# Certification Project – Insurance Domain

Submitted by – Mohammed Abu Rihan

Submitted to – Vikul mentor

Batch no - DevOpsSA2504024

Date of submission – 21-09-2025

# Insurance Domain

## Step 1: Launch an aws ec2 instance and install terraform and aws cli

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

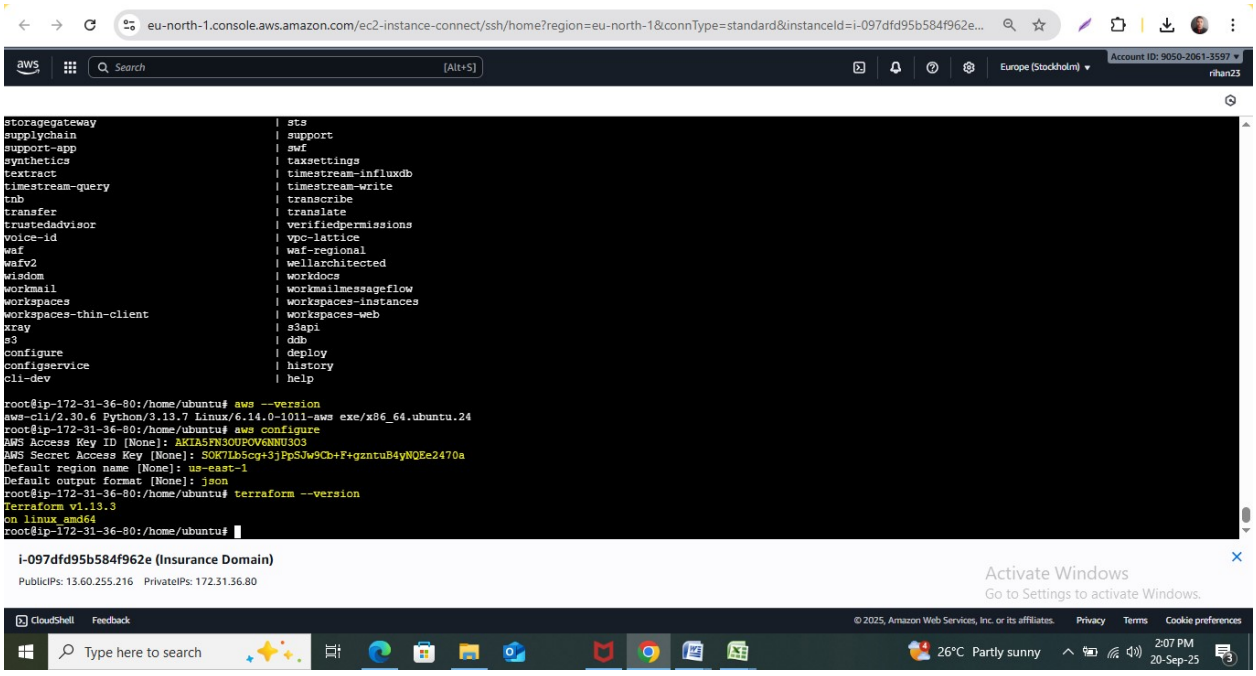
curl -sS "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o awscliv2.zip

