

Module - Capstone Project

Banking and Finance Domain Project

Submitted by:- Jonna Padmarao

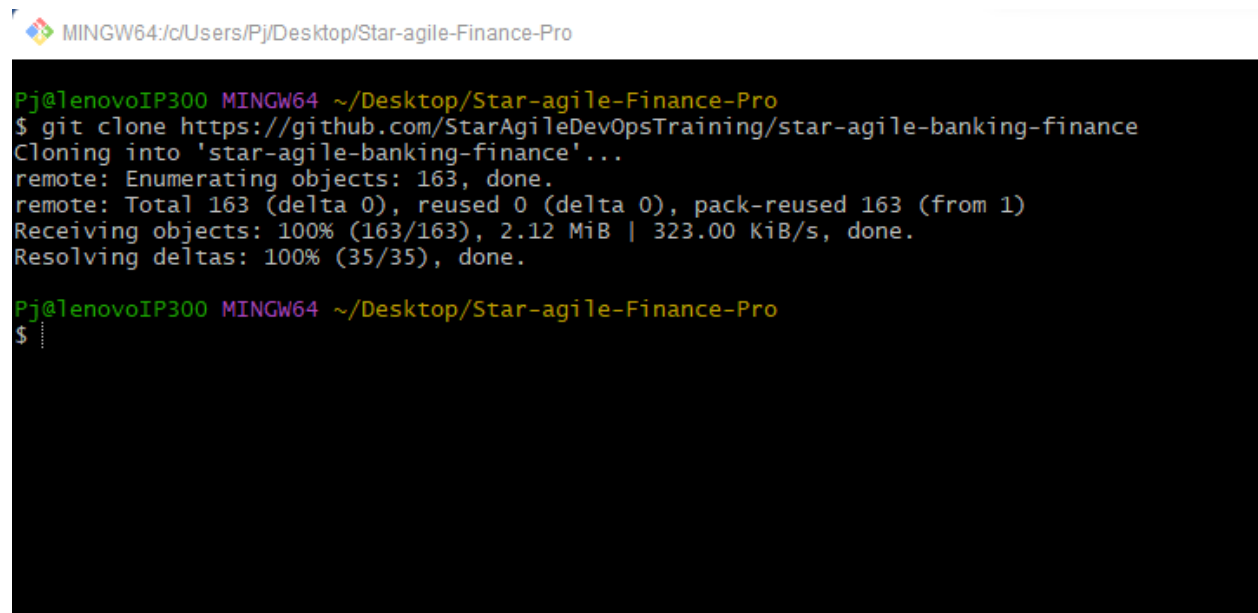
Submission Date:- 17/05/2025

Resubmission Date:-

Step1:- On the desktop create a new folder (star-agile-finance-pro) and enter into that folder and open the git bash in that folder

Step2:- Now give git clone

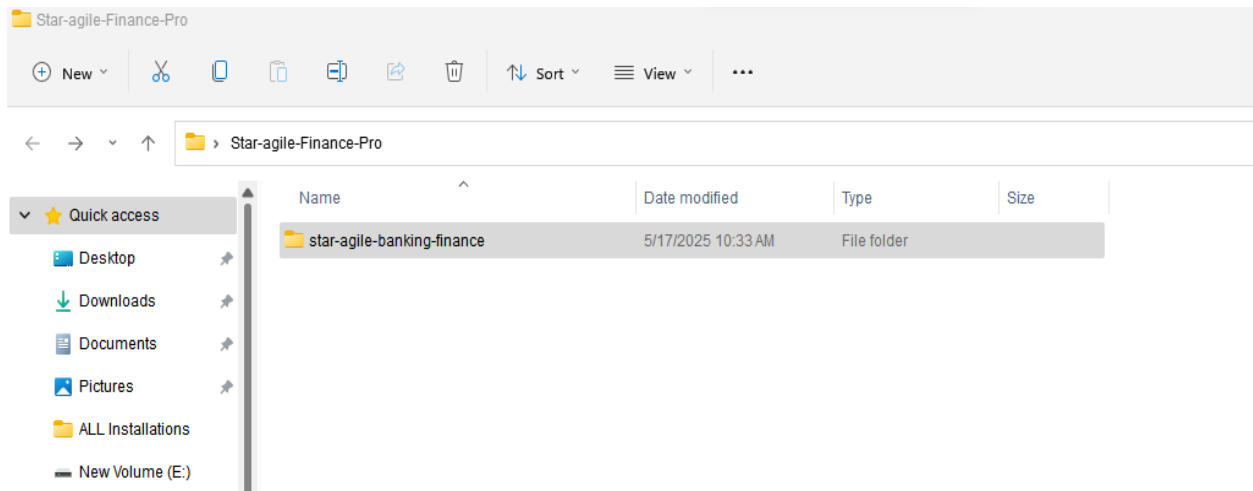
<https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance>
to get the project code in to that folder



```
MINGW64:/c/Users/Pj/Desktop/Star-agile-Finance-Pro

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro
$ git clone https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance
Cloning into 'star-agile-banking-finance'...
remote: Enumerating objects: 163, done.
remote: Total 163 (delta 0), reused 0 (delta 0), pack-reused 163 (from 1)
Receiving objects: 100% (163/163), 2.12 MiB | 323.00 KiB/s, done.
Resolving deltas: 100% (35/35), done.

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro
$
```



Step3:- Now go to the folder that we get from git clone and again open git bash there and check the origin and remove that origin

git remote -v → To get origin list

git remote remove origin → to remove the origin

```
MINGW64:/c/Users/Pj/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance

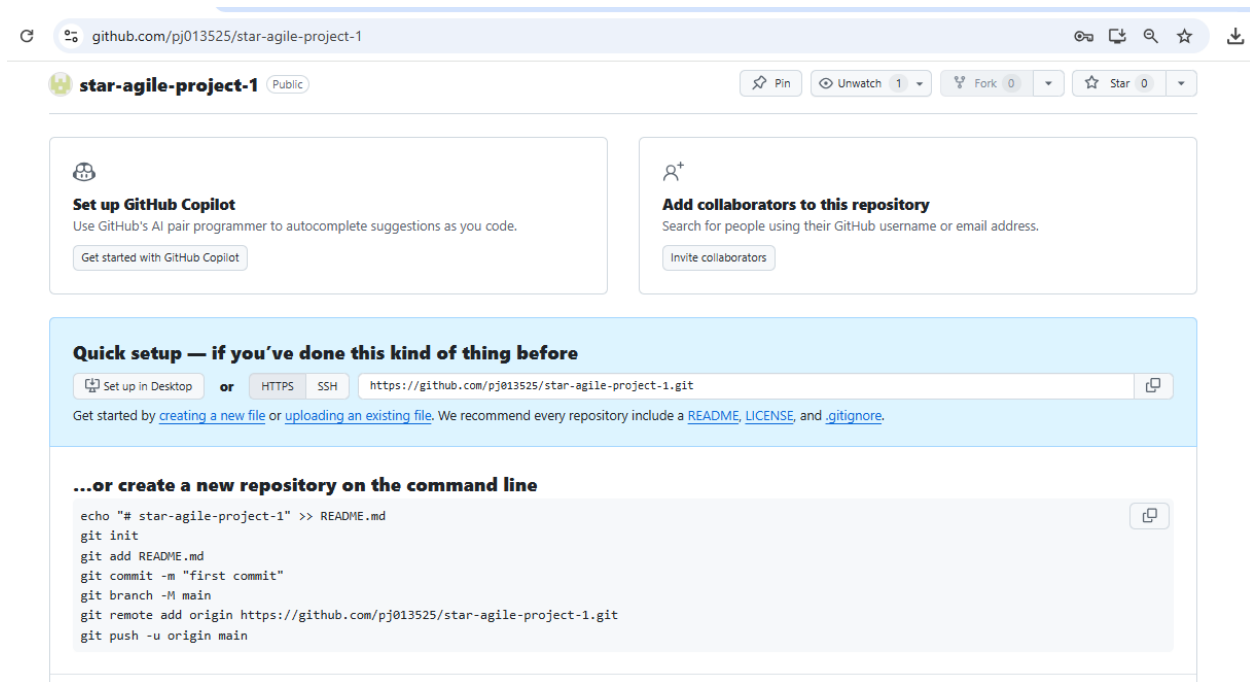
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote -v
origin https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance (fetch)
origin https://github.com/StarAgileDevOpsTraining/star-agile-banking-finance (push)

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote remove origin

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote -v

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$
```

Step4:- Now go to github and create a new repo and copy the url in the gitbash



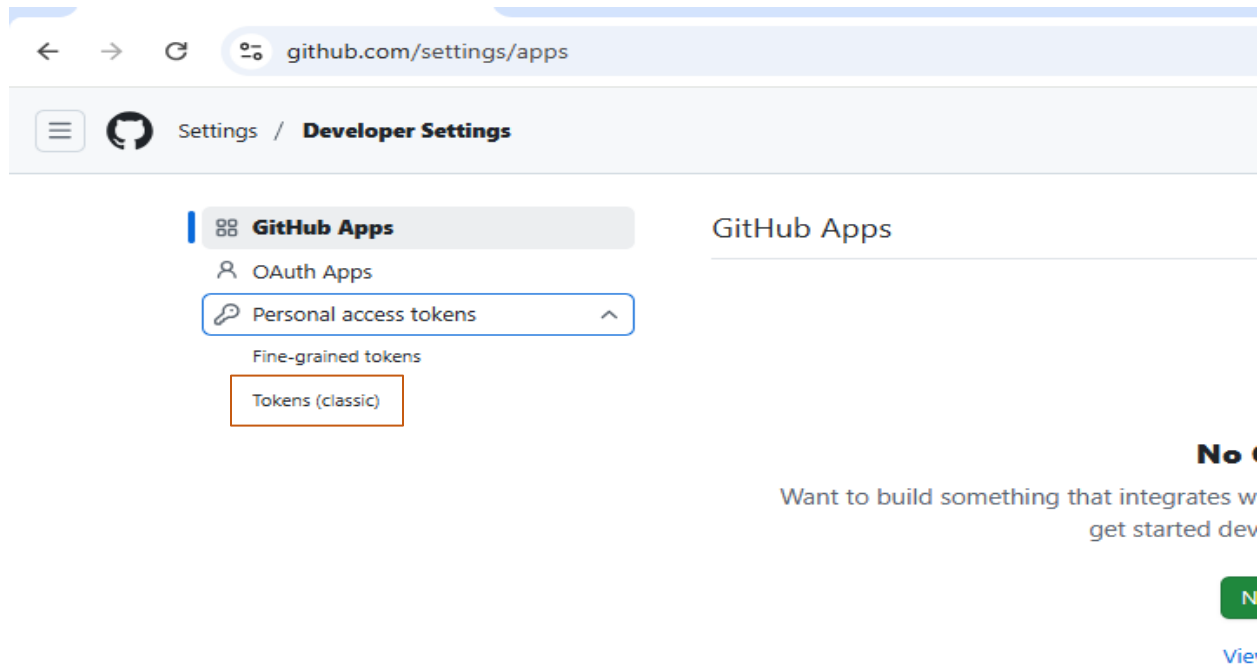
Step5:- Now again go to the gitbash and add this git repo url in the project by using `git remote add <git-repo-url>` and verify

```
MINGW64:/c/Users/Pj/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote add origin https://github.com/pj013525/star-agile-project-1.git

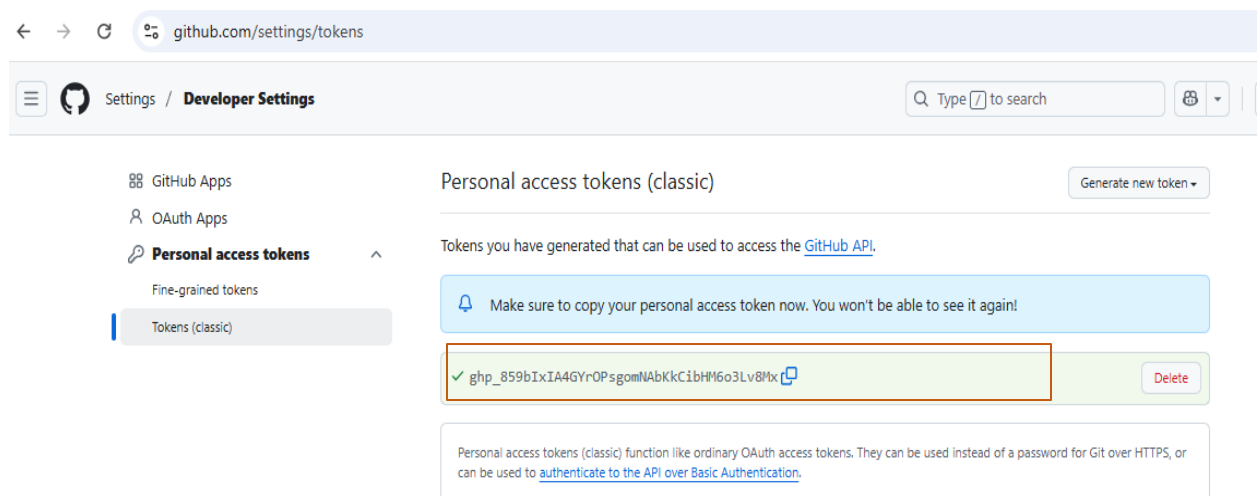
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git remote -v
origin https://github.com/pj013525/star-agile-project-1.git (fetch)
origin https://github.com/pj013525/star-agile-project-1.git (push)

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$
```

