Certification Project – Medicure Healthcare Domain

Submitted by – Mohammed Abu Rihan

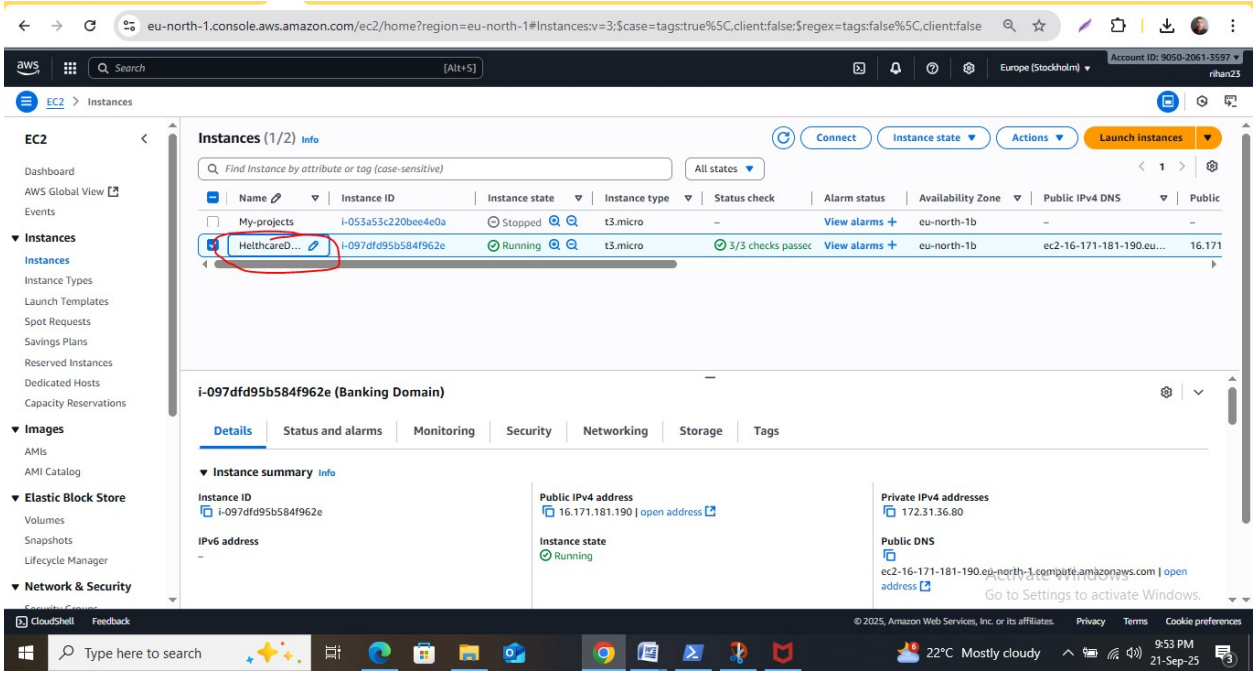
Submitted to – Vikul mentor

Batch no - DevOpsSA2504024

Date of submission – 24-09-2025

Medicure Healthcare Domain

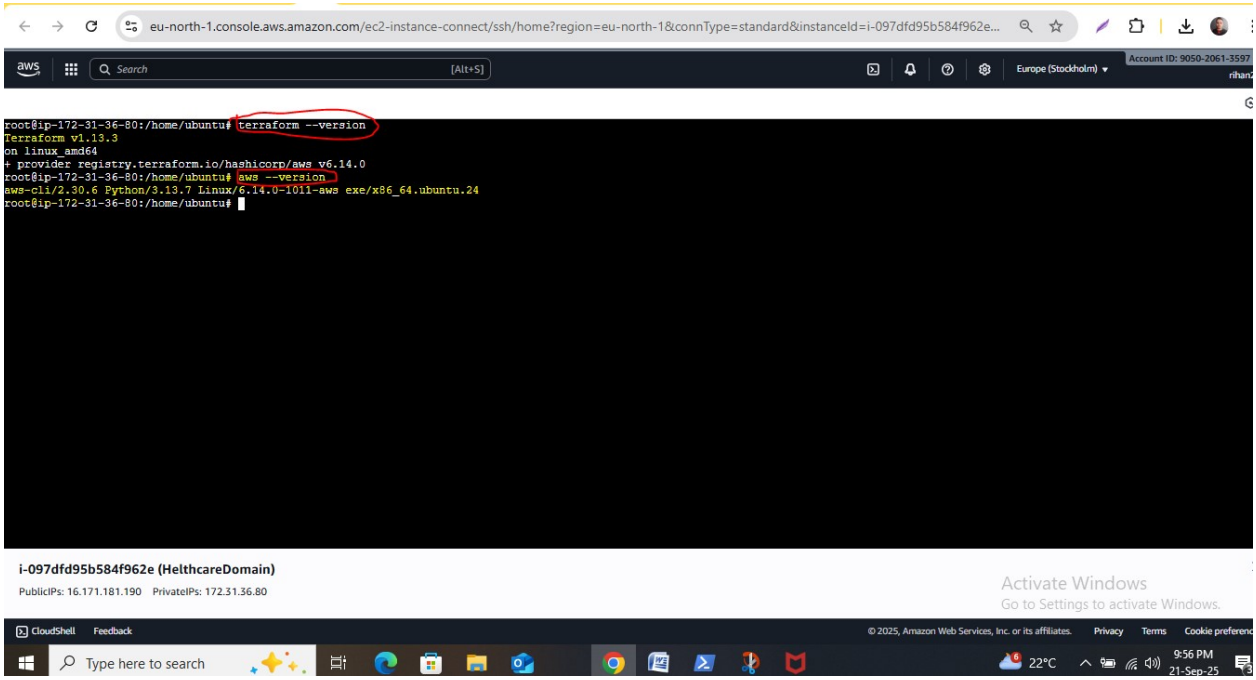
Step 1: Launch a EC2 ubuntu instance and connect it to a terminal



Step 2: Install Terraform and aws cli using the installation command

curl -fsSL https://apt.releases.hashicorp.com/gpg | gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(. /etc/os-release; echo $VERSION_CODENAME) main" \
| tee /etc/apt/sources.list.d/hashicorp.list
apt update -y && apt install -y terraform

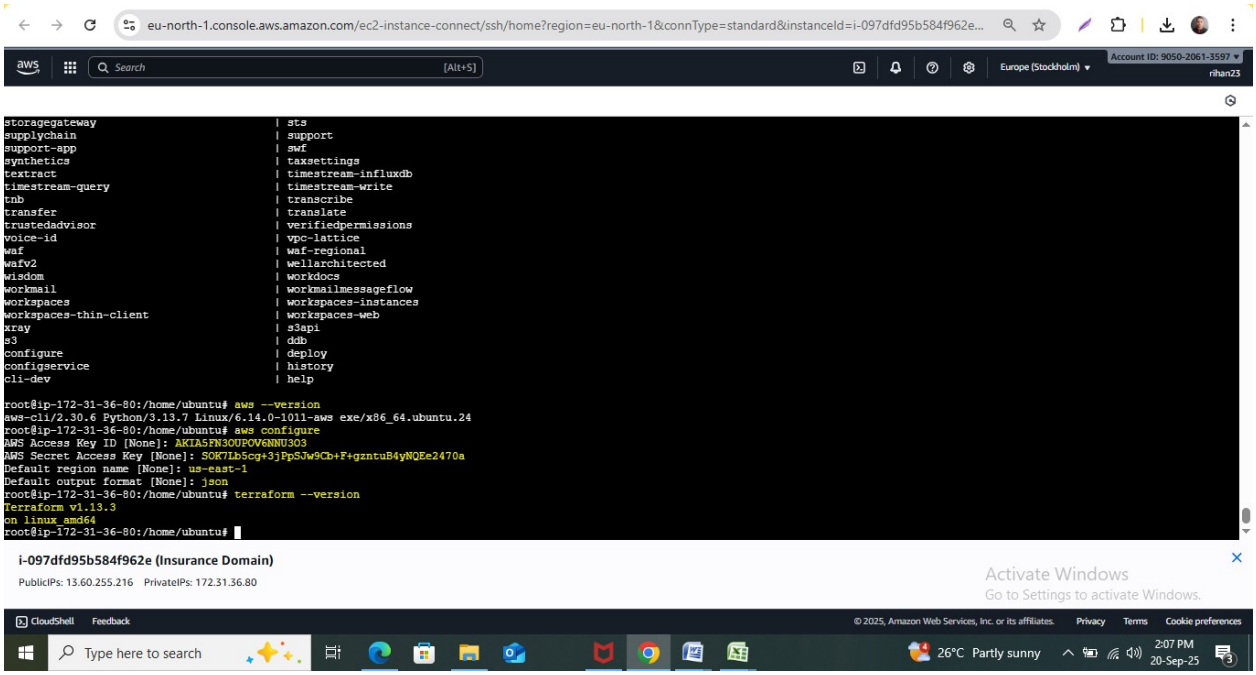
apt update –y
sudo apt install -y unzip
curl -sS "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o awscliv2.zip
sudo apt update && sudo apt install unzip –y
unzip awscliv2.zip
sudo ./aws/install
aws –version



