

Certification Project – Insurance Domain

Submitted by – Mohammed Abu Rihan

Submitted to – Vikul mentor

Batch no - DevOpsSA2504024

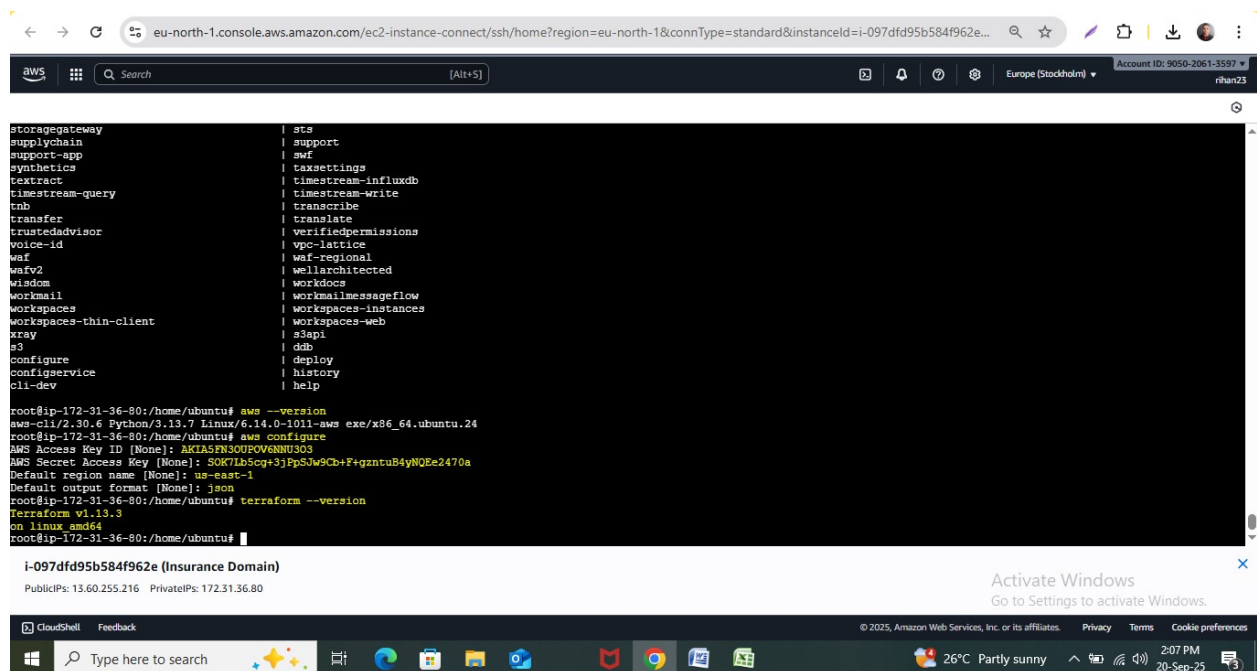
Date of submission – 21-09-2025

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"]  
  }  
}
```

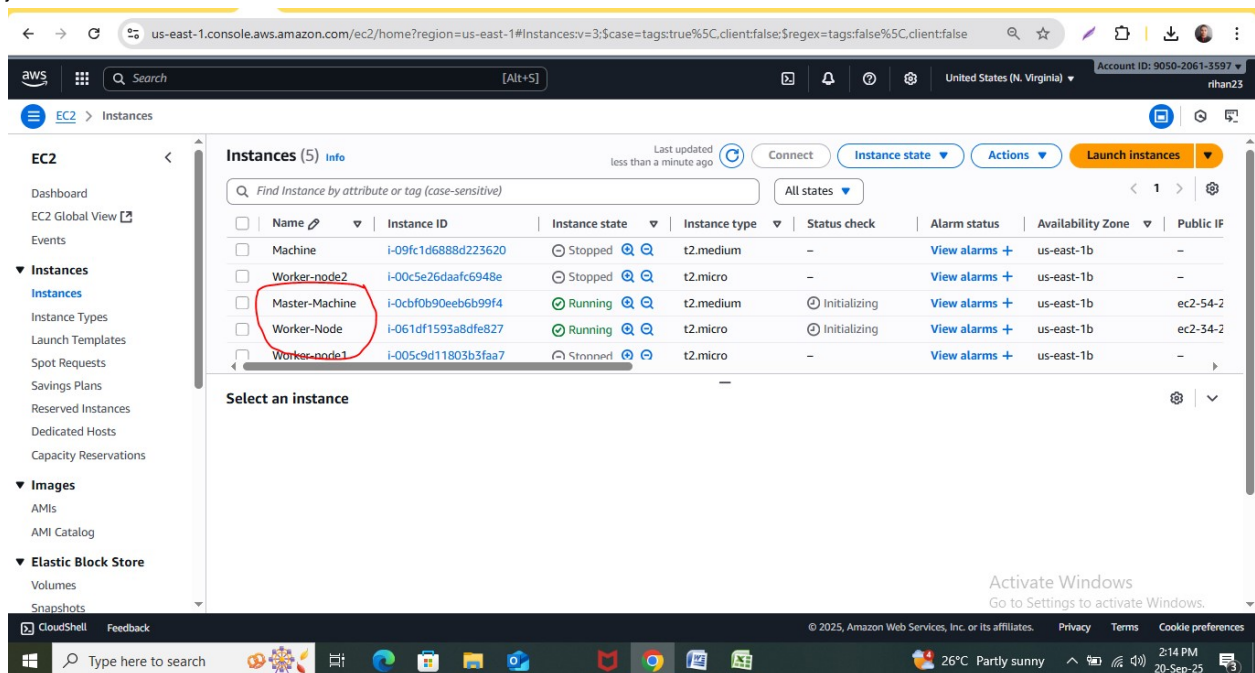
```
}
```

```
# 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"
}
}
```

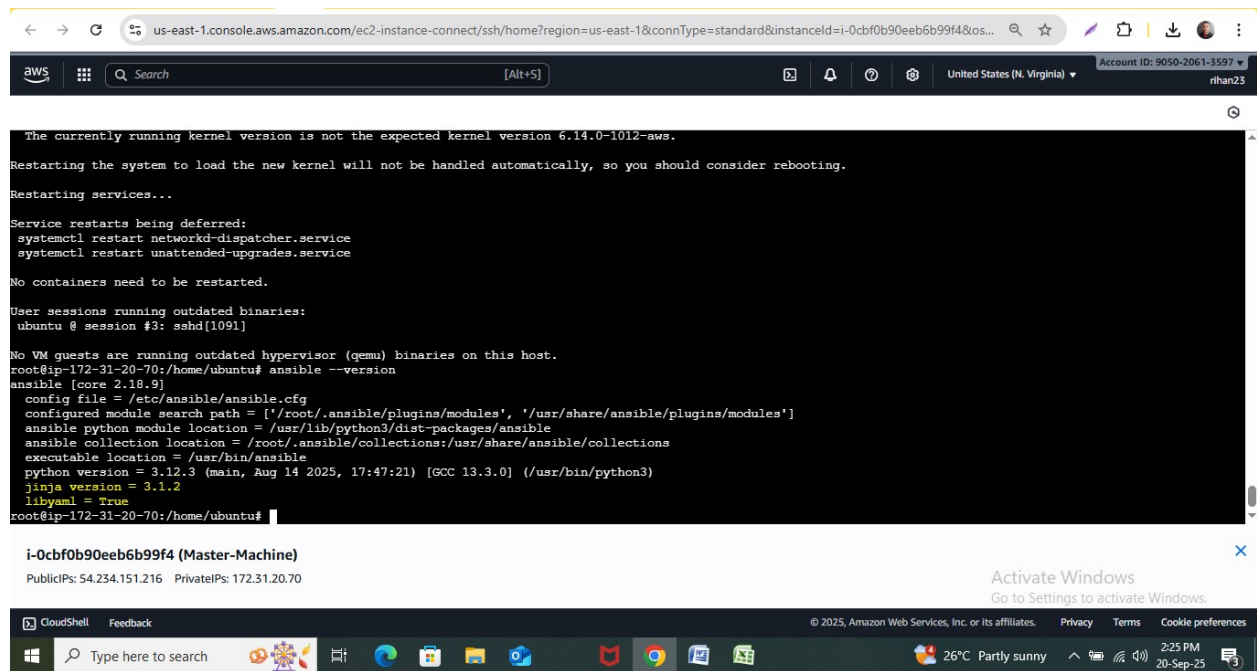


The screenshot displays the AWS Management Console for the 'us-east-1' region, specifically the 'Instances' page. The left sidebar shows the navigation menu with 'Instances' selected. The main content area shows a table of five EC2 instances. The 'Master-Machine' instance is highlighted with a red circle. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP.

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	Initializing	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	-

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#
```

