

Certification Project – Insurance Domain

Submitted by – Mohammed Abu Rihan

Submitted to – Vikul mentor

Batch no - DevOpsSA2504024

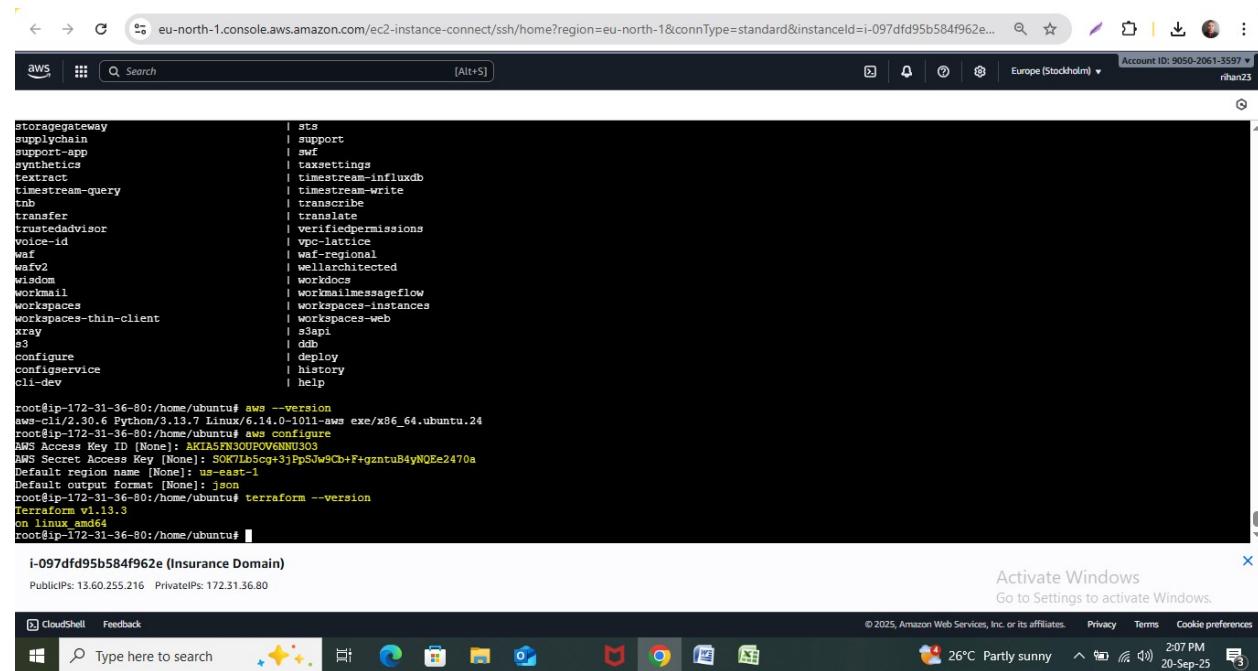
Date of submission – 21-09-2025

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
```



```
eu-north-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=eu-north-1&connType=standard&instanceId=i-097dfd95b584f962e... Account ID: 9050-2061-3597 rihan23

storagegateway      | sts
supplychain        | support
support-app        | swf
synthetics          | taxsettings
extract             | timestream-influxdb
timestream-query   | timestream-write
mb                 | transcribe
transfer            | translate
trustedadvisor     | verifiedpermissions
voice-id           | vpc-lattice
waf                | waf-regional
wafv2              | wellarchitected
wisdom              | workdocs
workmail            | workmailmessageflow
workspaces          | workspaces-instances
workspaces-thin-client | workspaces-web
xray               | xray
xg                 | xray
configure          | deploy
configservice       | history
cli-dev            | help

root@ip-172-31-36-80:/home/ubuntu# aws --version
aws-cli/2.30.6 Python/3.13.7 Linux/6.14.0-1011-aws exe/x86_64.ubuntu.24
root@ip-172-31-36-80:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIA5N3J0UPOV6NU303
AWS Secret Access Key [None]: SOK7Lbcg+3jPp5Ow9Cb+F+gntuB4yNQEe2470a
Default region [None]: us-east-1
Default output format [None]: json
root@ip-172-31-36-80:/home/ubuntu# terraform --version
Terraform v1.13.3
on linux amd64
root@ip-172-31-36-80:/home/ubuntu#
```

i-097dfd95b584f962e (Insurance Domain)

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback

Type here to search

26°C Partly sunny 2:07 PM 20-Sep-25

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"
}
}