Step 3: We need to configured AWS Access Key and Secret access keys for that first we create an IAM user with admin access. Configure it in aws

Steps to generate access keys and secrete access

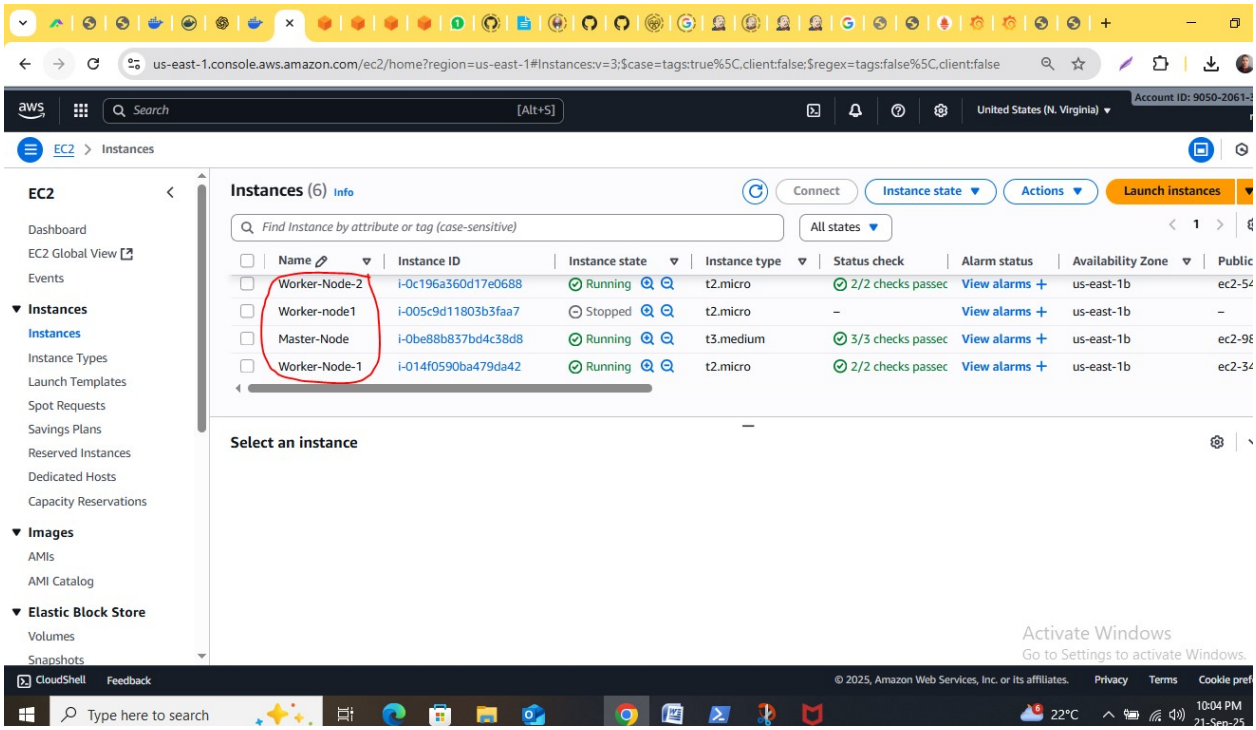
click on account id → My security credentials →create access keys



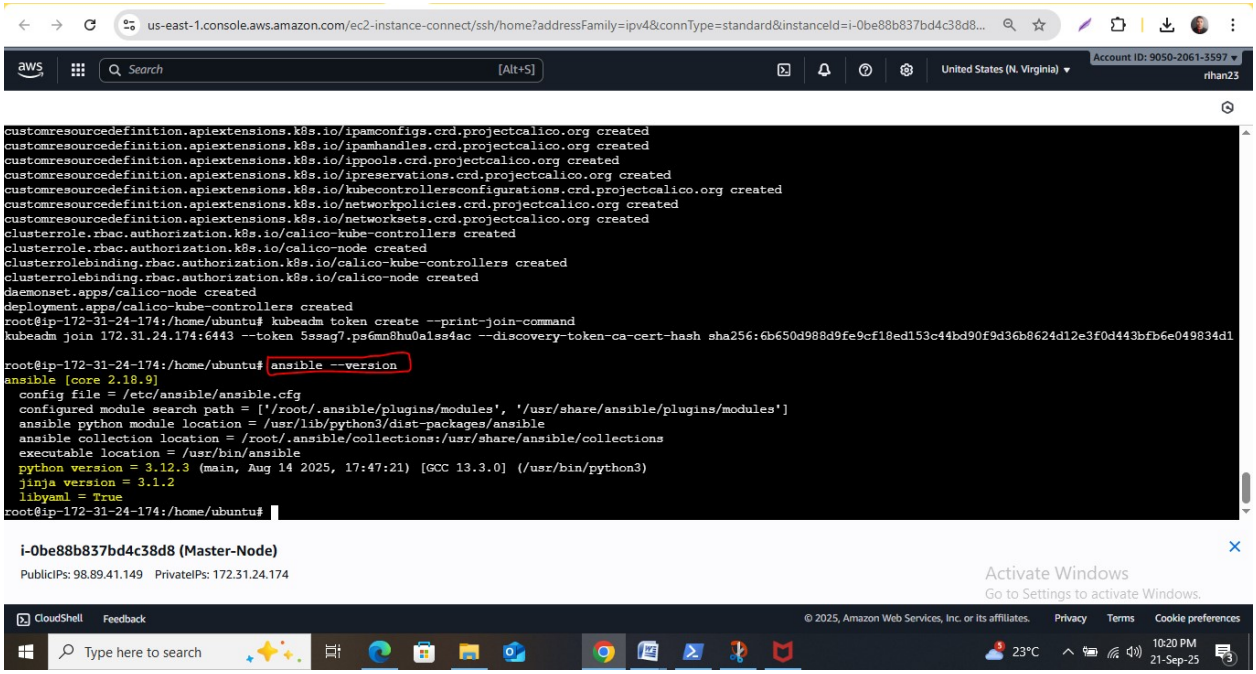
Step 4: Create Terraform file (**main.tf)** to write the command to create the instances

```
provider "aws" {
region = "us-east-1"
}
# Get default VPC
data "aws_vpc" "default" {
default = true
}
# Security Group: SSH
resource "aws_security_group" "ssh_sg" {
name = "ssh-sg"
description = "Allow SSH"
vpc_id = data.aws_vpc.default.id
ingress {
from_port = 22
to_port = 22
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"]
```

```
}
}
# Security Group: All Traffic
resource "aws_security_group" "all_traffic_sg" {
name = "all-traffic-sg"
description = "Allow all inbound and outbound"
vpc_id = data.aws_vpc.default.id
ingress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
# Master Instance (t3.medium)
resource "aws_instance" "master" {
ami = "ami-0360c520857e3138f"
instance_type = "t3.medium"
key_name = "usnv1"
vpc_security_group_ids = [
aws_security_group.ssh_sg.id,
aws_security_group.all_traffic_sg.id
]
tags = {
Name = "Master-Node"
}
}
# Worker Instances (2 × t2.micro)
resource "aws_instance" "workers" {
ami = "ami-0360c520857e3138f"
instance_type = "t2.micro"
key_name = "usnv1"
count = 2
vpc_security_group_ids = [
aws_security_group.ssh_sg.id,
aws_security_group.all_traffic_sg.id
]
tags = {
Name = "Worker-Node-${count.index + 1}"
}
}
```