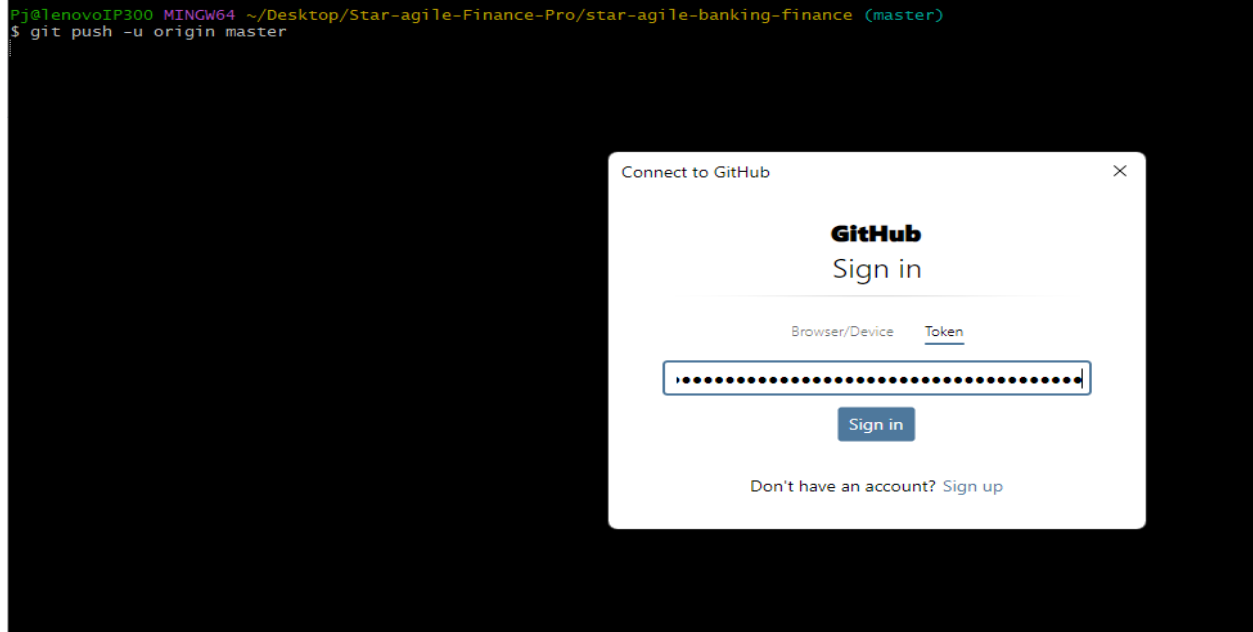
Step6:- Now again go to github → Profile setting → Developer settings → Personal access token → Tokens(classic) →Generate new token



Step7:- Now a token will be generated , copy this token that generated since it is only available for one time only



Step8:- Now give link this the remote repo with gitbash using this token
git push -u origin master and paste the token the copied from the
github and press sign in

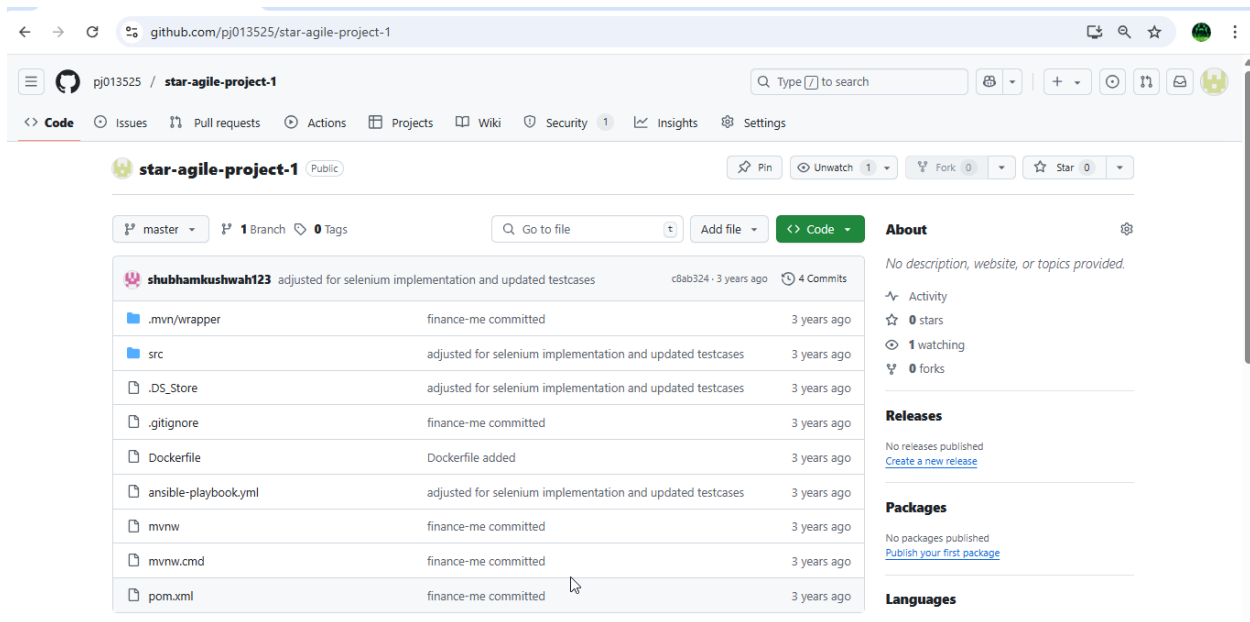


Step9:- Now the master branch will be set to our repo by default

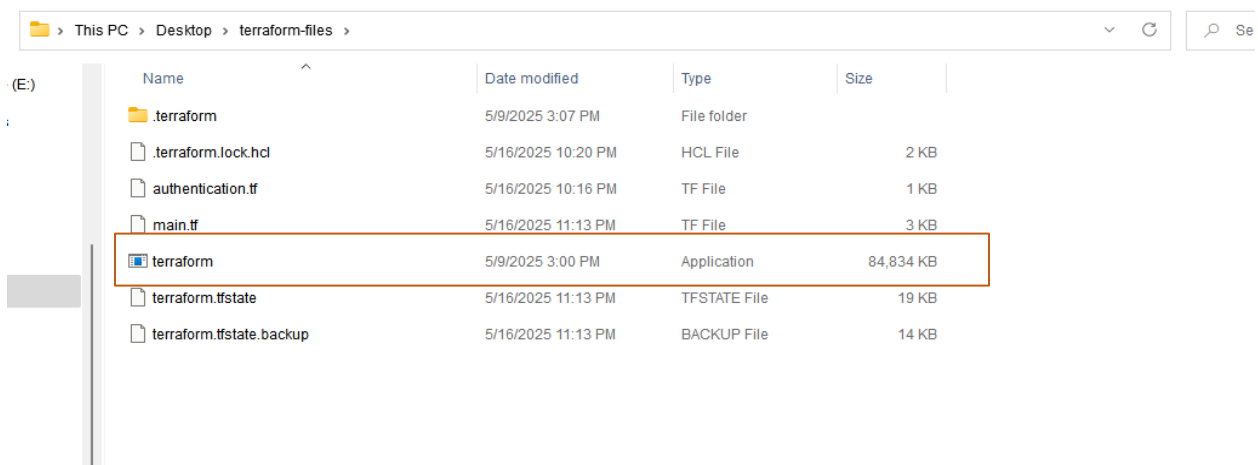
```
Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ git push -u origin master
Enumerating objects: 163, done.
Counting objects: 100% (163/163), done.
Delta compression using up to 4 threads
Compressing objects: 100% (106/106), done.
Writing objects: 100% (163/163), 2.12 MiB | 1.47 MiB/s, done.
Total 163 (delta 35), reused 163 (delta 35), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (35/35), done.
To https://github.com/pj013525/star-agile-project-1.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Pj@lenovoIP300 MINGW64 ~/Desktop/Star-agile-Finance-Pro/star-agile-banking-finance (master)
$ |
```

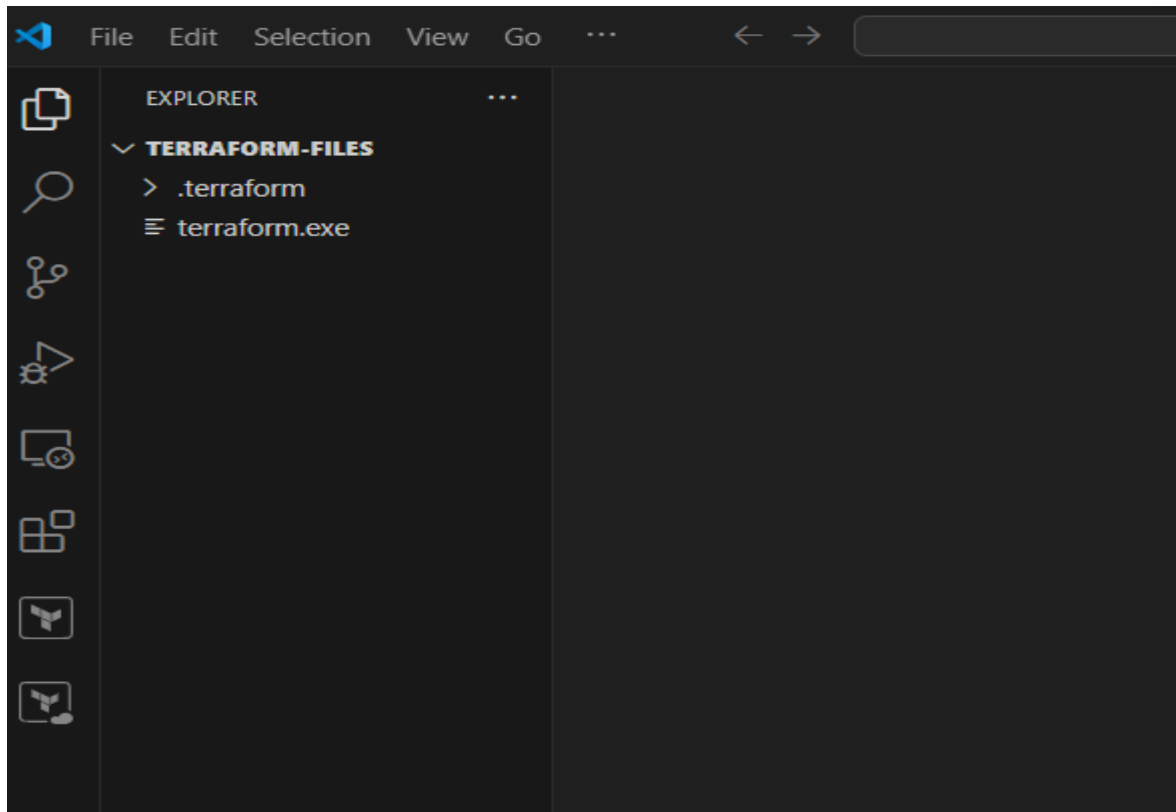
Step10:- Now go to the github repo and you will see the source code in that repo



