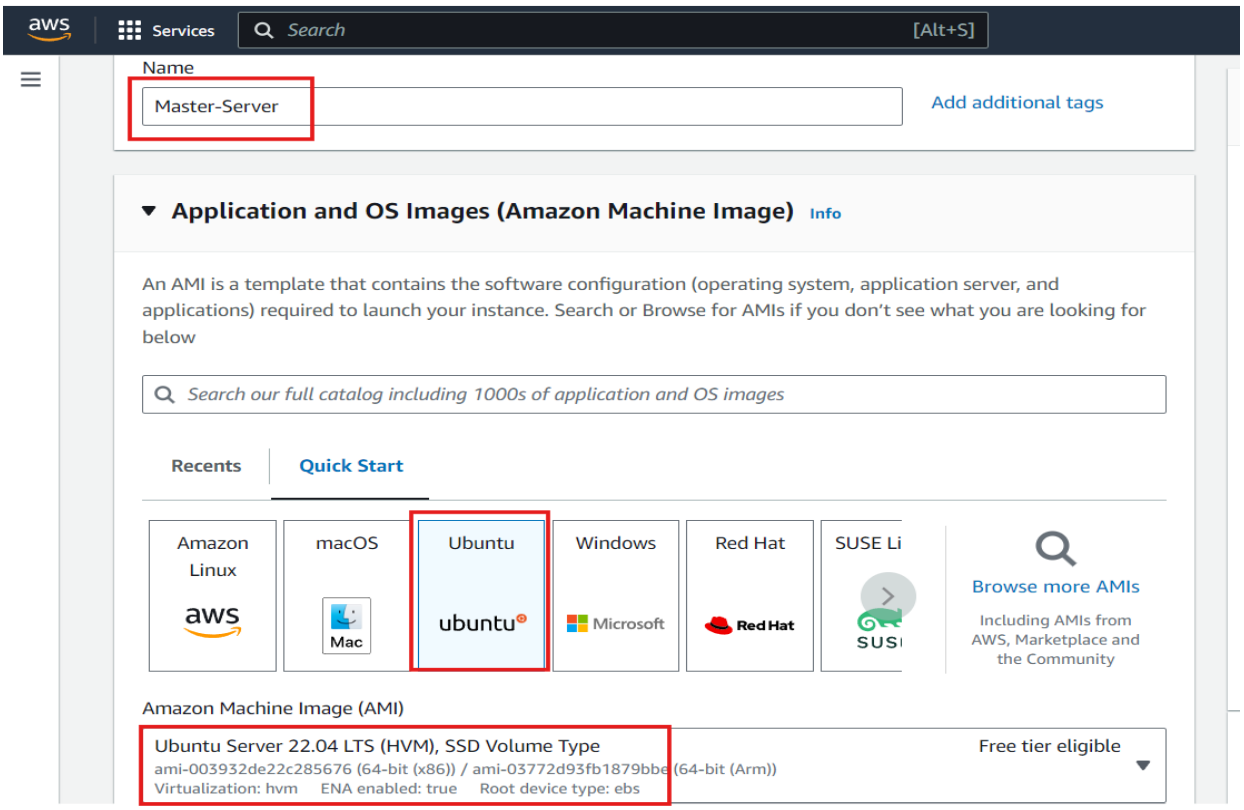
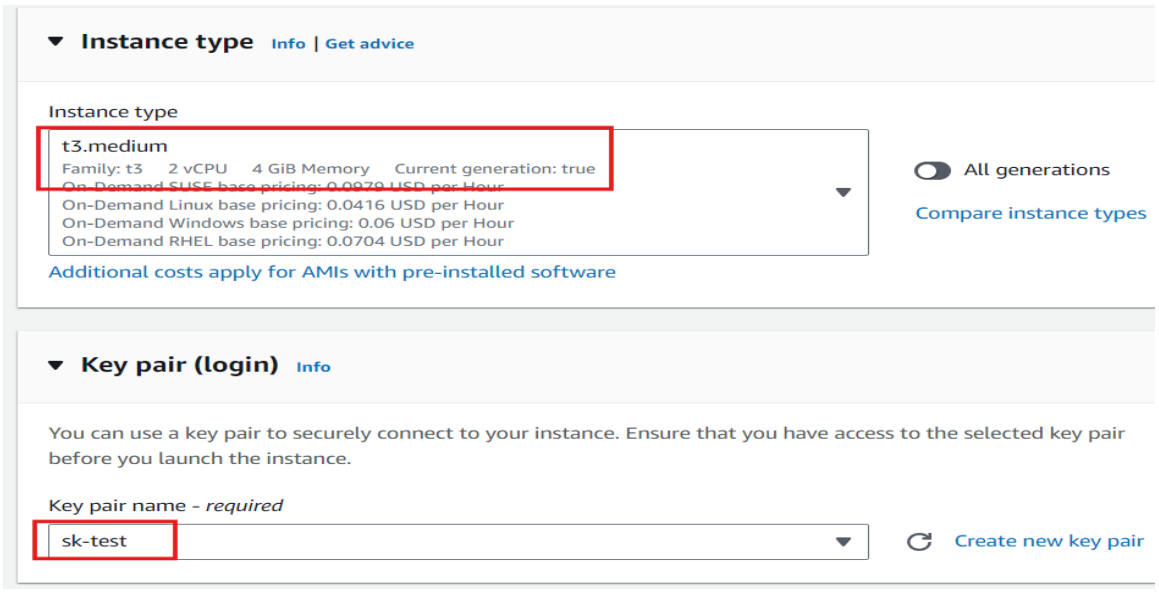
Capstone Project:- FinanceMe

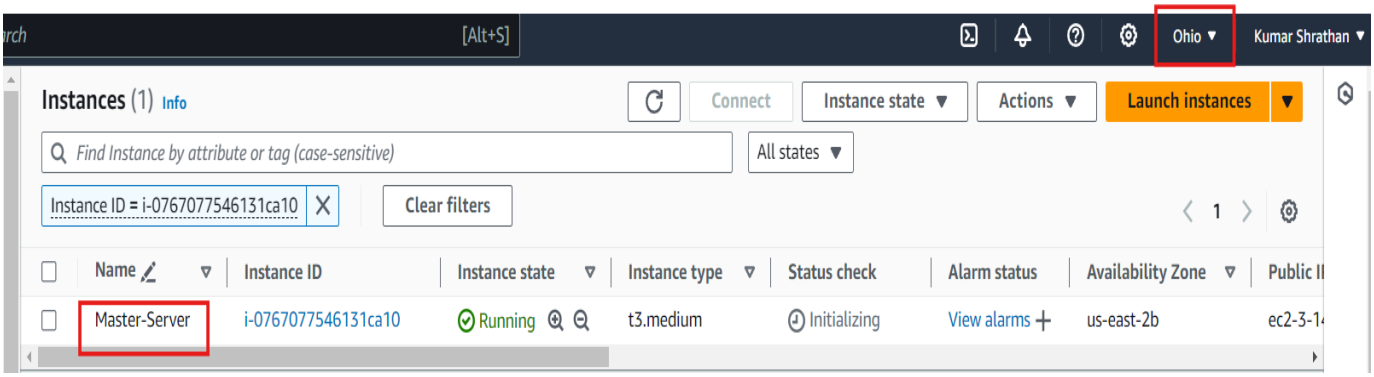
Submitted By – Shrathan Kumar

Date of submission – 05/08/2024

Step 1: Launching Master server in AWS, using EC2 instance, Ubuntu 22.04 AMI, T3.medium, Region Ohio(US-east-2).







Step 2: Connect to master server

```
asus@LAPTOP-LKCPIG9S MINGW64 ~/Downloads
$ ssh -i "sk-test.pem" ubuntu@ec2-3-14-148-204.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-14-148-204.us-east-2.compute.amazonaws.com (3.14.148.204)' can't be established.
ED25519 key fingerprint is SHA256:UewpTaVxwVPPSlBKtOhrY14cv61spK23lEHdnNr1jPo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-14-148-204.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Aug  1 14:09:16 UTC 2024

  System load:  0.02              Processes:             103
  Usage of /:   9.8% of 16.29GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for ens5: 172.31.31.102
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Aug  1 14:07:39 2024 from 3.16.146.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-31-102:~$
```

Step 3: Install the required software like git, java, maven, Jenkins, docker, Ansible.

Required software has been installed.

```
root@ip-172-31-31-102:/home/ubuntu# java --version
openjdk 11.0.24 2024-07-16
OpenJDK Runtime Environment (build 11.0.24+8-post-Ubuntu-1ubuntu322.04)
OpenJDK 64-Bit Server VM (build 11.0.24+8-post-Ubuntu-1ubuntu322.04, mixed mode, sharing)
root@ip-172-31-31-102:/home/ubuntu# mvn --version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.24, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.5.0-1022-aws", arch: "amd64", family: "unix"
root@ip-172-31-31-102:/home/ubuntu# git --version
git version 2.34.1
root@ip-172-31-31-102:/home/ubuntu# which docker
/usr/bin/docker
root@ip-172-31-31-102:/home/ubuntu# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2024-08-01 14:14:36 UTC; 4min 22s ago
   Main PID: 6194 (java)
      Tasks: 44 (limit: 4574)
     Memory: 1.2G
        CPU: 1min 15.588s
     CGroup: /system.slice/jenkins.service
             └─6194 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 01 14:14:06 ip-172-31-31-102 jenkins[6194]: f3cdb5bbf1444caeb5d98b7532e2251f
Aug 01 14:14:06 ip-172-31-31-102 jenkins[6194]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 01 14:14:06 ip-172-31-31-102 jenkins[6194]: ************************************************************
Aug 01 14:14:06 ip-172-31-31-102 jenkins[6194]: ************************************************************
Aug 01 14:14:06 ip-172-31-31-102 jenkins[6194]: ************************************************************
Aug 01 14:14:36 ip-172-31-31-102 jenkins[6194]: 2024-08-01 14:14:36.406+0000 [id=30]       INFO       jenkins.InitReactorRunner$1#onAttained: Compl
Aug 01 14:14:36 ip-172-31-31-102 jenkins[6194]: 2024-08-01 14:14:36.446+0000 [id=22]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins i
Aug 01 14:14:36 ip-172-31-31-102 systemd[1]: Started Jenkins Continuous Integration Server.
Aug 01 14:14:36 ip-172-31-31-102 jenkins[6194]: 2024-08-01 14:14:36.710+0000 [id=47]       INFO       h.m.DownloadService$Downloadable#load: Obtain
Aug 01 14:14:36 ip-172-31-31-102 jenkins[6194]: 2024-08-01 14:14:36.711+0000 [id=47]       INFO       hudson.util.Retrier#start: Performed the acti
lines 1-20/20 (END)
root@ip-172-31-31-102:/home/ubuntu# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu2~22.04.1
root@ip-172-31-31-102:/home/ubuntu#
```

Add Jenkins to sudo group and Jenkins to docker group.

```
root@ip-172-31-31-102:/home/ubuntu# sudo usermod -aG sudo jenkins
root@ip-172-31-31-102:/home/ubuntu# sudo usermod -aG docker jenkins
root@ip-172-31-31-102:/home/ubuntu#
```

Step 4: Now will create the Prod and Grafana server using terraform

Creating prod server and Grafana server infrastructure using Terraform.(I will create 2 servers, and for Prod server I will attach elastic ip)

Installed terraform

```
root@ip-172-31-31-102:/home/ubuntu# terraform --version
Terraform v1.9.3
on linux_amd64
root@ip-172-31-31-102:/home/ubuntu#
```

Creating Terraform directory and creating main.tf file to create infrastructure

```
root@ip-172-31-31-102:/home/ubuntu# cd Terraform/
root@ip-172-31-31-102:/home/ubuntu/Terraform# vi main.tf
```

Terraform init

Terraform apply

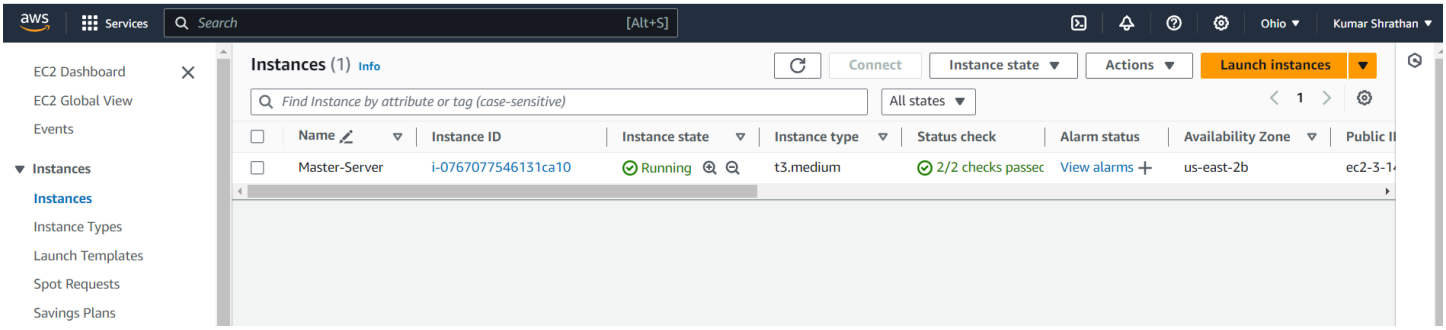
Terraform main.tf file

```
# Specify the provider
provider "aws" {
  region = "us-east-2"
}

# Create a security group allowing SSH access
resource "aws_security_group" "instance_sg" {
  name        = "instance-sg"
  description = "Allow SSH access"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

# Create an EC2 instance
resource "aws_instance" "prod_server" {
  ami           = "ami-003932de22c285676"
  instance_type = "t3.medium"
  key_name      = "sk-test"
  security_groups = [aws_security_group.instance_sg.name]

  tags = {
    Name = "Prod-server"
  }
}

resource "aws_instance" "grafana" {
```