```

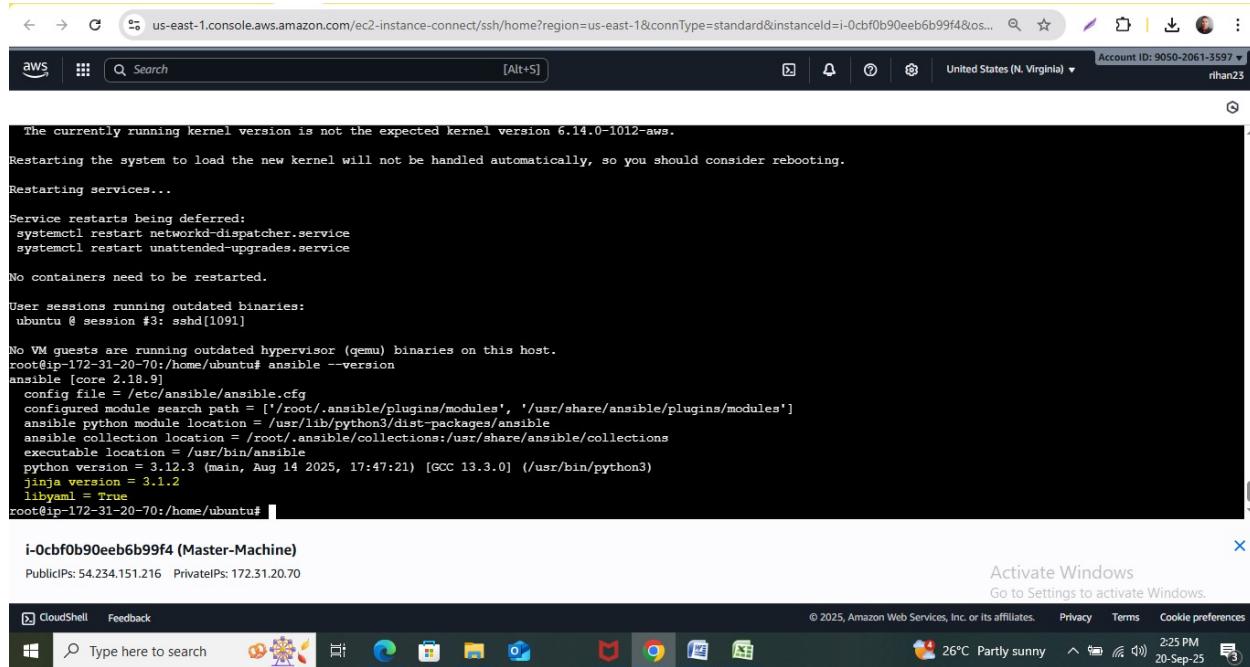
The screenshot shows the AWS EC2 Instances page with the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
Machine	i-09fc1d6888d223620	Stopped	t2.medium	-	View alarms +	us-east-1b	-
Worker-node2	i-00c5e26daafc6948e	Stopped	t2.micro	-	View alarms +	us-east-1b	-
Master-Machine	i-0cbf0b90eeb6b99f4	Running	t2.medium	Initializing	View alarms +	us-east-1b	ec2-54-2
Worker-Node	i-061df1593a8dfe827	Running	t2.micro	Initializing	View alarms +	us-east-1b	ec2-34-2
Worker-node1	i-005c9d11803b3faa7	Stopped	t2.micro	-	View alarms +	us-east-1b	-

The 'Master-Machine' instance is circled in red.

