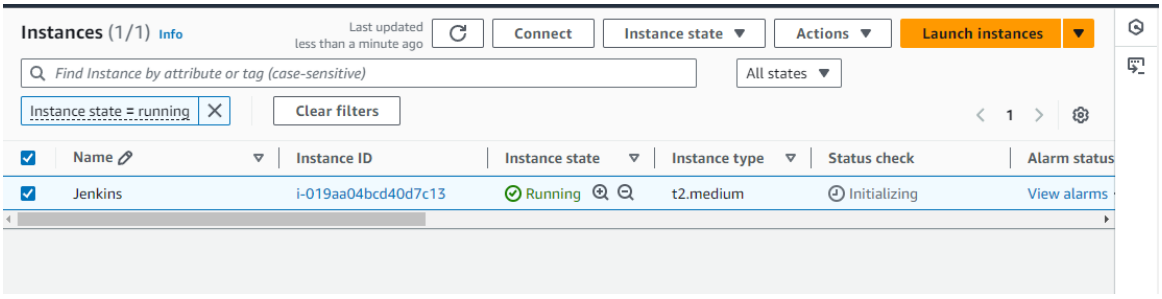
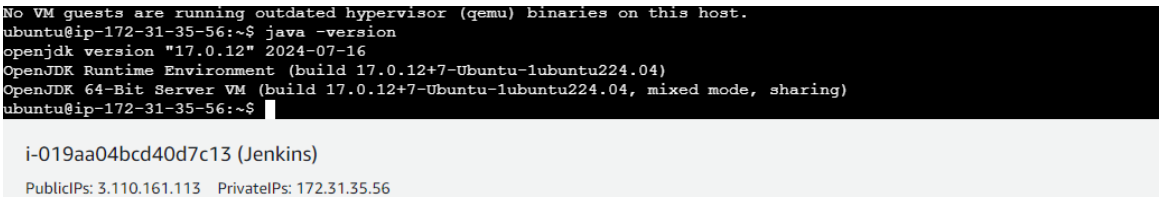
Banking and Finance Domain Project

Step1:

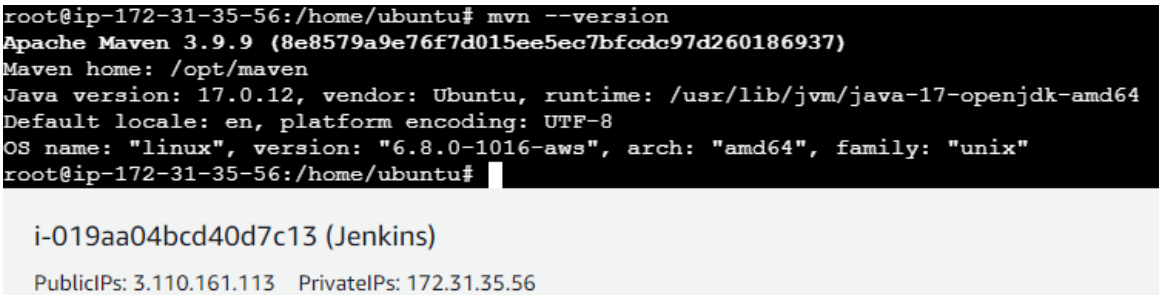
Create an instance named Jenkins



- Install Java and Maven which we need in the Jenkins tools part



- Maven install



- Checking the path

```
root@ip-172-31-35-56:~# nano ~/.bashrc
root@ip-172-31-35-56:~# vi .bashrc
root@ip-172-31-35-56:~# nano ~/.bashrc
root@ip-172-31-35-56:~# source ~/.bashrc
root@ip-172-31-35-56:~# echo $JAVA_HOME
/usr/lib/jvm/java-17-openjdk-amd64
root@ip-172-31-35-56:~# echo $MAVEN_HOME
/opt/maven
root@ip-172-31-35-56:~#
```

i-019aa04bcd40d7c13 (Jenkins)

PublicIPs: 3.110.161.113 PrivateIPs: 172.31.35.56

- **To see if maven is working**

```
root@ip-172-31-35-56:~# mvn
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.083 s
[INFO] Finished at: 2024-10-28T06:33:54Z
[INFO] ------------------------------------------------------------------------
[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goa
plugin-group-id>:<plugin-artifact-id>[:<plugin-version>]:<goal>. Available lifecycle phases are: pre-c
ze, generate-sources, process-sources, generate-resources, process-resources, compile, process-classes
es, generate-test-resources, process-test-resources, test-compile, process-test-classes, test, prepare
tegration-test, post-integration-test, verify, install, deploy, pre-site, site, post-site, site-deploy
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articl
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoGoalSpecifiedException
root@ip-172-31-35-56:~#
```

i-019aa04bcd40d7c13 (Jenkins)

PublicIPs: 3.110.161.113 PrivateIPs: 172.31.35.56

Step 2

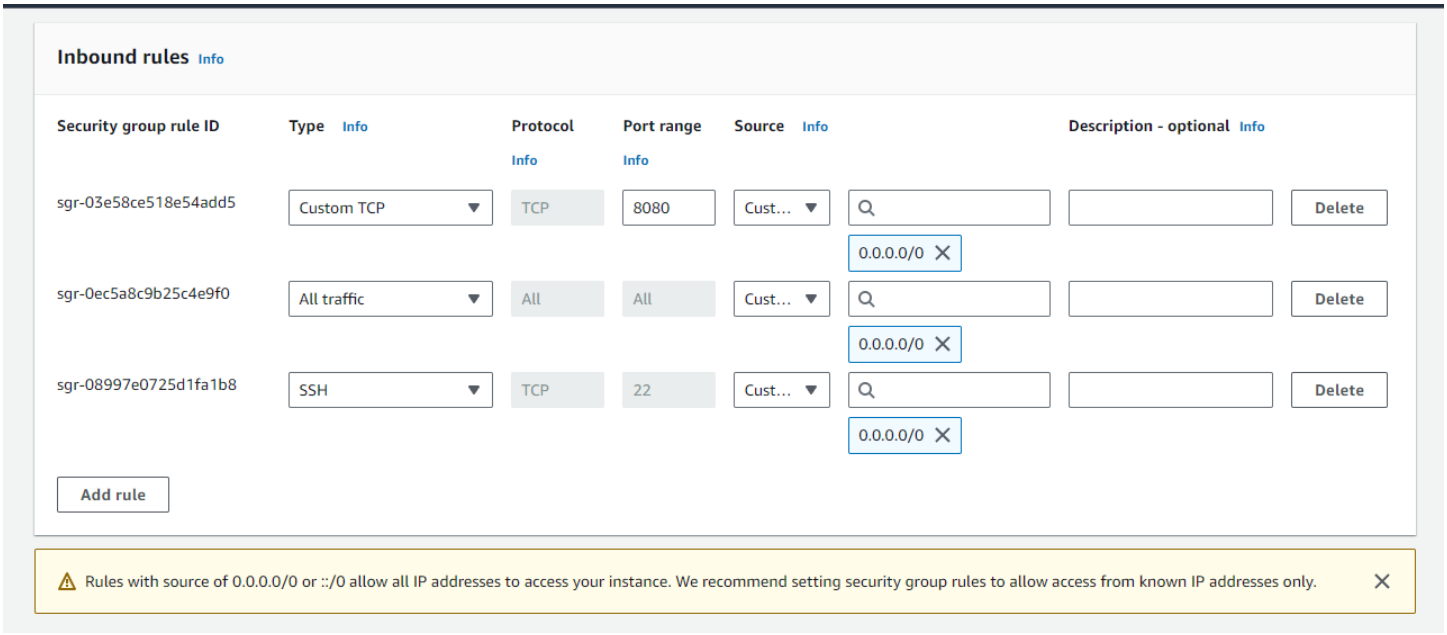
- Installing jenkins

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/usr/share/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

sudo systemctl enable jenkins

sudo systemctl start jenkins

sudo systemctl status jenkins
~
~
```

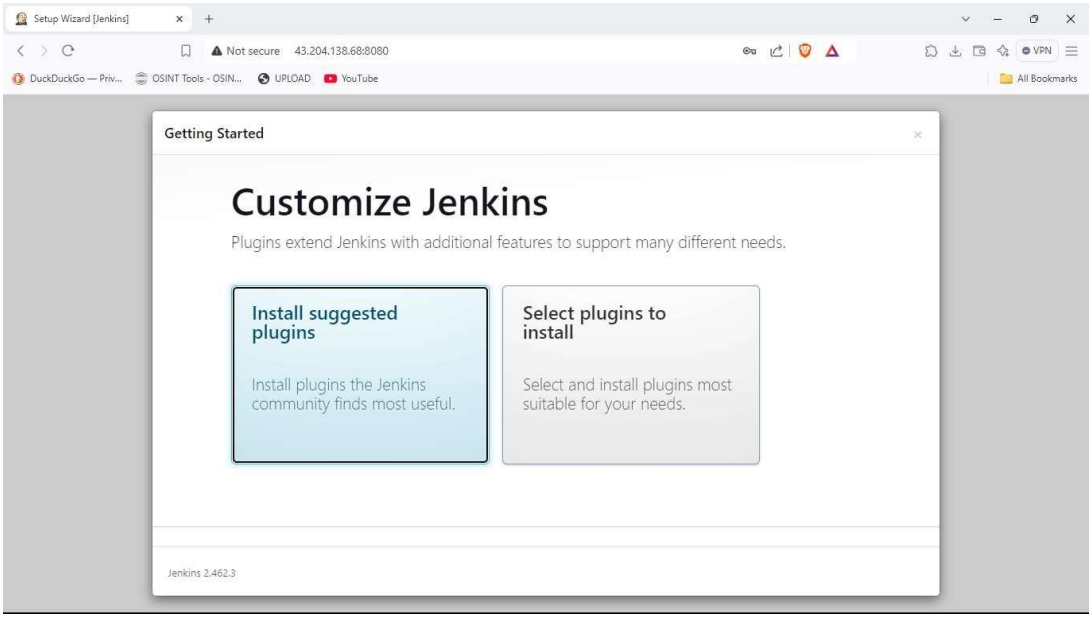
- Verifying the Jenkins installed

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
85eb092f36214b09a86b61c2399a956d
root@ip-172-31-26-213:/home/ubuntu# java --version
openjdk 17.0.12 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu224.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu224.04, mixed mode, sharing)
root@ip-172-31-26-213:/home/ubuntu# jenkins --version
2.462.3
root@ip-172-31-26-213:/home/ubuntu#
```

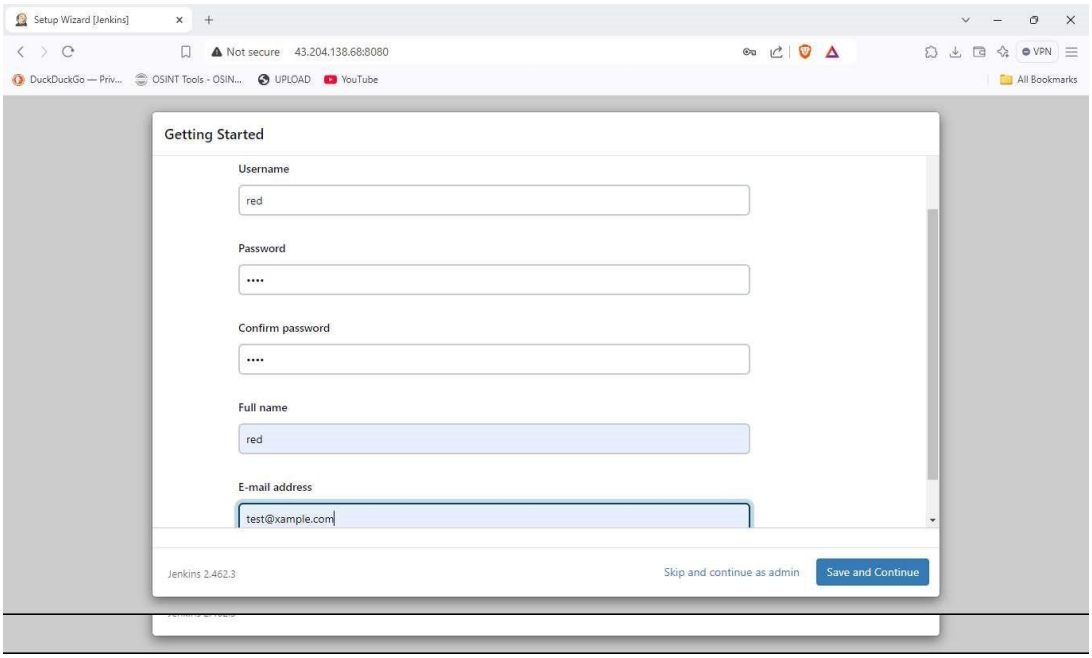
- Allow 8080 port in the security groups