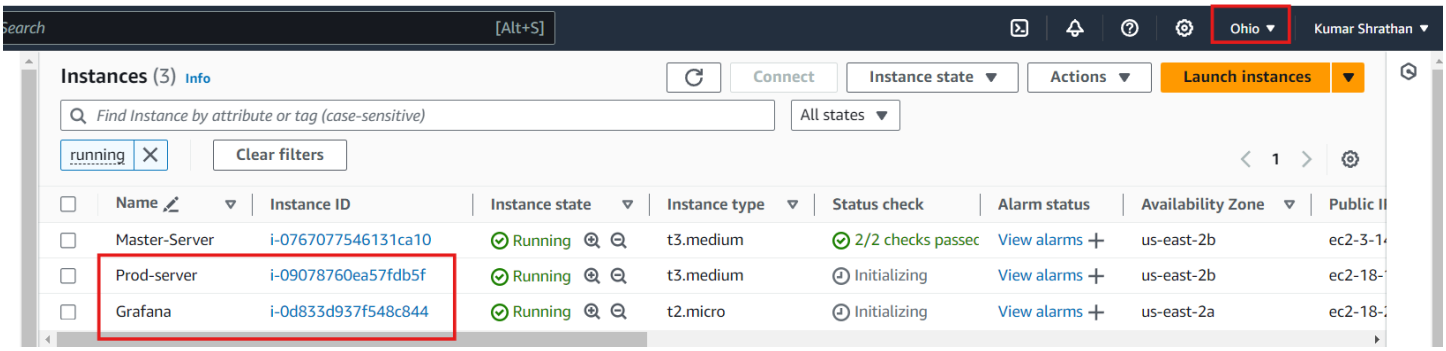
```
  ami           = "ami-003932de22c285676"
  instance_type = "t2.micro"
  key_name      = "sk-test"
  security_groups = [aws_security_group.instance_sg.name]

  tags = {
    Name = "Grafana"
  }
}

# Allocate an Elastic IP
resource "aws_eip" "instance_eip" {
  instance = aws_instance.prod_server.id
}
```

```
root@ip-172-31-31-102:/home/ubuntu/Terraform# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.60.0...
- Installed hashicorp/aws v5.60.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@ip-172-31-31-102:/home/ubuntu/Terraform# terraform fmt
main.tf
root@ip-172-31-31-102:/home/ubuntu/Terraform# terraform validate
Success! The configuration is valid.

root@ip-172-31-31-102:/home/ubuntu/Terraform# terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following sym
  + create

Terraform will perform the following actions:

  # aws_eip.instance_eip will be created
  + resource "aws_eip" "instance_eip" {
      + allocation_id        = (known after apply)
      + arn                  = (known after apply)
      + association_id       = (known after apply)
```

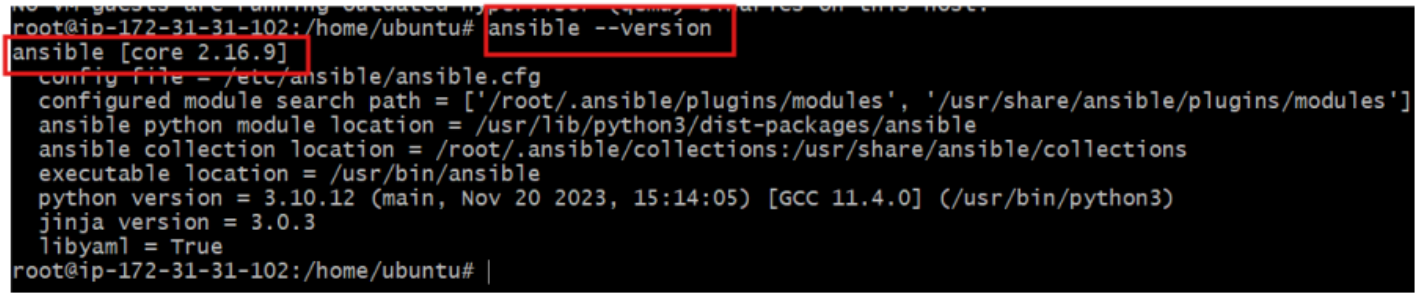
Before in Ohio region had only 1 instance



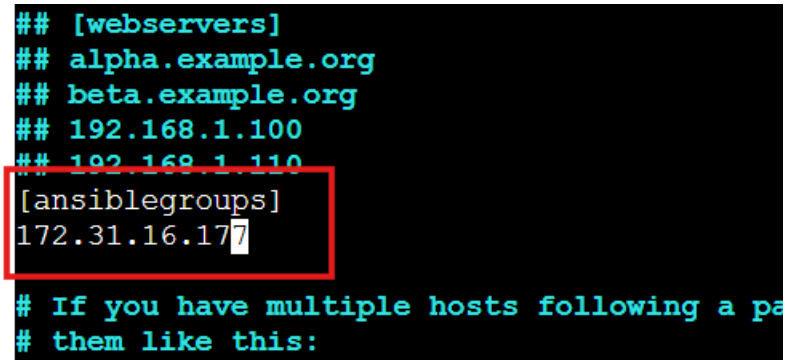
After running terraform file.



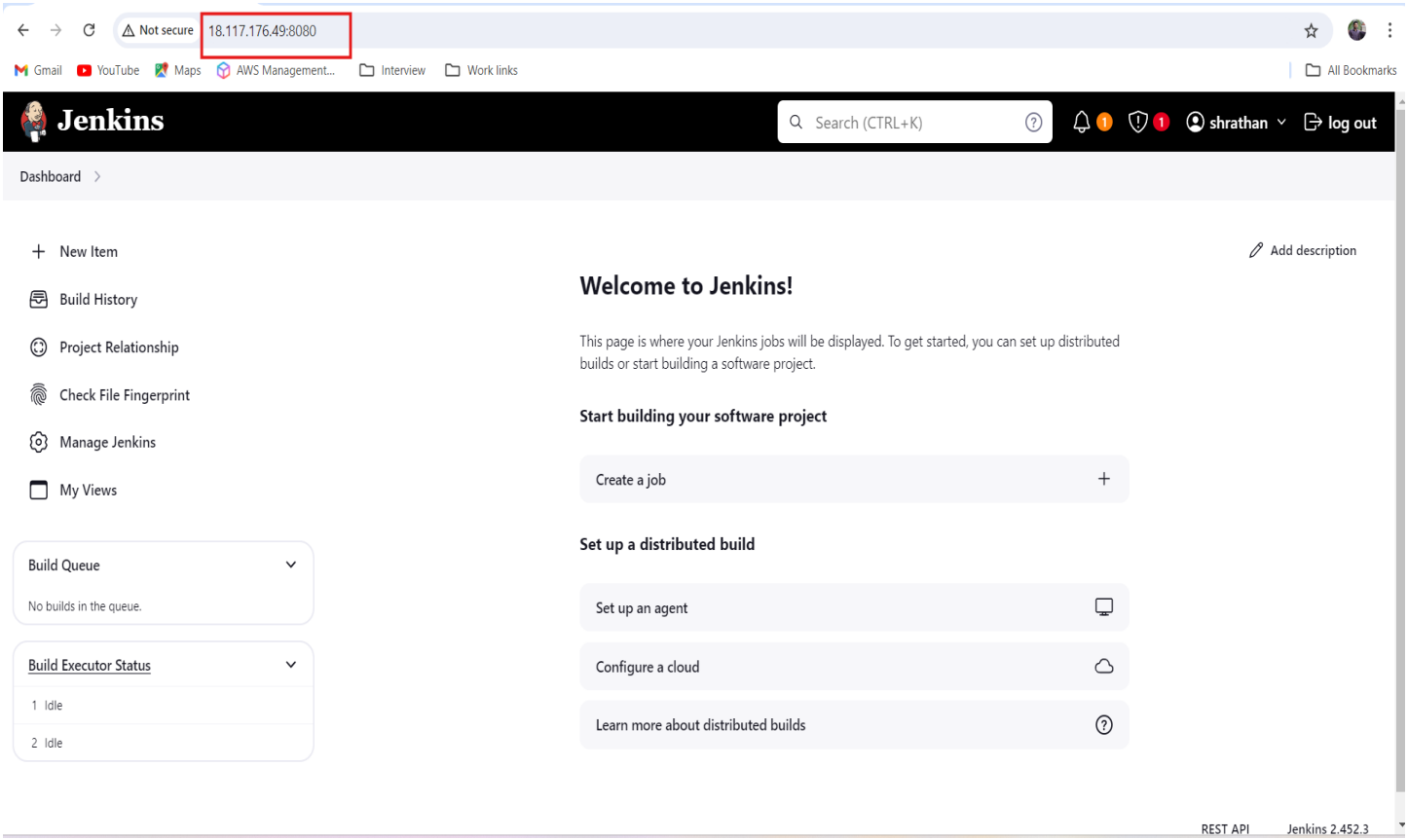
Installed and configured Ansible



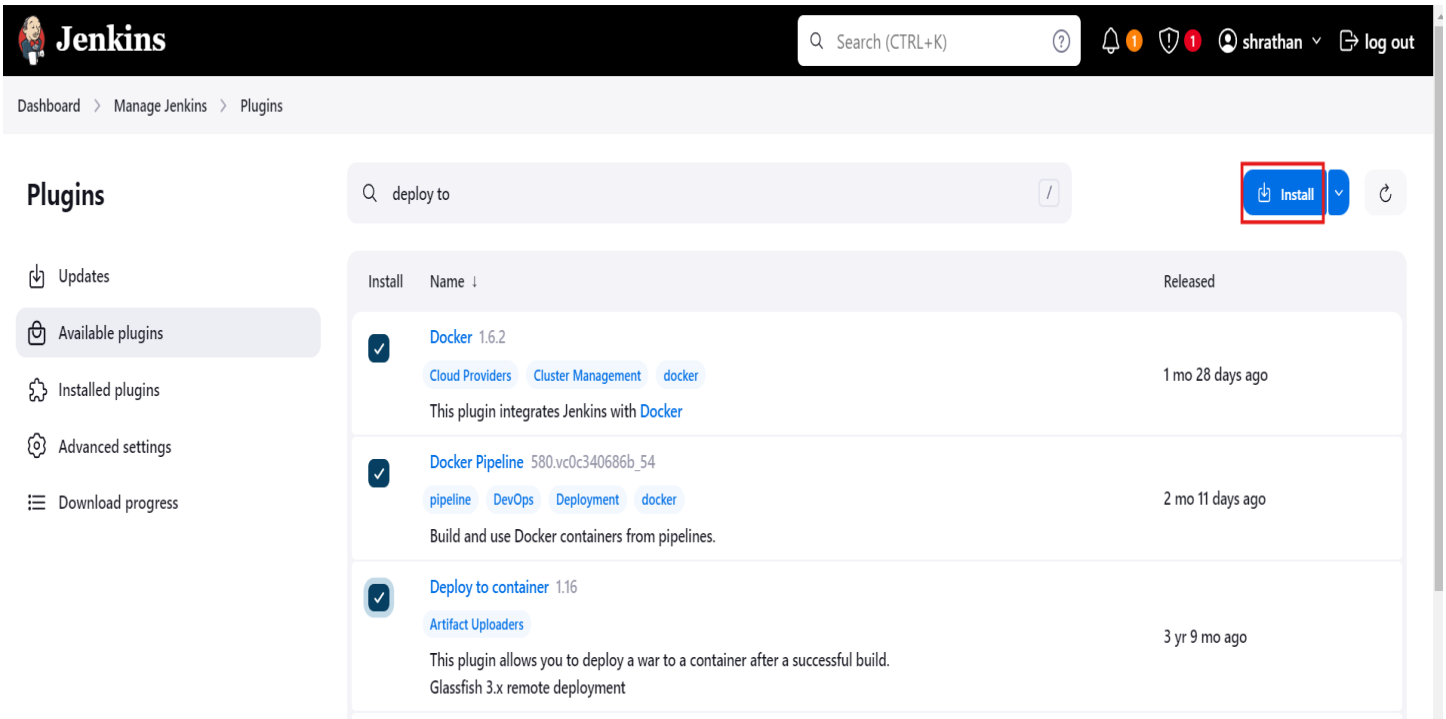
Added Private ip of Prod server in /etc/ansible/host in master.



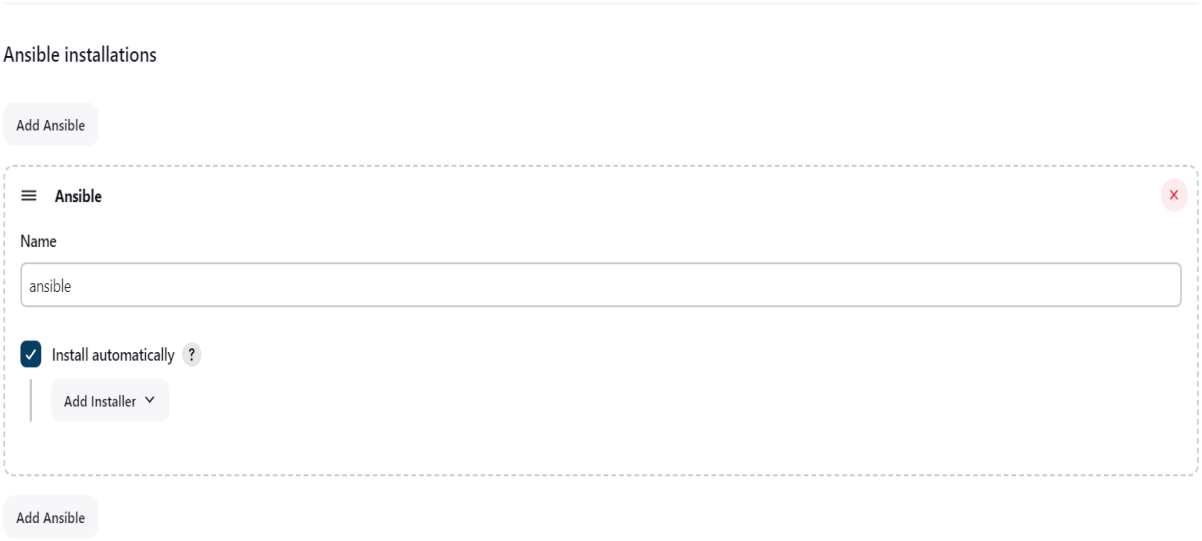
Step 5: Goto Master instance and Login into Jenkins dashboard and configure.



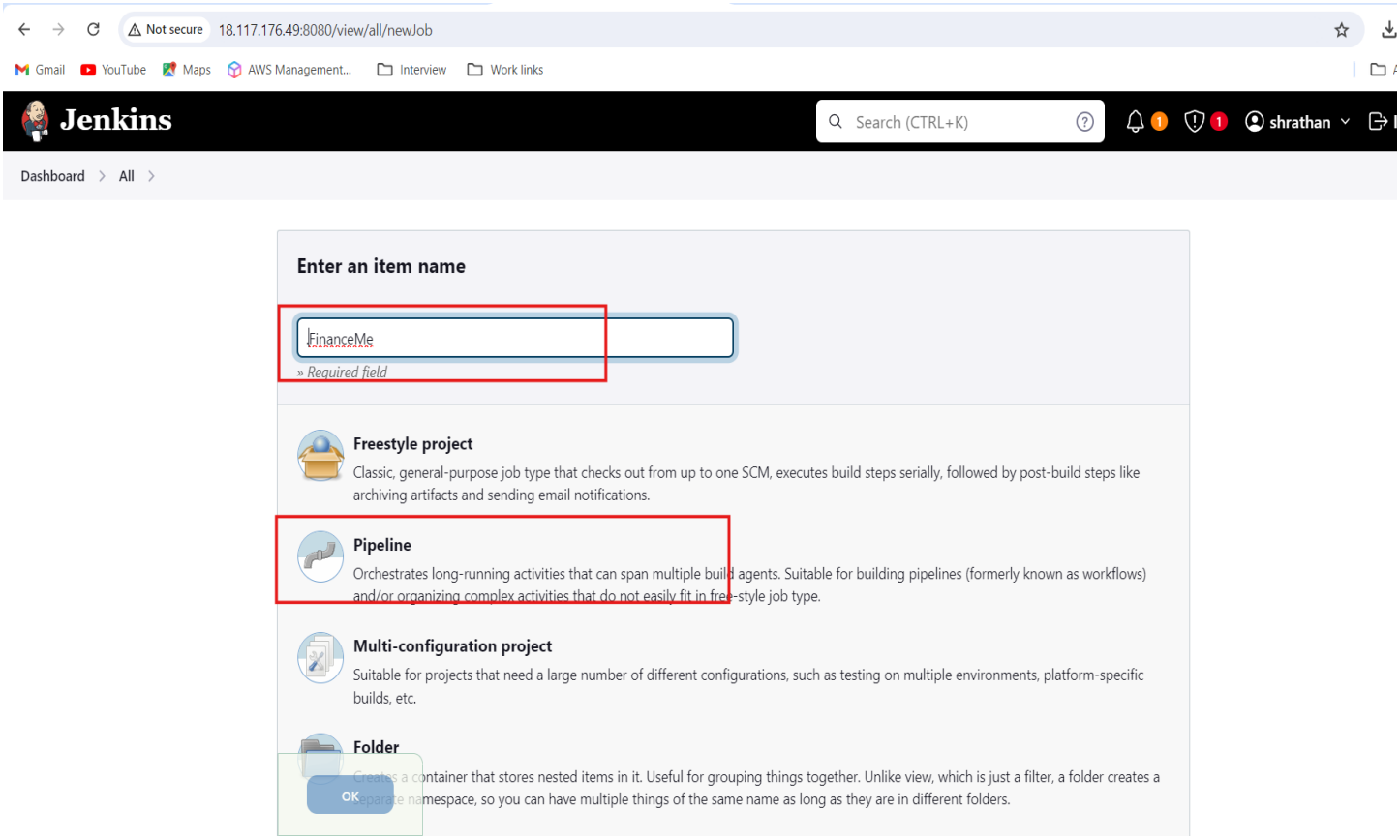
Install the plugins required and configure the java, maven in Jenkins.

