CAPSTONE PROJECT

Banking and Finance Domain

SUBMITTED BY: SALMAN SADIQ

DATE:05/03/2024

Banking and Finance Domain

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the code should be checked out, compiled, tested, packaged and containerized. A new test-server should be provisioned using terraform and should be automatically configured using Ansible with all the required software's and as soon as the server is available, the application must be deployed to the test-server automatically.

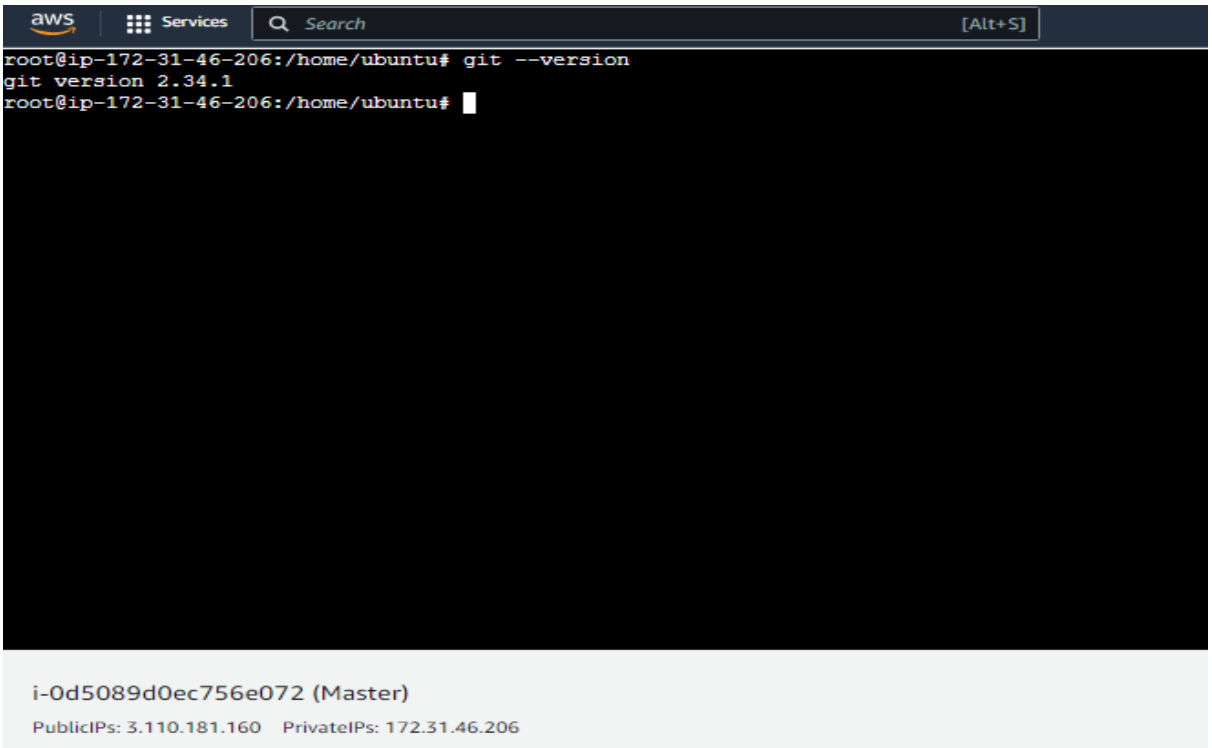
The deployment should then be tested using a test automation tool, and if the build is
Successful, Prod server must be configured with all the software it should be pushed to the prod server. All this should happen automatically and should be triggered from a push to the GitHub master branch. Continuous monitoring server must be configured to monitor the test as well as prod server using Prometheus and Grafana should be configured to display a dashboard with following metrics.

1. CPU utilization
2. Disk Space Utilization
3. Total Available Memory

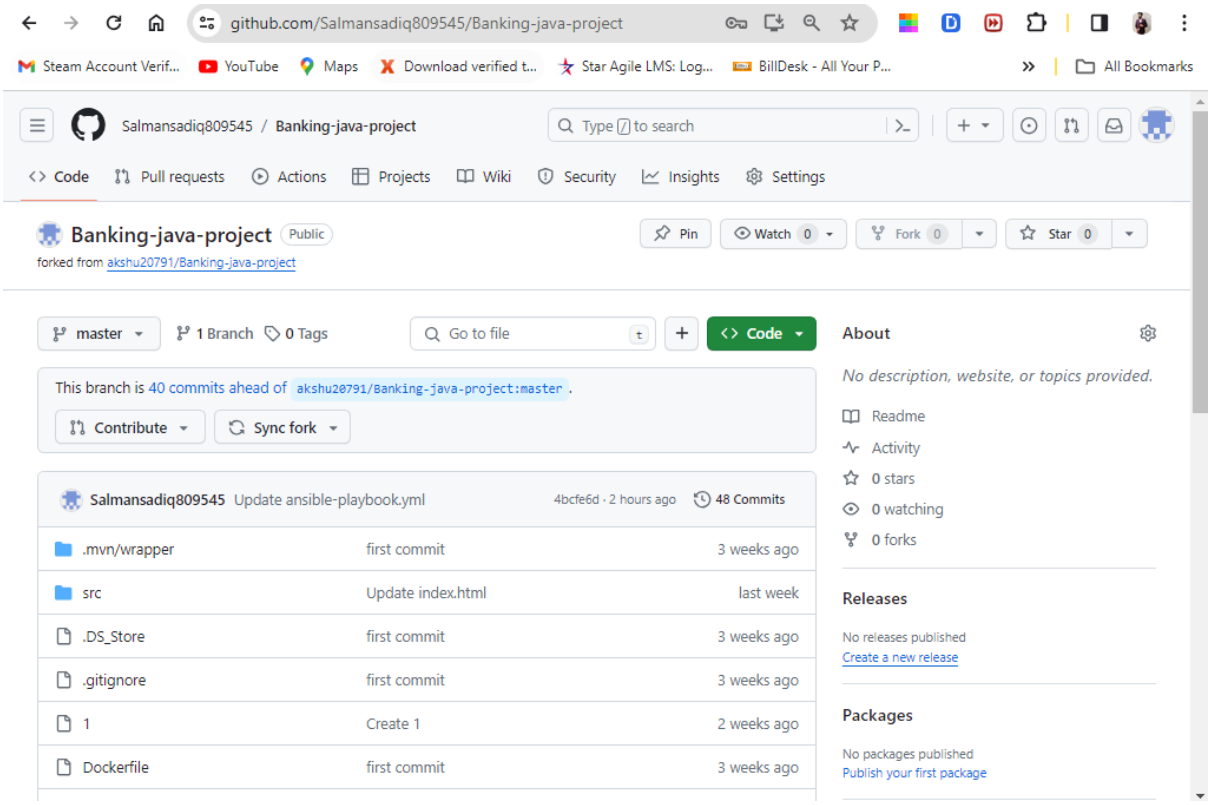
Later, you need to implement Continuous Integration & Continuous Deployment using following tools:

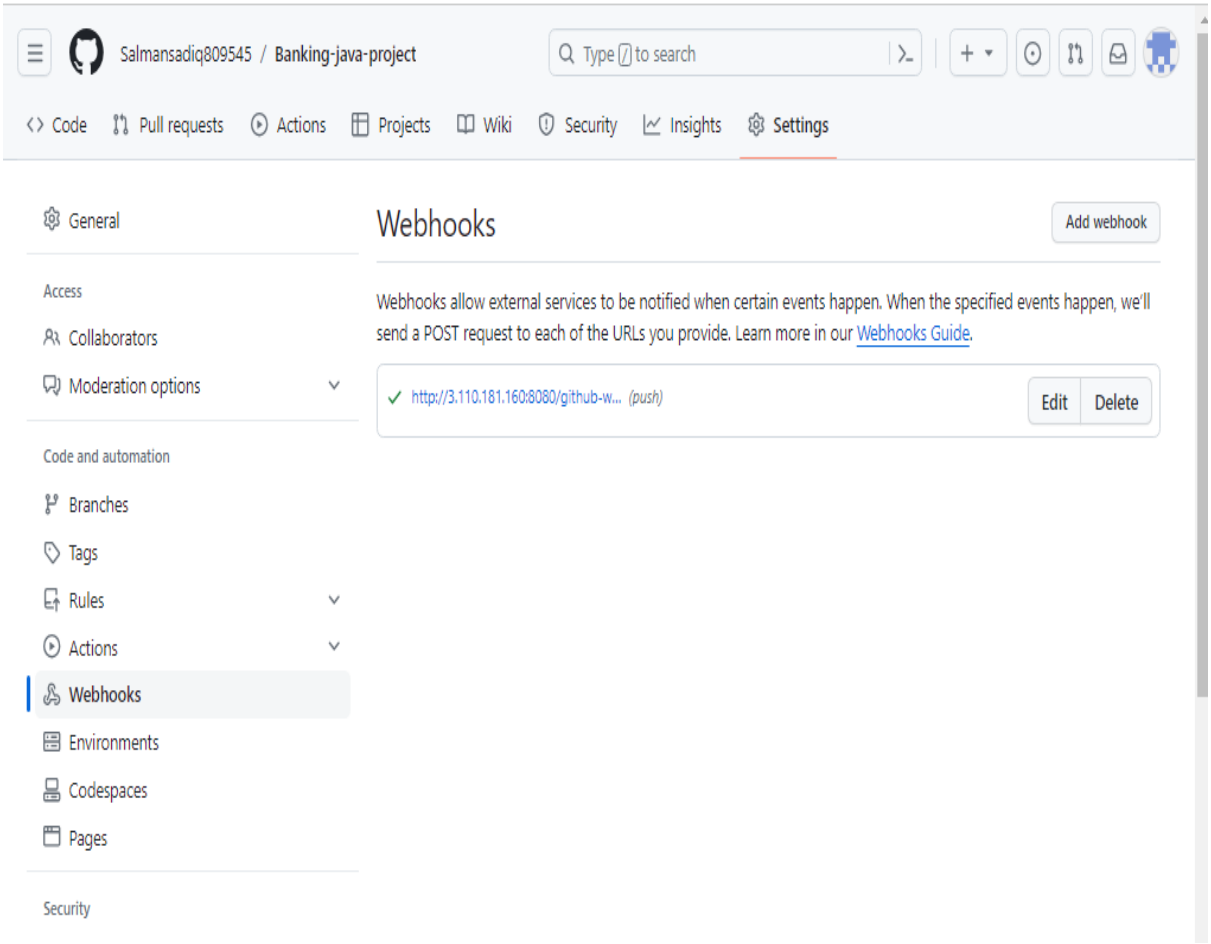
- Git - For version control for tracking changes in the code files
- Maven – For Continuous Build
- Jenkins - For continuous integration and continuous deployment
- Docker - For deploying containerized applications
- Ansible - Configuration management tools
- Terraform - For creation of infrastructure.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

A) GIT-For Version Control



Git has been installed in master





This is the git repository in which code is kept. And the git-webhook has been configured for CI/CD pipeline

B) Maven- For Continuous Build.

```
aws   Services   Search                                   [Alt+S]
root@ip-172-31-46-206:/home/ubuntu# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.22, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.5.0-1014-aws", arch: "amd64", family: "unix"
root@ip-172-31-46-206:/home/ubuntu#
```