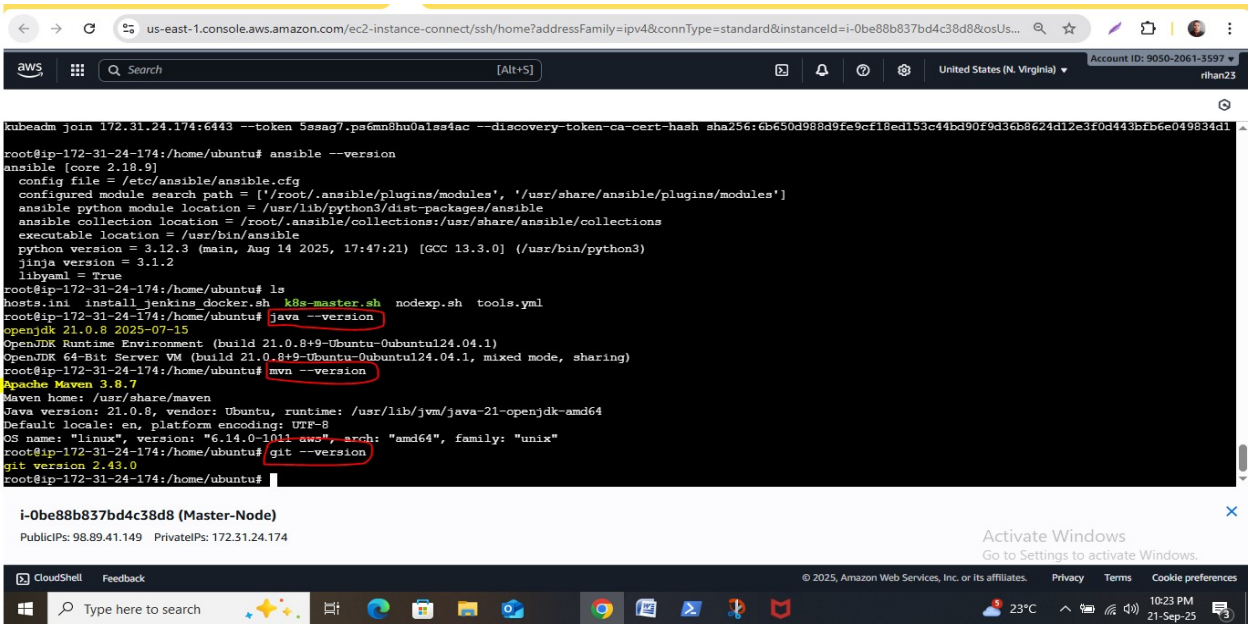


Step 5: Install ansible and Create ansible playbook and install Git, Java, Maven installation command



Creating playbook

```
- name: Install Git, Java, Maven, and Docker
hosts: all
become: yes

tasks:
apt:
update_cache: yes

apt:
name: git
state: present

apt:
name: openjdk-17-jdk
state: present

apt:
name: maven
state: present

apt:
name: "{{ item }}"
state: present
loop:
- apt-transport-https
- ca-certificates
- curl
- software-properties-common
- gnupg-agent

- name: Add Docker GPG key
apt_key:
url: https://download.docker.com/linux/ubuntu/gpg
state: present

- name: Add Docker repository
apt_repository:
repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable
state: present

- name: Install Docker
apt:
name: docker-ce
state: present
update_cache: yes

- name: Enable and start Docker service
systemd:
name: docker
enabled: yes
state: started
```



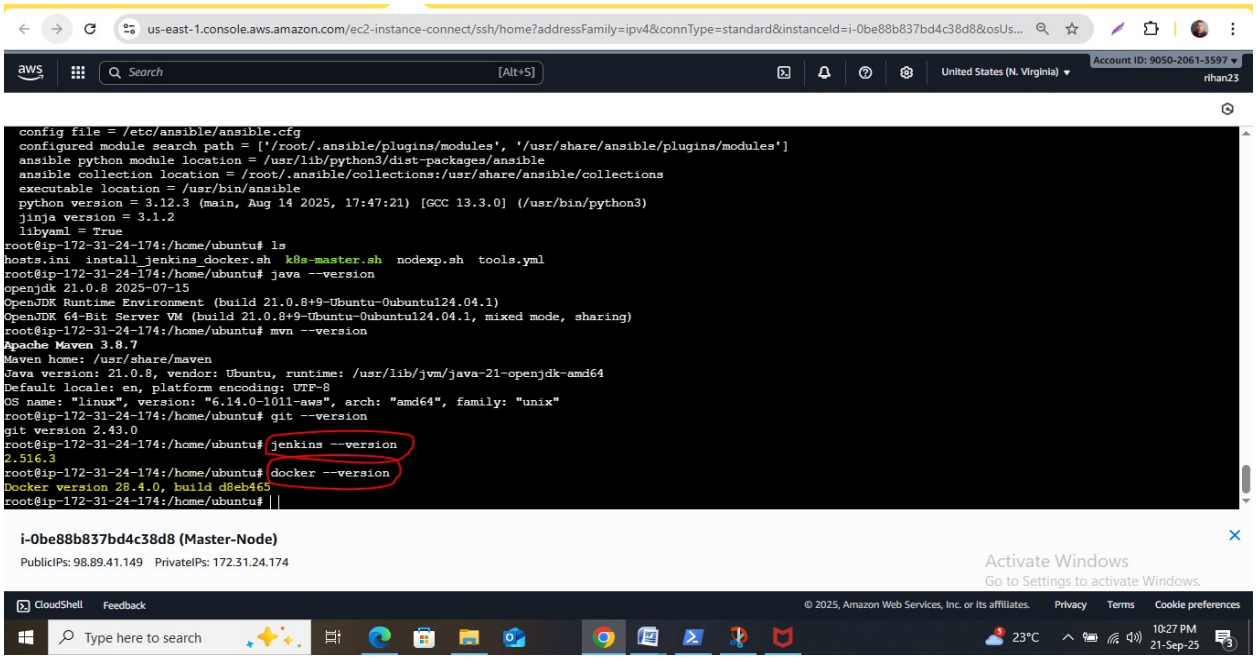
Step 6: Install Jenkins and docker using command