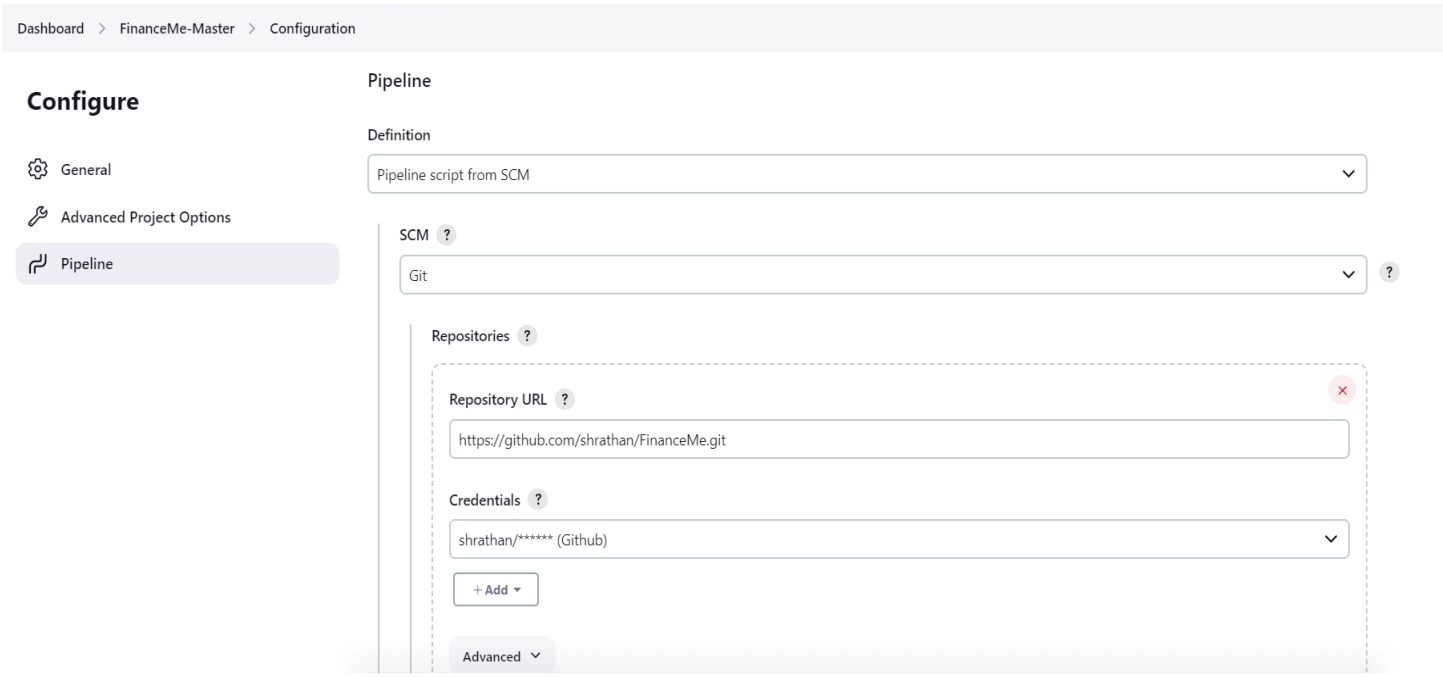


Configure ansible

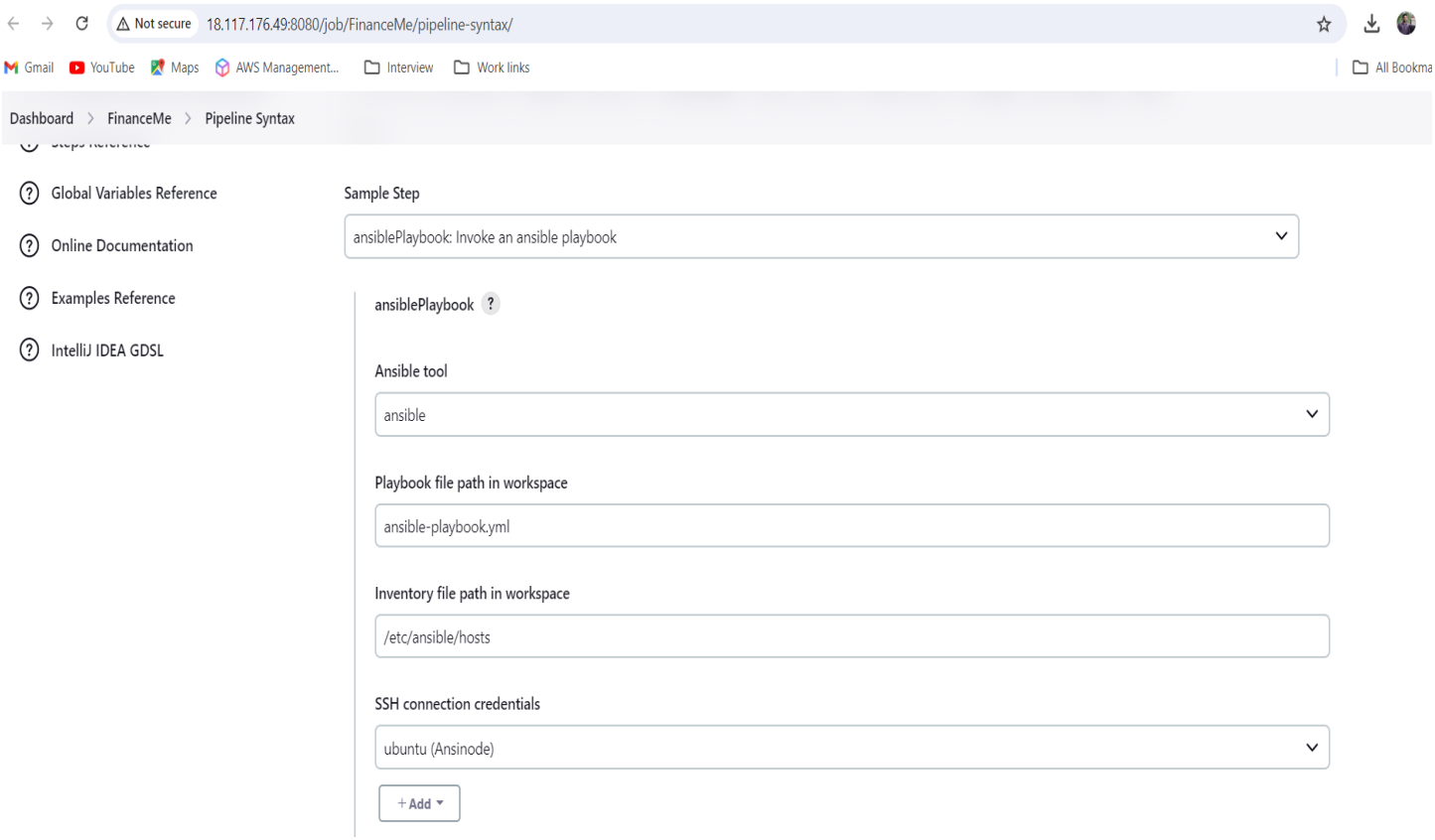


Step 6: Create new pipeline project to launch financeme java project using docker container also integrate ansible.

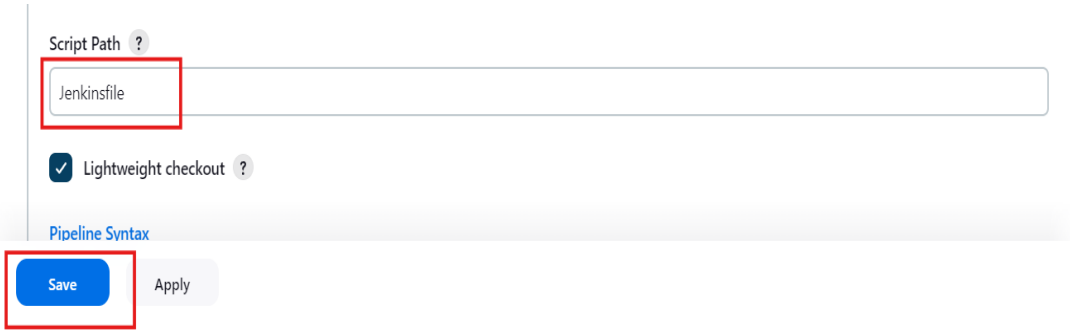




From Pipeline Script configure Ansible



Copy this and use in Jenkins script to invoke ansible in pipeline.



Jenkins Pipeline file :-

```
pipeline{
   agent any
   stages{
      stage('checkout the code from github'){
         steps{
            git url: 'https://github.com/shrathan/FinanceMe.git', branch: "master"
            echo 'github url checkout'
         }
      }
      stage('Code package '){
         steps{
            sh 'mvn clean package'
         }
      }
      stage('Build dockerfiles'){
       steps{
          sh 'docker build -t shrathan/banking:v1 .'
        }
       }
       stage('Docker login & Push') {
         steps {
            withCredentials([usernamePassword(credentialsId: 'dockerhub-pwd', passwordVariable:
'PASS', usernameVariable: 'USER')]) {
               sh "echo $PASS | docker login -u $USER --password-stdin"
```

```
                sh 'docker push shrathan/banking:v1'
            }
          }
        }
        stage('Run Container using ansible'){
          steps{
ansiblePlaybook become: true, credentialsId: 'Ansinode', disableHostKeyChecking: true, installation:
'ansible', inventory: '/etc/ansible/hosts', playbook: 'ansible-playbook.yml', vaultTmpPath: ''          }
        }
    }
}
```