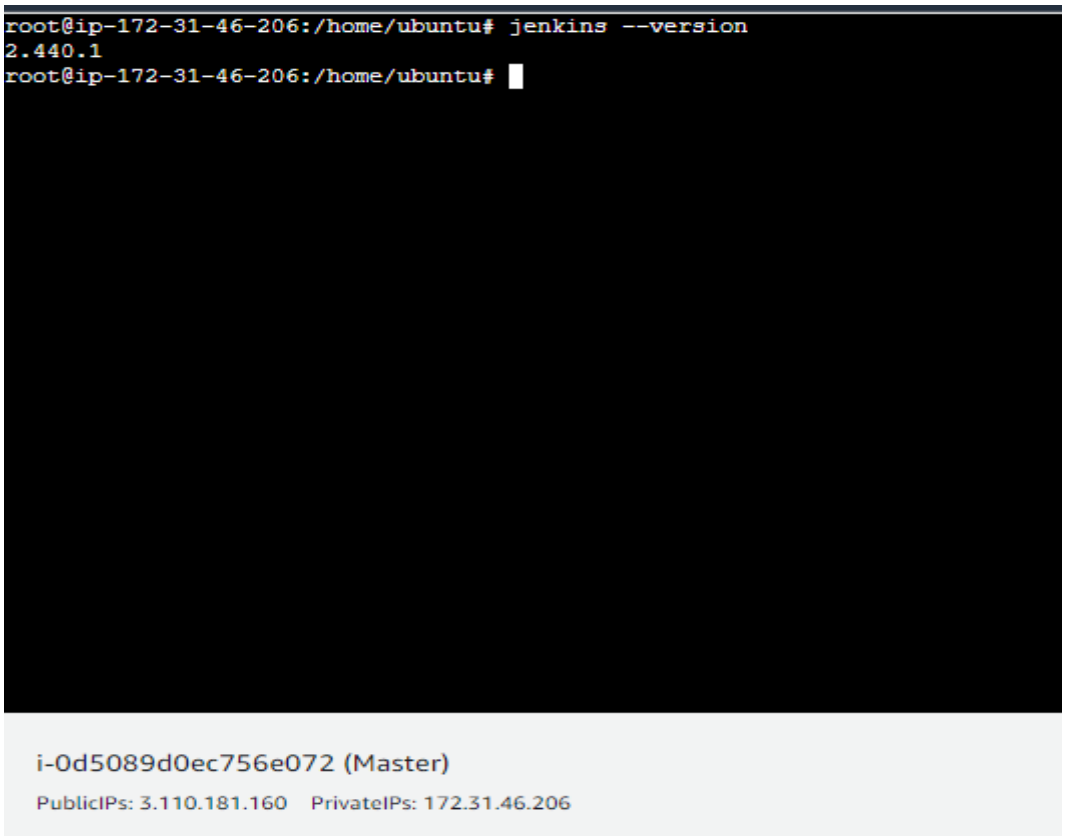
i-0d5089d0ec756e072 (Master)

PublicIPs: 3.110.181.160 PrivateIPs: 172.31.46.206

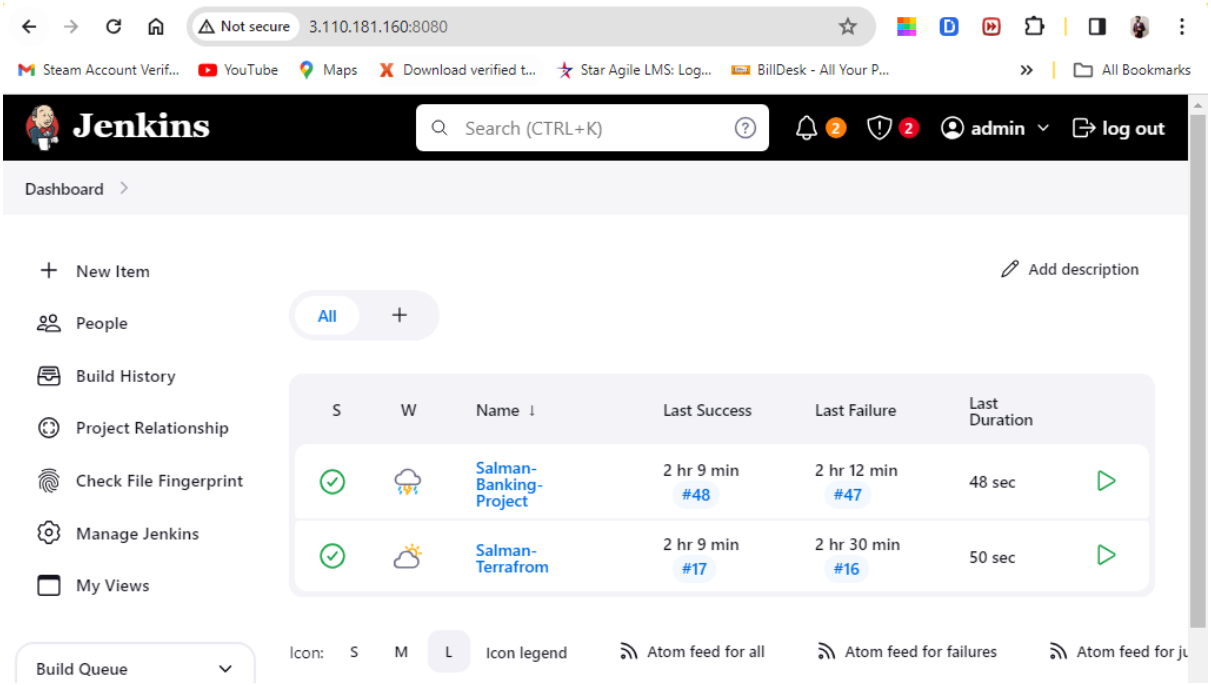
```
}
stage('Maven Compile')
{
    sh 'mvn compile'
}


stage('Maven Package')
{
    sh 'mvn clean package'
}
```

Here the maven is installed and maven command have been listed in above figure

C) Jenkins – For Continuous Integration and Continuous Deployment



Here we can see Jenkins have been installed in master



Here we can see Jenkins Pipeline for Test and Production Server

Dashboard > Salman-Banking-Project > Configuration

Configure

General

Advanced Project Options

Pipeline

Pipeline

Definition

Pipeline script

Script

```
node{
    stage('Git Code Checkout')
    {
        git 'https://github.com/Salmansadiq809545/Banking-java-project'
    }
    stage('Maven Compile')
    {
        sh 'mvn compile'
    }


    stage('Maven Package')
    {
        sh 'mvn clean package'
    }
    stage('Docker Image')
    {
```

Here we can see the pipeline for getting the code from git and converting into package using maven commands and containerizing project