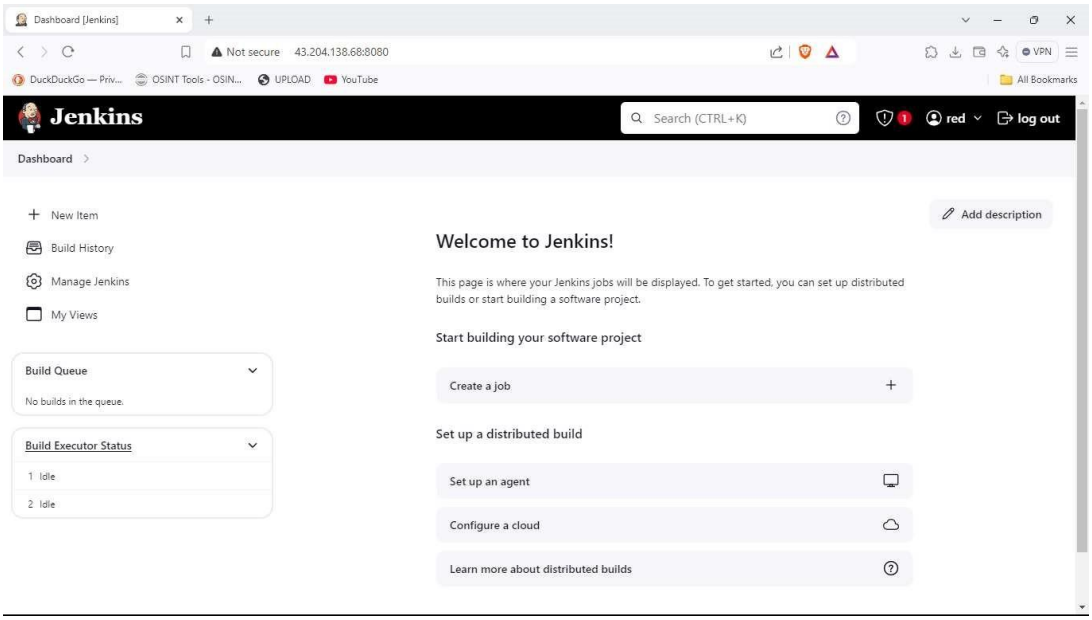


- Copy the public IP of jenkin machine with port 8080



- Give the username and password to use jenkins

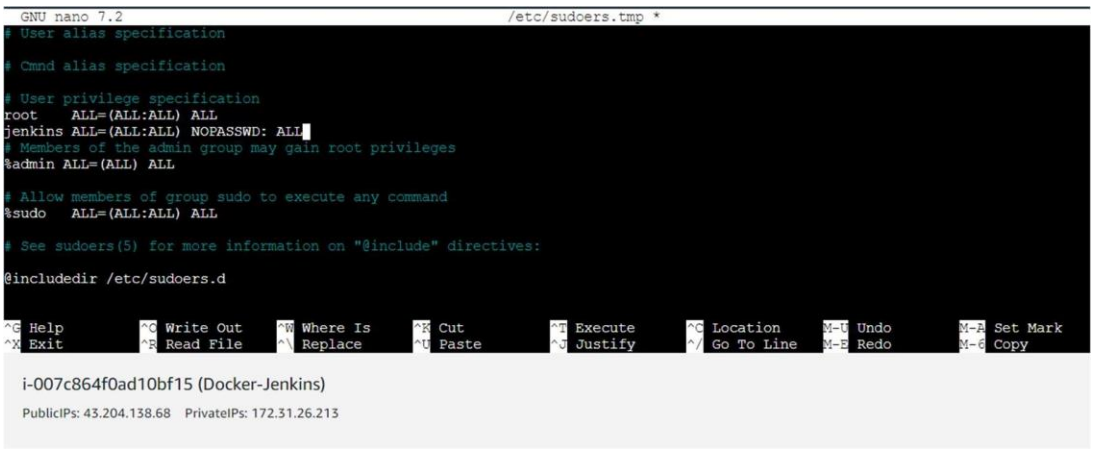




Step 4:

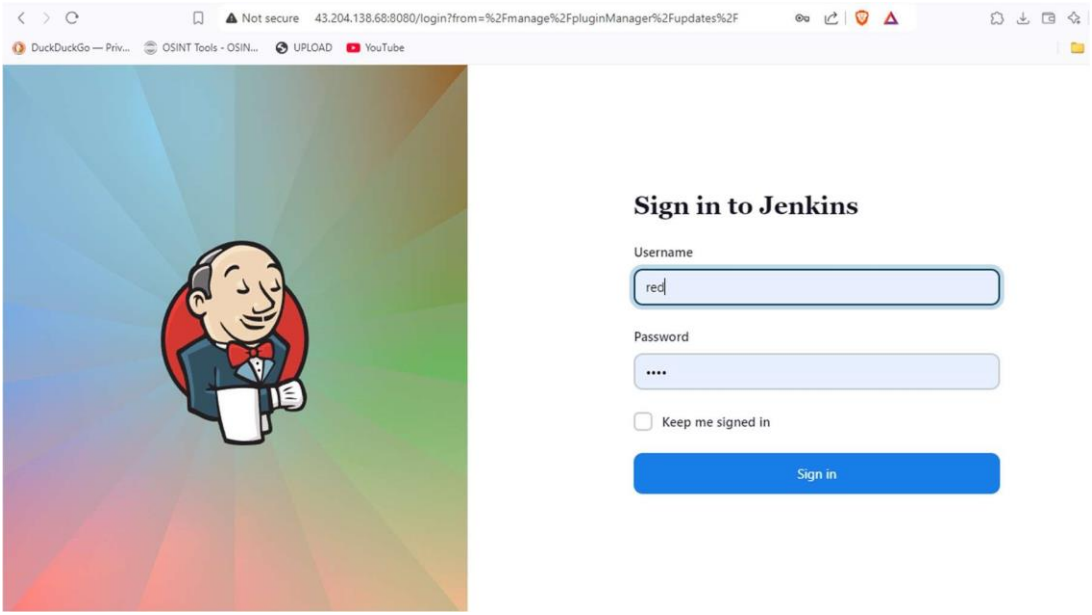
- Before starting using jenkins pipeline we need to allow the Jenkins user to execute any command without being prompted for a password

```
root@ip-172-31-26-213:/home/ubuntu# visudo
```



- Restart the jenkins and login again

service jenkins restart



Step 5:

We need to install docker in the same machine and then we will give jenkins the permission to access docker

sudo apt install docker.io -y

sudo usermod -aG docker jenkins

service docker restart

```
root@ip-172-31-35-56:~# systemctl restart docker
root@ip-172-31-35-56:~# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-10-28 06:49:40 UTC; 3s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 6678 (dockerd)
      Tasks: 9
     Memory: 24.0M (peak: 24.3M)
        CPU: 236ms
     CGroup: /system.slice/docker.service
             └─6678 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

i-019aa04bcd40d7c13 (Jenkins)

PublicIPs: 3.110.161.113 PrivateIPs: 172.31.35.56

Step 6:

Install Ansible :

```
root@ip-172-31-35-56:~# ansible --version
ansible [core 2.16.3]
  config file = /root/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, Sep 11 2024, 14:17:37) [GCC 13.2.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-35-56:~#
```

Step 7: Installing dependency plugins required for the project

Preparation	• Checking internet connectivity • Checking update center connectivity • Success
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
Pipeline Maven Plugin API	Success
Config File Provider	Success
Pipeline Maven Integration	Success
Maven Invoker	Success
Generic Webhook Trigger	Success
GitHub Integration	Success
Ansible	Success
Cloud Statistics	Success
Authentication Tokens API	Success
Docker Commons	Success
Apache HttpComponents Client 5.x API	Success
Docker API	Success
Docker	Success
Docker Pipeline	Success
Loading plugin extensions	Success
HTML Publisher	Success
Loading plugin extensions	Success

Step 8:

Install Terraform

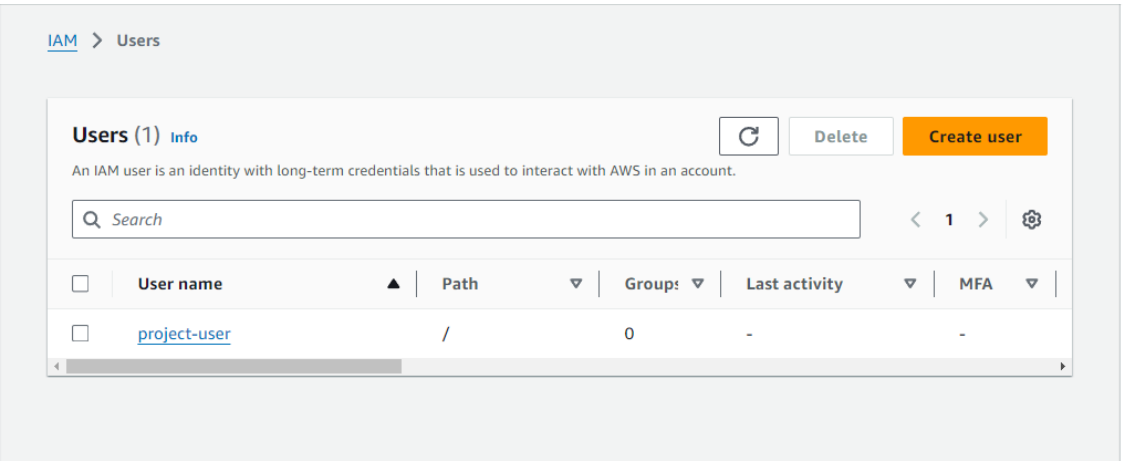
```
root@ip-172-31-35-56:~# terraform -version
Terraform v1.9.8
on linux_amd64
root@ip-172-31-35-56:~#
```

i-019aa04bcd40d7c13 (Jenkins)

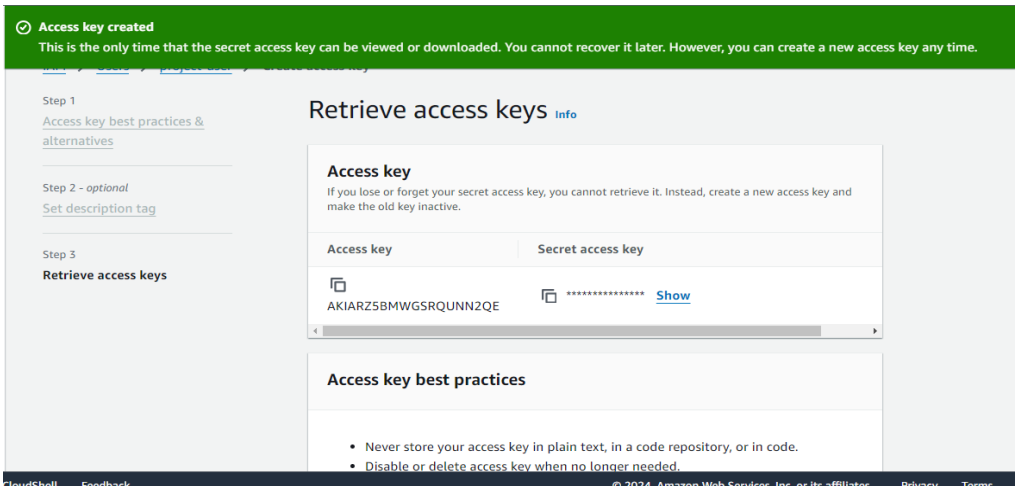
PublicIPs: 3.110.161.113 PrivateIPs: 172.31.35.56

- Once the image is pushed we need to deploy our application .we deploy in the new instances so we use Terraform to create these new instances.

- To use terraform we need to configure AWS CLI which need secret keys
- Create a user and give admin policies



- Create access keys



Step 9:

Go to tools and give the paths for the java,maven,ansible,terraform which we installed In the jenkins names instance.

JDK installations

Add JDK

JDK

Name

JAVA_HOME

JAVA_HOME

/usr/lib/jvm/java-17-openjdk-amd64

Install automatically ?

Add JDK

Maven installations

Add Maven

Maven

Name

$MAVEN_HOME

MAVEN_HOME

/opt/maven/

Install automatically ?

Add Maven

Ansible installations

Add Ansible

Ansible

Name

ansible

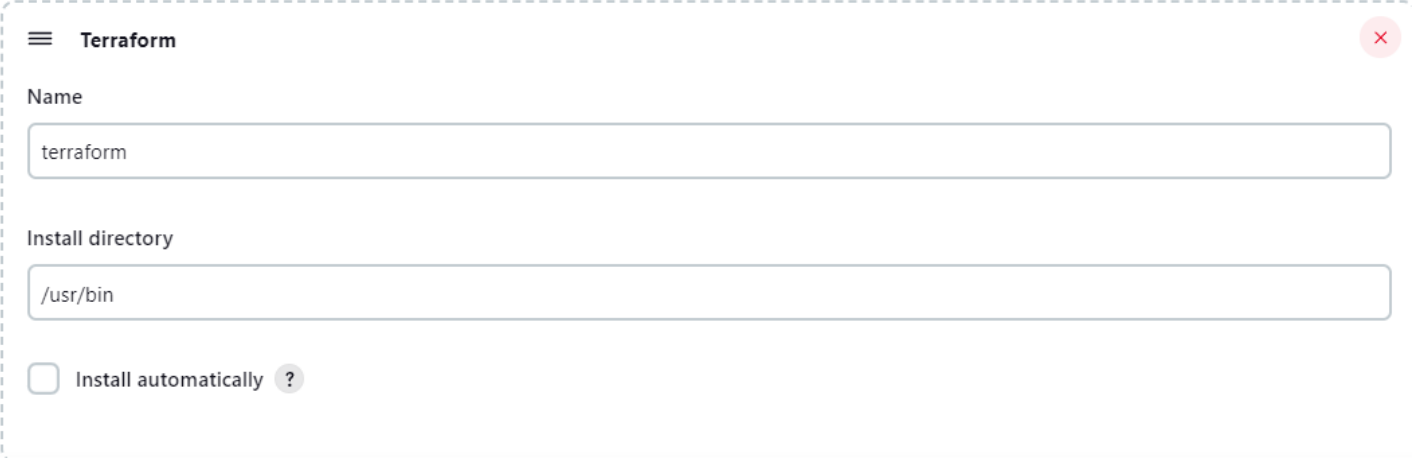
Path to ansible executables directory

/usr/bin/

Install automatically ?

Add Ansible

Docker installations

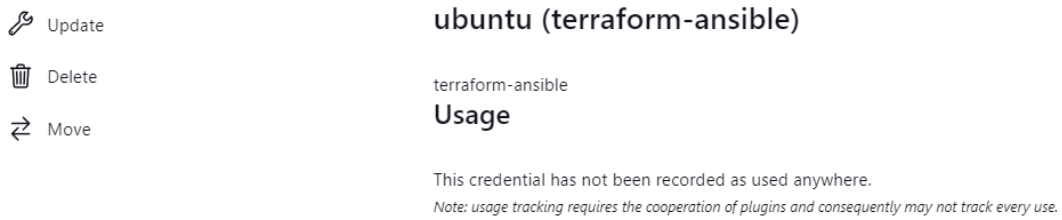


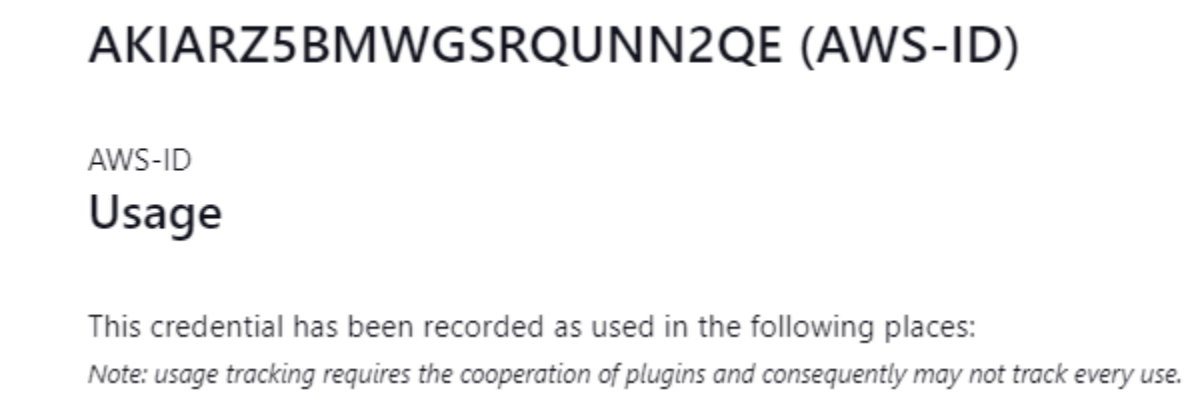
Step 10:

Go to credentials and give necessary credentials which we require to run our pipeline according to Jenkinsfile

- Docker credentials required to build image and push to hub

- To Push application to production using Terraform and Ansible we need terraform credentials and AWS ACCESS KEYS






