BANKING AND FINANCE PROJECT

Submission by : Pallavi
Submitted to : Vikul
Date of submission: 06-12-24

Banking and Finance Project

The project involves creating a fully automated pipeline with Terraform, Jenkins, Docker, Ansible, Prometheus, and Grafana for monitoring.

- ✓ Git - For version control for tracking changes in the code files
- ✓ Maven – For Continuous Build
- ✓ Jenkins - For continuous integration and continuous deployment
- ✓ Docker - For deploying containerized applications
- ✓ Ansible - Configuration management tools
- ✓ Terraform - For creation of infrastructure.
- ✓ Prometheus and Grafana – For Automated Monitoring and Report Visualization

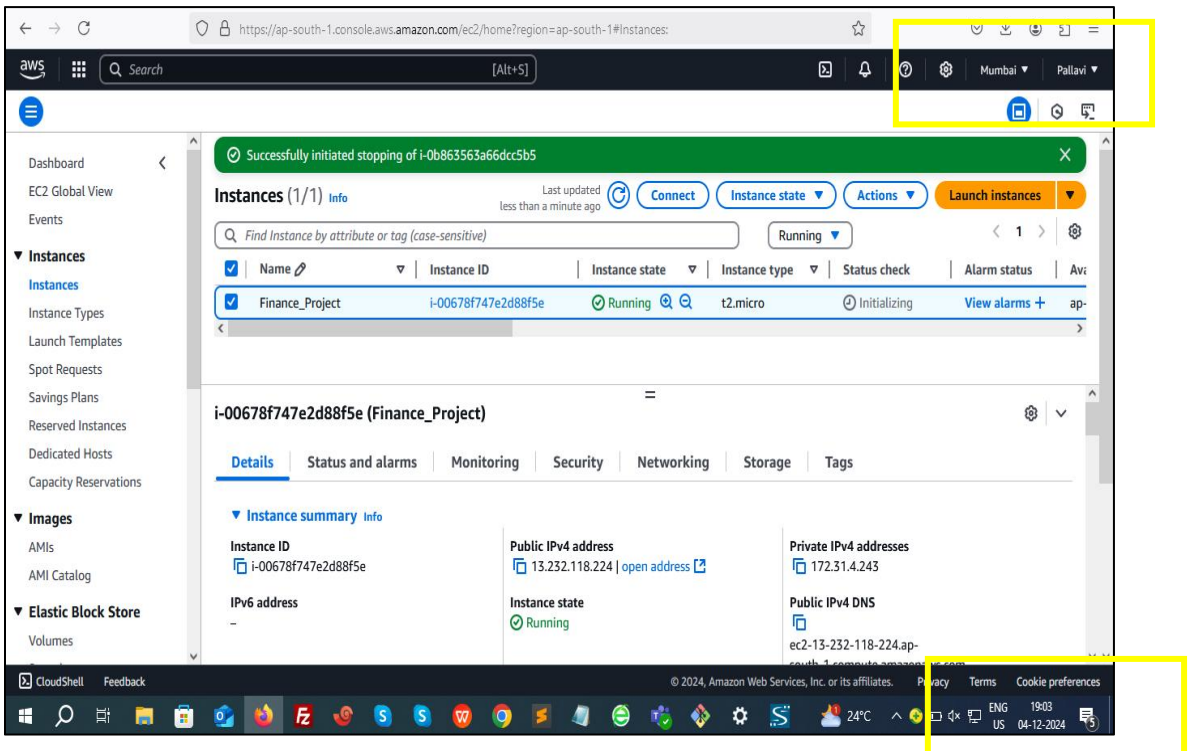
FinanceMe project code is attached below

https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance.git

Step 1: Create a VM and Install Terraform

1.Launch a VM (Ubuntu 22.04) in AWS:

1. Instance Type: t2.micro
2. Security Group: Allow **All traffic (Anywhere)**.

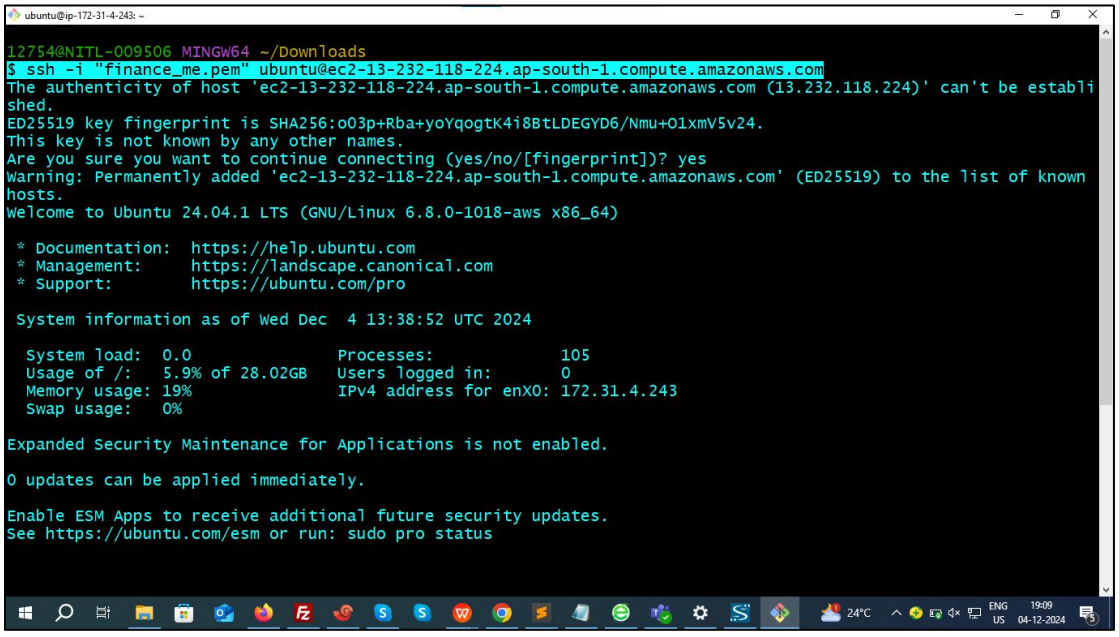


2. Connect to the VM:

Use SSH to connect.

EC2>>Instances>>i00678f747e2d88f5e>>Connect_to_instance

ssh -i "finance_me.pem" ubuntu@ec2-13-232-118-224.ap-south-1.compute.amazonaws.com



3. Install Terraform:

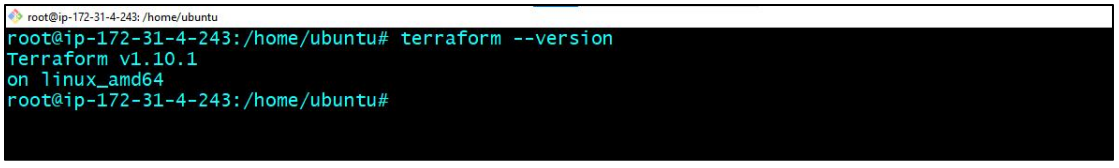
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list

sudo apt update && sudo apt install terraform

Verify installation.

terraform --version



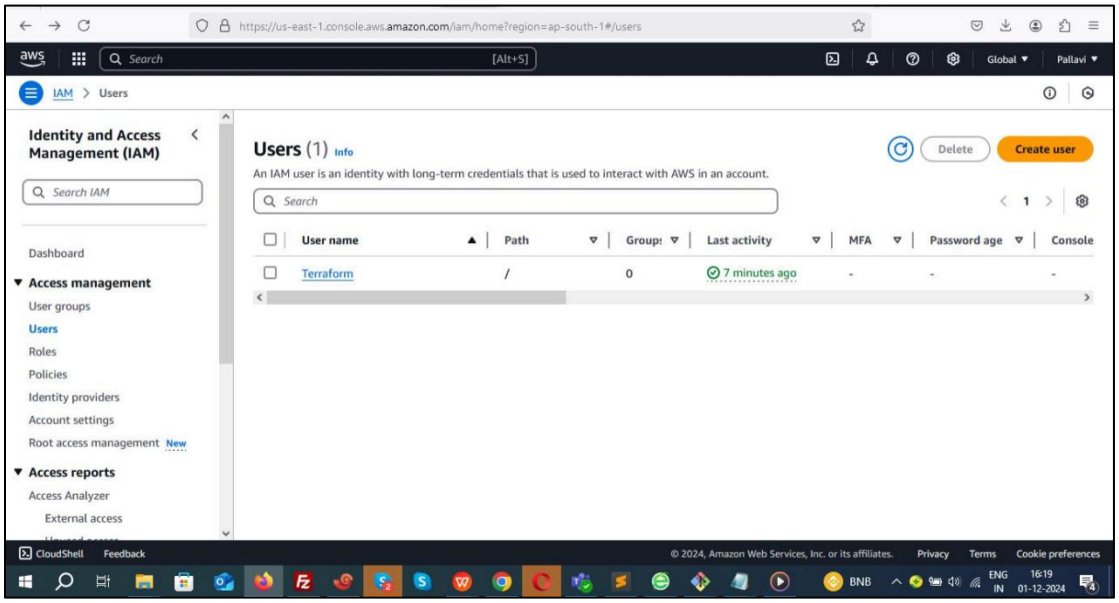
4.Establishing a connection between Terraform and AWS.

Create AWS Access Key and Secret Key.

Creating an Access Key and Secret Key in AWS is a crucial step to allow Terraform to authenticate and interact with AWS resources.

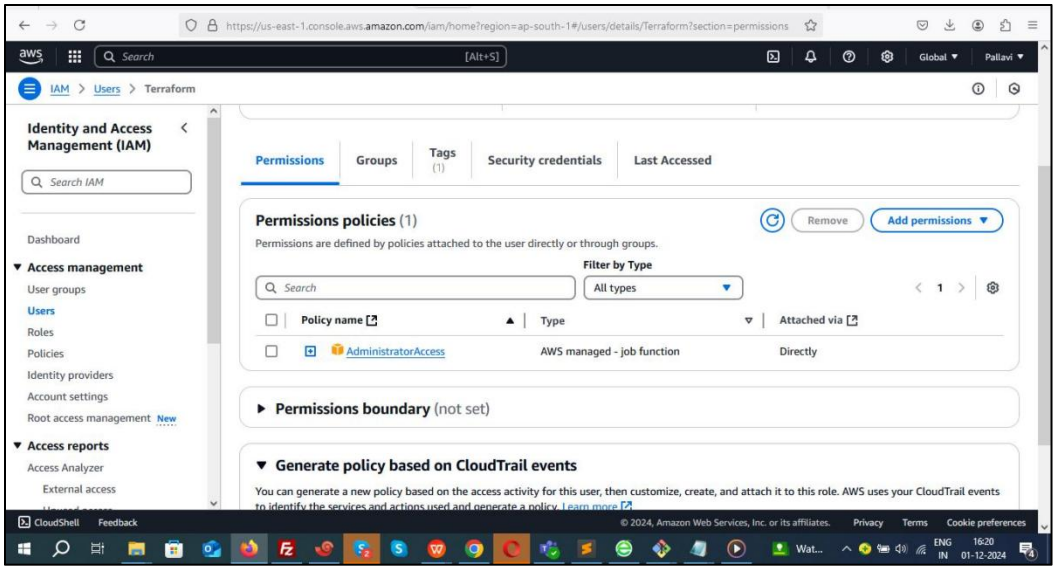
1. **Create an AWS IAM User with Administrator Access**

Login to AWS Management Console.
Navigate to IAM > Users.
Create a New User: Enable **Programmatic Access**.



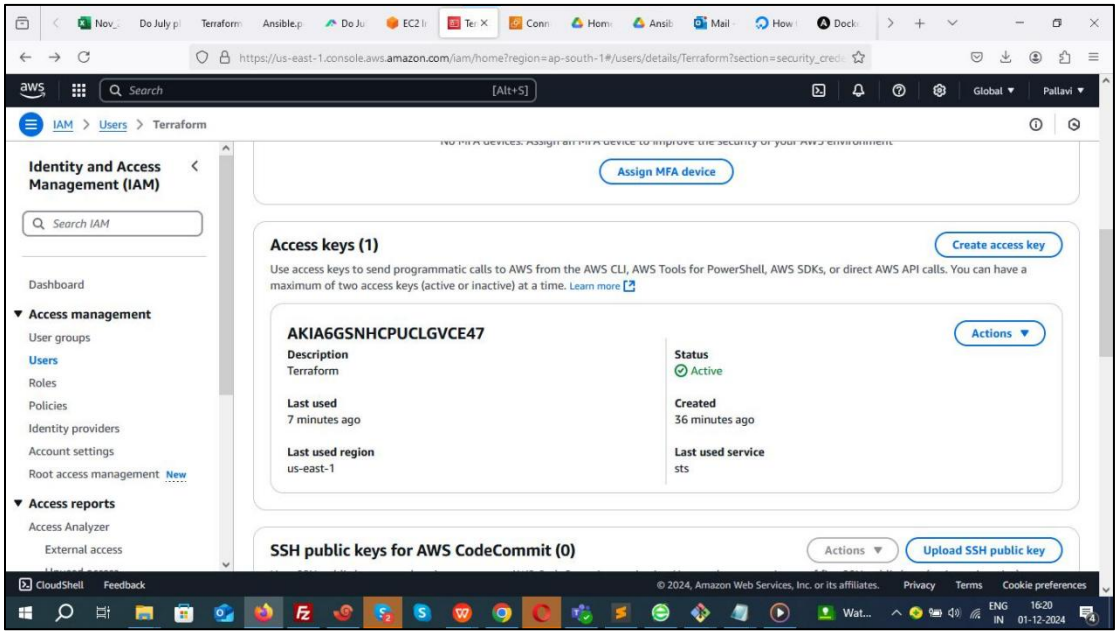
Attach a Policy:

Attach the **AdministratorAccess** policy to the user.



Generate Access Keys:

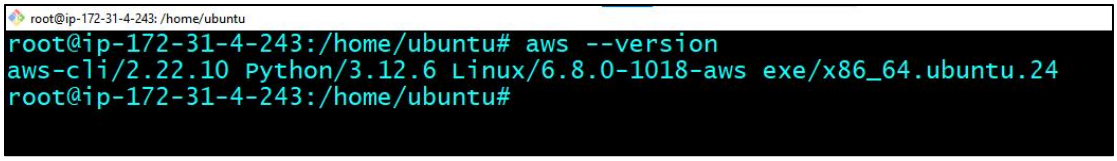
Save the **Access Key ID** and **Secret Access Key** securely (you will need these).



1. **Configure AWS CLI**

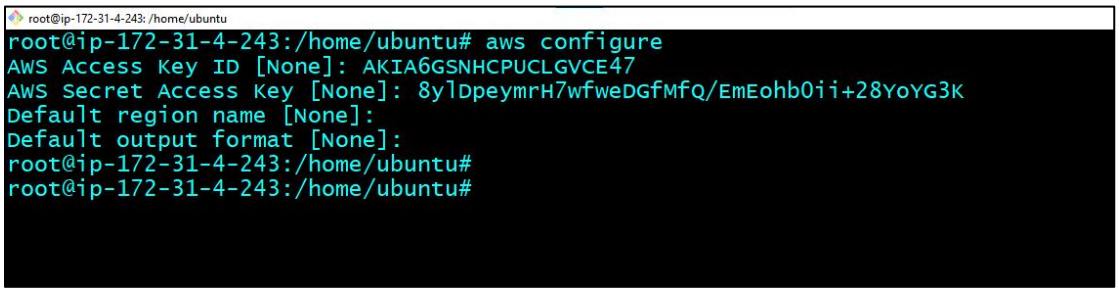
Install AWS CLI (if not installed):

sudo apt install awscli



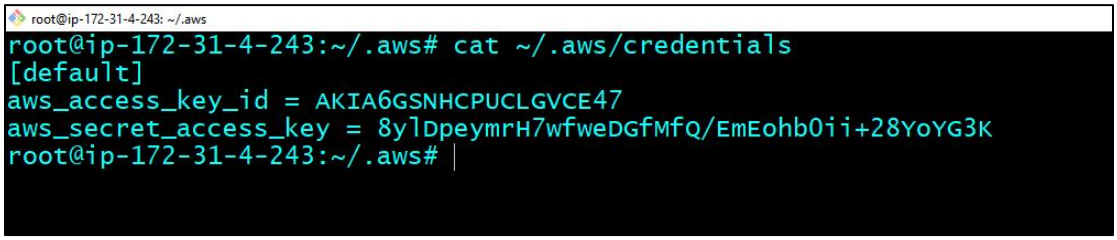
aws configure

Enter the **Access Key ID**, **Secret Access Key**, region (e.g., us-east-1), and output
format (json).



Verify AWS Configuration:

cat ~/.aws/credentials



5.Write Terraform Configuration File:

Create dedicated directory and navigate.

mkdir finance_me && cd finance_me

vim aws_infra.tf

Paste the provided Terraform configuration code to create an EC2 instance, VPC, Subnet, Security Group, and install Ansible.