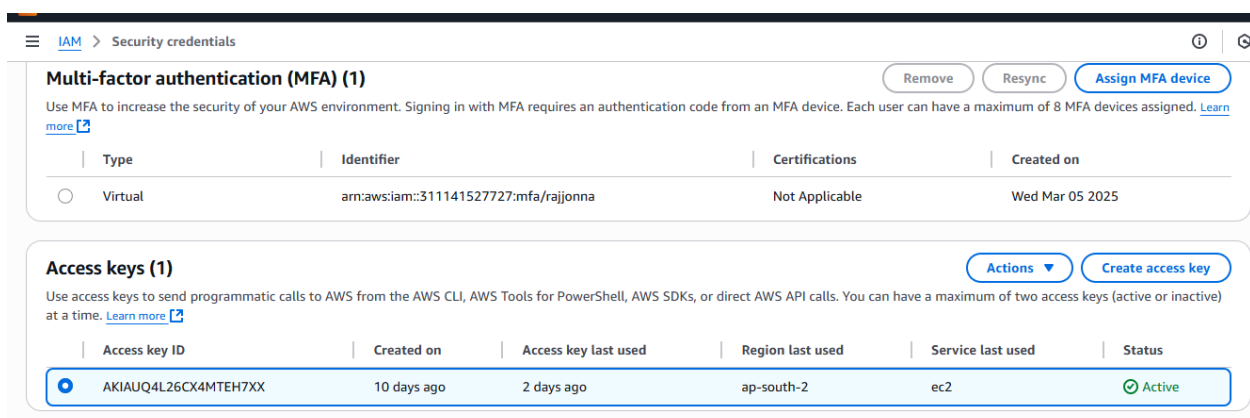
Step11:- Now create an instance using terraform as IaC, and for that create a folder on desktop and go to browser download terraform for windows then a terraform application will be generated, now copy this application into that folder and save



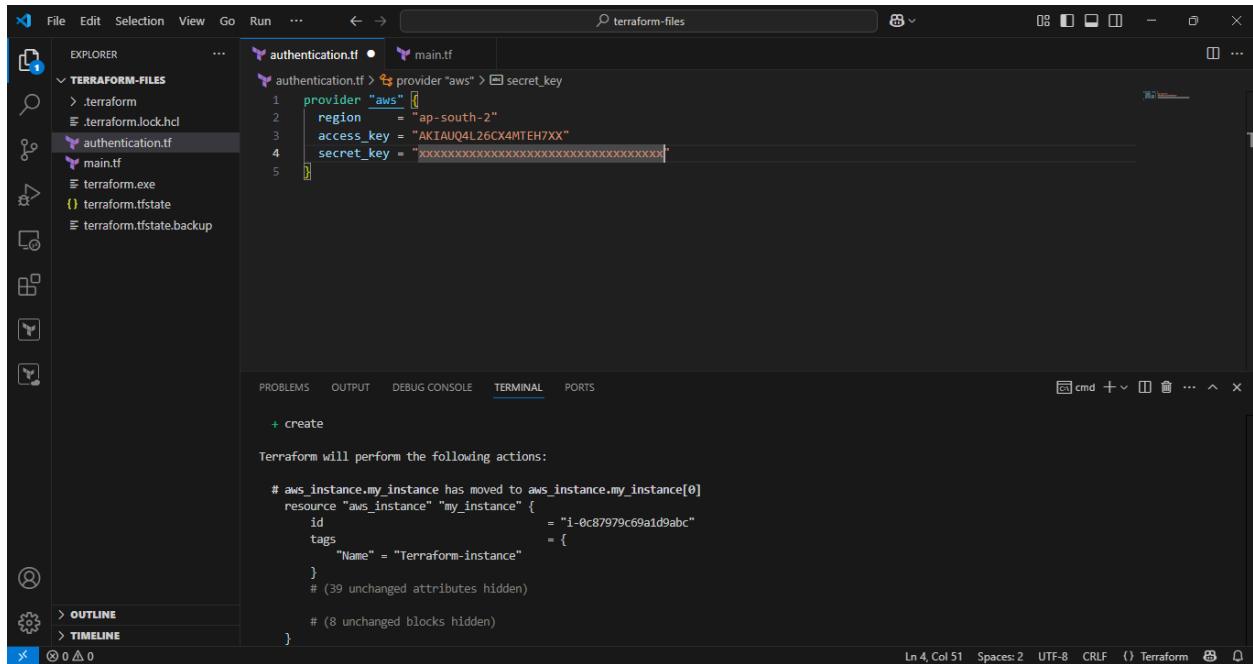
Step12:- open visual studio code and go to terraform folder



Step13:- Now create a file authentication.tf and give the provider and for that select the region in which you want to launch the server and go to aws account and go to profile → credentials and go to access keys



Step14:- now copy this access key details and paste it in this authentication.tf file and initialize it



Step15:- After it is successful now create a new file main.tf and give resources details to create instance


```
main.tf > resource "aws_instance" "my_instance" > count
79 }
80
81 # EC2 Instance
82 resource "aws_instance" "my_instance" {
83   ami           = "ami-053a0835435bf4f45"
84   instance_type = "t3.large"
85   count         = "2"
86   subnet_id     = aws_subnet.my_subnet.id
87   vpc_security_group_ids = [aws_security_group.my_sg.id]
88   key_name      = "new-key" # Use the key already created in AWS
89   associate_public_ip_address = true
90
91   tags = {
92     Name = "Terraform-instance"
93   }
94 }
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

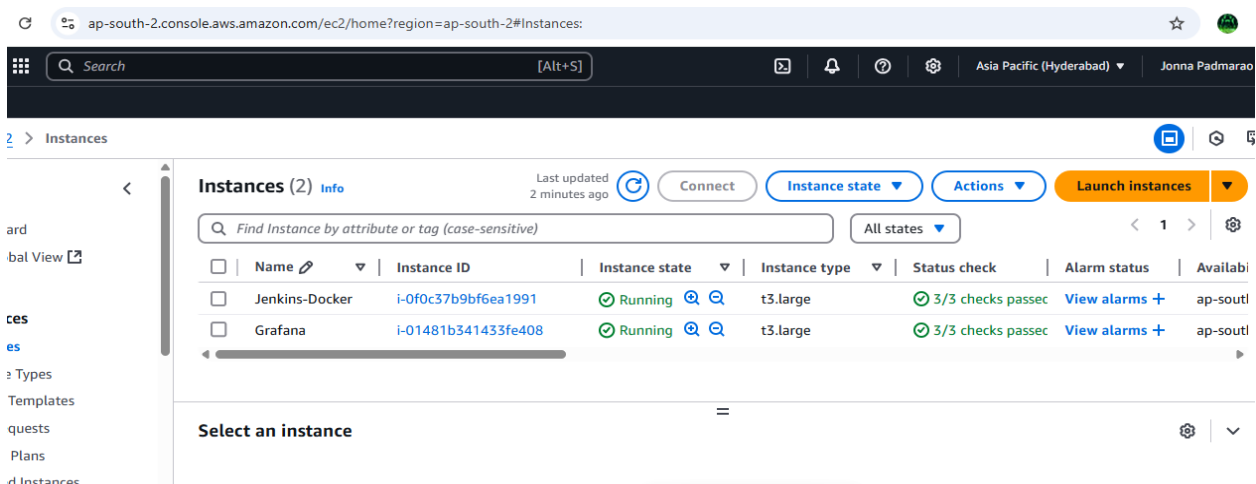
```
+ root_block_device (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.
aws_instance.my_instance[1]: Creating...
aws_instance.my_instance[0]: Creating...
aws_instance.my_instance[0]: Still creating... [10s elapsed]
aws_instance.my_instance[1]: Still creating... [10s elapsed]
aws_instance.my_instance[1]: Creation complete after 14s [id=i-01481b341433fe408]
aws_instance.my_instance[0]: Creation complete after 14s [id=i-0f0c37b9bf6ea1991]

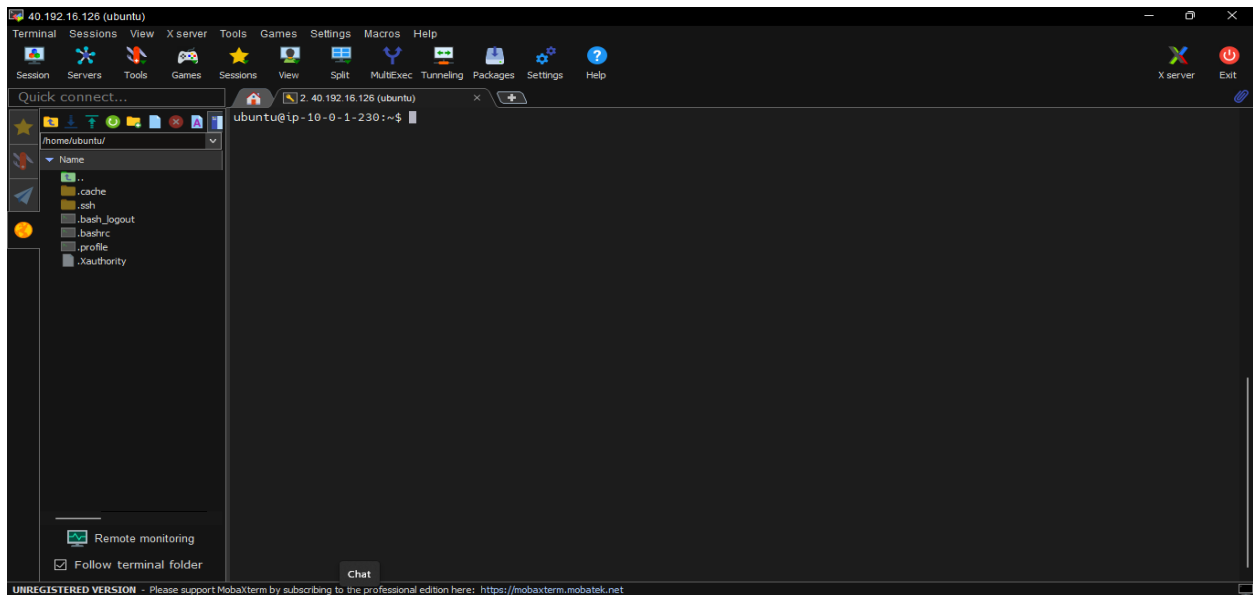
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

