

# Certification Project – Medicure Healthcare Domain

Submitted by – Mohammed Abu Rihan

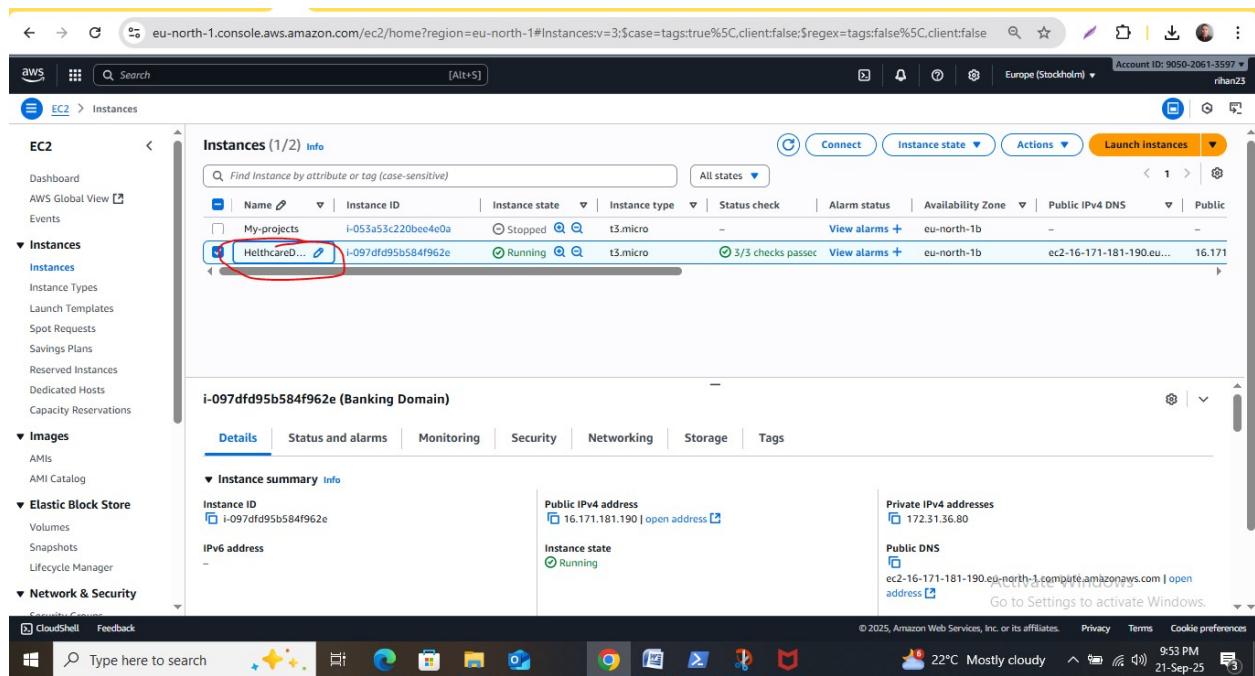
Submitted to – Vikul mentor

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

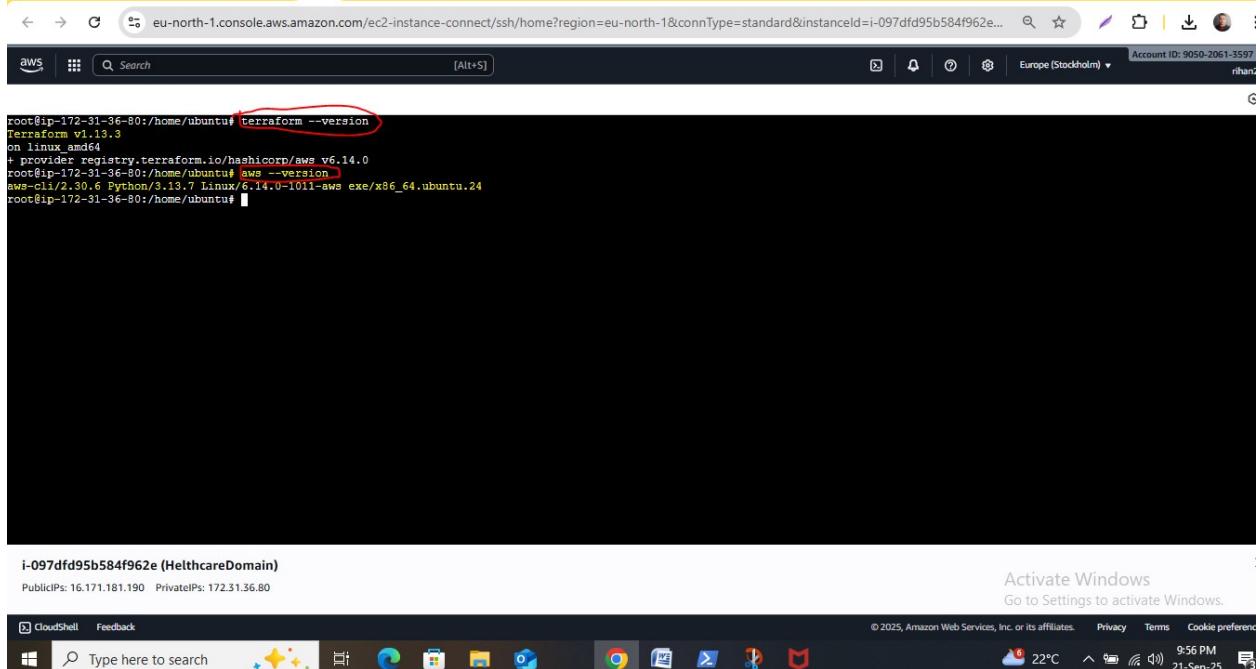


The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like EC2, Dashboard, AWS Global View, Events, Instances (with sub-options like Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (Security Groups). The main content area has a heading 'Instances (1/2) Info' with a search bar and filters. A table lists two instances: one stopped and one running. The running instance is highlighted with a red circle around its name 'Healthcare...'