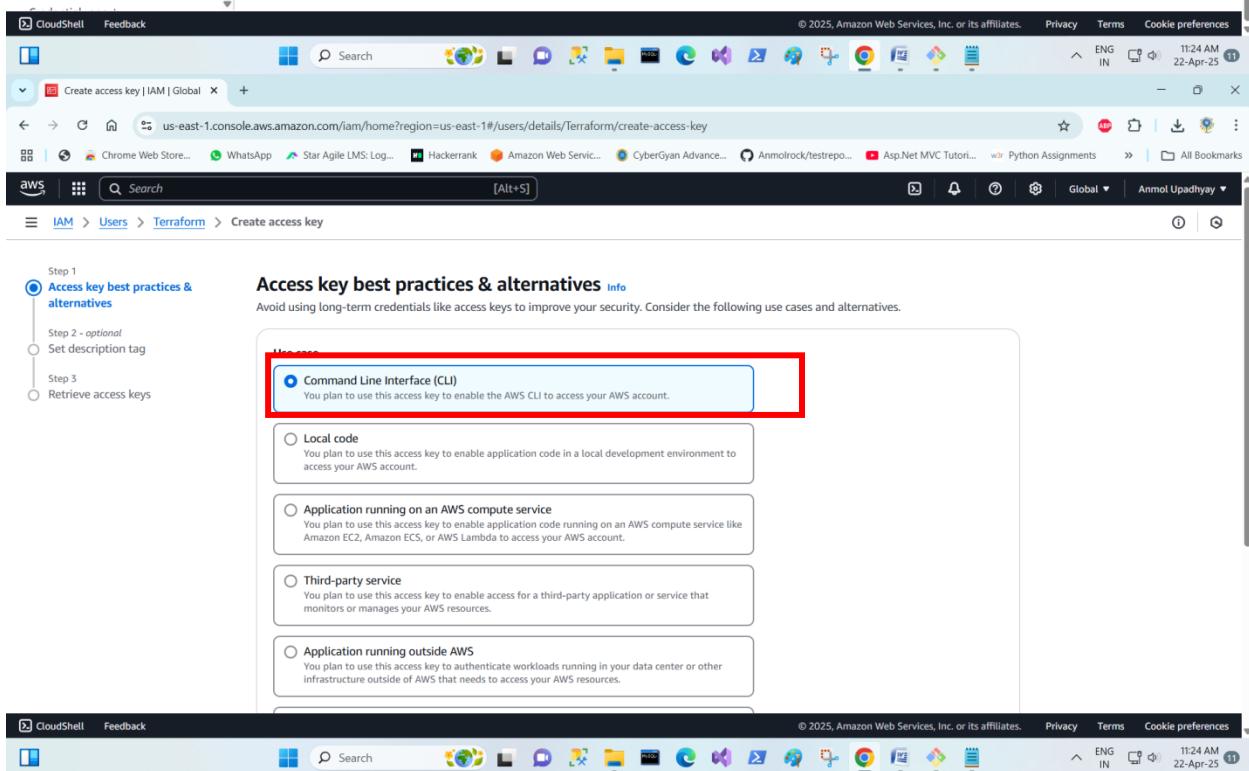
[  0%] 3.89K --.-KB/s in 0s
2025-04-22 05:49:58 (1023 MB/s) - written to stdout [3980/3980]

root@terraform-Server:~# echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.release.s.hashicorp.com jammy main
root@terraform-Server:~# sudo apt update && sudo apt install terraform
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [4.1 MB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5.652 kB]
Get:8 https://apt.releases.hashicorp.com jammy/main amd64 Packages [177 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2499 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [409 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [18.5 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3336 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2222 kB]
```

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



The screenshot shows the AWS IAM 'Users' page. There are two users listed: 'ANMOL' and 'Terraform'. The 'Terraform' user is highlighted with a red box. The 'ANMOL' user has a yellow exclamation mark icon next to their name, indicating they have not signed in recently.

The screenshot shows the 'Create access key' step in the AWS IAM console. The 'Access key best practices & alternatives' section is displayed. The 'Command Line Interface (CLI)' option is selected and highlighted with a red box. Other options include 'Local code', 'Application running on an AWS compute service', 'Third-party service', and 'Application running outside AWS'.

The screenshot shows the AWS IAM 'Create access key' page. A green success message box at the top states 'Access key created' and 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, a section titled 'Access key best practices & alternatives' is visible. Under 'Step 3: Retrieve access keys', there is a table with two columns: 'Access key' and 'Secret access key'. The 'Access key' column contains the value 'AKIA34AMDIBPXJGPFH72'. The 'Secret access key' column has a 'Show' button next to a redacted password. A red box highlights this entire row. At the bottom right of the page are 'Download .csv file' and 'Done' buttons.

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

The screenshot shows a terminal window with the following command history:

```
root@ip-172-31-83-137:~/myproject# vim aws_infra.tf
```

A red box highlights the command 'vim aws\_infra.tf'. The rest of the terminal window is blacked out.

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "s1-vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "s1-vpc"
  }
}

resource "aws_subnet" "subnet-1"{
  vpc_id = aws_vpc.s1-vpc.id
  cidr_block = "10.0.1.0/24"
  depends_on = [aws_vpc.s1-vpc]
  map_public_ip_on_launch = true
  tags = {
    Name = "s1-subnet"
  }
}

resource "aws_route_table" "s1-route-table"{
  vpc_id = aws_vpc.s1-vpc.id
  tags = {
    Name = "s1-route-table"
  }
}
```

24,4 Top

```
resource "aws_route_table" "s1-route-table"{
  vpc_id = aws_vpc.s1-vpc.id
  tags = {
    Name = "s1-route-table"
  }
}

resource "aws_route_table_association" "a" {
  subnet_id     = aws_subnet.subnet-1.id
  route_table_id = aws_route_table.s1-route-table.id
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.s1-vpc.id
  depends_on = [aws_vpc.s1-vpc]
  tags = {
    Name = "s1-gw"
  }
}

resource "aws_route" "s1-route" {
  route_table_id = aws_route_table.s1-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" "s1-sg" {
  name      = "sg_rule"
  vpc_id   = aws_vpc.s1-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%

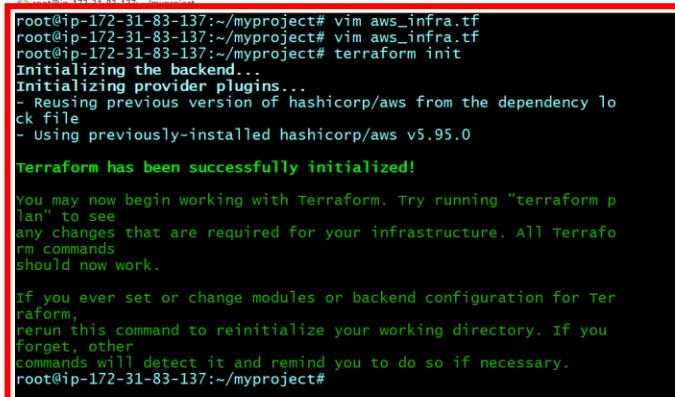
```
}
}
egress {
  from_port     = 0
  to_port       = 0
  protocol      = "-1"
  cidr_blocks   = ["0.0.0.0/0"]
}

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.s1-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.s1-sg.id]
  tags = {
    Name = "Project-EC2-TEST"
  }
}
```

115,1 Bot

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

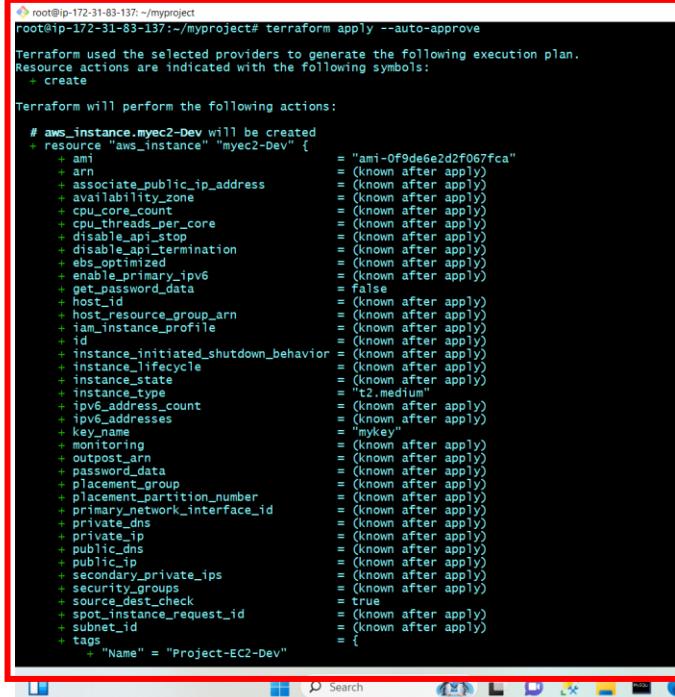


```
root@ip-172-31-83-137:~/myproject# vim aws_infra.tf
root@ip-172-31-83-137:~/myproject# vim aws_infra.tf
root@ip-172-31-83-137:~/myproject# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using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

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-172-31-83-137:~/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-0f9de6ed2f067fca"
  arn                             = (known after apply)
  associate_public_ip_address      = (known after apply)
  availability_zone                = (known after apply)
  cpu_core_count                   = (known after apply)
  ebs_optimized                    = (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
  ipv4_address_count               = (known after apply)
  ipv6_addresses                   = "mykey"
  key_name                         = (known after apply)
  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                  = 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

The screenshot shows the AWS Management Console interface for the EC2 service. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots). The main content area is titled 'Instances (5) Info' and displays a table of EC2 instances. The columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Five instances are listed: Project-EC2-TEST, Project-EC2-Dev, Terraform-Server, Grafana-Server, and Prometheus-Server. All instances are currently running. A red box highlights the first three instances: Project-EC2-TEST, Project-EC2-Dev, and Terraform-Server. The bottom of the screen shows the Windows taskbar with various pinned icons.