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
```



```
The currently running kernel version is not the expected kernel version 6.14.0-1012-aws.
Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.
Restarting services...
Service restarts being deferred:
systemctl restart networkd-dispatcher.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
ubuntu @ session #3: sshd[1091]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-20-70:/home/ubuntu# ansible --version
ansible [core 2.18.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['~/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = ~/root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-20-70:/home/ubuntu#
```

i-0cbf0b90eeb6b99f4 (Master-Machine)
Public IPs: 54.234.151.216 Private IPs: 172.31.20.70

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

26°C Partly sunny 2:25 PM 20-Sep-25

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

```

```

*** System restart required ***
Last login: Sat Sep 20 10:19:33 2025 from 18.206.107.27
ubuntu@ip-172-31-20-70:~$ sudo su
root@ip-172-31-20-70:/home/ubuntu# ls
install_tools.yml powershell.yml
root@ip-172-31-20-70:/home/ubuntu# java --version
Command 'java' not found, but can be installed with:
apt install openjdk-17-jre-headless # version 17.0.16+8~us1-Ubuntu1-24.04.1, or
apt install openjdk-21-jre-headless # version 21.0.8+9~us1-Ubuntu1-24.04.1
apt install default-jre # version 2:1.17~75
apt install openjdk-11-jre-headless # version 11.0.28+6~ubuntui-24.04.1
apt install openjdk-8-jre-headless # version 8u42~ga~us1-Ubuntu2-24.04.2
apt install openjdk-19-jre-headless # version 19.0.24~7~4
apt install openjdk-20-jre-headless # version 20.0.24~9~1
apt install openjdk-22-jre-headless # version 22-22ea-1
root@ip-172-31-20-70:/home/ubuntu# mvn --version
Command 'mvn' not found, but can be installed with:
apt install maven
root@ip-172-31-20-70:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-20-70:/home/ubuntu# docker --version
Command 'docker' not found, but can be installed with:
snap install docker # version 28.1.1+1, or
apt install docker.io # version 27.5.1~Ubuntu3-24.04.2
apt install podman-docker # version 4.9.3+ds1-1ubuntu0.2
See 'snap info docker' for additional versions.
root@ip-172-31-20-70:/home/ubuntu#

```

i-0cbf0b90eeb6b99f4 (Master-Machine)