```
stage('Config & Deployment') {
     steps {

         withCredentials([aws(accessKeyVariable: 'AWS_ACCESS_KEY_ID', credentialsId: 'AWS-ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY')]) {
             dir('terraform-files') {
             sh 'sudo chmod 600 docker.pem'
             sh 'terraform init'
             sh 'terraform validate'
             sh 'terraform apply --auto-approve'
```

Step 11:

Create the terraform main.tf file to create an . To Push application to production using Terraform .

star-agile-banking-finance / terraform-files / main.tf in master

Edit Preview Code 55% faster with GitHub Copilot

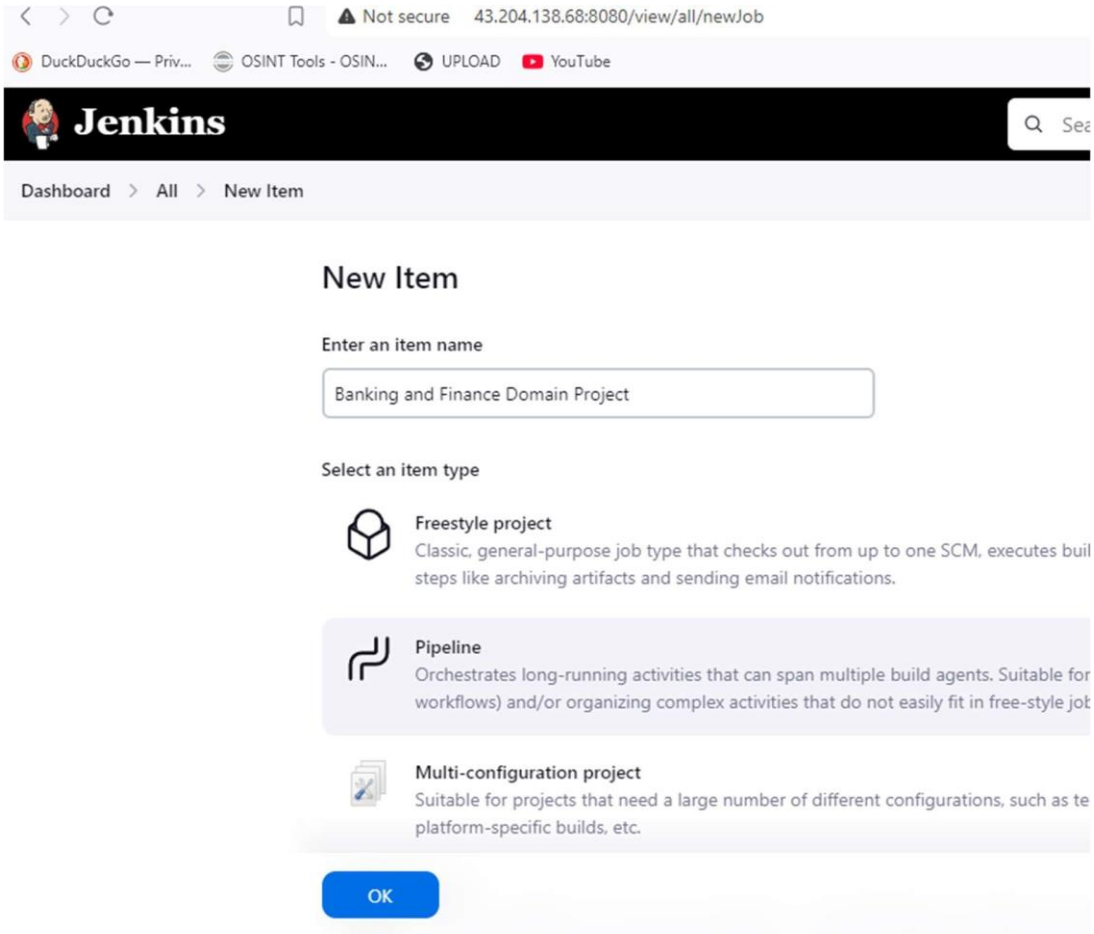
```
resource "aws_instance" "test-server" {
  ami = "ami-0dee22c13ea7a9a67"
  instance_type = "t2.micro"
  key_name = "docker"
  vpc_security_group_ids = ["sg-0fc8b1c023e7b4e38"]
   connection {
     type = "ssh"
     user = "ubuntu"
     private_key = file("./docker.pem")
     host = self.public_ip
     }
  provisioner "remote-exec" {
     inline = ["echo 'wait to start the instance' "]
  }
  tags = {
     Name = "test-server"
     }
  provisioner "local-exec" {
     command = "echo ${aws_instance.test-server.public_ip} > inventory"
     }
  provisioner "local-exec" {
     command = "ansible-playbook /var/lib/jenkins/workspace/Banking and Finance Domain Project
/terraform-files/ansibleplaybook.yml"
     }
  }
```

- Ansible playbook.yml file to install necessary package tools in the server into the production server.

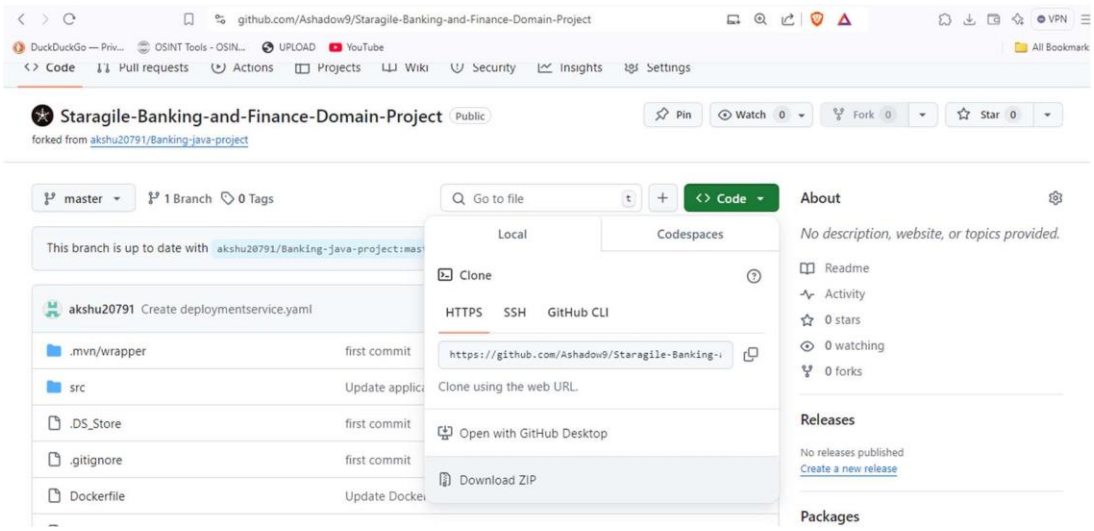
```
- name: Configure Docker on EC2 Instances
  hosts: all
  become: true
  connection: ssh
  tasks:
    - name: Updating apt
      command: sudo apt update

    - name: Install required packages
      command: sudo apt install -y apt-transport-https ca-certificates curl software-properties-common