## Step6: Install Ansible on this server

```
root@Dev-Server:~# 
Unpacking python3-winrm (0.3.0-2) ...
Selecting previously unselected package sshpass.
Preparing to unpack .../12-sshpss_1.09-1_amd64.deb ...
Unpacking sshpass (1.09-1) ...
Setting up python3-ntlm-auth (1.4.0-1) ...
Setting up python3-resolvelib (0.8.1-1) ...
Setting up python3-kerberos (1.1.14-3.1build5) ...
Setting up ansible-core (2.17.11-1ppa~jammy) ...
Setting up sshpass (1.09-1) ...
Setting up python3-xmltodict (0.12.0-2) ...
Setting up python3-jmespath (0.10.0-1) ...
Setting up python3-requests-kerberos (0.12.0-2) ...
Setting up ansible (10.7.0-1ppa~jammy) ...
Setting up python3-nacl (1.5.0-2) ...
Setting up python3-requests-ntlm (1.1.0-1.1) ...
Setting up python3-winrm (0.3.0-2) ...
Setting up python3-paramiko (2.9.3-0ubuntu1.3) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@Dev-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:~# |
```



```
- 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

File Search Start Task View File Home Insert Edit View Tools Help

```
root@Dev-Server:~# vim playbook-install.yml
root@Dev-Server:~# ansible-playbook playbook-install.yml
[WARNING]: provided hosts list is empty, only localhost is available
Be 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:~#
```

File Search Start Task View File Home Insert Edit View Tools Help

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

The screenshot displays two windows. The top window is a web browser showing the Jenkins 'Getting Started' page with a red box highlighting the 'Unlock Jenkins' section. The bottom window is a terminal window showing the command `cat /var/lib/jenkins/secrets/initialAdminPassword` and its output, which is also highlighted with a red box.

Instance details | EC2 | us-east-1 Sign in [Jenkins] +

Not secure 54.164.183.224:8080/login?from=%2F

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

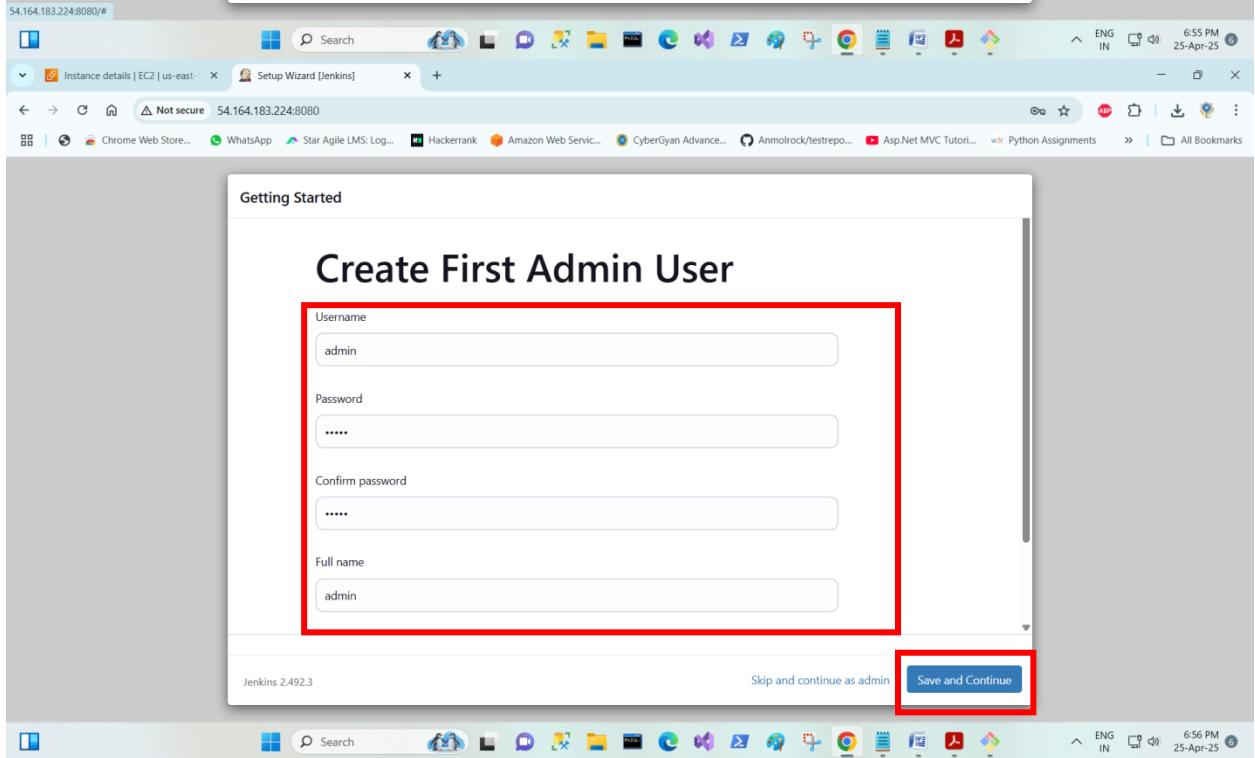
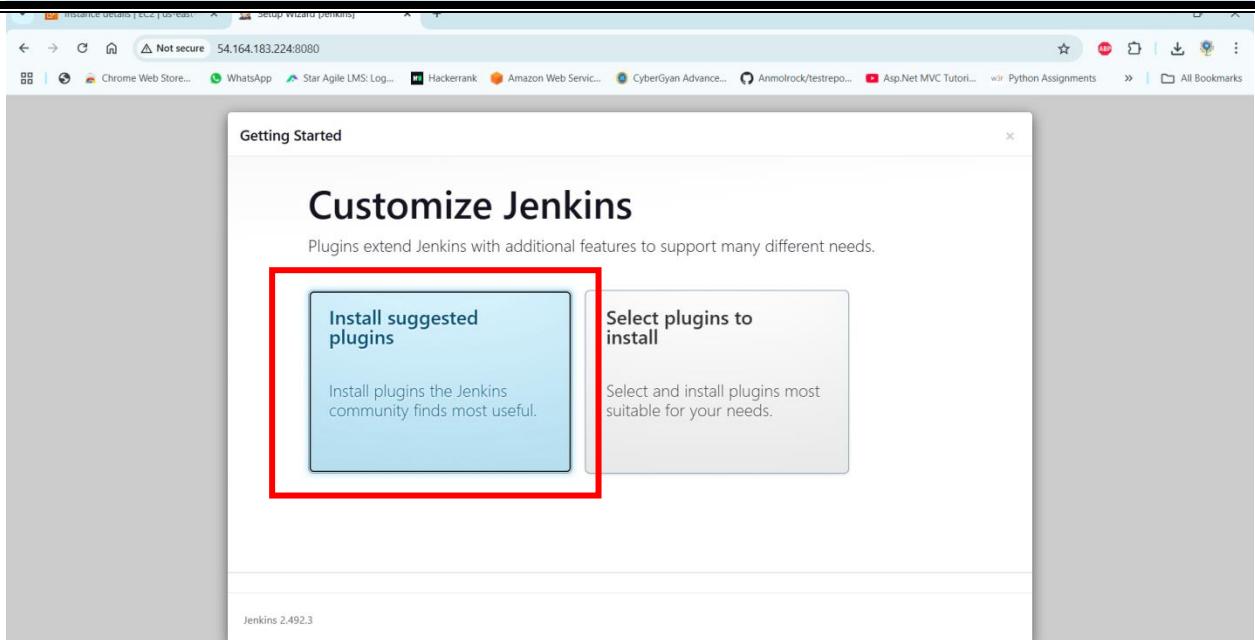
/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

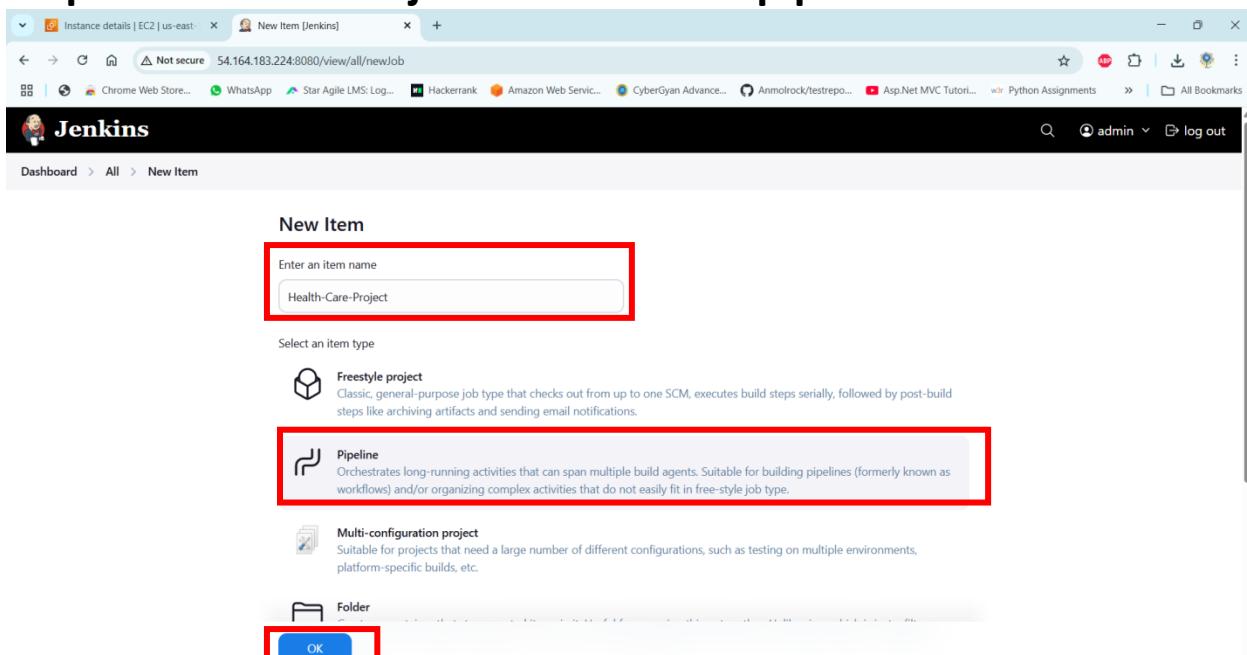
Administrator password

Continue

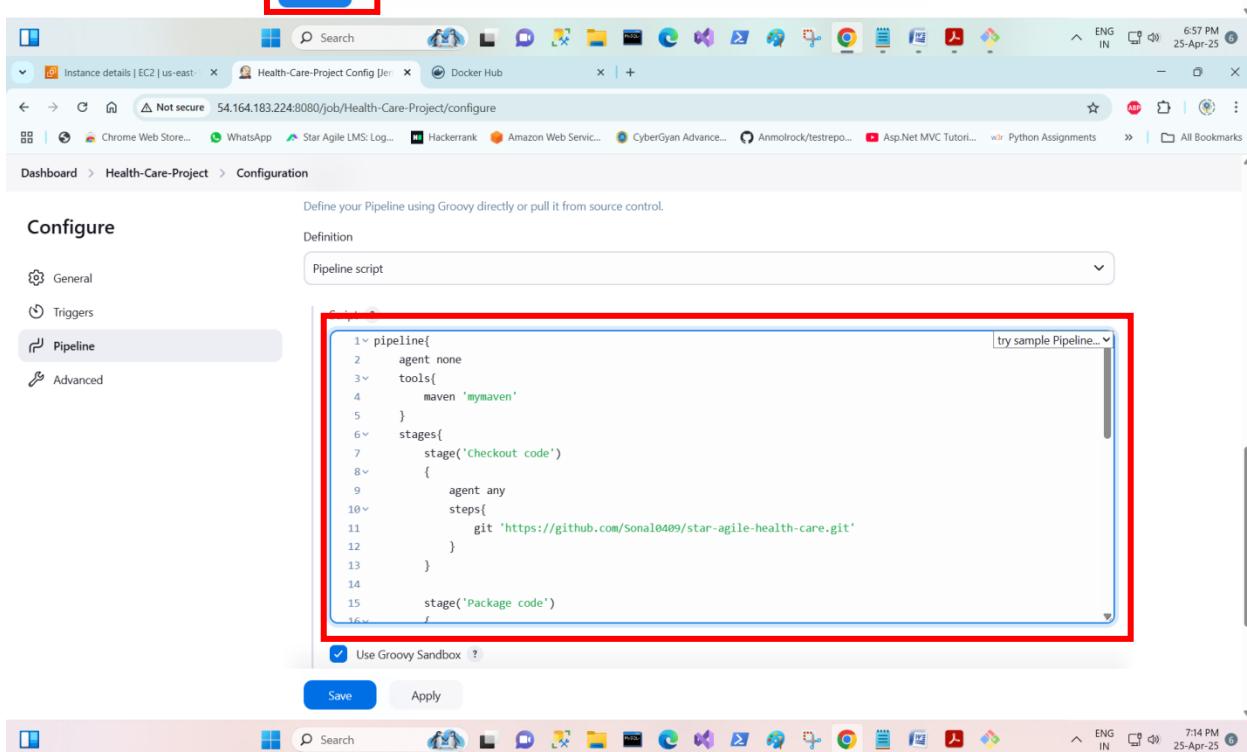
```
root@Dev-Server:~# cat /var/lib/jenkins/secrets/initialAdminPassword
759e6a01ad104e8db558e0d14c4e31b7
root@Dev-Server:~#
```



## Step9: Create a new job and Create a pipeline



The screenshot shows the Jenkins 'New Item' creation interface. A red box highlights the 'Health-Care-Project' entry in the 'Enter an item name' input field. Another red box highlights the 'Pipeline' option under 'Select an item type', which is described as 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' A blue 'OK' button is visible at the bottom of this section.



The screenshot shows the Jenkins 'Configuration' screen for the 'Health-Care-Project'. A red box highlights the Groovy pipeline script editor, which contains the following code:

```
1~ pipeline{
2~     agent none
3~     tools{
4~         maven 'my.maven'
5~     }
6~     stages{
7~         stage('checkout code'){
8~             agent any
9~             steps{
10~                 git 'https://github.com/Sonal0409/star-agile-health-care.git'
11~             }
12~         }
13~         stage('Package code')
14~     }
15~ }
```

Below the script editor, there is a checked checkbox for 'Use Groovy Sandbox' and two buttons: 'Save' and 'Apply'.

The screenshot shows two windows side-by-side. The left window is a Jenkins pipeline configuration page titled 'Configure' for a project named 'Health-Care-Project'. Under the 'Pipeline' tab, there is a 'Pipeline script' section containing Groovy code:

```
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      }
30
```

A red box highlights this code block. Below the script is a checkbox labeled 'Use Groovy Sandbox' with a question mark icon. At the bottom are 'Save' and 'Apply' buttons.

The right window is a terminal session on a Windows operating system. The command entered is:

```
root@Dev-Server:~# chmod -R 777 /var/run/docker.sock
```

This command grants full permissions (777) to the Docker socket file. A red box highlights this command in the terminal window. The terminal window has a standard Windows taskbar at the bottom.

Dashboard > Health-Care-Project > Pipeline Syntax

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

Windows taskbar: Search, File Explorer, Task View, Start button, Icons, Language: ENG IN, Date: 7:09 PM 25-Apr-25

Instance details | EC2 | us-east-1 | Health-Care-Project Config [jenkins] | Pipeline Syntax Snippet Generator | anmol792 (anmol upadhyay) | Docker Home

Dashboard > 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

Jenkins 2.492.3

Windows taskbar: Search, File Explorer, Task View, Start button, Icons, Language: ENG IN, Date: 7:08 PM 25-Apr-25

The screenshot shows the Jenkins Pipeline configuration page. On the left, a sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Configure" and "Definition". A red box highlights the "Pipeline script" section, which contains the following Groovy code:

```
20
21
22
23
24
25
26
27
28
29
30
31      stage('push the image to dockerhub')
32    {
33      agent any
34      steps{
35        withCredentials([string(credentialsId: 'DOCKER_HUB_PASSWD', variable: 'DOCKER_HUB_PASSWD')])
36        {
37          sh 'docker login -u ammol792 -p ${DOCKER_HUB_PASSWD}'
38        }
39        sh 'docker tag myimage:project2 ammol792/myimage:project2'
40        sh 'docker push ammol792/myimage:project2'
41      }
42    }
43  }
44 }
```

Below the script, there is a checkbox labeled "Use Groovy Sandbox" with a question mark icon. At the bottom are "Save" and "Apply" buttons.

## Step10: Save the job and run the pipeline

The screenshot shows the Jenkins job console for job #2. The log output is displayed in a red box:

```
03127cd4b479b: Preparing
9c742cd6c7a5: Preparing
293d5db30c9f: Waiting
03127cd4b479b: Waiting
9c742cd6c7a5: Waiting
0b55156abf26: Mounted from library/openjdk
826c3ddbb29c: Mounted from library/openjdk
b626401ef603: Mounted from library/openjdk
7b7f3078e1db: Mounted from library/openjdk
293d5db30c9f: Mounted from library/openjdk
03127cd4b479b: Mounted from library/openjdk
9c742cd6c7a5: Mounted from library/openjdk
20b4d4fd34bb: Pushed
project2: digest: sha256:bd54894736f8264c71fc15eccde365ed16cc921cfb3ad4fef36f4343de6b9f8 size: 2007
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline]
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
```

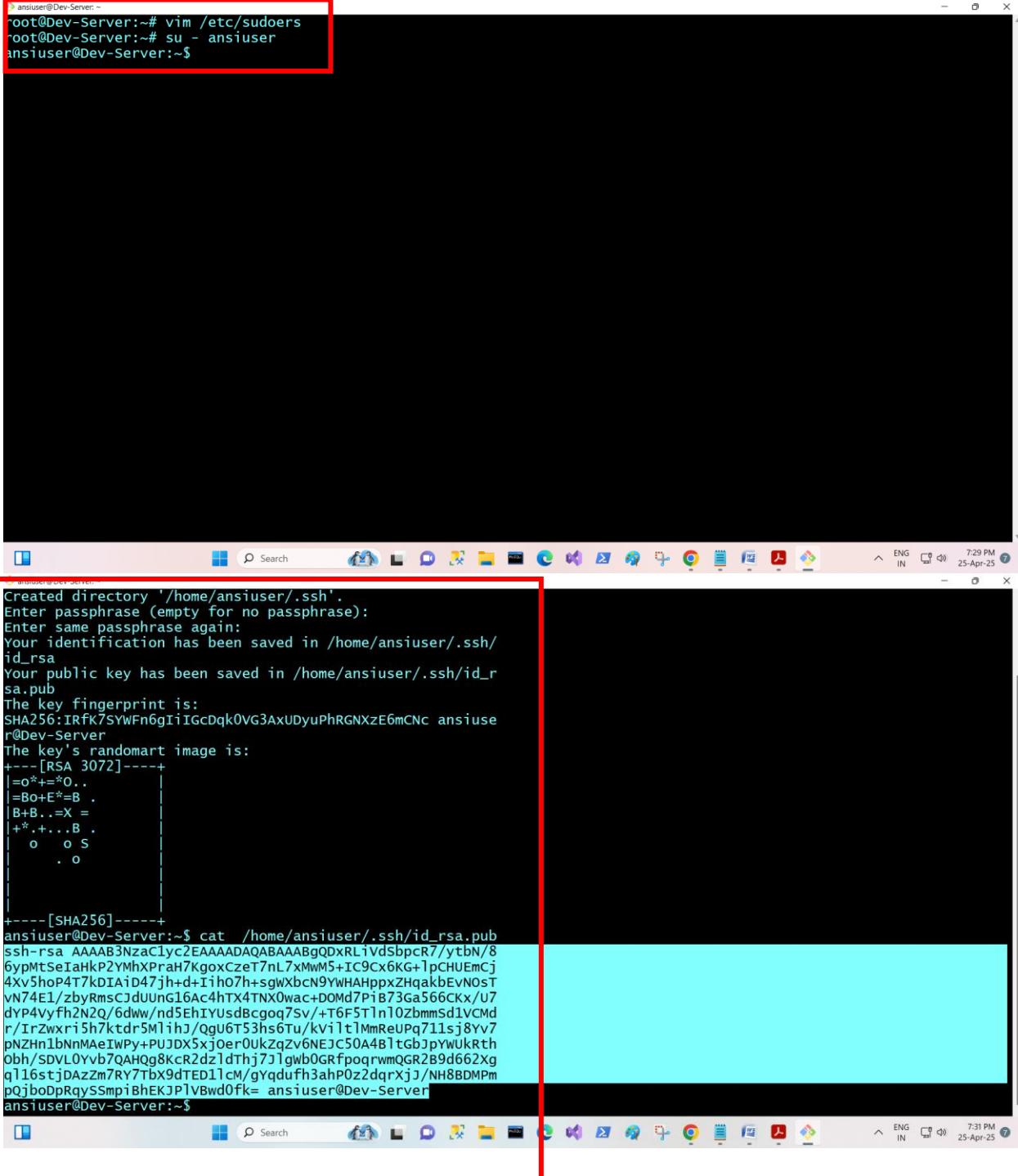
The screenshot shows the Docker Hub interface. In the top left, there's a dropdown menu with "anmol792" selected. Below it, a sidebar has "Repositories" highlighted. The main area is titled "Repositories" and shows two entries: "anmol792/myimage" and "anmol792/nginximage01". A red box highlights the entire list of repositories.

Name	Last Pushed	Contains	Visibility	Scout
anmol792/myimage	1 minute ago	IMAGE	Public	Inactive
anmol792/nginximage01	about 1 month ago	IMAGE	Public	Inactive

The screenshot shows a terminal window with the following command history:

```
root@Dev-Server:~# chmod -R 777 /var/run/docker.sock
root@Dev-Server:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
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



```
ansiuser@Dev-Server:~# vim /etc/sudoers
root@Dev-Server:~# su - ansiuser
ansiuser@Dev-Server:~$
```

```
Created directory '/home/ansiuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansiuser/.ssh/
id_rsa
Your public key has been saved in /home/ansiuser/.ssh/id_r
sa.pub
The key fingerprint is:
SHA256:IRfk7SWFn6gIiIGcDqk0VG3AxUDyuPhRGNXzE6mCnC ansius
e@Dev-Server
The key's randomart image is:
+---[RSA 3072]---+
|o*+=*O..
|=Bo+E*=B .
|B+B..=X =
+*+.+...B .
o o S
. o

+---[SHA256]---+
ansiuser@Dev-Server:~$ cat /home/ansiuser/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQDXRLiVdsbpcR7/ytbN/8
6ypMtSeIahkP2YmhxPrah7KgoxCzeT7nL7xmW5+IC9Cx6K+1pCHUEmCj
4xv5hoP4T7kdIAiD47jh+d+Iho7h+sgwXbcN9YWHAHppxZHqakbEvNOST
vn74El/zbyRmsCJduUng16Ac4htx4TNx0wac+DOMd7PiB73Ga566CKx/07
dyP4vyfh2NQ/6dw/nd5EhIYUsd8cgog7sv/+T6F5Tlnl0zbmmSd1Vcmd
r/IrZwxri5h7ktdr5Mlihj/QgU6T53hs6Tu/kviltlMmReUPq711sj8Yv7
pnZhnlbNmMaeIWPy+PUJDx5xjoeer0ukZqZv6NEjC50A48ltgbJpWUkRth
Obh/SDVL0Yvb7QAHQg8Kcr2dz1dTbj7Jlgwb0GRfpqrwmQGR2B9d662xg
q116stjdAzzm7RY7Tbx9dTED11cm/gYqdufh3ahP0z2dqrxjj/NH8BDMPm
pQibodPrqySSmpibhEKJP1Vbwdfk= ansiuser@Dev-Server
ansiuser@Dev-Server:~$
```

```
ansiuser@ip-10-0-1-152:~# adduser ansiuser
Adding user 'ansiuser' ...
Adding new group 'ansiuser' (1001) ...
Adding new user 'ansiuser' (1001) with group 'ansiuser' ...
Creating home directory '/home/ansiuser' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ansiuser
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []
Is the information correct? [Y/n] y
root@ip-10-0-1-152:~# vim /etc/sudoers
root@ip-10-0-1-152:# su - ansiuser
ansiuser@ip-10-0-1-152:$ mkdir .ssh
ansiuser@ip-10-0-1-152:$ ls -al
total 24
drwxr-x--- 3 ansiuser ansiuser 4096 Apr 25 14:04 .
drwxr-xr-x 4 root      root   4096 Apr 25 14:02 ..
-rw-r--r-- 1 ansiuser ansiuser 220 Apr 25 14:02 bash_logout
-rw-r--r-- 1 ansiuser ansiuser 3771 Apr 25 14:02 bashrc
-rw-r--r-- 1 ansiuser ansiuser 807 Apr 25 14:02 profile
drwxrwxr-x 2 ansiuser ansiuser 4096 Apr 25 14:04 .ssh
ansiuser@ip-10-0-1-152:$ echo "ssh-rsa AAAAB3nzaC1yc2EAAAQABAAQBgQDXRLivdSbpcR7/ybtN/86yptSeIaHKP2YMXPrAh7KgoxczET7L7XMM5+IC9Cx6Kg+1pCHUEmCj4xv5hoP4T7kDIAid47jh+d+Iih07h+sgwXbcN9YWHAHppxZHqakbEvNoSTvN74e1/zbyRmscJduUNG16Ac4hTx4TNX0wac+DOMd7P1B73Ga566Ckx/U7dyP4vyfh2NzQ/6dwv/nd5EhTYusdbcgog7sv/+t6F5Tlnl0Zbmmsd1VCMdr/Irzwxr1sh7ktdrSM1h1/qgu6t53hs6Tu/kv1it1MmrreUpq71lsj8YY7pnZhn1bnRMaeIPy+PUJDx5joeR0UkZqzV6NEJC50A4BltcbJpyWUkrthobh/SDVL0rvb7QAHQq8Kcr2dz1dhj7J1gwB0GRfpodqrwmQGR2B9d62Xgq1l6stjDAZm7RY7tbX9dTEd1lCM/gyduduh3ahp022dqrXjJ/NH8BDMPmpqjb0bpRqysSmplbhEKJPlVbw0fk= ansiuser@Dev-Server" >> ~/.ssh/authORIZED_KEYS
ansiuser@ip-10-0-1-152:~$
```