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
```

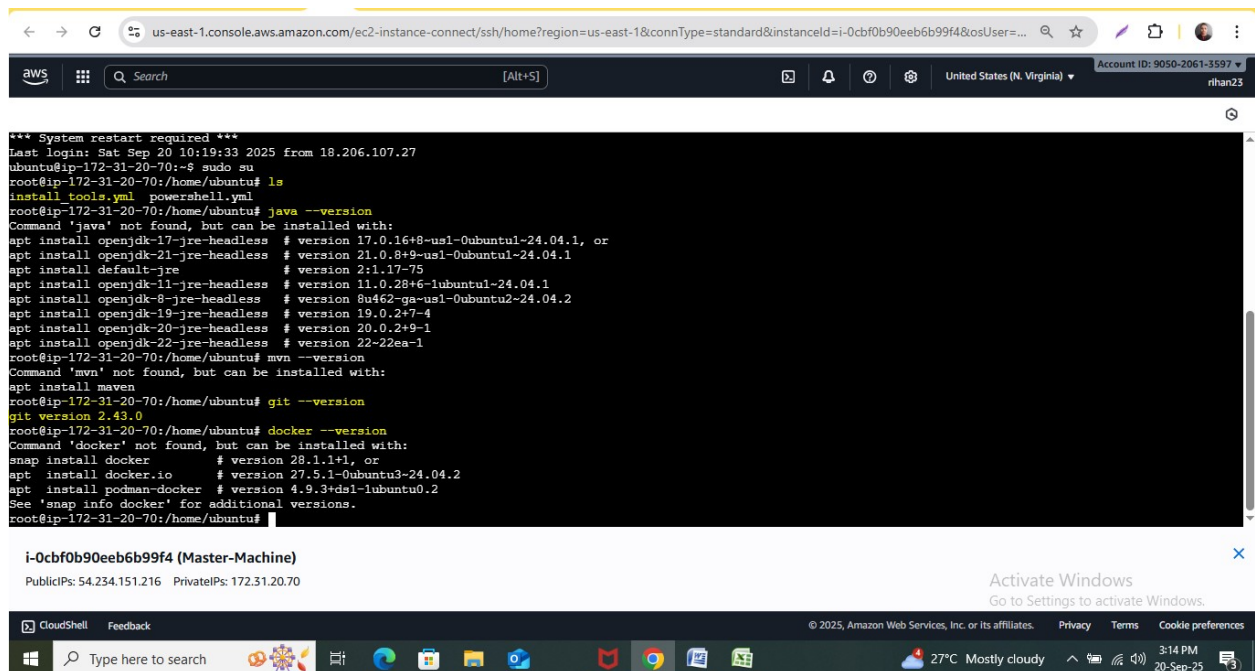
```
apt:
name: "{{ item }}"
state: present
loop:
- apt-transport-https
- ca-certificates
- curl
- software-properties-common
- gnupg-agent

- 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 10:10:33 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-us1-0ubuntu1-24.04.1, or
apt install openjdk-21-jre-headless # version 21.0.8+9-us1-0ubuntu1-24.04.1
apt install default-jre # version 2:1.17-75
apt install openjdk-11-jre-headless # version 11.0.28+6-1ubuntu1-24.04.1
apt install openjdk-8-jre-headless # version 8u462-ga-us1-0ubuntu2-24.04.2
apt install openjdk-19-jre-headless # version 19.0.2+7-4
apt install openjdk-20-jre-headless # version 20.0.2+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 27.5.1-0ubuntu3-24.04.2
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)
PublicIPs: 54.234.151.216 PrivateIPs: 172.31.20.70

Activate Windows
Go to Settings to activate Windows.