Dashboard > Salman-Terrafrom > Configuration

Configure

General

Advanced Project Options

Pipeline

Definition

Pipeline script

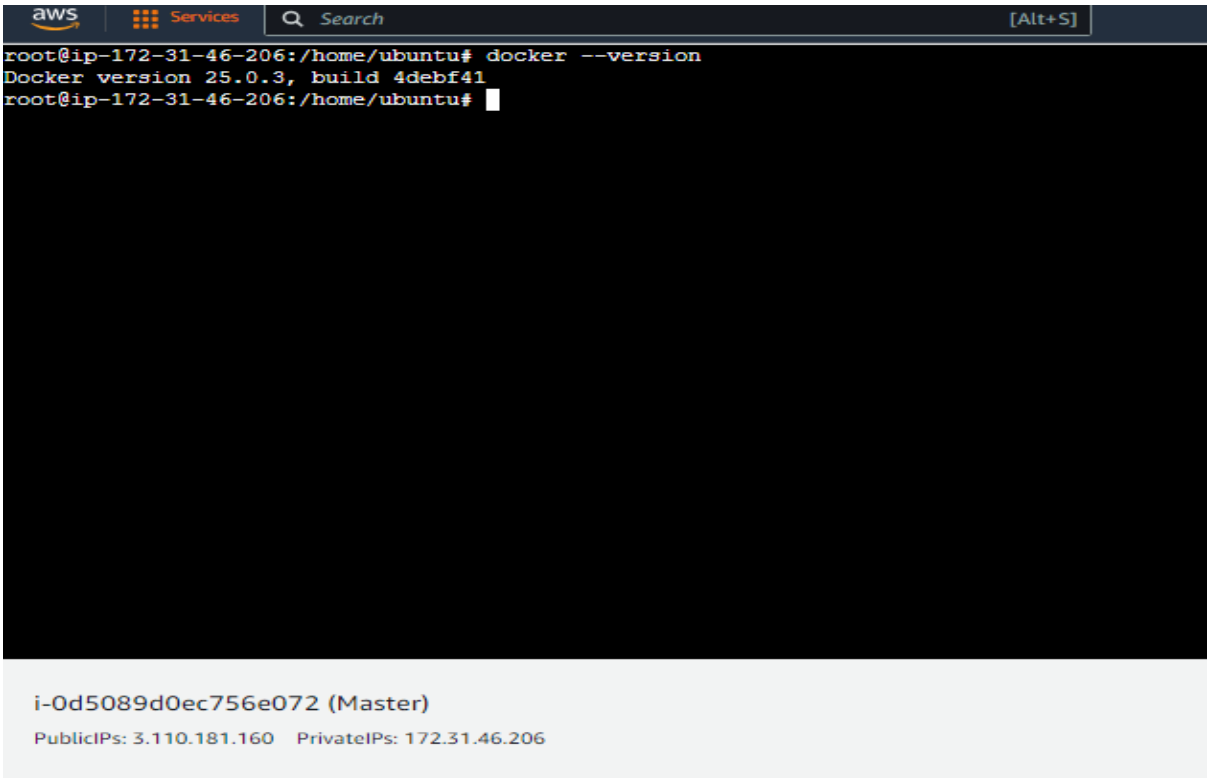
Script

```
pipeline{
    agent any

    stages{
        stage('git')
        {
            steps{
                git 'https://github.com/Salmansadiq809545/Banking-java-project'
            }
        }
        stage('Terafrom init')
        {
            steps{
                sh 'terraform init'
            }
        }
        stage('Terafrom plan')
        {
```

Use Groovy Sandbox

Here we can see Pipeline for terraform infrastructure creation

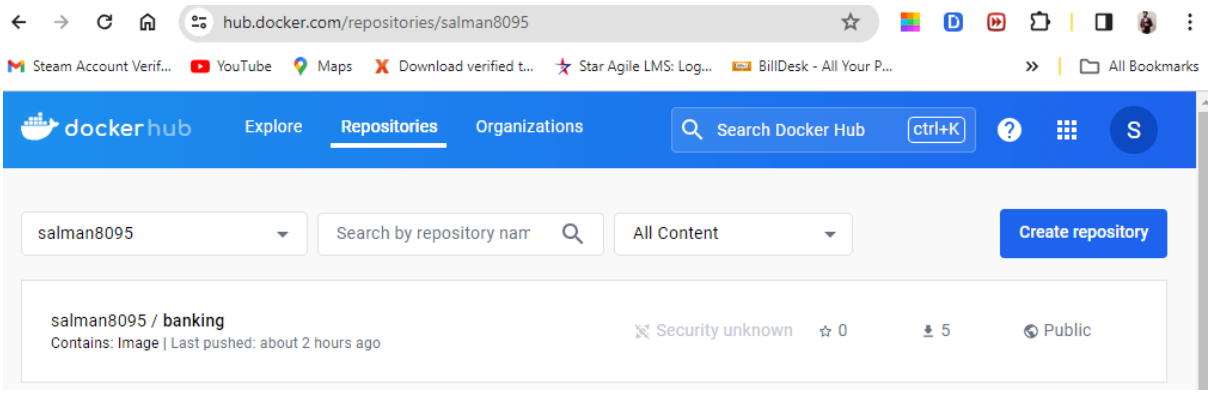
C) Docker – For Deploying containerized application