C:\Users\Pj\Desktop\terraform-files>
```

Step16:- After it is successful go and check the aws console and rename them as Jenkins-docker and Grafana instances



Step17:- Now connect to Jenkins-Docker server using MobaXterm agent and launch an instance

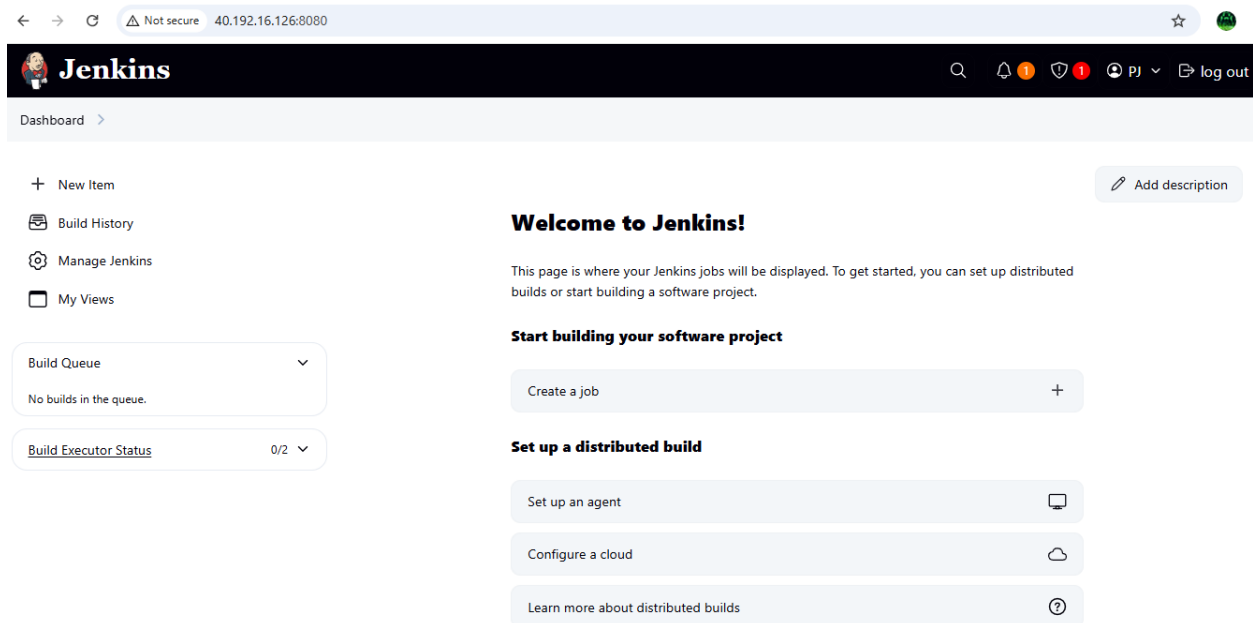


Step18:- Now install java and Jenkins and docker in this is sever and enable All traffic in the security group of this server and add Jenkins group to docker and give root permissions to the Jenkins user in the sudoers file as under root give

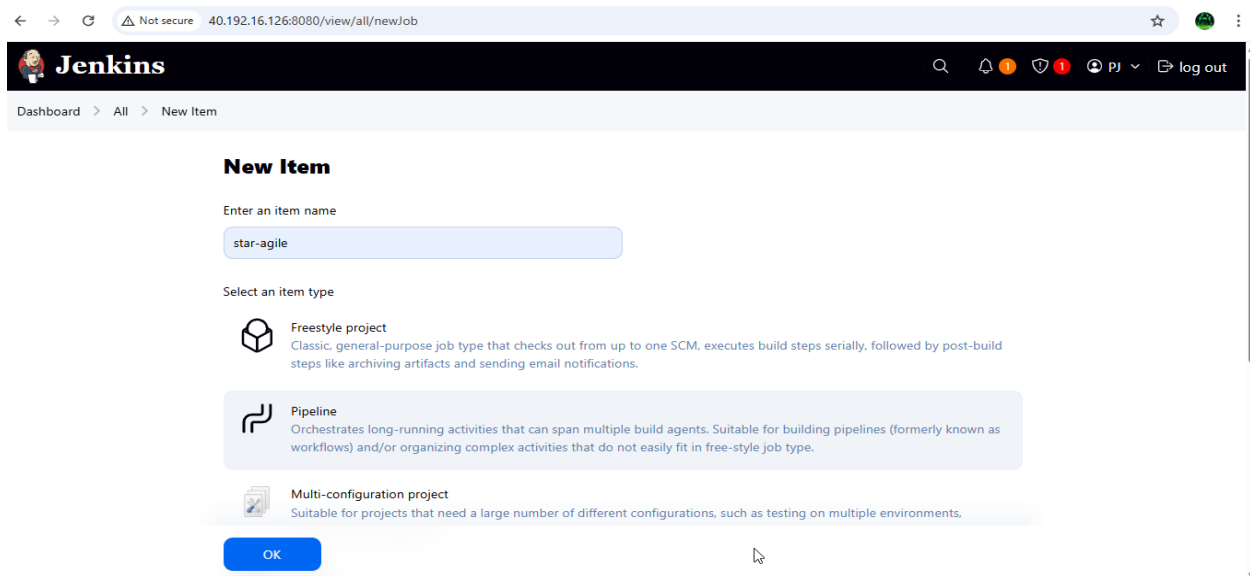
jenkins ALL=(ALL:ALL) NOPASSWD: ALL restart the jenkins

```
root@ip-10-0-1-230:/home/ubuntu# docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
root@ip-10-0-1-230:/home/ubuntu# jenkins --version
2.504.1
root@ip-10-0-1-230:/home/ubuntu# java --version
openjdk 17.0.15 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu124.04, mixed mode, sharing)
root@ip-10-0-1-230:/home/ubuntu# mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 17.0.15, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1024-aws", arch: "amd64", family: "unix"
root@ip-10-0-1-230:/home/ubuntu# git --version
git version 2.43.0
root@ip-10-0-1-230:/home/ubuntu# visudo
root@ip-10-0-1-230:/home/ubuntu# service jenkins restart
root@ip-10-0-1-230:/home/ubuntu#
```

Step19:- Go to the any browser and give the details and click on recommended plugins and login to the Jenkins



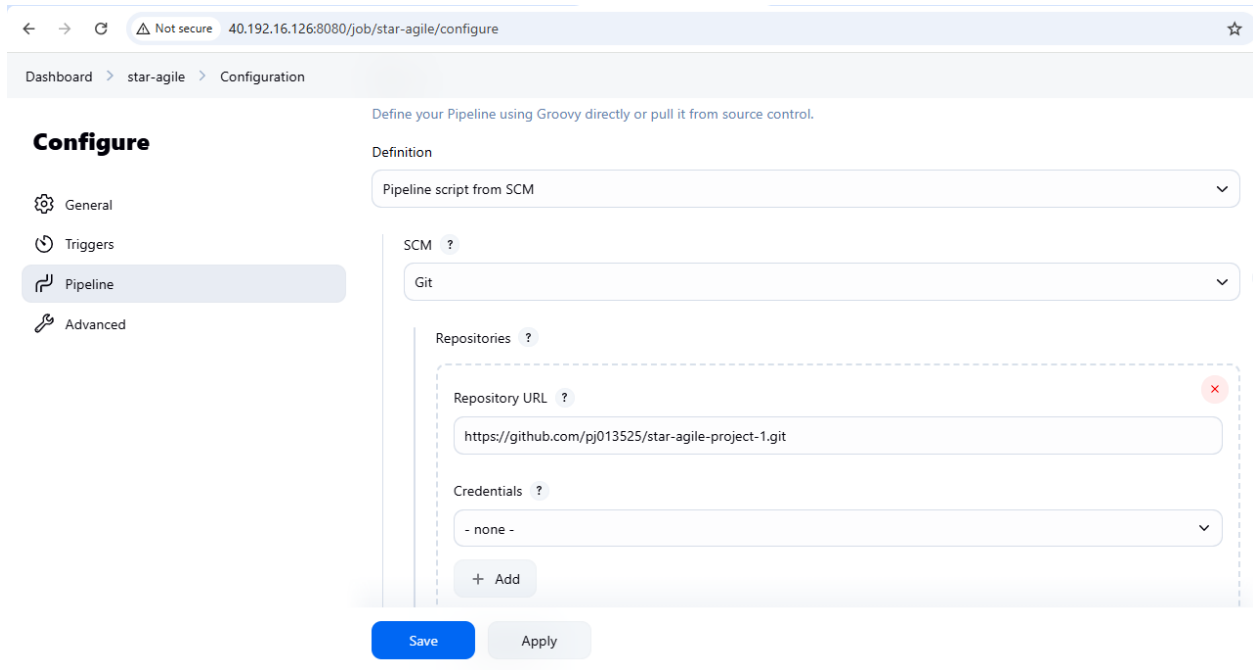
Step20:- Now in the Jenkins dashboard click on new item and give any name and select pipeline project as type and click on ok



Step21:- Now go to github repo and create a new file with name as Jenkinsfile and press commit changes

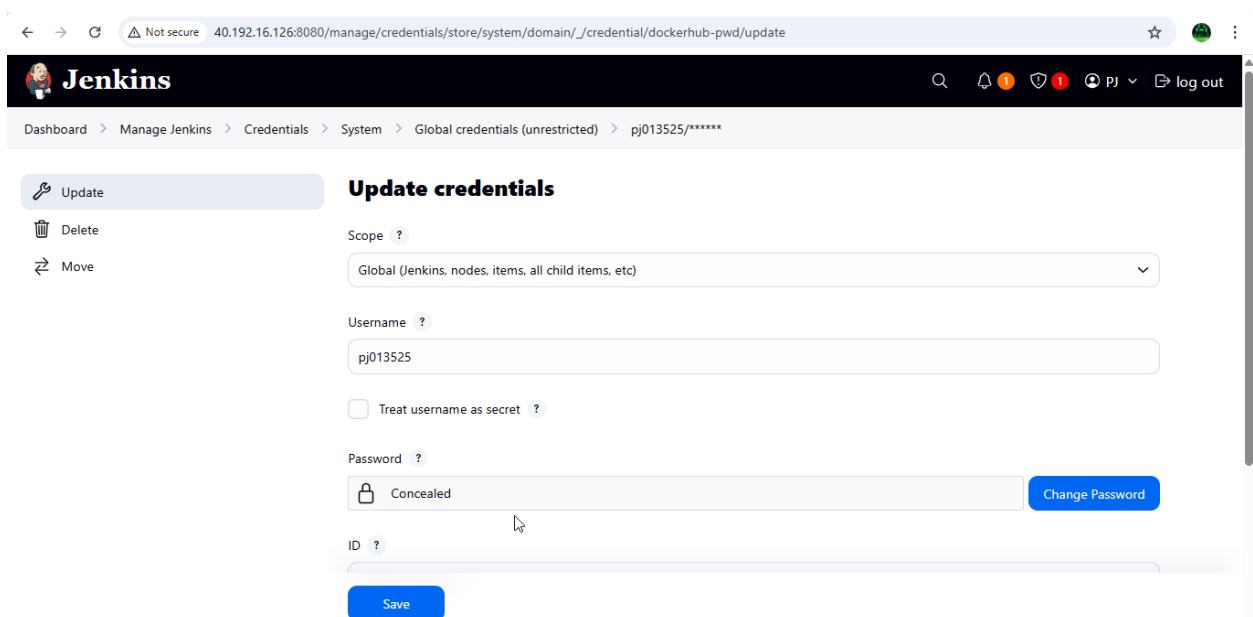


Step22:- Now go to project in the Jenkins and in the pipeline and do as follows and apply and save



The screenshot shows the Jenkins 'Configure' page for a pipeline named 'star-agile'. The left sidebar has tabs for 'General', 'Triggers', 'Pipeline', and 'Advanced', with 'Pipeline' selected. The main area is titled 'Configure' and contains a 'Definition' dropdown set to 'Pipeline script from SCM'. Below this, the 'SCM' dropdown is set to 'Git'. A 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/pj013525/star-agile-project-1.git' and a 'Credentials' dropdown set to '- none -'. At the bottom are 'Save' and 'Apply' buttons.

Step23:- Now in dash board → manage Jenkins → credentials → global → add credentials and give dockerhub user name and password and click on create



The screenshot shows the Jenkins 'Update credentials' page. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Credentials', 'System', and 'Global credentials (unrestricted)'. The left sidebar has buttons for 'Update', 'Delete', and 'Move', with 'Update' selected. The main area is titled 'Update credentials' and contains a 'Scope' dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)'. Below this, the 'Username' field contains 'pj013525'. There is a checkbox for 'Treat username as secret' which is unchecked. The 'Password' field is labeled 'Concealed' and has a 'Change Password' button next to it. At the bottom is a 'Save' button.

Step24:- Now install docker and other required plugins in the Jenkins

Pipeline

Git Plugin

Docker Pipeline Plugin

Credentials Binding Plugin

Docker Commons Plugin

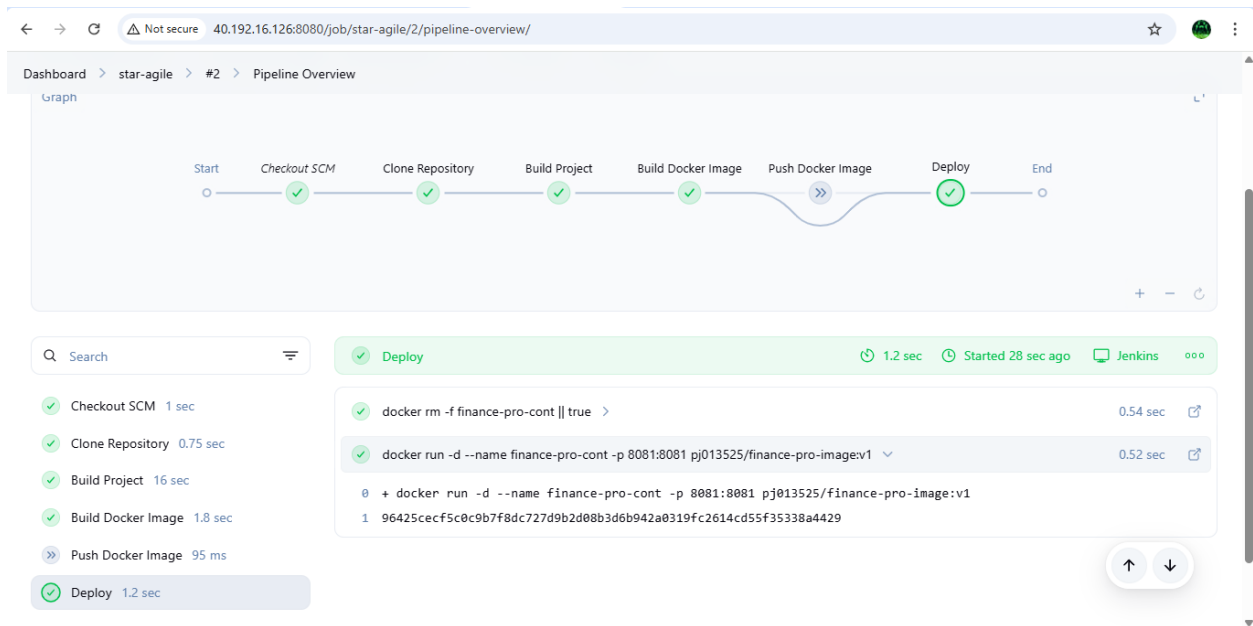
Pipeline: GitHub

Maven Integration Plugin

The screenshot shows the Jenkins 'Download progress' page. The browser address bar indicates the URL is 40.192.16.126:8080/manage/pluginManager/updates/. The page has a sidebar with navigation links: Updates, Available plugins, Installed plugins, Advanced settings, and Download progress (which is highlighted). The main content area is titled 'Download progress' and shows a list of plugins with their download status. The status is indicated by a green checkmark for 'Success' and a blue circle with three dots for 'Pending'.

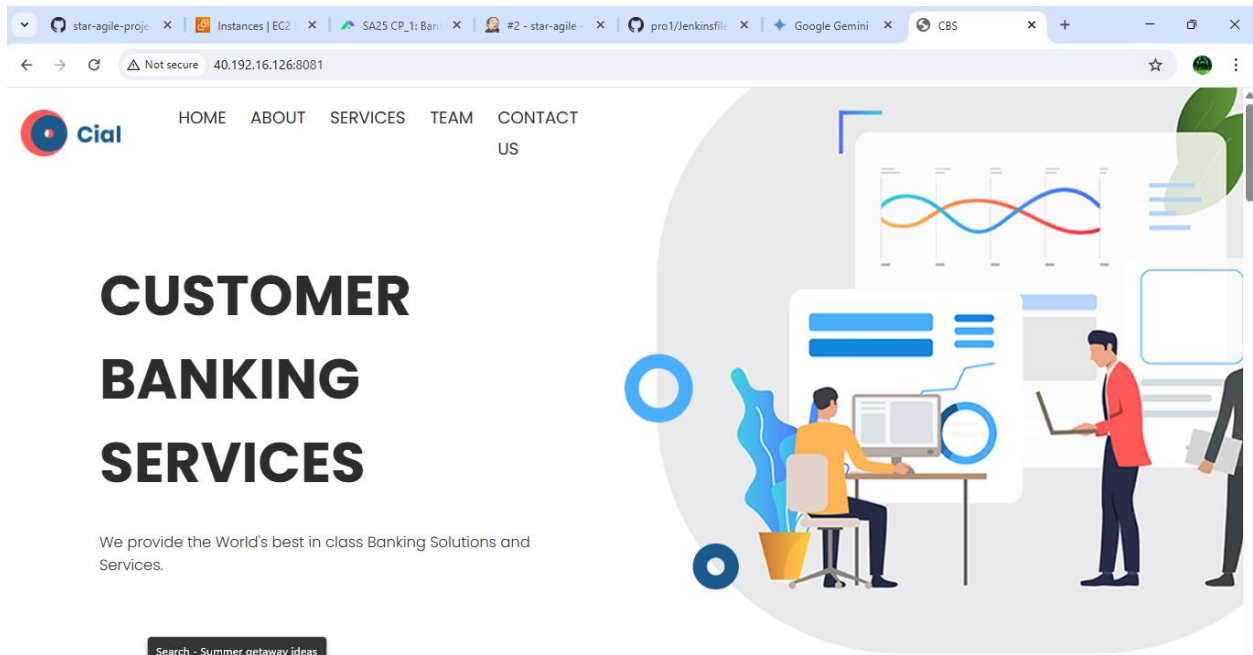
| Plugin Name | Status |
|--------------------------------------|--|
| Preparation | • Checking internet connectivity • Checking update center connectivity • Success |
| Authentication Tokens API | Success |
| Docker Commons | Success |
| Docker Pipeline | Success |
| Javadoc | Pending |
| Dev Tools Symbols API | Pending |
| jsoup API | Pending |
| JSch dependency | Pending |
| Maven Integration | Pending |
| Apache HttpComponents Client 5.x API | Pending |
| Commons Compress API | Pending |
| Docker API | Pending |
| docker-build-step | Pending |
| Maven Integration | Pending |
| Pipeline: REST API | Pending |
| Pipeline: Stage View | Pending |
| Loading plugin extensions | Pending |

Step25:- Now again go back to Jenkins project and click on Build now to check the status of the build and as you can see that the build is successful and a docker container is also created in the ec2