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

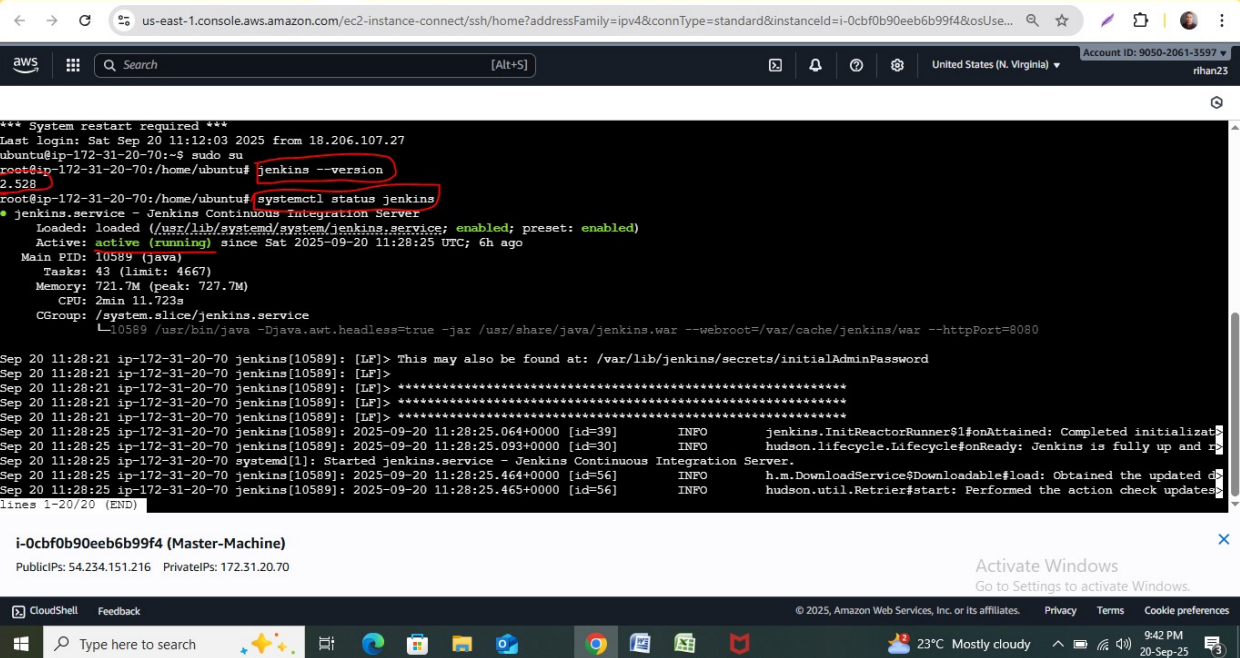
CloudShell Feedback

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
```



The screenshot shows a terminal window with the following content:

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

Below the terminal output, the instance information is displayed:

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

At the bottom of the screenshot, there is a Windows taskbar with the date 20-Sep-25 and time 9:42 PM.

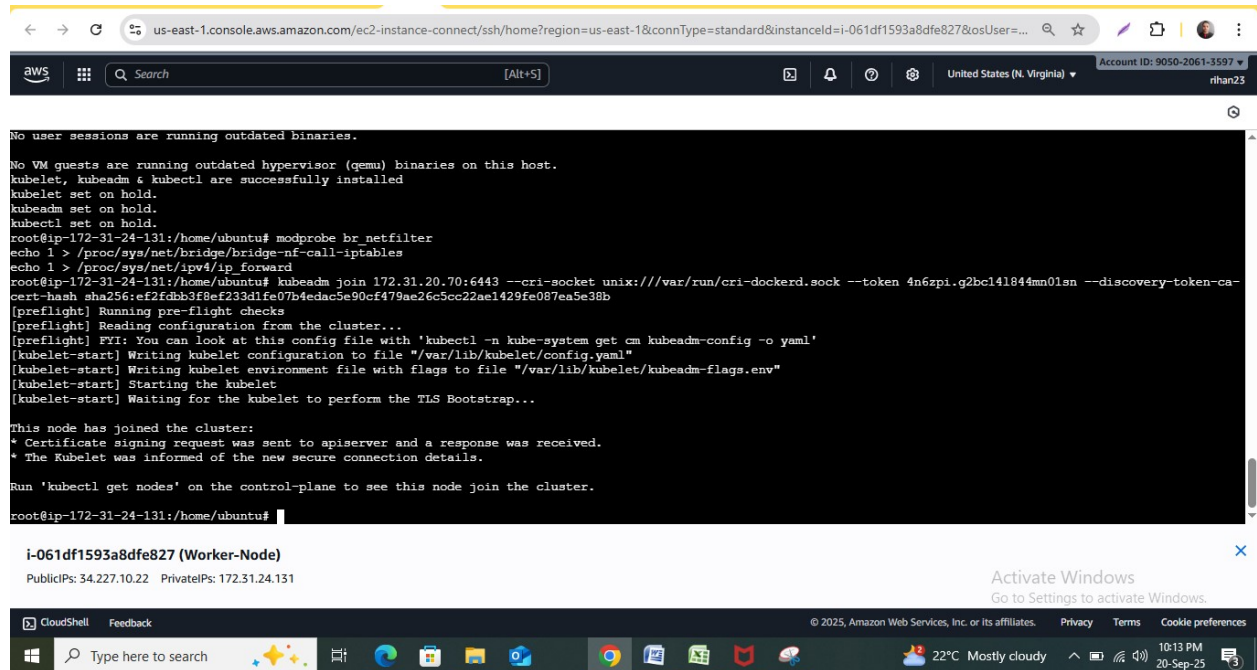
Installed Jenkins plugins now Jenkins is ready to use

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is 54.234.151.216:8080. The Jenkins logo and name are at the top left. Below it, there are links for 'New Item' and 'Build History'. A 'Build Queue' section shows 'No builds in the queue.' and a 'Build Executor Status' section shows '0/2'. The main content area has a 'Welcome to Jenkins!' heading, followed by a message: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this is a 'Start building your software project' section with a 'Create a job' button. Further down is a 'Set up a distributed build' section with buttons for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom right, there is an 'Activate Windows' watermark and a 'REST API' link. The bottom of the browser shows the Windows taskbar with various icons and the system clock showing 10:10 PM on 20-Sep-25.

Step 5: k8s has been installed

The screenshot shows the AWS Management Console in a browser. The address bar indicates the URL is us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-0cbf0b90eeb6b99f4&osUse... The console shows the output of a kubectl command executed on an EC2 instance. The output lists various Kubernetes resources created, including CustomResourceDefinition, ClusterRole, and ClusterRoleBinding. The output also shows the output of the command 'kubectl get nodes', which displays a table