Public IPs: 54.234.151.216 Private IPs: 172.31.20.70

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Type here to search

27°C Mostly cloudy 3:14 PM 20-Sep-25

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
```

```
** System restart required ***
Last login: Sat Sep 20 11:12:03 2025 from 18.206.107.27
ubuntu@ip-172-31-20-70:~$ sudo su
root@ip-172-31-20-70:/home/ubuntu# jenkins --version
2.528
root@ip-172-31-20-70:/home/ubuntu# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Sat 2025-09-20 11:28:25 UTC, 6h ago
     Main PID: 10589 (java)
        Tasks: 43 (limit: 4667)
       Memory: 721.7M (peak: 727.7M)
          CPU: 2min 11.723s
      CGroup: /system.slice/jenkins.service
              └─10589 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 20 11:28:21 ip-172-31-20-70 jenkins[10589]: [LF> This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 20 11:28:21 ip-172-31-20-70 jenkins[10589]: [LF>
Sep 20 11:28:21 ip-172-31-20-70 jenkins[10589]: [LF> ****
Sep 20 11:28:25 ip-172-31-20-70 jenkins[10589]: 2025-09-20 11:28:25.064+0000 [id=39]           INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
Sep 20 11:28:25 ip-172-31-20-70 jenkins[10589]: 2025-09-20 11:28:25.093+0000 [id=30]           INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Sep 20 11:28:25 ip-172-31-20-70 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 20 11:28:25 ip-172-31-20-70 jenkins[10589]: 2025-09-20 11:28:25.464+0000 [id=56]           INFO    h.m.DownloadService$Downloadable#load: Obtained the updated download
Sep 20 11:28:25 ip-172-31-20-70 jenkins[10589]: 2025-09-20 11:28:25.465+0000 [id=56]           INFO    hudson.util.Retrier#start: Performed the action check update
lines 1-20/20 (END)
```

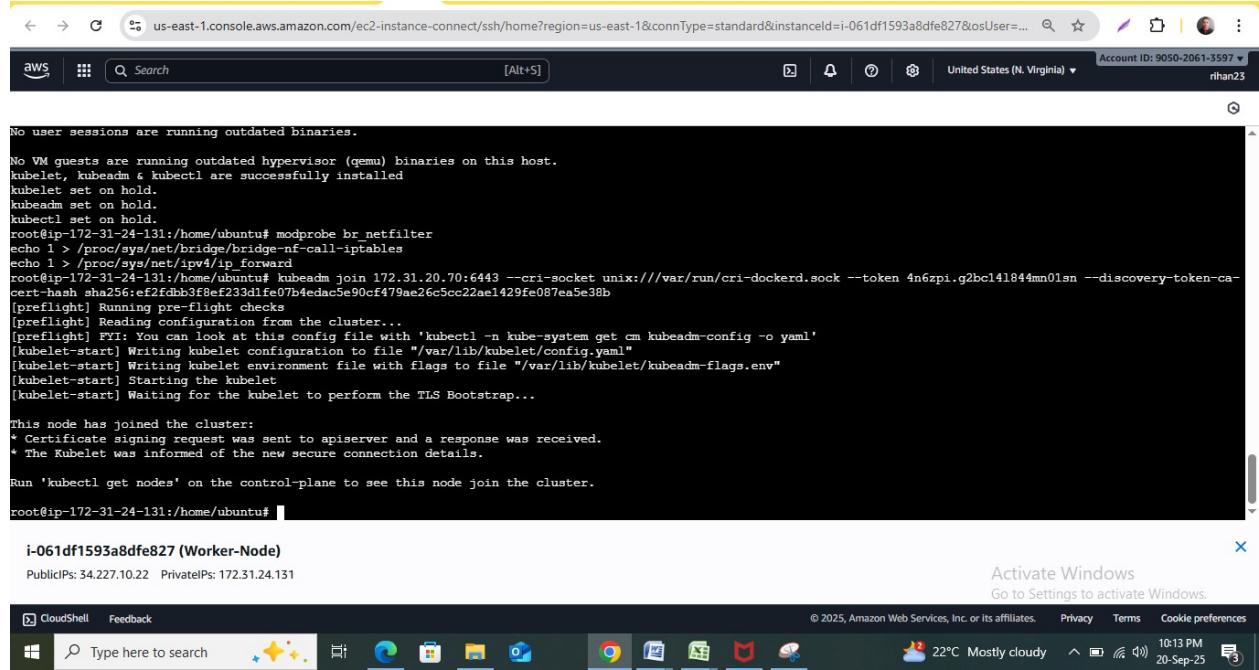
Installed Jenkins plugins now Jenkins is ready to use

The screenshot shows the Jenkins dashboard at 54.234.151.216:8080. The top navigation bar includes links for 'New Item', 'Build History', 'Add description', and a user profile. The main content area features a 'Welcome to Jenkins!' message, a 'Build Queue' section (empty), and a 'Build Executor Status' section (0/2). A central button 'Create a job' is surrounded by links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom status bar shows system information like '22°C Mostly cloudy' and the date '20-Sep-25'.

Step 5: k8s has been installed

The screenshot shows an AWS CloudShell session with the URL <https://us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-0cbf0b90eef6b99f4&osUser=rhan23>. The session displays the output of a `kubectl get nodes` command, showing a single node named `ip-172-31-20-70` in the `control-plane` role. The status is `Ready` and the version is `v1.29.15`. The AWS navigation bar at the top includes 'Search', 'CloudShell', 'Feedback', and 'CloudWatch Metrics'.

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



The screenshot shows a terminal session on an AWS EC2 instance. The terminal window title is "aws" and the URL is "us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=us-east-1&connType=standard&instanceId=i-061df1593a8dfe827&osUser=...". The terminal content displays the output of a "kubeadm join" command, which includes configuration details like certificates and flags. The session ends with a root prompt at "root@ip-172-31-24-131:/home/ubuntu#". Below the terminal, a status bar shows "CloudShell Feedback" and the AWS footer "© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences". A weather widget indicates "22°C Mostly cloudy" and the date "20-Sep-25".

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'
}
}
}
}

