

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

```
aws Services Search [Alt+S]
root@ip-172-31-46-206:/home/ubuntu# git --version
git version 2.34.1
root@ip-172-31-46-206:/home/ubuntu#
```

i-Od5089d0ec756e072 (Master)
PublicIPs: 3.110.181.160 PrivateIPs: 172.31.46.206

Git has been installed in master

github.com/Salmansadiq809545/Banking-java-project

Steam Account Verif... YouTube Maps Download verified t... Star Agile LMS: Log... BillDesk - All Your P... All Bookmarks

Salmansadiq809545 / Banking-java-project

Type to search

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Banking-java-project Public

forked from akshu20791/Banking-java-project

Pin Watch 0 Fork 0 Star 0

master 1 Branch 0 Tags

Go to file

+ <> Code

About

No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

This branch is 40 commits ahead of akshu20791/Banking-java-project:master.

Contribute Sync fork

Salmansadiq809545	Update ansible-playbook.yml	4bcfe6d · 2 hours ago	48 Commits
mvnw/wrapper	first commit	3 weeks ago	
src	Update index.html	last week	
.DS_Store	first commit	3 weeks ago	
.gitignore	first commit	3 weeks ago	
1	Create 1	2 weeks ago	
Dockerfile	first commit	3 weeks ago	

The screenshot shows the GitHub repository settings page for 'Banking-java-project' by user 'Salmansadiq809545'. The 'Settings' tab is selected in the top navigation bar. On the left sidebar, the 'Webhooks' option is highlighted under the 'Code and automation' section. The main content area is titled 'Webhooks' and includes an 'Add webhook' button. Below this, a text block explains that webhooks allow external services to be notified of events and sends a POST request to the provided URLs. A single webhook is listed with a green checkmark, the URL 'http://3.110.181.160:8080/github-w...', and the event type '(push)'. 'Edit' and 'Delete' buttons are visible for this webhook.

Salmandadiq809545 / Banking-java-project

Type to search

Code Pull requests Actions Projects Wiki Security Insights Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Webhooks

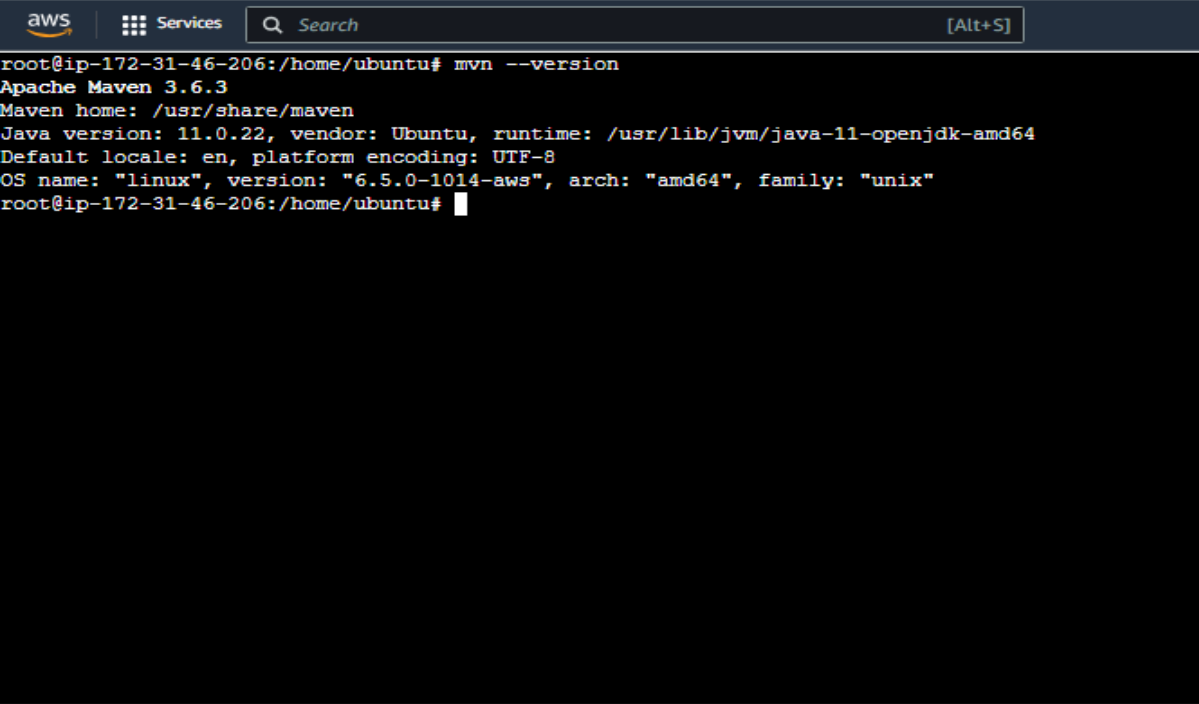
Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ http://3.110.181.160:8080/github-w... (push) Edit Delete