Docker is installed in Master machine

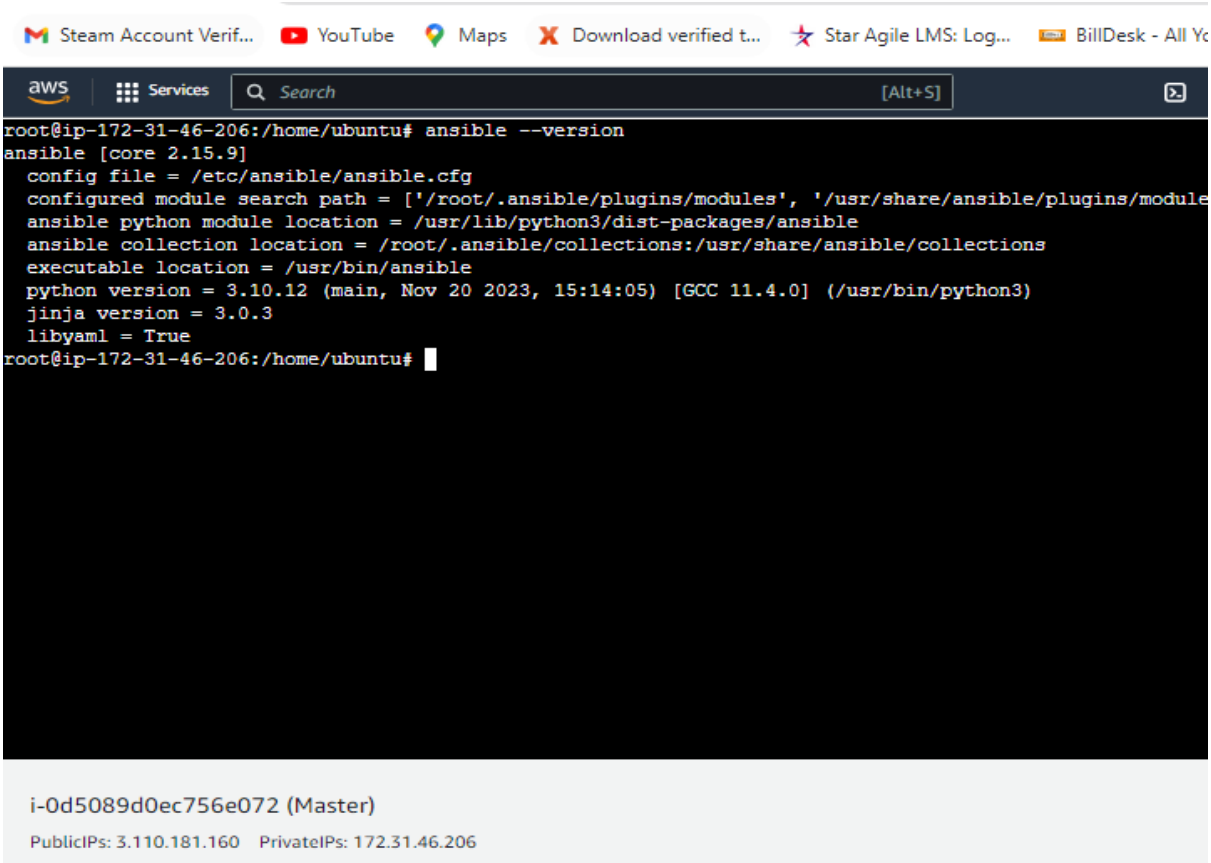
```
    sh 'docker rmi -f salman8095/banking:v1'

     sh 'docker build -t salman8095/banking:v1 .'
  }
   stage('Docker push')
  {
      withCredentials([usernamePassword(credentialsId: 'dockerhub-pwd', passwordVari
                sh "echo $PASS | docker login -u $USER --password-stdin"
                sh 'docker push salman8095/banking:v1'
            }
  }
```

Above are the commands for creating docker image and docker container



The docker image has been pushed in docker hub

D) Ansible – For Configuration Management Tools



Ansible has been installed in Master machine

```
root@ip-172-31-46-206:/etc/ansible# cat hosts
[demo]
35.154.27.87
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers:

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:

## [webservers]
```

Ansible has been configured for its hosts with the following Ip address 35.154.27.57 as demo group

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

  - name : Install Docker
    command : sudo apt-get install -y docker.io
    become : yes
    become_user : root

  - name : Start Docker Service
    command : sudo systemctl start docker
    become : yes
    become_user : root

  - name: STOP CONTAINER
    command: sudo docker stop C01

  - name: removeContainer
    command: sudo docker rm C01

  - name: Deploy Docker Container
    command: docker run -itd -p 8083:8091 --name C01 salman8095/banking:v1
```

Simple Ansible-playbook has been written for deploying the application as container in port 8083

```
        }
    }
    stage('Ansible')
    {
      ansiblePlaybook become: true, credentialsId: 'ansible', disableHostKeyChecking: true, installation: 'ans
    }
```

se Groovy Sandbox ?

Above is the Jenkins code for exececuting ansible-playbook.yml file

E) Terraform – For creation of infrastructure

```
root@ip-172-31-46-206:/home/ubuntu# terraform --version
Terraform v1.7.4
on linux_amd64
root@ip-172-31-46-206:/home/ubuntu#
```

Terraform has been installed in Master machine