. Below the table, a detailed view for the running instance 'i-097dfd95b584f962e (Banking Domain)' is shown. It includes tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under Details, it shows Instance ID (i-097dfd95b584f962e), Public IPv4 address (16.171.181.190), Private IPv4 address (172.31.36.80), and Public DNS (ec2-16-171-181-190.eu-north-1.compute.amazonaws.com). The instance state is listed as Running.

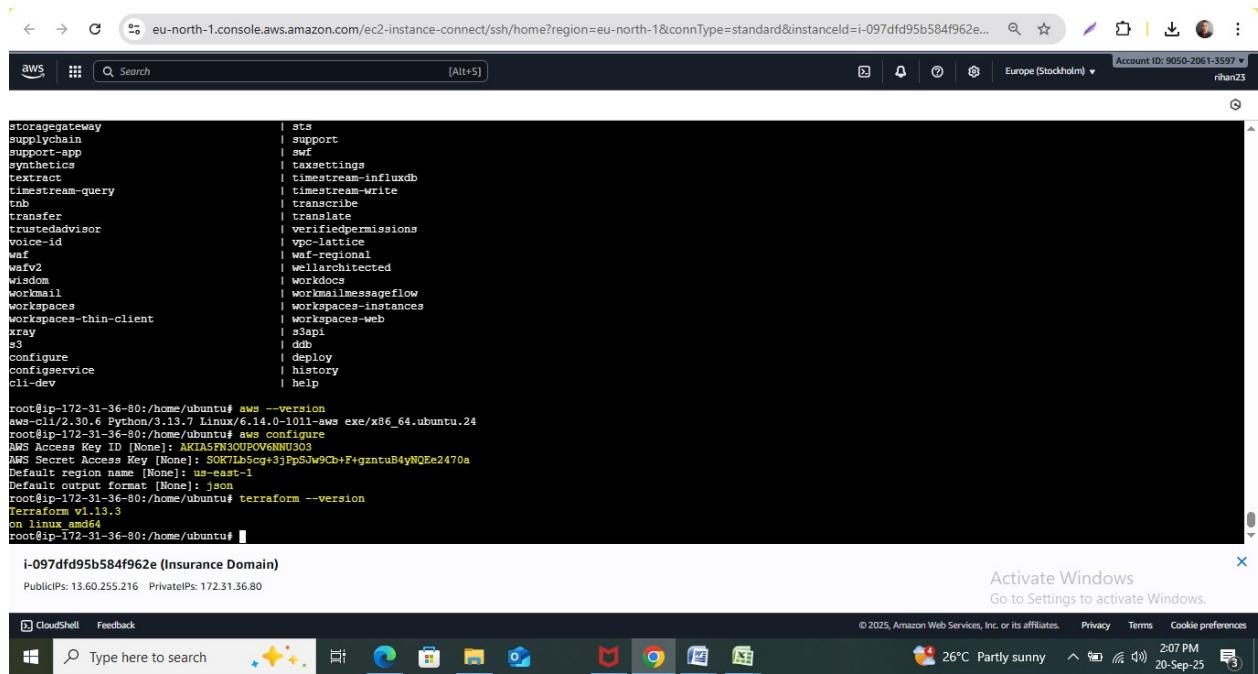
## Step 2: Install Terraform and aws cli using the installation command

```
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



