

Setting Up Prometheus & Grafana

1. First Clone the git repo mentioned below to get all of the Docker-compose.yml and Prometheus.yml and other config files.

git clone <https://github.com/ratr45/Docker-Monitoring-Service.git>

```
ubuntu@ip-172-31-32-116:~$ sudo su
root@ip-172-31-32-116:/home/ubuntu# git clone https://github.com/ratr45/Docker-Monitoring-Service.git
Cloning into 'Docker-Monitoring-Service'...
remote: Enumerating objects: 820, done.
remote: Counting objects: 100% (820/820), done.
remote: Compressing objects: 100% (334/334), done.
remote: Total 820 (delta 475), reused 813 (delta 468), pack-reused 0
Receiving objects: 100% (820/820), 5.79 MiB | 27.58 MiB/s, done.
Resolving deltas: 100% (475/475), done.
root@ip-172-31-32-116:/home/ubuntu#
```

i-01f2abdbdb0d592e12 (Manager)

PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

ls

(so this directory has all the components that are required to run our docker monitoring)

```
root@ip-172-31-32-116:/home/ubuntu# ls
Docker-Monitoring-Service
root@ip-172-31-32-116:/home/ubuntu#
```

i-01f2abdbdb0d592e12 (Manager)

PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

cd Docker-Monitoring-Service

```
root@ip-172-31-32-116:/home/ubuntu# cd Docker-Monitoring-Service/
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service#
```

i-01f2abdbdb0d592e12 (Manager)

PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

ls

```
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service# ls
LICENSE README.md alertmanager caddy docker-compose.traefik.yml docker-compose.yml grafana node-exporter prometheus test-compose.yml weave-compose.yml
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service#
```

i-01f2abdb0d592e12 (Manager)
PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

2. Initialize a docker swarm which you can use for deploying the various tools

docker swarm init

Use the join token commands given when you initialize a new swarm and use it on the nodes you want to work with.

docker swarm join --token

come back to manager instance (we have two nodes)

docker node ls

```
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service# docker node ls
ID                                HOSTNAME                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
2sgx9mu165qzx1zrsnzm9z8m9 *    ip-172-31-32-116       Ready     Active           Leader             24.0.5
50ickfymzwr961fjcgulthiu       ip-172-31-38-39        Ready     Active           -                  24.0.5
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service#
```

i-01f2abdb0d592e12 (Manager)
PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

3. Now Deploy the monitoring services using docker stack deploy using the command below

deploy stack deploy -c docker-compose.yml swarmprom

```
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service# docker stack deploy -c docker-compose.yml swarmprom
Creating network swarmprom_net
Creating config swarmprom_caddy_config
Creating config swarmprom_node_rules
Creating config swarmprom_task_rules
Creating service swarmprom_caddy
Creating service swarmprom_cadvisor
Creating service swarmprom_grafana
Creating service swarmprom_node-exporter
Creating service swarmprom_prometheus
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service#
```

i-01f2abdb0d592e12 (Manager)
PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