apt install docker-compose –y

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

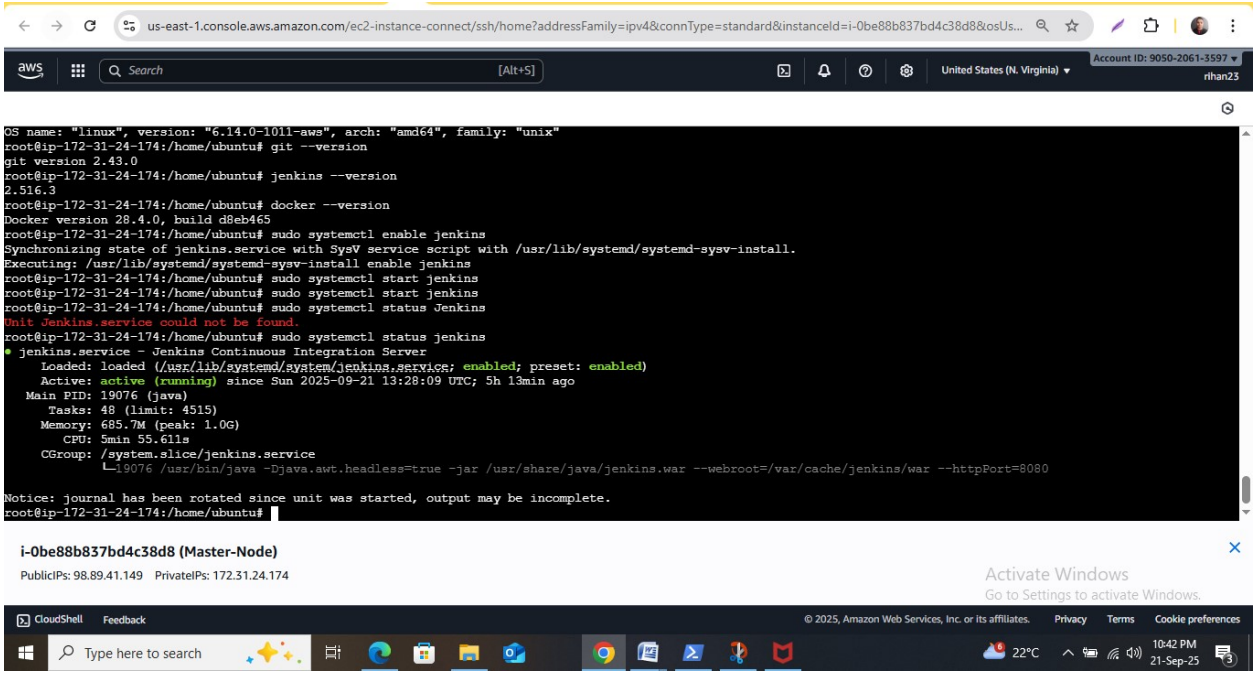
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update
sudo apt-get install jenkins –y
Jenkins –version

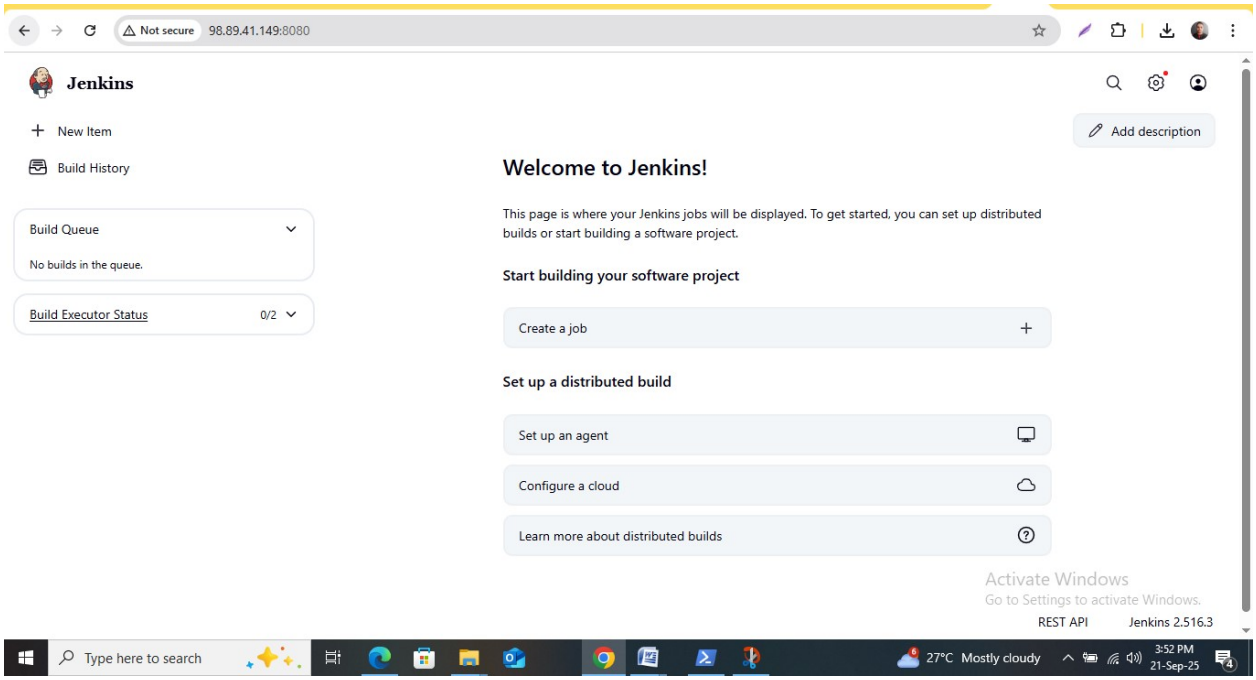


to installing Jenkins plugins we need to execute commands

sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status Jenkins

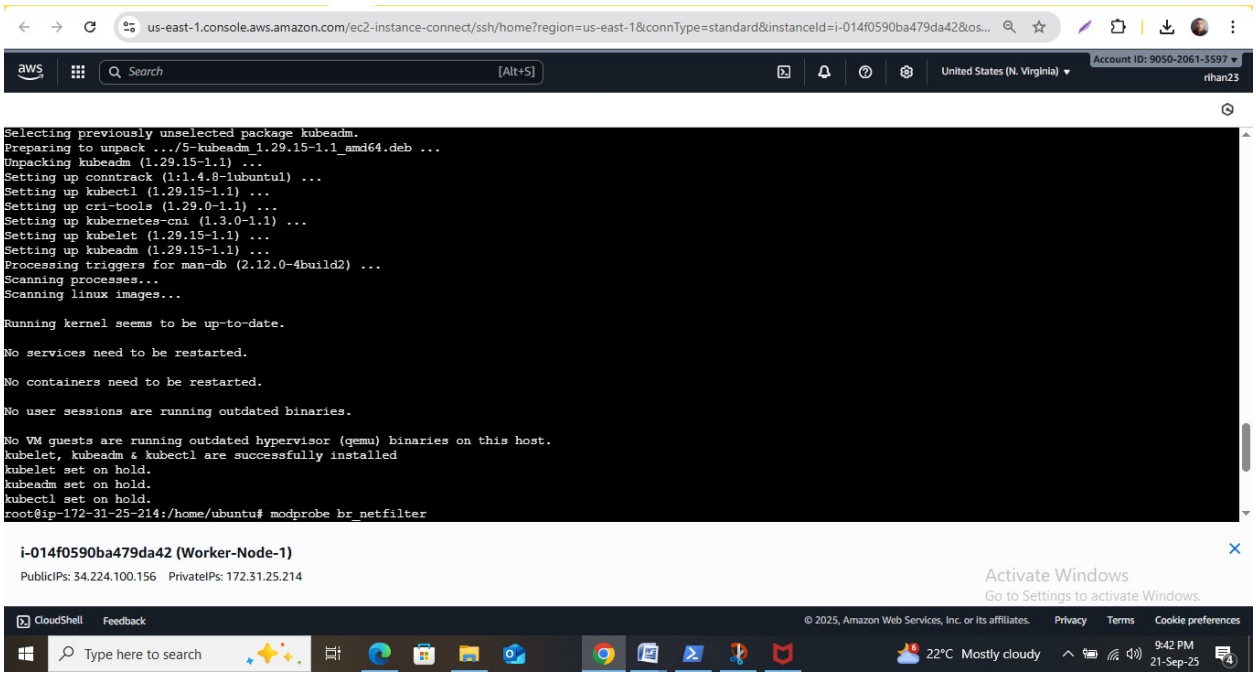


Installed Jenkins plugins now Jenkins is ready to use



Step 7: Installing K8s using installation commands

wget https://raw.githubusercontent.com/akshu20791/Deployment-script/main/k8s-master.sh
ls -l
ls
chmod 777 k8s-master.sh
ls -l
./k8s-master.sh