with columns NAME, STATUS, ROLES, AGE, and VERSION. The table shows one node: ip-172-31-20-70, Ready, control-plane, 49s, v1.29.15. Below the output, the instance ID i-0cbf0b90eeb6b99f4 (Master-Machine) is shown, along with its PublicIPs (54.234.151.216) and PrivateIPs (172.31.20.70). The bottom of the browser shows the Windows taskbar with various icons and the system clock showing 10:01 PM on 20-Sep-25.

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



```
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
kubelet, kubeadm & kubectl are successfully installed
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-24-131:/home/ubuntu# modprobe br_netfilter
echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
root@ip-172-31-24-131:/home/ubuntu# kubeadm join 172.31.20.70:6443 --cri-socket unix:///var/run/cri-dockerd.sock --token 4n6zpi.g2bc141844mn01sn --discovery-token-ca-cert-hash sha256:e2f5db3f68cf233d1fe07b5edac3e90cf479ae26c5cc22ae1429fe087ea5e38b
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apisserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
root@ip-172-31-24-131:/home/ubuntu#
```

i-061df1593a8dfe827 (Worker-Node)
PublicIPs: 34.227.10.22 PrivateIPs: 172.31.24.131

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 22°C Mostly cloudy 10:13 PM 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 a Jenkins console window for a job named 'demo-insurance' at build #17. The console output displays the execution of a pipeline stage 'Deploying to k8s cluster'. The steps performed are:

- `sh 'sudo kubectl apply -f kubernetesfile.yml'`: This command successfully creates a deployment, service, and replicaset.
- `sh 'sudo kubectl get all'`: This command provides a summary of the deployed resources.

The output of `kubectl get all` is as follows:

```

NAME                                READY    STATUS              RESTARTS   AGE
pod/insureme-64c6b94f4-wl82s        0/1     ContainerCreating    0           0s

NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/insureme                    NodePort      10.101.152.123 <none>         8081:32415/TCP   0s
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           0s

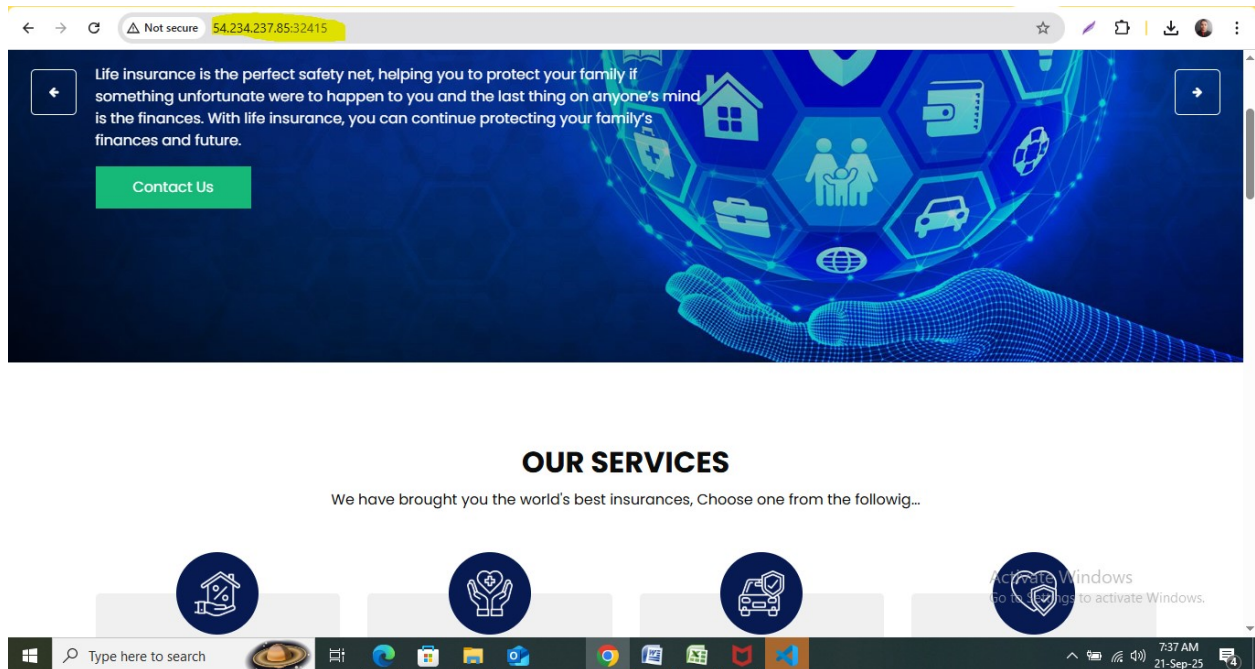
NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/insureme-64c6b94f4  1         1         0       0s

```

The pipeline concludes with the message **Finished: SUCCESS**.

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
```

← → ↻ ⚠ Not secure 174.129.156.62:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h ☆ | 🔍 | 👤 | ⋮

Prometheus Alerts Graph Status ▾ Help ⚙ 🌙 ⓘ

☐ Use local time ☐ Enable query history ☒ Enable autocomplete ☒ Enable highlighting ☒ Enable linter

🔍 Expression (press Shift+Enter for newlines) ⌵ ⓘ Execute

Table Graph

< Evaluation time >

No data queried yet

Remove Panel

Add Panel

Windows Taskbar: Type here to search, File Explorer, Microsoft Edge, Google Chrome, Margot Robbie, Colin...