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-Od5089d0ec756e072 (Master)
PublicIPs: 3.110.181.160 PrivateIPs: 172.31.46.206

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

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

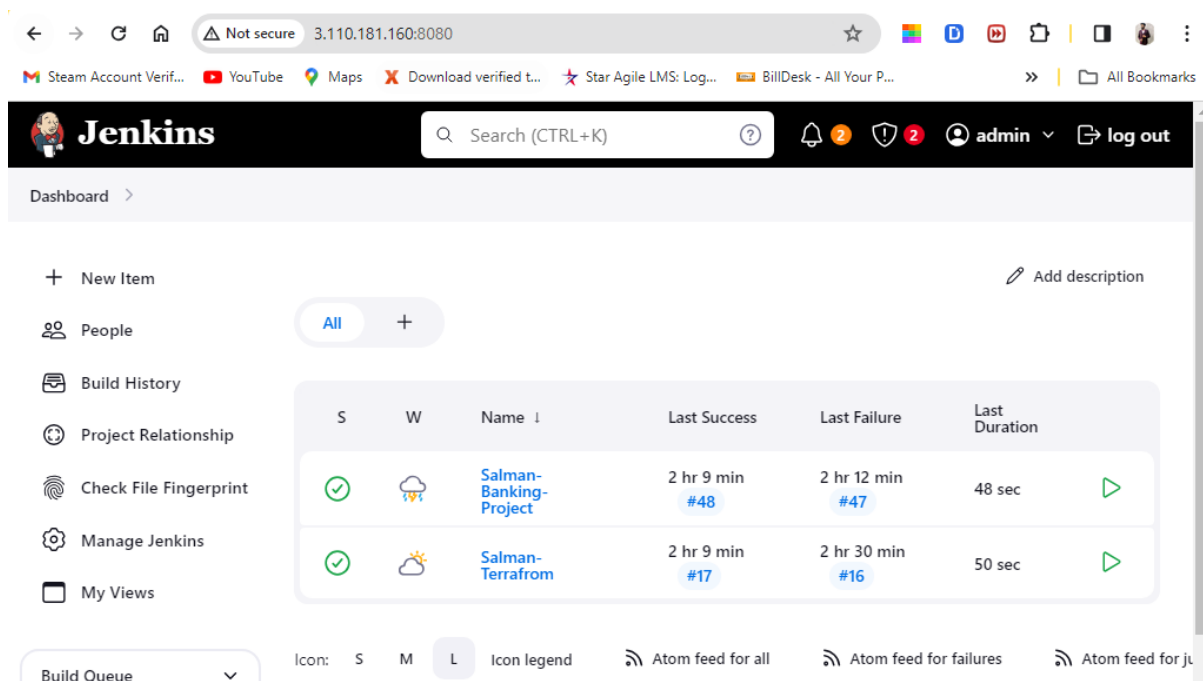
Here the maven is installed and maven command have been listed in above figure

C) Jenkins – For Continuous Integration and Continuous Deployment

```
root@ip-172-31-46-206:/home/ubuntu# jenkins --version
2.440.1
root@ip-172-31-46-206:/home/ubuntu#
```

i-Od5089d0ec756e072 (Master)
PublicIPs: 3.110.181.160 PrivateIPs: 172.31.46.206

Here we can see Jenkins have been installed in master



Here we can see Jenkins Pipeline for Test and Production Server

Configure



General



Advanced Project Options



Pipeline

Pipeline

Definition

Pipeline script

Script ?

```
1 node{
2   stage('Git Code Checkout')
3   {
4     git 'https://github.com/Salmansadiq809545/Banking-java-project'
5   }
6   stage('Maven Compile')
7   {
8     sh 'mvn compile'
9   }
10
11
12   stage('Maven Package')
13   {
14     sh 'mvn clean package'
15   }
16   stage('Docker Image')
17 }
```