    - name: Update apt again
      command: sudo apt update

    - name: Install Docker
      command: sudo apt install docker.io -y

    - name: Start Docker Service
      command: sudo systemctl start docker

    - name: Deploy Docker Container
      command: docker run -itd -p 8084:8081 skywalkerdarth/banking-project-demo:3.0
```

Step 12:

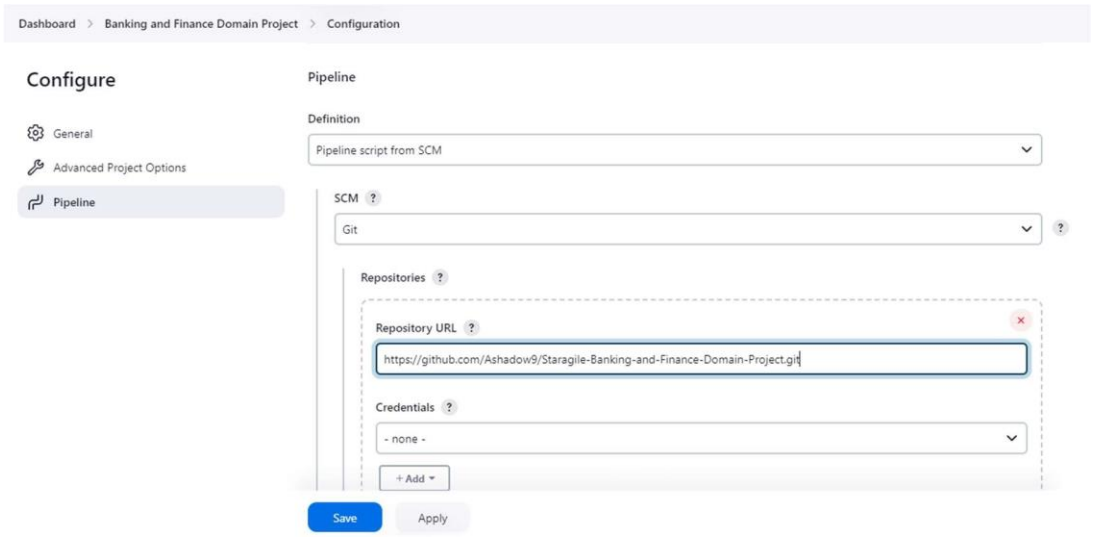
- Now,Select a new item from the Jenkins dashboard to create the job and slect the pipeline



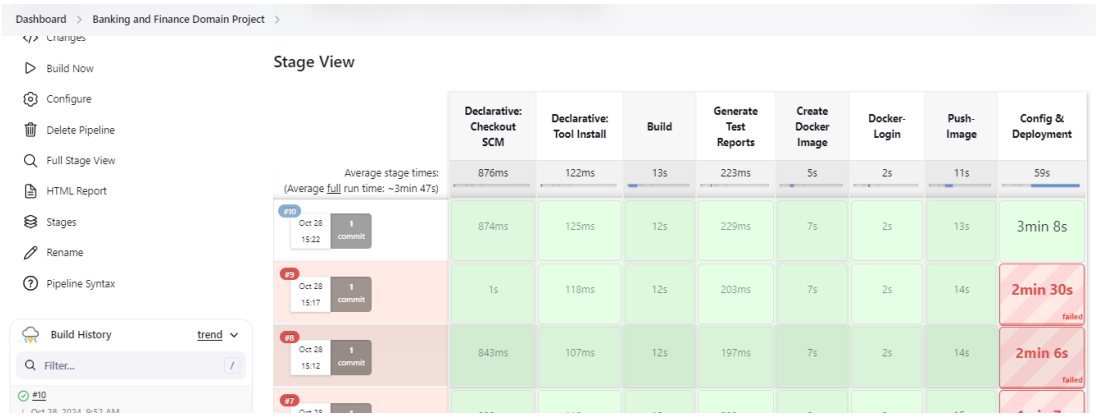
- Copy the GitHub repo in which we have the Jenkin file



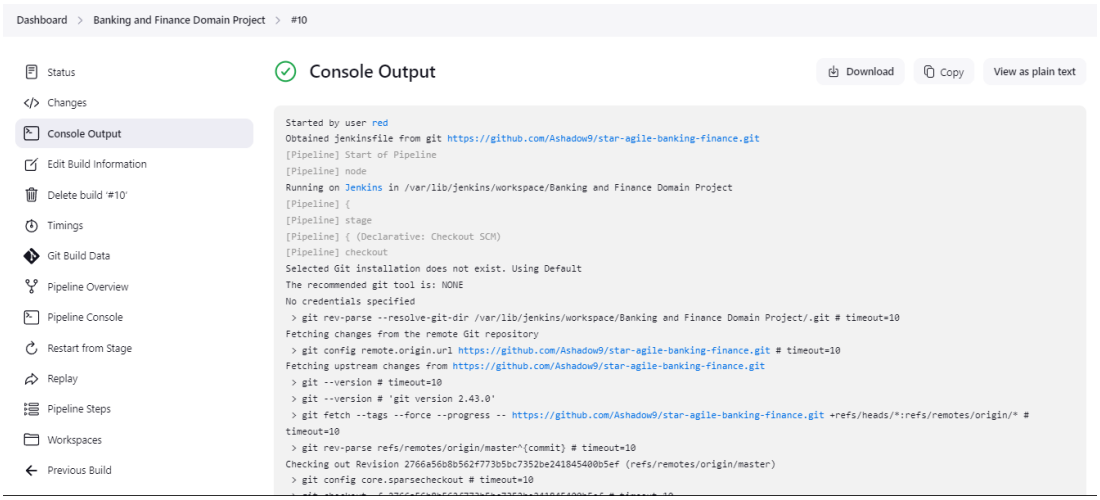
- paste it in scm in the pipeline configure option

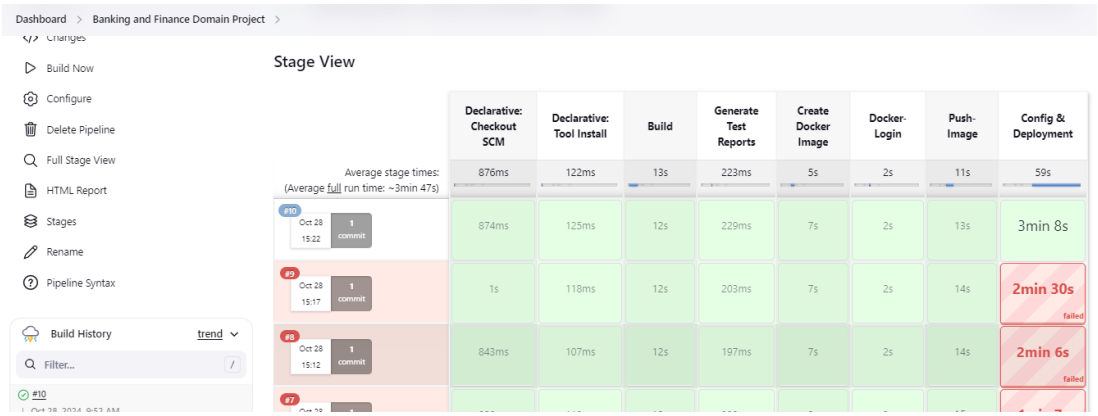


- Now,start the build and we can see the process:



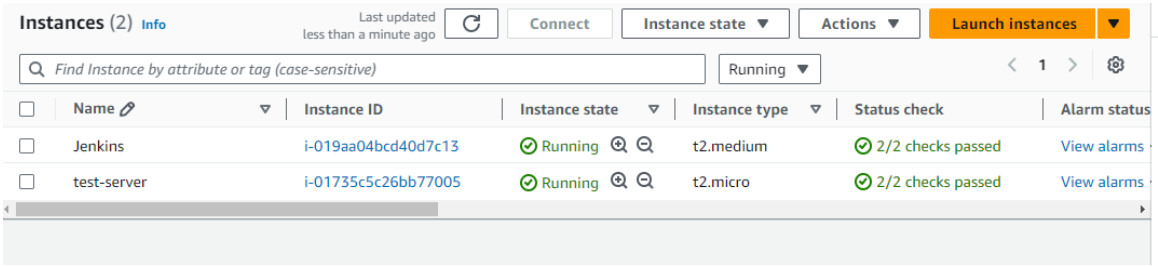
- The build is success :





Step 13:

- The instance is created

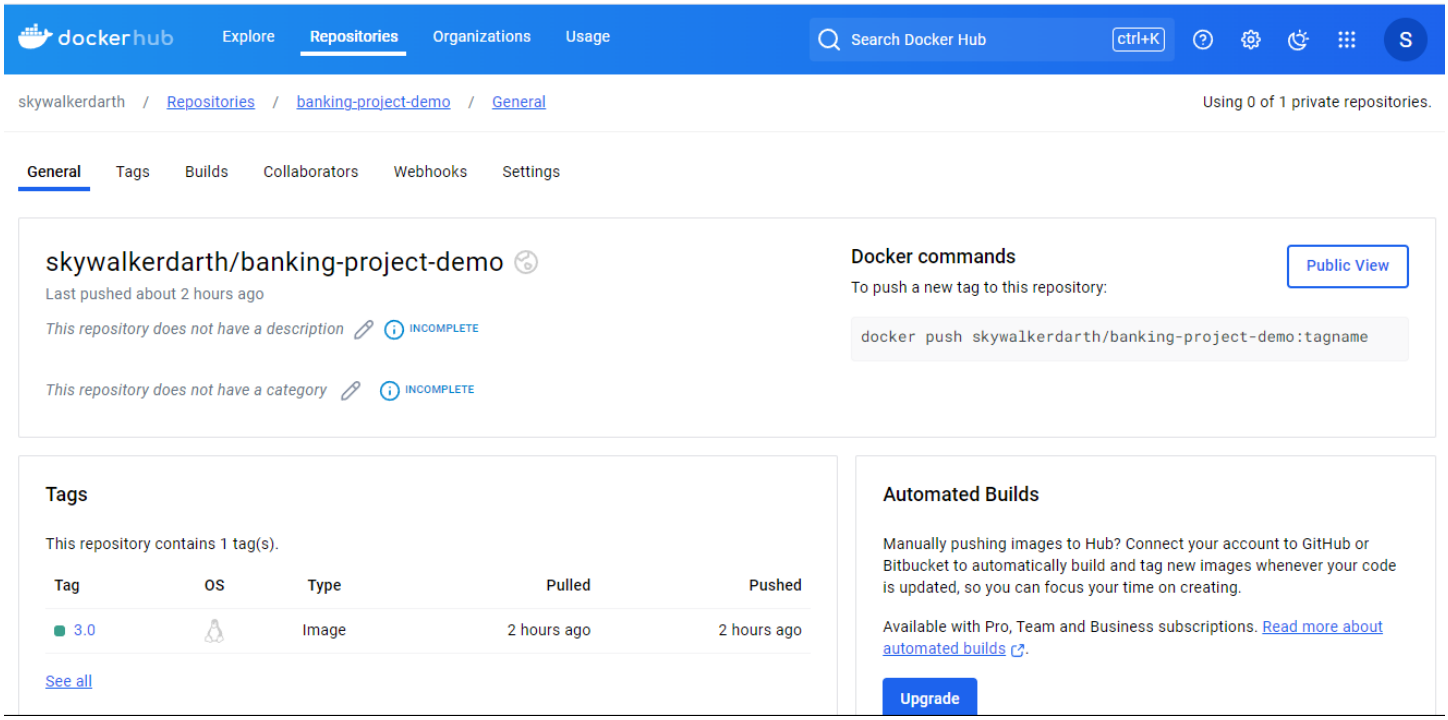


- We can see the docker images and docker conatiners build from the pipeline