```

The screenshot shows the Jenkins pipeline console output for job 'demo-insurance' run #17. The logs show the execution of a pipeline stage named 'Deploying to k8s cluster'. It includes commands like 'sudo kubectl apply -f kubernetesfile.yml' and 'sudo kubectl get all'. The output also displays Kubernetes status for pods, services, and replicaset. A yellow box highlights the final message 'Finished: SUCCESS'.

```

[Pipeline] stage
[Pipeline] { (Deploying to k8s cluster)
[Pipeline] sh
+ sudo kubectl apply -f kubernetesfile.yml
deployment.apps/insureme created
service/insureme created
[Pipeline] sh
+ sudo kubectl get all
NAME READY STATUS RESTARTS AGE
pod/insureme-64c6b94f4-wl82s 0/1 ContainerCreating 0 8s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/insureme NodePort 10.101.152.123 <none> 8081:32415/TCP 8s
service/kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 9h

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/insureme 0/1 1 0 8s

NAME DESIRED CURRENT READY AGE
replicaset.apps/insureme-64c6b94f4 1 1 0 8s
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

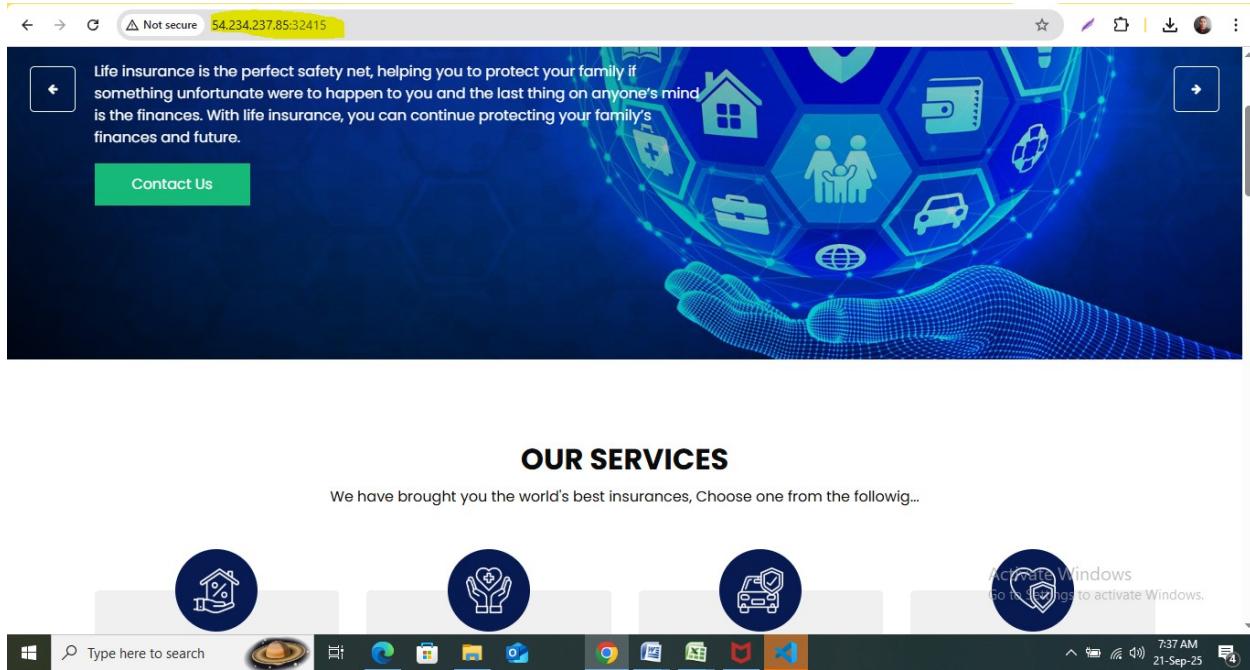
```