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

```
  shared_credentials_files = ["~/.aws/credentials"]
}

resource "tls_private_key" "mykey" {
  algorithm = "RSA"
}

resource "aws_key_pair" "aws_key" {
  key_name   = "web-key"
  public_key = tls_private_key.mykey.public_key_openssh

  provisioner "local-exec" {
    command = "echo
'${tls_private_key.mykey.private_key_openssh}' > ./web-key.pem"
  }
}

resource "aws_vpc" "sl-vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "sl-vpc"
  }
}
```

```
resource "aws_subnet" "subnet-1" {
 vpc_id                = aws_vpc.sl-vpc.id
 cidr_block            = "10.0.1.0/24"
 depends_on            = [aws_vpc.sl-vpc]
 map_public_ip_on_launch = true
 tags = {
  Name = "sl-subnet"
 }
}

resource "aws_route_table" "sl-route-table" {
 vpc_id = aws_vpc.sl-vpc.id
 tags = {
  Name = "sl-route-table"
 }
}

resource "aws_route_table_association" "a" {
 subnet_id      = aws_subnet.subnet-1.id
 route_table_id = aws_route_table.sl-route-table.id
}
```

```
resource "aws_internet_gateway" "gw" {
  vpc_id     = aws_vpc.sl-vpc.id
  depends_on = [aws_vpc.sl-vpc]
  tags = {
    Name = "sl-gw"
  }
}

resource "aws_route" "sl-route" {
  route_table_id         = aws_route_table.sl-route-table.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.gw.id
}

variable "sg_ports" {
  type    = list(number)
  default = [8080, 80, 22, 443]
}

resource "aws_security_group" "sl-sg" {
  name   = "sg_rule"
  vpc_id = aws_vpc.sl-vpc.id
```

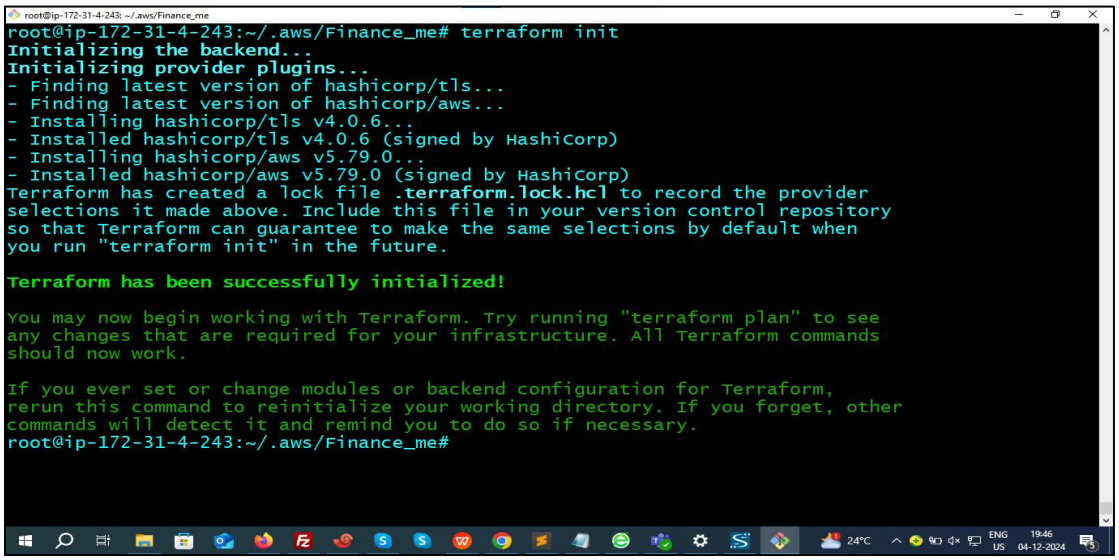
```
  dynamic "ingress" {
    for_each = var.sg_ports
    iterator = port
    content {
      from_port   = port.value
      to_port     = port.value
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "myec2" {
  ami           = "ami-005fc0f236362e99f"
  instance_type = "t2.medium"
  key_name      = "web-key"
```

```
  subnet_id     = aws_subnet.subnet-1.id
  security_groups = [
   aws_security_group.sl-sg.id
  ]
  tags = {
   Name = "Finance_me"
  }

  provisioner "remote-exec" {
   connection {
     type       = "ssh"
     user       = "ubuntu"
     private_key = tls_private_key.mykey.private_key_pem
     host       = self.public_ip
   }
   inline = [
     "sudo apt update",
     "sudo apt install software-properties-common",
     "sudo add-apt-repository --yes --update
ppa:ansible/ansible",
     "sudo apt install ansible -y"
   ]
  }
```

```
}
```

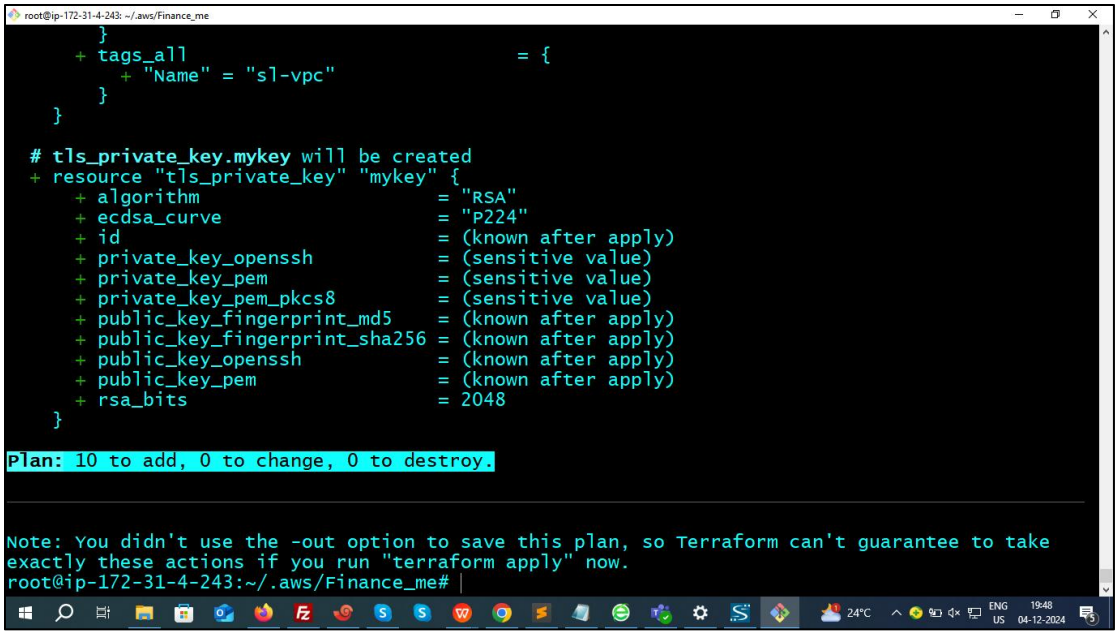
Initialize and Apply Terraform:

terraform init - Prepares the working directory for Terraform by downloading plugins and setting up the backend.

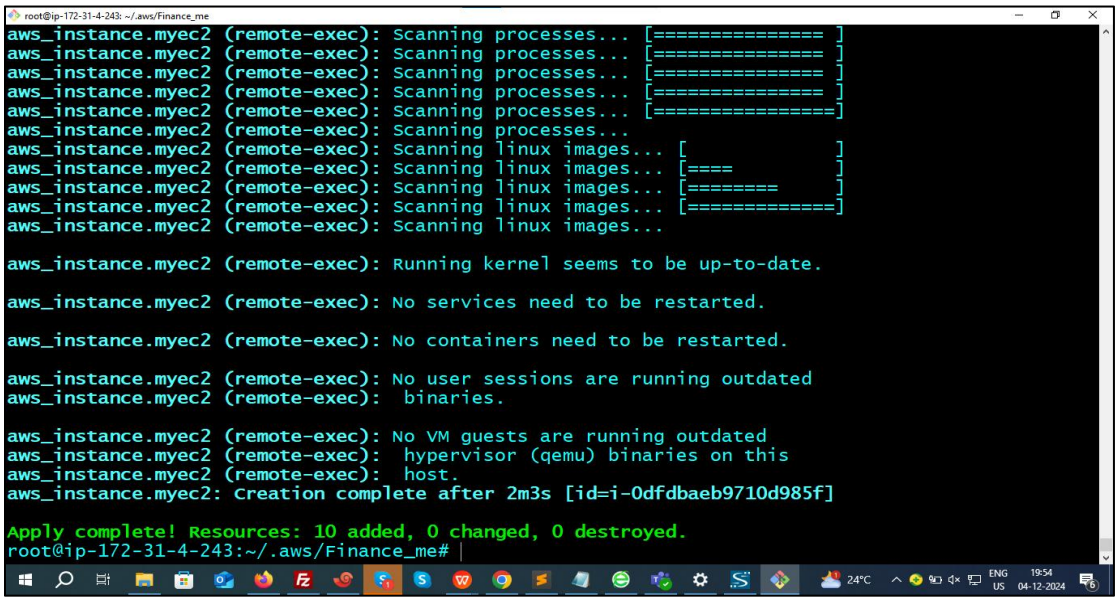


terraform validate - Checks for syntax errors and configuration validity.

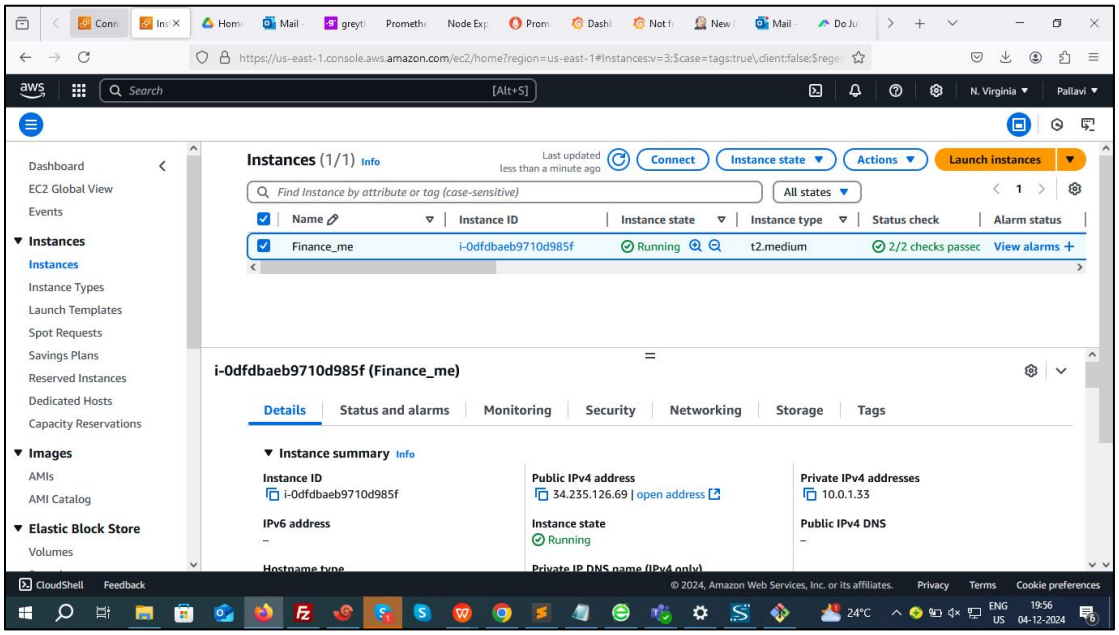
terraform plan - Generates and displays an execution plan without making changes.



terraform apply --auto-approve - Executes the plan and creates or updates infrastructure without requiring manual confirmation.



Instance has been created in us-east-1



Step 2: Configure Ansible on the Created EC2 Instance

1. **Connect to the Newly Created EC2 Instance**:

Use SSH with the generated private key (web-key.pem).

2. Check Ansible Installation

ansible --version

3. **Install Java, git and docker Using Ansible Playbook**:

Create an Ansible playbook:

vim finance_me.yml

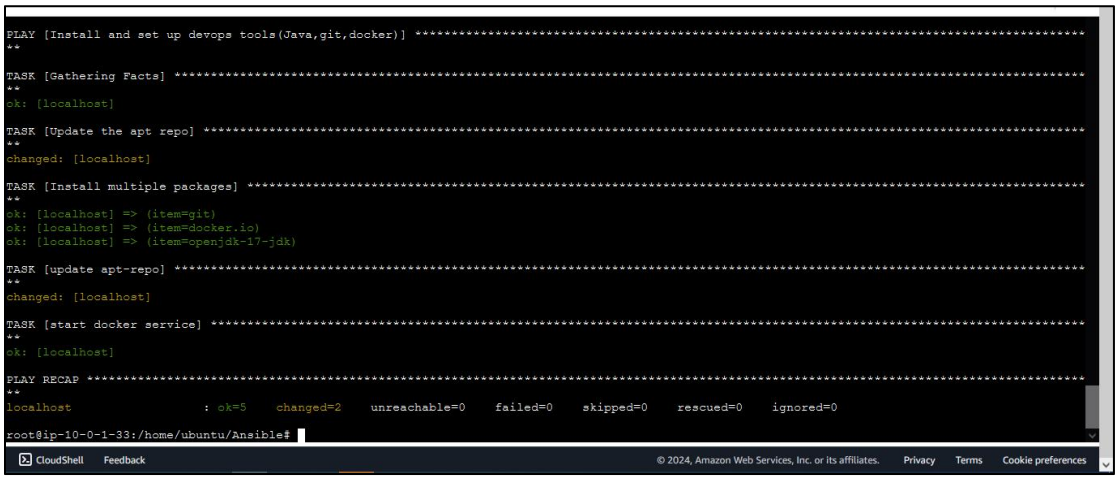
Add the following:

```
- name: Install and set up devops tools(Java,git,docker)
  hosts: localhost
  become: true
  tasks:
  - name: Update the apt repo
```

```
    command: apt-get update
  - name: Install multiple packages
    package: name={{item}} state=present
    loop:
      - git
      - docker.io
      - openjdk-17-jdk
  - name: update apt-repo
    command: sudo apt-get update
  - name: start docker service
    service: name=docker state=started
```

Run the playbook:

ansible-playbook finance_me.yml



Verify installations.

```
root@ip-10-0-1-33:/home/ubuntu/Ansible# java --version
openjdk 17.0.13 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu122.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu122.04, mixed mode, sharing)
root@ip-10-0-1-33:/home/ubuntu/Ansible# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu2~22.04.1
root@ip-10-0-1-33:/home/ubuntu/Ansible# git --version
git version 2.34.1
root@ip-10-0-1-33:/home/ubuntu/Ansible#
```

Step 3: Installing and setup Jenkins CI/CD Pipeline

1. Install Jenkins.

ADD GPG KEY:

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

ADD Jenkins repository:

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

UPDATE:

sudo apt-get update

INSTALL

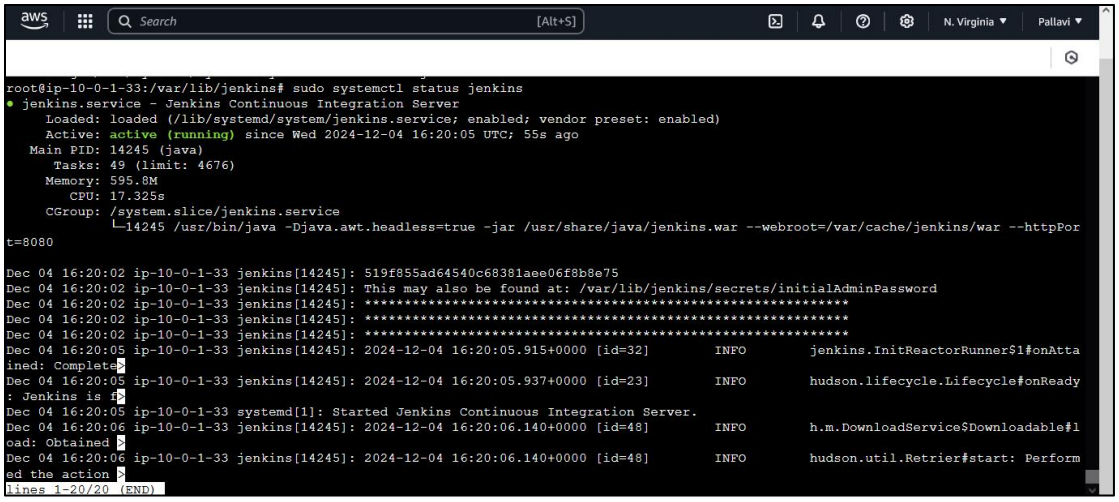
sudo apt-get install jenkins -y

Start and Enable Jenkins:

sudo systemctl start jenkins
sudo systemctl enable jenkins

Verify:

sudo systemctl status jenkins



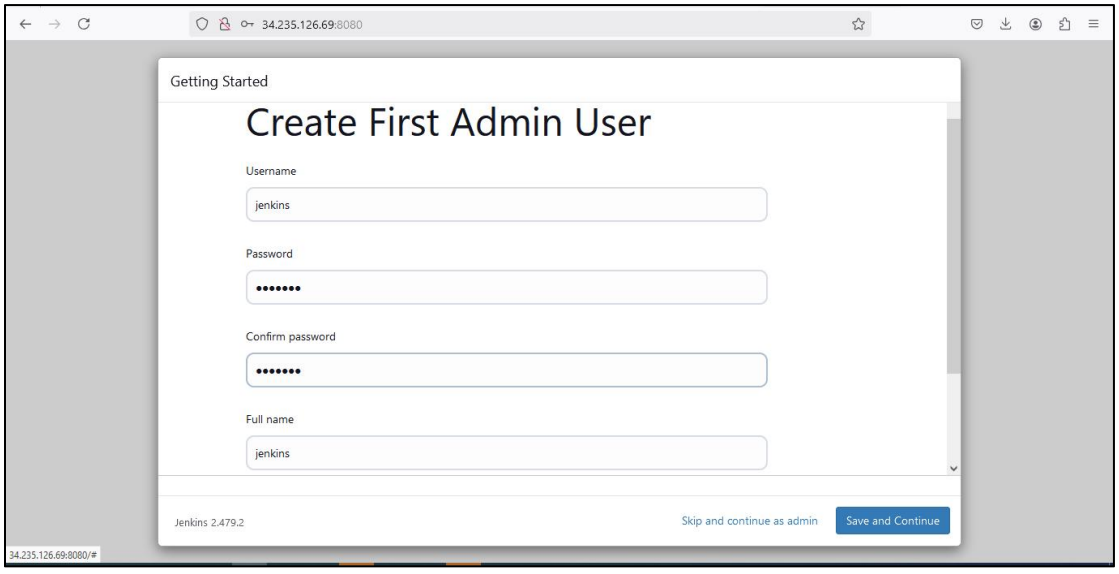
1. Now access to the jenkins dashboard using below url:

http://34.235.126.69:8080

Default port: 8080.