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

Configure



General



Advanced Project Options



Pipeline

Definition

Pipeline script

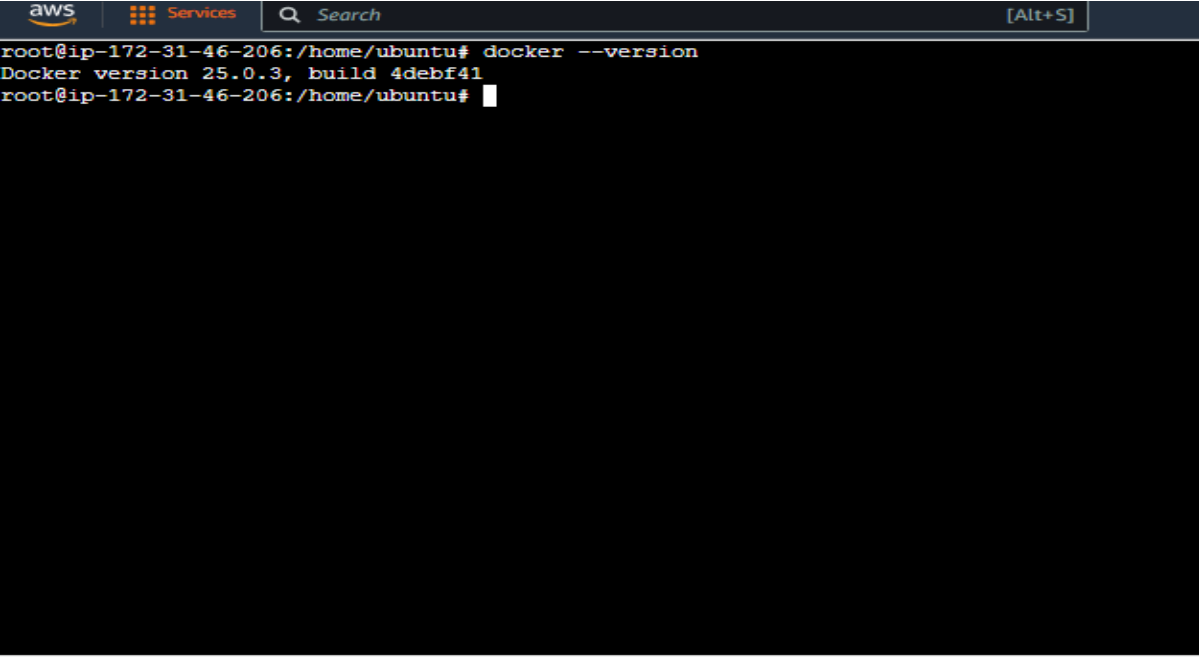
Script ?

```
1 pipeline{
2   agent any
3
4   stages{
5     stage('git')
6     {
7       steps{
8         git 'https://github.com/Salmansadiq809545/Banking-java-project'
9       }
10    }
11    stage('Terafrom init')
12    {
13      steps{
14        sh 'terraform init'
15      }
16    }
17    stage('Terafrom plan')
18  }
```

☒ Use Groovy Sandbox ?

Here we can see Pipeline for terraform infrastructure creation

C) Docker – For Deploying containerized application



The screenshot shows an AWS console terminal window. The top bar includes the AWS logo, a 'Services' menu, a search bar, and a '[Alt+S]' shortcut. The terminal output shows the command `docker --version` being executed, resulting in the output: `Docker version 25.0.3, build 4debf41`. The prompt indicates the user is root at ip-172-31-46-206 in the /home/ubuntu directory. Below the terminal window, the instance details are shown: `i-Od5089d0ec756e072 (Master)` with Public IPs: 3.110.181.160 and Private IPs: 172.31.46.206.

```
aws Services Search [Alt+S]
root@ip-172-31-46-206:/home/ubuntu# docker --version
Docker version 25.0.3, build 4debf41
root@ip-172-31-46-206:/home/ubuntu#
```