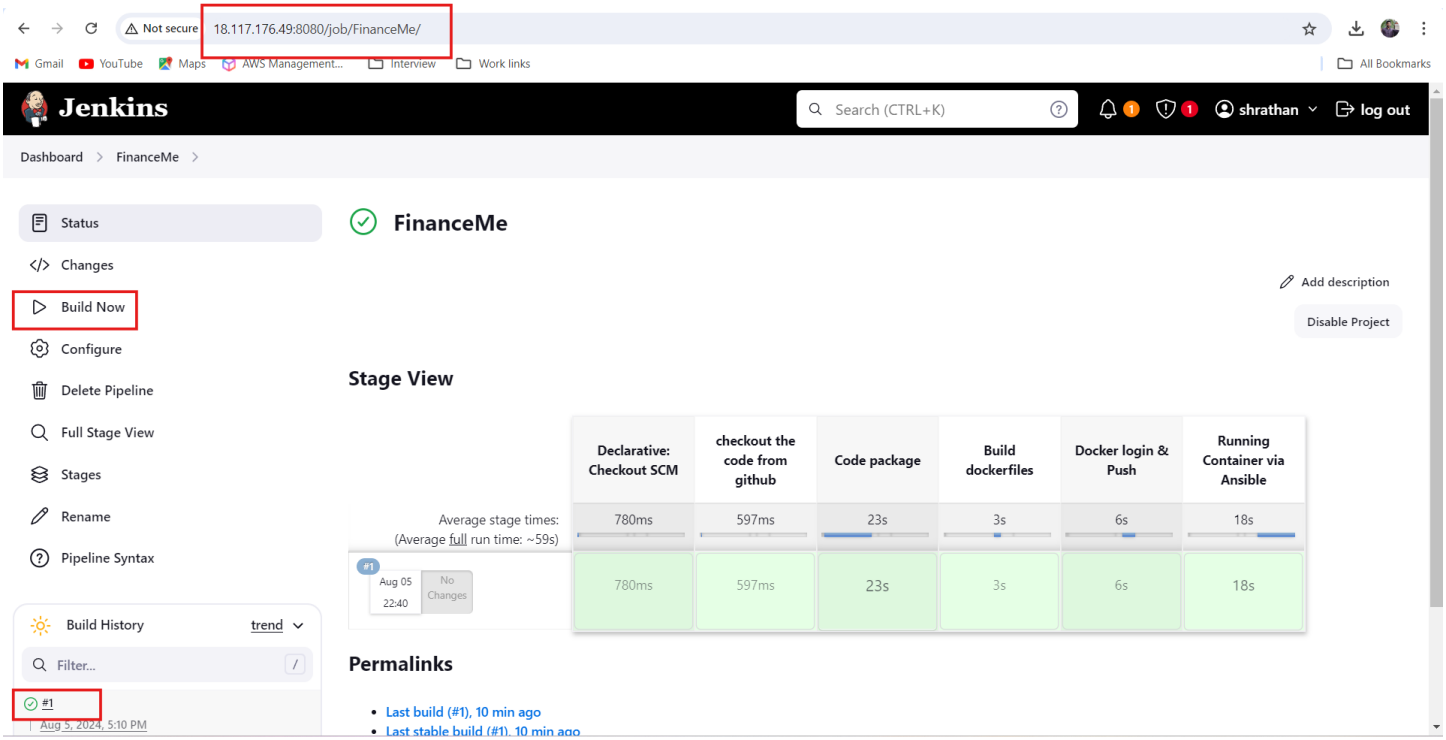
Ansible YAML file

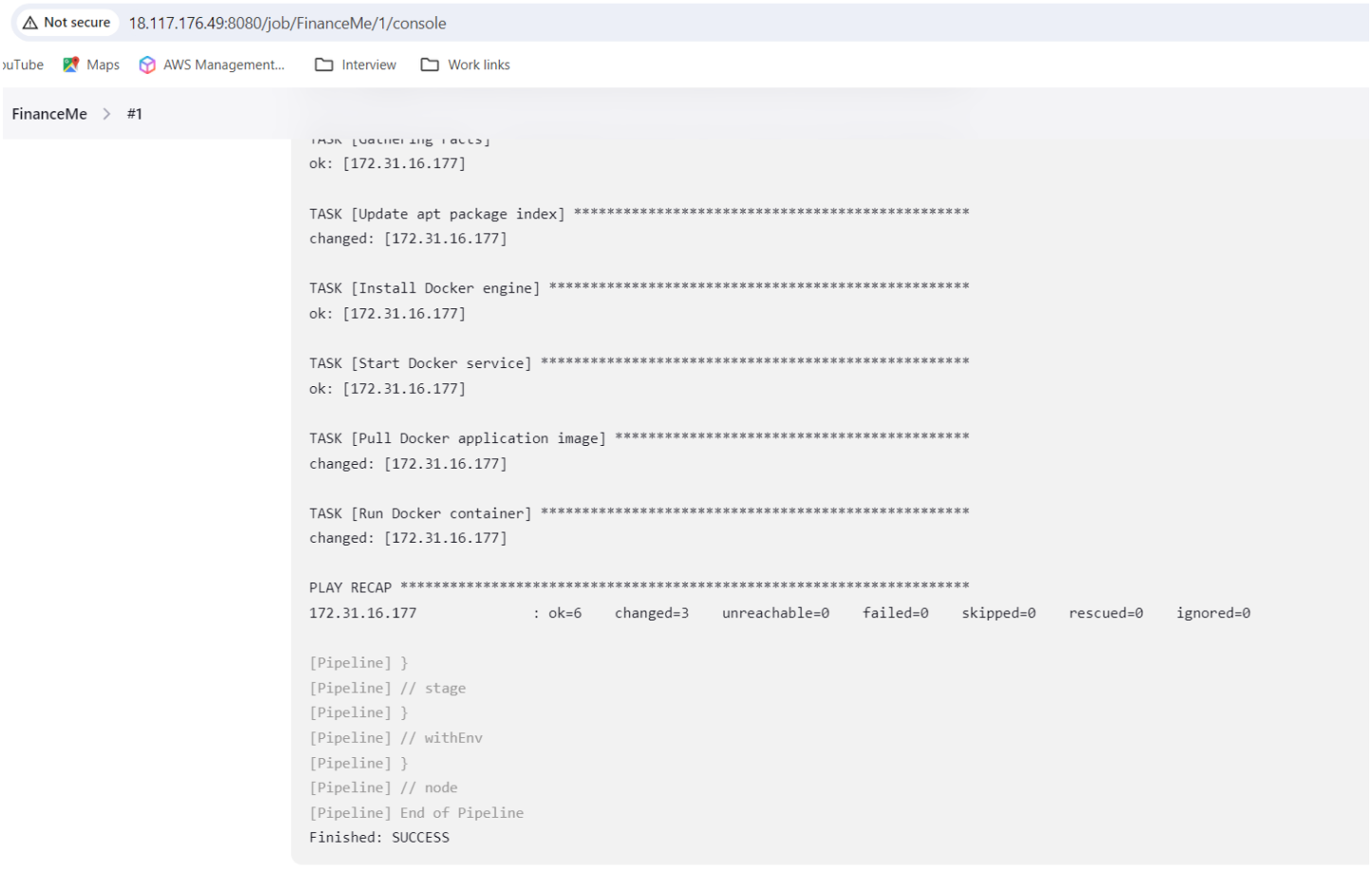
```
---
- name: Docker
  hosts: all
  become: yes
  connection: ssh
  tasks:
    - name: Update apt package index
      command: sudo apt update

    - name: Install Docker engine
      apt:
        name: docker.io
        state: present

    - name: Start Docker service
      service:
        name: docker
        state: started
        enabled: yes
      become: yes

    - name: Pull Docker application image
      docker_image:
        name: shrathan/banking:v1
        source: pull
        state: present

    - name: Run Docker container
      docker_container:
        name: finance
        image: shrathan/banking:v1
```

```
    state: started
    restart_policy: always
    ports:
      - "8005:8005"
```

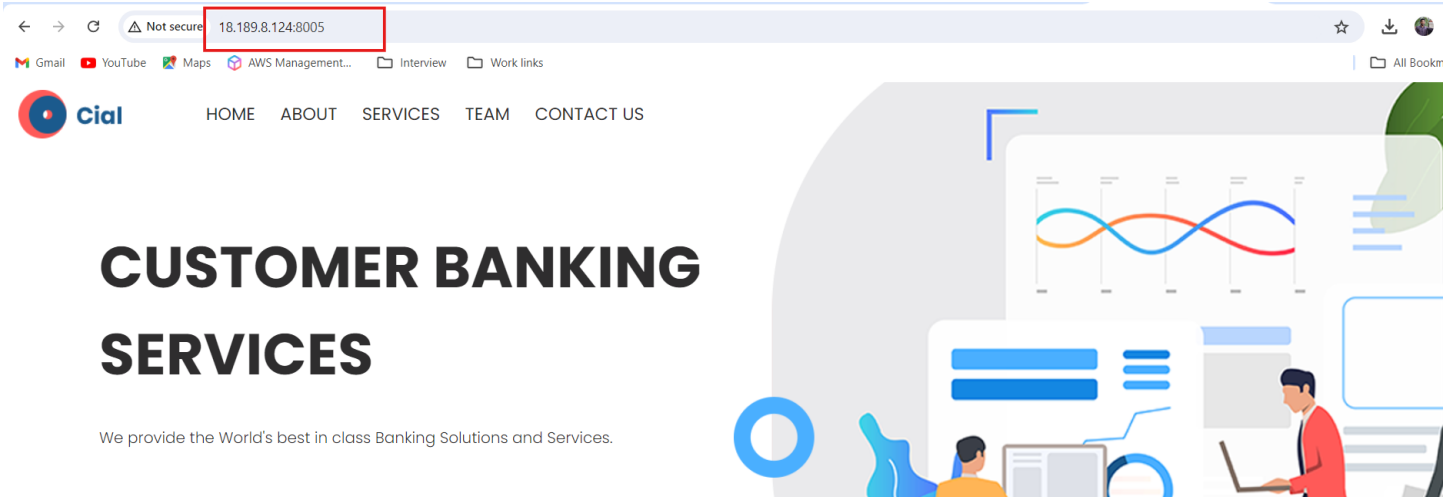
Click on Build Now to execute the pipeline

Build success

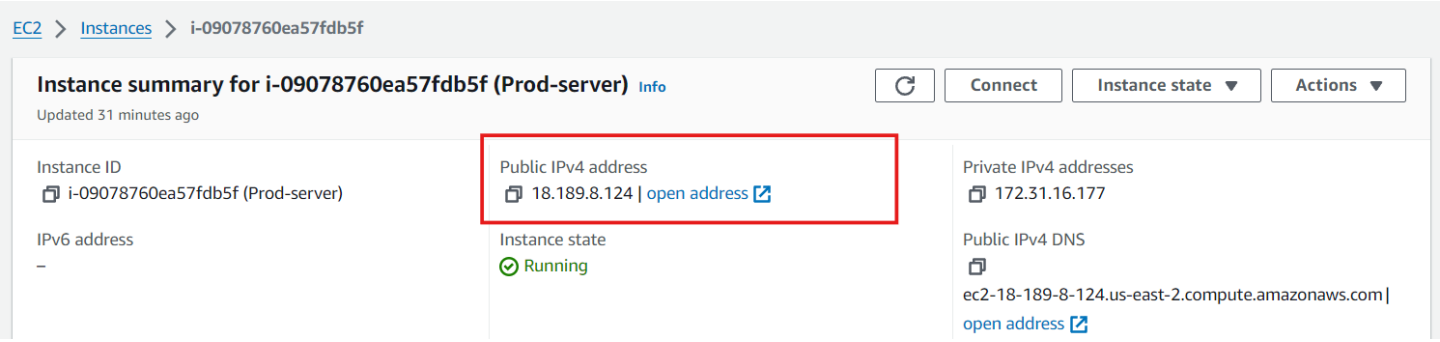




Now u can access the container using pod server ip with port



Prod server IP

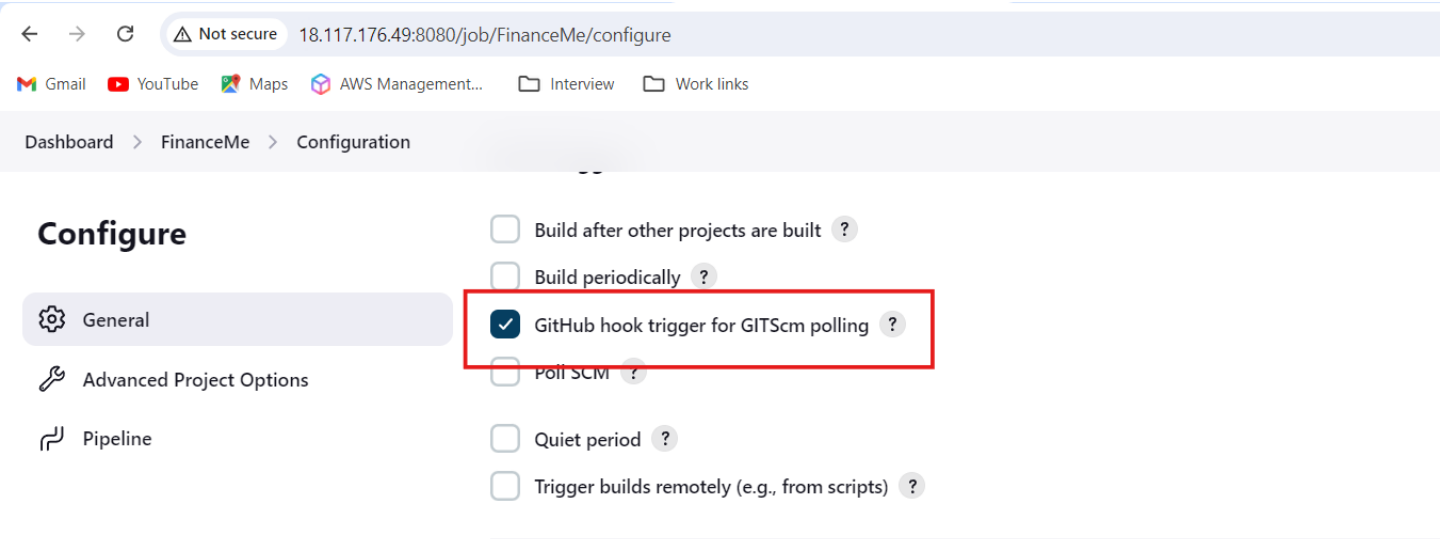


You can access my all code files in my git repository:-

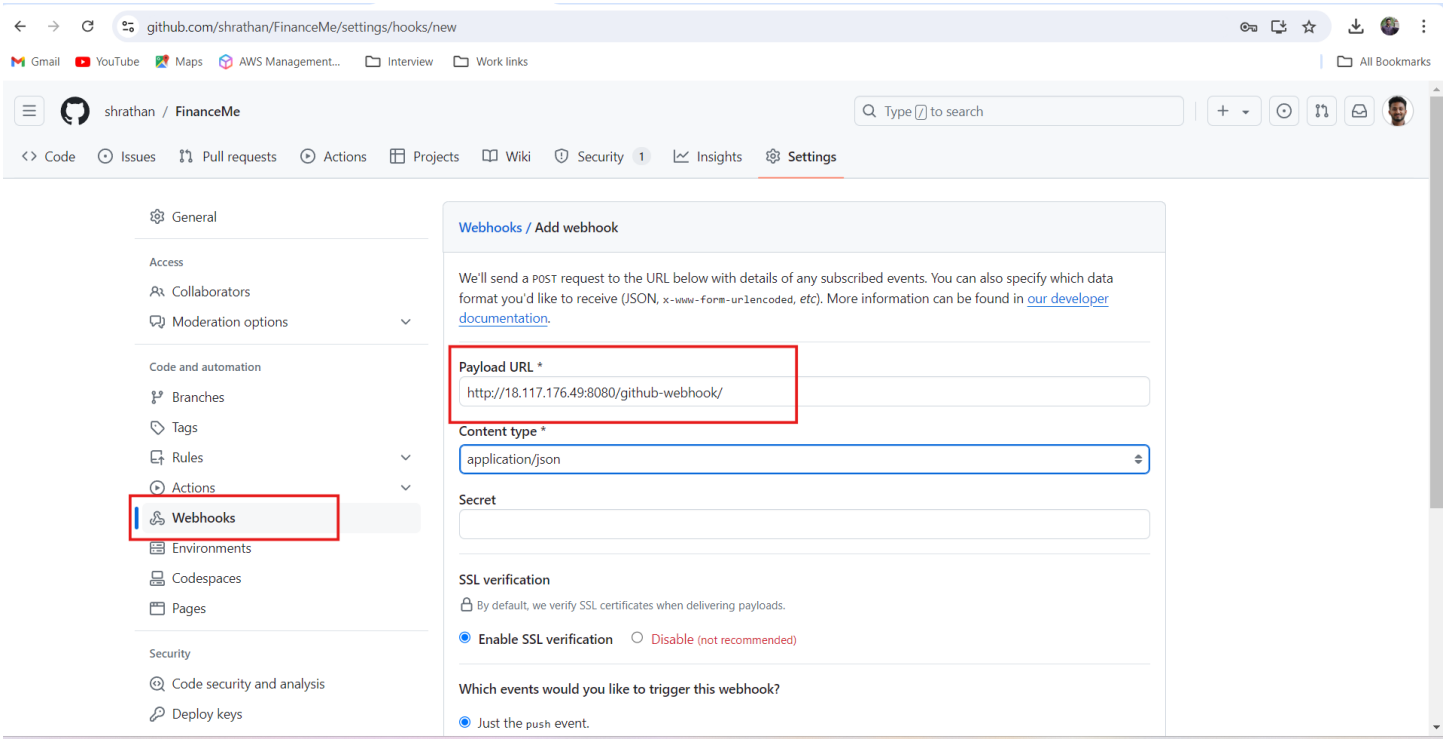
https://github.com/shrathan/FinanceMe

Step 7: Creating github webhook triggers.