Similarly copy the commands in worker node to connect it with master

wget https://raw.githubusercontent.com/akshu20791/Deployment-script/main/k8s-nodes.sh

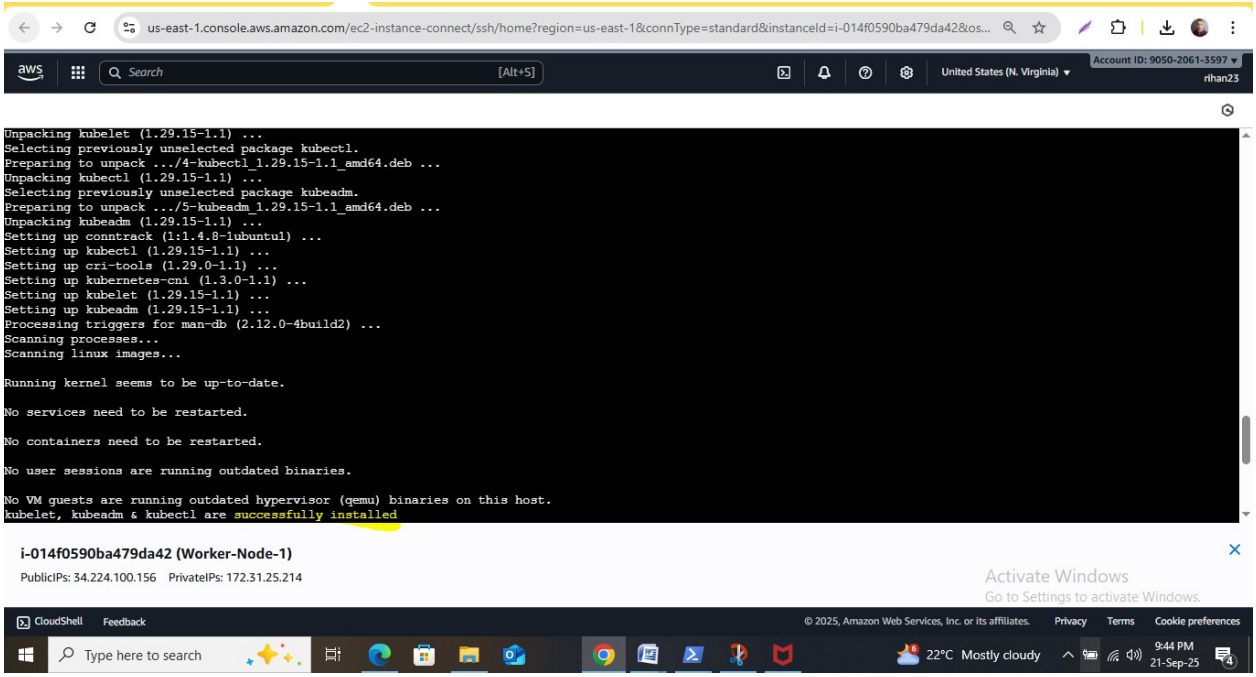
ls

ls -l

chmod 777 k8s-nodes.sh

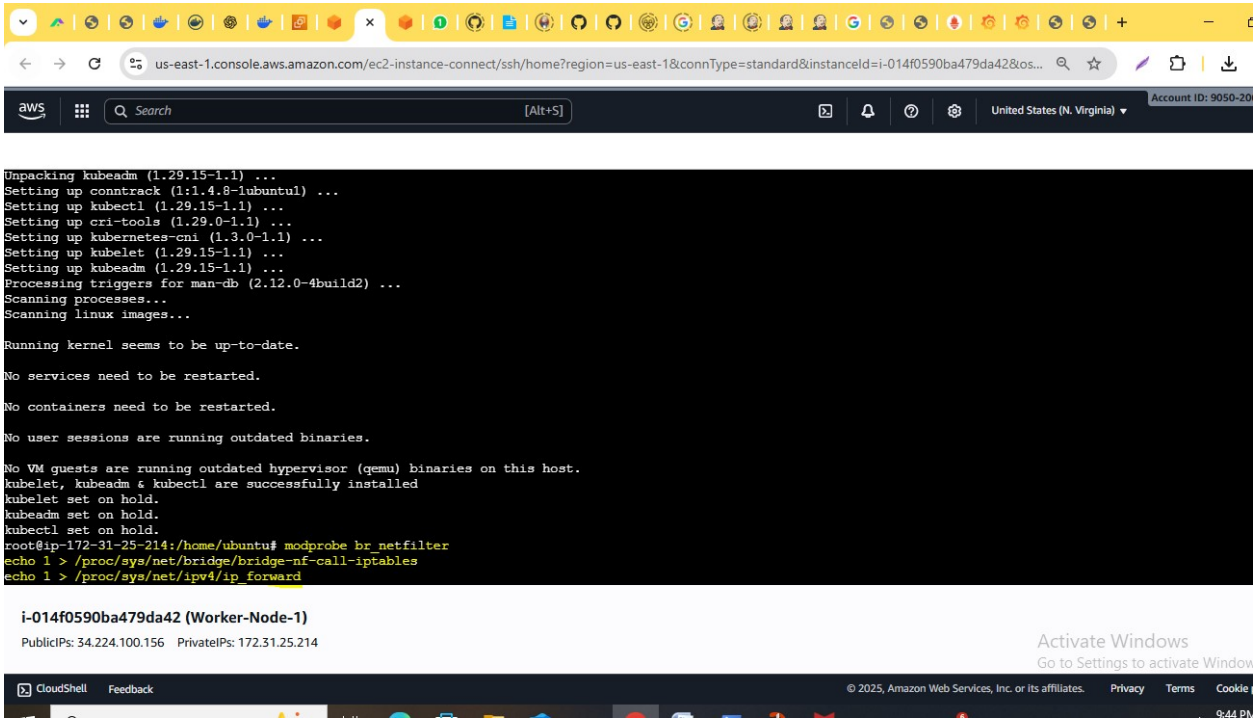
ls –l

./k8s-nodes.sh

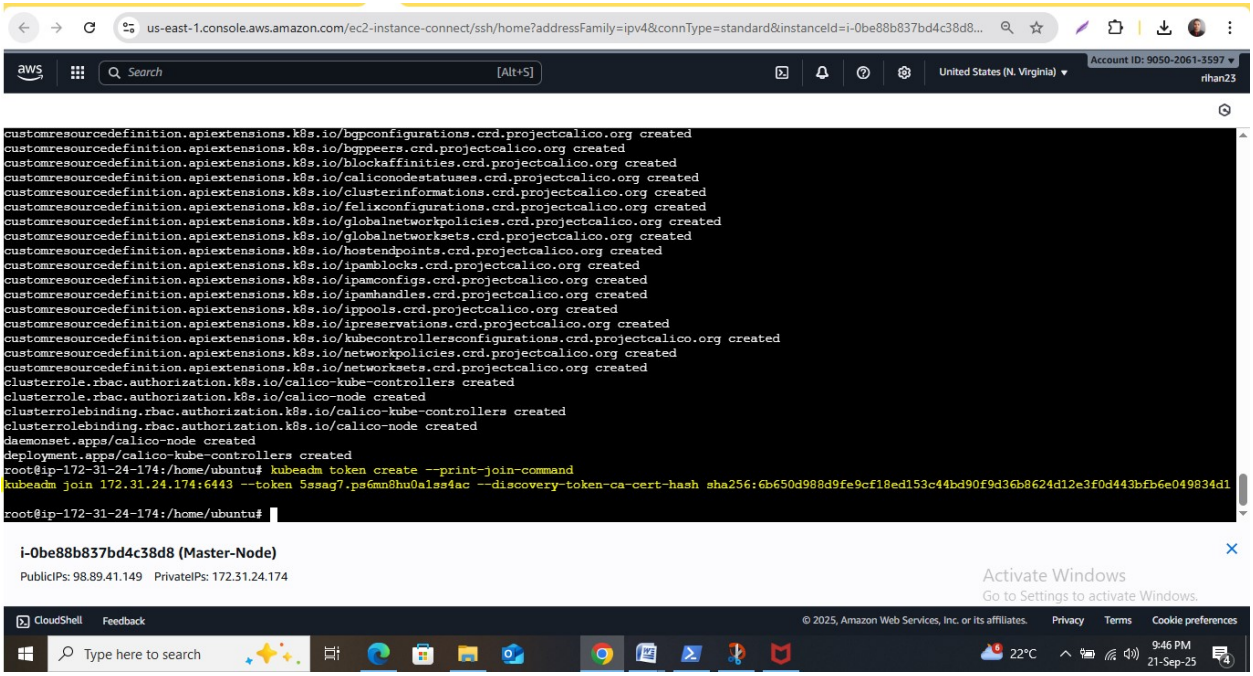


To configure and run the commands in nodes, we need to establish networking so we write commands →

modprobe br_netfilter
echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
echo 1 > /proc/sys/net/ipv4/ip_forward



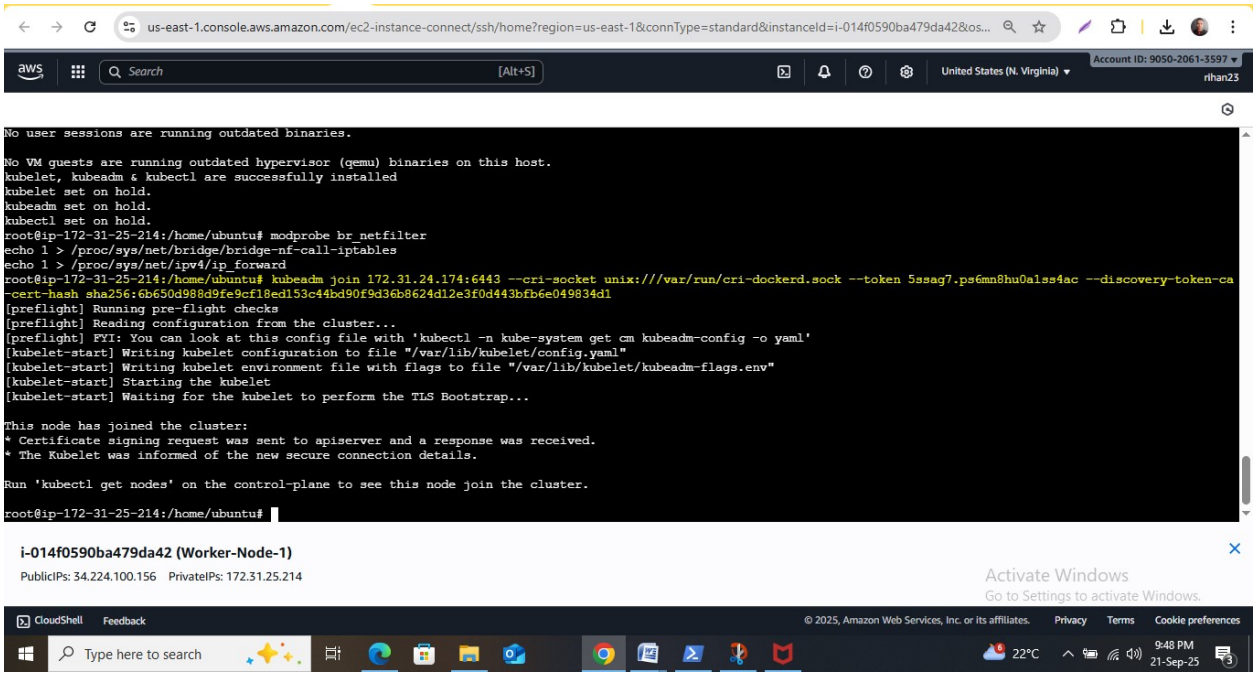