## Step12: Create the inventory file and ansible.cfg file

```
ansiuser@Dev-Server:~
[webserver]
54.162.163.19/
```

```
[defaults]
inventory = /root/myinventory
```

-- INSERT --

2,30

All



```
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**

```
ansiuser@Dev-Server:~$ ls
ansible.cfg myinventory playbook-test-server.yml
ansiuser@Dev-Server:~$ ansible-playbook playbook-test-server.yml
ERROR! The playbook: playbook-test-server could not be found
ansiuser@Dev-Server:~$ ansible-playbook playbook-test-server.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 installation 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]

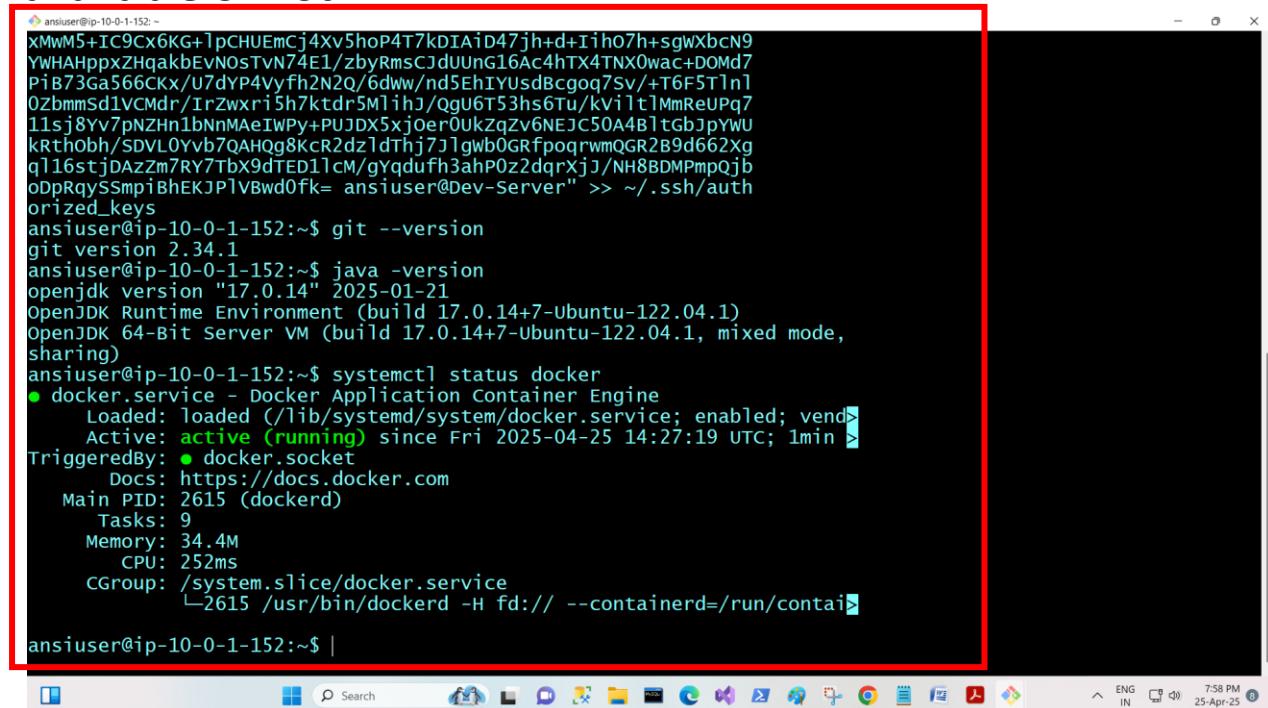
TASK [Install multiple packages] ****
ok: [54.162.163.197] => (item=nginx)
changed: [54.162.163.197] => (item=docker.io)
changed: [54.162.163.197] => (item=openjdk-17-jdk)

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

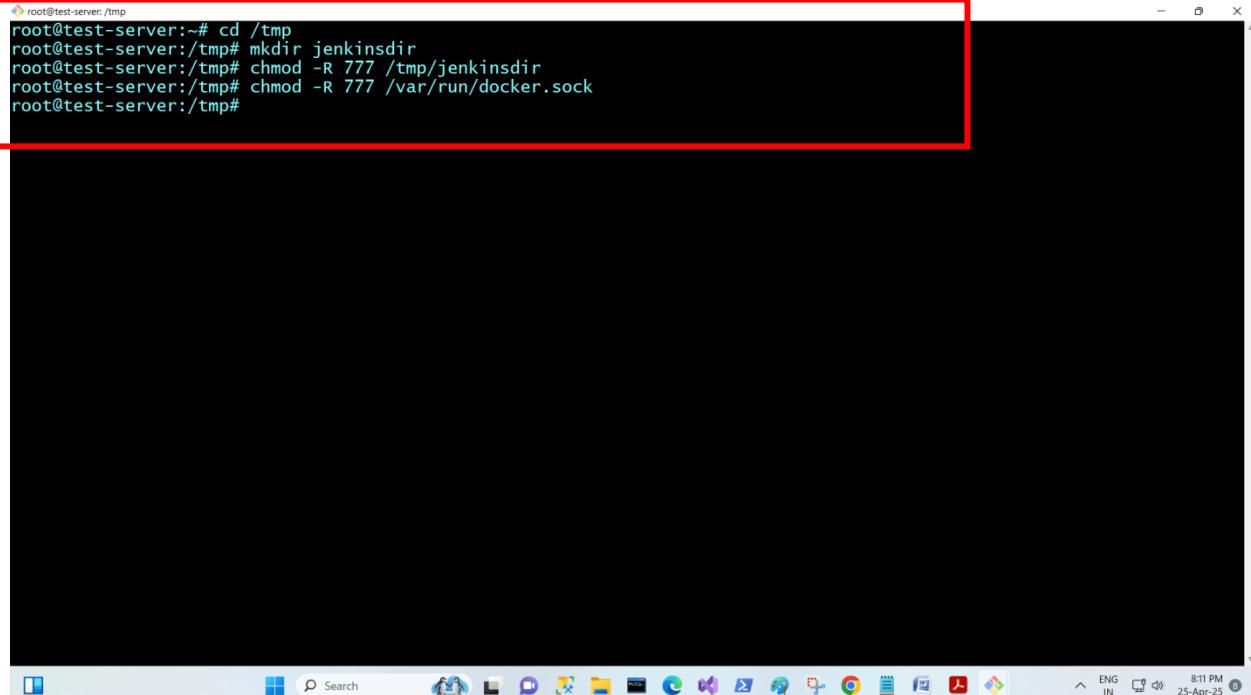
ansiuser@Dev-Server:~$ |
```

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