```

storagegateway      | sts
supplychain        | support
support-app        | swf
synthetics         | taxsettings
textract           | timestream-infuxdb
timestream-query   | timestream-write
tnb                | transcribe
transfer           | translate
trustedadvisor     | vehiclepermissions
voice-id           | vpc-lattice
wave               | waf-regional
wafv2              | wellarchitected
wisdom             | workdocs
workmail           | workmailmessageflow
workspaces          | workspaces-instances
workspaces-thin-client | workspaces-web
xray               | s3api
s3                 | ddb
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]: AKIA5FK3OUPOVVNU303
AWS Secret Access Key [None]: SOK7Lb5cg+3jPpSJw9Cd+F+gznTuB4yNQEe2470a
Default region name [None]: us-east-1
Default output format [None]: json
root@ip-172-31-36-80:/home/ubuntu# terraform --version
Terraform v1.19.3
on Linux_amd64
root@ip-172-31-36-80:/home/ubuntu#

```

i-097fd95b584f962 (Insurance Domain)  
 PublicIPs: 13.60.255.216 PrivateIPs: 172.31.36.80

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 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 x 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}"
  }
}

```

**Instances (6) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public
Worker-Node-2	i-0c196a360d17e0688	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54
Worker-node1	i-005c9d11803b3fba7	Stopped	t2.micro	-	View alarms	us-east-1b	-
Master-Node	i-0be88b837bd4c38d8	Running	t3.medium	3/3 checks passed	View alarms	us-east-1b	ec2-98
Worker-Node-1	i-014f0590ba479da42	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54

Select an instance

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

```

customresourcedefinition.apirextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/kubecontrollerconfigurations.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apirextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
root@ip-172-31-24-174:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.24.174:6443 --token $asag7.psmgn8hu0alss4ac --discovery-token-ca-cert-hash sha256:6b650d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443fb6e049834d1
root@ip-172-31-24-174:/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-172-31-24-174:/home/ubuntu#

```

i-0be88b837bd4c38d8 (Master-Node)

Public IPs: 98.89.41.149 Private IPs: 172.31.24.174

## 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
```

```

kubeadm join 172.31.24.174:6443 --token 5ssag7.ps6mn8hu0alss4ac --discovery-token-ca-cert-hash sha256:6b650d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443bf86e049834d1
root@ip-172-31-24-174:/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
  executable location = /root/.ansible/collections:/usr/share/ansible/collections
  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-172-31-24-174:/home/ubuntu# ls
hosts.ini install_jenkins_docker.sh k8s-master.sh nodeexp.sh tools.yml
root@ip-172-31-24-174:/home/ubuntu# java --version
openjdk 21.0.8 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
root@ip-172-31-24-174:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 21.0.8, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.14.0-1011-aws", arch: "amd64", family: "unix"
root@ip-172-31-24-174:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-24-174:/home/ubuntu#

```

**i-0be88b837bd4c38d8 (Master-Node)**

PublicIPs: 98.89.41.149 PrivateIPs: 172.31.24.174

Activate Windows  
Go to Settings to activate Windows.

CloudShell Feedback Type here to search 23°C 10:23 PM 21-Sep-25

## 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
```

```

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-172-31-24-174:/home/ubuntu# ls
hosts.ini  install_jenkins_docker.sh  k8s-master.sh  nodep.sh  tools.yml
root@ip-172-31-24-174:/home/ubuntu# java --version
openjdk 21.0.8 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
root@ip-172-31-24-174:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 21.0.8, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.14.0-1011-aws", arch: "amd64", family: "unix"
root@ip-172-31-24-174:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-24-174:/home/ubuntu# jenkins --version
2.516.3
root@ip-172-31-24-174:/home/ubuntu# docker --version
Docker version 28.4.0, build d8eb465
root@ip-172-31-24-174:/home/ubuntu# ||

```

i-0be88b837bd4c38d8 (Master-Node)

PublicIPs: 98.89.41.149 PrivateIPs: 172.31.24.174

Activate Windows  
Go to Settings to activate Windows.