Activate Windows
Go to Settings to activate Windows.
REST API Jenkins 2.528

Type here to search 7:31 AM 24°C Partly sunny 21-Sep-25

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
```

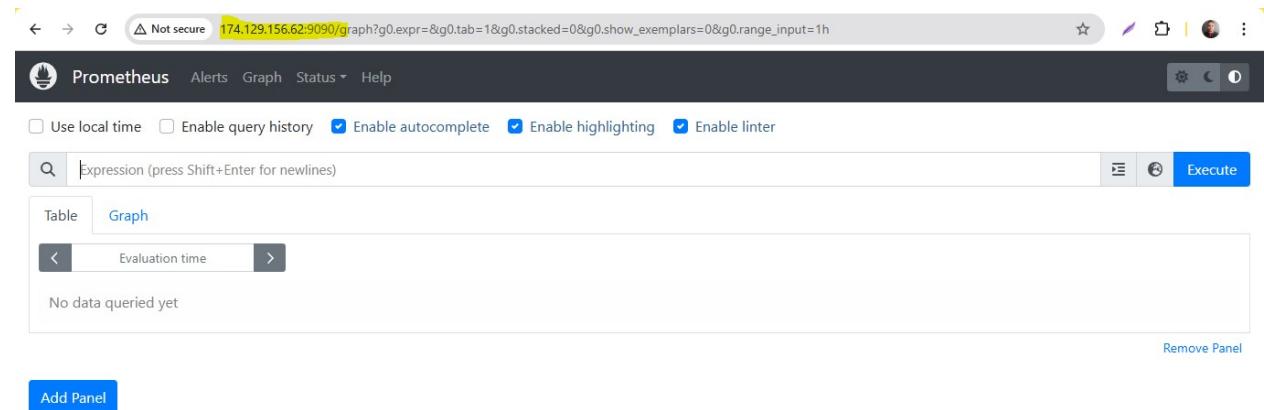
```
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