```
root@ip-10-0-1-230:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
96425cecf5c0   pj013525/finance-pro-image:v1      "java -jar /app.jar"    38 seconds ago Up 38 seconds  0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
finance-pro-cont
root@ip-10-0-1-230:/home/ubuntu#
```

Step26:- Now go to any browser and give the IPaddress:8081 and click enter the you will see the home page of the project and thus the project deployment is successful.



Step27:- Now monitor the docker container using Prometheus and Grafana , for that install Prometheus in Jenkins-Docker server and Grafana in another server

```
root@ip-10-0-1-205:/home/ubuntu# wget https://dl.grafana.com/enterprise/release/grafana-enterprise-8.4.4.linux-amd64.tar.gz
ls
--2025-05-17 12:28:16-- https://dl.grafana.com/enterprise/release/grafana-enterprise-8.4.4.linux-amd64.tar.gz
Resolving dl.grafana.com (dl.grafana.com)... 151.101.38.217, 2a04:4e42:9::729
Connecting to dl.grafana.com (dl.grafana.com)|151.101.38.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84007981 (80M) [application/x-tar]
Saving to: 'grafana-enterprise-8.4.4.linux-amd64.tar.gz'

grafana-enterprise-8.4.4.linux-amd 100%[=====] 80.12M 3.07MB/s in 10s

2025-05-17 12:28:30 (7.85 MB/s) - 'grafana-enterprise-8.4.4.linux-amd64.tar.gz' saved [84007981/84007981]

grafana-enterprise-8.4.4.linux-amd64.tar.gz
root@ip-10-0-1-205:/home/ubuntu#
```

WhatsApp

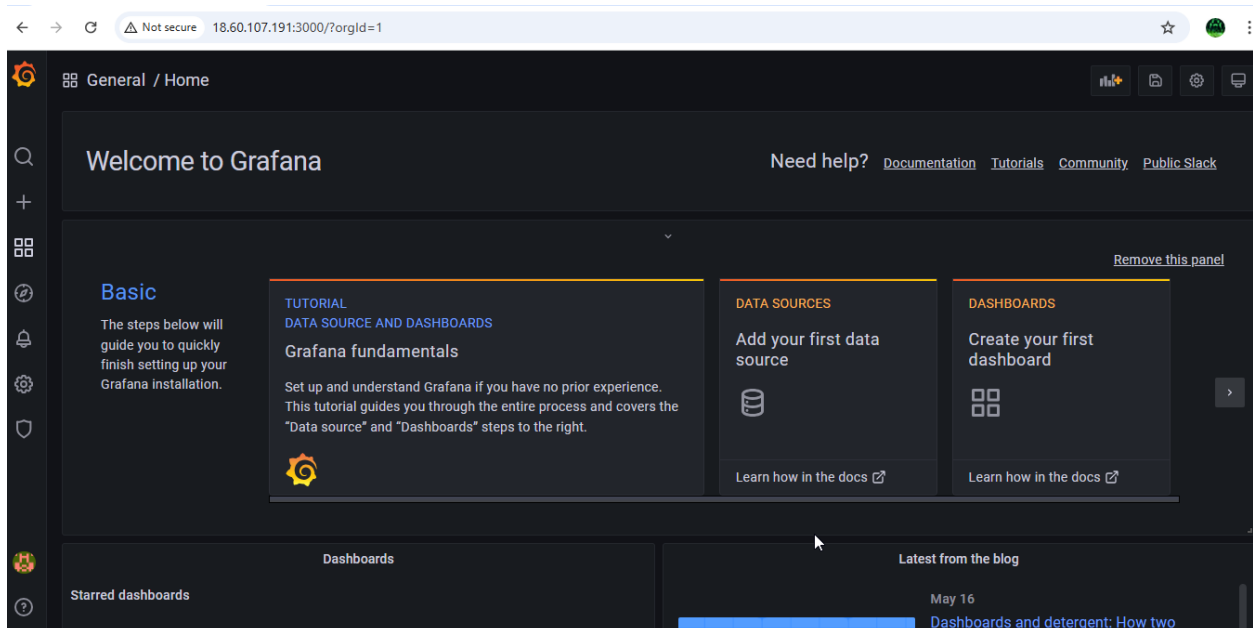
baXterm by s/subscribe to the professional edition here: <https://mobaxterm.mobatek.net>


```
ntent-disposition=attachment%3B%20filename%3Dprometheus-2.34.0.linux-amd64.tar.gz&response-content-type=application%
wing]
--2025-05-17 12:30:52-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/408c8e
a51d5db85?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250517%2Fus-east-1%2Fs3%2Faws
20250517T123052Z&X-Amz-Expires=300&X-Amz-Signature=234a94e8a223abb35f896d6f6ba36edaff9e13c7b447a16285fd0d3871c427ff8
ost&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.34.0.linux-amd64.tar.gz&response-content-ty
t-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 76299772 (73M) [application/octet-stream]
Saving to: 'prometheus-2.34.0.linux-amd64.tar.gz'

prometheus-2.34.0.linux-amd64.tar. 100%[=====] 72.76M 9.5
2025-05-17 12:31:00 (11.0 MB/s) - 'prometheus-2.34.0.linux-amd64.tar.gz' saved [76299772/76299772]

root@ip-10-0-1-230:/home/ubuntu# ls
jenkins.sh jenkins.sh.1 prometheus-2.34.0.linux-amd64.tar.gz
root@ip-10-0-1-230:/home/ubuntu# tar xzvf prometheus-2.34.0.linux-amd64.tar.gz
prometheus-2.34.0.linux-amd64/
prometheus-2.34.0.linux-amd64/conssoles/
prometheus-2.34.0.linux-amd64/conssoles/index.html.example
prometheus-2.34.0.linux-amd64/conssoles/node-cpu.html
prometheus-2.34.0.linux-amd64/conssoles/node-disk.html
prometheus-2.34.0.linux-amd64/conssoles/node-overview.html
prometheus-2.34.0.linux-amd64/conssoles/node.html
prometheus-2.34.0.linux-amd64/conssoles/prometheus-overview.html
prometheus-2.34.0.linux-amd64/conssoles/prometheus.html
prometheus-2.34.0.linux-amd64/console_libraries/
prometheus-2.34.0.linux-amd64/console_libraries/menu.lib
prometheus-2.34.0.linux-amd64/console_libraries/prom.lib
prometheus-2.34.0.linux-amd64/prometheus.yml
prometheus-2.34.0.linux-amd64/LICENSE
prometheus-2.34.0.linux-amd64/NOTICE
prometheus-2.34.0.linux-amd64/prometheus
prometheus-2.34.0.linux-amd64/promtool
root@ip-10-0-1-230:/home/ubuntu#
```

Step28:- After successful installation of Grafana now go to browser and give grafana server ip-address:3000 (3000 is default port number for grafana) and use admin and admin as username and password as they are default and login to the grafana home page



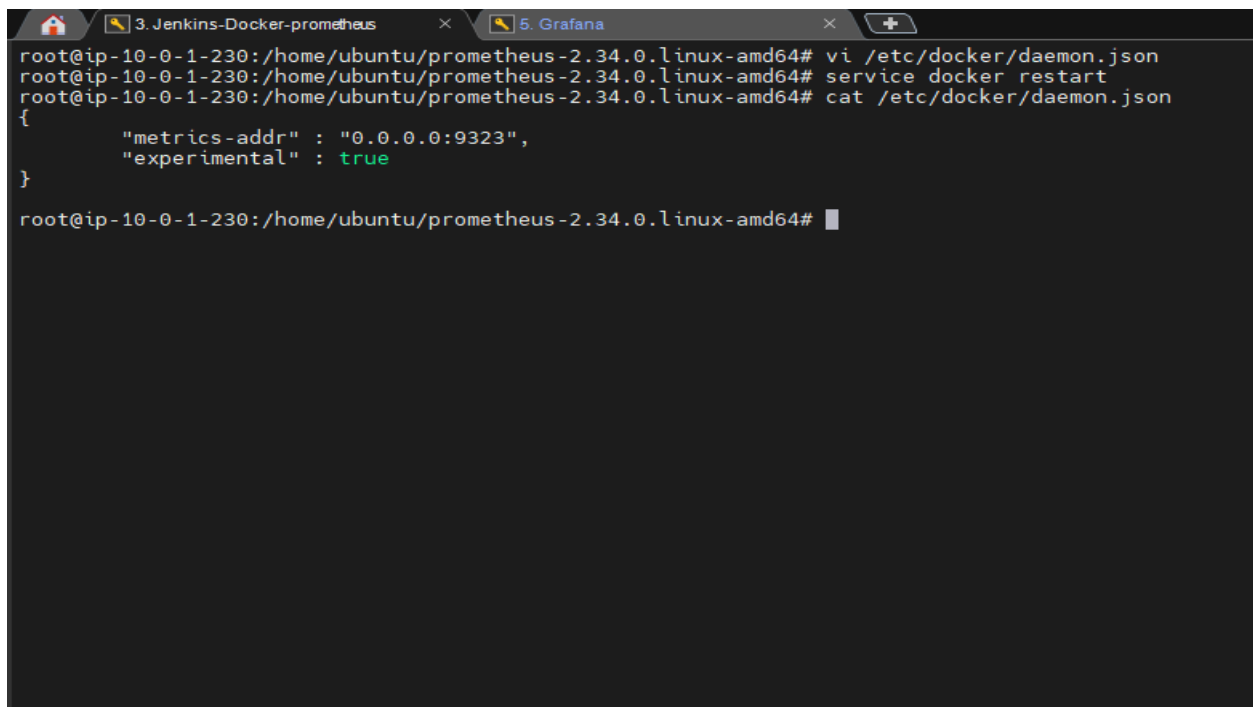
Step29:- Now setup the docker and Prometheus in another using by telling docker that Prometheus would track docker on port 9323

i.e., vi /etc/docker/daemon.json

press I to insert