To check what services are running

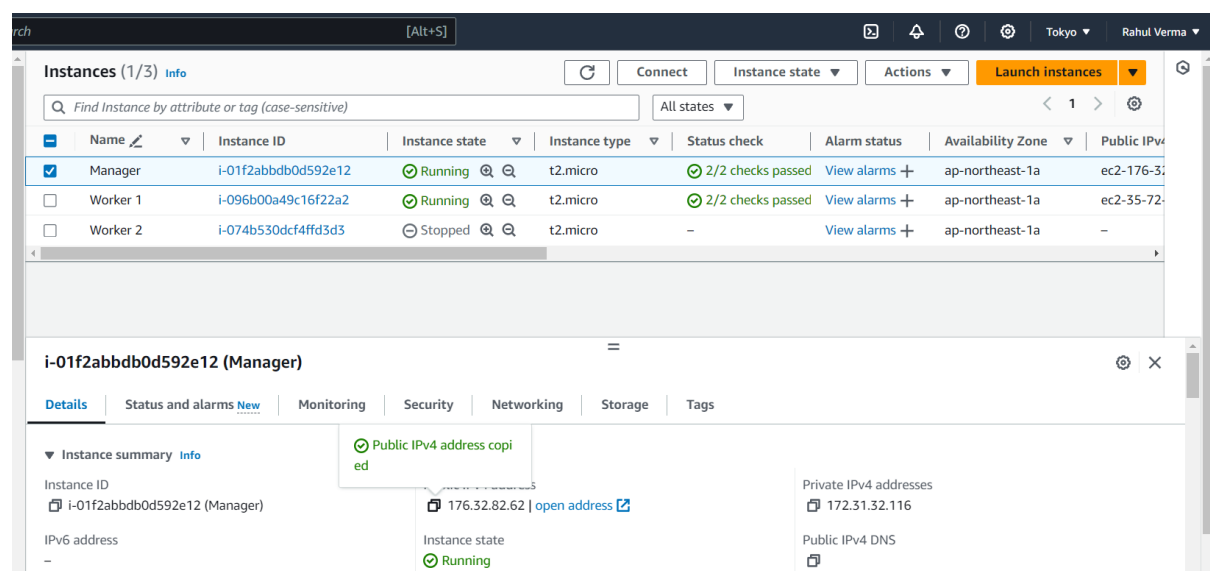
docker service ls

```
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service# docker service ls
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
qkgac6zpnbbhb    service-A            replicated           2/2                  nginx:latest
c7g7194afo56     service-B            replicated           2/2                  nginx:latest
bk73x8aambff     service-c            replicated           1/1                  nginx:latest
pihwnzz3np3      swarmprom_caddy      replicated           1/1                  stefanprodan/caddy:latest      *:3000->3000/tcp, *:9090->9090/tcp, *:9094->9094/tcp
cp               swarmprom_cadvisor    global              0/0                  google/cadvisor:latest
ml75pt52gavd     swarmprom_grafana     replicated           1/1                  stefanprodan/swarmprom-grafana:5.3.4
v58q7iydv29h     swarmprom_node-exporter global              2/2                  stefanprodan/swarmprom-node-exporter:v0.16.0
sro228vce4ch     swarmprom_prometheus  replicated           1/1                  stefanprodan/swarmprom-prometheus:v2.5.0
root@ip-172-31-32-116:/home/ubuntu/Docker-Monitoring-Service#
```

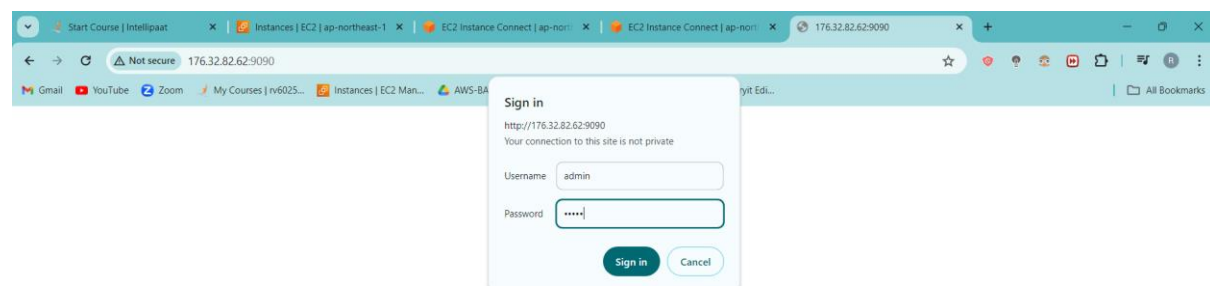
i-01f2abdb0d592e12 (Manager)
PublicIPs: 176.32.82.62 PrivateIPs: 172.31.32.116

4. Head to the browser and type in the public IP address of the manager node with the port at which Prometheus is mapped at. This should open up Prometheus for you.