sudo apt update && sudo apt install unzip –y

unzip awscliv2.zip

sudo ./aws/install

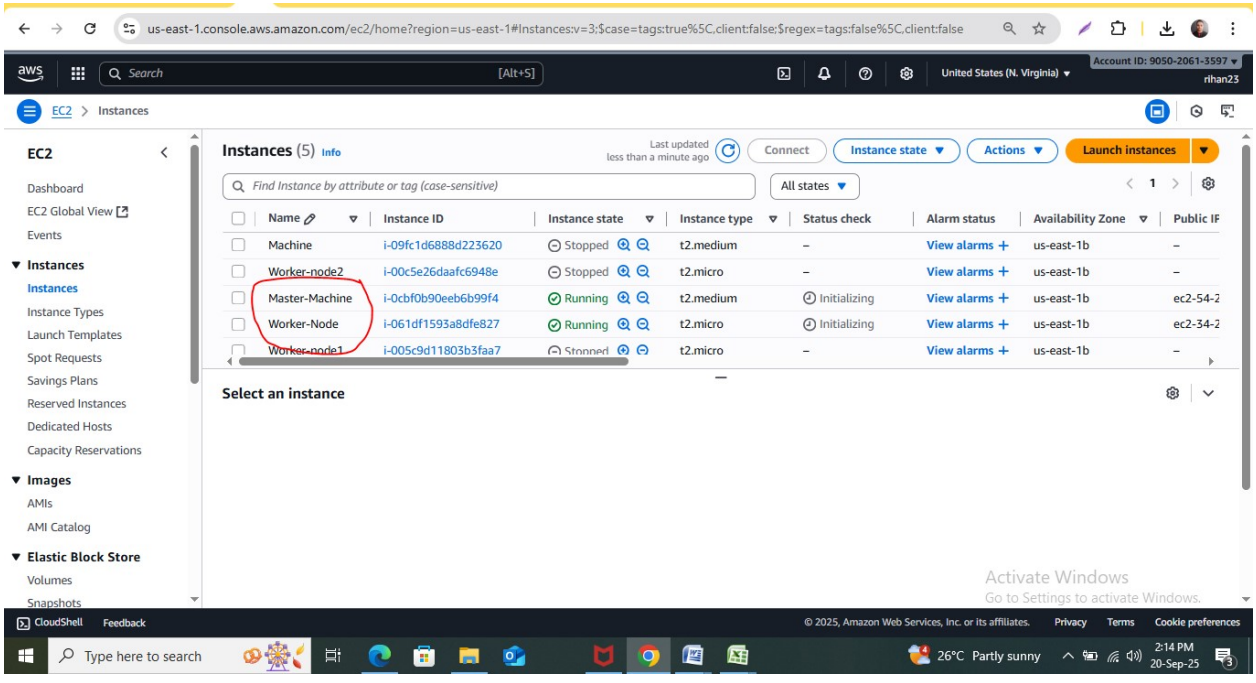
aws –version



## Step 2:Creating instance using terraform

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

# Get default VPC
data "aws_vpc" "default" {
default = true
}

# Security Group: SSH only
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 traffic"
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"]
}
```

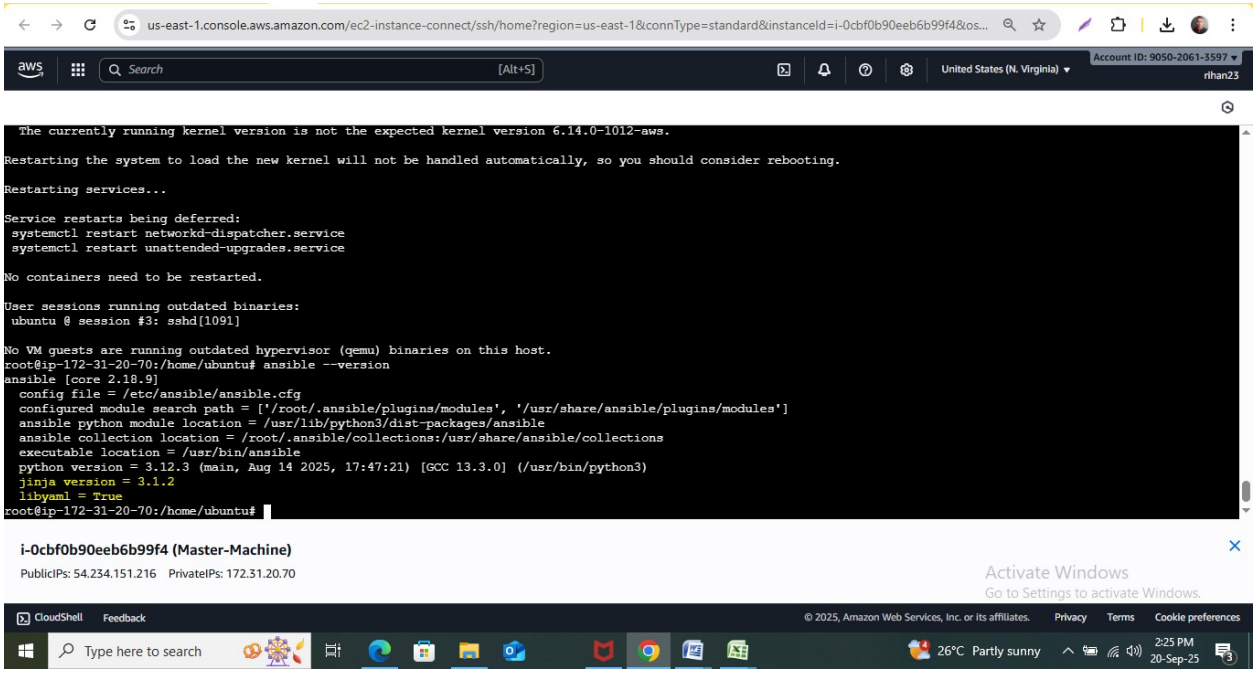
```
}

# Master Machine (t2.medium)
resource "aws_instance" "master" {
ami = "ami-0360c520857e3138f"
instance_type = "t2.medium"
key_name = "usnv1"
vpc_security_group_ids = [
aws_security_group.ssh_sg.id,
aws_security_group.all_traffic_sg.id
]

tags = {
Name = "Master-Machine"
}
}

# Worker Node (t2.micro)
resource "aws_instance" "worker" {
ami = "ami-0360c520857e3138f"
instance_type = "t2.micro"
key_name = "usnv1"
vpc_security_group_ids = [
aws_security_group.ssh_sg.id,
aws_security_group.all_traffic_sg.id
]