```
stage('Create Docker Image') {
    steps {
        sh 'docker build -t skywalkerdarth/banking-project-demo:3.0 .'
            }
        }
stage('Docker-Login') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'Docker-login', passwordVariable: 'dockerpassword', usernameVariable: 'dockerlogin')]) {
        sh 'docker login -u ${dockerlogin} -p ${dockerpassword}'
                        }
                }
        }
stage('Push-Image') {
    steps {
        sh 'docker push skywalkerdarth/banking-project-demo:3.0'
            }
        }
```

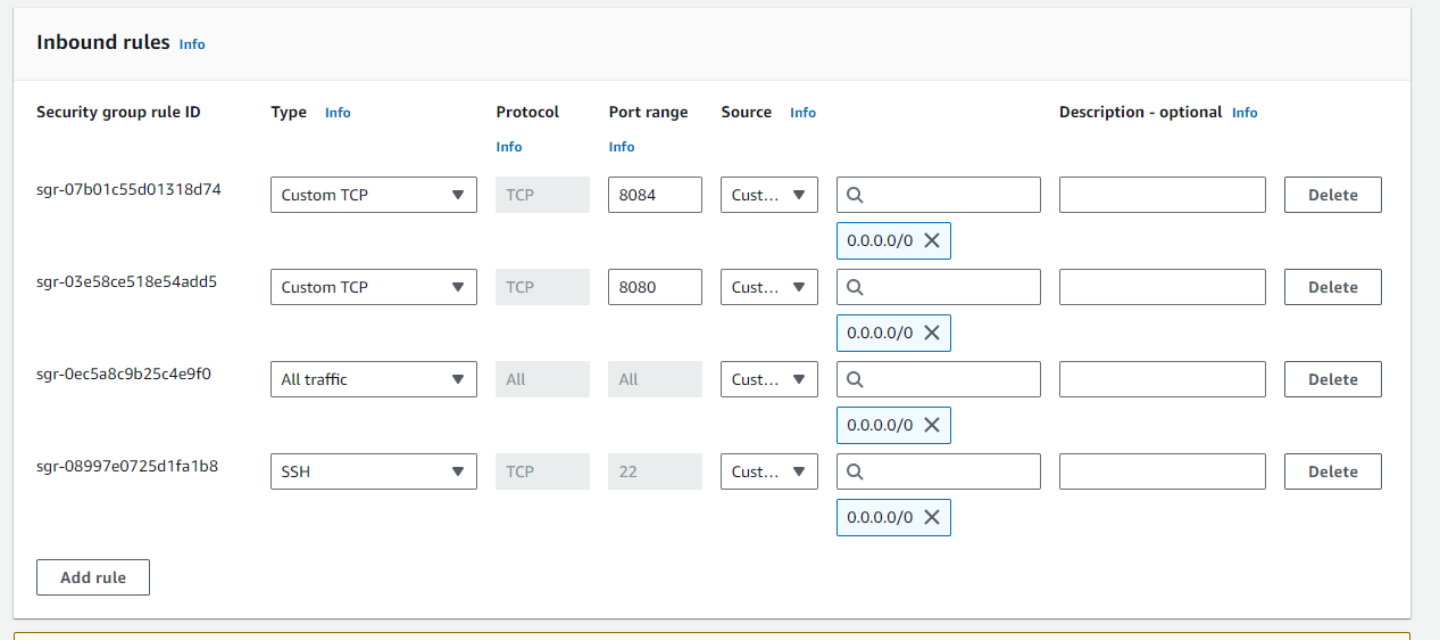
```
root@ip-172-31-26-213:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS         PORTS                                       NAMES
a95601fba189   myimg     "java -jar /app.jar"   8 minutes ago   Up 8 minutes   0.0.0.0:8091->8091/tcp, :::8091->8091/tcp   c000
root@ip-172-31-26-213:/home/ubuntu# docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
myimg        latest    8cb3f8b3a13c   8 minutes ago   696MB
openjdk      11        47a932d998b7   2 years ago     654MB
root@ip-172-31-26-213:/home/ubuntu#
```

- The image is pushed into the docker hub



Step 14:

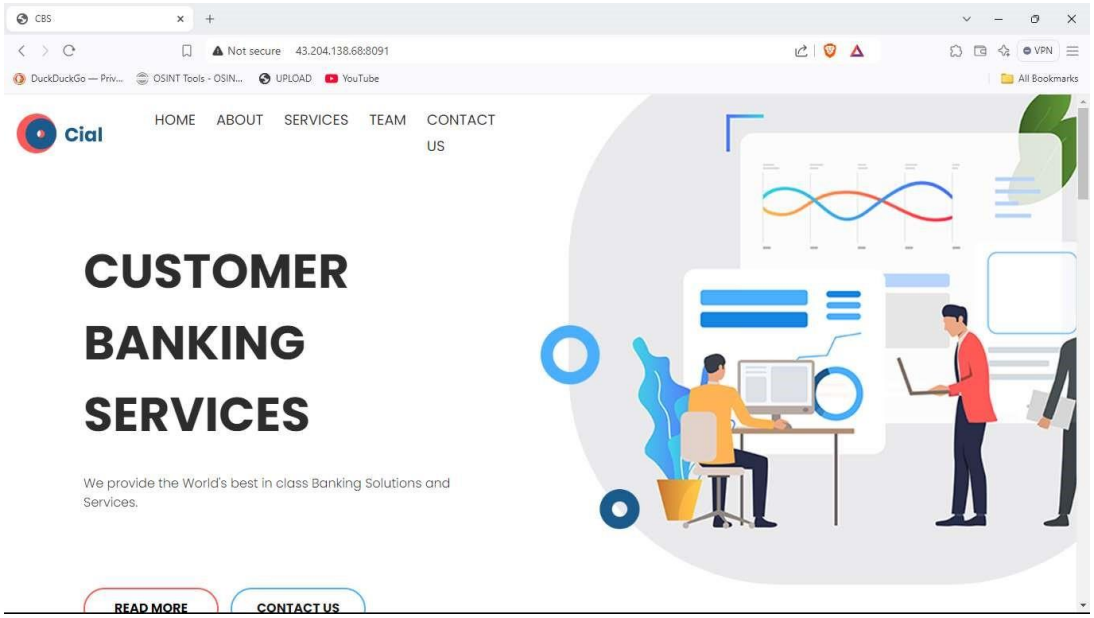
Allow the port 8084 in the security group of the Instance

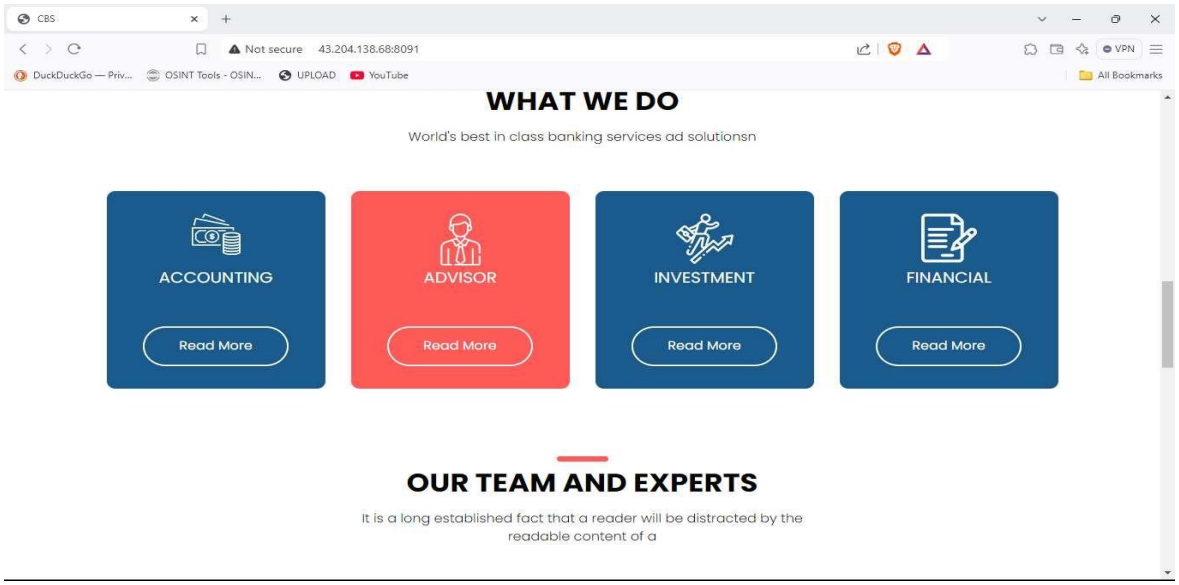


Step 15:

Now run the public IP address of the server with the port mentioned in the Jenkins file .

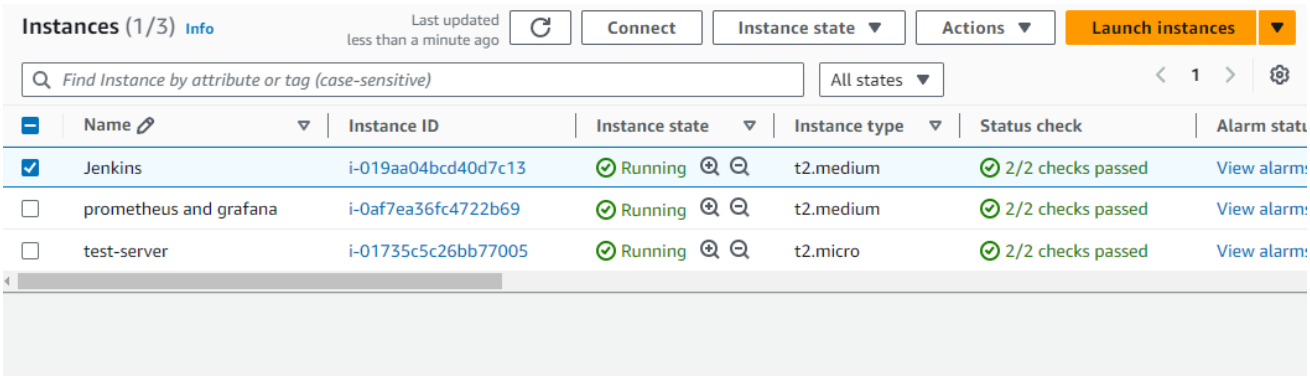
We can see the website is working on the mentioned port





Step 16 :

- Create an instance and install Prometheus and Grafana in that.



- **Installation steps:**

Wget https://github.com/prometheus/prometheus/releases/download/v2.43.0/prom etheus-2.43.0.linux-amd64.tar.gz

tar -xvf prometheus-2.43.0.linux-amd64.tar.gz

sudo mv prometheus-2.43.0.linux-amd64 /usr/local/Prometheus

cd /usr/local/Prometheus

- vi prometheus.yml

- job_name: 'node_exporter'

static_configs:

- targets: [publicip:9100']

```
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: 'node_exporter'