2. Set up plugins and admin user.

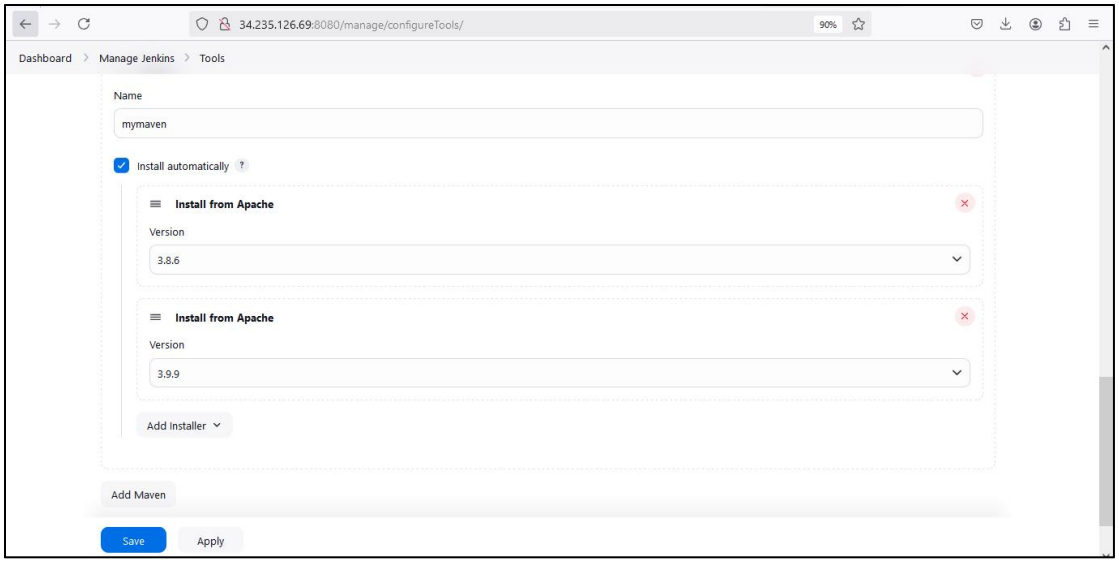
cat /var/lib/jenkins/secrets/initialAdminPassword



3. **Configure Maven**:

Go to **Manage Jenkins > Global Tool Configuration**.

Add Maven and set it up.



Make sure to give same name for maven in pipeline.

4. **Create Pipeline**:

Points to be noted:

In source code go to src/main/resources/application.properties

Check server port

docker run -d -p 8081:8081 myproject1:$BUILD_NUMBER

Map the internal container port (8081) to an external port on your host machine

docker run -d -p 8081:8081 myproject1:$BUILD_NUMBER

Enable 8081 in your ec2 security groups.

Navigate to **New Item > Pipeline**.

```
pipeline {
    agent any
    tools {
        maven 'mymaven'
    }
    stages {
        stage('Clone Repo') {
            steps {
                git 'https://github.com/CSPPallavi/star-agile-banking-finance.git'
            }
        }
        stage('Test Code') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Build Code') {
            steps {
                sh 'mvn package'
            }
        }
        stage('Build Image') {
```

```
            steps {
                sh 'docker build -t myproject1:$BUILD_NUMBER .'
            }
        }
        stage('Deploy the Image') {
            steps {
                sh 'docker run -d -p 8081:8081 myproject1:$BUILD_NUMBER'
            }
        }
    }
}
```

· Clone Repo: Clone the Git repository containing the project.

· Test Code: Run tests using Maven to ensure that the code behaves as expected.

· Build Code: Compile and package the code into an artifact (like a JAR or WAR).

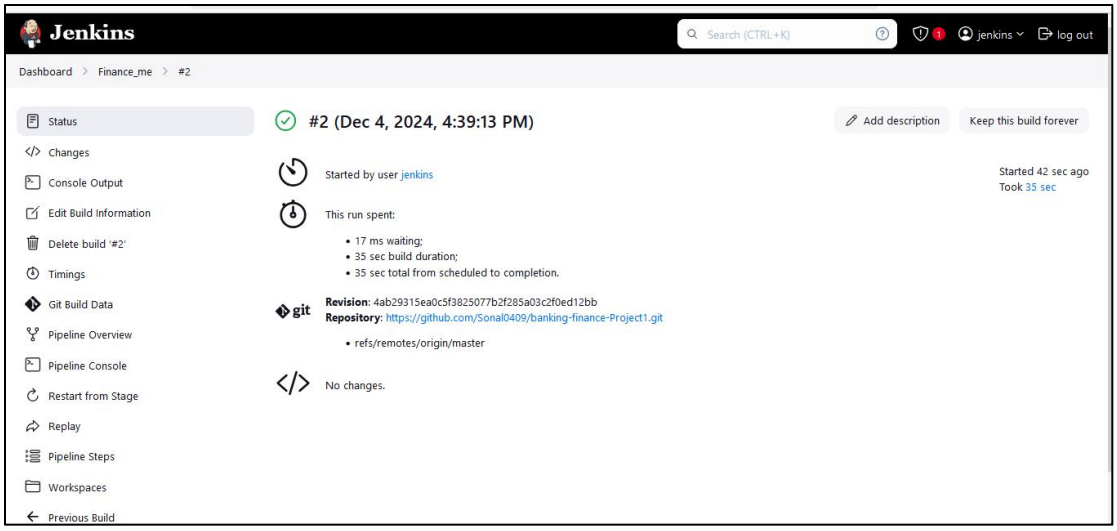
· Build Image: Create a Docker image based on the project's Dockerfile.

· Deploy the Image: Run the newly built Docker image in a container.

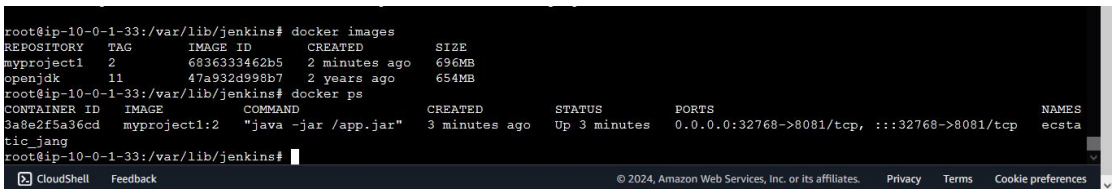
Befor building the job allow Jenkins to Run Docker Commands:

chmod -R 777 /var/run/docker.sock

Build the job successfully.

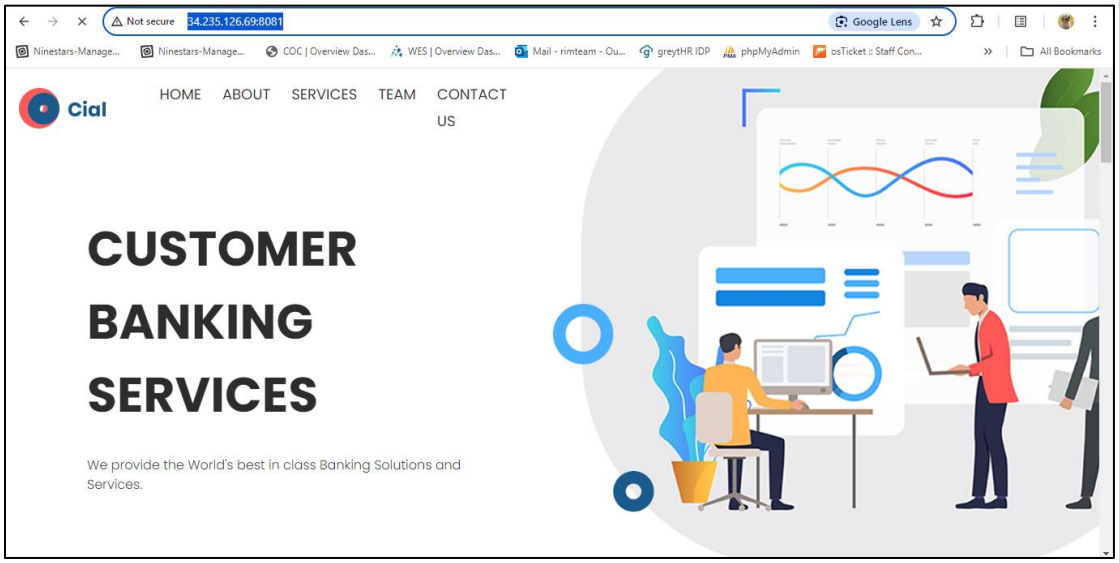


Verify docker image and container.



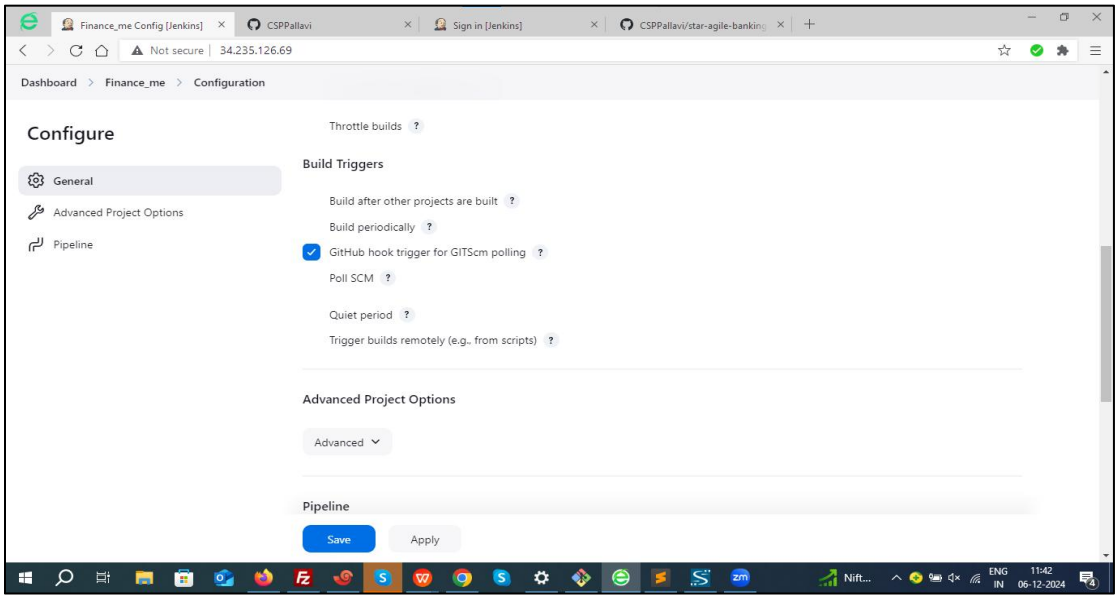
Access the container

http://34.235.126.69:8081



STEP 4: WEBHOOK CONFIGURATION:

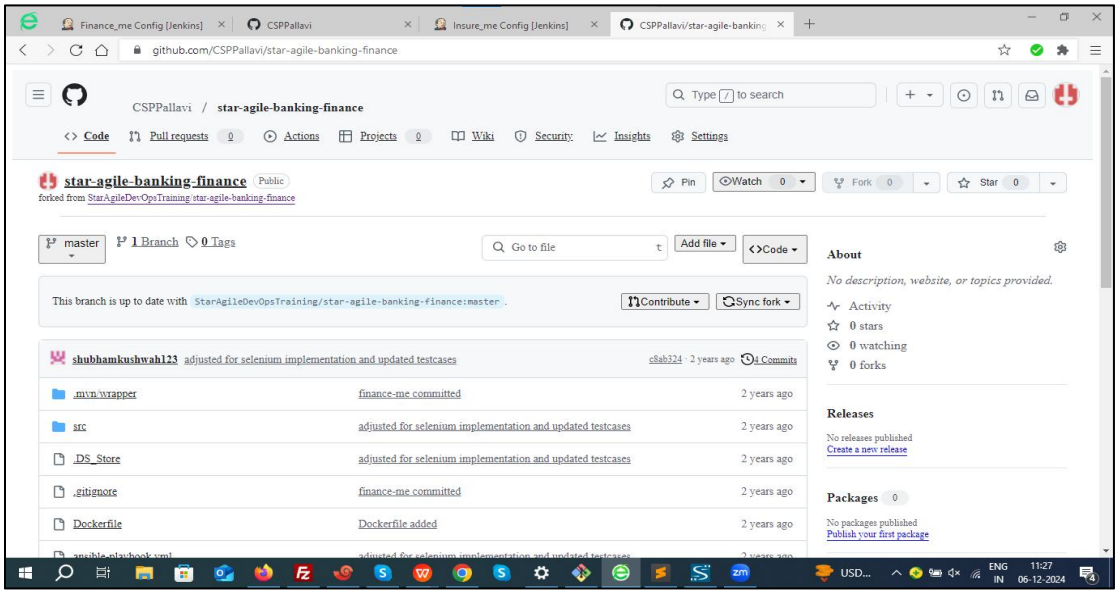
Go to your Jenkins dashboard and add build trigger- **GitHub hook trigger for GITScm polling**



Configuring a webhook in Git is a crucial step to enable automated actions, such as triggering a CI/CD pipeline in Jenkins or notifying an application about repository events.

Step 1: Open Your Repository Settings

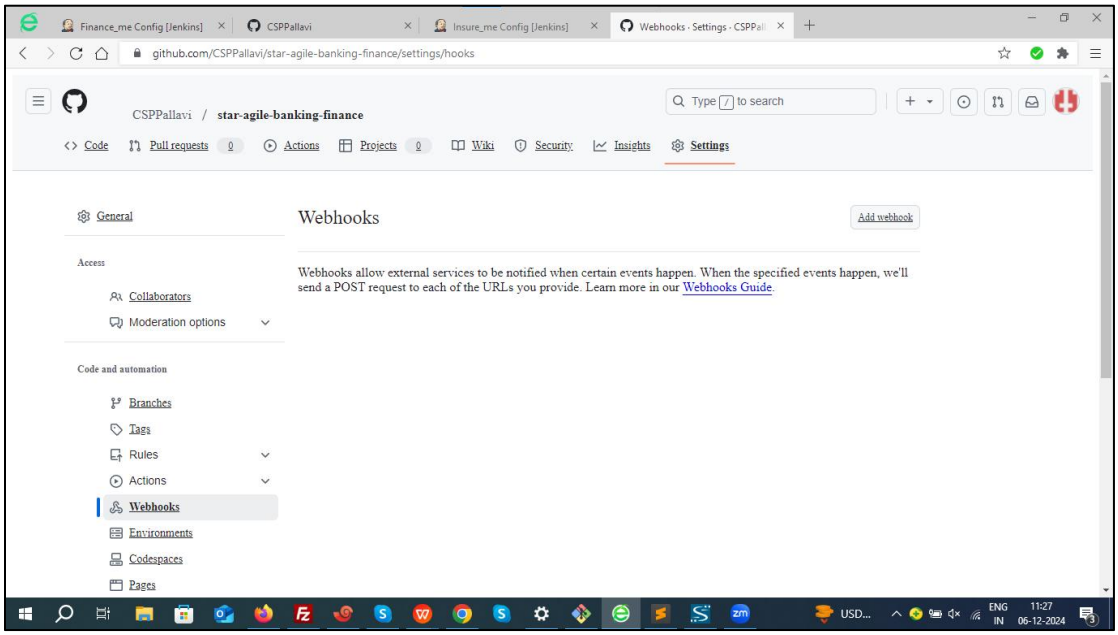
1. Navigate to the repository where you want to configure the webhook.