tags = {
Name = "Worker-Node"
}
}
```



```
```

## Step 3: Install ansible and Create ansible playbook and install Git, Java, Maven installation command

sudo apt update -y
sudo apt upgrade -y
sudo apt install software-properties-common -y
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible -y
ansible –version




### Creating playbook

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

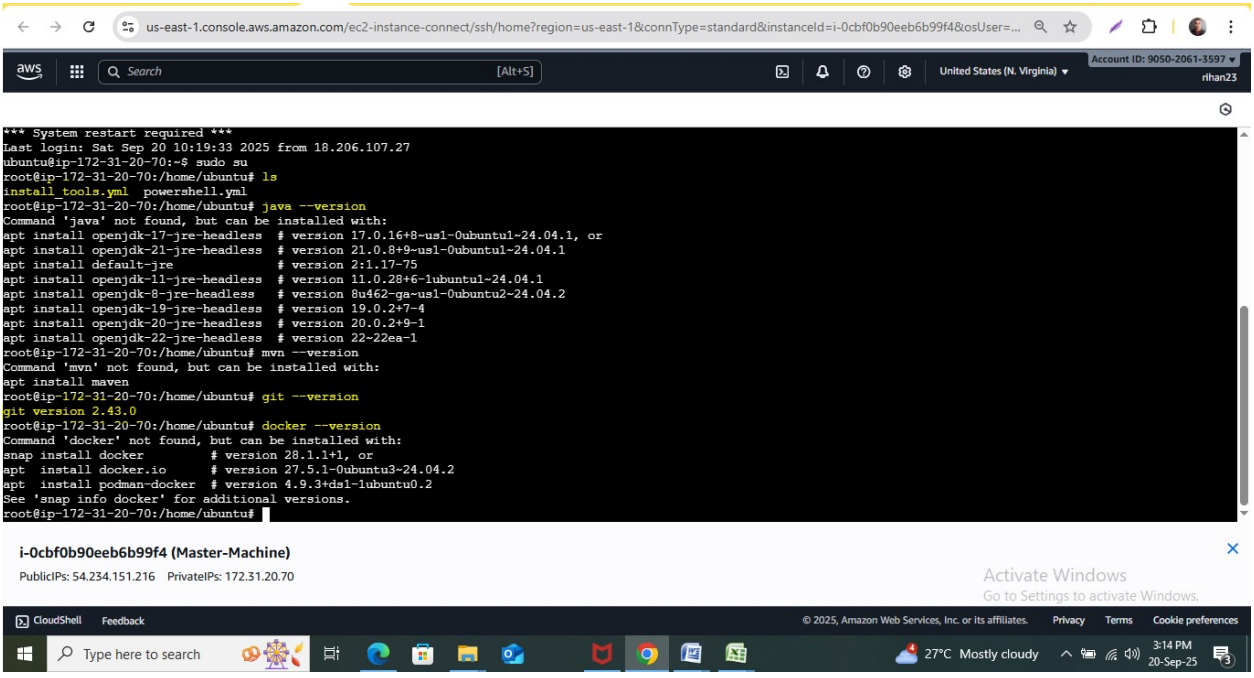
tasks:
apt:
update_cache: yes

apt:
name: git
state: present

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

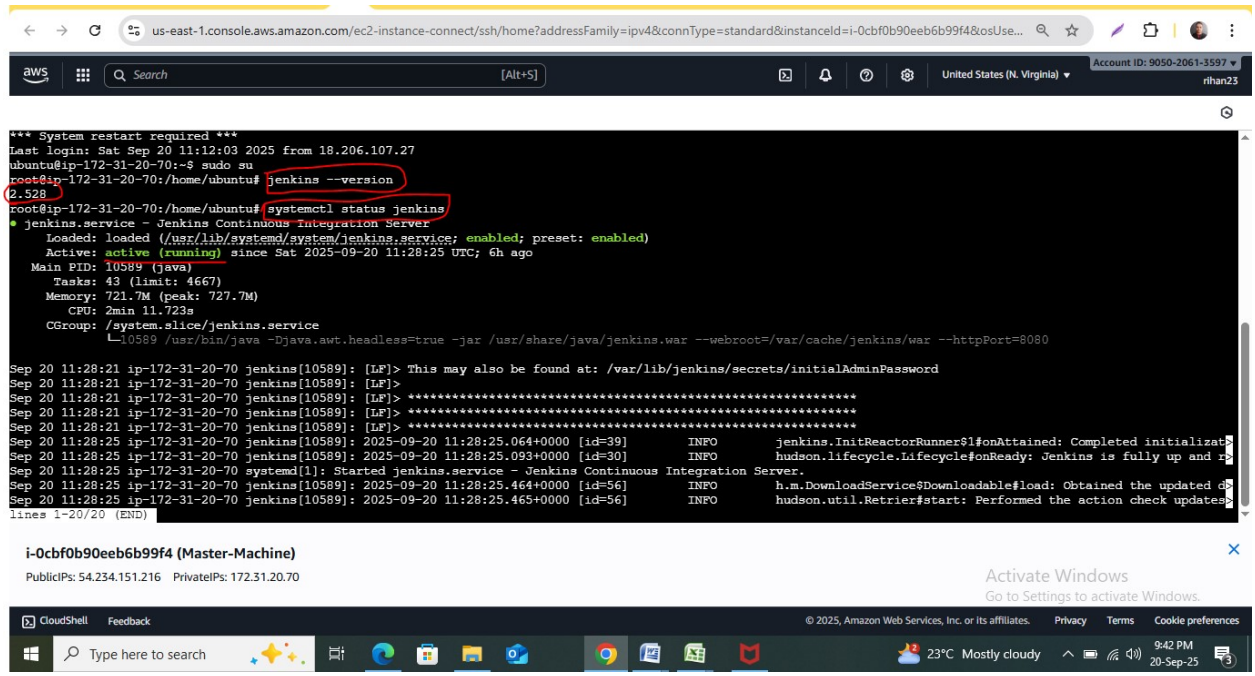
apt:
name: maven
state: present