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
```



The screenshot shows the Prometheus web interface at the URL `174.129.156.62:9090/graph`. The page has a dark header with the Prometheus logo and navigation links for Alerts, Graph, Status, and Help. Below the header is a toolbar with checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete' (which is checked), 'Enable highlighting' (which is checked), and 'Enable linter'. A search bar contains the placeholder 'Expression (press Shift+Enter for newlines)'. To the right of the search bar are buttons for 'Table' (selected), 'Graph' (disabled), and 'Execute'. Below the search bar is a date range selector with 'Evaluation time' and arrows. The main content area displays the message 'No data queried yet'. At the bottom of the interface is a blue button labeled 'Add Panel'.

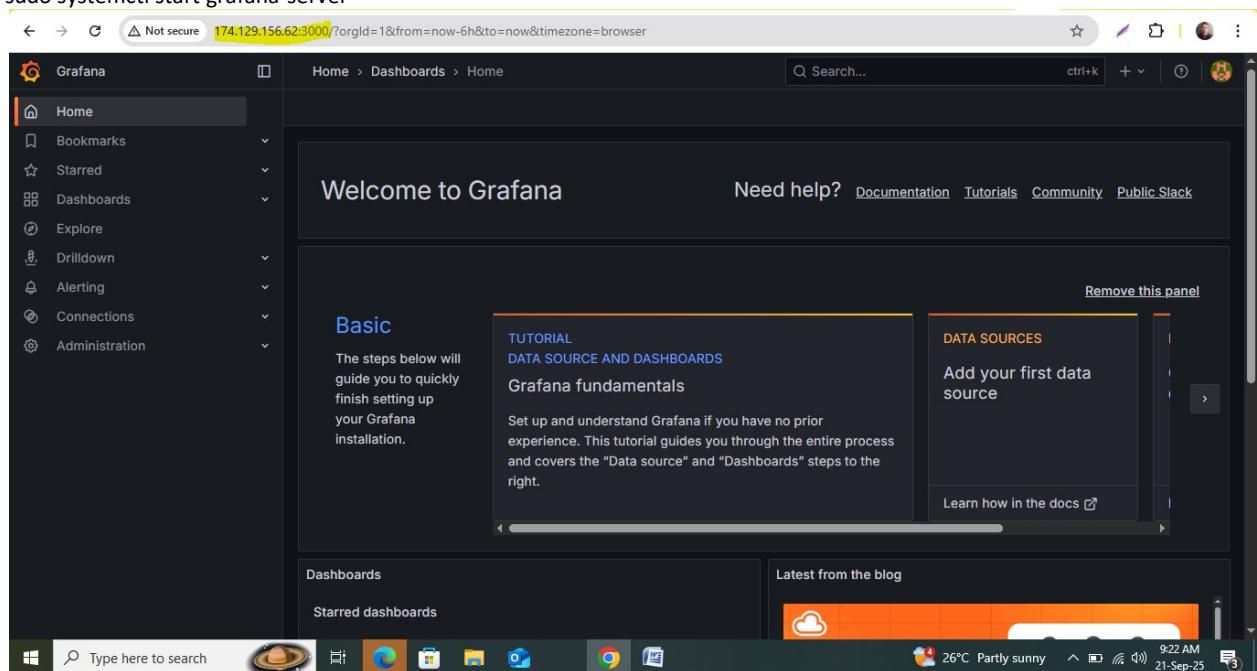


2. Install Grafana

```
# 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

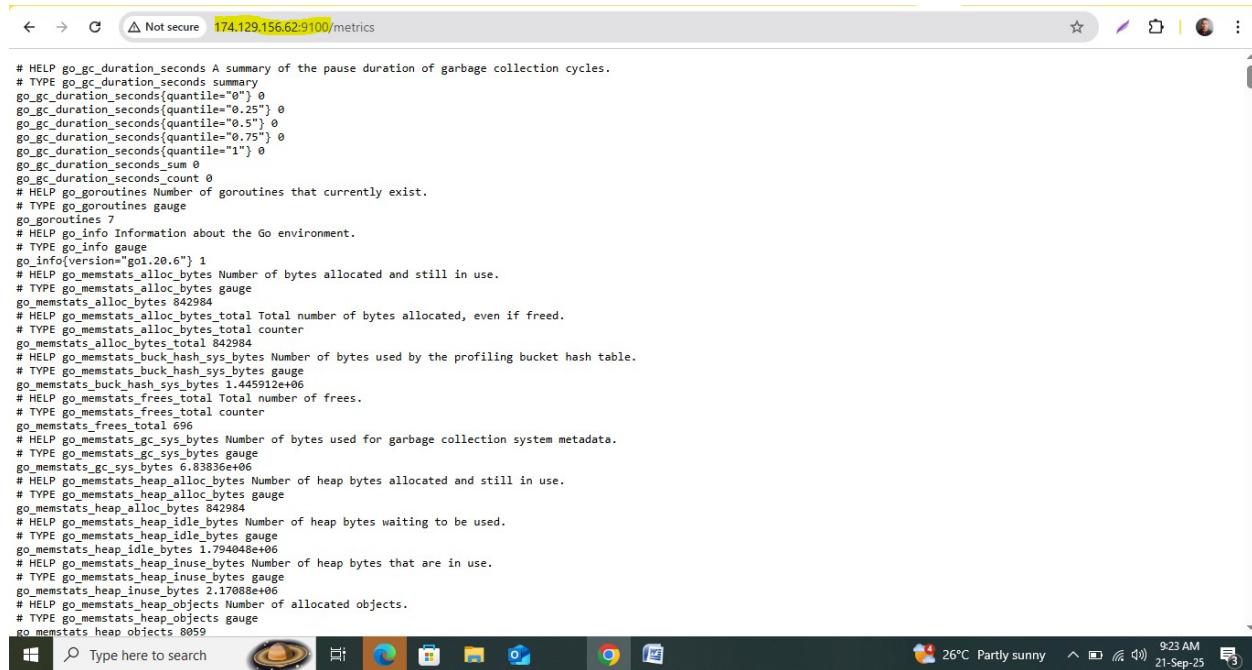
# Install Grafana
sudo apt-get update -y
sudo apt-get install -y grafana