```
# Creating ec2 Instance
resource "aws_instance" "prod_server8095" {
  ami           = "ami-03bb6d83c60fc5f7c"
  instance_type = "t2.micro"
  availability_zone = "ap-south-1b"
  key_name = "Tom"
  network_interface {
  device_index = 0
  network_interface_id = aws_network_interface.proj-nt.id
}
user_data = <<-EOF
#!/bin/bash
    sudo apt-get update -y
    sudo apt-get update -y
    sudo apt-get install docker.io -y
    sudo systemctl enable docker
    # sudo docker stop C01
    # sudo docker rm C01
    sudo docker run -itd -p 8084:8091 --name C01 salman8095/banking:v1
    sudo docker start $(docker ps -aq)
  EOF
  tags = {
    Name = "Salman-BankProd-Server"
  }
}
```

Above code shows the terraform infrastructure creation of ec2 instance

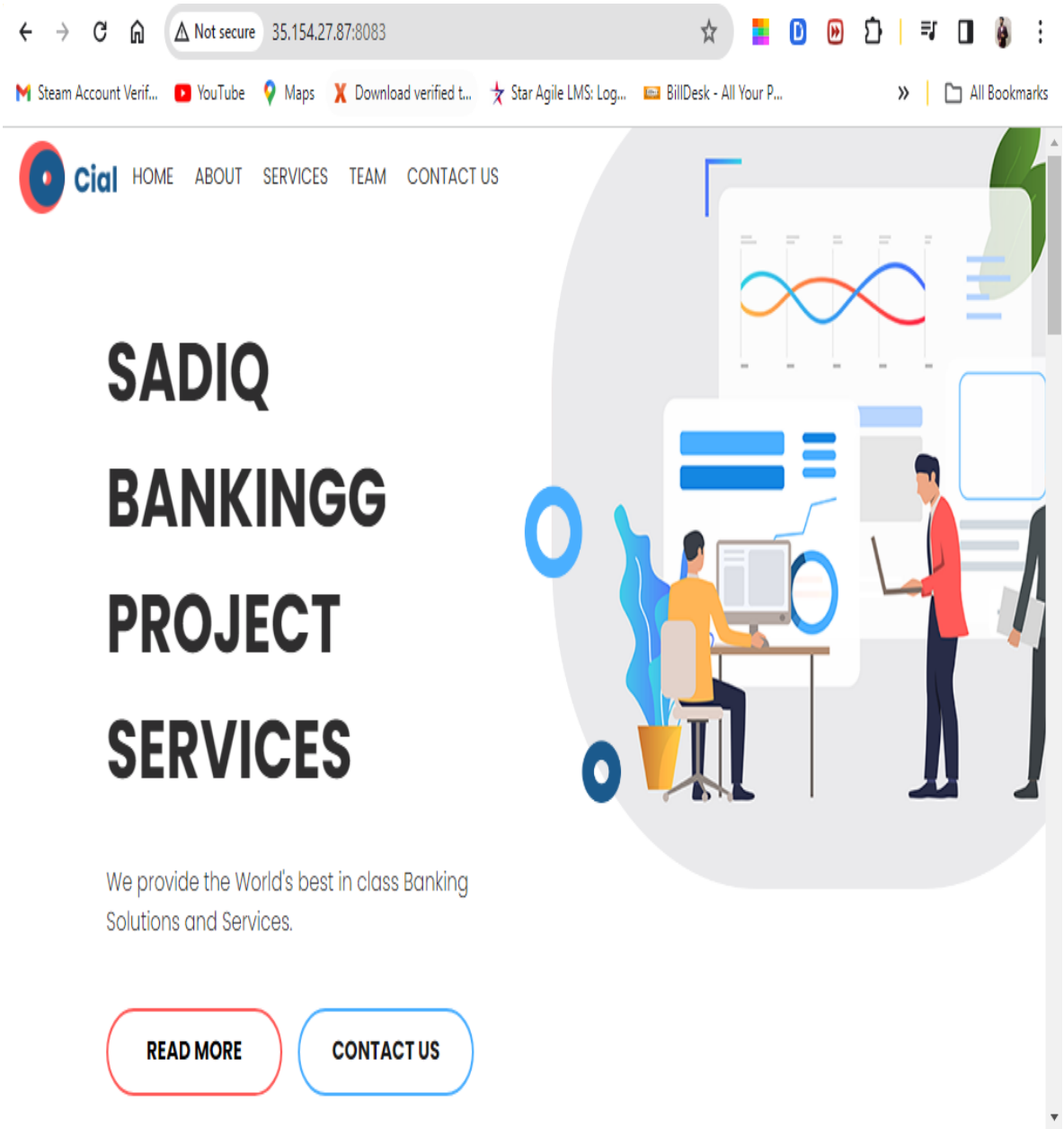
Script ?