StarAgile

Step 2: Find the Webhooks Section

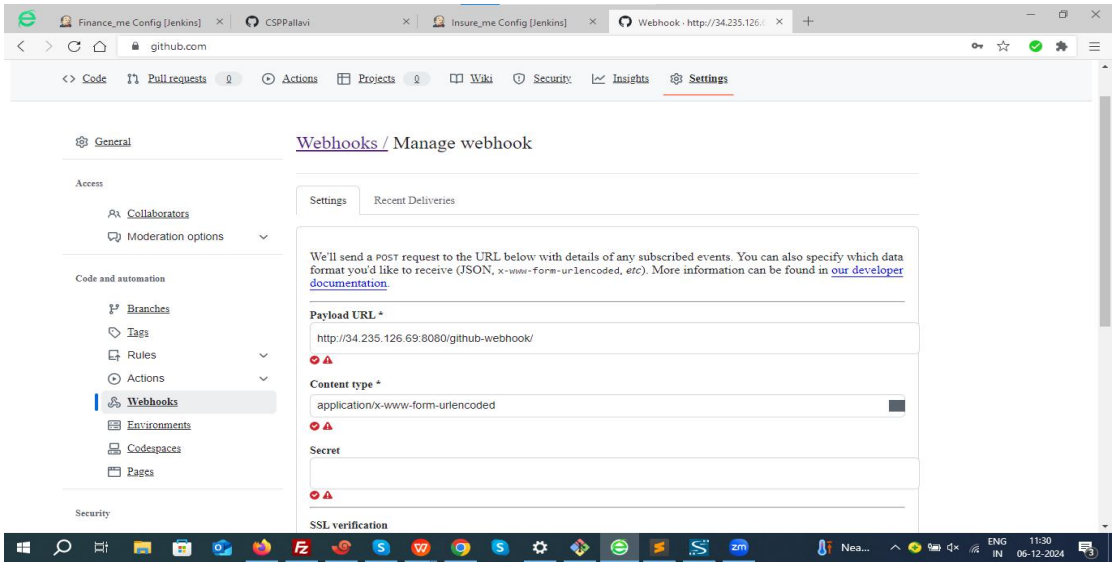
Go to **Repository Settings** > **Webhooks**.



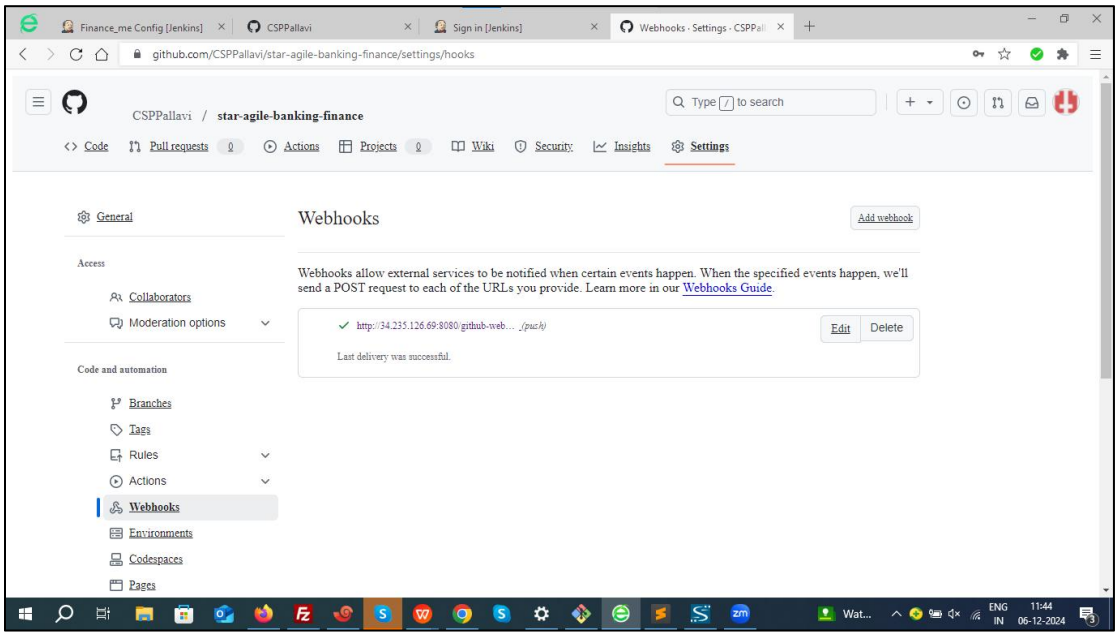
Step 3: Add a New Webhook

Fill in the webhook details:
Payload URL:
· This is the endpoint that will receive the webhook payload.
· http://<Jenkins-URL>/github-webhook/

http://34.235.126.69:8080/github-webhook/

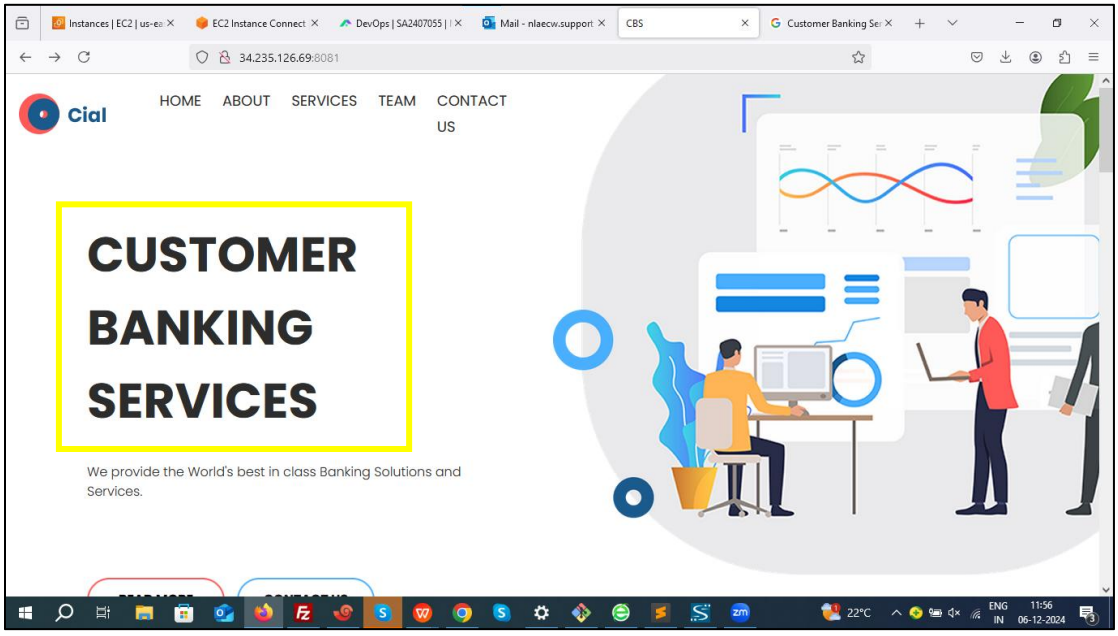


Step 4 : Commit changes and save.



Step 5 : Testing webhook.

Lets make changes in thehome page of the application.

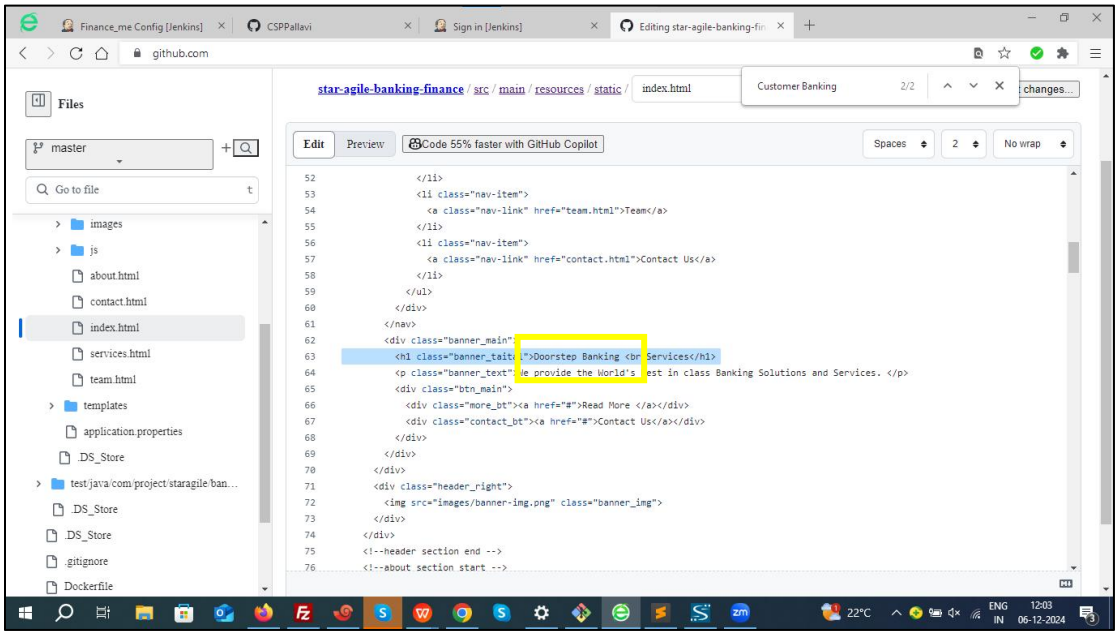


Let's make changes in welcome page where it says **Customer Banking Services.**

Let's change this to **Doorstep Banking Services**

Go to index.html path in your git hub

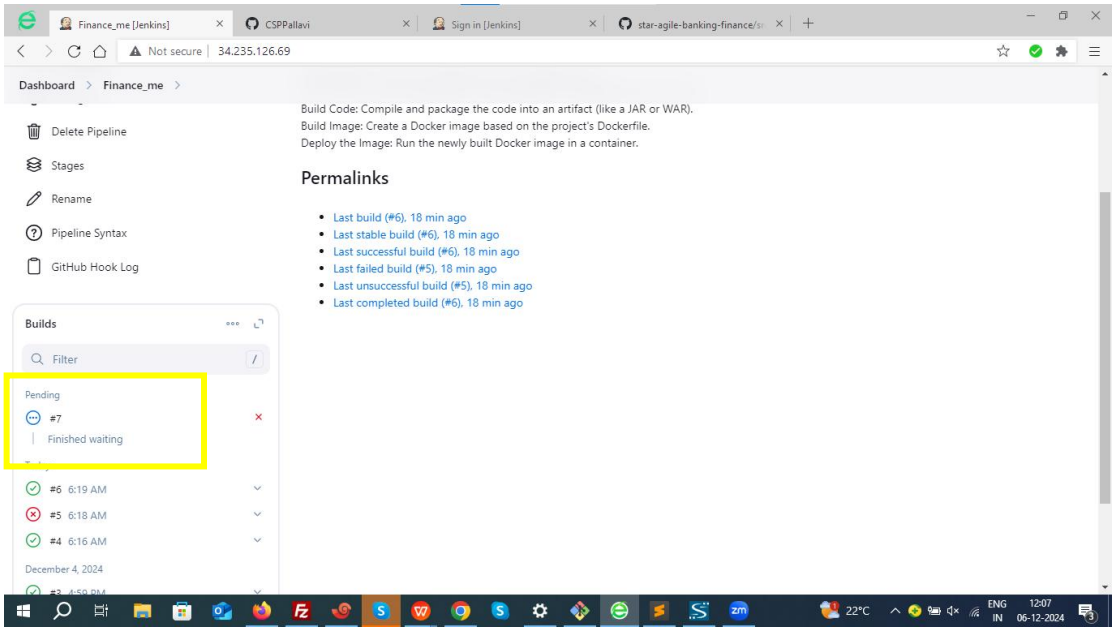
star-agile-insurance-project/src/main/resources/static



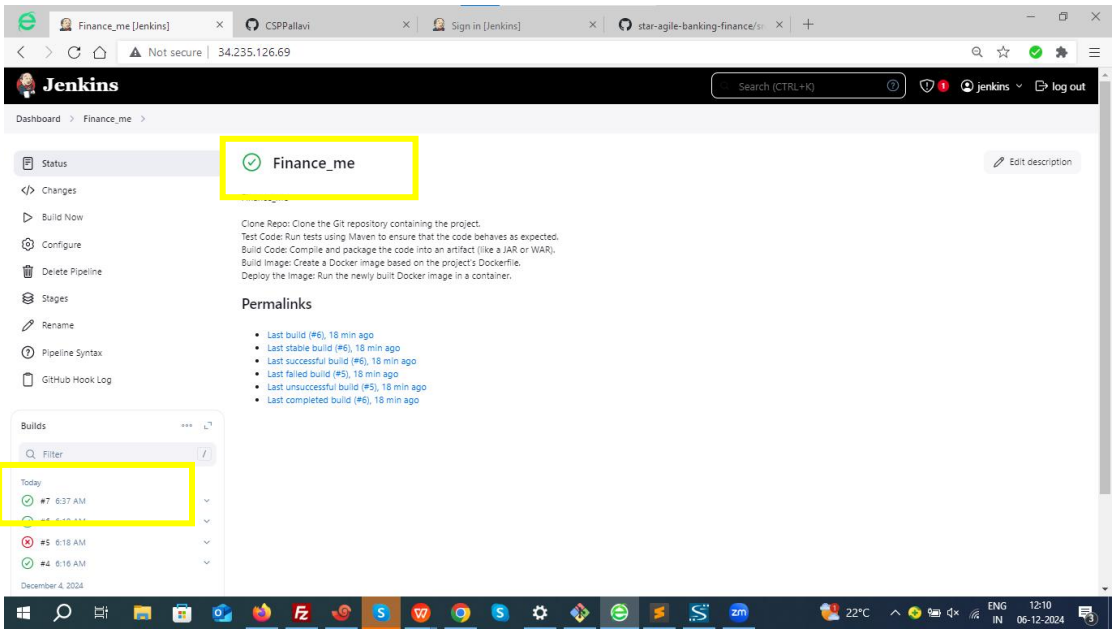
Now commit the changes.

Immidiately a new builds shoud trigger.

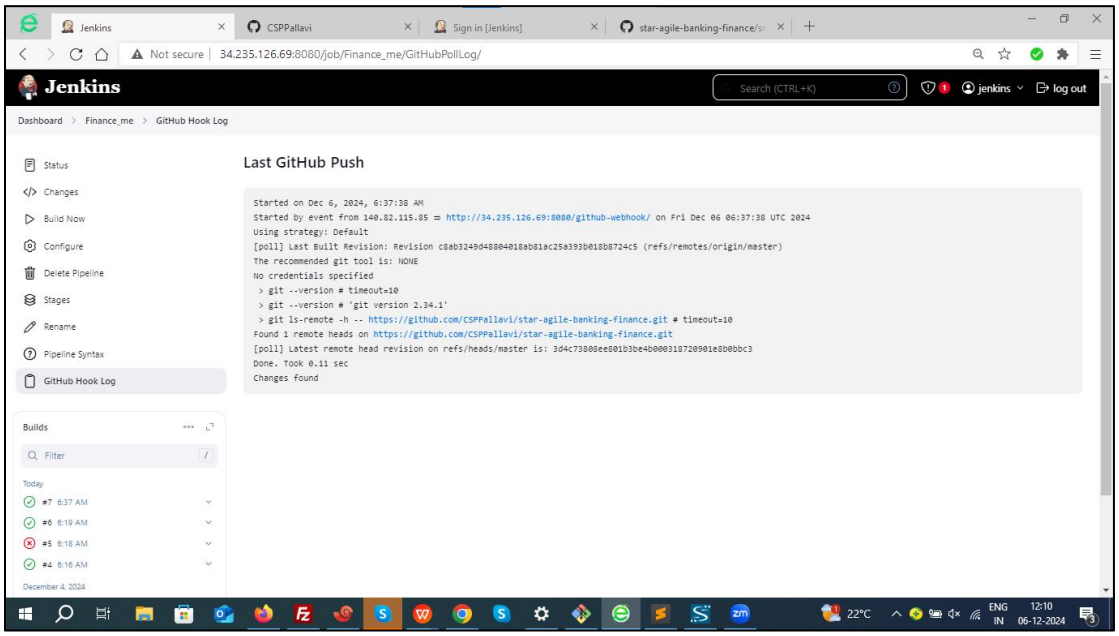
In the below snapshot it is evident a new build 7 has been triggered.