to installing Jenkins plugins we need to execute commands

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status Jenkins

```

OS name: "linux", version: "6.14.0-1011-aws", arch: "amd64", family: "unix"
root@ip-172-31-24-174:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-24-174:/home/ubuntu# jenkins --version
2.516.3
root@ip-172-31-24-174:/home/ubuntu# docker --version
Docker version 28.4.0, build d8eb465
root@ip-172-31-24-174:/home/ubuntu# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-24-174:/home/ubuntu# sudo systemctl start jenkins
root@ip-172-31-24-174:/home/ubuntu# sudo systemctl status Jenkins
Unit Jenkins.service could not be found.
root@ip-172-31-24-174:/home/ubuntu# sudo systemctl status jenkins
● Jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-09-21 13:28:09 UTC; 5h 13min ago
     Main PID: 19076 (java)
        Tasks: 48 (limit: 4515)
       Memory: 685.7M (peak: 1.0G)
          CPU: 8min 55.61ms
         CGroup: /system.slice/jenkins.service
             └─19076 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Notice: journal has been rotated since unit was started, output may be incomplete.
root@ip-172-31-24-174:/home/ubuntu# ||

```

i-0be88b837bd4c38d8 (Master-Node)

PublicIPs: 98.89.41.149 PrivateIPs: 172.31.24.174

Activate Windows  
Go to Settings to activate Windows.

## Installed Jenkins plugins now Jenkins is ready to use

The screenshot shows the Jenkins dashboard at the URL <http://98.89.41.149:8080>. The title bar indicates it's a 'Not secure' connection. The dashboard features a 'Jenkins' logo and navigation links for 'New Item' and 'Build History'. A prominent 'Welcome to Jenkins!' message is displayed, along with instructions for setting up distributed builds. Below this, there are sections for 'Build Queue' (empty), 'Build Executor Status' (0/2), and a 'Create a job' button. A 'Set up a distributed build' section includes links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom of the screen shows a Windows taskbar with various icons and system status information.

## Step 7: Installing K8s using installation commands

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

```

Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.29.15-1.1_amd64.deb ...
Unpacking kubelet (1.29.15-1.1) ...
Setting up conctrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.29.15-1.1) ...
Setting up cri-tools (1.29.0-1.1) ...
Setting up kubernetes-cni (1.3.0-1.1) ...
Setting up kubelet (1.29.15-1.1) ...
Setting up kubeadm (1.29.15-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
kubelet, kubeadm & kubectl are successfully installed
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-25-214:/home/ubuntu# modprobe br_netfilter

```

**i-014f0590ba479da42 (Worker-Node-1)**  
PublicIPs: 34.224.100.156 PrivateIPs: 172.31.25.214

Activate Windows  
Go to Settings to activate Windows.

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

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
```

```

Unpacking kubelet (1.29.15-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../4-kubelet_1.29.15-1.1_amd64.deb ...
Unpacking kubelet (1.29.15-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../5-kubeadm_1.29.15-1.1_amd64.deb ...
Unpacking kubeadm (1.29.15-1.1) ...
Setting up conctrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.29.15-1.1) ...
Setting up cri-tools (1.29.0-1.1) ...
Setting up kubernetes-cni (1.3.0-1.1) ...
Setting up kubelet (1.29.15-1.1) ...
Setting up kubeadm (1.29.15-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
kubelet, kubeadm & kubectl are successfully installed