Step8: After this we need to generate the tokens in master machine using command → kubeadm token create --print-join-command



whatever token comes up ...copy the token and paste it in notepad and after that in the command you will see 6443 written ...after that paste --cri-socket unix:///var/run/cri-dockerd.sock

kubeadm join 172.31.24.174:6443 --cri-socket unix:///var/run/cri-dockerd.sock --token 5ssag7.ps6mn8hu0a1ss4ac --discovery-token-ca-cert-hash sha256:6b650d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443bfb6e049834d1

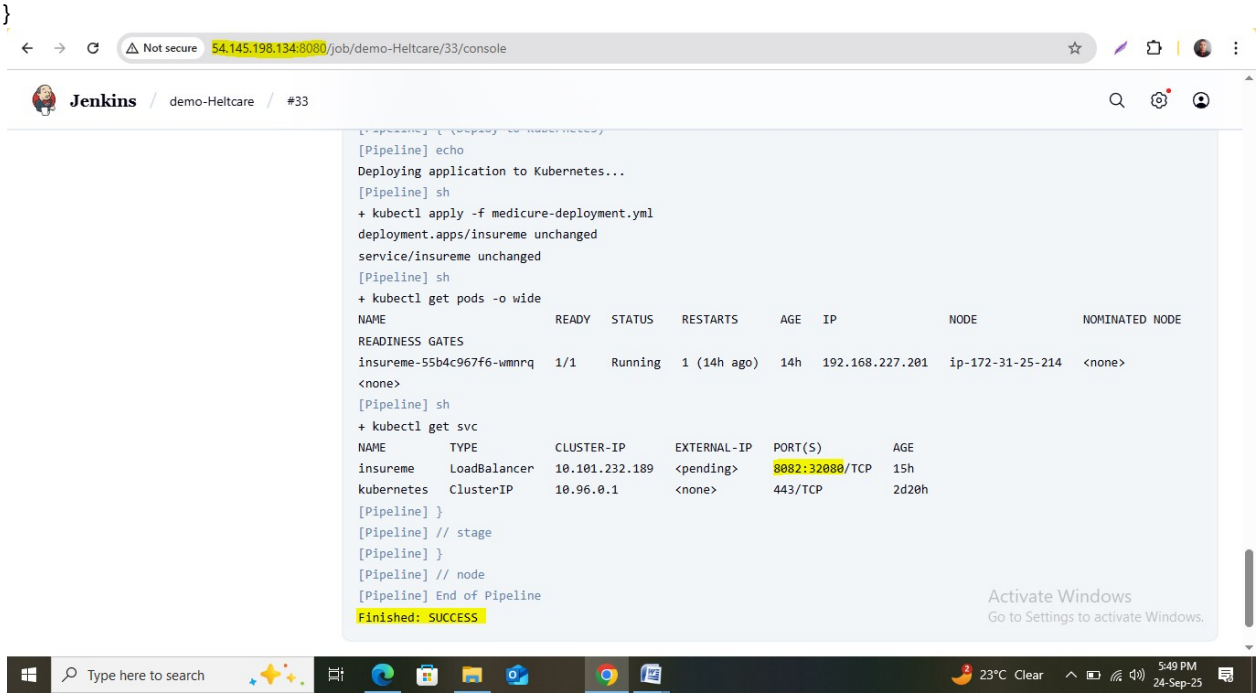
Paste the above token in nodes to establish the connect with master