```
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
```

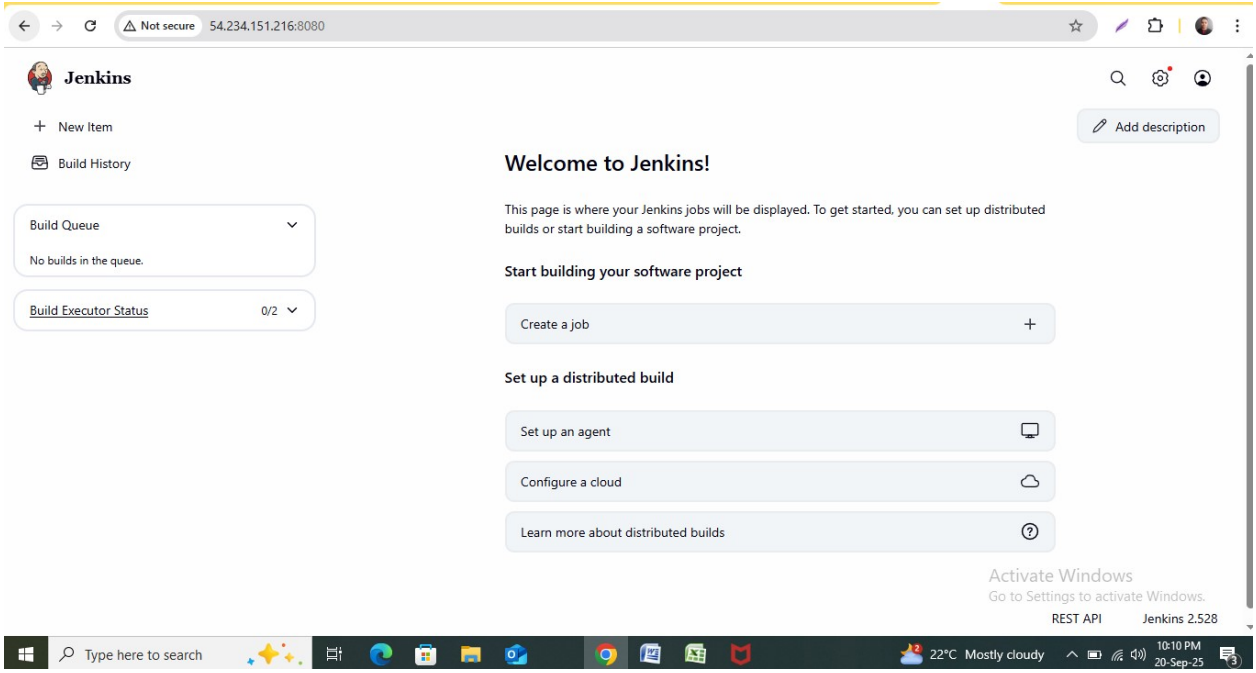


## Step 4: Install Jenkins using the command

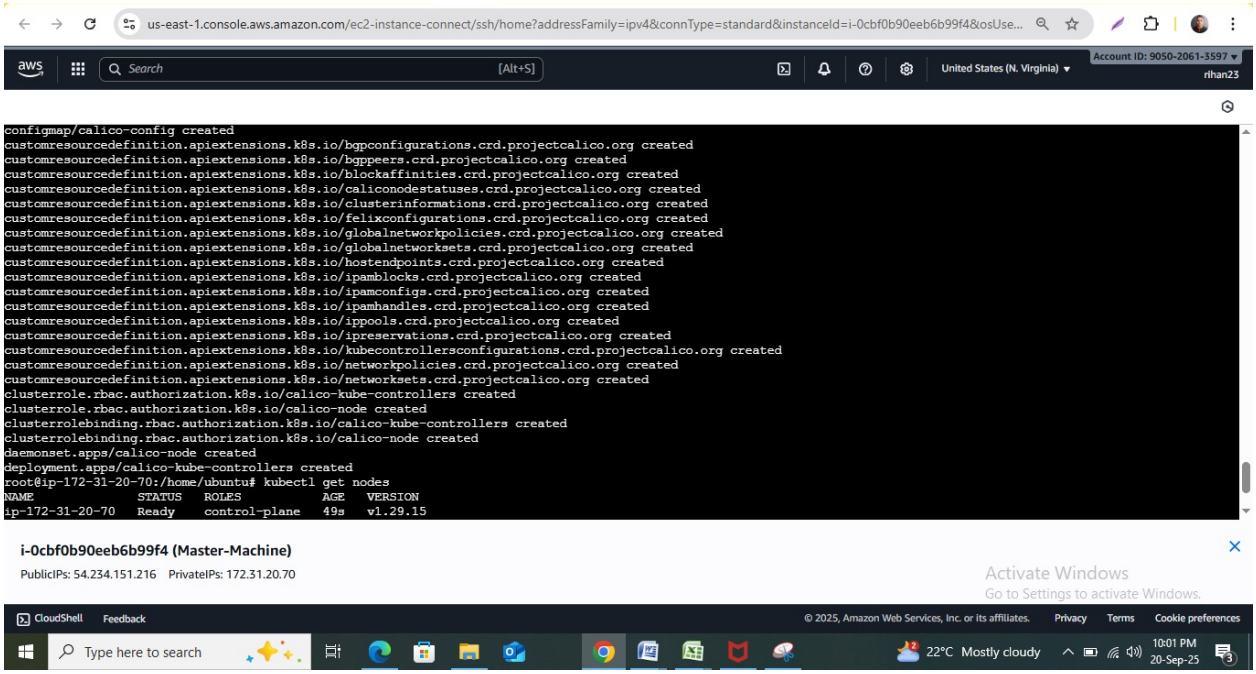
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt update