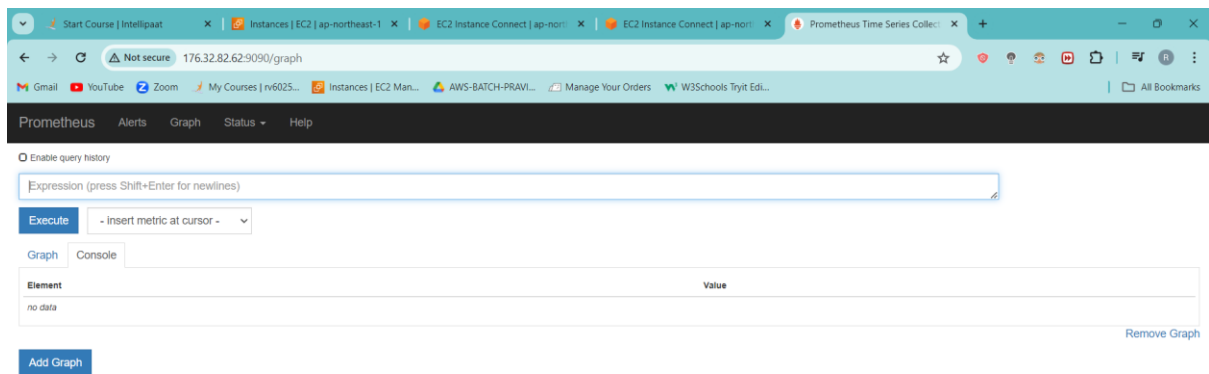
Copy manager ip address



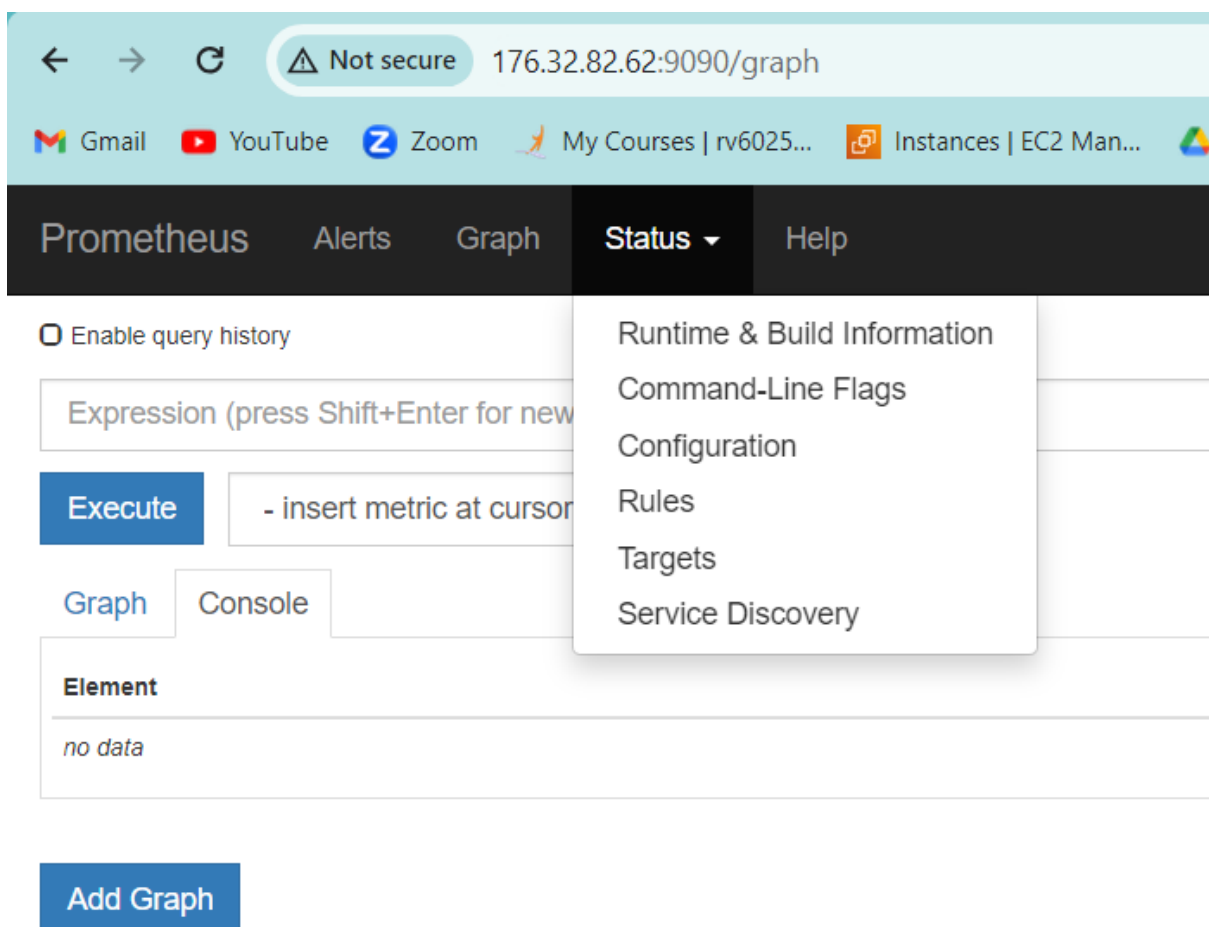
<Manager ip>:9090



Prometheus-



5. Now open up targets using the drop down from status.



Here we will verify if we are receiving data from the different parts of our services and sources.

As you can see the state is 'UP' which means Prometheus is able to receive data from it.

← → ↻ ⚠ Not secure 176.32.82.62:9090/targets

Gmail

YouTube

Zoom

My Courses | rv6025...

Instances | EC2 Man...

AWS-BATCH-PRAVI...

Manage Your Orders

W3Schools Tryit Edi...