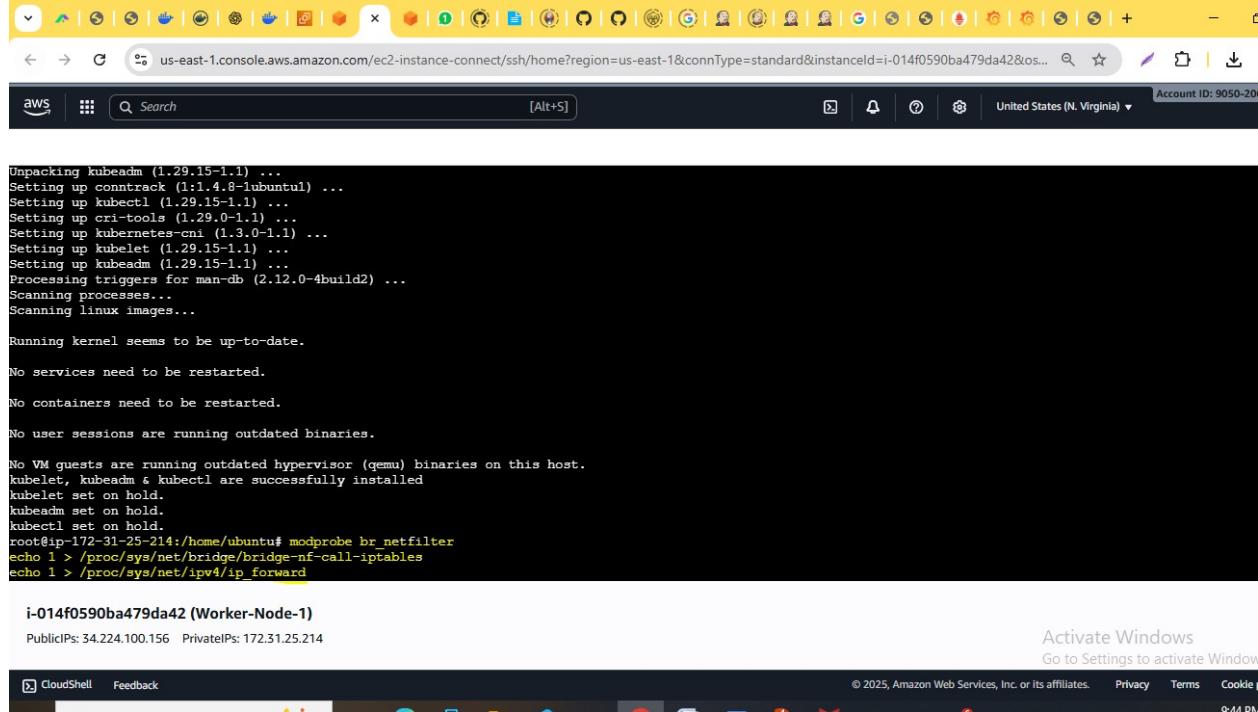
i-014f0590ba479da42 (Worker-Node-1)
PublicIPs: 34.224.100.156 PrivateIPs: 172.31.25.214
```

Activate Windows  
Go to Settings to activate Windows.

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

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
```



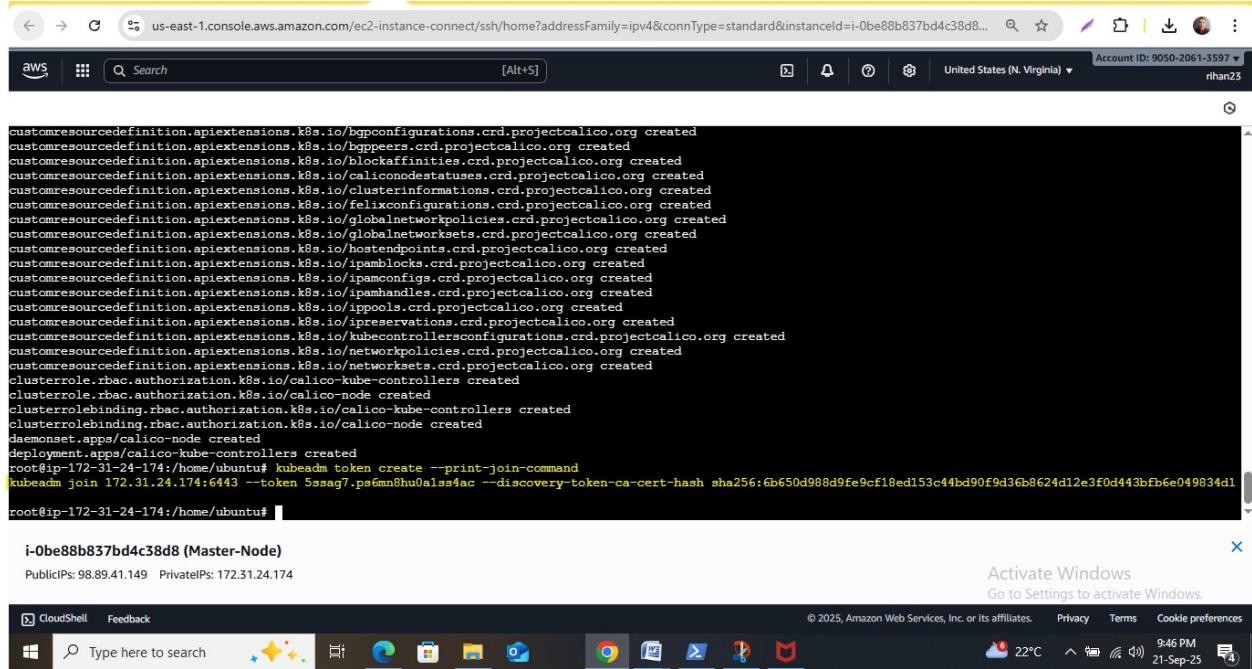
The screenshot shows a browser-based AWS CloudShell interface. The URL is `us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=us-east-1&connType=standard&instanceId=i-014f0590ba479da42&os...`. The AWS logo is at the top left, and the account ID is 9050-20. The main area is a terminal window displaying the output of the following commands:

```
Unpacking kubelet (1.29.15-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubectl (1.29.15-1.1) ...
Setting up cri-tools (1.29.0-0.1) ...
Setting up kubernetes-cni (1.3.0-1.1) ...
Setting up kubelet (1.29.15-1.1) ...
Setting up kubeadm (1.29.15-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
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.
kubelet, kubeadm & kubectl are successfully installed
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
root@ip-172-31-25-214:/home/ubuntu# modprobe br_netfilter
echo 1 > /proc/sys/net/bridge/bridge-nf-call-iptables
echo 1 > /proc/sys/net/ipv4/ip_forward
```