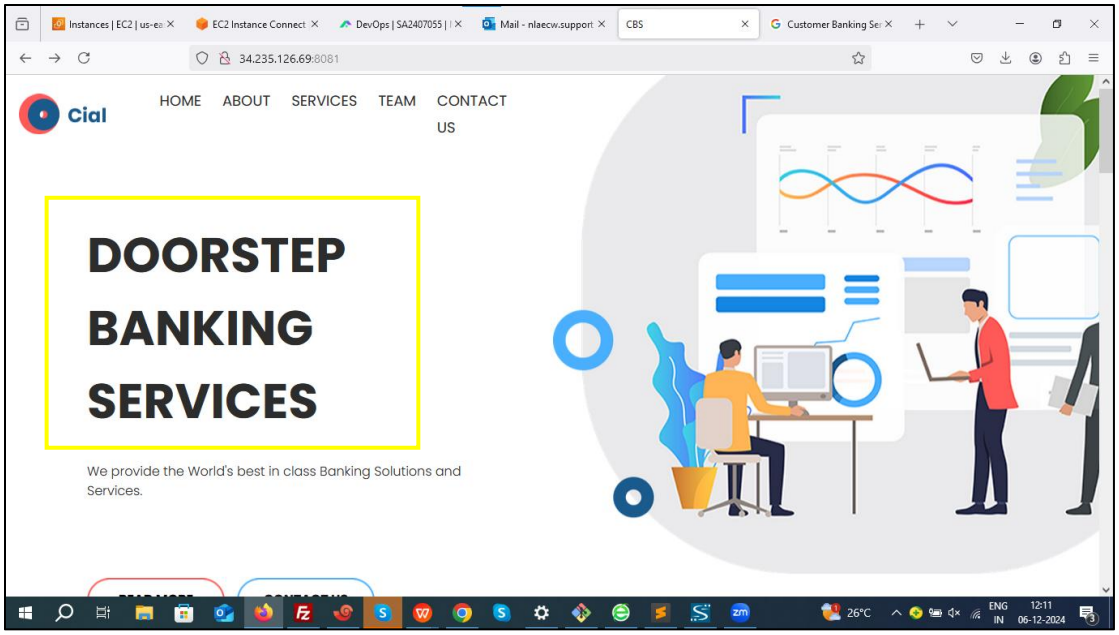
Build 7 is success



Verify the Github hook logs in Jenkins.



Verify the application to ensure changes has been occurred.



Step 5: Setup Monitoring with Prometheus and Grafana

Prometheus Setup

Step 1: Download the Latest Prometheus

Go to Prometheus Releases and get the latest version link.

Replace <version> with the appropriate version.

wget https://github.com/prometheus/prometheus/releases/latest/download/prometheus-<version>.linux-amd64.tar.gz

wget https://github.com/prometheus/prometheus/releases/download/v3.0.0/prometheus-3.0.0.linux-amd64.tar.gz

Step 2: Extract the Archive

tar -xvzf prometheus-<version>.linux-amd64.tar.gz

tar -xvzf prometheus-3.0.0.linux-amd64.tar.gz

```
2024-12-05 01:45:23 (102 MB/s) - 'prometheus-3.0.0.linux-amd64.tar.gz' saved [112998239/112998239]

root@ip-10-0-1-33:/home/ubuntu# tar -xvzf prometheus-3.0.0.linux-amd64.tar.gz
prometheus-3.0.0.linux-amd64/
prometheus-3.0.0.linux-amd64/promtool
prometheus-3.0.0.linux-amd64/LICENSE
prometheus-3.0.0.linux-amd64/prometheus
prometheus-3.0.0.linux-amd64/prometheus.yml
prometheus-3.0.0.linux-amd64/NOTICE
root@ip-10-0-1-33:/home/ubuntu# ll
total 110392
drwxr-x--- 6 ubuntu ubuntu      4096 Dec  5 01:45 ./
drwxr-xr-x 3 root   root        4096 Dec  4 14:19 ../
-rw------- 1 ubuntu ubuntu        10 Dec  4 15:38 .bash_history
-rw-r--r-- 1 ubuntu ubuntu       220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu      3771 Jan  6  2022 .bashrc
drwx------ 2 ubuntu ubuntu      4096 Dec  4 14:19 .cache/
-rw-r--r-- 1 ubuntu ubuntu       807 Jan  6  2022 .profile
drwx------ 2 ubuntu ubuntu      4096 Dec  4 14:19 .ssh/
-rw-r--r-- 1 ubuntu ubuntu         0 Dec  4 14:19 .sudo_as_admin_successful
drwxrwxrwx 2 root   root        4096 Dec  4 16:07 [illegible]/
drwxr-xr-x 2   1001 docker      4096 Nov 14 16:54 prometheus-3.0.0.linux-amd64/
-rw-r--r-- 1 root   root   112998239 Nov 14 16:58 prometheus-3.0.0.linux-amd64.tar.gz
root@ip-10-0-1-33:/home/ubuntu#
```

cd prometheus-<version>.linux-amd64/

cd prometheus-3.0.0.linux-amd64

Step 3: Move Files to Appropriate Directories

sudo mv prometheus /usr/local/bin/
sudo mv promtool /usr/local/bin/
sudo mkdir -p /etc/prometheus /var/lib/prometheus
sudo mv prometheus.yml /etc/prometheus/

```
-rwxr-xr-x 1   1001 docker 139895586 Nov 14 16:42 promtool*
root@ip-10-0-1-33:/home/ubuntu/prometheus-3.0.0.linux-amd64# sudo mv prometheus /usr/local/bin/
root@ip-10-0-1-33:/home/ubuntu/prometheus-3.0.0.linux-amd64# sudo mv promtool /usr/local/bin/
root@ip-10-0-1-33:/home/ubuntu/prometheus-3.0.0.linux-amd64# sudo mkdir -p /etc/prometheus /var/lib/prometheus
root@ip-10-0-1-33:/home/ubuntu/prometheus-3.0.0.linux-amd64# sudo mv prometheus.yml /etc/prometheus/
root@ip-10-0-1-33:/home/ubuntu/prometheus-3.0.0.linux-amd64#
```

Step 4: Create Prometheus User

sudo useradd --no-create-home --shell /usr/sbin/nologin prometheus
sudo chown -R prometheus:prometheus /etc/prometheus /var/lib/prometheus
sudo chown prometheus:prometheus /usr/local/bin/prometheus /usr/local/bin/promtool

Step 5: Set Up the Systemd Service

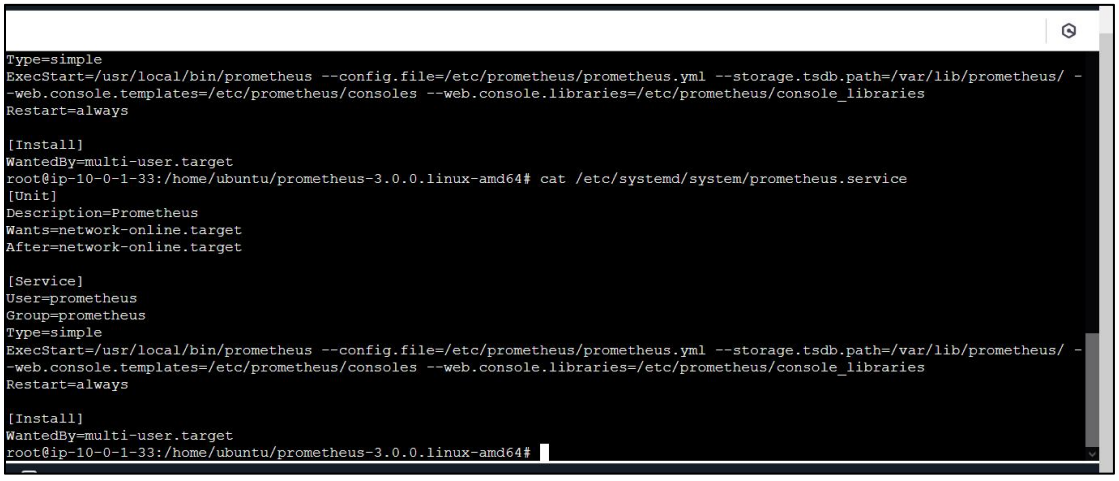
sudo nano /etc/systemd/system/prometheus.service

Create the service file:

sudo nano /etc/systemd/system/prometheus.service

Paste the following configuration:

```
[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
Restart=always

[Install]
WantedBy=multi-user.target
```

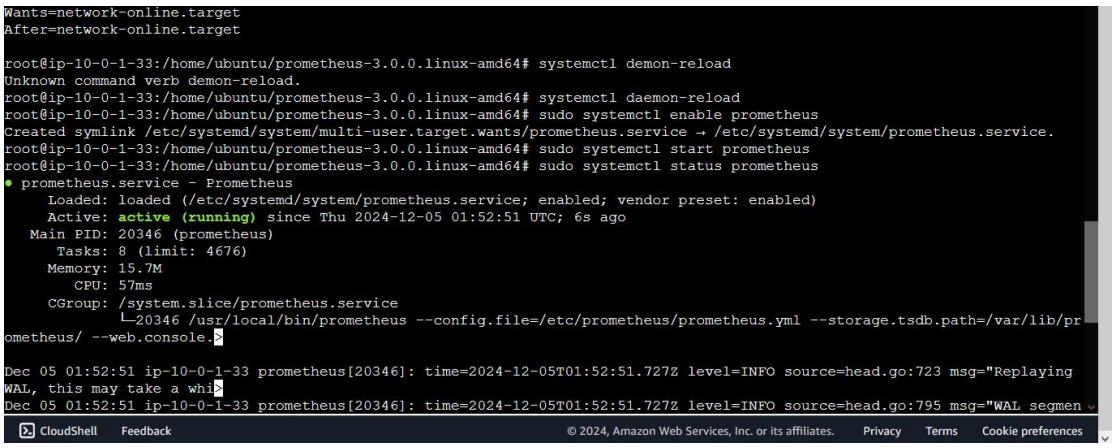


Reload systemd and start Prometheus:

```
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start prometheus
```

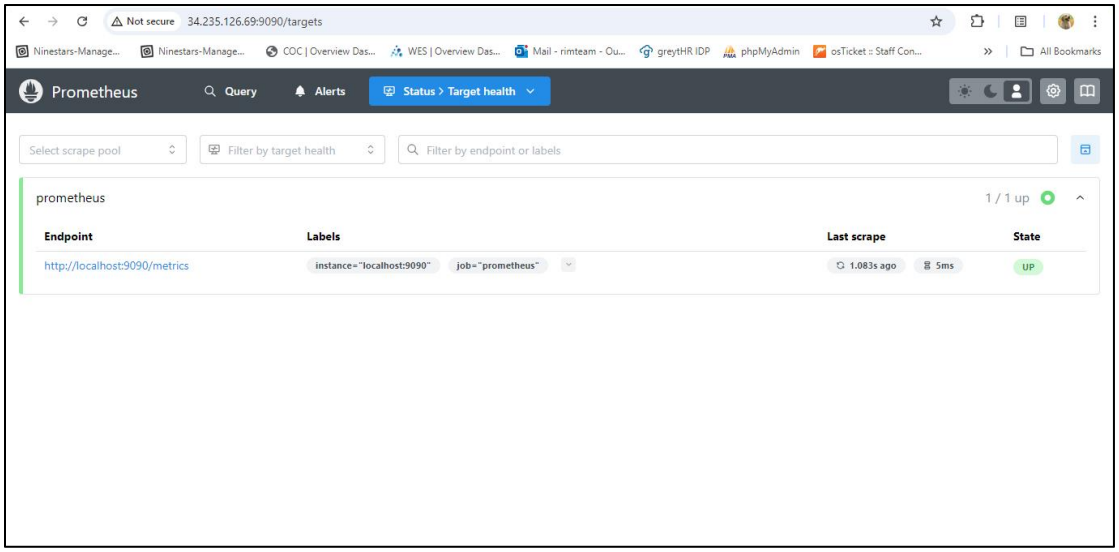
5. Verify the Installation

sudo systemctl status prometheus



Check Prometheus at http://<server-ip>:9090 in your browser.

http://34.235.126.69:9090



Node ExporterSetup

1. Download Node Exporter

Get the latest release of Node Exporter: Visit the Node Exporter GitHub Releases page to find the latest version.

Download the latest Node Exporter tarball: Replace <version> with the desired version (e.g., 1.6.1).

wget https://github.com/prometheus/node_exporter/releases/download/v<version>/node_exporter-<version>.linux-amd64.tar.gz

wget https://github.com/prometheus/node_exporter/releases/download/v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz

Extract the tarball:

tar -xvzf node_exporter-<version>.linux-amd64.tar.gz

tar -xvzf node_exporter-1.8.2.linux-amd64.tar.gz

Move the binary to /usr/local/bin**:**

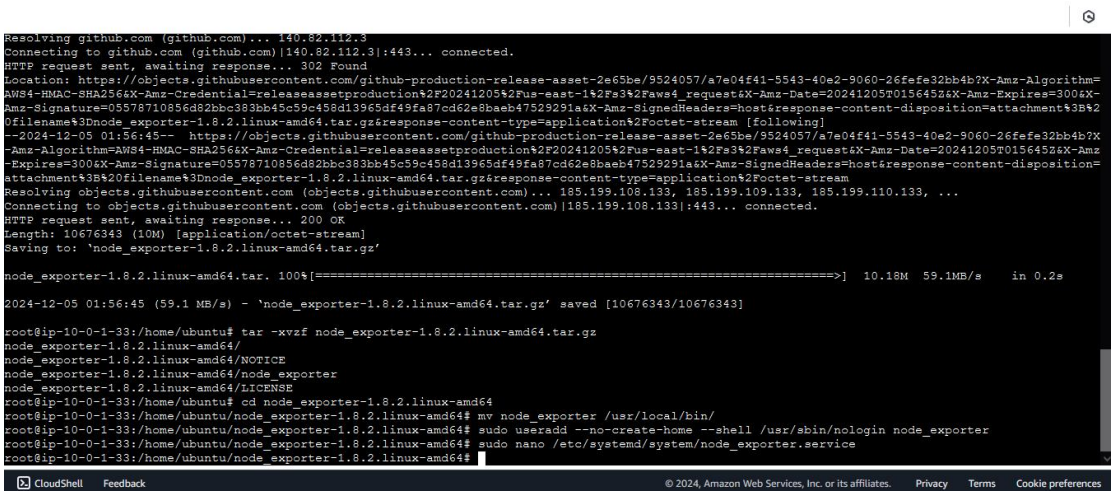
sudo mv node_exporter-<version>.linux-amd64/node_exporter /usr/local/bin/

mv node_exporter /usr/local/bin/

2. Create a Node Exporter User

For security, create a dedicated user without shell access:

sudo useradd --no-create-home --shell /usr/sbin/nologin node_exporter



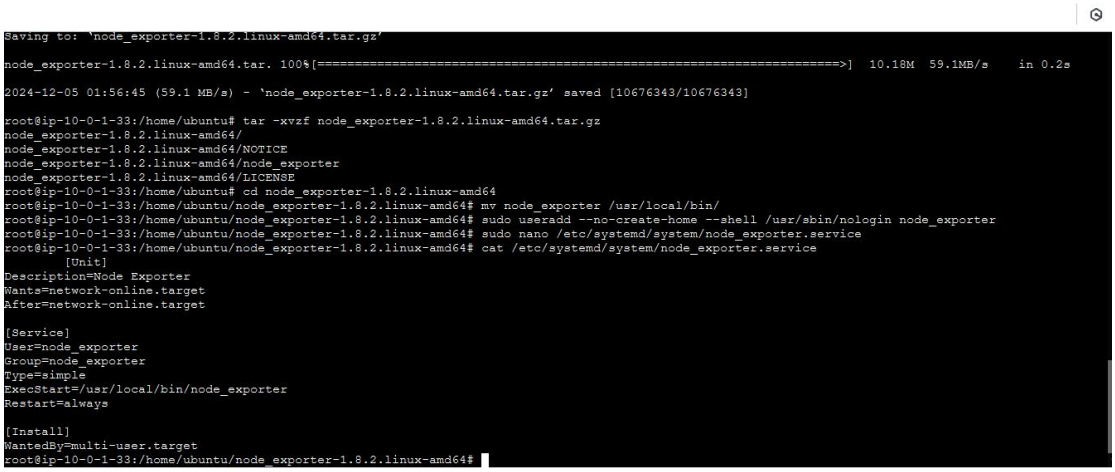
3. Set Up a Systemd Service

Create a systemd service file for Node Exporter:

sudo nano /etc/systemd/system/node_exporter.service

Add the following content:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
Restart=always

[Install]
WantedBy=multi-user.target
```



Reload the systemd daemon:

sudo systemctl daemon-reload

Enable and start Node Exporter:

sudo systemctl enable node_exporter
sudo systemctl start node_exporter

Verify the service is running:

sudo systemctl status node_exporter

```
ExecStart=/usr/local/bin/node_exporter
Restart=always

[Install]
WantedBy=multi-user.target
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl daemon-reload
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl start node_exporter
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-12-05 01:59:28 UTC; 6s ago
   Main PID: 20484 (node_exporter)
      Tasks: 5 (limit: 4676)
     Memory: 2.5M
        CPU: 7ms
     CGroup: /system.slice/node_exporter.service
             └─20484 /usr/local/bin/node_exporter

Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=time
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=timex
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=udp_queues
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=uname
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=vmstat
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=watchdog
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=xfs
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=node_exporter.go:118 level=info collector=zfs
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=tls_config.go:313 level=info msg="Listening on" address=[::]:
Dec 05 01:59:28 ip-10-0-1-33 node_exporter[20484]: ts=2024-12-05T01:59:28.315Z caller=tls_config.go:316 level=info msg="TLS is disabled." http2=fal
lines 1-20/20 (END)
```

4. Configure Prometheus to Scrape Node Exporter

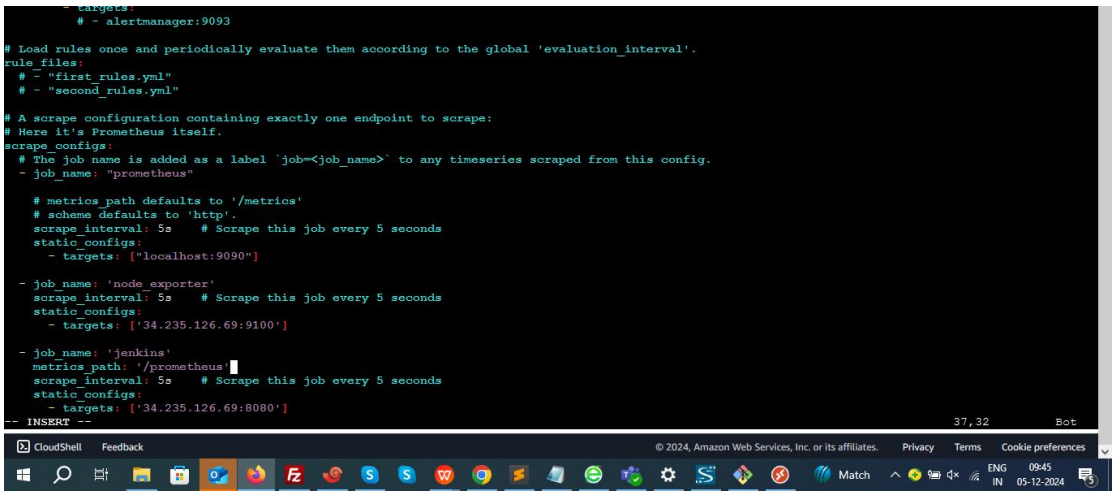
Edit the Prometheus configuration file:

sudo nano /etc/prometheus/prometheus.yml

Add the Node Exporter job under scrape_configs**:**

```
- job_name: 'node_exporter'
  scrape_interval: 5s    # Scrape this job every 5 seconds
  static_configs:
    - targets: ['34.235.126.69:9100']
```

Replace <node_exporter_host> with the server's IP or hostname where Node Exporter is running.



Restart Prometheus:

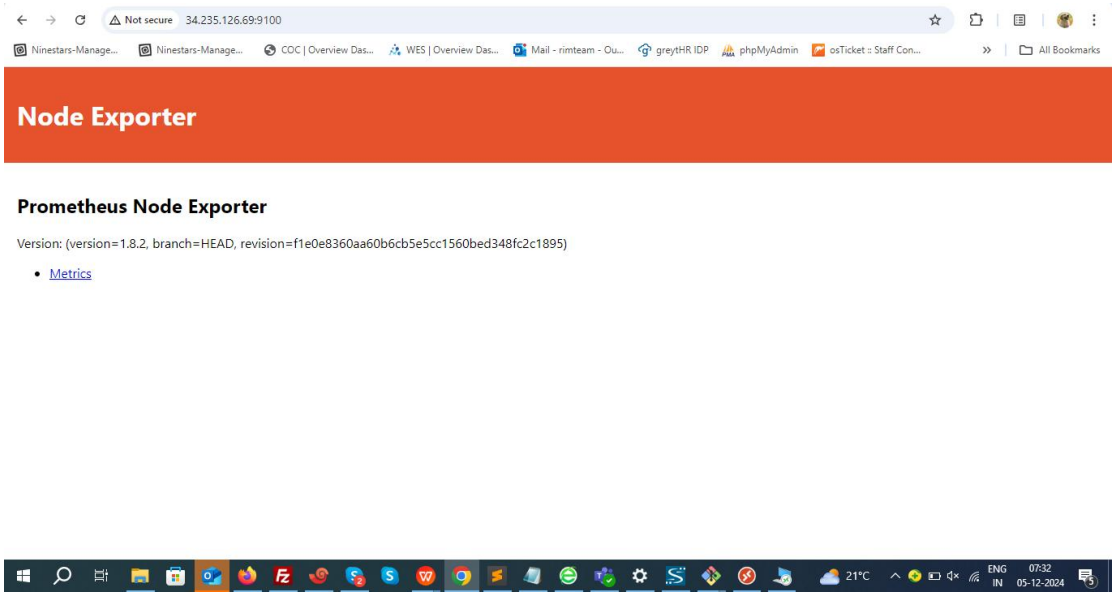
sudo systemctl restart prometheus

5. Verify Node Exporter Integration

Access Node Exporter metrics: Open your browser and visit:

http://<server_ip>:9100/metrics

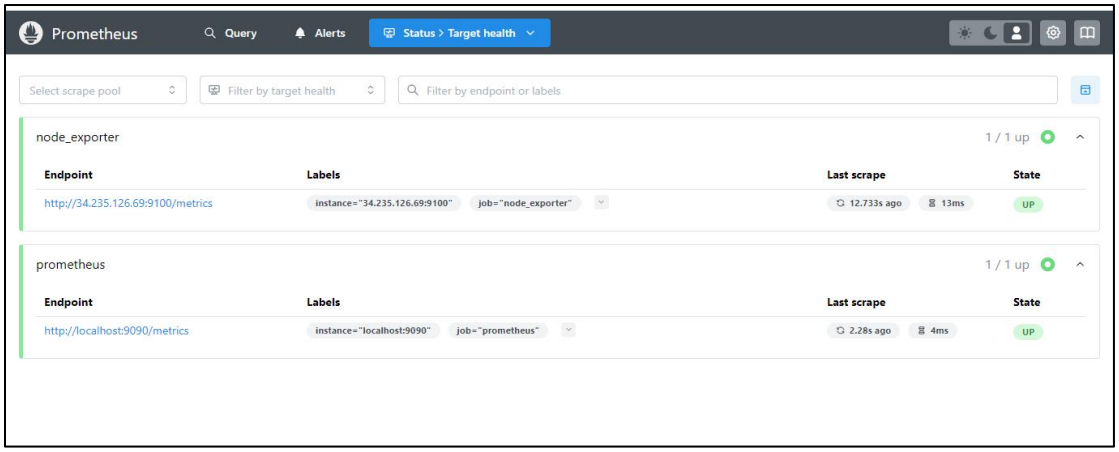
http://34.235.126.69:9100



Check node exporter setup in Prometheus:

Open Prometheus at http://<prometheus_ip>:9090, go to **Status > Targets**, and ensure the Node Exporter job is listed and up.

http://34.235.126.69:9090/targets



This setup ensures Node Exporter is installed, runs securely, and integrates with Prometheus.

Grafana installation:

Step 1: Download and Add the Grafana GPG Key

1. Open your terminal.

Download the Grafana GPG key and add it to your APT keyring:

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

Step 2: Add the Grafana Repository to APT Sources

Add the Grafana repository to your system's APT sources:

sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"

Step 3: Refresh the APT Cache

Update your package list to include the newly added Grafana repository:

sudo apt update

Step 4: Install Grafana

Now, you can install Grafana by running:

sudo apt install grafana

Step 5: Start the Grafana Server

Once Grafana is installed, start the Grafana service:

sudo systemctl start grafana-server

Step 6: Verify Grafana is Running

Check the status of the Grafana service to ensure it is running properly:

sudo systemctl status grafana-server

```
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.
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl start grafana-server
root@ip-10-0-1-33:/home/ubuntu/node_exporter-1.8.2.linux-amd64# sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; disabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-12-05 02:06:37 UTC; 7s ago
       Docs: http://docs.grafana.org
   Main PID: 22070 (grafana)
      Tasks: 16 (limit: 4676)
     Memory: 82.1M
        CPU: 3.028s
     CGroup: /system.slice/grafana-server.service
             └─22070 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=

Dec 05 02:06:43 ip-10-0-1-33 grafana[22070]: logger=provisioning.dashboard t=2024-12-05T02:06:43.6858182742 level=info msg="starting to provision d
Dec 05 02:06:43 ip-10-0-1-33 grafana[22070]: logger=provisioning.dashboard t=2024-12-05T02:06:43.6860071042 level=info msg="finished to provision d
Dec 05 02:06:43 ip-10-0-1-33 grafana[22070]: logger=plugin.angulardetectorsprovider.dynamic t=2024-12-05T02:06:43.7115415142 level=info msg="Patter
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=plugin.installer t=2024-12-05T02:06:44.0467485382 level=info msg="Installing plugin" pluginId=g
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=installer.fs t=2024-12-05T02:06:44.1772336712 level=info msg="Downloaded and extracted grafana-
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=plugins.registration t=2024-12-05T02:06:44.2481414862 level=info msg="Plugin registered" plugin
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=plugin.backgroundinstaller t=2024-12-05T02:06:44.2481968092 level=info msg="Plugin successfully
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=grafana-apiserver t=2024-12-05T02:06:44.3432466422 level=info msg="Adding GroupVersion playlist
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=grafana-apiserver t=2024-12-05T02:06:44.3437372752 level=info msg="Adding GroupVersion featuret
Dec 05 02:06:44 ip-10-0-1-33 grafana[22070]: logger=grafana-apiserver t=2024-12-05T02:06:44.3448976592 level=info msg="Adding GroupVersion iam.graf
lines 1-21/21 (END)
```

Step 7: Enable Grafana to Start on Boot

To ensure Grafana starts automatically when the server boots up, run:

sudo systemctl enable grafana-server

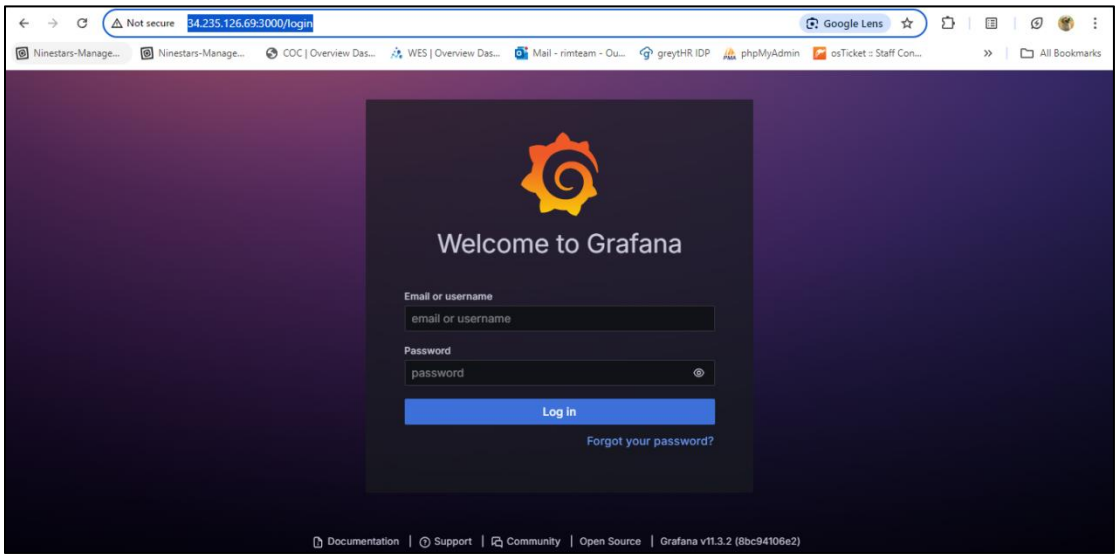
Step 8: Access Grafana in Browser

Open a web browser and navigate to:

http://<instance_ip>:3000

Replace <instance_ip> with the IP address of your server.

http://34.235.126.69:3000



Step 9: Default Login Credentials

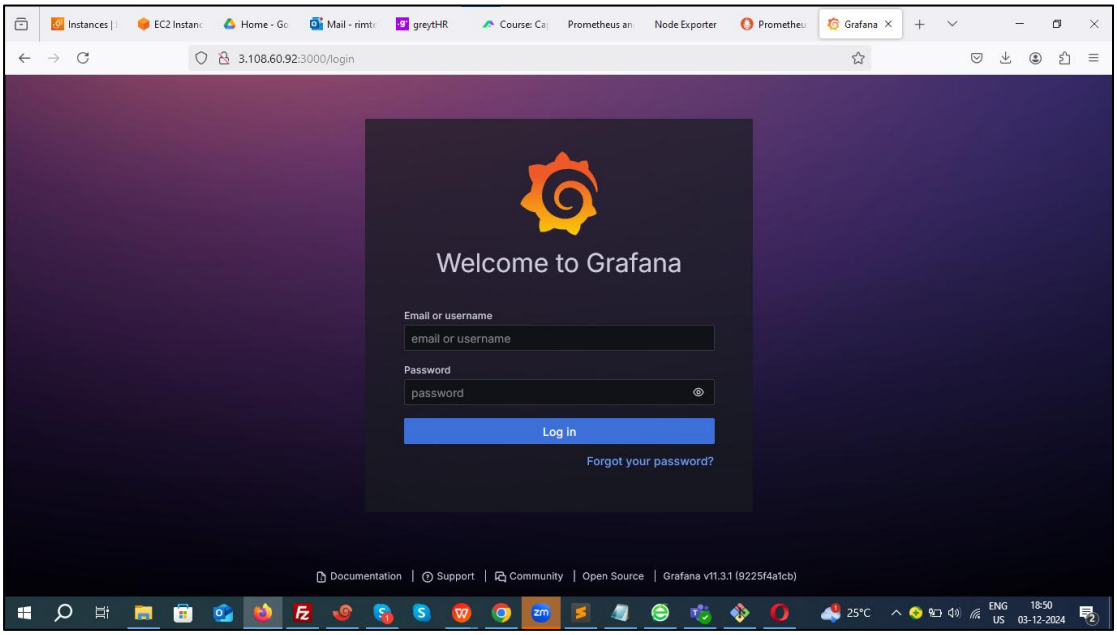
Use the default login credentials:

Username: admin

Password: admin

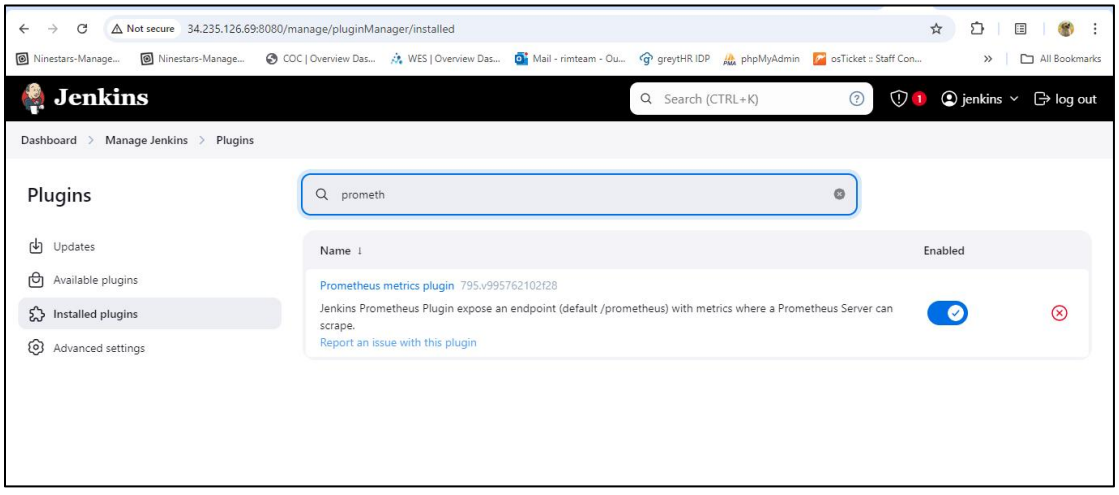
Once logged in, you will be prompted to change the default password.

That's it! You should now have Grafana installed and accessible on your Ubuntu machine.



1. Install prometheus plugin in Jenkins dashboard:

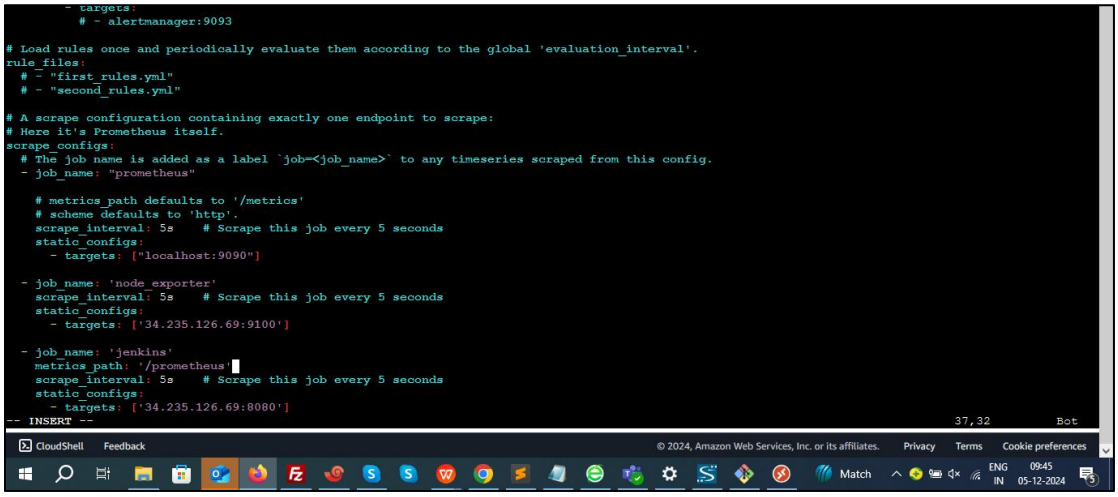
Setup dashboard → install prometheus plugin -> restart Jenkins



2. Ensure Prometheus is Monitoring Jenkins.

Go to prometheus.yaml file and paste below content.