```
{  
    "metrics-addr" : "0.0.0.0:9323",  
    "experimental" : true
```

} then save and exit and restart the docker

A terminal window with two tabs: '3. Jenkins-Docker-prometheus' and '5. Grafana'. The terminal shows the following commands and output:

```
root@ip-10-0-1-230:/home/ubuntu/prometheus-2.34.0.linux-amd64# vi /etc/docker/daemon.json  
root@ip-10-0-1-230:/home/ubuntu/prometheus-2.34.0.linux-amd64# service docker restart  
root@ip-10-0-1-230:/home/ubuntu/prometheus-2.34.0.linux-amd64# cat /etc/docker/daemon.json  
{  
    "metrics-addr" : "0.0.0.0:9323",  
    "experimental" : true  
}
```

Step30:- Now go to any browser and give

[docker ip-address:9323/metrics](http://docker-ip-address:9323/metrics)

and in the below image you will see that the docker stats have been started successfully

← → ↺ ⚠ Not secure 40.192.16.126:9323/metrics

```
# HELP builder_builds_failed_total Number of failed image builds
# TYPE builder_builds_failed_total counter
builder_builds_failed_total{reason="build_canceled"} 0
builder_builds_failed_total{reason="build_target_not_reachable_error"} 0
builder_builds_failed_total{reason="command_not_supported_error"} 0
builder_builds_failed_total{reason="dockerfile_empty_error"} 0
builder_builds_failed_total{reason="dockerfile_syntax_error"} 0
builder_builds_failed_total{reason="error_processing_commands_error"} 0
builder_builds_failed_total{reason="missing_onbuild_arguments_error"} 0
builder_builds_failed_total{reason="unknown_instruction_error"} 0
# HELP builder_builds_triggered_total Number of triggered image builds
# TYPE builder_builds_triggered_total counter
builder_builds_triggered_total 0
# HELP engine_daemon_container_actions_seconds The number of seconds it takes to process each container action
# TYPE engine_daemon_container_actions_seconds histogram
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="changes",le="+Inf"} 1
engine_daemon_container_actions_seconds_sum{action="changes"} 0
engine_daemon_container_actions_seconds_count{action="changes"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.005"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.01"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.025"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.05"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.25"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="0.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="1"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="2.5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="5"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="10"} 1
engine_daemon_container_actions_seconds_bucket{action="commit",le="+Inf"} 1
```

Step31:- Now add docker job in the Prometheus.yml file to give this stats to Prometheus

vi prometheus.yml

- job_name: "docker"

metrics_path defaults to '/metrics'

scheme defaults to 'http'.

static_configs:

- targets: ["localhost:9323"]

Save the file and exit and start the Prometheus using ./prometheus

```
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        # - alertmanager:9093

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

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

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

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "docker"

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

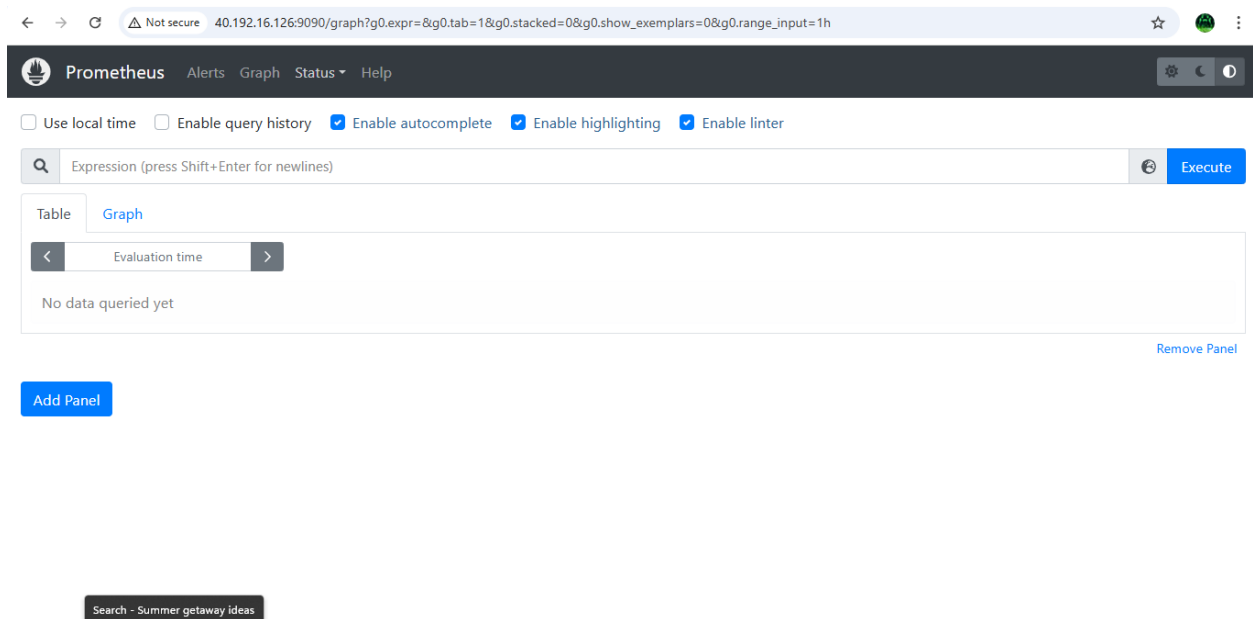
    static_configs:
      - targets: ["localhost:9323"]