Step 9: Now go to Jenkins dashboard create a pipeline and write a code to build image and push the image to docker and deploy

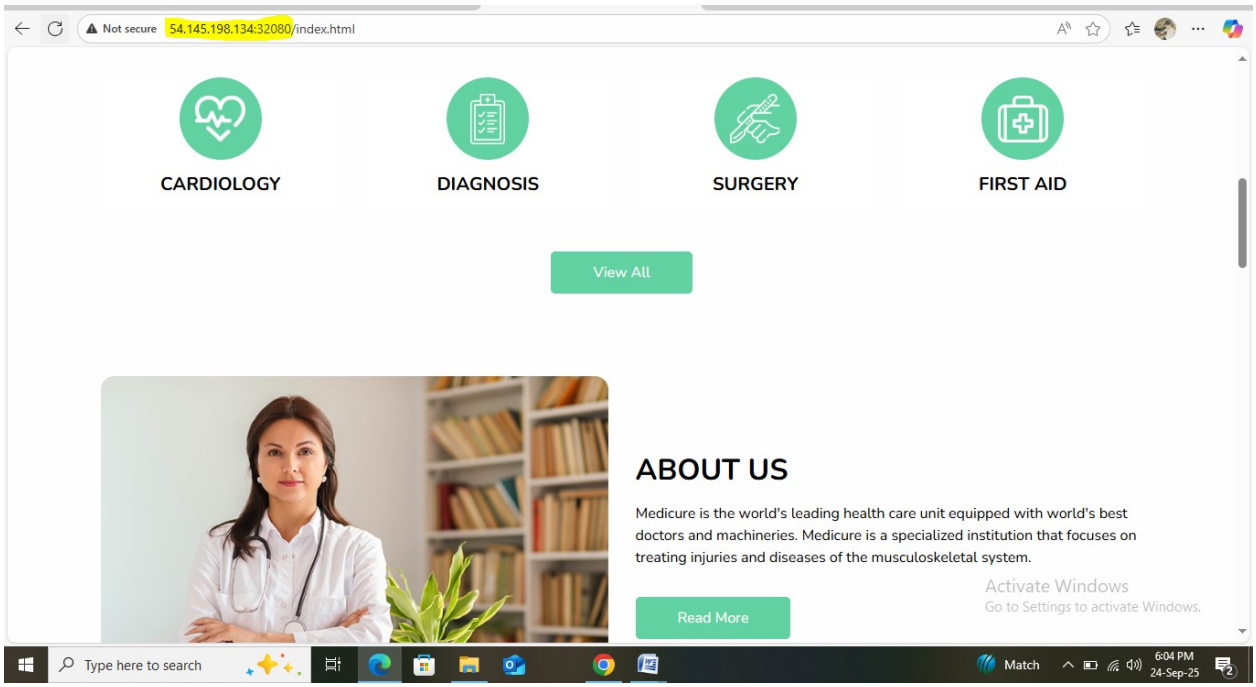
```
pipeline {
agent any

stages {
stage('Checkout') {
steps {
git 'https://github.com/Rihan297/star-agile-health-care.git'
}
}

stage('Build') {
steps {
sh 'mvn clean package -DskipTests'
}
}

stage('Build Docker Image') {
steps {
sh 'docker build -t rihan297/demohealthcare:1.0 .'
}
}

stage('Push Docker Image') {
steps {
withCredentials([usernamePassword(credentialsId: 'dockerhub',
usernameVariable: 'DOCKER_USER',
passwordVariable: 'DOCKER_PASS')]) {
sh '"
```

```
echo "$DOCKER_PASS" | docker login -u "$DOCKER_USER" --password-stdin
docker push rihan297/demohealthcare:1.0
'''
}
}
}

stage('Deploy to Kubernetes') {
steps {
echo "Deploying application to Kubernetes..."
sh 'kubectl apply -f medicure-deployment.yml'
sh 'kubectl get pods -o wide'
sh 'kubectl get svc'
}
}
}
}
```



Our image is successfully deployed

Step 10: Copy the port number which is highlighted in image paste it in a browser with public ip to run our website



Step 11: Installing Prometheus and graphana in a machine using shell script

1.Prometheus

```
# Update system
sudo apt update -y
# Create a Prometheus user
sudo useradd --no-create-home --shell /bin/false prometheus
# Create required directories
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
# Download Prometheus
cd /tmp
curl -LO https://github.com/prometheus/prometheus/releases/download/v2.53.2/prometheus-2.53.2.linux-amd64.tar.gz
# Extract
tar xvf prometheus-2.53.2.linux-amd64.tar.gz
cd prometheus-2.53.2.linux-amd64
# Move binaries
sudo mv prometheus /usr/local/bin/
sudo mv promtool /usr/local/bin/
# Move config files
sudo mv consoles /etc/prometheus
sudo mv console_libraries /etc/prometheus
sudo mv prometheus.yml /etc/prometheus/prometheus.yml
# Set ownership
sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
sudo chown prometheus:prometheus /usr/local/bin/prometheus
```

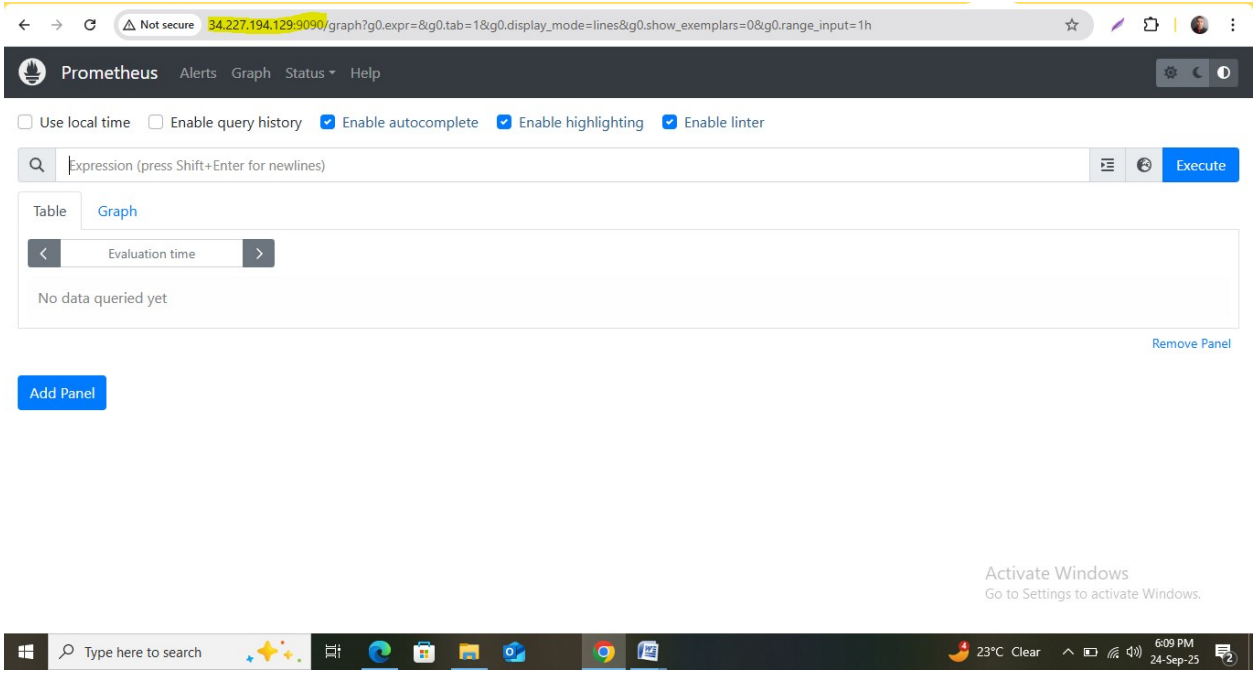
sudo chown prometheus:prometheus /usr/local/bin/promtool