```
sudo vim /etc/prometheus/prometheus.yml

- job_name: 'jenkins'

  metrics_path: '/prometheus'

  scrape_interval: 5s    # Scrape this job every 5 seconds

  static_configs:

    - targets: ['34.235.126.69:8080']
```

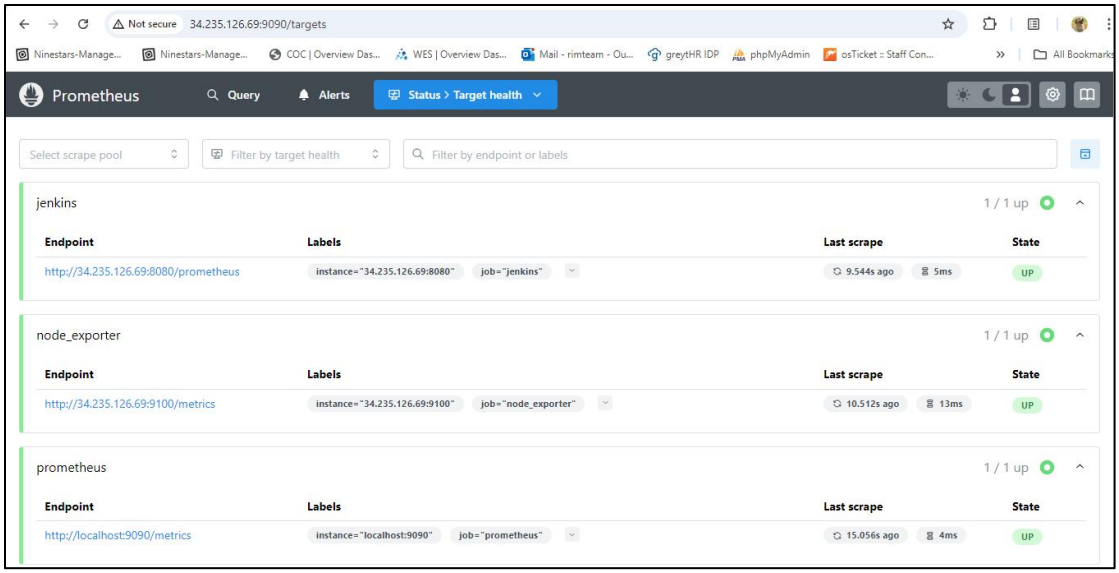


3. Restart prometheus

After changes in config file we need to restart prometheus