```
xMwW5+IC9Cx6KG+1pCHUEmCj4Xv5hoP4T7kDIAiD47jh+d+Iih07h+sgWxbCN9
YWHAHppxZHqakbEvN0sTVN74E1/zbyRmsCJdUUUnG16Ac4hTX4TNX0wac+DOMd7
PiB73Ga566CKx/U7dYP4Vvhf2N2Q/6dww/nd5Eh1YUsd8cg0q7Sv/+T6F5Tlnl
0zbmmSd1VCMdr/IrZwxri5h7ktdr5M1ihj/QgU6T53hs6Tu/kvItlMmReUPq7
11sj8Yv7pnZhn1bNmMaeIWPy+PUJDX5xjoer0ukZqv6NEJC50A4BltgbjpYwU
krthobh/SDVL0Yvb7QAHQg8KcR2dzldThj7jlgb0GRfpqqrwmQGR2B9d662Xg
q116stjDAZm7RY7Tbx9dTd1lcM/gYqdufh3ahp0z2dqrXjj/NH8BDMPmpQjb
oDpRqySSmpibhEKJP1VBwd0fk= ansiuser@Dev-Server" >> ~/.ssh/auth
orized_keys
ansiuser@ip-10-0-1-152:~$ git --version
git version 2.34.1
ansiuser@ip-10-0-1-152:~$ java -version
openjdk version "17.0.14" 2025-01-21
OpenJDK Runtime Environment (build 17.0.14+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.14+7-Ubuntu-122.04.1, mixed mode,
sharing)
ansiuser@ip-10-0-1-152:~$ systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vend>
  Active: active (running) since Fri 2025-04-25 14:27:19 UTC; 1min >
TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
  Main PID: 2615 (dockerd)
    Tasks: 9
   Memory: 34.4M
      CPU: 252ms
     CGroup: /system.slice/docker.service
             └─2615 /usr/bin/dockerd -H fd:// --containerd=/run/contai>
ansiuser@ip-10-0-1-152:~$ |
```

## Step15: create Jenkins root directory



```
root@test-server:~# cd /tmp
root@test-server:/tmp# mkdir jenkinsdir
root@test-server:/tmp# chmod -R 777 /tmp/jenkinsdir
root@test-server:/tmp# chmod -R 777 /var/run/docker.sock
root@test-server:/tmp#
```

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

The image shows two screenshots of the Jenkins interface. The top screenshot displays the 'Create New Item' page for a new node named 'testserver'. It includes fields for Name, Description, Number of executors (set to 1), and Remote root directory (/tmp/jenkinsdir). The bottom screenshot shows the 'Edit Node' page for the 'testserver' node, which has been assigned a label 'test\_server' and a usage restriction 'Only build jobs with label expressions matching this node'. The 'Launch method' is set to 'Launch agents via SSH', and the 'Host' is specified as 54.162.163.197. Both screenshots have a red box highlighting the configuration details.

**Create New Item (Top Screenshot):**

- Name: testserver
- Description:
- Number of executors: 1
- Remote root directory: /tmp/jenkinsdir

**Edit Node (Bottom Screenshot):**

- Remote root directory: /tmp/jenkinsdir
- Labels: test\_server
- Usage: Only build jobs with label expressions matching this node