PrometheusAlertsGraphStatus ▾Help

Targets

AllUnhealthy

cadvisor (0/0 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration
----------	-------	--------	-------------	-----------------

dockerd-exporter (0/0 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration
----------	-------	--------	-------------	-----------------

node-exporter (2/2 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.0.2.15:9100/metrics	UP	instance="10.0.2.15:9100"job="node-exporter"	9.948s ago	19.02ms	
http://10.0.2.16:9100/metrics	UP	instance="10.0.2.16:9100"job="node-exporter"	11.469s ago	20.09ms	

prometheus (1/1 up)show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090"job="prometheus"	12.676s ago	6.539ms	

41°C Haze

6. Now open up Grafana by accessing port 3000 of the same manage node.

<manager_ip>:3000

Start Chrome (IntelliJ)Instances (EC2 (ap-north-1) x 1EC2 Instance Connect (a...EC2 Instance Connect (a...Prometheus Time SeriesGrafana

← → ⚠ Not secure 176.32.82.62:3000/login

Gmail

YouTube

Zoom

My Courses | rv6025...

Instances | EC2 Man...

AWS-BATCH-PRAVI...

Manage Your Orders

W3Schools Tryit Edi...

email or username

password

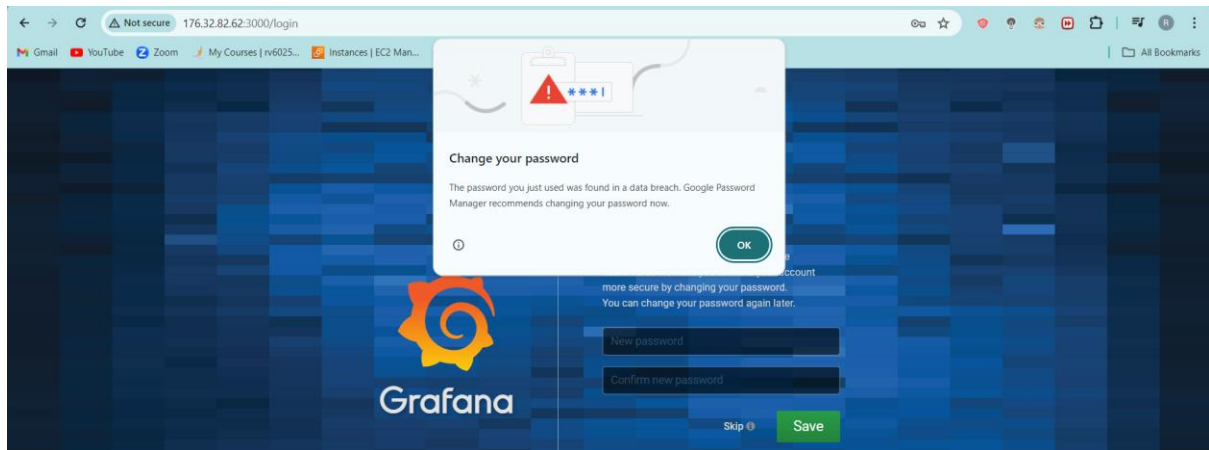
Log In

Forgot your password?

© Grafana | Support Plans | Get Community | Grafana v6.3.4 (commit: 4300300) | New version available!