```
sudo systemctl restart prometheus
```

Check whether Jenkins is added to target health in Prometheus.

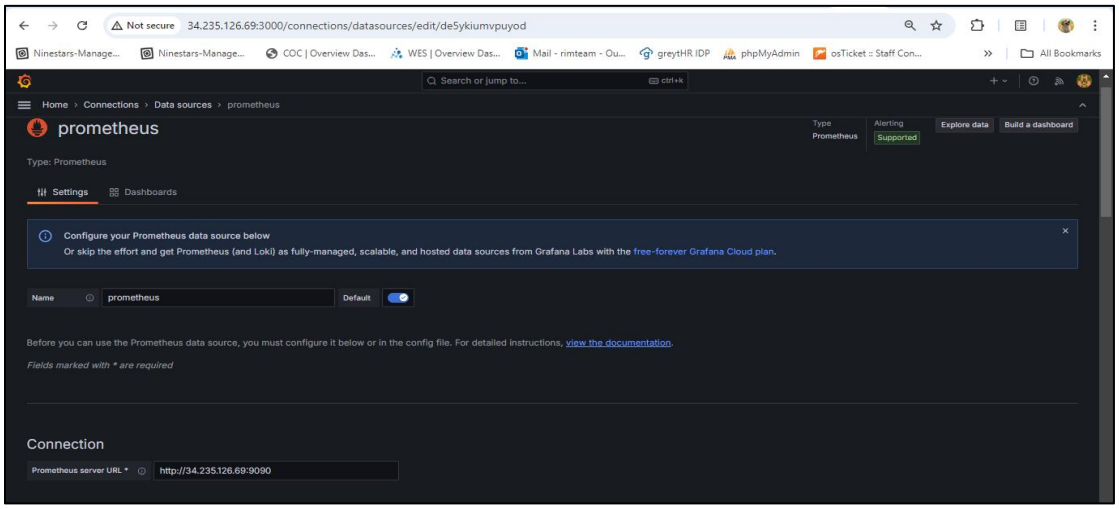


Setting up Grafana Dashboard:

Step 1: Add Prometheus as a Data Source in Grafana

1. Navigate to Home **> Connections** in Grafana.
2. Click **Add data source** and select **Prometheus**.

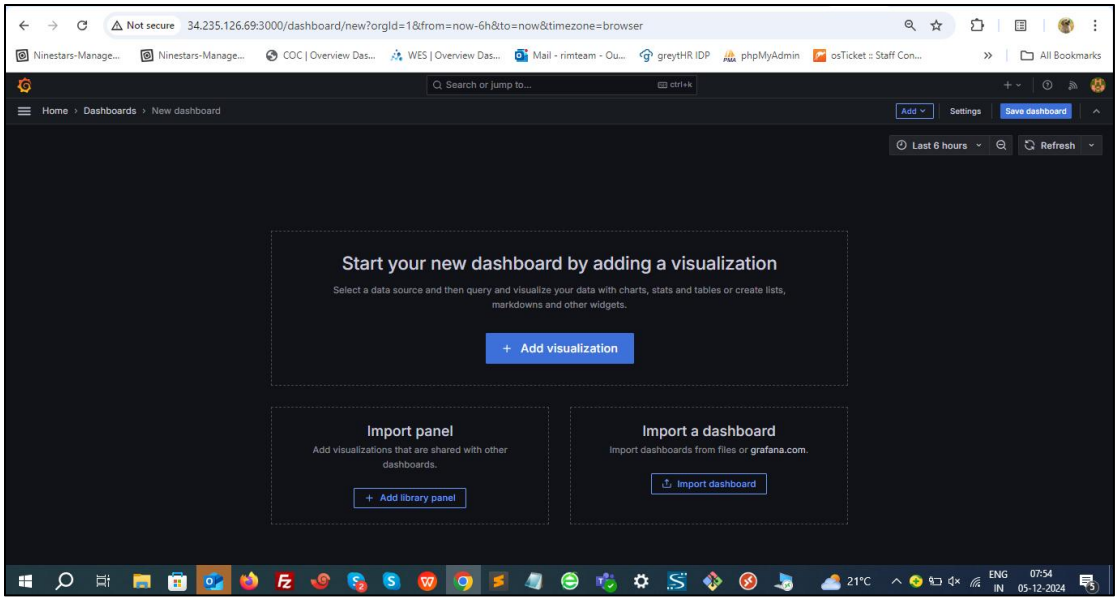
Add prometheus server url: http://34.235.126.69:9090



Save&Test

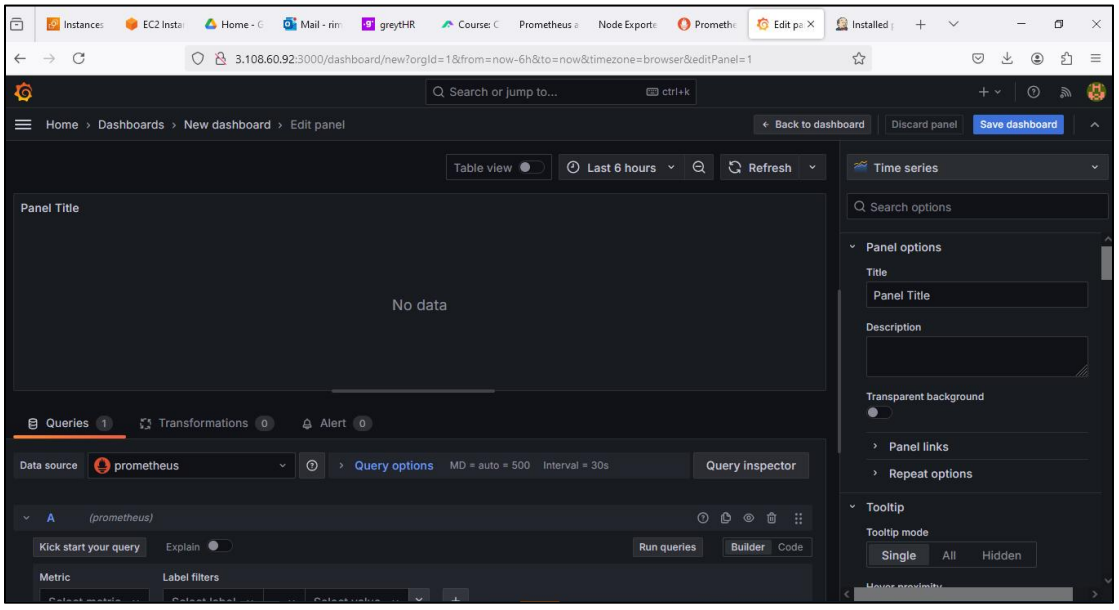
Step 3: Create a New Dashboard

1. In Grafana, click **+ > Dashboard > Add Visualization**



Select **Prometheus** as the data source.

Step 4: Add Panels for CPU Utilization, Memory Utilization and Disk Space Utilization



Panel 1: CPU UtilizationQuery:

PromQL Queries

Add below mentioned queries one after the other.

node_load1

node_load5

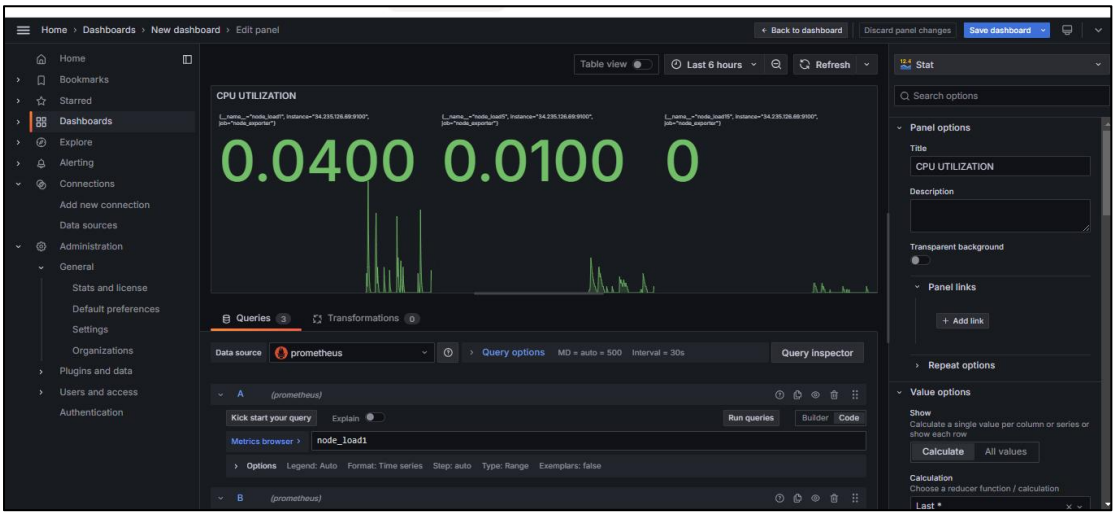
node_load15

- 1-Minute Load Average: node_load1
- 5-Minute Load Average: node_load5
- 15-Minute Load Average: node_load15

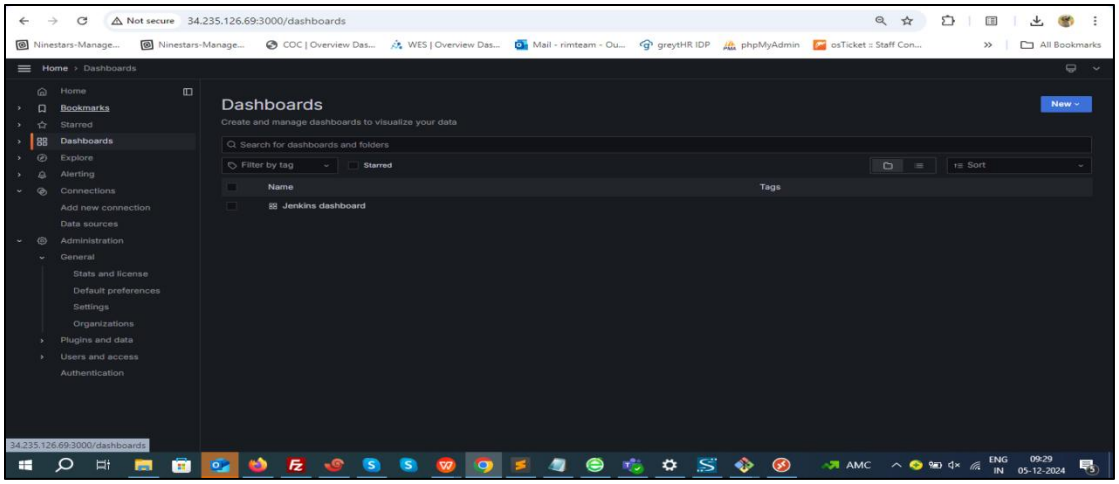
Visualization:

Select **Time Series** or **Gauge** based on your preference..

Title: Name it **"CPU Utilization"**.



Save the Dashboard as Jenkins dashboard.

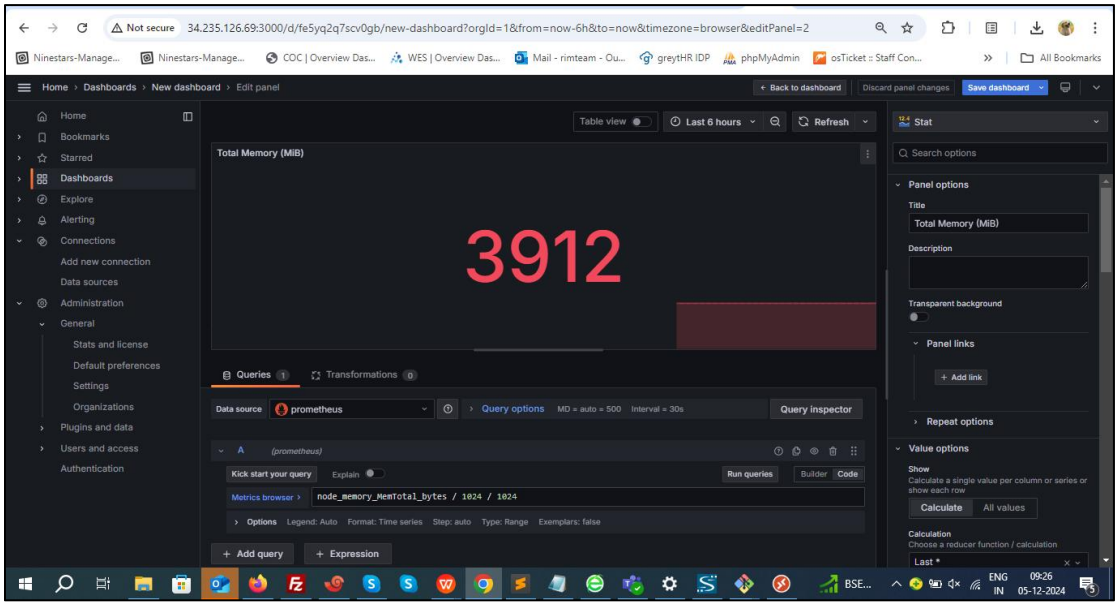


Panel 2: Memory Utilization

PromQL Queries for Memory Metrics

1. Total Memory (MiB):

node_memory_MemTotal_bytes / 1024 / 1024

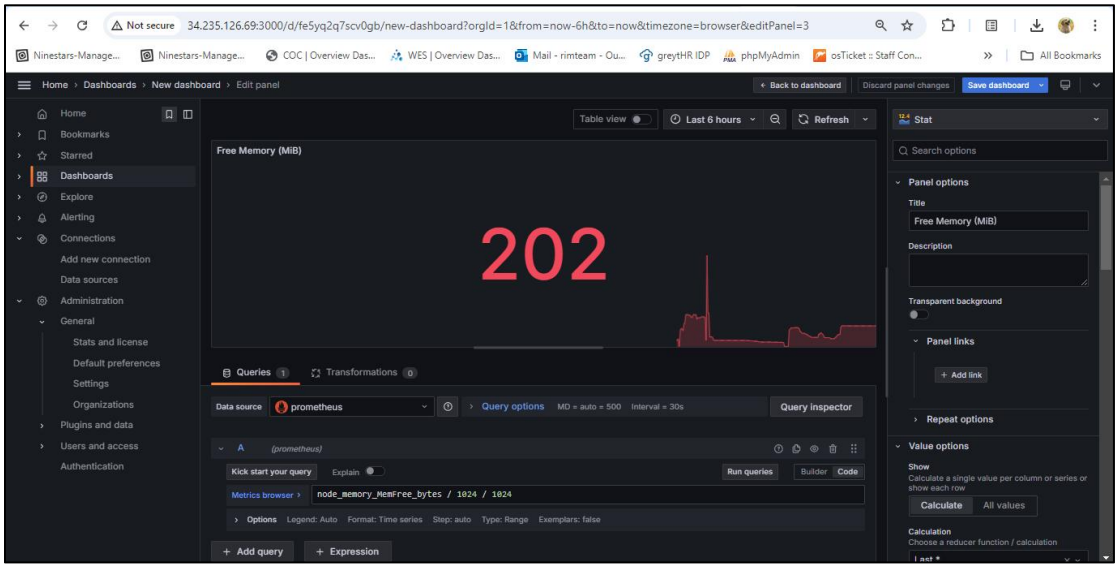


This gives the total memory in MiB.

2. Free Memory (MiB):

node_memory_MemFree_bytes / 1024 / 1024

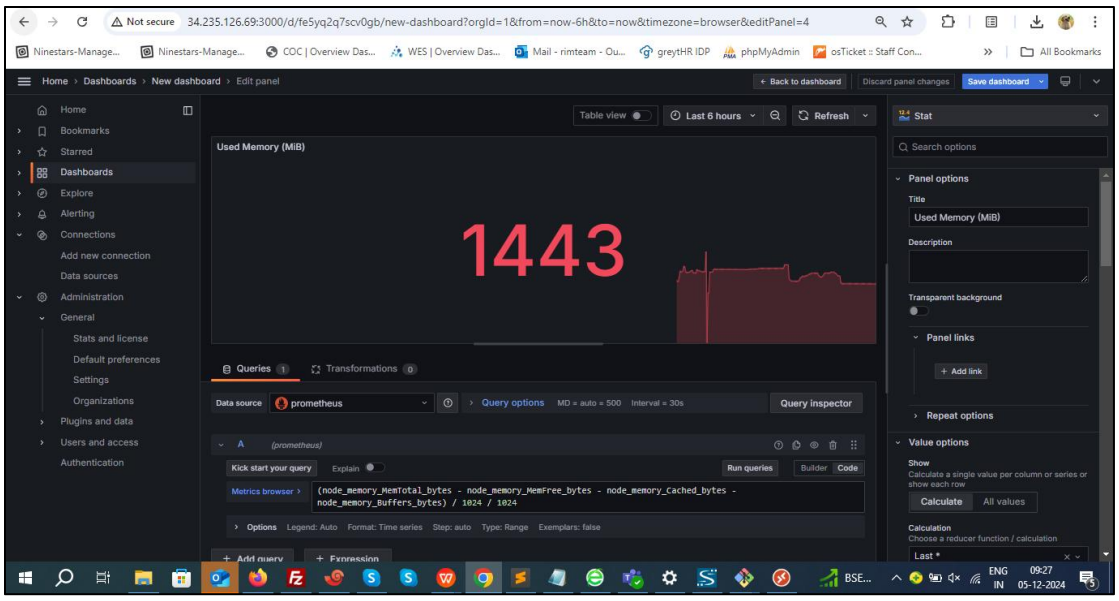
This gives the free memory in MiB.



3. Used Memory (MiB): Used memory can be calculated as:

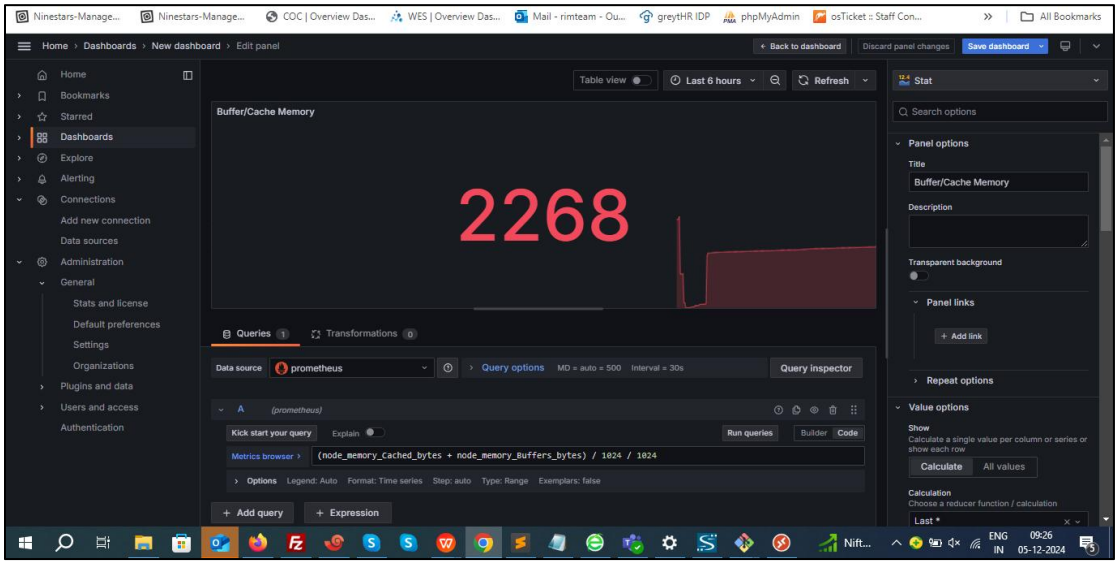
(node_memory_MemTotal_bytes - node_memory_MemFree_bytes - node_memory_Cached_bytes - node_memory_Buffers_bytes) / 1024 / 1024

This excludes the memory used for cache and buffers.



4. Buffer/Cache Memory (MiB):

(node_memory_Cached_bytes + node_memory_Buffers_bytes) / 1024 / 1024



Visualization:

Use **Time Series** or **Gauge**.

Title: Name it **according to Memory Utilization.**

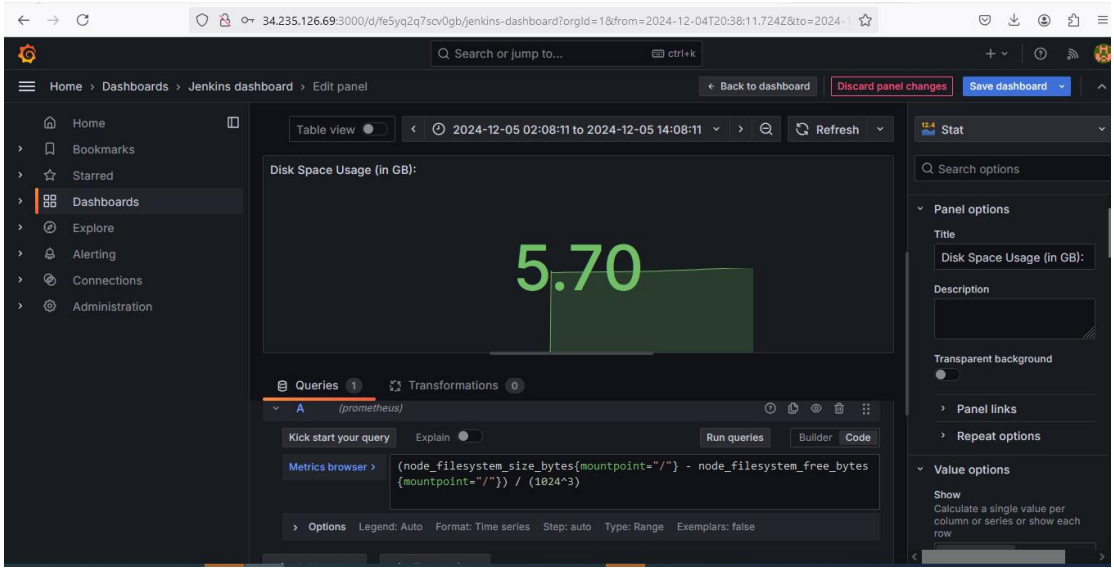
Panel 3: Disk Space Utilization

1. Disk Space Usage (in GB):

Query:

(node_filesystem_size_bytes{mountpoint="/"} - node_filesystem_free_bytes{mountpoint="/"}) / (1024^3)

Visualization: Gauge or Time series to display the disk space usage percentage.

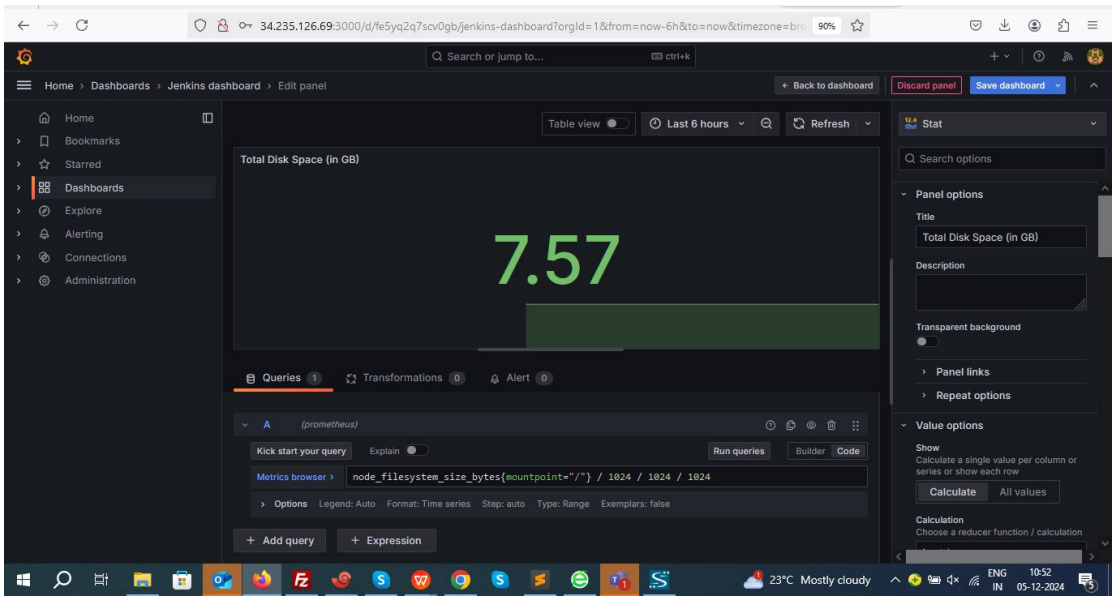


2. Total Disk Space (in GB):

Query:

node_filesystem_size_bytes{mountpoint="/"} / 1024 / 1024 / 1024

Visualization: Stat to show the total size.

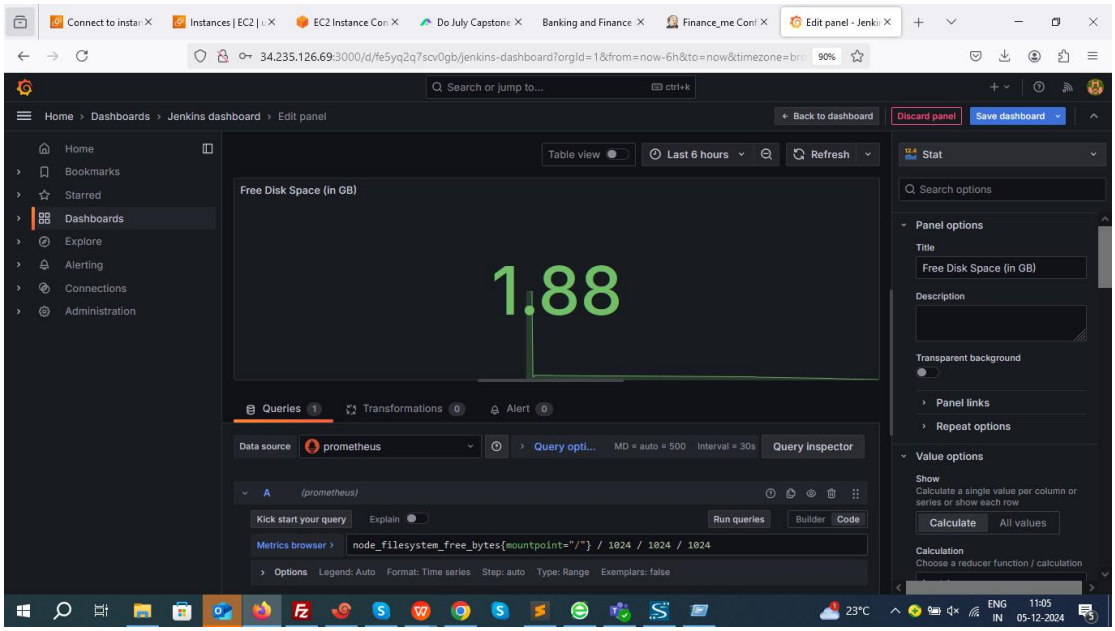


3.Free Disk Space (in GB):

Query:

node_filesystem_free_bytes{mountpoint="/"} / 1024 / 1024 / 1024

Visualization: Stat or Bar gauge to show the amount of free space.

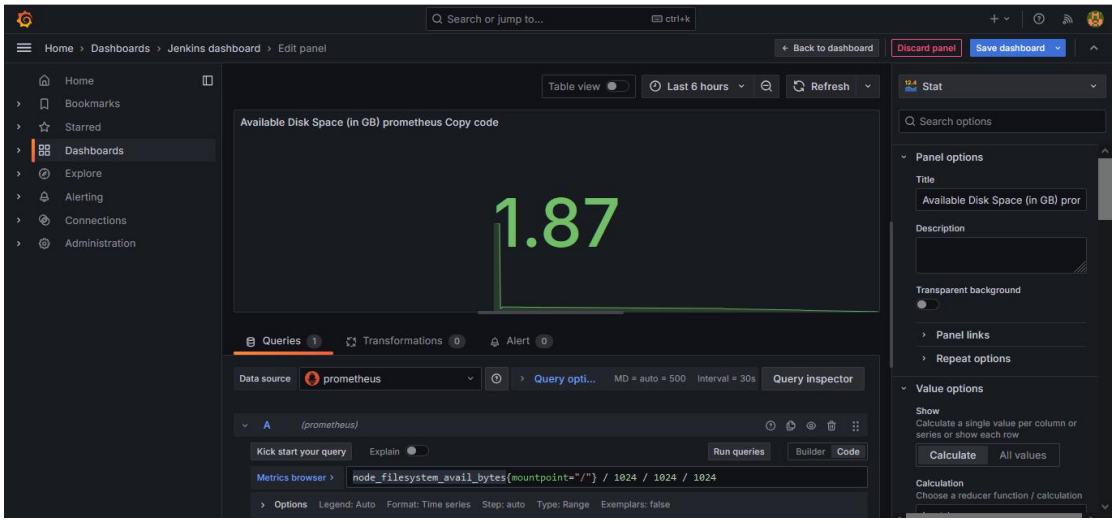


4. Available Disk Space(in GB):

Query:

node_filesystem_avail_bytes{mountpoint="/"} / 1024 / 1024 / 1024

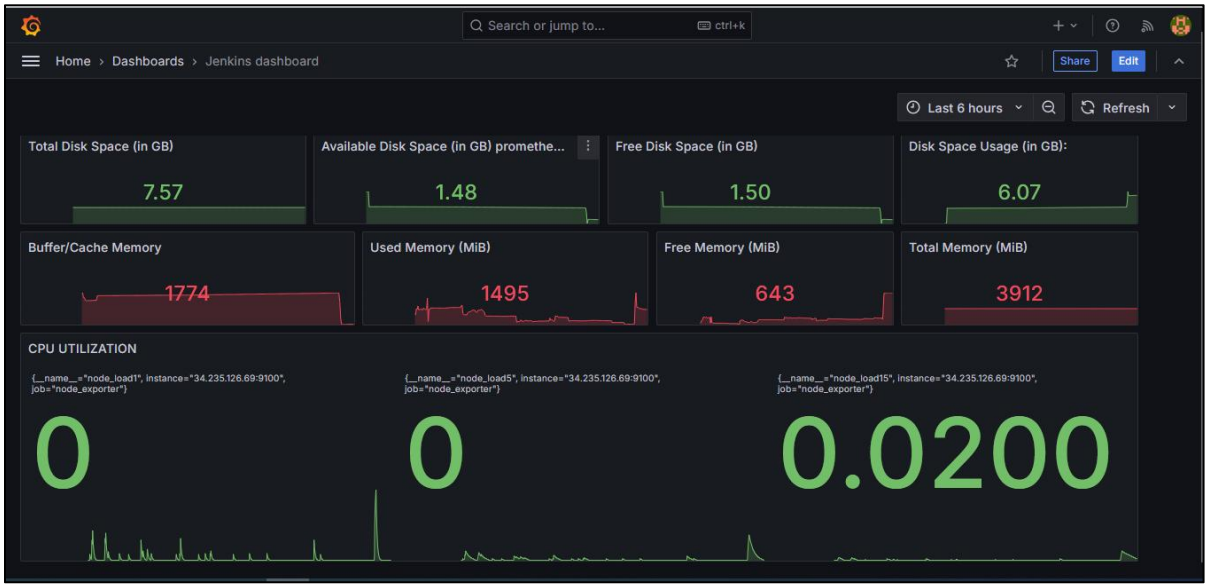
Visualization: Stat or Gauge to show available space for users.



Comparing manually from server data

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/root	7.6G	5.7G	1.9G	76%	/

Customize the visualizations and titles accordingly.



CONCLUSION:

The application is successfully deployed and theinfrastructure is now fully automated and continuously monitored.