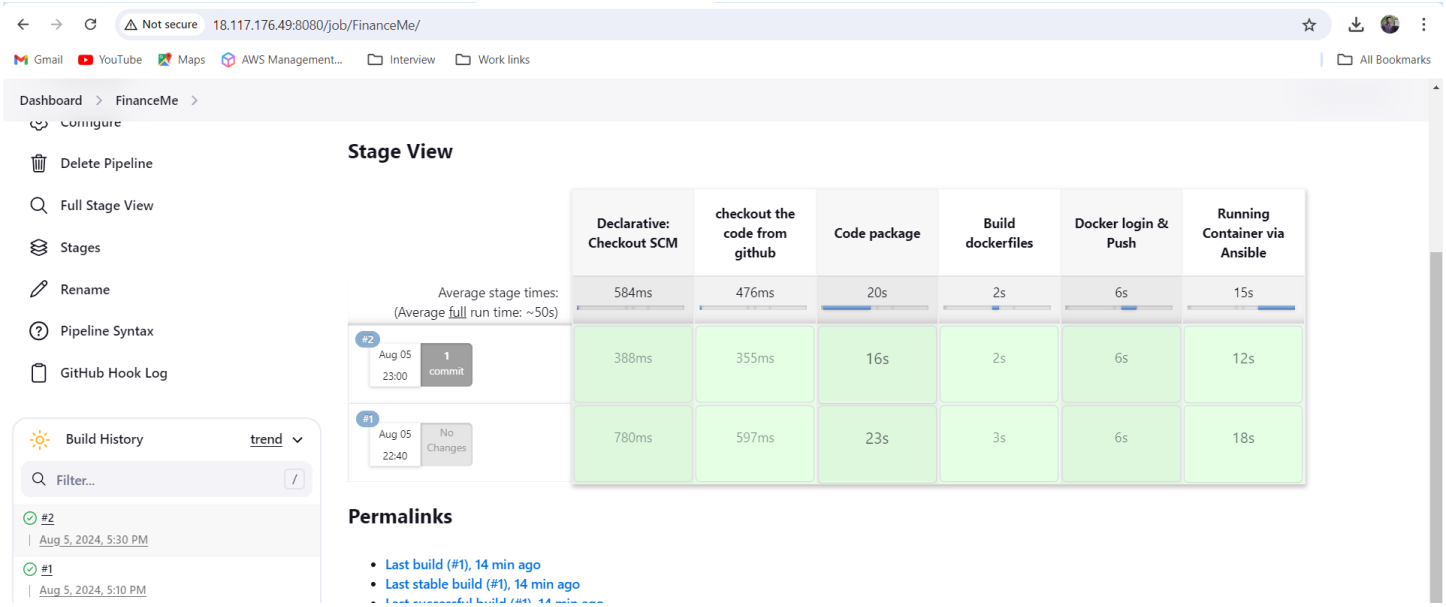
Enable Github webhook option in Jenkins.

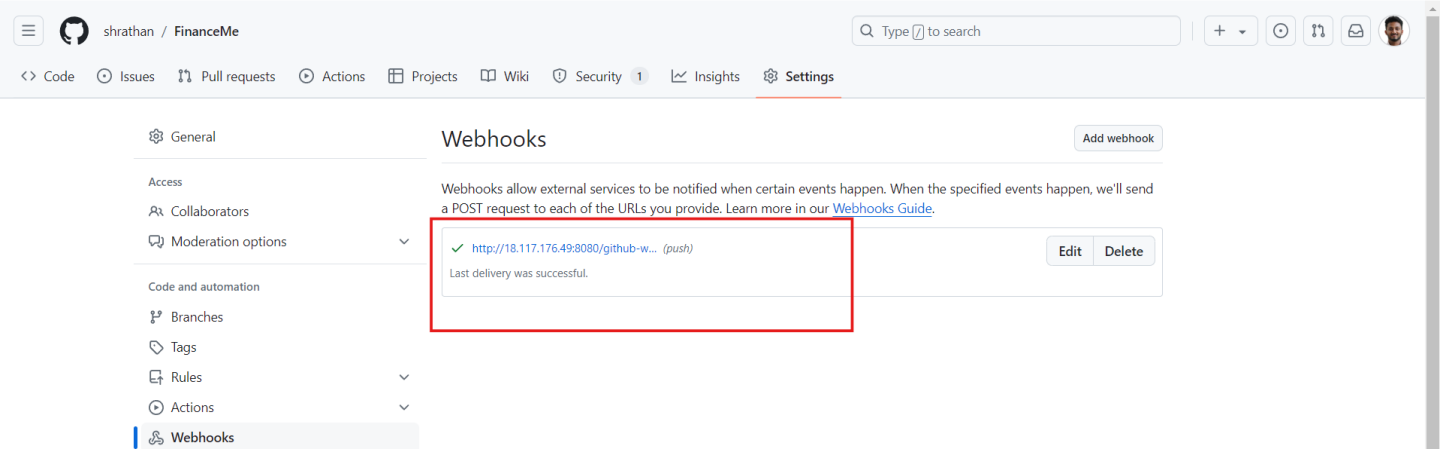


Configure Webhook in Github

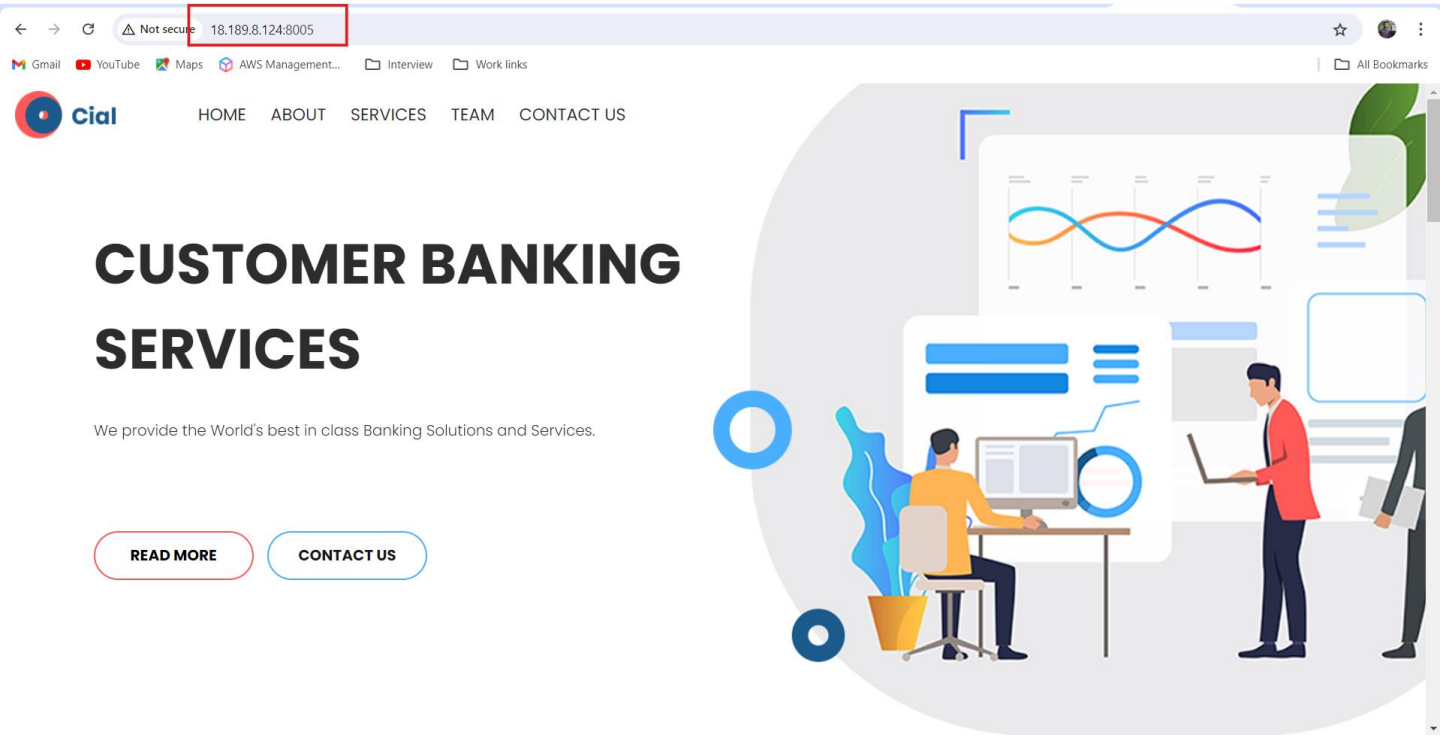


I have changed the some code in github, trigger was successful





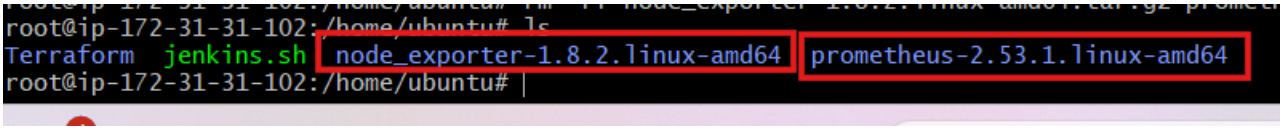
Now will access the Prod-server and check the running application.



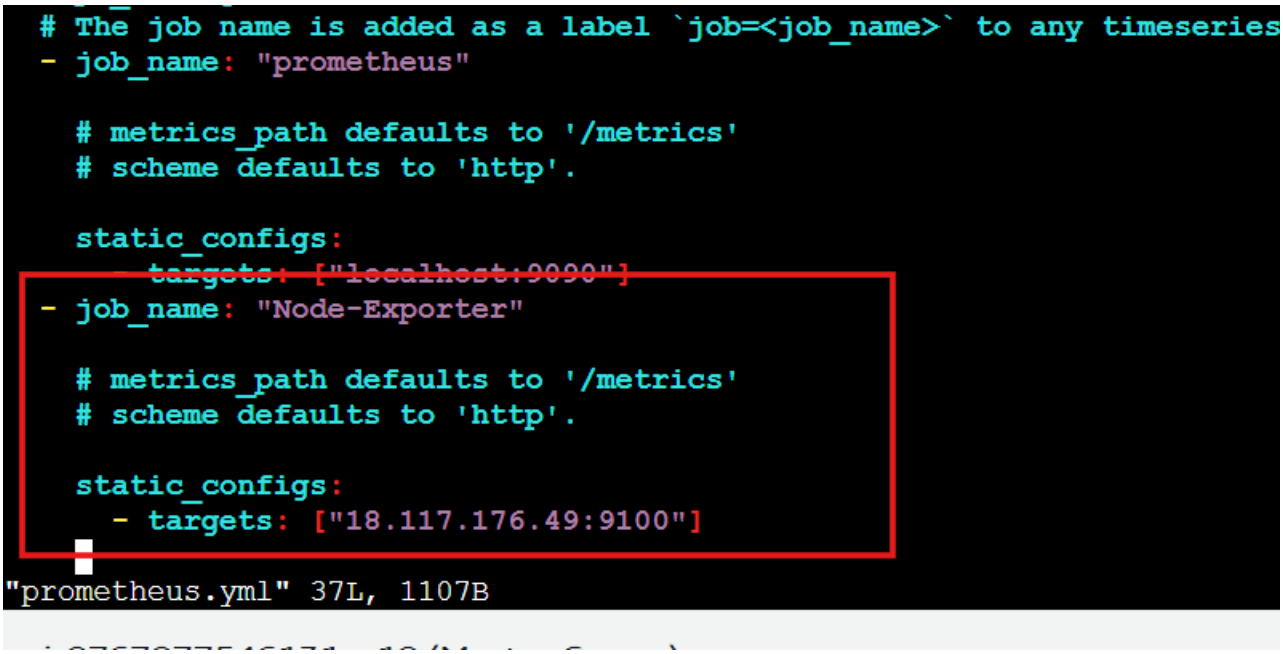
We have successfully launched docker container in Prod server.

Step 7: Now will install and configure Prometheus and node exporter in Master and Prod server.