- Launch method: Launch agents via SSH
- Host: 54.162.163.197
- Credentials: - none -

Dashboard > Manage Jenkins > Nodes >

Launch method: Launch agents via SSH

Host: 54.162.163.197

Credentials: ubuntu (TestServercredentials)

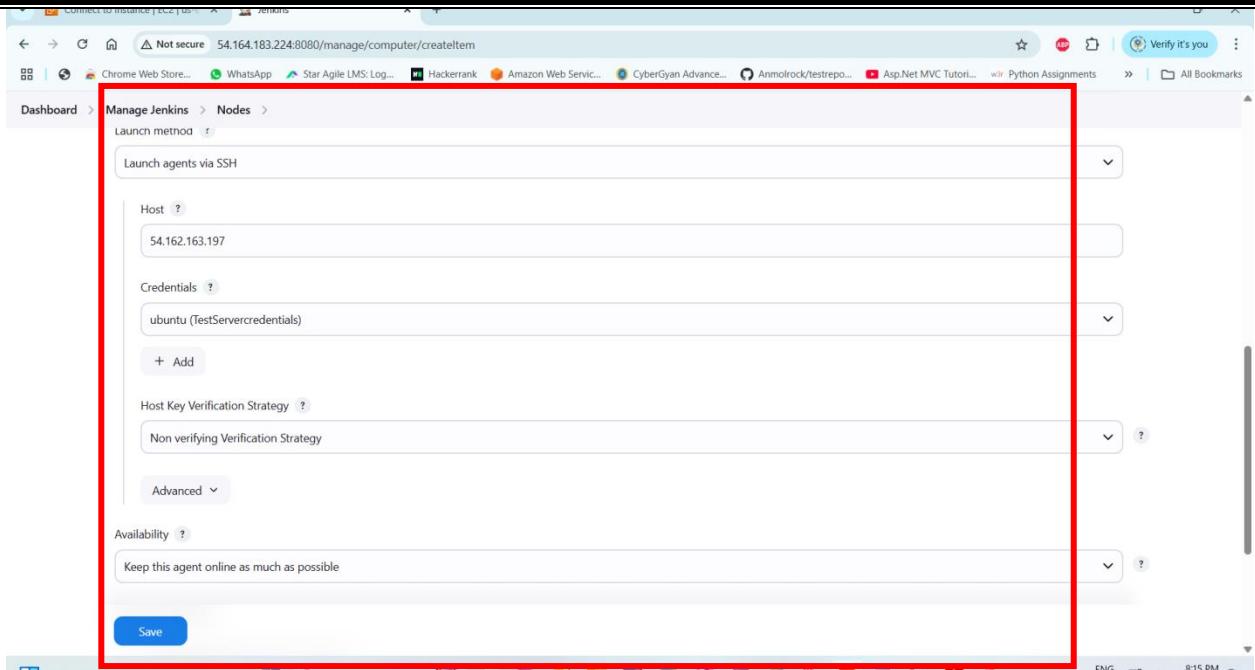
+ Add

Host Key Verification Strategy: Non verifying Verification Strategy

Advanced

Availability: Keep this agent online as much as possible

Save



Dashboard > Manage Jenkins > Nodes >

Usage: Only

Jenkins Credentials Provider: Jenkins

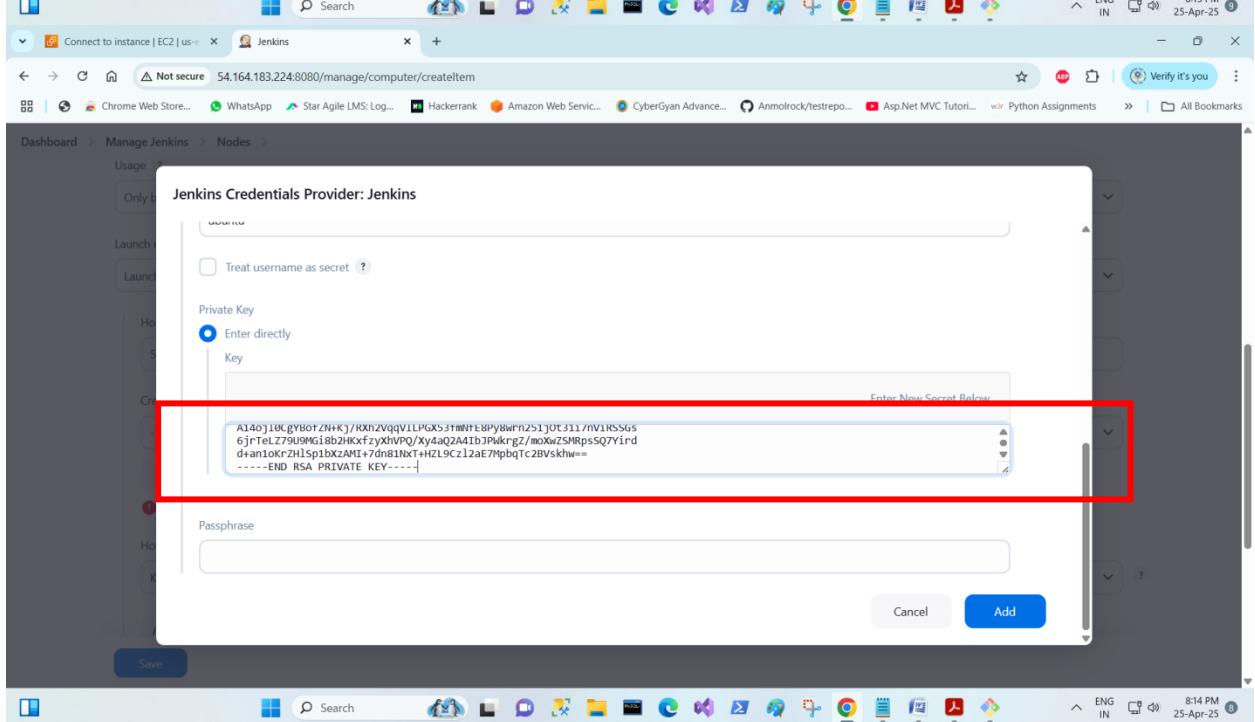
Treat username as secret:

Private Key: Enter directly

Key: A140j10KgYbotZN+KJ/kXn2VqgV1LPGX53tMNT8Py8wfn251j0t311/nV1RSSG56jrtElZ79U9MGmibzb2HkfxyxhvPQ/Xy4aQ2a41b7Pwkrpz/moxWzSMRpsSQ7Yirdd+an1oKrZH1Sp1bxzAM1+dnd81NXT+HZL9Cz12afE7MpbqTc2BVskhw==  
-----END RSA PRIVATE KEY-----

Passphrase:

Cancel Add



The screenshot shows the Jenkins interface for managing nodes. The top navigation bar includes links for 'Nodes [Jenkins]' and 'Manage Jenkins'. The main content area is titled 'Nodes' and displays a table of build nodes. The columns are: S (Status), Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, Free Temp Space, and Response Time. A 'New Node' button is located at the top right of the table header. On the left side, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (two entries: 'Built-In Node' and 'testserver'). A legend at the bottom right indicates that the 'S' icon represents a Slave node.

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 GB	0 B	2.91 GB	0ms
	testserver		N/A	N/A	N/A	N/A	N/A
	Data obtained		4 ms	4 ms	3 ms	2 ms	1 ms

Icon: S M L Legend

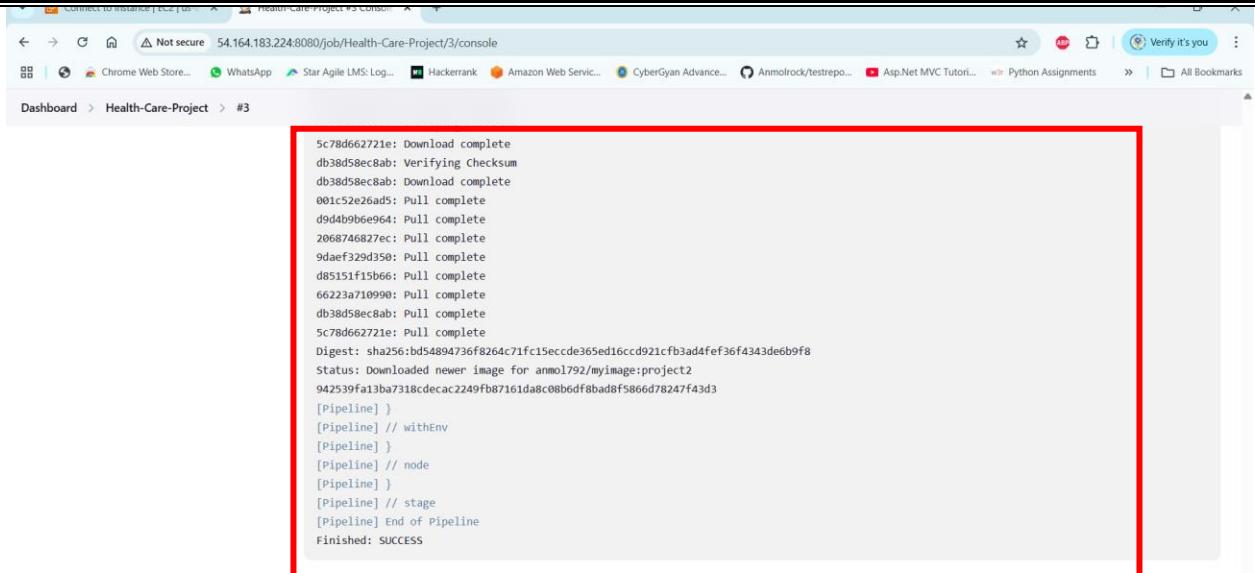
The screenshot shows a web browser window with the URL <https://54.164.183.224:8080/job/Health-Care-Project/configure>. The page title is "Health-Care-Project Config [jen]".

The main content area displays a Groovy script for a Jenkins pipeline:

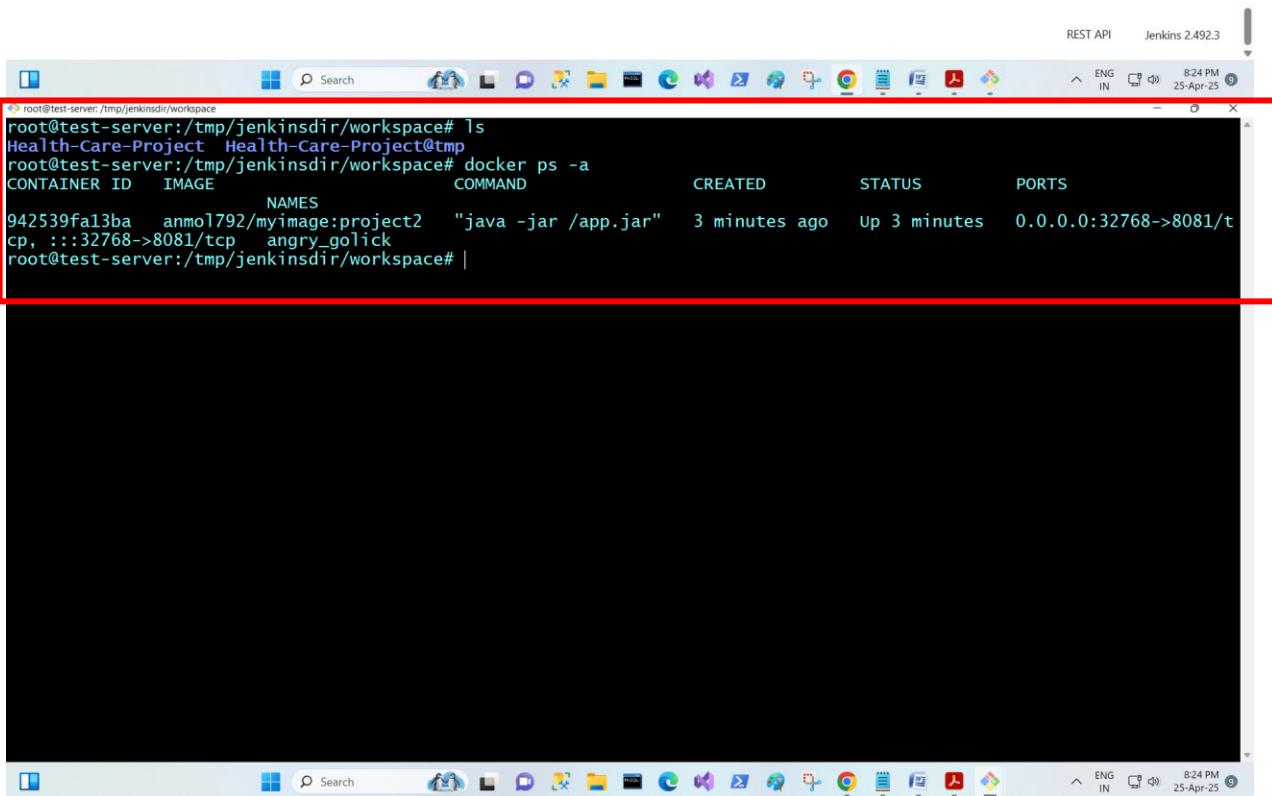
```
40      sh 'docker push amol1792/myimage:project2'
41    }
42  }
43 stage('deploy on Test server')
44 {
45   agent {
46     label 'test_server'
47   }
48   steps{
49     sh 'docker run -d -P amol1792/myimage:project2'
50   }
51 }
52
53
54
```

A red box highlights the entire code block. At the bottom of the code editor, there is a checkbox labeled "Use Groovy Sandbox" with a checked status.

At the bottom of the page, there are "Save" and "Apply" buttons.