System Tray: 9:21 AM, 21-Sep-25

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

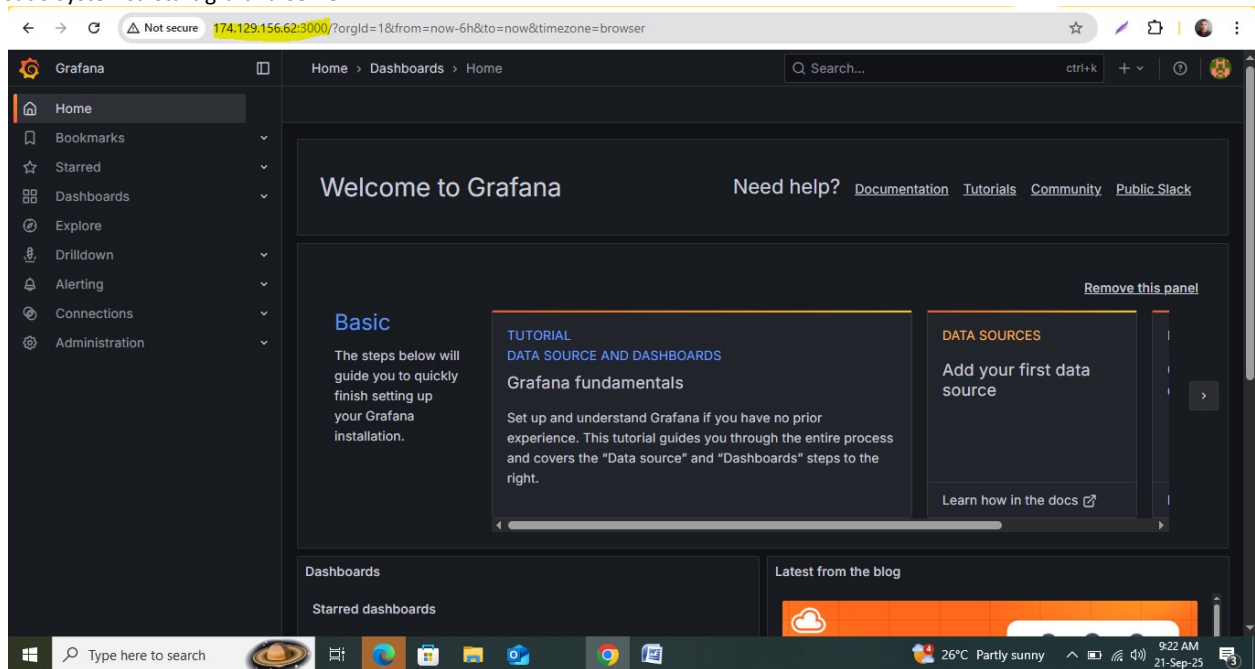
```
sudo apt-get update -y
```

```
sudo apt-get install -y grafana
```

Start Grafana

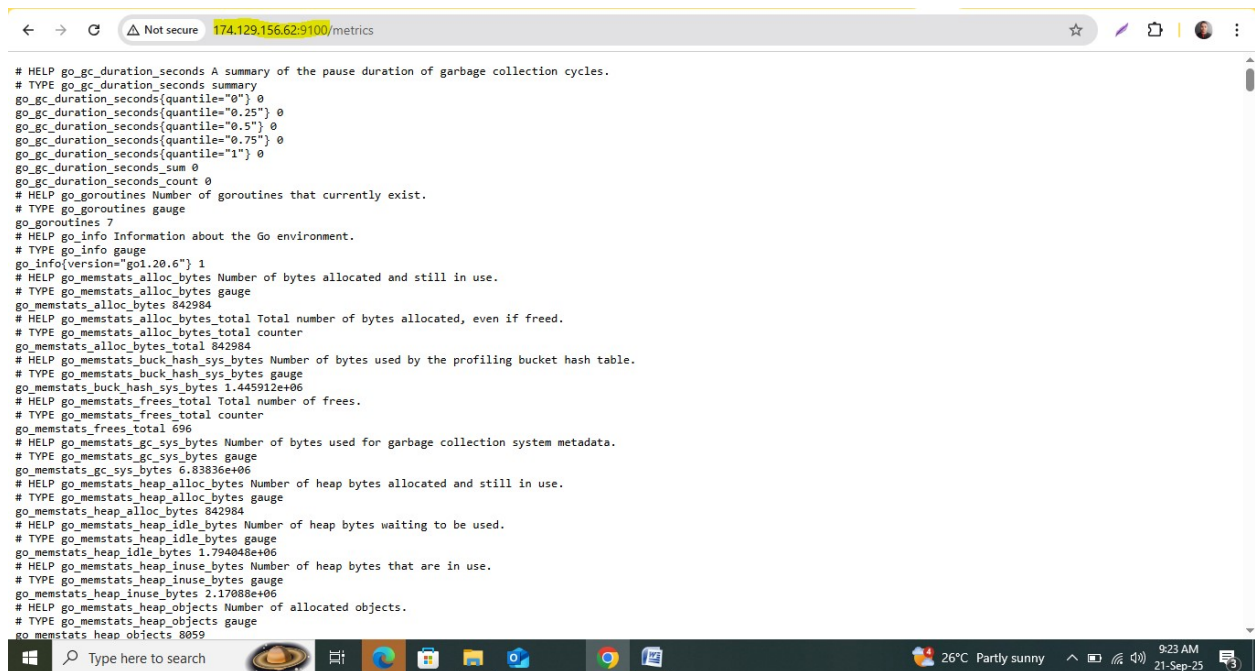
```
sudo systemctl enable grafana-server
```

```
sudo systemctl start grafana-server
```