root@ip-10-0-1-230:/home/ubuntu/prometheus-2.34.0.linux-amd64#
```

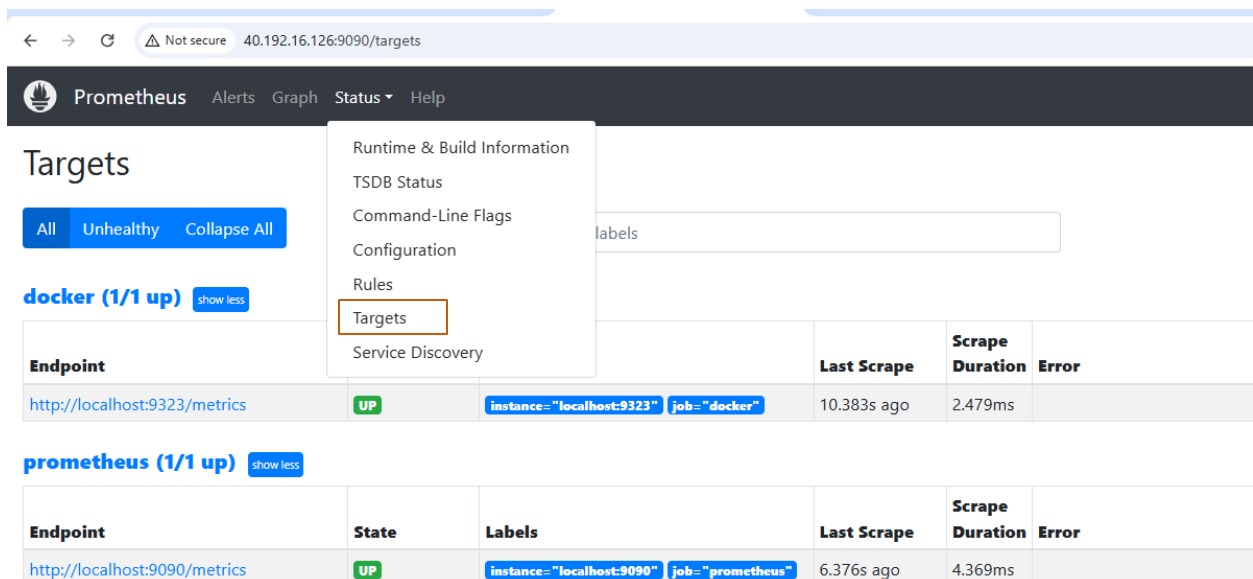
```
root@ip-10-0-1-230:/home/ubuntu/prometheus-2.34.0.linux-amd64# ./prometheus
ts=2025-05-17T12:59:34.854Z caller=main.go:479 level=info msg="No time or size retention was set so using the default time retention" duration=15s
ts=2025-05-17T12:59:34.854Z caller=main.go:516 level=info msg="Starting Prometheus" version="(version=2.34.0, branch=HEAD, revision=881111fec4332c33094a6fb2680c71fffc427275)"
ts=2025-05-17T12:59:34.854Z caller=main.go:521 level=info build_context="(go=go1.17.8, user=root@121ad7ea5487, date=20220315-15:18:00)"
ts=2025-05-17T12:59:34.854Z caller=main.go:522 level=info host_details="(Linux 6.8.0-1024-aws #26-Ubuntu SMP Tue Feb 18 17:22:37 UTC 2025 x86_64 ip-10-0-1-230 (none))"
ts=2025-05-17T12:59:34.854Z caller=main.go:523 level=info fd_limits="(soft=1024, hard=1048576)"
ts=2025-05-17T12:59:34.854Z caller=main.go:524 level=info vm_limits="(soft=unlimited, hard=unlimited)"
ts=2025-05-17T12:59:34.857Z caller=web.go:540 level=info component=web msg="Start listening for connections" address=0.0.0.0:9090
ts=2025-05-17T12:59:34.857Z caller=main.go:937 level=info msg="Starting TSDB ..."
ts=2025-05-17T12:59:34.860Z caller=ts_config.go:195 level=info component=web msg="TLS is disabled." http2=false
ts=2025-05-17T12:59:34.862Z caller=head.go:493 level=info component=tsdb msg="Replaying on-disk memory mappable chunks if any"
ts=2025-05-17T12:59:34.862Z caller=head.go:536 level=info component=tsdb msg="On-disk memory mappable chunks replay completed" duration=2.299µs
ts=2025-05-17T12:59:34.862Z caller=head.go:542 level=info component=tsdb msg="Replaying WAL, this may take a while"
ts=2025-05-17T12:59:34.862Z caller=head.go:613 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=0
ts=2025-05-17T12:59:34.862Z caller=head.go:619 level=info component=tsdb msg="WAL replay completed" checkpoint_replay_duration=39.105µs wal_replay_duration=255.613µs total_replay_duration=324.055µs
ts=2025-05-17T12:59:34.863Z caller=main.go:958 level=info fs_type=EXT4_SUPER_MAGIC
ts=2025-05-17T12:59:34.863Z caller=main.go:961 level=info msg="TSDB started"
ts=2025-05-17T12:59:34.863Z caller=main.go:1142 level=info msg="Loading configuration file" filename=prometheus.yml
ts=2025-05-17T12:59:34.867Z caller=main.go:1179 level=info msg="Completed loading of configuration file" filename=prometheus.yml totalDuration=3.835138ms db_storage=823ns remote_storage=1.783µs web_handler=400ns query_engine=916ns scrape=3.378297ms scrape_sd=70.415µs notify=33.463µs notify_sd=9.979µs rules=1.46µs tracing=5.136µs
ts=2025-05-17T12:59:34.868Z caller=main.go:910 level=info msg="Server is ready to receive web requests."
```

As you can see that the Prometheus have been started from the above image

Step32:- Now go browser and give docker ip:9090 and enter , then you will be successfully enter into the Prometheus homepage



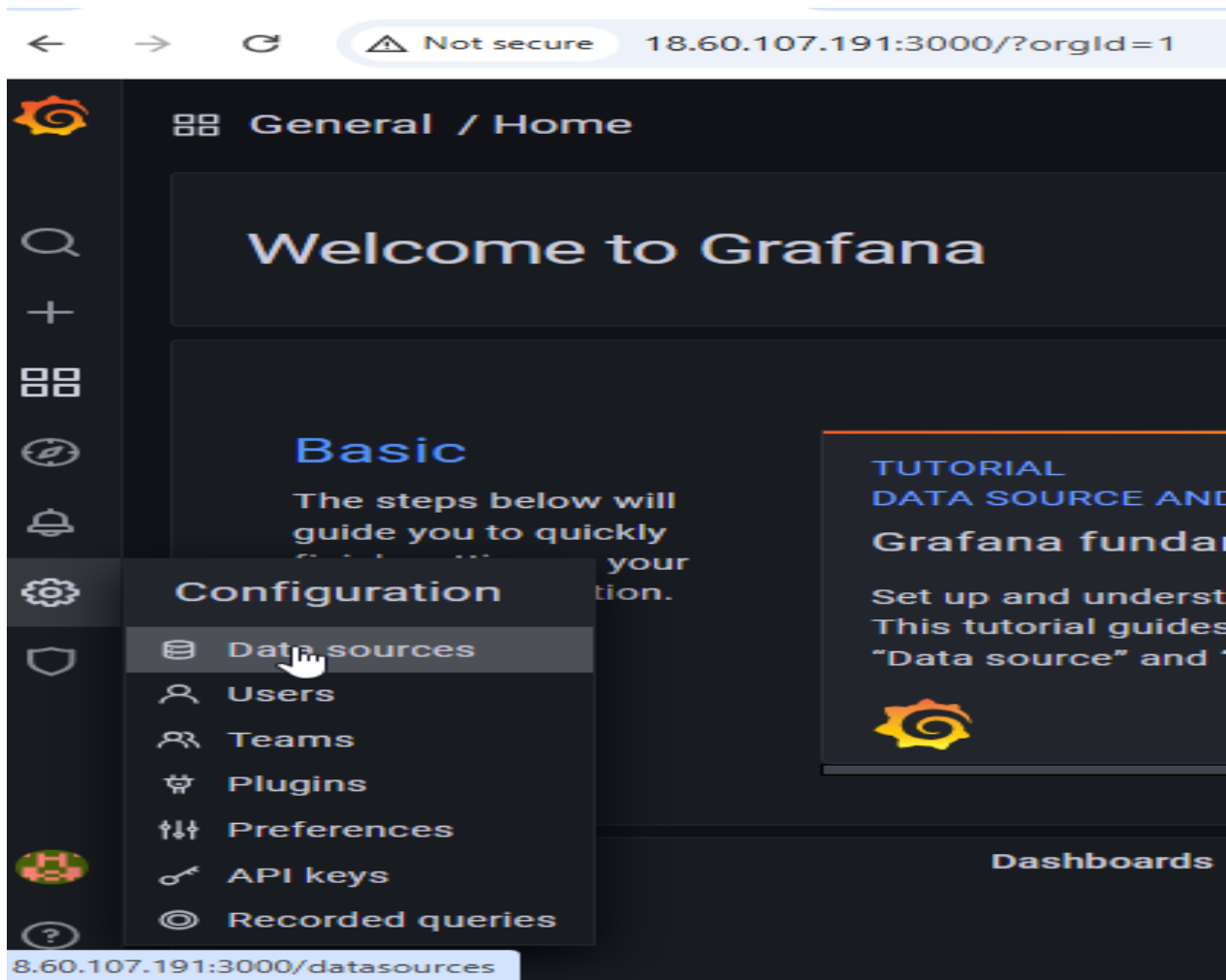
Step33:- Now click on status → targets then you will see the status of the docker and Prometheus



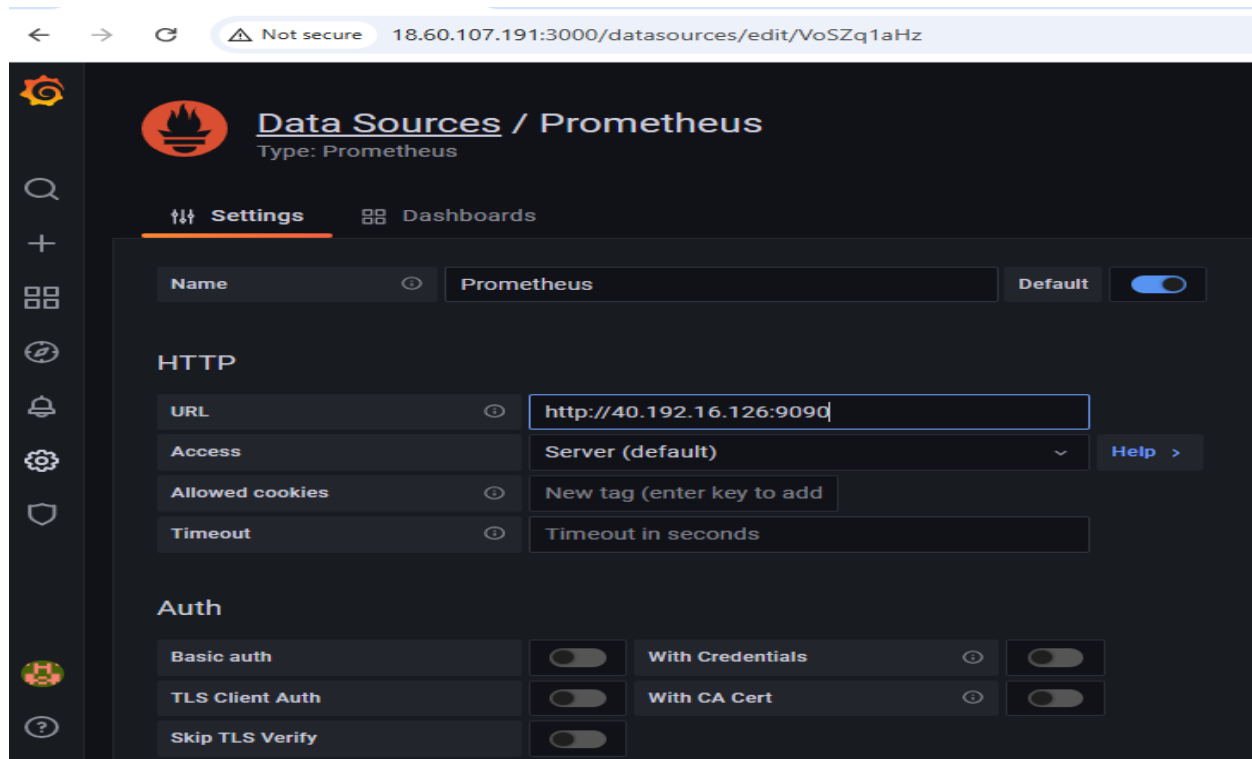
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
|-------------------------------|-------|--|-------------|-----------------|-------|
| http://localhost:9323/metrics | UP | instance="localhost:9323" job="docker" | 10.383s ago | 2.479ms | |

| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
|-------------------------------|-------|--|-------------|-----------------|-------|
| http://localhost:9090/metrics | UP | instance="localhost:9090" job="prometheus" | 6.376s ago | 4.369ms | |

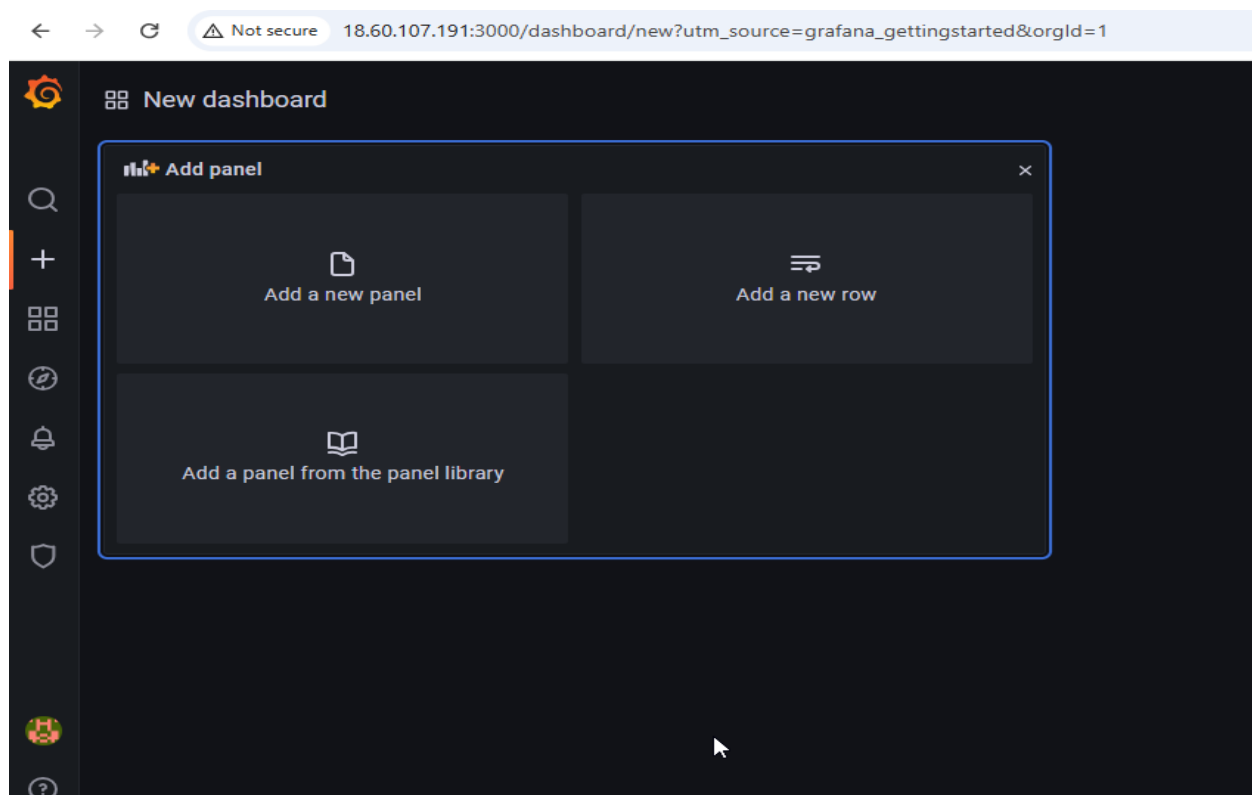
Step34:- Now go to grafana homepage → configurations → Data sources



Step35:- Now click on add Data sources → Prometheus and give ipaddress:9090 and click on save and test

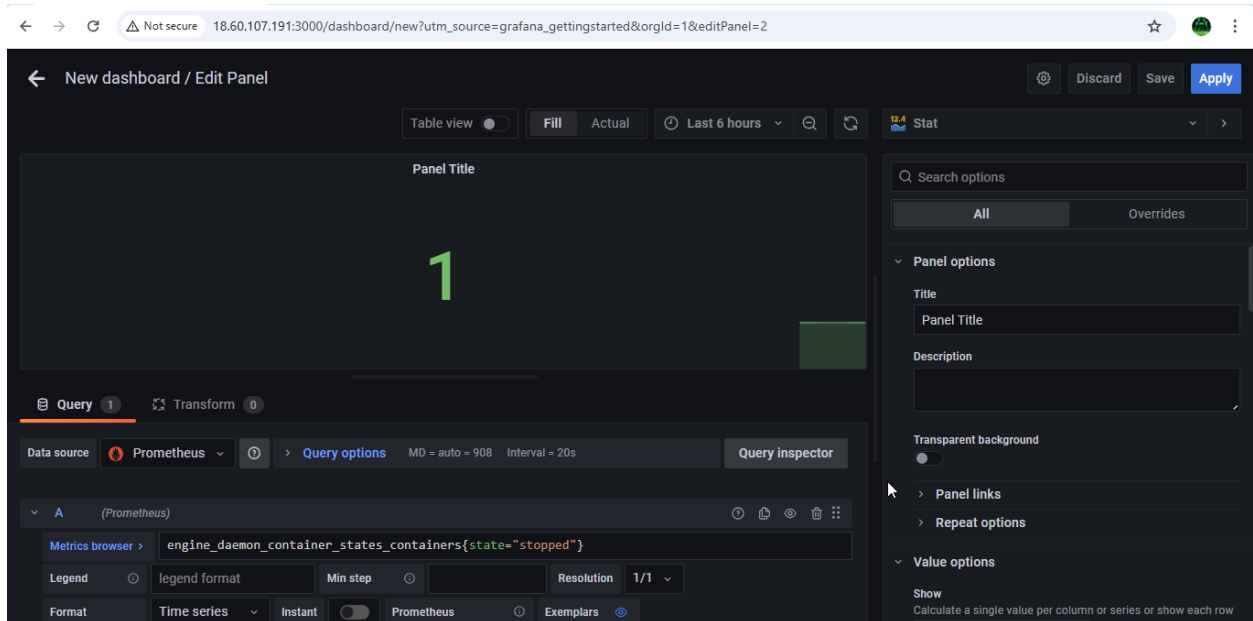


Step36:- Now click on Dash board → add new panel

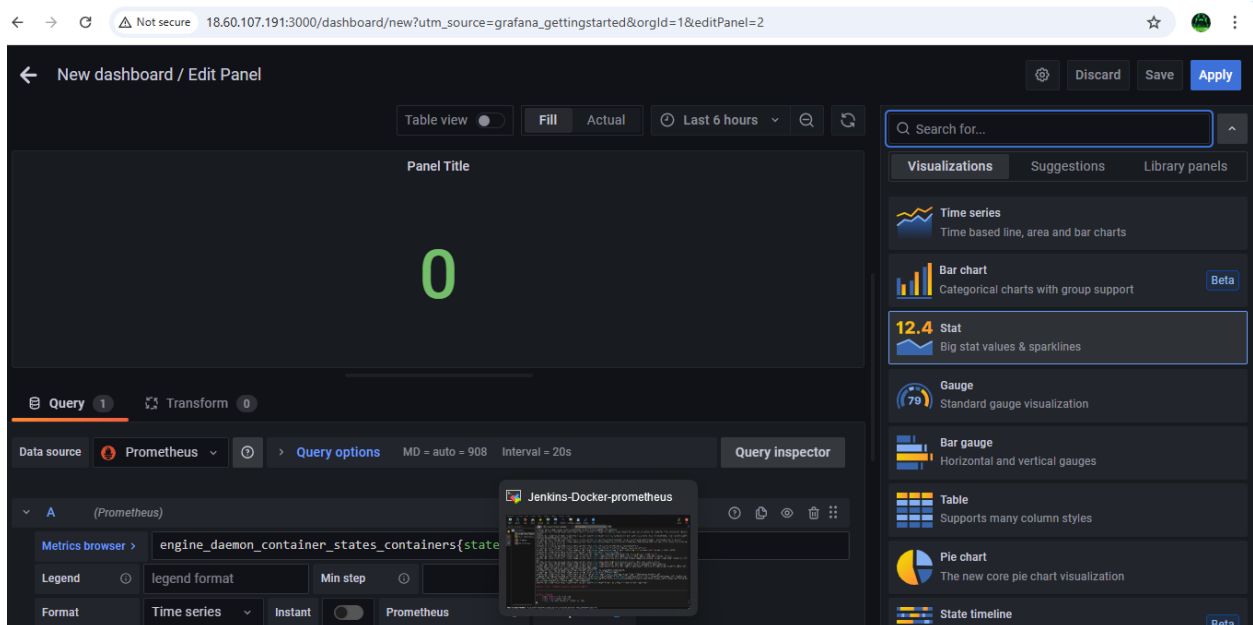


Step37:- Now in the metrics browser give

engine daemon container states containers{state="stopped"} and you will see the result that same as in the metrics from the browser



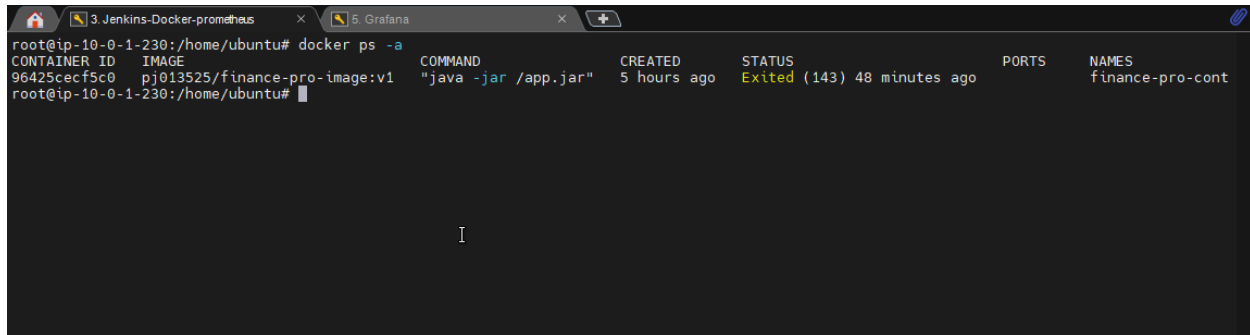
engine daemon container states containers{state="running"}



Step38:- The values shown in the panel must be equal to the that of shown in the docker stats

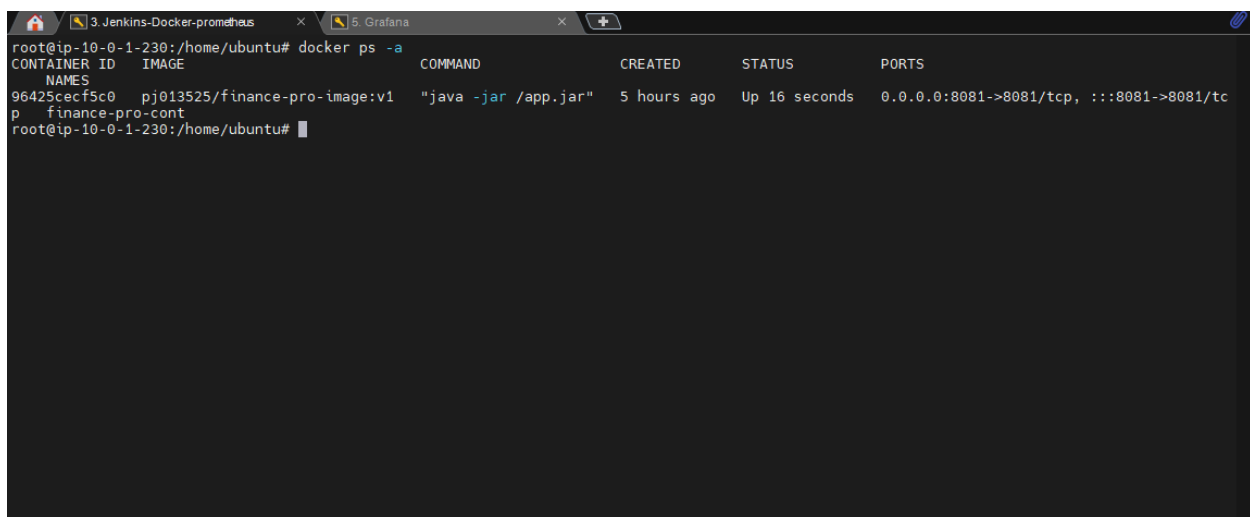