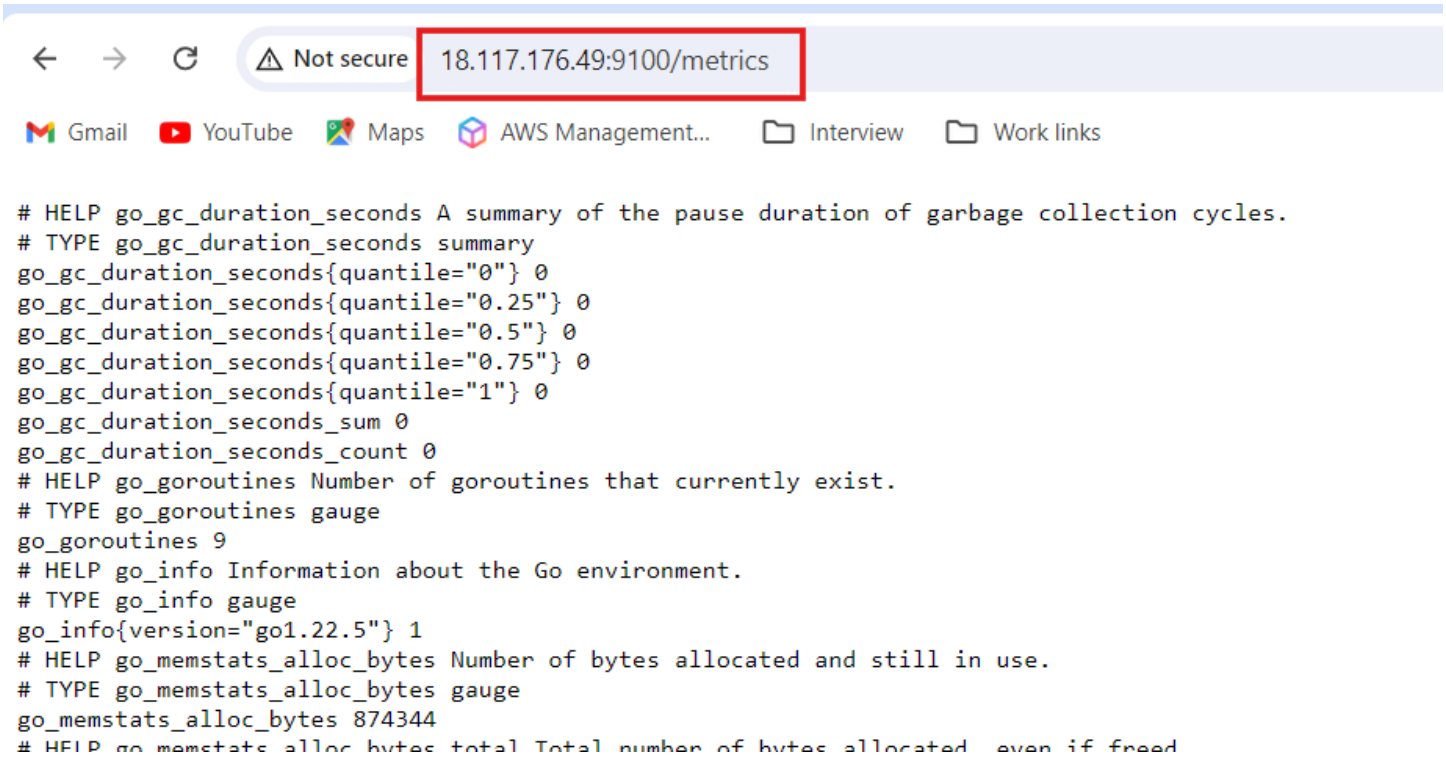
Master server



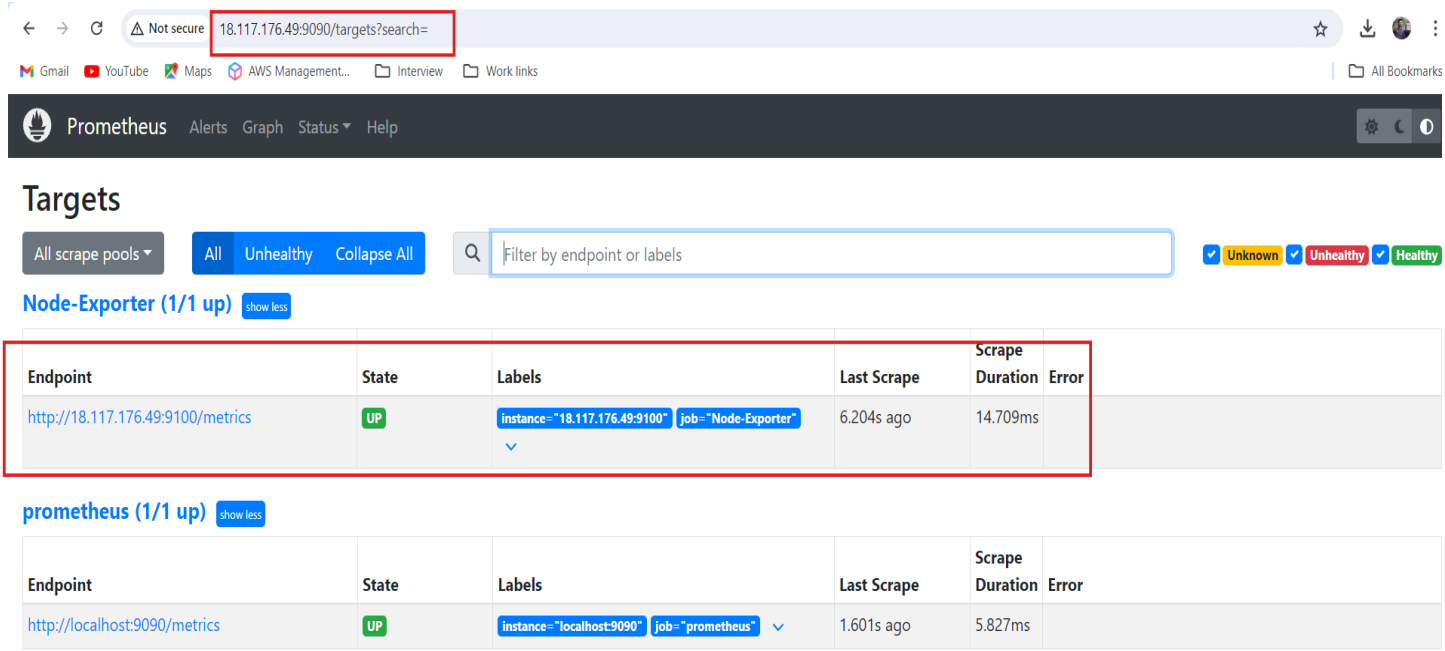
Adding Node exporter in Prometheus yaml file in master server



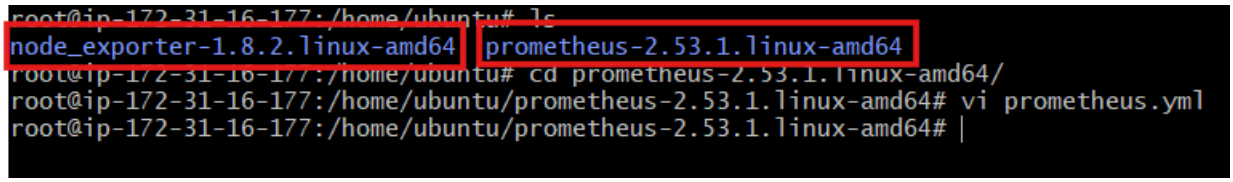
Node exporter is collecting metrics from Master server

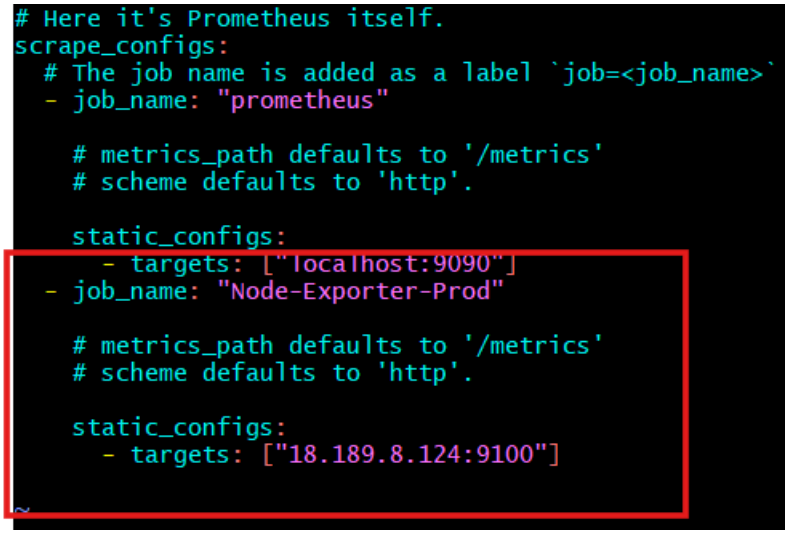


Now will access the Prometheus server

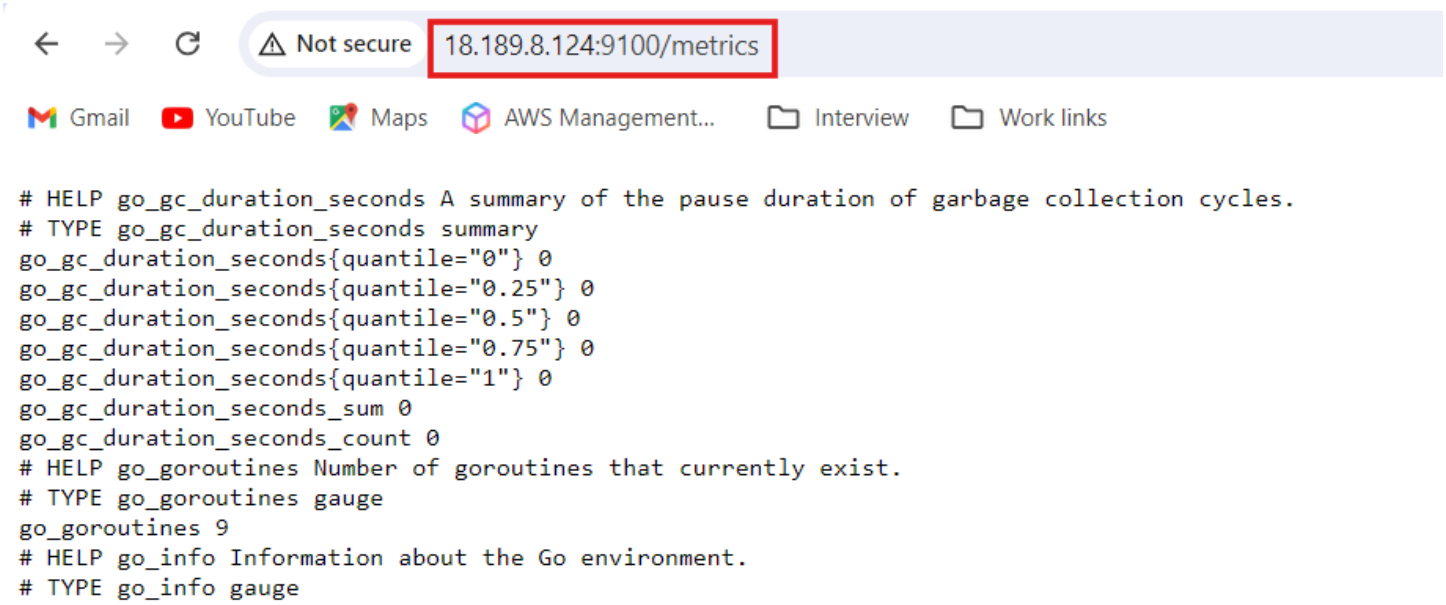


Now will configure same in Prod server

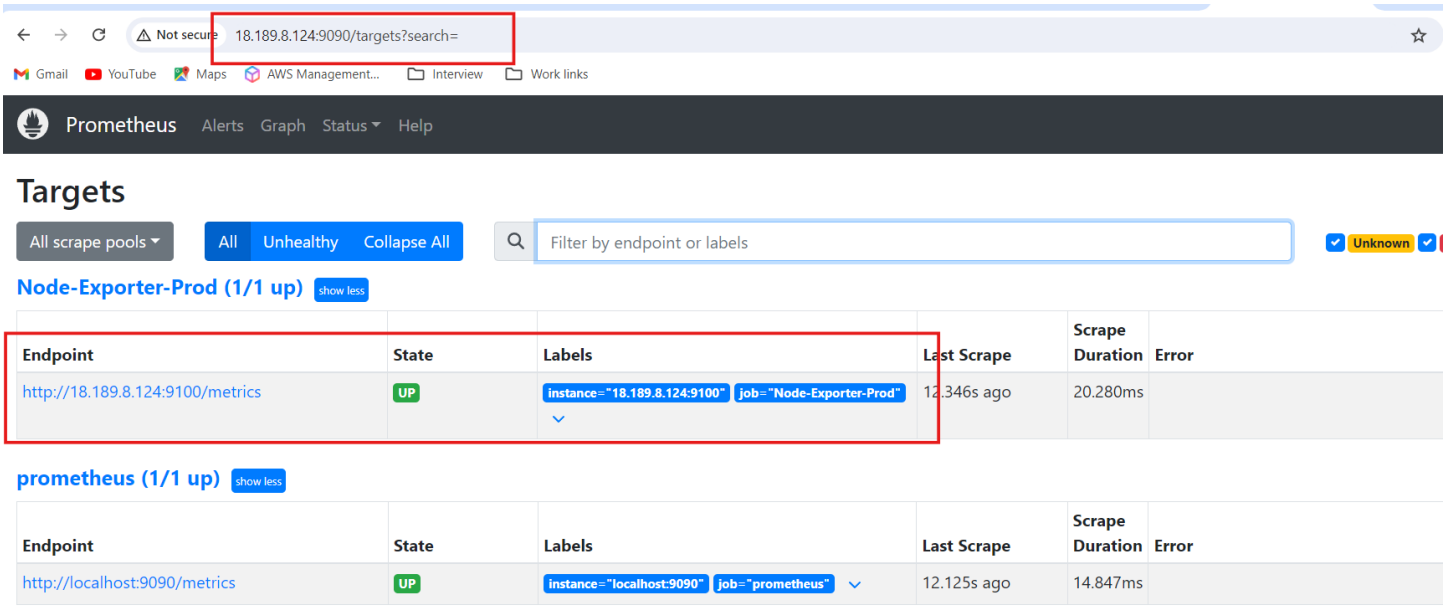




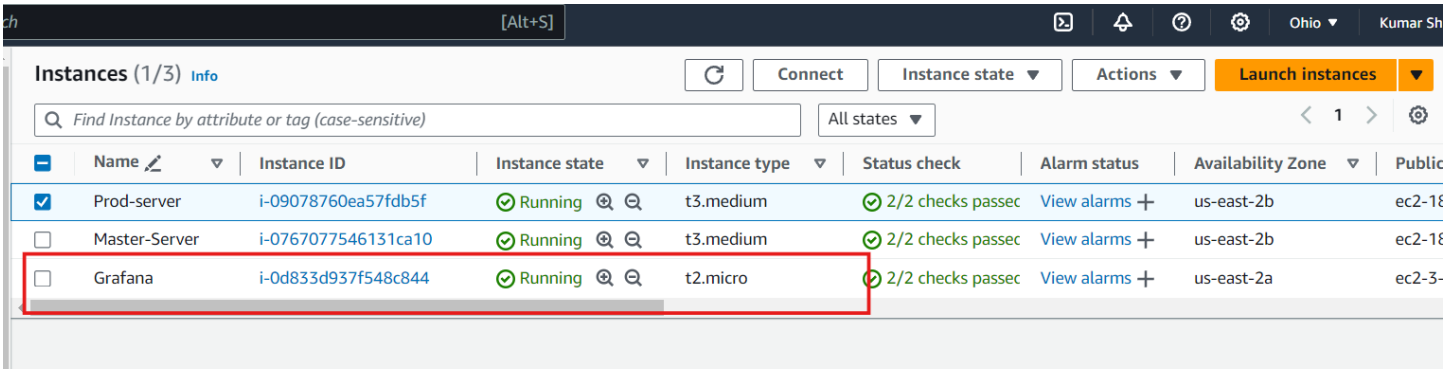
Accessing node exporter



Accessing Prometheus server in Prod.



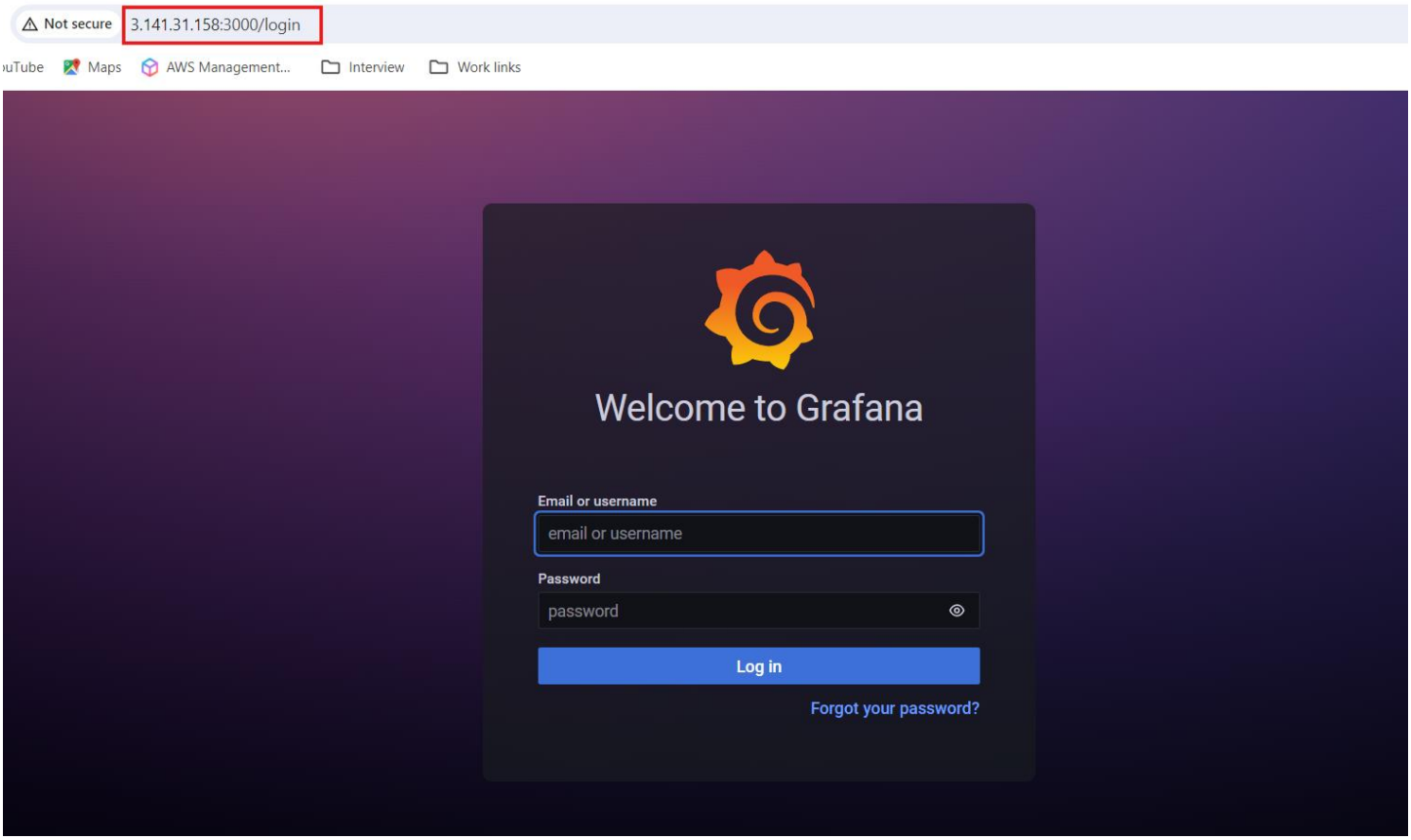
Step 8: Now lets configure grafana server.