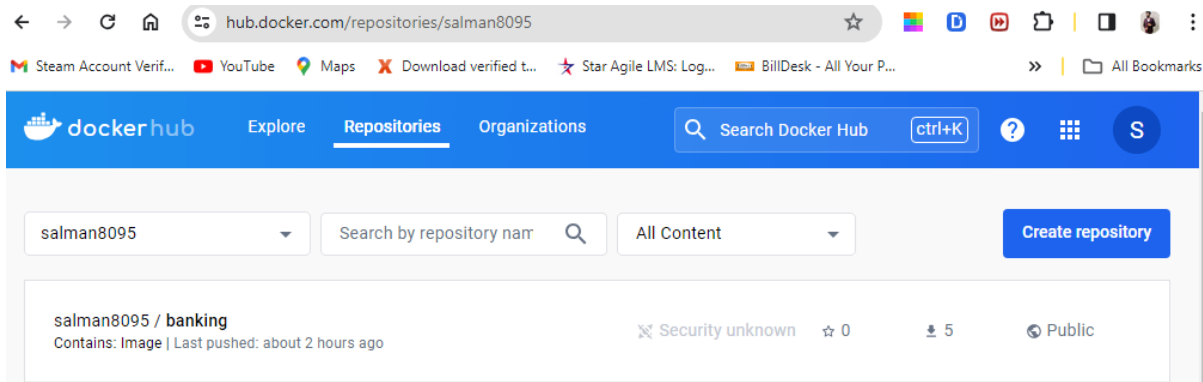
i-Od5089d0ec756e072 (Master)
PublicIPs: 3.110.181.160 PrivateIPs: 172.31.46.206

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', passwordVar:
    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

A screenshot of an AWS terminal window. The terminal shows the command 'ansible --version' being executed. The output displays various configuration details for Ansible, including the config file, module search paths, Python version (3.10.12), Jinja version (3.0.3), and libyaml status. The terminal window has an AWS logo and 'Services' in the top left, and a search bar in the top right. Below the terminal output, there's a footer showing the instance ID 'i-0d5089d0ec756e072 (Master)' and IP addresses: 'PublicIPs: 3.110.181.160' and 'PrivateIPs: 172.31.46.206'.

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

```

1  - name : Configure Docker on EC2 Instances
2    hosts : all
3    become: true
4    connection : ssh
5    tasks :
6      - name: updating apt
7        command : sudo apt-get update
8
9      - name : Install Docker
10       command : sudo apt-get install -y docker.io
11       become : yes
12       become_user : root
13
14     - name : Start Docker Service
15       command : sudo systemctl start docker
16       become : yes
17       become_user : root
18
19     - name: STOP CONTAINER
20       command: sudo docker stop C01
21
22     - name: removeContainer
23       command: sudo docker rm C01
24
25     - name: Deploy Docker Container
26       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

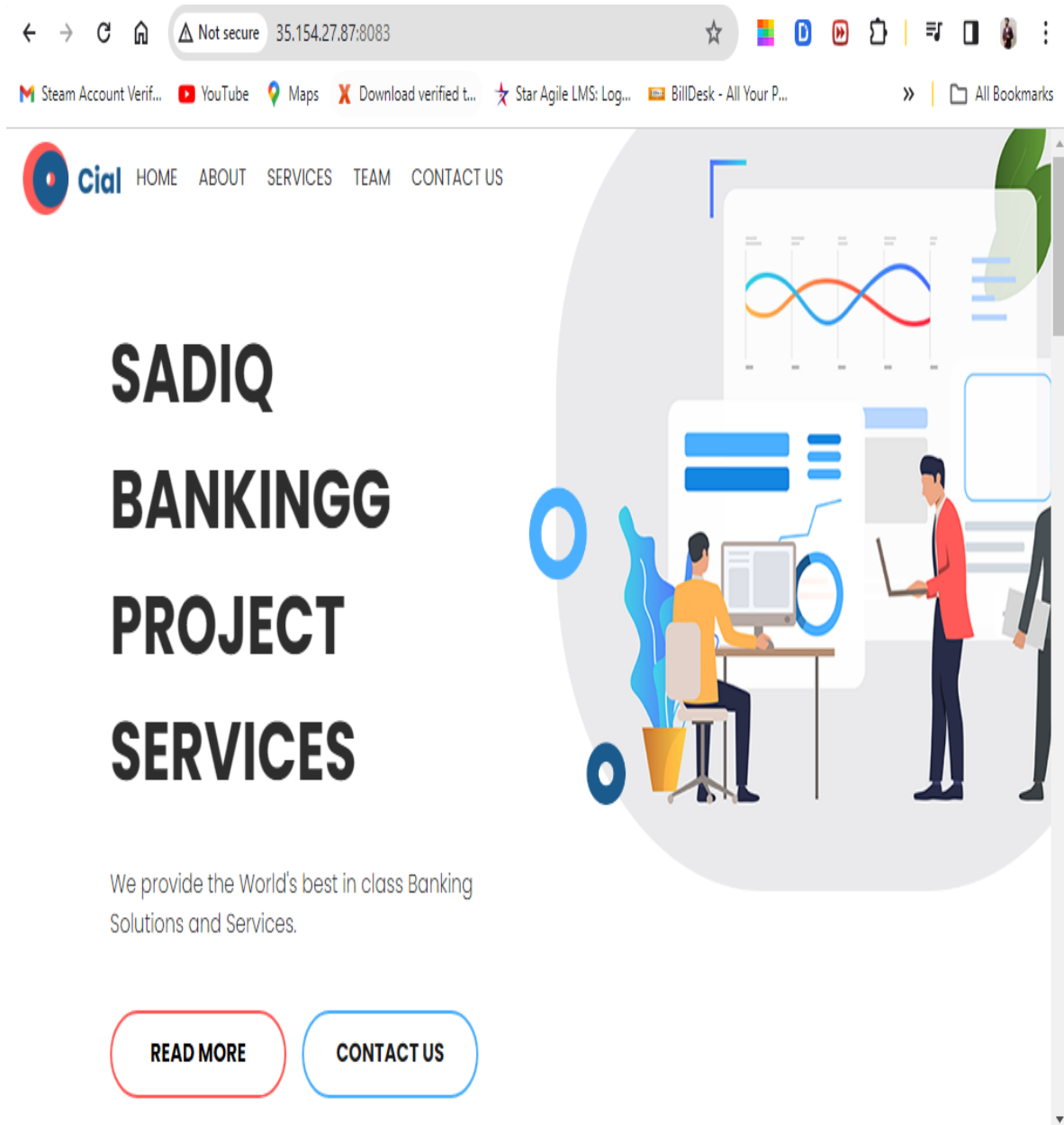
Script ?