sudo apt install jenkins
systemctl status jenkins
jenkins –version



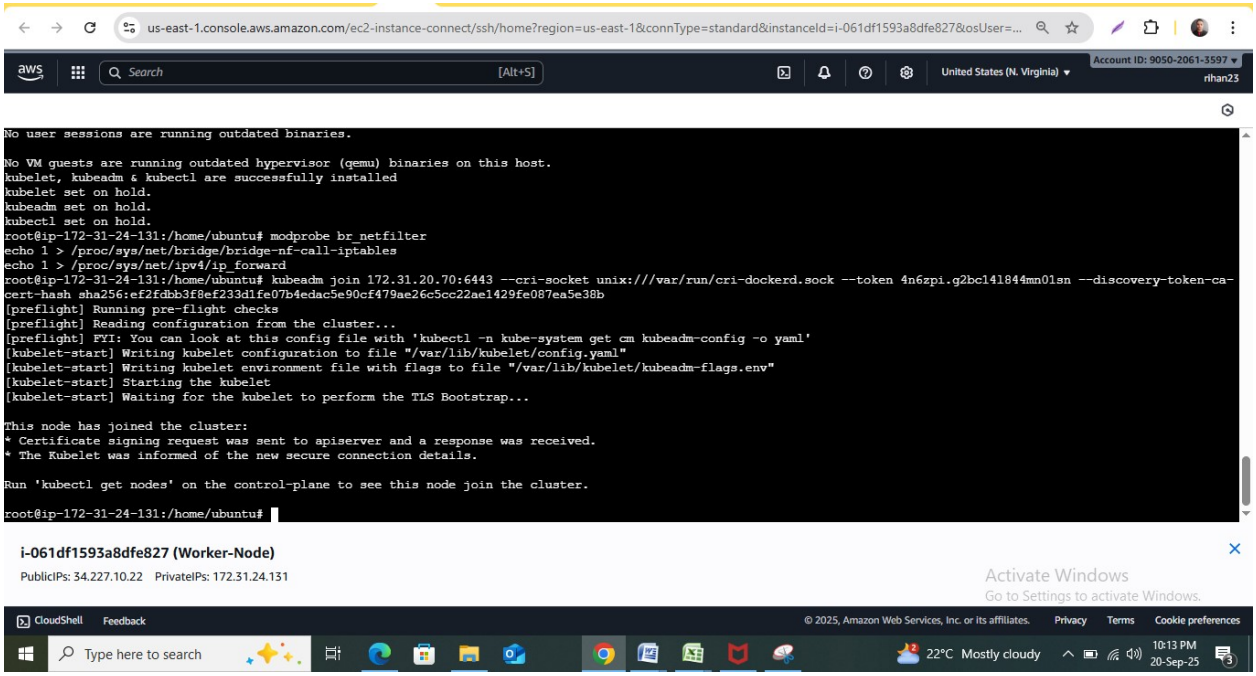
Installed Jenkins plugins now Jenkins is ready to use




## Step 5: k8s has been installed



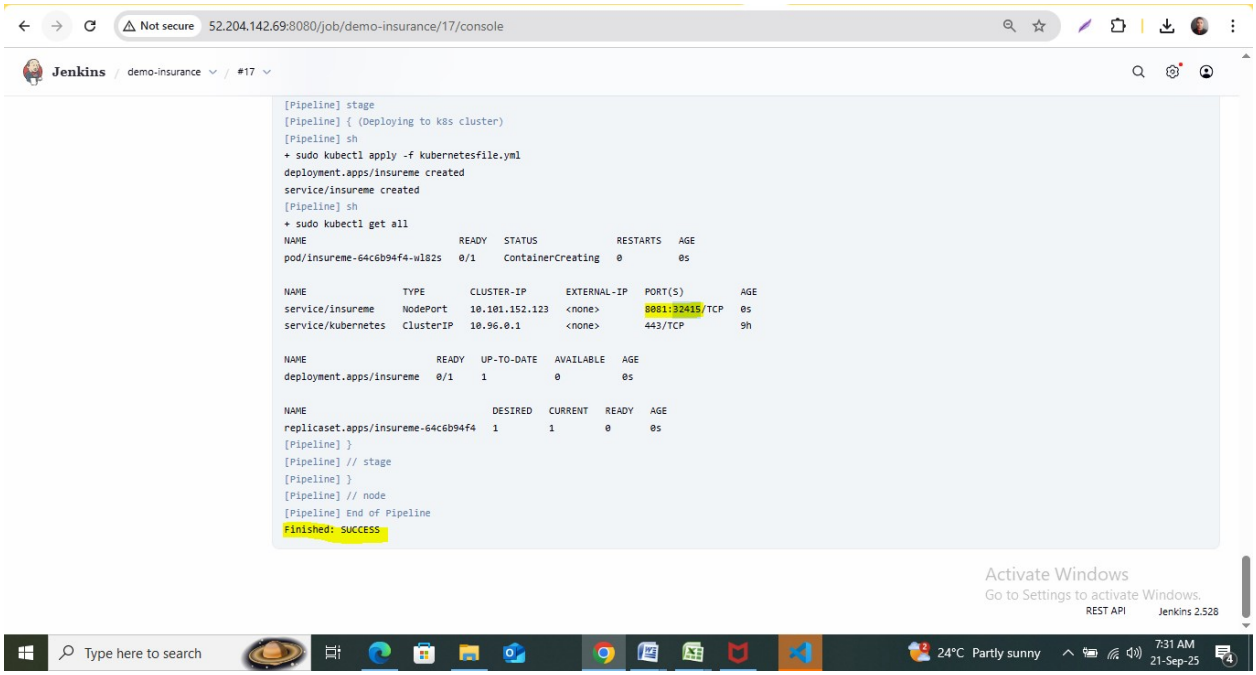