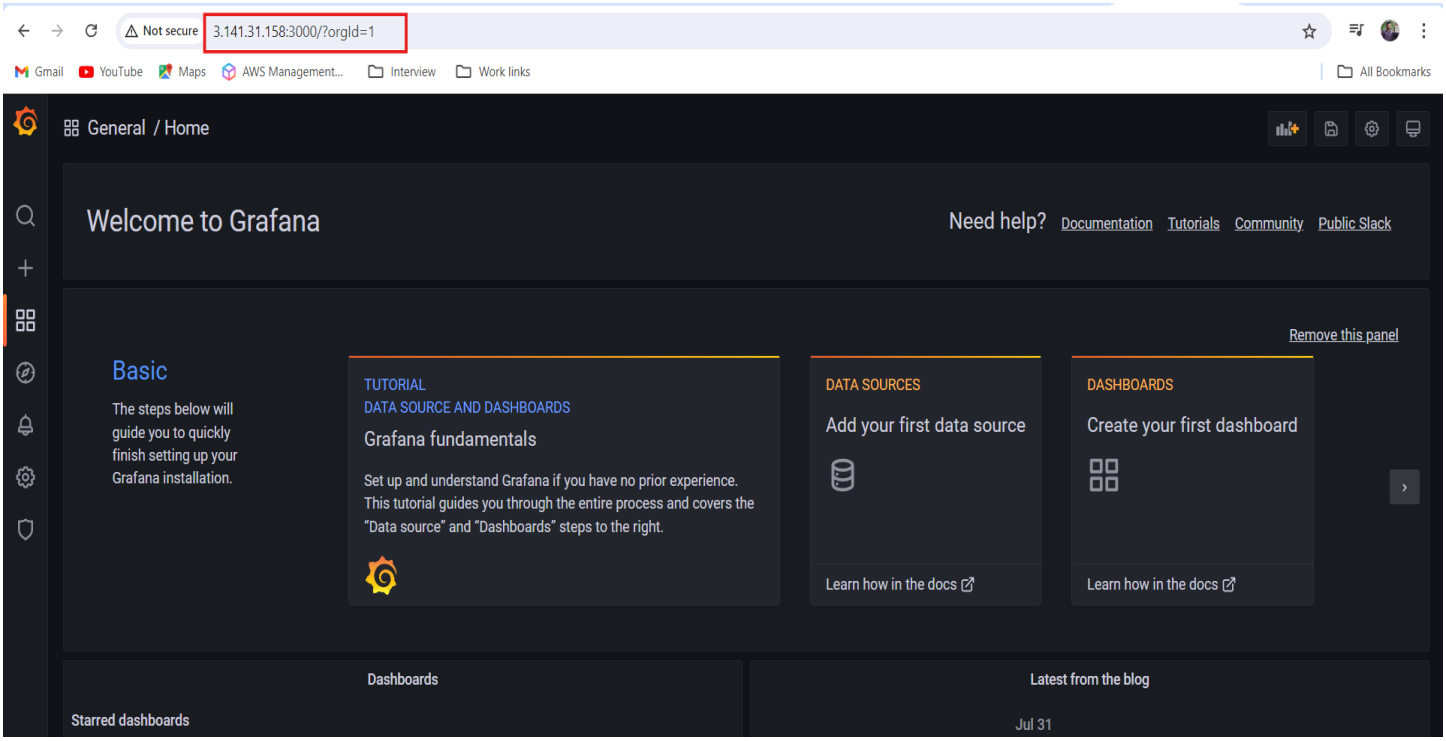


Launch instance and install grafana

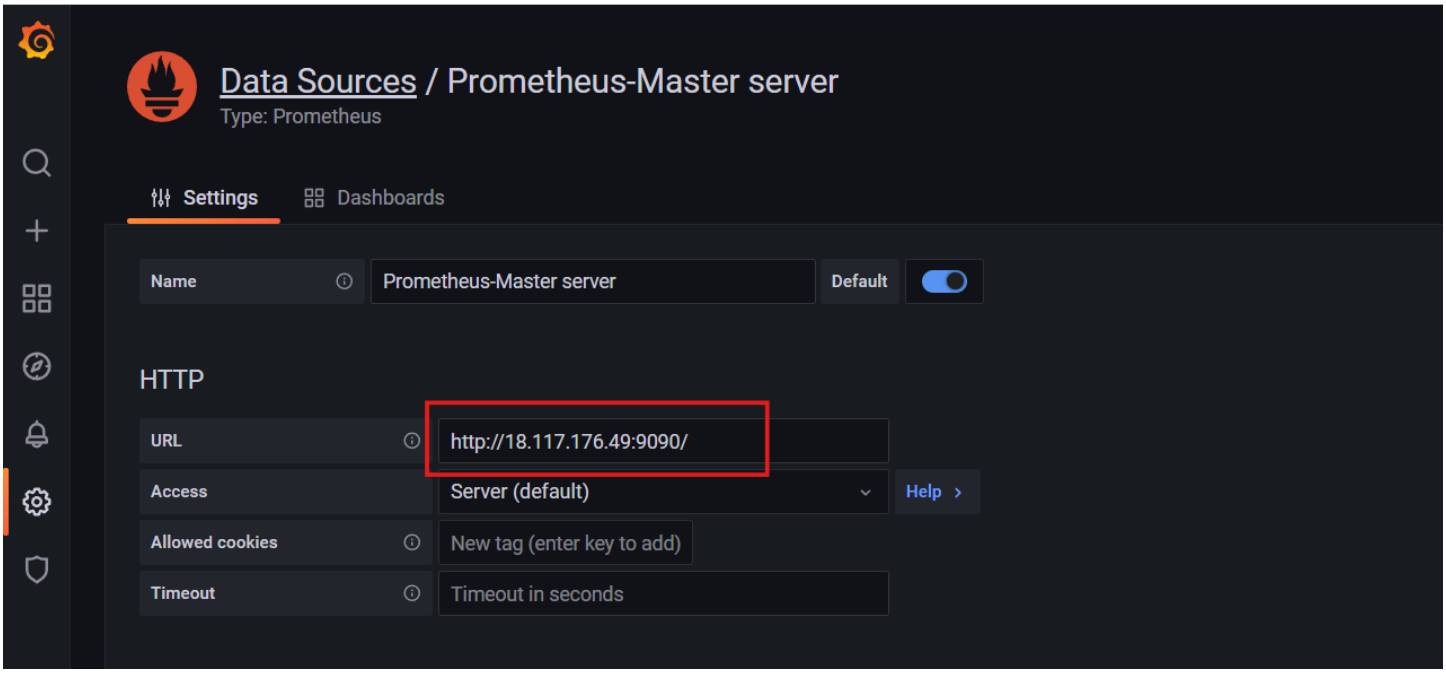
```
root@ip-172-31-12-167:/home/ubuntu/grafana-8.4.4# ./bin/grafana-server
Grafana server is running with elevated privileges. This is not recommended
INFO[08-02|06:06:39] Starting Grafana                         logger=settings version=8.4.4 commit=fcfb01fae branch=HEAD co
WARN[08-02|06:06:39] falling back to legacy setting of 'min_interval_seconds'; please use the configuration option in the `
alerts are enabled. logger=settings
INFO[08-02|06:06:39] Config loaded from                       logger=settings file=/home/ubuntu/grafana-8.4.4/conf/defaults
INFO[08-02|06:06:39] Path Home                                logger=settings path=/home/ubuntu/grafana-8.4.4
INFO[08-02|06:06:39] Path Data                                logger=settings path=/home/ubuntu/grafana-8.4.4/data
INFO[08-02|06:06:39] Path Logs                                logger=settings path=/home/ubuntu/grafana-8.4.4/data/log
```