At the bottom of the terminal window, it says **i-014f0590ba479da42 (Worker-Node-1)**, PublicIPs: 34.224.100.156 PrivateIPs: 172.31.25.214. To the right, there are links for "Activate Windows" and "Go to Settings to activate Window". The footer includes "CloudShell Feedback", "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", "Cookie", and the time "9:44 PM".

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



```
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
root@ip-172-31-24-174:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.24.174:6443 --token 5ssag7.ps6mn8hu0a1ss4ac --discovery-token-ca-cert-hash sha256:6b650d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443bf6e049834d1
root@ip-172-31-24-174:/home/ubuntu#
```

i-0be88b837bd4c38d8 (Master-Node)  
Public IPs: 98.89.41.149 Private IPs: 172.31.24.174

Activate Windows  
Go to Settings to activate Windows.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 9:46 PM 22°C 21-Sep-25

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 sha256:6b650d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443bf6e049834d1

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

```
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (gemu) binaries on this host.  
kublet, kubeADM & kubectl are successfully installed  
kublet set on hold.  
kubeADM set on hold.  
kubectl set on hold.  
root@ip-172-31-25-214:/home/ubuntu# modprobe br_netfilter  
echo 1 > /proc/sys/net/bridges/bridge-nf-call-ipTables  
echo 1 > /proc/sys/net/ipv4/ip_forward  
root@ip-172-31-25-214:/home/ubuntu# kubeadm join 172.31.24.174:6443 --cri-socket unix:///var/run/cri-dockerd.sock --token 5sseag7.ps6mn0hu0alss4ac --discovery-token-ca-cert-hash sha256:6b50d988d9fe9cf18ed153c44bd90f9d36b8624d12e3f0d443bfb6e049834d1  
[preflight] Running pre-flight checks  
[preflight] Reading configuration from the cluster...  
[preflight] FII: You can look at this config file with 'kubectl -n kube-system get cm kubeADM-config -o yaml'  
[kublet-start] Writing kublet configuration to file "/var/lib/kublet/config.yaml"  
[kublet-start] Writing kubelet environment file with flags to file "/var/lib/kublet/kubeADM-flags.env"  
[kublet-start] Starting the kublet  
[kublet-start] Waiting for the kublet to perform the TLS Bootstrap...  
  
This node has joined the cluster:  
* Certificate signing request was sent to apiserver 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-172-31-25-214:/home/ubuntu#  
  
i-014f0590ba479da42 (Worker-Node-1)  
Public IPs: 34.224.100.156 Private IPs: 172.31.25.214
```

Activate Windows  
Go to Settings to activate Windows.