    static_configs:
      - targets: ['35.154.59.115:9100']
"prometheus.yml" [readonly] 33L, 1029B
```

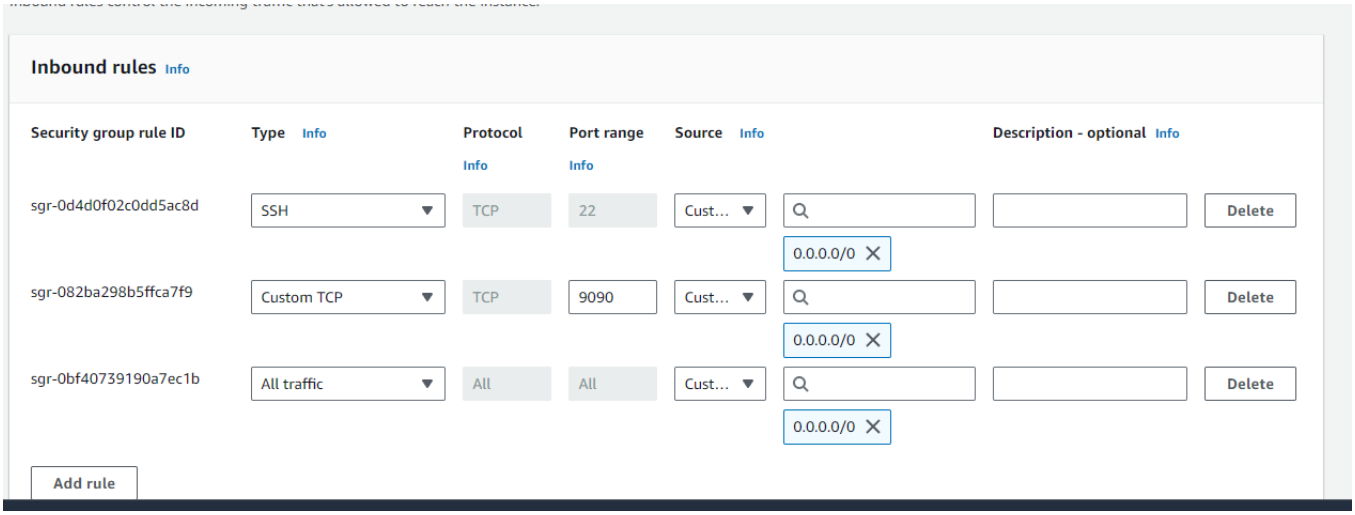
- Start Prometheus by running:

./prometheus --config.file=prometheus.yml

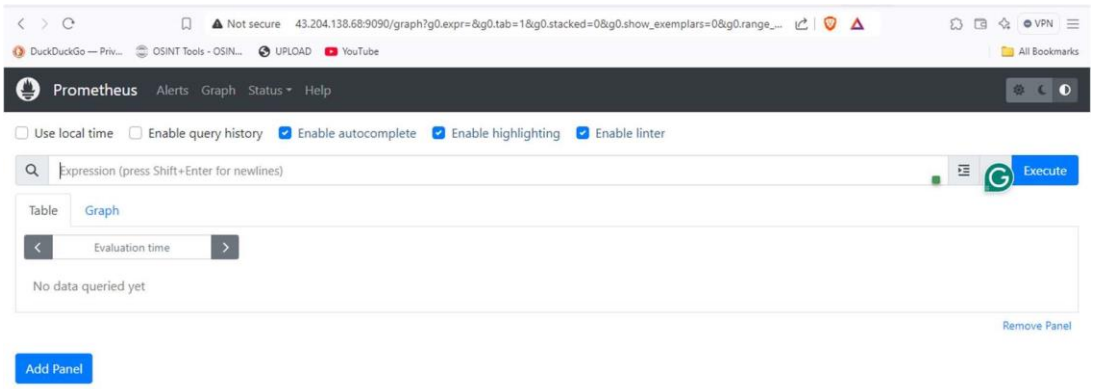
```
root@ip-172-31-26-213:/usr/local/prometheus# vi prometheus.yml
root@ip-172-31-26-213:/usr/local/prometheus# ./prometheus --config.file=prometheus.yml
ts=2024-10-05T06:41:03.428Z caller=main.go:520 level=info msg="No time or size retention was set so using th
d
ts=2024-10-05T06:41:03.428Z caller=main.go:564 level=info msg="Starting Prometheus Server" mode=server versi
ision=edfc3bcd025dd6fe296c167a14a216cab1e552ee)"
ts=2024-10-05T06:41:03.428Z caller=main.go:569 level=info build_context="(go=go1.19.7, platform=linux/amd64,
1-12:56:07, tags=netgo,builtinassets)"
ts=2024-10-05T06:41:03.428Z caller=main.go:570 level=info host_details="(Linux 6.8.0-1016-aws #17~Ubuntu SMP
ip-172-31-26-213 (none))"
ts=2024-10-05T06:41:03.429Z caller=main.go:571 level=info fd_limits="(soft=1048576, hard=1048576)"
ts=2024-10-05T06:41:03.429Z caller=main.go:572 level=info vm_limits="(soft=unlimited, hard=unlimited)"
ts=2024-10-05T06:41:03.435Z caller=web.go:561 level=info component=web msg="Start listening for connections"
ts=2024-10-05T06:41:03.436Z caller=main.go:1005 level=info msg="Starting TSDB ..."
ts=2024-10-05T06:41:03.444Z caller=tls_config.go:232 level=info component=web msg="Listening on" address=[::
ts=2024-10-05T06:41:03.444Z caller=tls_config.go:235 level=info component=web msg="TLS is disabled." http2=f
ts=2024-10-05T06:41:03.451Z caller=head.go:587 level=info component=tsdb msg="Replaying on-disk memory mappa
ts=2024-10-05T06:41:03.451Z caller=head.go:658 level=info component=tsdb msg="On-disk memory mappable chunks
ts=2024-10-05T06:41:03.451Z caller=head.go:664 level=info component=tsdb msg="Replaying WAL, this may take a
ts=2024-10-05T06:41:03.466Z caller=head.go:735 level=info component=tsdb msg="WAL segment loaded" segment=0
ts=2024-10-05T06:41:03.467Z caller=head.go:735 level=info component=tsdb msg="WAL segment loaded" segment=1
ts=2024-10-05T06:41:03.467Z caller=head.go:772 level=info component=tsdb msg="WAL replay completed" checkpoi
ay_duration=15.42779ms wbl_replay_duration=154ns total_replay_duration=15.601857ms
ts=2024-10-05T06:41:03.468Z caller=main.go:1026 level=info fs_type=EXT4_SUPER_MAGIC
```

Prometheus will now be accessible via public ip :9090.

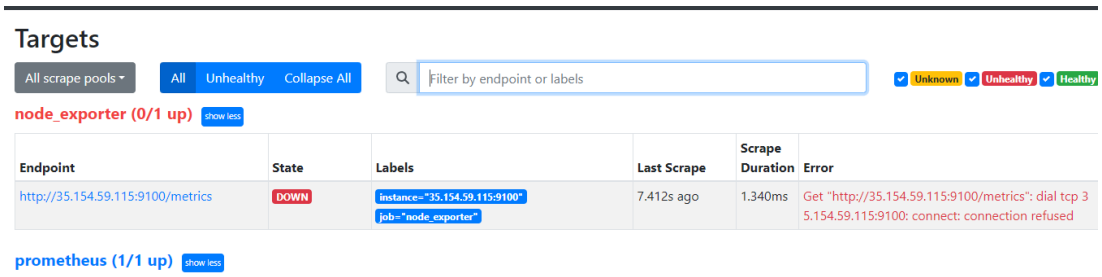
- Allow port 9090 in the security group of the Prometheus server



- Paste the public ip with port 9090 to access prometheus



- Node Exporter is down so we need to install node exporter



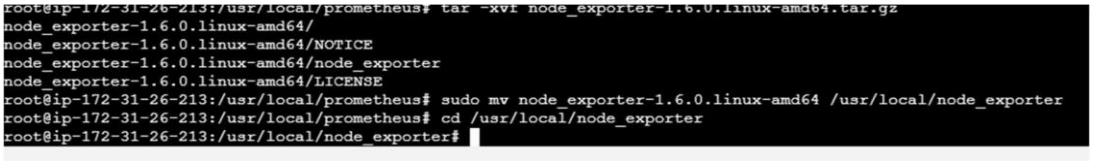
- Install Node Exporter (For Server Metrics)

wgeth ps://github.com/prometheus/node_exporter/releases/download/v1.6.0/node _exporter-1.6.0.linux-amd64.tar.gz

tar -xvf node_exporter-1.6.0.linux-amd64.tar.gz sudo mv node_exporter-1.6.0.linux-amd64

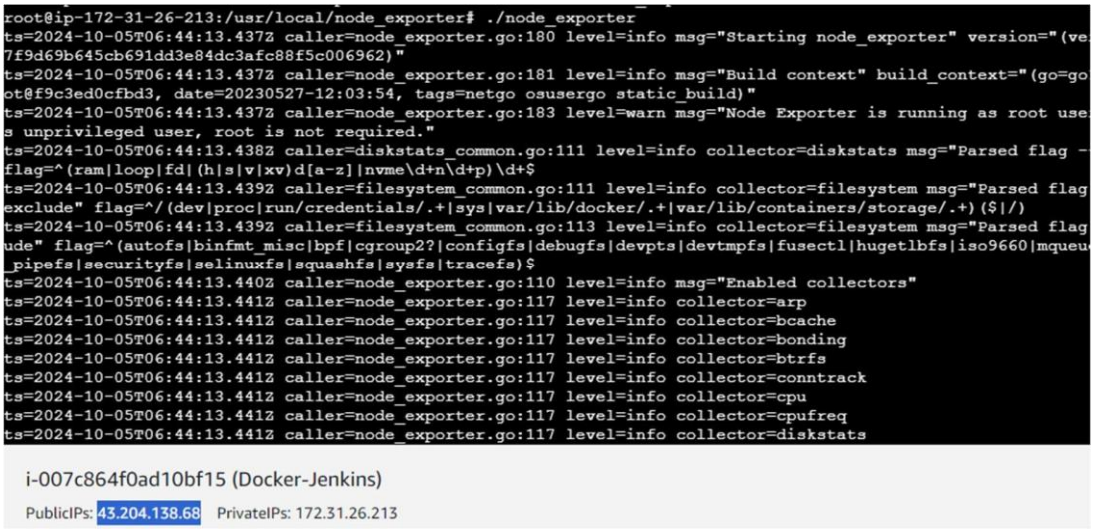
/usr/local/node_exporter

cd /usr/local/node_exporter

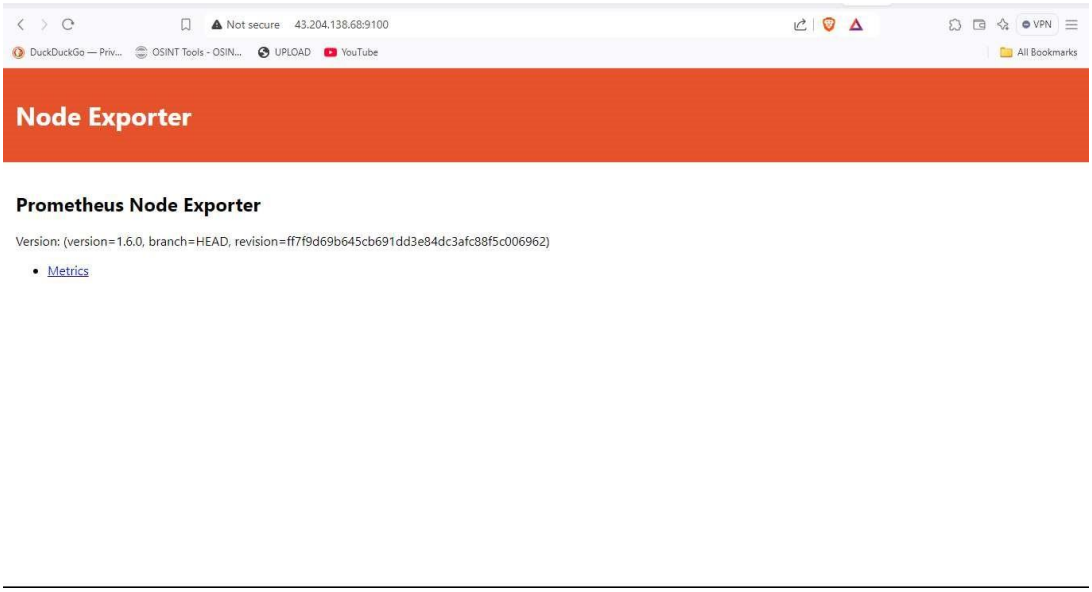


Start Node Exporter

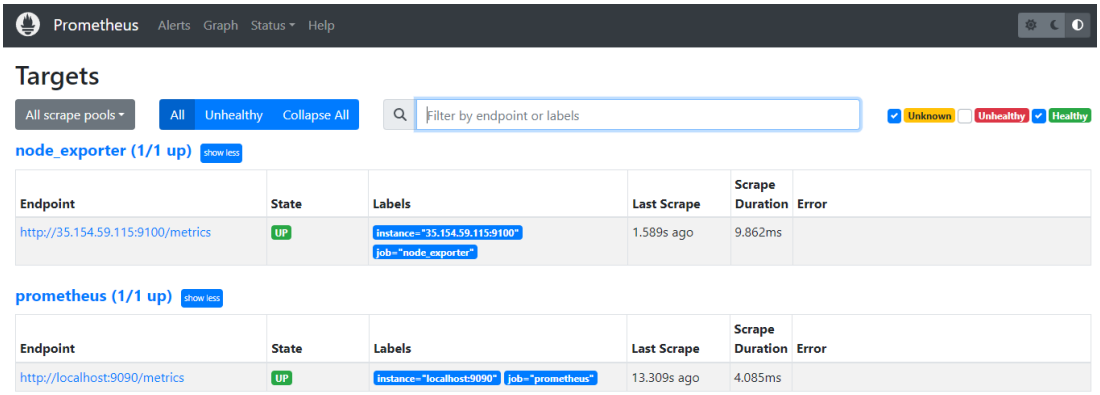
./node_exporter



http://43.204.138.68:9100/



- Node exporter is up



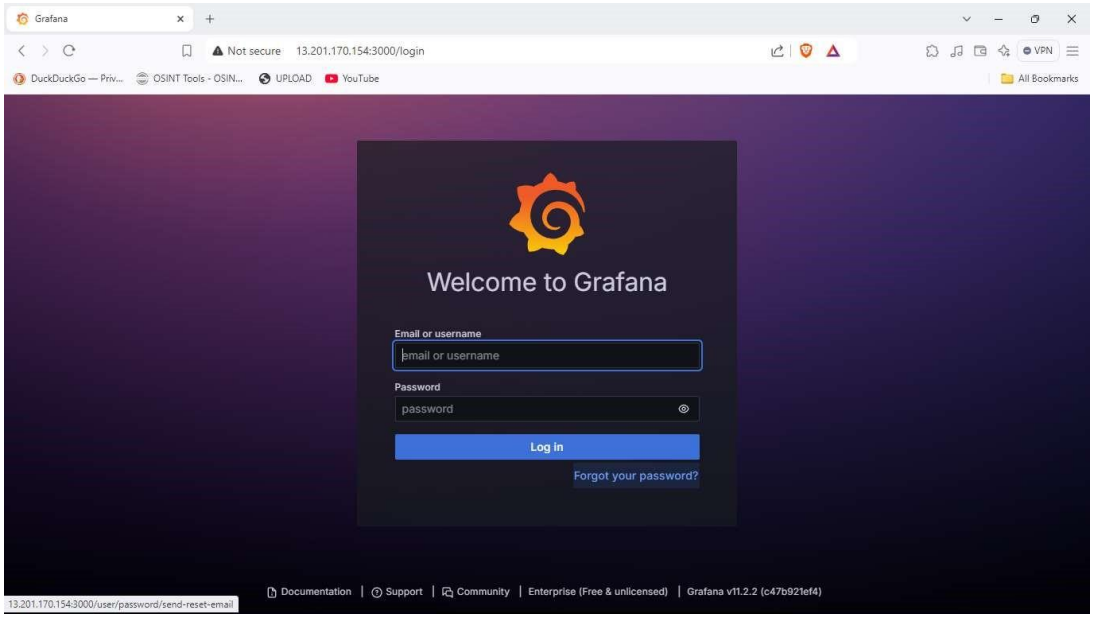
Step 17:Install Grafana :-

sudo apt-get update
sudo apt-get install -y adduser libfontconfig1 musl
wget https://dl.grafana.com/enterprise/release/grafanaenterprise_11.2.2_amd64.deb
sudo dpkg -i grafana-enterprise_11.2.2_amd64.deb
Log in using the default credentials:

- Username: admin
- Password: admin

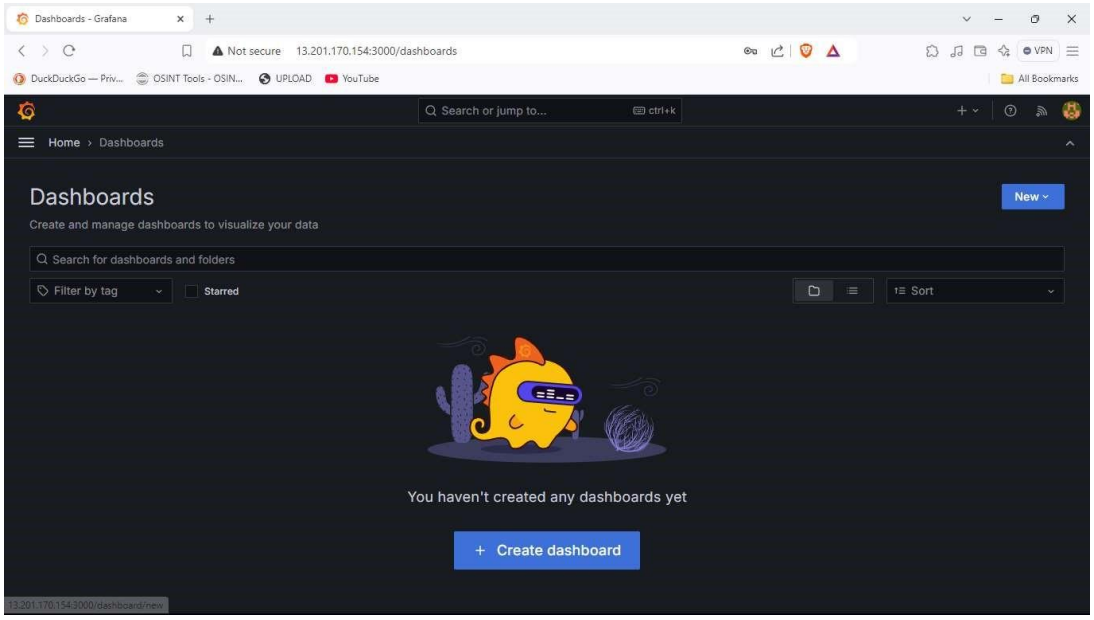
```
ubuntu@ip-172-31-8-38:~$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-10-28 11:04:30 UTC; 18s ago
       Docs: http://docs.grafana.org
   Main PID: 4736 (grafana)
      Tasks: 14 (limit: 4676)
     Memory: 76.3M (peak: 76.7M)
        CPU: 2.644s
     CGroup: /system.slice/grafana-server.service
             └─4736 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pid