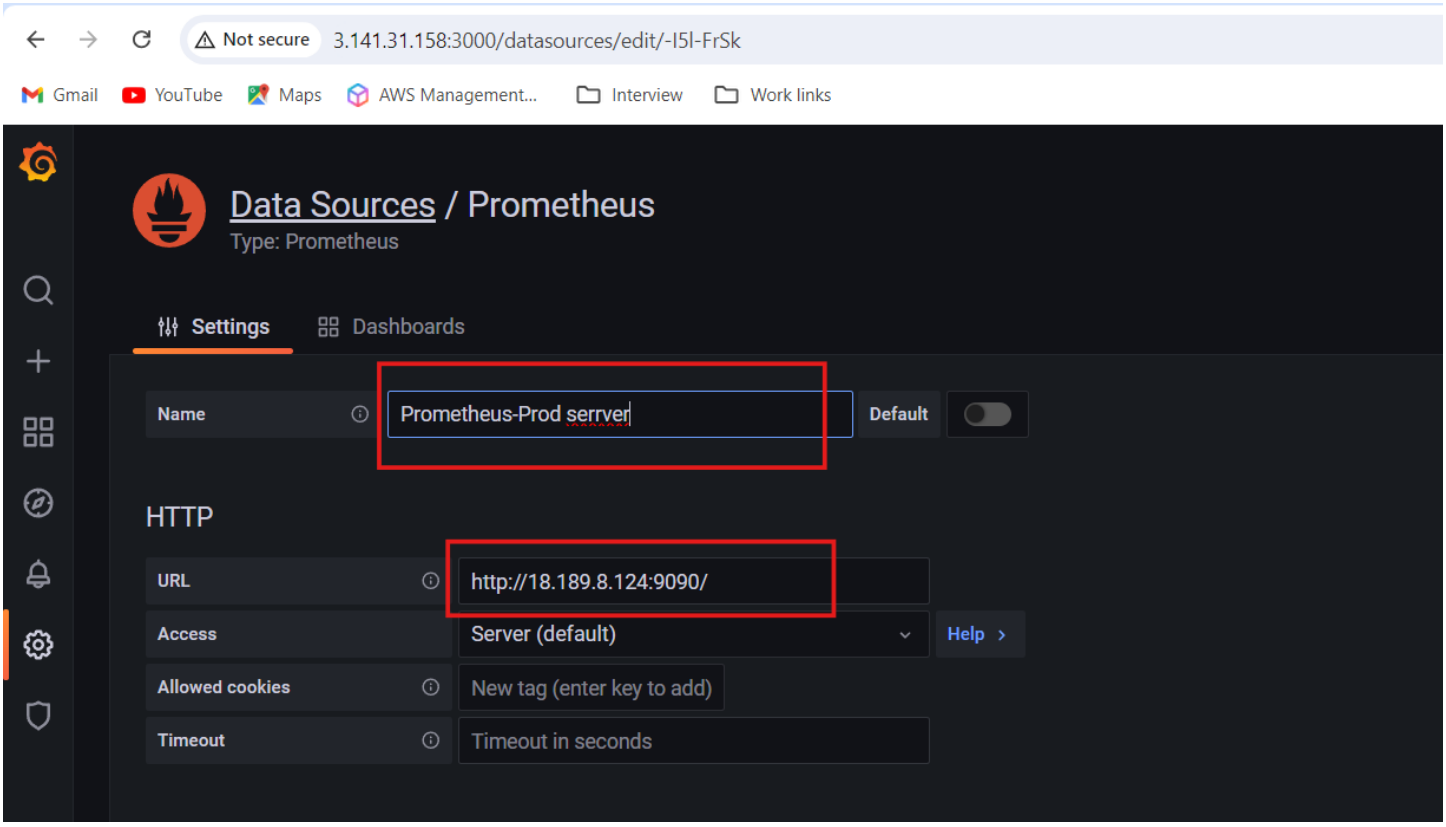
Access grafana server using port 3000



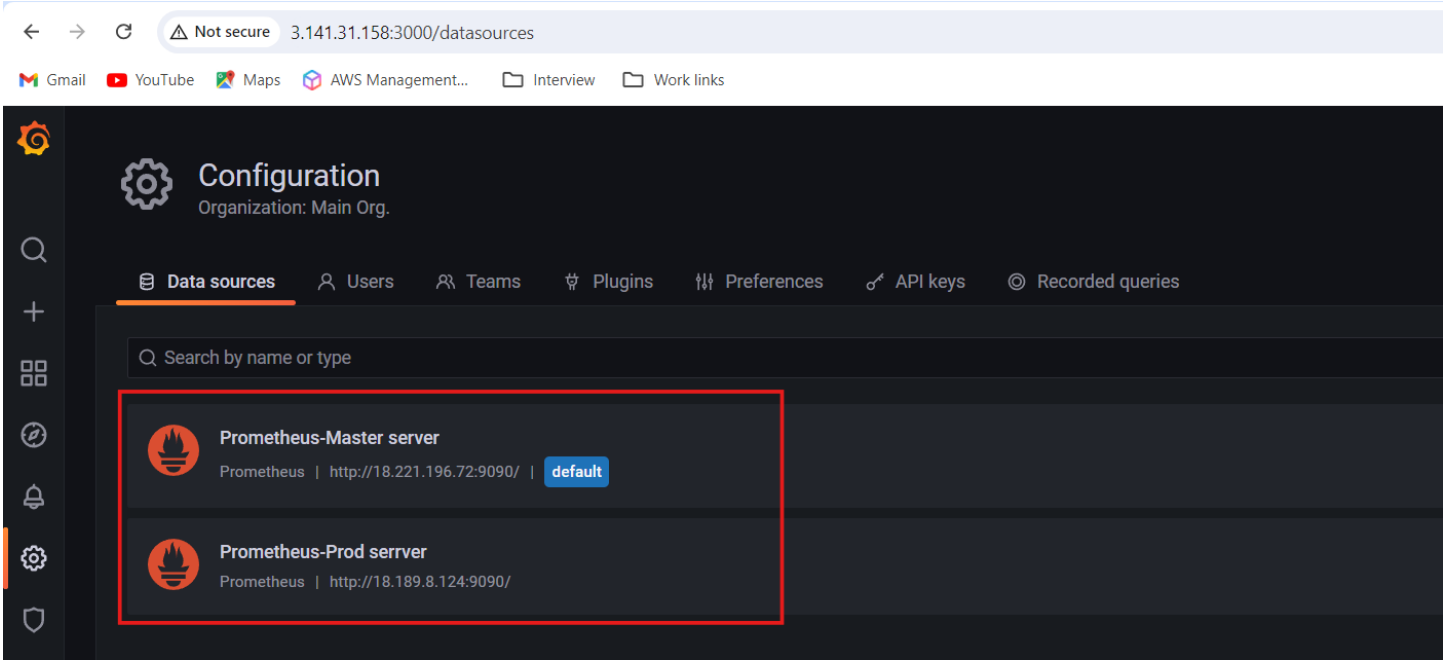


Now configure the Prometheus data source of both the server in grafana



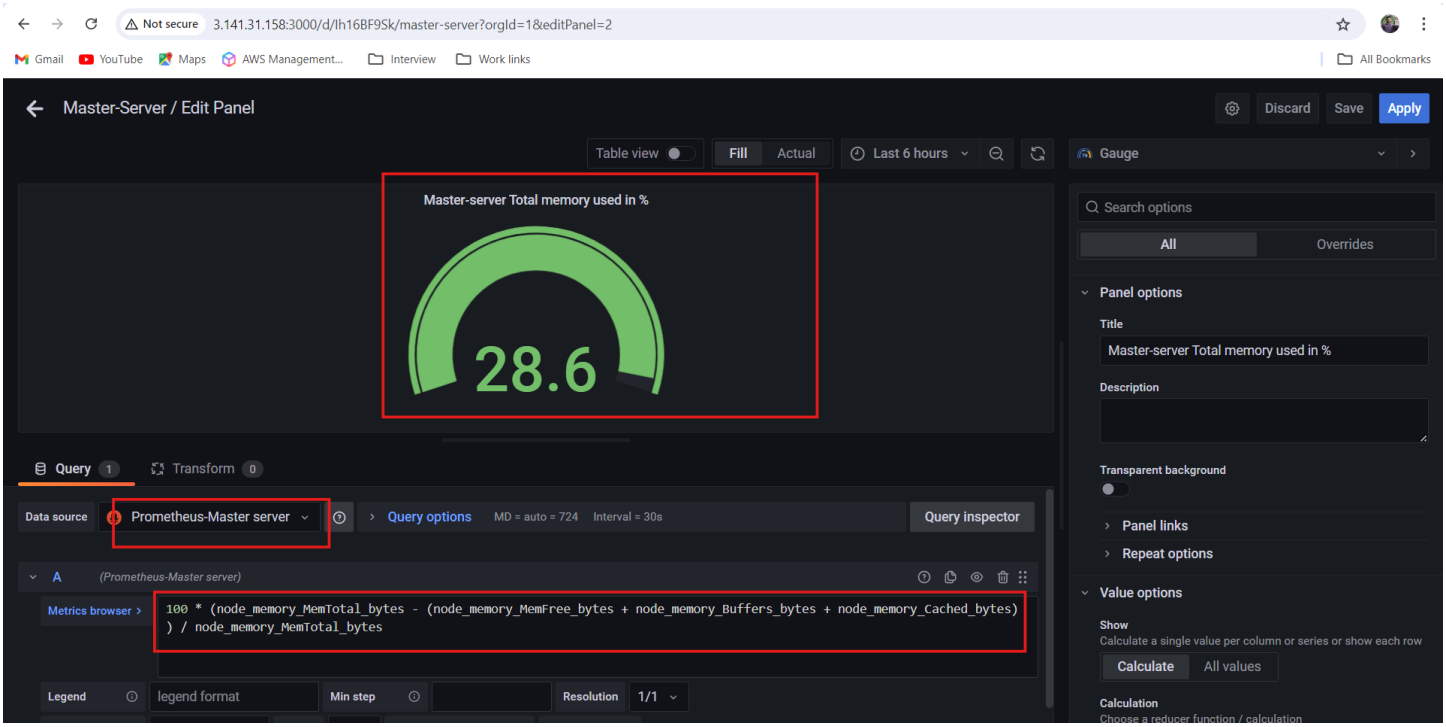


Both server data source has been added

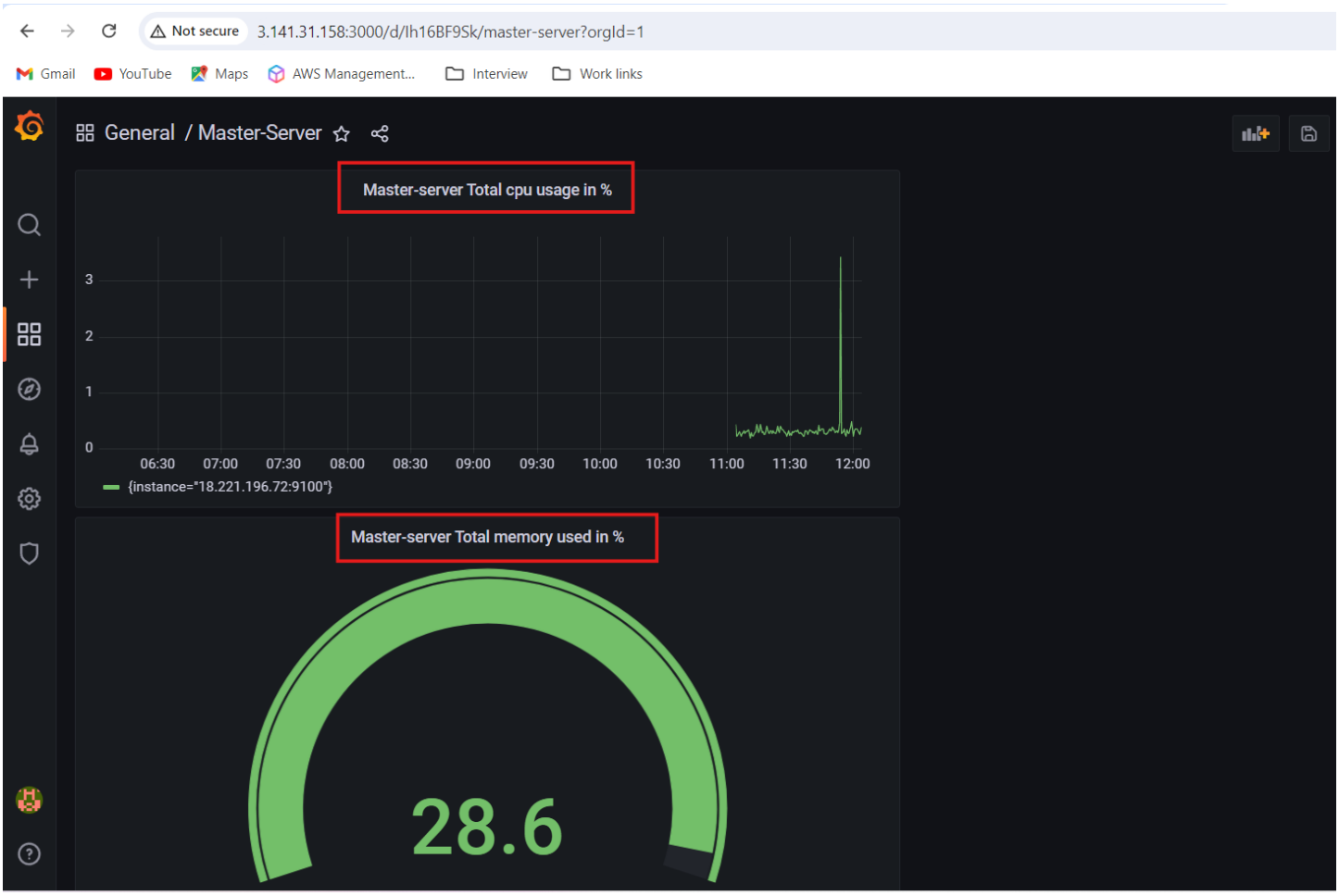


Now configure the dashboard for both the server.

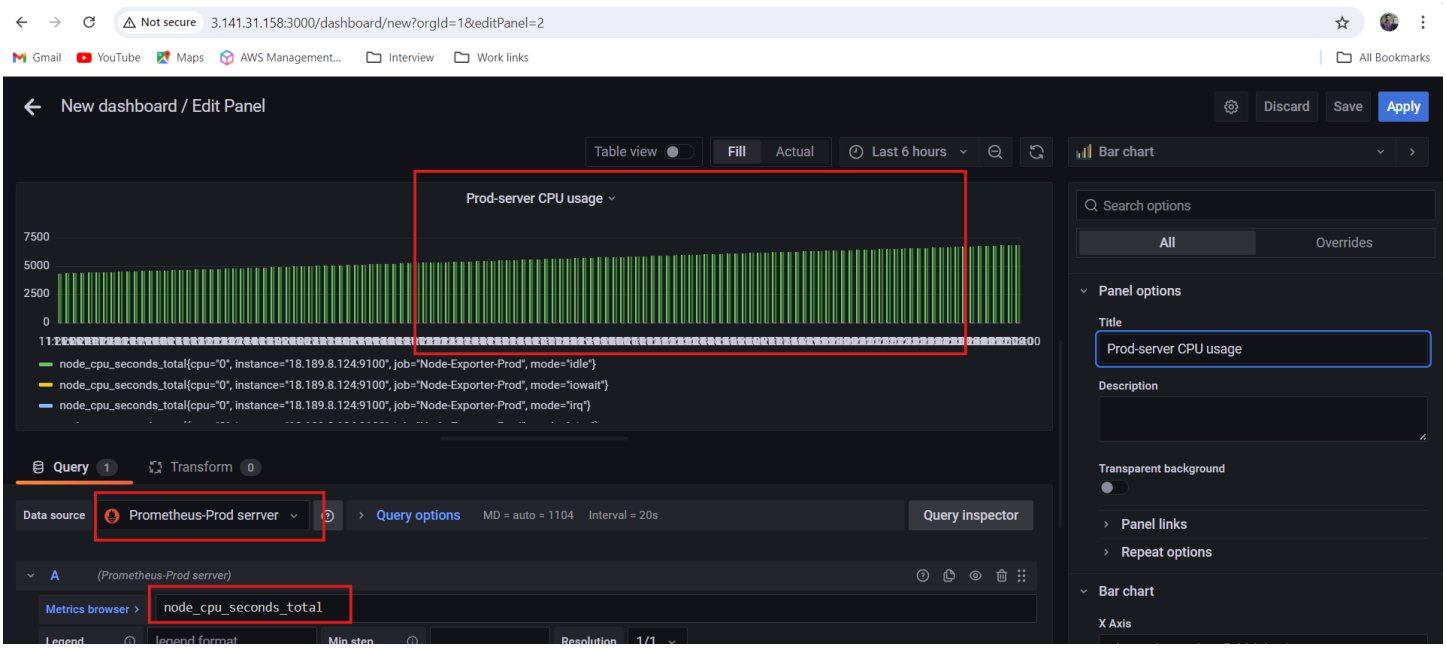
Master server



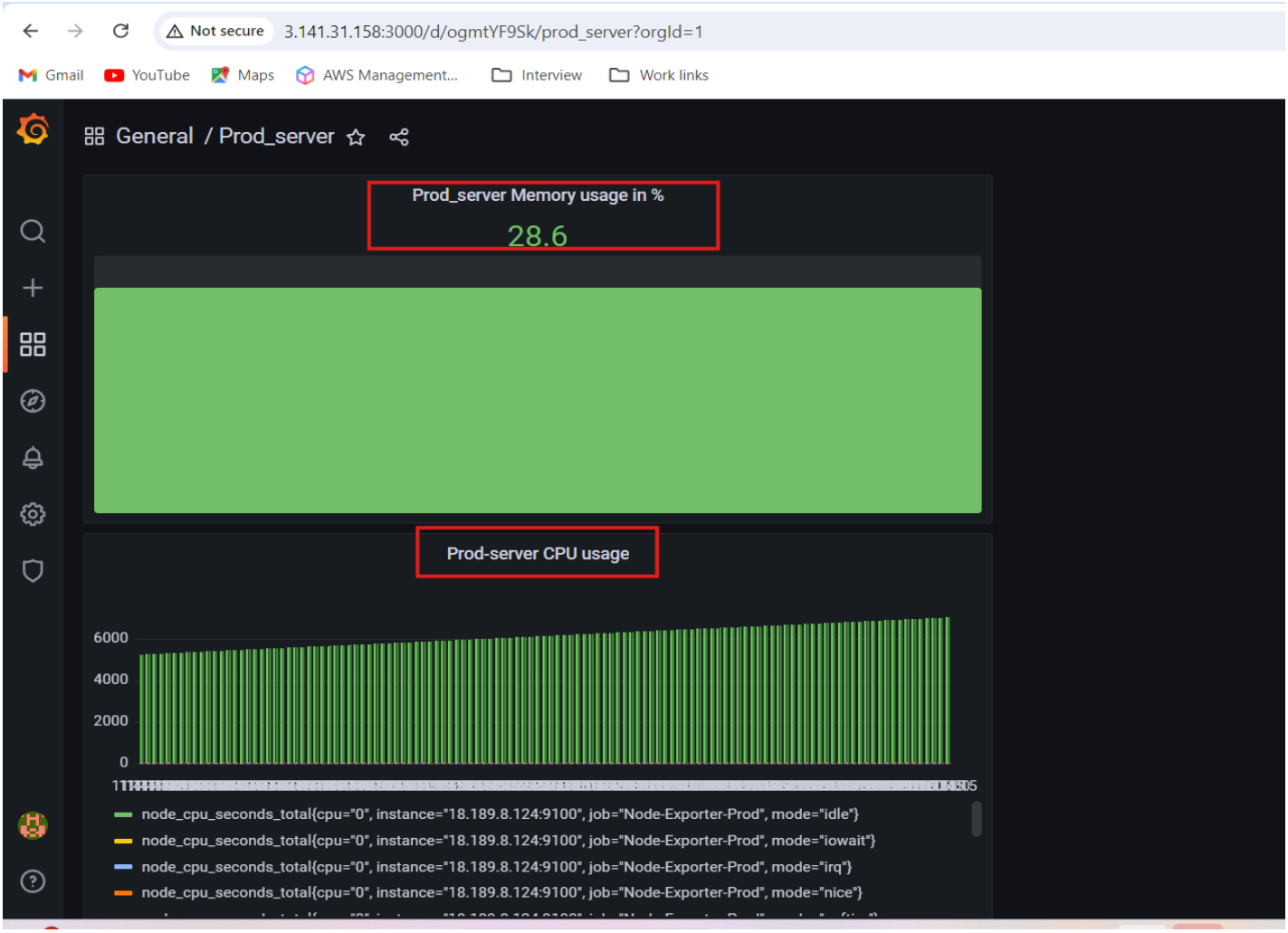
Master server cpu and memory usage graph



Prod server



Prod Server cpu and memory usage graph



!!!!!!!!!!!!.....We have successfully completed “FinanceMe” project .....!!!!!!!!!!!!!!!!