CloudShell Feedback Type here to search 22°C 9:48 PM 21-Sep-25

**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'
}
}
}
}
}

```

The screenshot shows a Jenkins pipeline console output. The logs indicate the deployment of an application to Kubernetes, including the application of a deployment configuration and the listing of pods and services. The Kubernetes status table shows one pod named 'insureme-55b4c967f6-wmnrq' in a 'Running' state with a status of '1/1'. The service table shows two entries: 'insureme' as a LoadBalancer service and 'kubernetes' as a ClusterIP service.

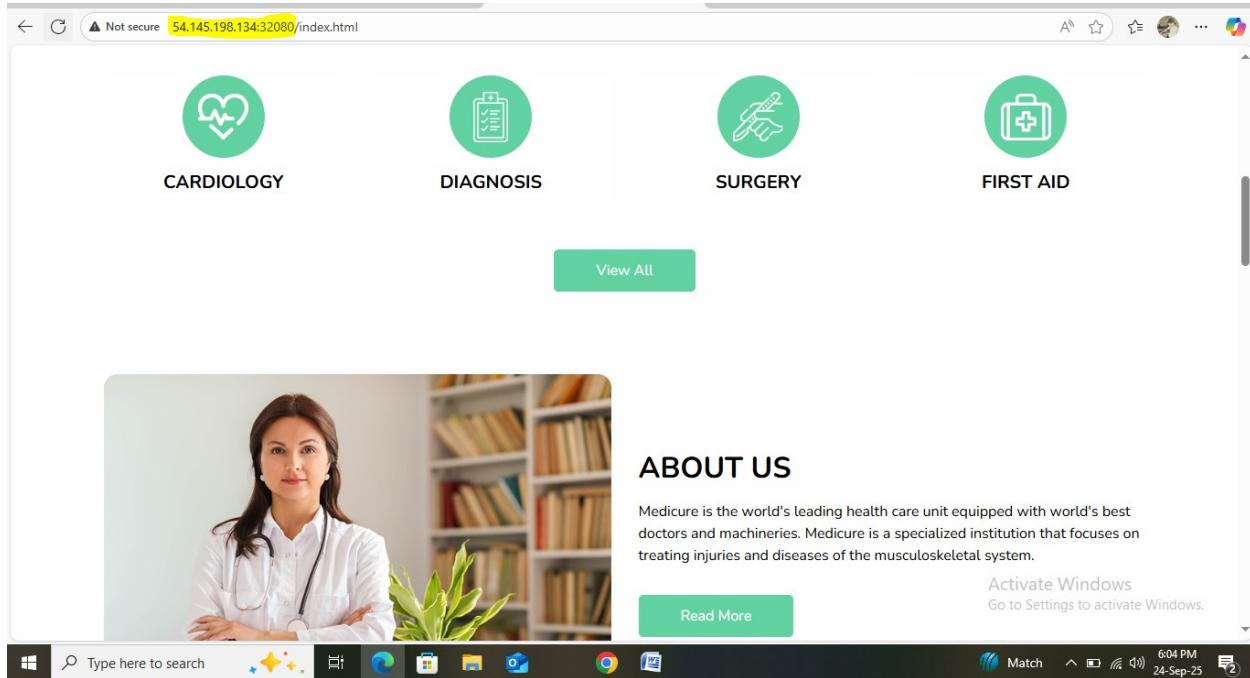
NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE
insureme-55b4c967f6-wmnrq	1/1	Running	1 (14h ago)	14h	192.168.227.201	ip-172-31-25-214	<none>

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
insureme	LoadBalancer	10.101.232.189	<pending>	8082:32880/TCP	15h
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	2d20h

**Activate Windows  
Go to Settings to activate Windows.**

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
```

Prometheus

Not secure 34.227.194.129:9090/graph?g0.expr=&g0.tab=1&g0.display\_mode=lines&g0.show\_exemplars=0&g0.range\_input=1h

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

Add Panel Remove Panel

Activate Windows  
Go to Settings to activate Windows.



## 2. Install Grafana

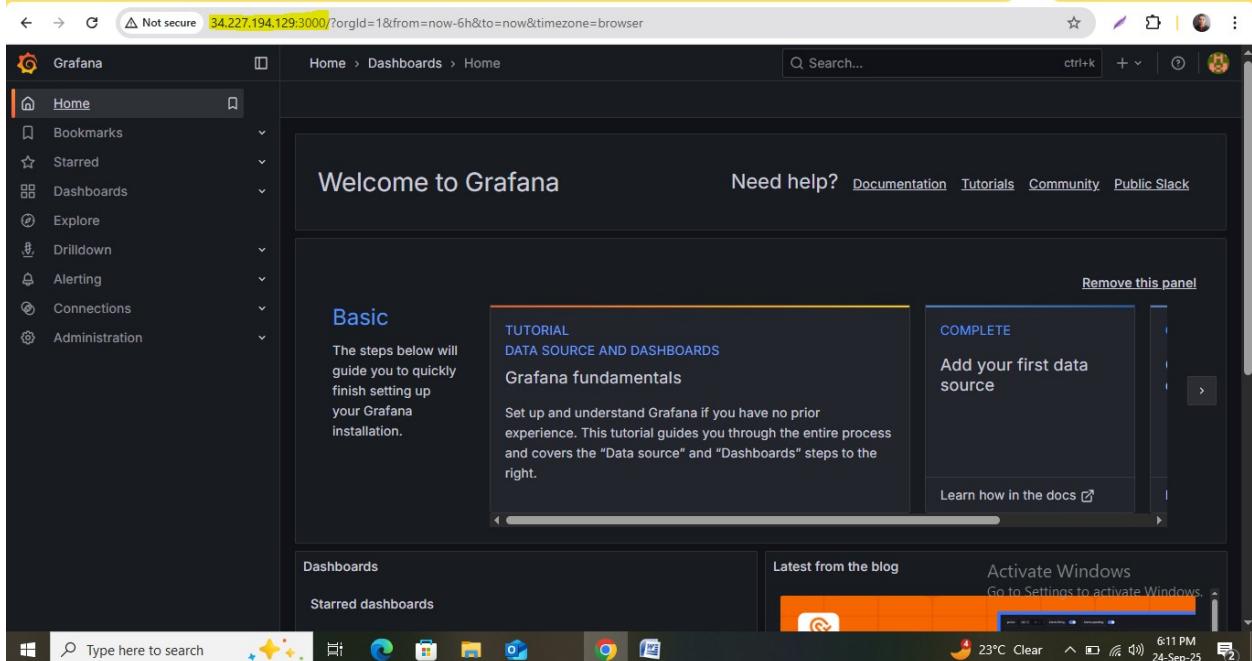
```

# 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