Oct 28 11:04:35 ip-172-31-8-38 grafana[4736]: logger=plugins.update.checker t=2024-10-28T11:04:35
Oct 28 11:04:36 ip-172-31-8-38 grafana[4736]: logger=grafana.update.checker t=2024-10-28T11:04:36
Oct 28 11:04:36 ip-172-31-8-38 grafana[4736]: logger=plugin.angulardetectorsprovider.dynamic t=20
Oct 28 11:04:36 ip-172-31-8-38 grafana[4736]: logger=grafana-apiserver t=2024-10-28T11:04:36.0531
Oct 28 11:04:36 ip-172-31-8-38 grafana[4736]: logger=grafana-apiserver t=2024-10-28T11:04:36.0543
Oct 28 11:04:36 ip-172-31-8-38 grafana[4736]: logger=grafana-apiserver t=2024-10-28T11:04:36.0557
Oct 28 11:04:37 ip-172-31-8-38 grafana[4736]: logger=plugin.installer t=2024-10-28T11:04:37.24782
Oct 28 11:04:37 ip-172-31-8-38 grafana[4736]: logger=installer.fs t=2024-10-28T11:04:37.358806364
Oct 28 11:04:37 ip-172-31-8-38 grafana[4736]: logger=plugins.registration t=2024-10-28T11:04:37.4
Oct 28 11:04:37 ip-172-31-8-38 grafana[4736]: logger=plugin.backgroundinstaller t=2024-10-28T11:0
lines 1-21/21 (END)
```

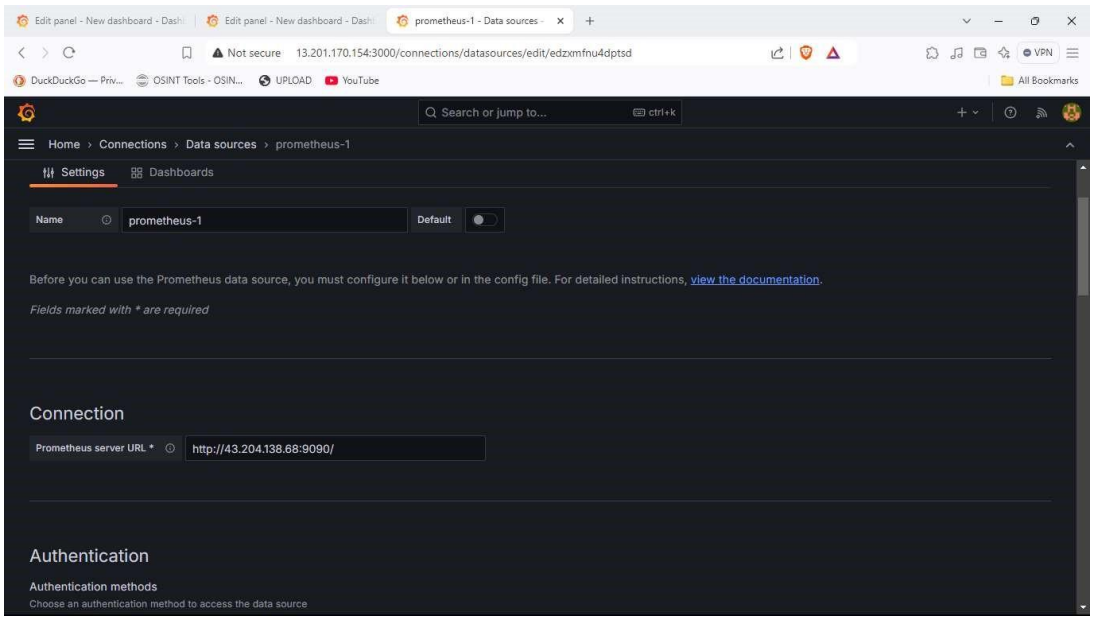
- **Grafana will be accessible via publicip: 3000.**

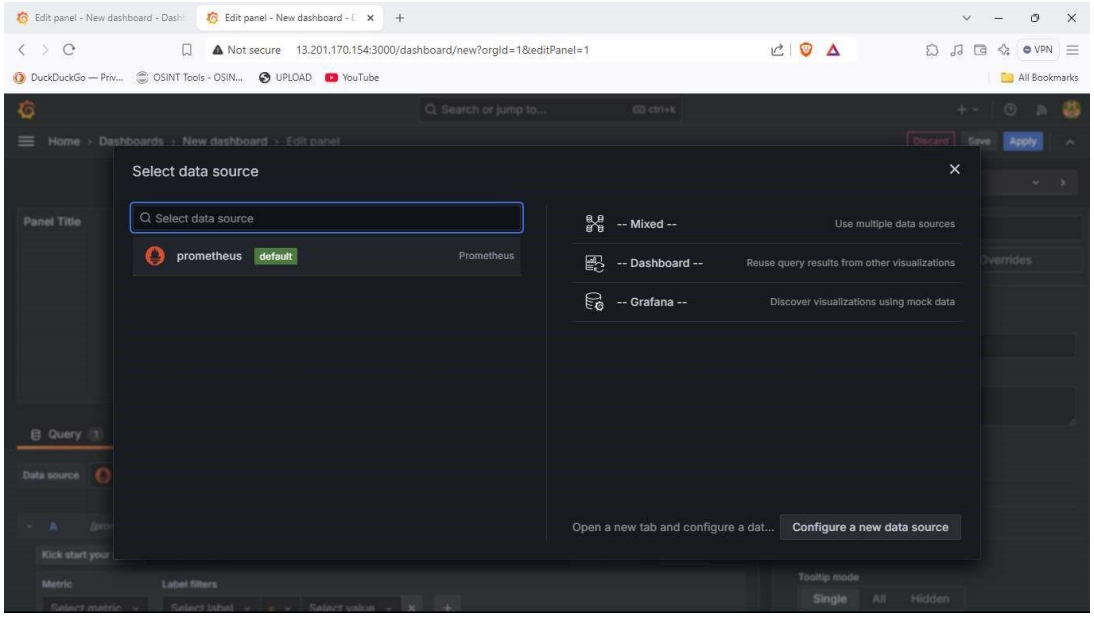


Dashboard of Grafana:

- navigate to **Configuration > Data Sources**.
- Click on **Add Data Source**, select **Prometheus**.
- Enter the URL of Prometheus: http://<your-server-ip>:9090.
- Click **Save & Test** to verify the connection.

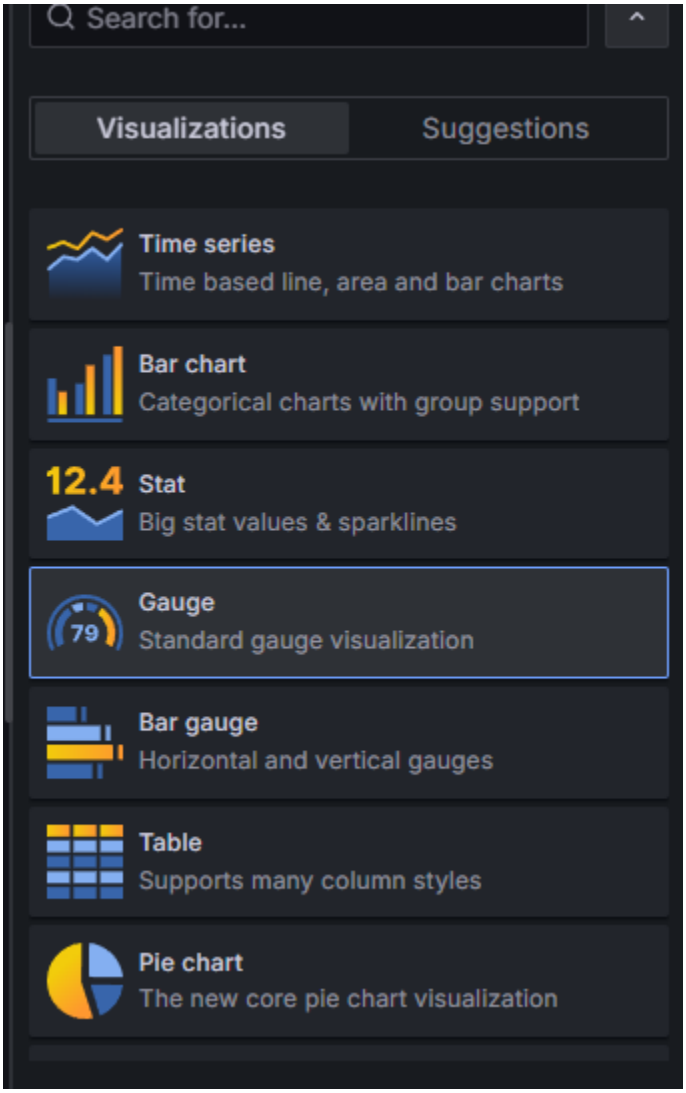

Dashboards - Grafana
Not secure 13.201.170.154:3000/dashboards
Search or jump to... ctrl+k
Home › Dashboards
Dashboards
Create and manage dashboards to visualize your data
New
Search for dashboards and folders
Filter by tag
Starred
Sort
You haven't created any dashboards yet
+ Create dashboard


prometheus-1 - Data sources
Not secure 13.201.170.154:3000/connections/datasources/edit/edzxmfnu4dptsd
Search or jump to... ctrl+k
Home › Connections › Data sources › prometheus-1
Settings
Dashboards
Name
prometheus-1
Default
Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.