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



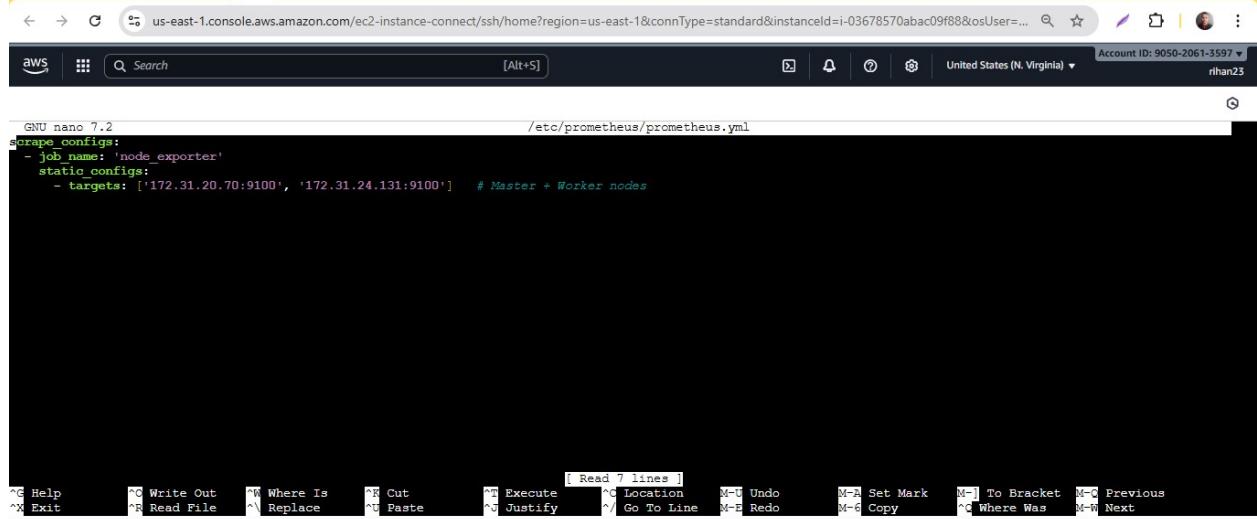
Step 9: Installing node exporter

```
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)



```
GNU nano 7.2                               /etc/prometheus/prometheus.yml
scrape_configs:
- job_name: 'node_exporter'
  static_configs:
    - targets: ['172.31.20.70:9100', '172.31.24.131:9100']  # Master + Worker nodes
```

The terminal window shows the configuration file `/etc/prometheus/prometheus.yml` being edited with `GNU nano 7.2`. The configuration defines a job named `'node_exporter'` with two static targets: `'172.31.20.70:9100'` and `'172.31.24.131:9100'`, which represent the Master and Worker nodes respectively.

Below the terminal window, the AWS CloudShell interface is visible, showing the instance ID `i-03678570abac09f88` and the Graphana URL `(premethious-Graphana)`. The status bar indicates Public IPs: 174.129.156.62 and Private IPs: 172.31.19.142. The system tray shows the date and time as 10:18 AM on 21-Sep-25, with a weather icon for 27°C Partly sunny.

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**