in worker node we installed k8s and connected with master using cri socket




## Step 6: Now go to Jenkins dashboard create a pipeline and write a code to build image and push the image to docker and deploy

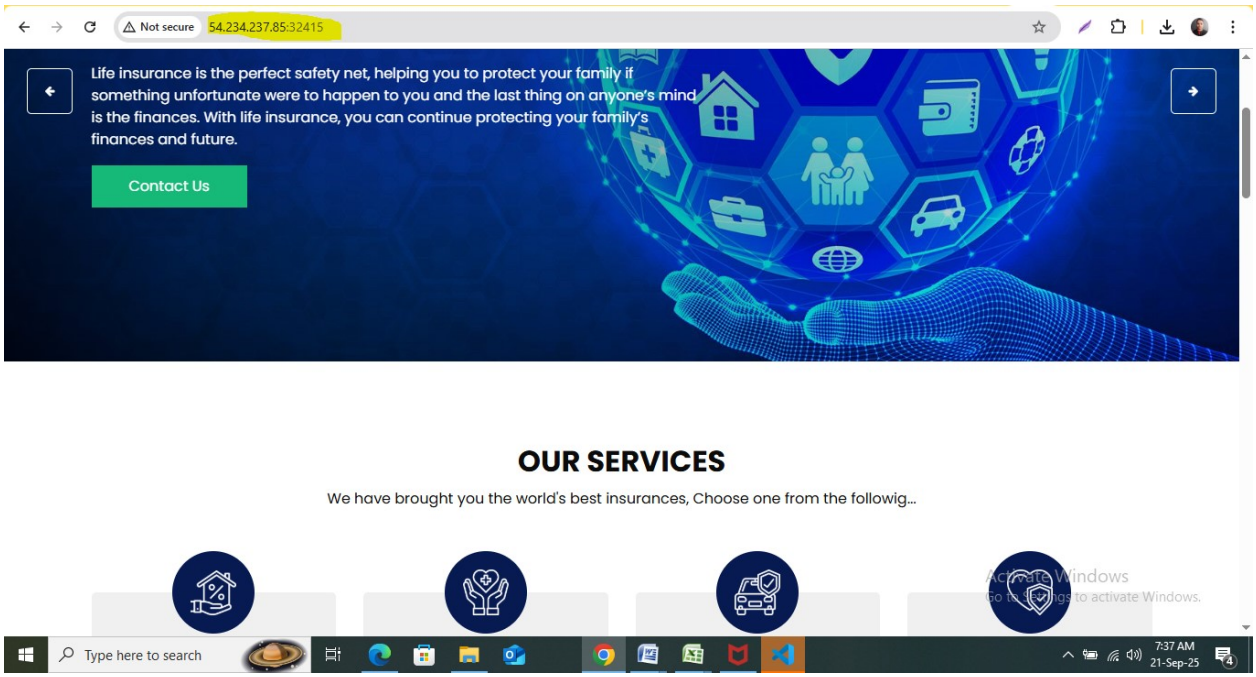
```
pipeline {
agent any