```
pipeline{
    agent any

    stages{
        stage('git')
        {
            steps{
                git 'https://github.com/Salmansadiq809545/Banking-java-projec
            }
        }
        stage('Terafrom init')
        {
            steps{
                sh 'terraform init'
            }
        }
        stage('Terafrom plan')
```

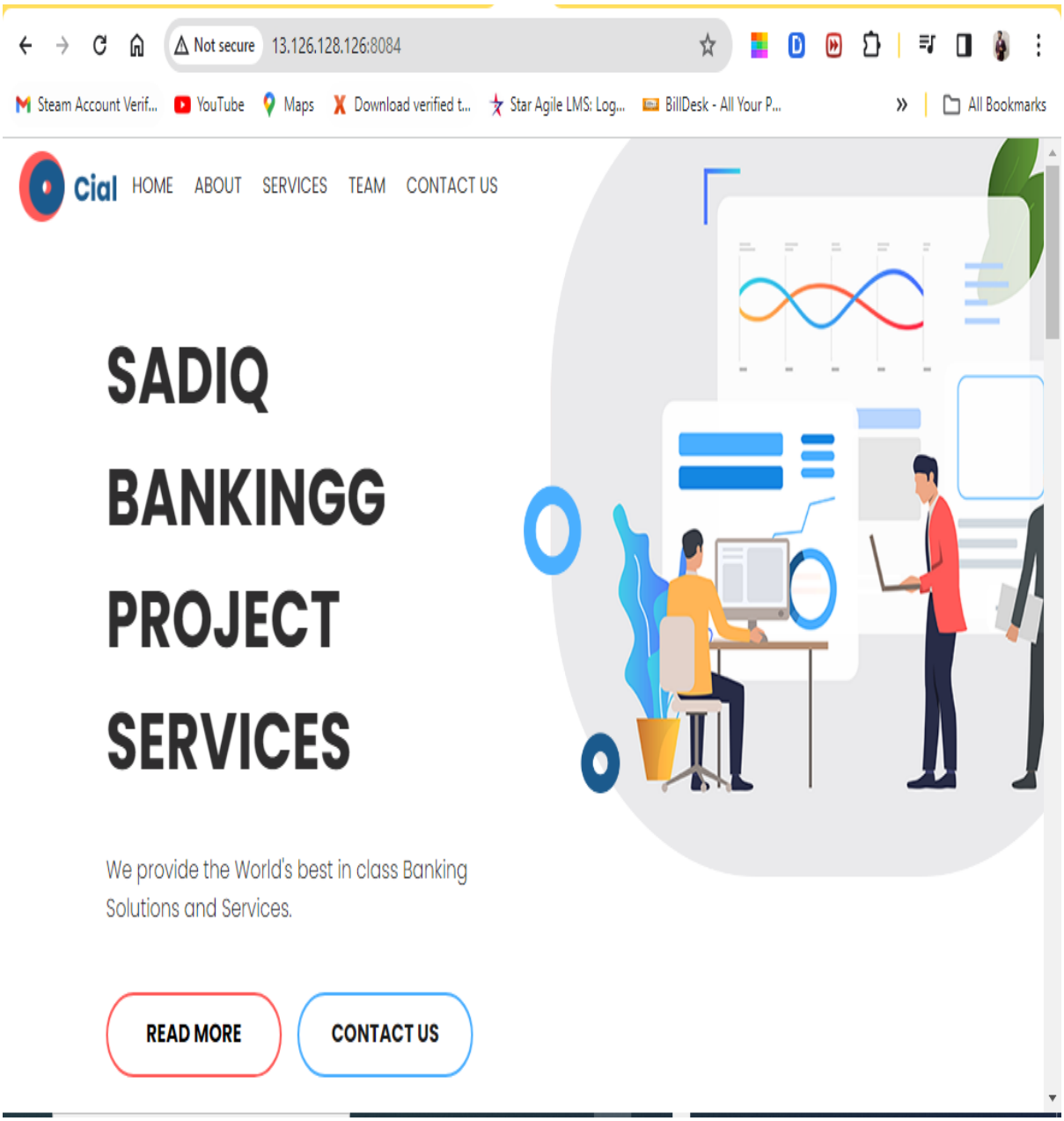
Above code shows the Jenkins pipeline for initializing planning and apply phase of terraform

Salman-BankProd-Server	i-0077481ac012c01d2	Running	t2.micro	2/2 checks passed	View

This is the Ec2 machine created from Terraform code

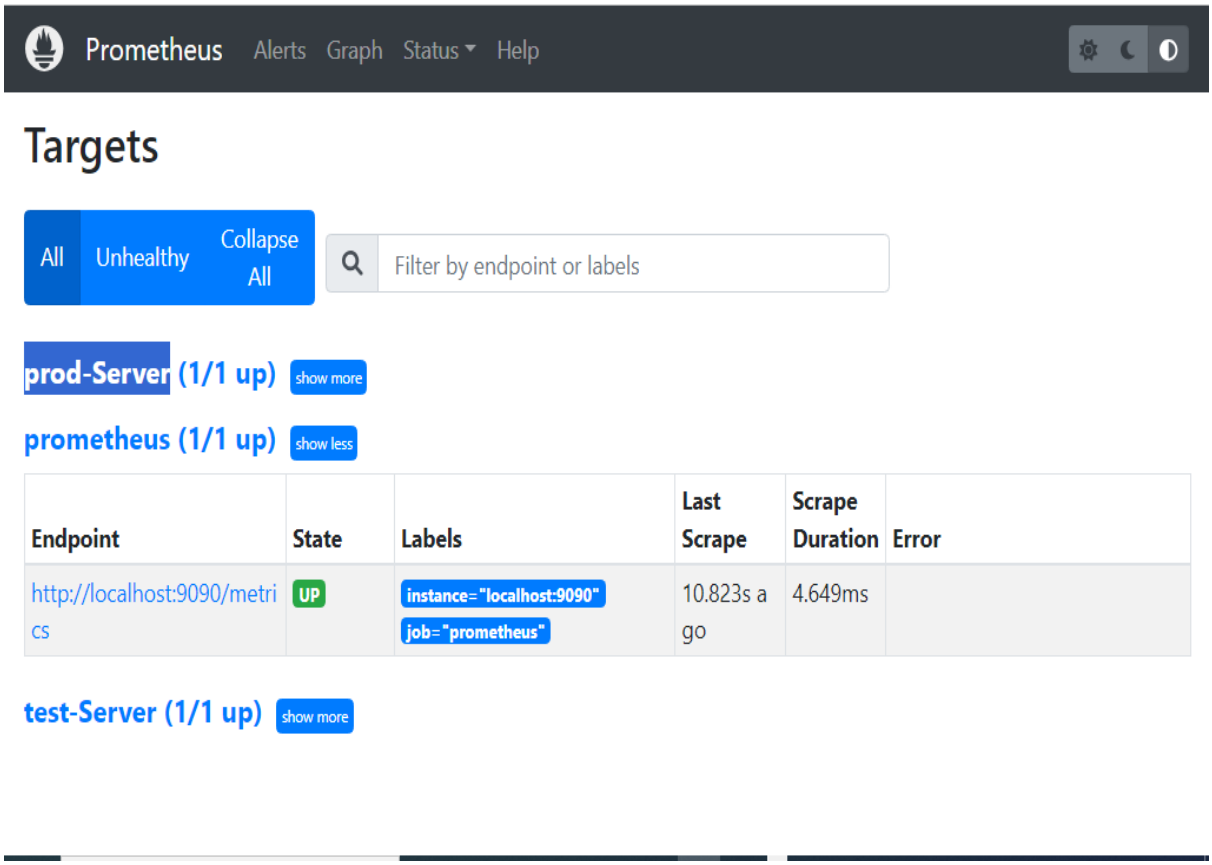


Above is the Container deployed using Ansible-playbook at port 8083



Above is the container deployed using terraform at port 8084

F) Prometheus and Grafana - For Automated Monitoring and report –virtualization

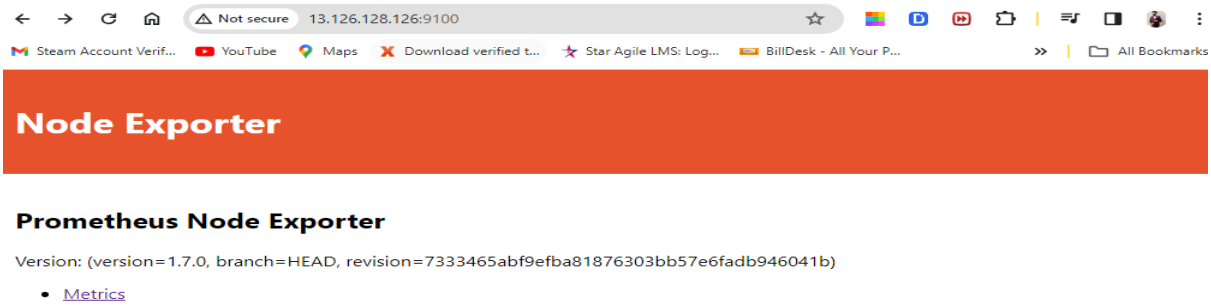


The images shows that Prometheus as been installed and is configured for test server and prod server

```
# 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: "test-Server"
    static_configs:
      - targets: ["35.154.27.87:9100"]

  - job_name: "prod-Server"
    static_configs:
      - targets: ["13.126.128.126:9100"]