Create a systemd service

```
sudo tee /etc/systemd/system/prometheus.service > /dev/null <<EOF
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
[Install]
WantedBy=multi-user.target
EOF
```

Start Prometheus

```
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start Prometheus
```

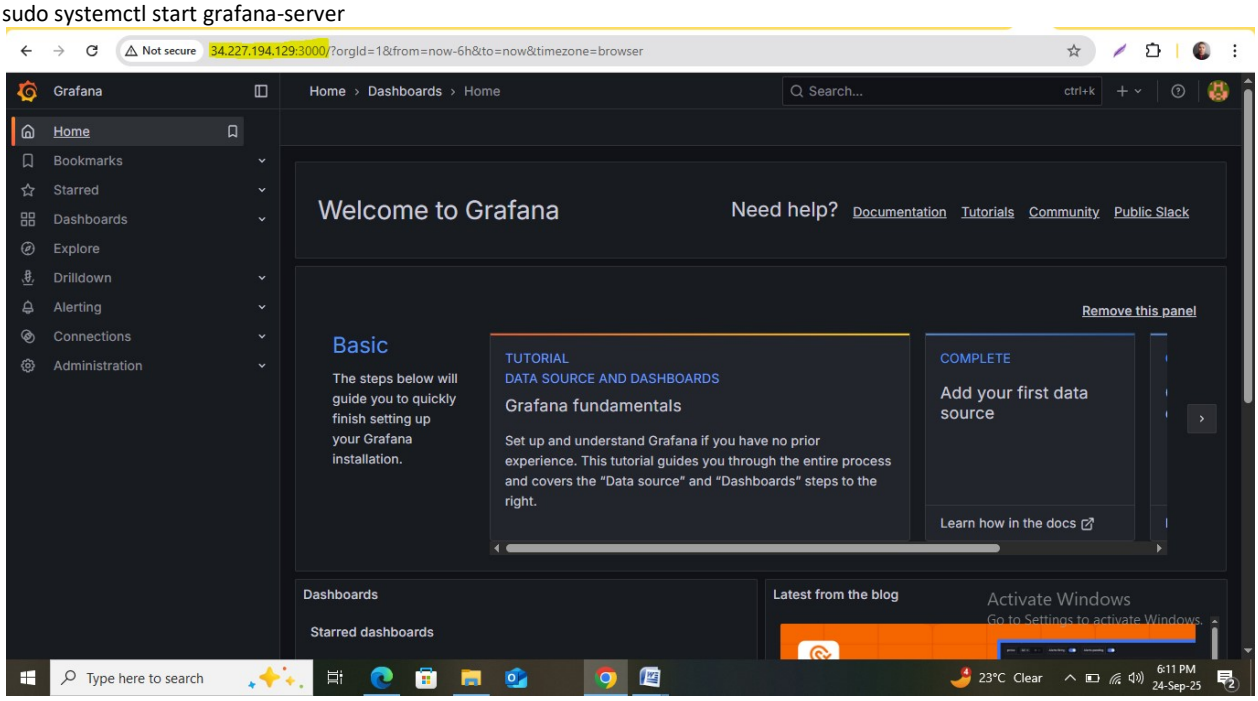


2. Install Grafana

# Add Grafana repo
sudo apt-get install -y apt-transport-https software-properties-common
sudo wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list

# Install Grafana
sudo apt-get update -y
sudo apt-get install -y grafana

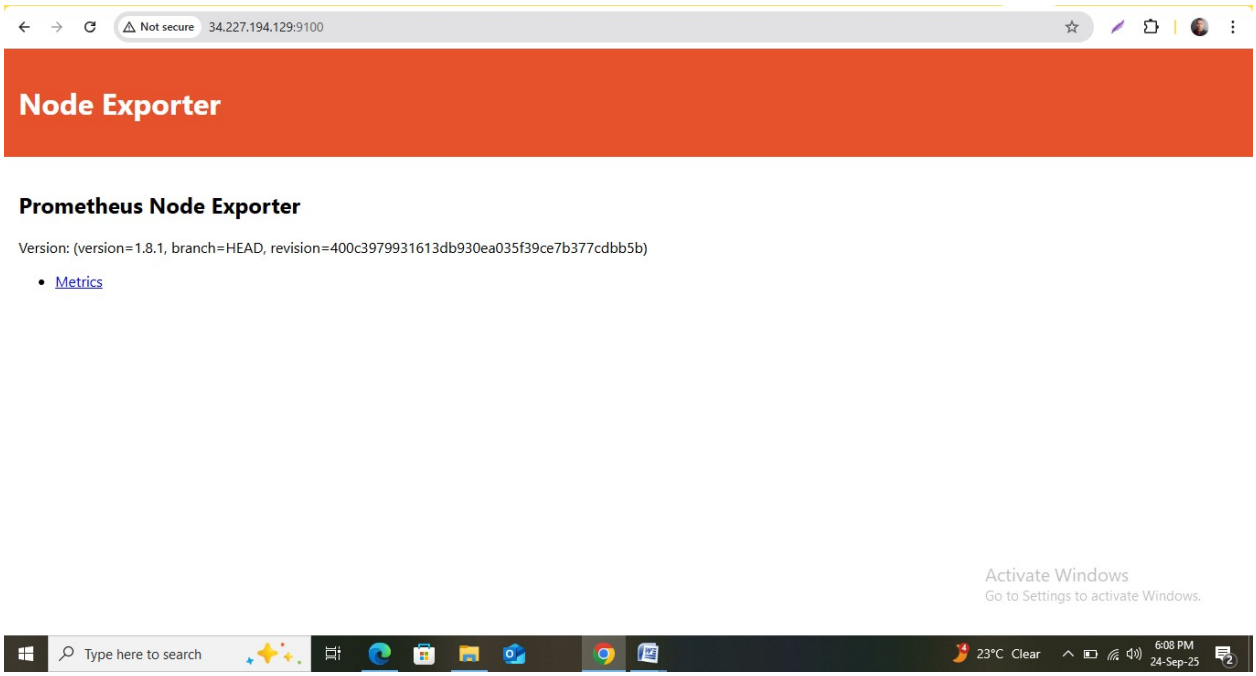
# Start Grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server



3. now we need to install node exporter using the command in both master and one of the worker node

cd /tmp
curl -LO https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz
Extract
tar xvf node_exporter-1.8.1.linux-amd64.tar.gz
sudo mv node_exporter-1.8.1.linux-amd64/node_exporter /usr/local/bin/
# Create a service
sudo tee /etc/systemd/system/node_exporter.service <<EOF
[Unit]
Description=Node Exporter
After=network.target
[Service]
User=nobody
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=default.target
EOF
Start service

sudo systemctl daemon-reexec
sudo systemctl enable --now node_exporter



Step 12 :First we need to import dashboard in graphana. So click on dashboard→new→import dashboard→enter dashboard id 1860→load→Prometheus data source→load

After that Make sure all nodes are in Prometheus. On your Prometheus server, open the config file (sudo nano /etc/prometheus/prometheus.yml sudo systemctl restart prometheus)

Step 13: Import Node Exporter Dashboard (ID 1860)

- In Grafana, go to **Dashboards → New → Import**.
- Enter **1860** → Load.
- Choose your **Prometheus data source** → Import.

This dashboard is already built to handle **multiple instances**. At the top of the dashboard, you will see a **dropdown**