```
5c78d662721e: Download complete
db38d58ec8ab: Verifying Checksum
db38d58ec8ab: Download complete
001c52e26ad5: Pull complete
d9d4b9b66964: Pull complete
2068746827ec: Pull complete
9daef3299d350: Pull complete
d85151f15b66: Pull complete
66223a710990: Pull complete
db38d58ec8ab: Pull complete
5c78d662721e: Pull complete
Digest: sha256:bd54894736f8264c71fc15eccde365ed16cc921cfb3ad4fef36f4343de6b9f8
Status: Downloaded newer image for ammol792/myimage:project2
942539fa13ba7318cdecac224fb87161da8c08b6df8bad8f5866d78247f43d3
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline]
[Pipeline] // stage
[Pipeline] End of Pipeline
Finished: SUCCESS
```



```
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   ammol792/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**

```
root@prod-server:~# kubectl get nodes
NAME        STATUS    ROLES      AGE   VERSION
dev-server  Ready     control-plane  4m59s  v1.29.15
test-server Ready     <none>     2m12s  v1.29.15
root@prod-server:~#
```

```
root@prod-server:~  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: mydeploy  
spec:  
  replicas: 3  
  selector:  
    matchLabels:  
      type: webserver  
  template:  
    metadata:  
      labels:  
        type: webserver  
    spec:  
      containers:  
        - name: c1  
          image: anmol1792/myimage:project2
```

```
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.yaml
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-1xmwj   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-1xmwj      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
service/mysvc1 created
root@ProdServer:~# kubectl create -f service.yml
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-1xmwj      1/1    Running   0          2m41s

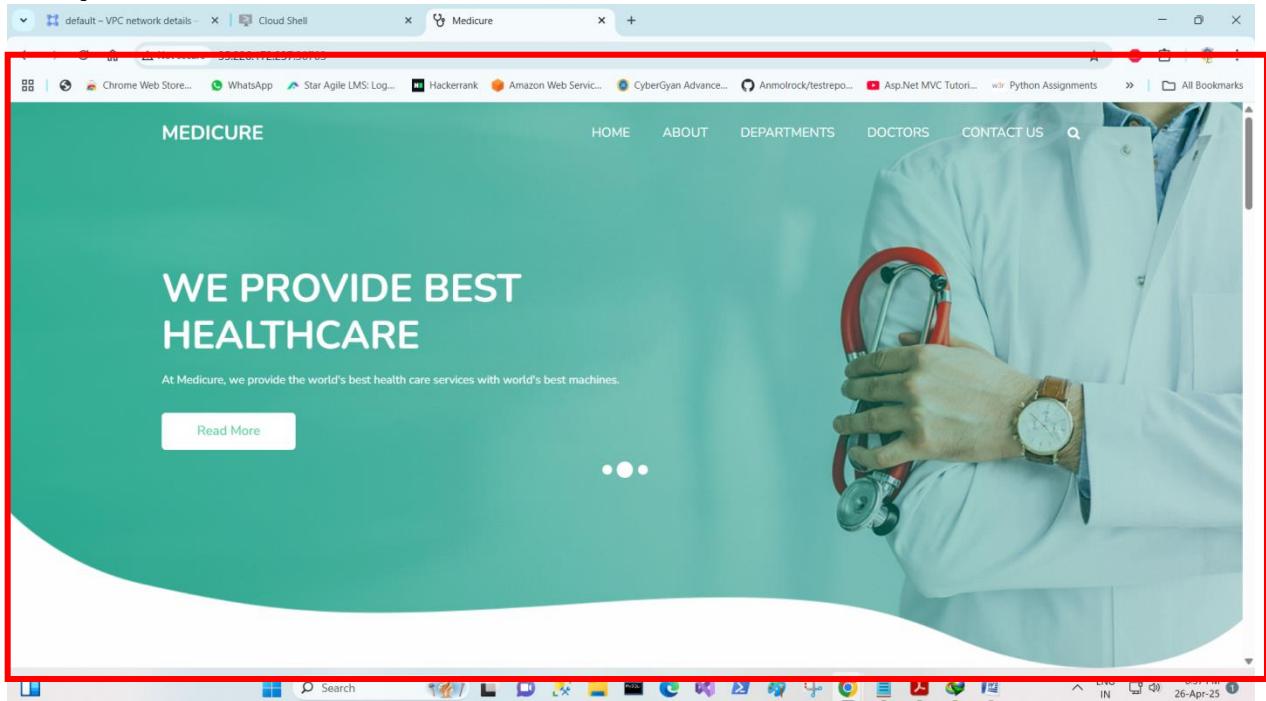
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes  ClusterIP  10.96.0.1   <none>        443/TCP   15m
service/mysvc1        NodePort   10.108.21.23 <none>        8081:32194/TCP 10s

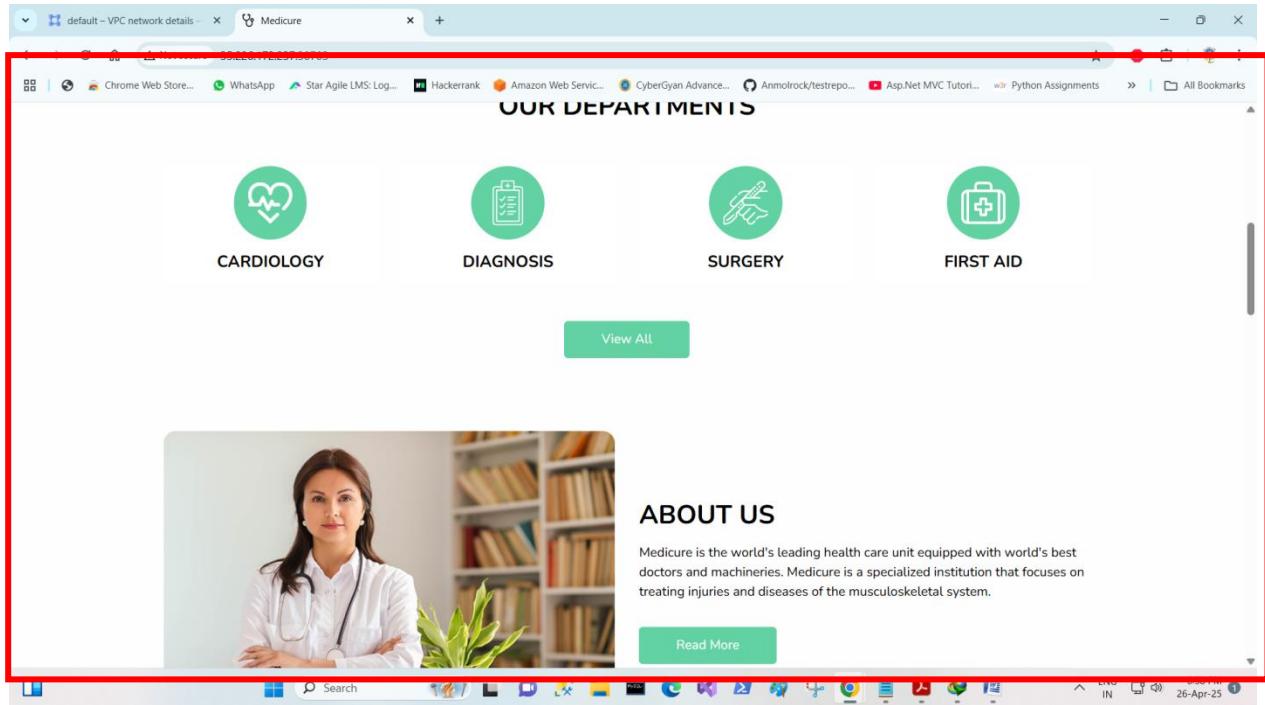
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeploy  3/3     3           3           2m41s

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 →





## **Step18: Monitoring Using Prometheus and Grafana**

The screenshot shows the Prometheus Query interface. At the top, there's a search bar with the URL "54.163.9.153:9090/query". Below the search bar are three tabs: "Table" (selected), "Graph", and "Explain". A large text input field says "Enter expression (press Shift+Enter for newlines)". To the right of the input field is a blue "Execute" button. Below the input field, it says "No data queried yet". At the bottom left is a blue "+ Add query" button.

The screenshot shows the Prometheus Status > Target health interface. At the top, there are three dropdown menus: "Select scrape pool", "Filter by target health", and "Filter by endpoint or labels". Below these are two sections of tables:

Jenkins_Server		Last scrape	State
Endpoint	Labels		
http://18.204.5.140:8080/prometheus	instance="54.164.183.224:8080" job="Jenkins_Server"	8.653s ago	21ms UP

prometheus		Last scrape	State
Endpoint	Labels		
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:de...

Apr 22 09:03:24 ip-172-31-30-193 grafana[2649]: logger=migrator t=2025-04-22T09:03:24.753944Z level=info msg="Migration successfully executed" id="create_index"
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_table"
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_login_attempt_table"
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_table"
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_user_auth_table"
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_user_id"
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_index_IDX_user_id"

```
1 lines 1-21/21 (END)

```

The screenshot shows the Grafana interface for managing data sources. The URL is 54.157.244.211:3000/connections/datasources/edit/fejnza1ko3aioe. The page displays configuration options for a 'prometheus' data source, including 'Incremental querying (beta)', 'Disable recording rules (beta)', and various 'Other' settings like 'Custom query parameters' (with an example: max\_source\_resolution=5m&timeout), 'HTTP method' (set to POST), and 'Use series endpoint'. Below these are sections for 'Exemplars' and 'Annotations'. A prominent green success message at the bottom states: 'Successfully queried the Prometheus API.' It also says 'Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.' At the bottom are 'Delete' and 'Save & test' buttons.

Not secure 54.157.244.211:3000/dashboard/import

Home > Dashboards > 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.

Change uid

Prometheus prometheus1

Import Cancel

A screenshot of a web browser showing the Grafana import dashboard interface. The URL is 54.157.244.211:3000/dashboard/import. The page title is "Importing dashboard from Grafana.com". It shows details about the dashboard being imported: "Published by haryan" and "Updated on 2023-08-24 15:04:53". Under "Options", there are fields for "Name" (set to "Jenkins: Performance and Health Overview") and "Folder" (set to "Dashboards"). A note about "Unique identifier (UID)" explains its purpose for consistency across multiple installations. Below these are dropdown menus for "Prometheus" (set to "prometheus1") and "Import" and "Cancel" buttons. The entire form area is highlighted with a red rectangle.