stages {
stage('git checkout') {
steps {
git 'https://github.com/Rihan297/star-agile-insurance-project.git'
}
}

stage('build the project') {
steps {
sh 'mvn clean package'
}
}

stage('build docker image') {
steps {
sh 'docker build -t rihan297/demoinsurancerihan:1.0 .'
}
}

stage('push docker image') {
steps {
withCredentials([string(credentialsId: 'dockerpass', variable: 'dockerhubpass')]) {
sh '''
echo "$dockerhubpass" | docker login -u rihan297 --password-stdin
docker push rihan297/demoinsurancerihan:1.0
```

```
'''
}
}
}
stage('Deploying to k8s cluster') {
steps {
sh 'sudo kubectl apply -f kubernetesfile.yml'
sh 'sudo kubectl get all'
}
}
}
}
```




Our image is successfully deployed

## Step 7: Copy the port number which is highlighted in image paste it in a browser with worker node ip we get our website




## Step 8: Installing Prometheus and graphana in a machine using shell script

## 1.Prometheus

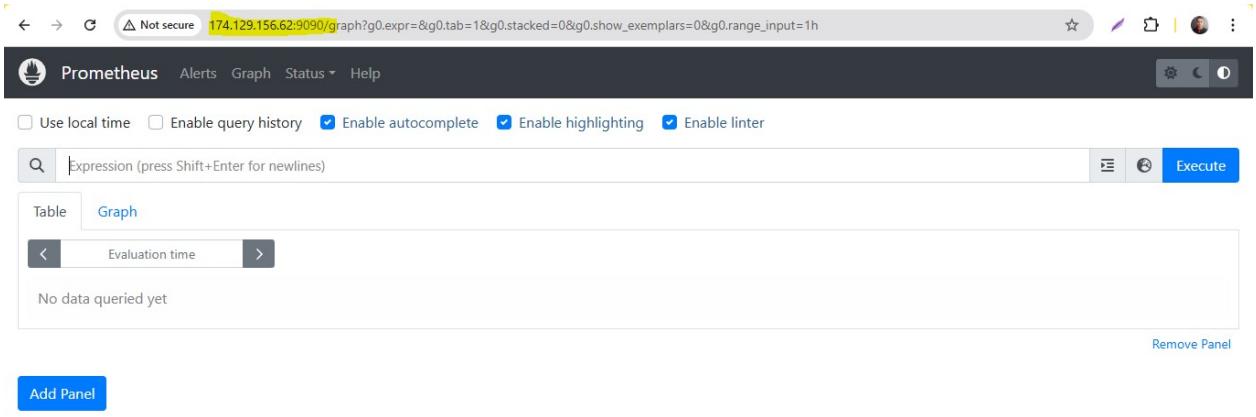
```
# Update system
sudo apt update -y
# Create a Prometheus user
sudo useradd --no-create-home --shell /bin/false prometheus
# Create required directories
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
# Download Prometheus
cd /tmp
curl -LO https://github.com/prometheus/prometheus/releases/download/v2.53.2/prometheus-2.53.2.linux-amd64.tar.gz
# 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