Step 9: Installing node exporter

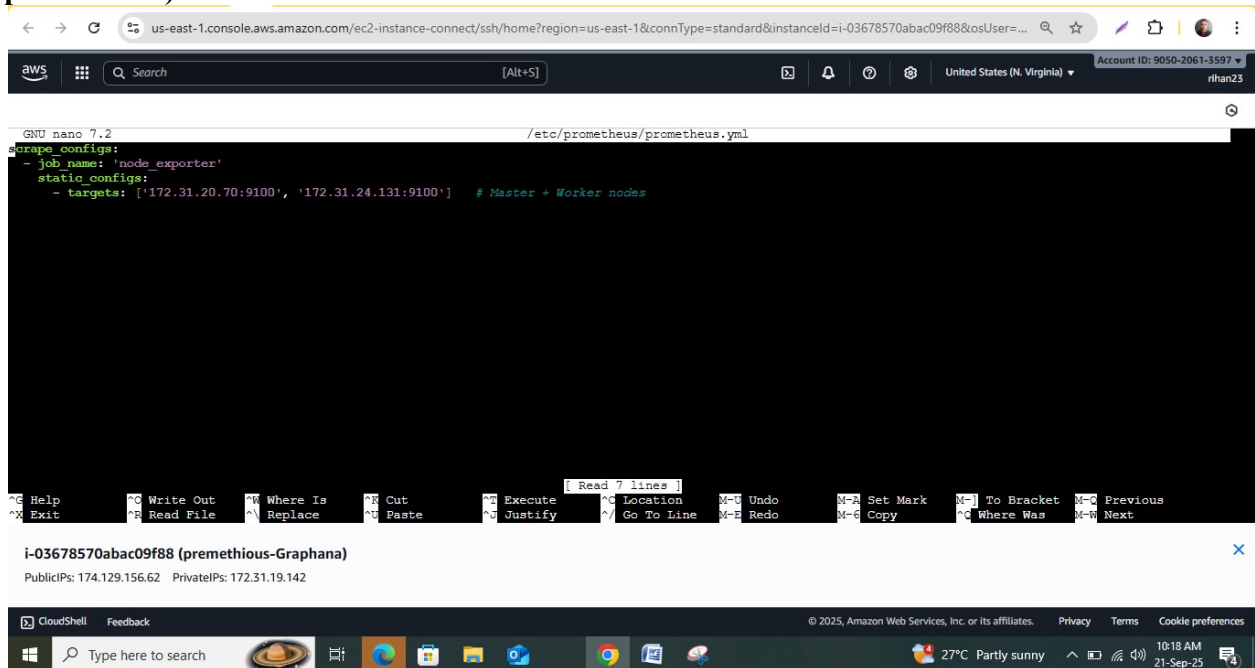
```
cd /tmp
curl -LO https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz
# Extract
tar xvf node_exporter-1.8.1.linux-amd64.tar.gz
sudo mv node_exporter-1.8.1.linux-amd64/node_exporter /usr/local/bin/
# Create a service
sudo tee /etc/systemd/system/node_exporter.service <<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
```



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 7
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.6"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 842984
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 842984
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.445912e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 696
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 6.83836e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 842984
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.794048e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 2.17088e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 8059
```

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
```

i-03678570abac09f88 (premethious-Graphana)
PublicIPs: 174.129.156.62 PrivateIPs: 172.31.19.142

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**