```
1 pipeline{
2   agent any
3
4   stages{
5     stage('git')
6     {
7       steps{
8         git 'https://github.com/Salmansadiq809545/Banking-java-project'
9       }
10    }
11    stage('Terafrom init')
12    {
13      steps{
14        sh 'terraform init'
15      }
16    }
17    stage('Terafrom plan')
18  }
```

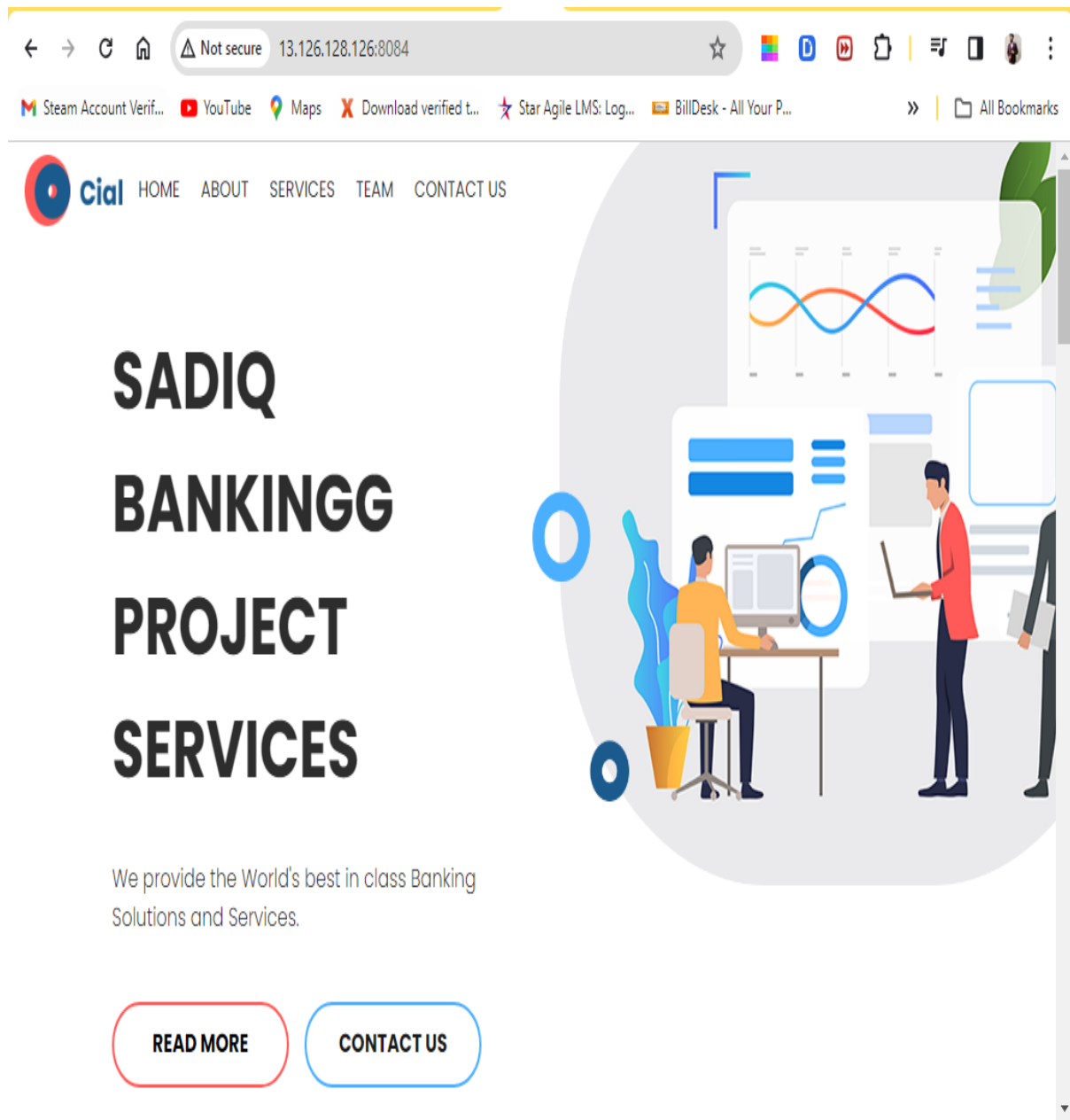
Above code shows the Jenkins pipeline for initializing planning and apply phase of terraform

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