```
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 0
engine_daemon_container_states_containers{state="stopped"} 1
# HELP engine_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
```

Here the container which we created is in exited state so it is showing as stopped state in stats



```
root@ip-10-0-1-230:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS      PORTS          NAMES
96425cecf5c0   pj013525/finance-pro-image:v1     "java -jar /app.jar"    5 hours ago   Exited (143) 48 minutes ago           finance-pro-cont
```

Step39:- Now start the container again and check the details again in the stats

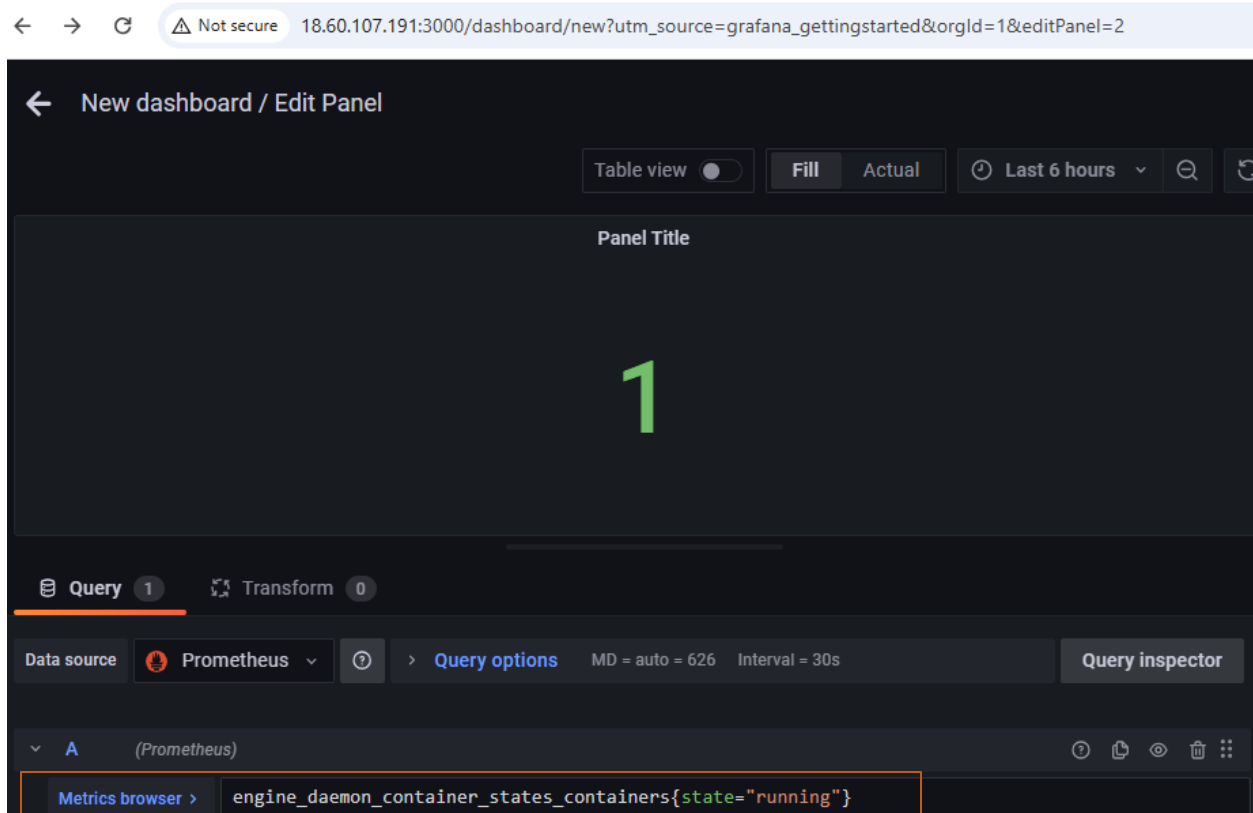


```
root@ip-10-0-1-230:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS      PORTS          NAMES
96425cecf5c0   pj013525/finance-pro-image:v1     "java -jar /app.jar"    5 hours ago   Up 16 seconds   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   finance-pro-cont
```

Now check tin the docker stats

```
# TYPE engine_daemon_container_states_containers gauge
engine_daemon_container_states_containers{state="paused"} 0
engine_daemon_container_states_containers{state="running"} 1
engine_daemon_container_states_containers{state="stopped"} 0
# HELP engine_daemon_engine_cpus_cpus The number of cpus that the host system of the engine has
# TYPF engine_daemon_engine_cpus_cpus gauge
```

As you see that the state=running is 1 then give this in the metrics browser in the Grafana dash board panel and see the value



As you can the value is changed from 0 to 1 in the running state as we started container from exited to running state

Step40:- This is how we monitor the health of a container automatically and visualising the report using Prometheus and Grafana.