"prometheus.yml" 39L, 1116B
```

Above images shows the jobs created for test and prod server

b) Node-Exported



The above images shows the node port configured at production Server

Node Exporter

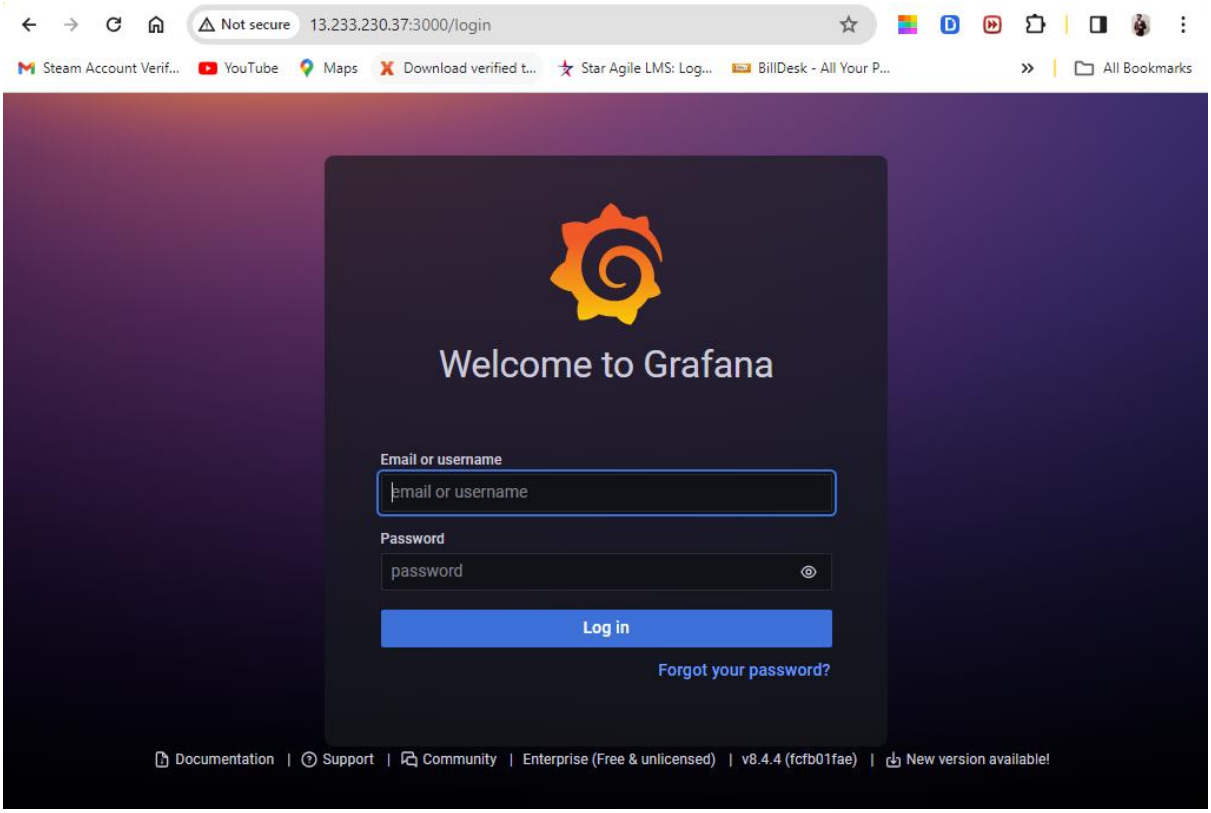
Prometheus Node Exporter

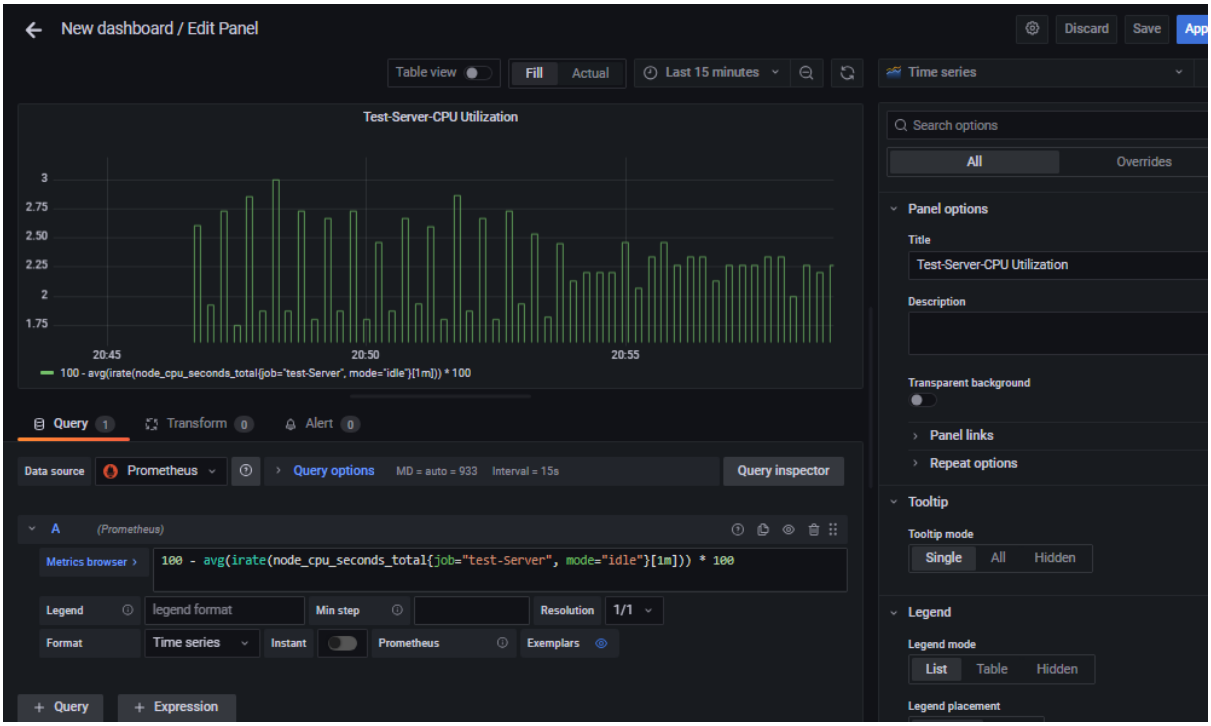
Version: (version=1.7.0, branch=HEAD, revision=7333465abf9efba81876303bb57e6fadb946041b)

- Metrics

Above image shows node-port exporter configured for test server

C) Grafana





The grafana has been installed and configured for monitoring test and production server

Here we can see grafana has been configured to monitor both the test and production server with metrics for

1)Cpu Utlization

2) Disk Space Utlization

3) Memory Utlization

THANK YOU