This is the Ec2 machine created from Terraform code

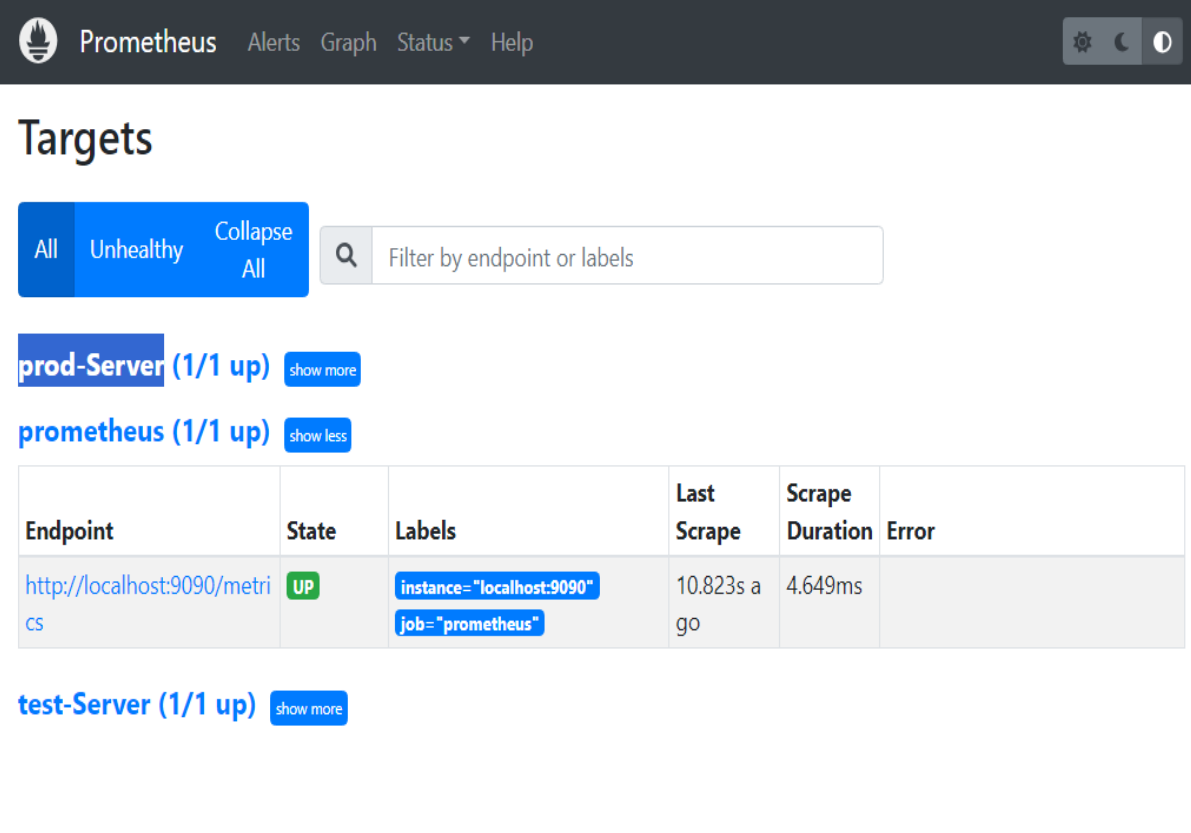


Above is the Container deployed using Ansible-playbook at port 8083



Above is the container deployed using terraform at port 8084

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



The screenshot displays the Prometheus web interface. At the top is a dark navigation bar with the Prometheus logo, menu items (Prometheus, Alerts, Graph, Status, Help), and settings icons. Below this is the 'Targets' section. It includes a filter bar