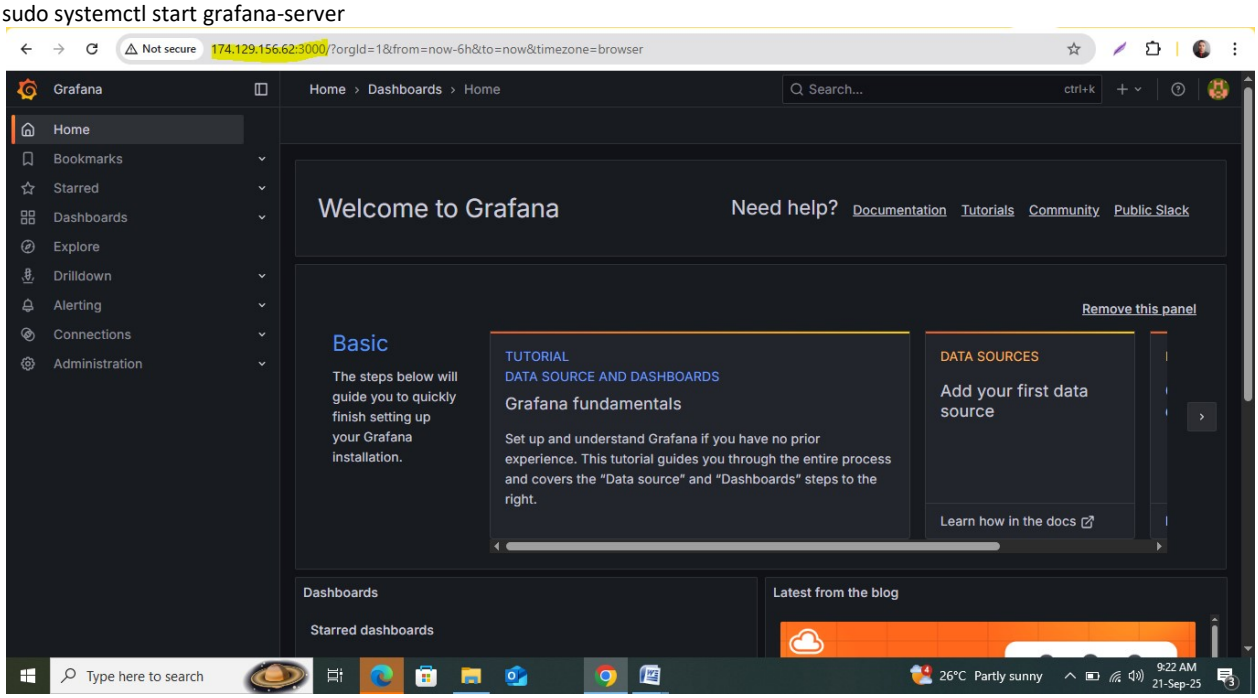



## 2. Install Grafana

# 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

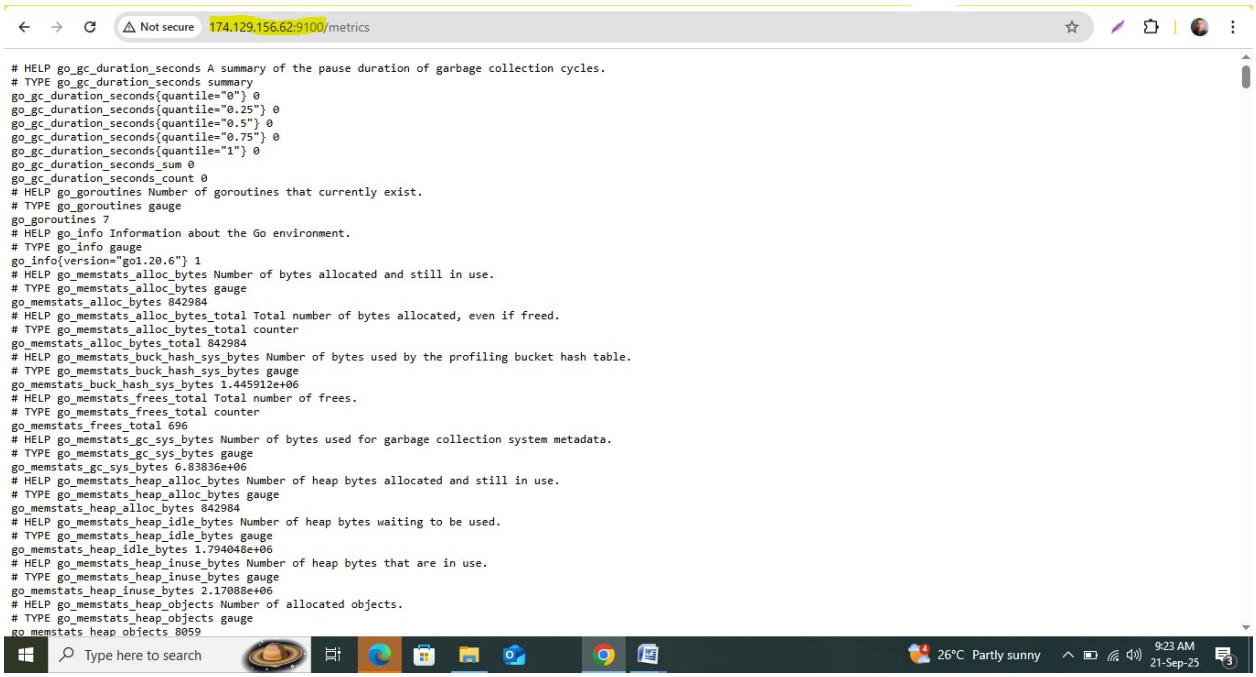
# Install Grafana
sudo apt-get update -y
sudo apt-get install -y grafana

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



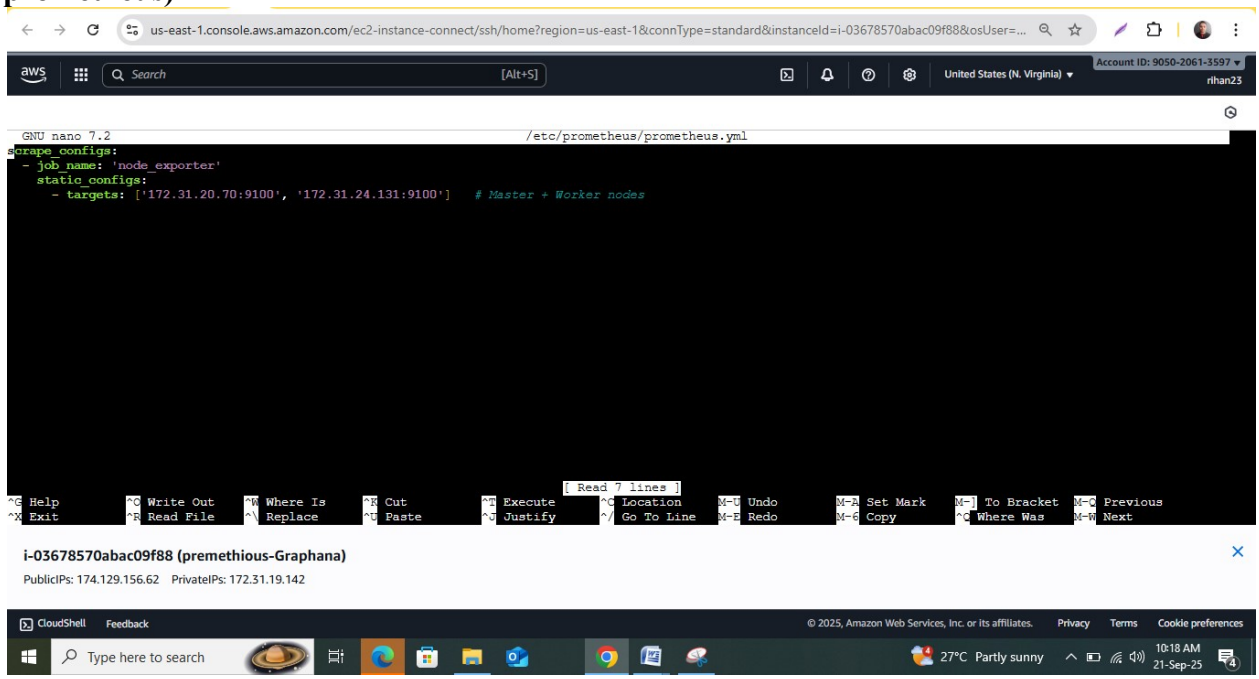
# Step 9: Installing node exporter

```
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 10 :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 11: 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**