Fields marked with * are required
Connection
Prometheus server URL *
http://43.204.138.68:9090/
Authentication
Authentication methods
Choose an authentication method to access the data source


Edit panel - New dashboard
Not secure 13.201.170.154:3000/dashboard/new?orgId=1&editPanel=1
Search or jump to... ctrl+k
Home › Dashboards › New dashboard › Edit panel
Discard
Save
Apply
Select data source
Select data source
prometheus
default
Prometheus
-- Mixed --
Use multiple data sources
-- Dashboard --
Reuse query results from other visualizations
-- Grafana --
Discover visualizations using mock data
Open a new tab and configure a dat...
Configure a new data source
Panel Title
Query
Data source
Metric
Label filters
Tooltip mode
Single
All
Hidden

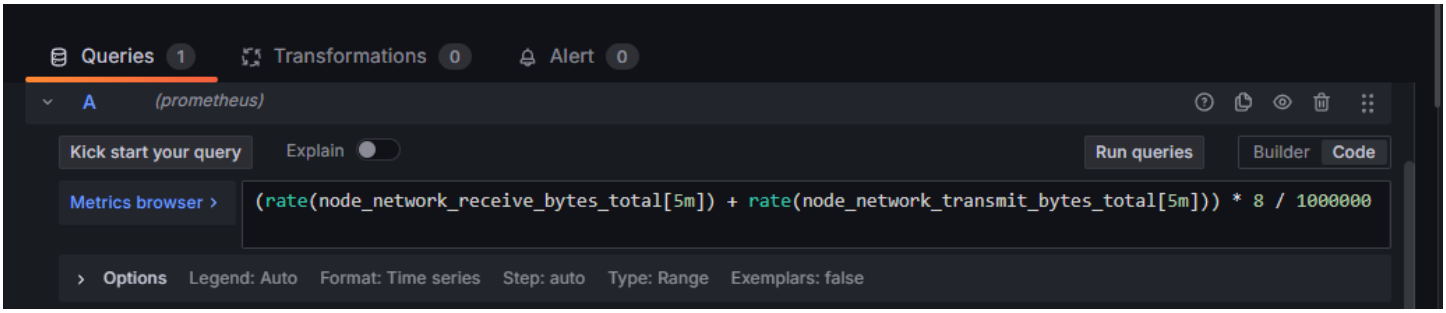
Step 18 : Metric Visualization:

- Select gauge visualization to see the metric



- Select the Misc and percent (0-100) from standard options in unit

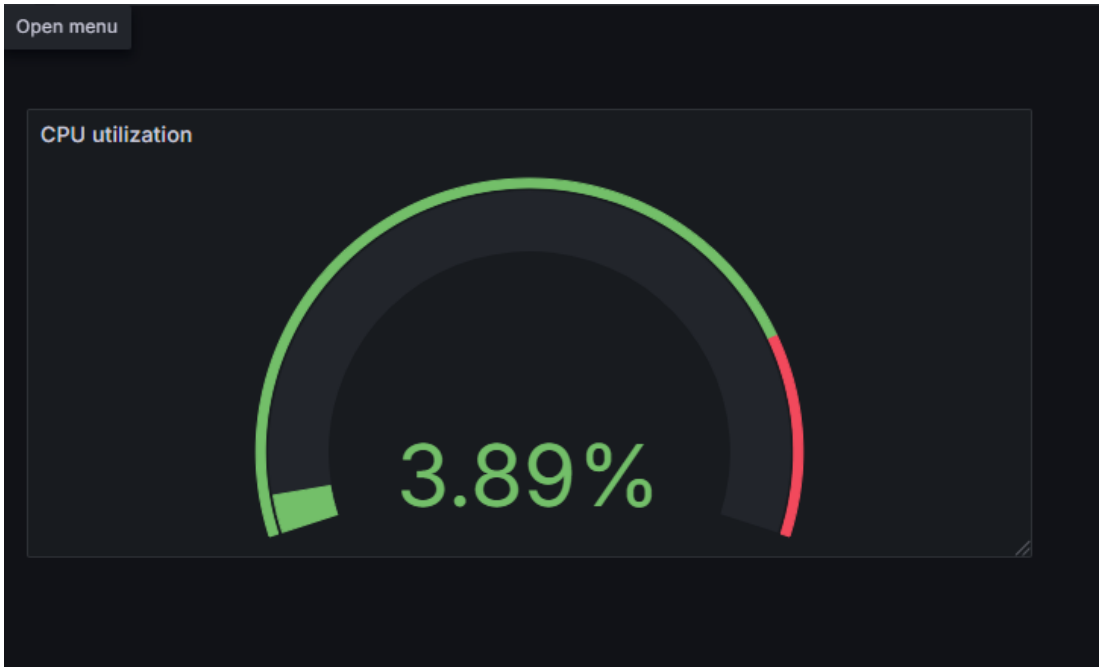
- Select the metric browser option and run queries and we can see the output visualization



Step 19:

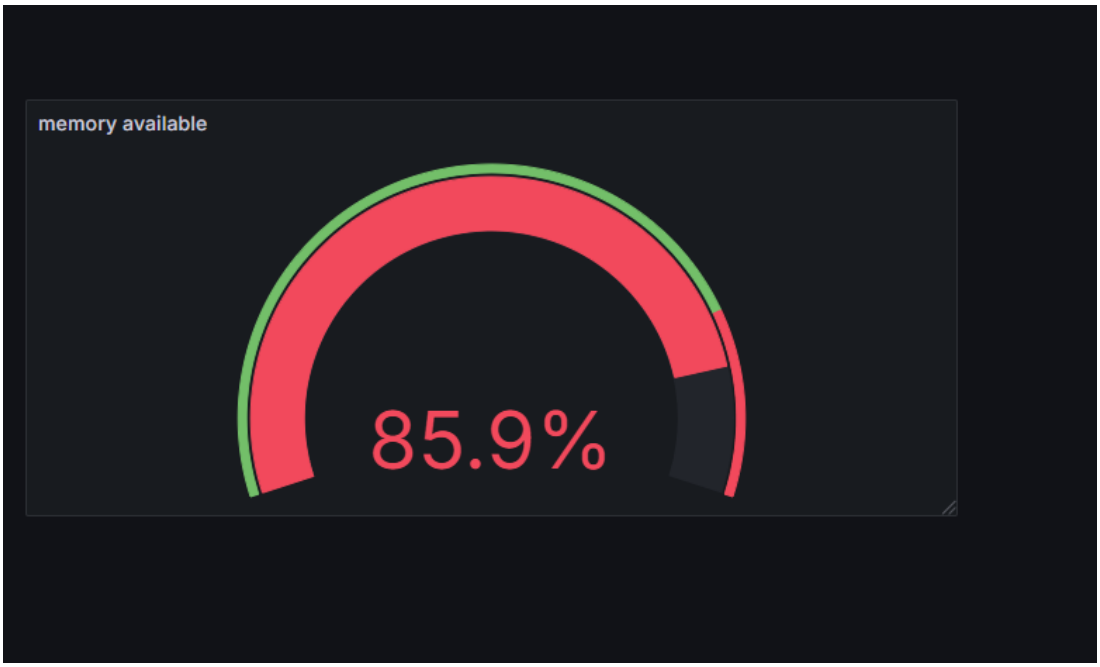
1. **CPU utilization**: To see CPU utilization of the server use gauge and select misc and run queries and paste this

100* (1-avg by(instance) (rate(node_cpu_seconds_total{mode-"idle"}[5m])))



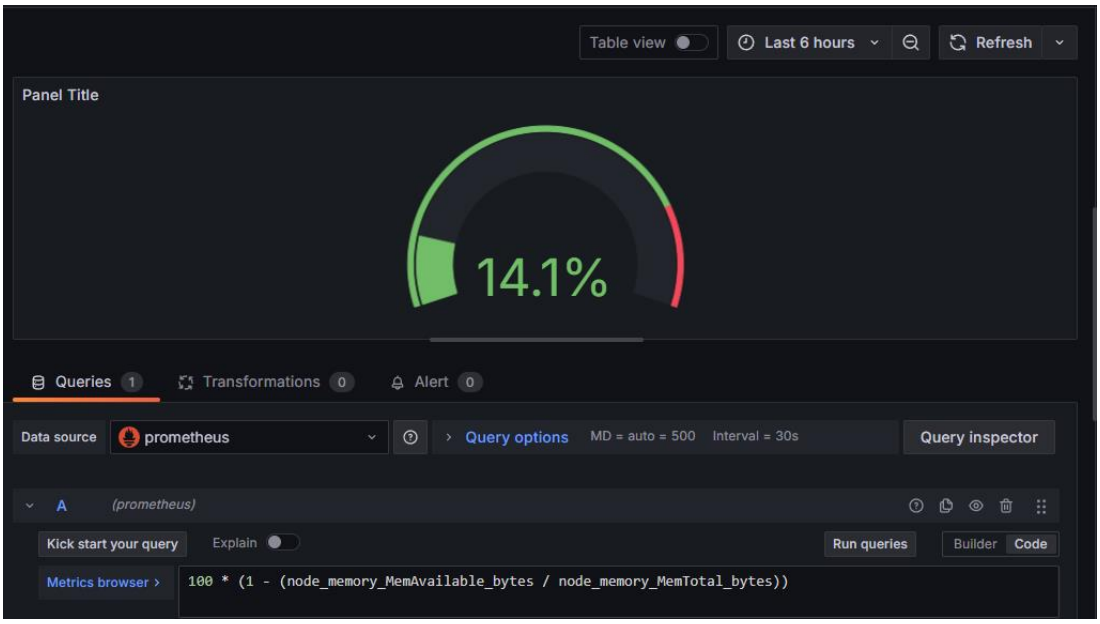
2. **Total Available Memory**:- To see available memory of the server

100 * (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes)



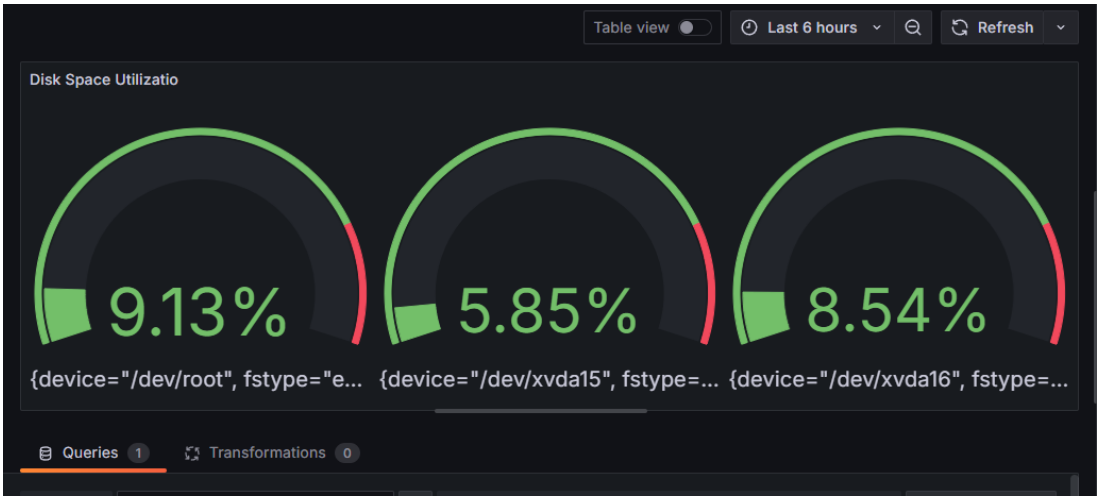
- To see the total memory

100 * (1 - (node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes))



3. Disk Space Utilization

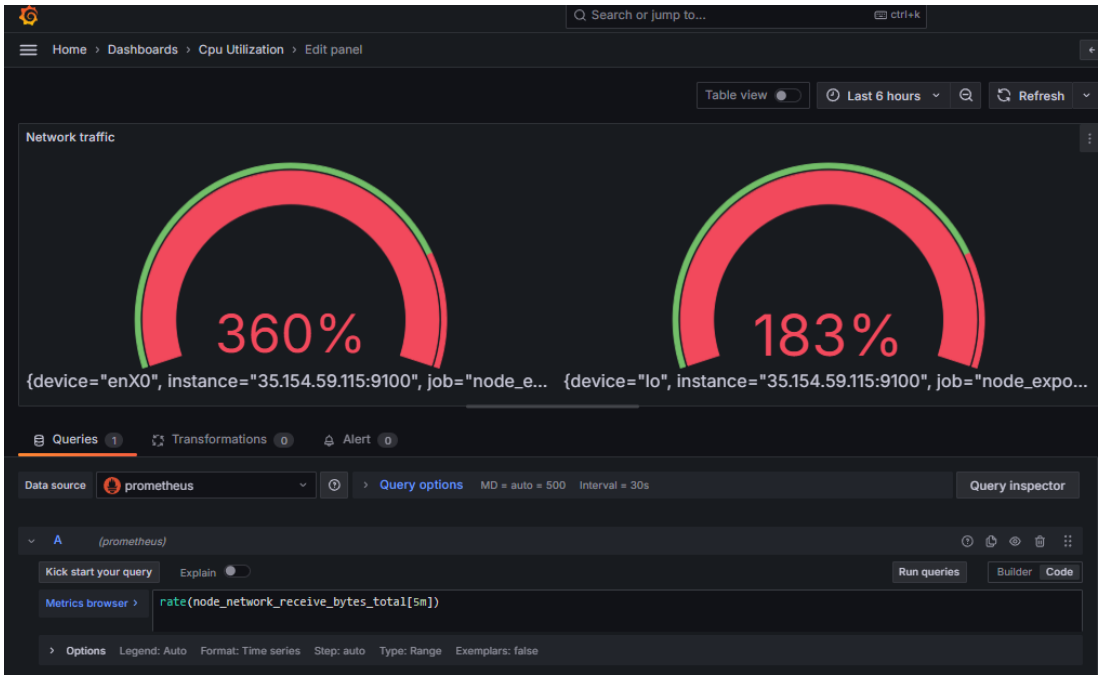
node_filesystem_size_bytes (Total Disk Size)



4.Network traffic

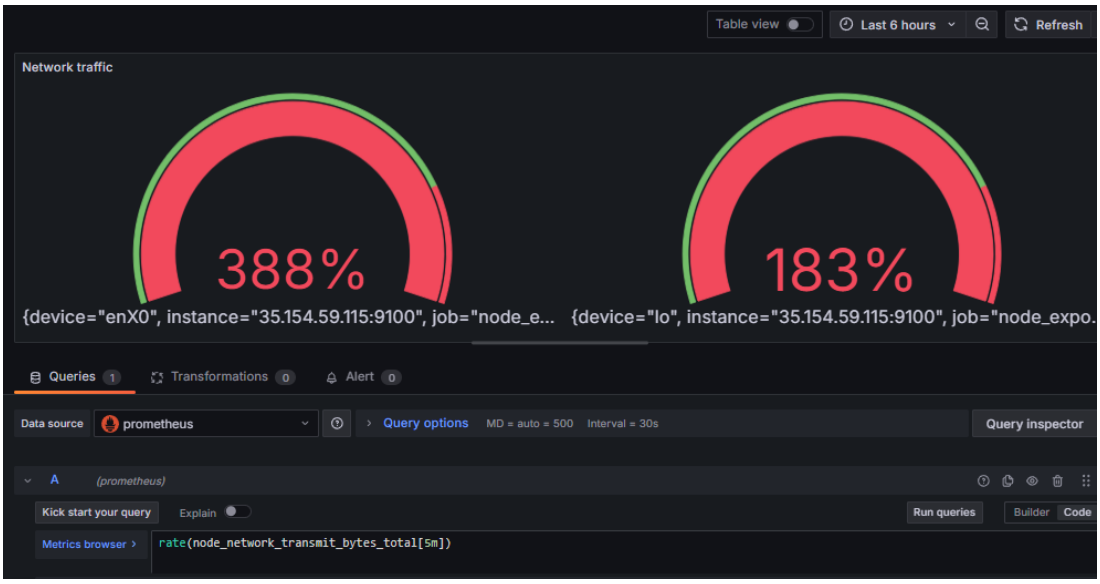
Incoming traffic:

rate(node_network_receive_bytes_total[5m])



Outgoing traffic

rate(node_network_transmit_bytes_total[5m])



Step 20:Make sure to terminate all the instances