with 'All' and 'Unhealthy' tabs, a 'Collapse All' button, and a search input field labeled 'Filter by endpoint or labels'. The main content area shows two target groups: 'prod-Server (1/1 up)' with a 'show more' button, and 'prometheus (1/1 up)' with a 'show less' button. A table lists the targets, with one entry visible for 'http://localhost:9090/metrics'. The table has columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. The 'prod-Server' group is expanded, showing the table data.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	10.823s ago	4.649ms	

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"]

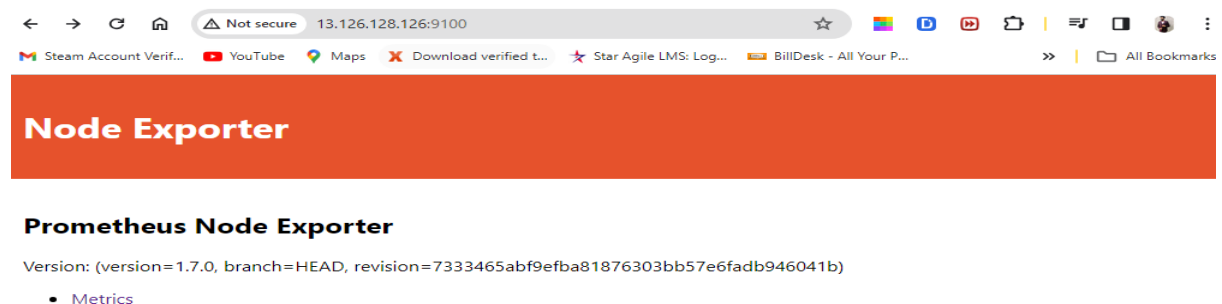
  - job_name: "test-Server"
    static_configs:
      - targets: ["35.154.27.87:9100"]

  - 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

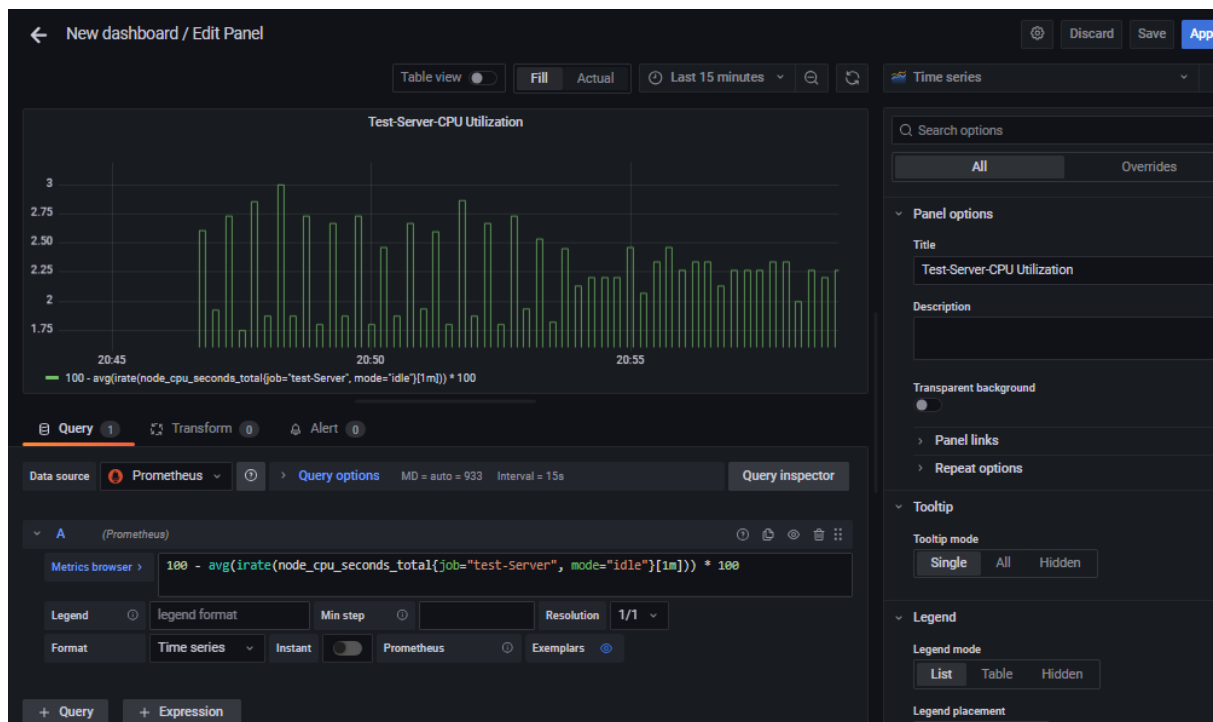
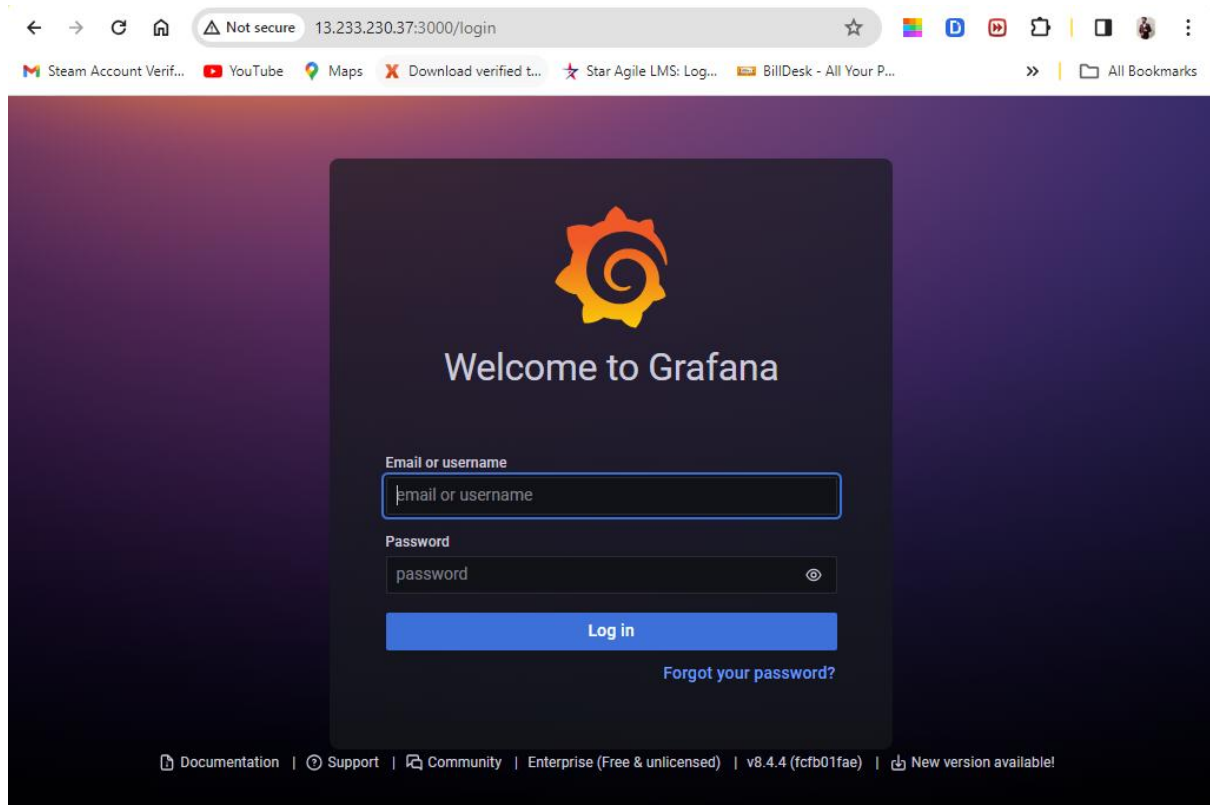
Prometheus Node Exporter

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 Utilization

2) Disk Space Utilization

3) Memory Utilization

THANK YOU