```



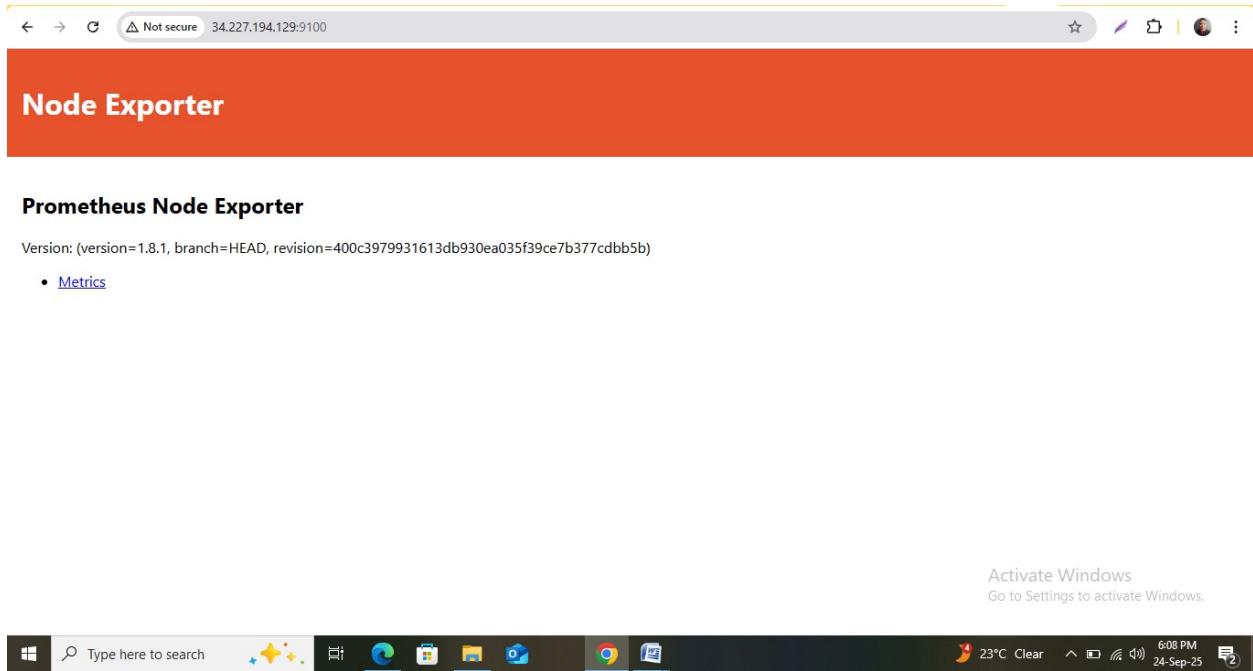
### 3. now we need to install node exporter using the command in both master and one of the worker node

```

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)**

```

GNU nano 7.2
/etc/prometheus/prometheus.yml
scrape_configs:
- job_name: 'node_exporter'
  static_configs:
    - targets: ['172.31.24.174:9100', '172.31.17.244:9100'] # Master + Worker nodes

```

i-0c196a360d17e0688 (Worker-Node-2)  
Public IPs: 34.227.194.129 Private IPs: 172.31.17.244

Activate Windows  
Go to Settings to activate Windows.

## 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**