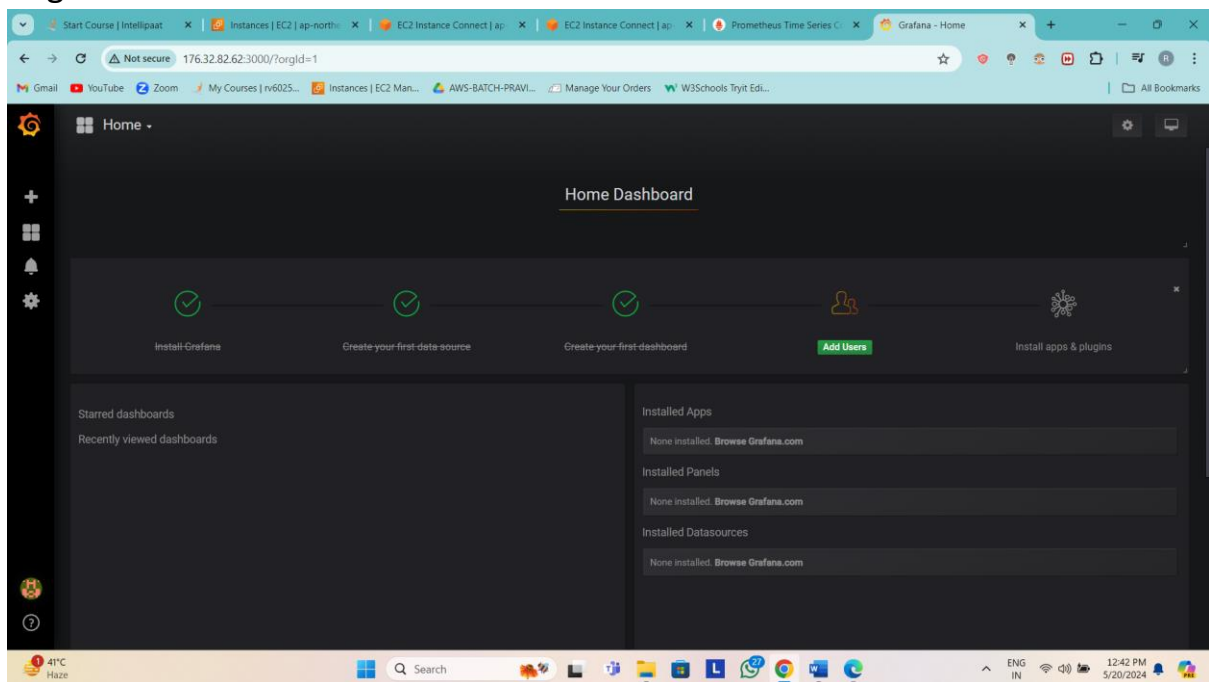
41°C Haze

Choose a new and strong password for yourself

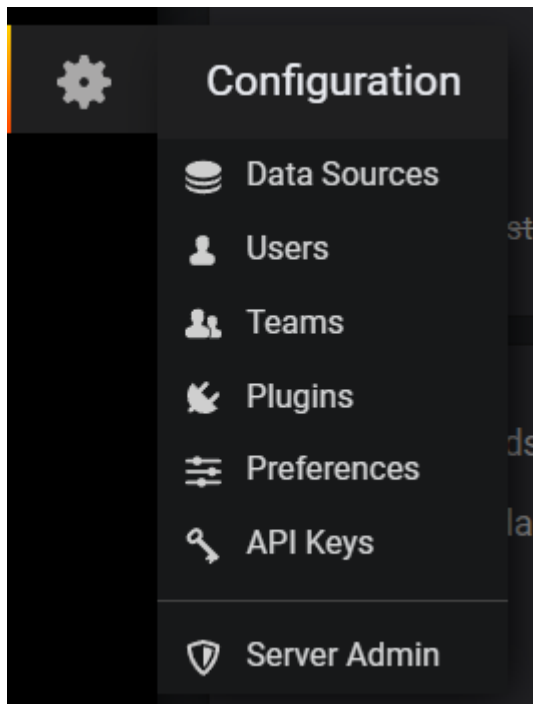


Password: RahulGrafana@123#

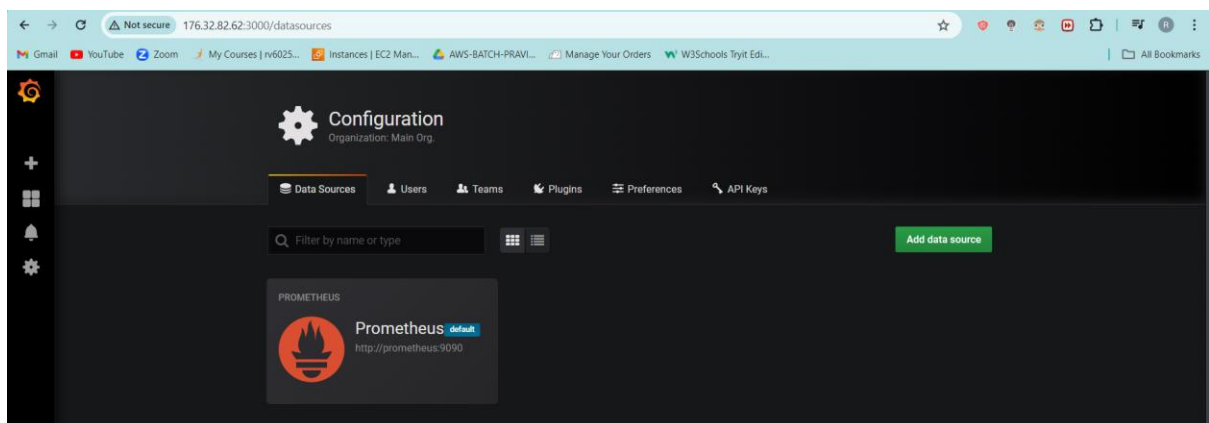
This the dashboard you will see unless they have updated UI for newer version of grafana



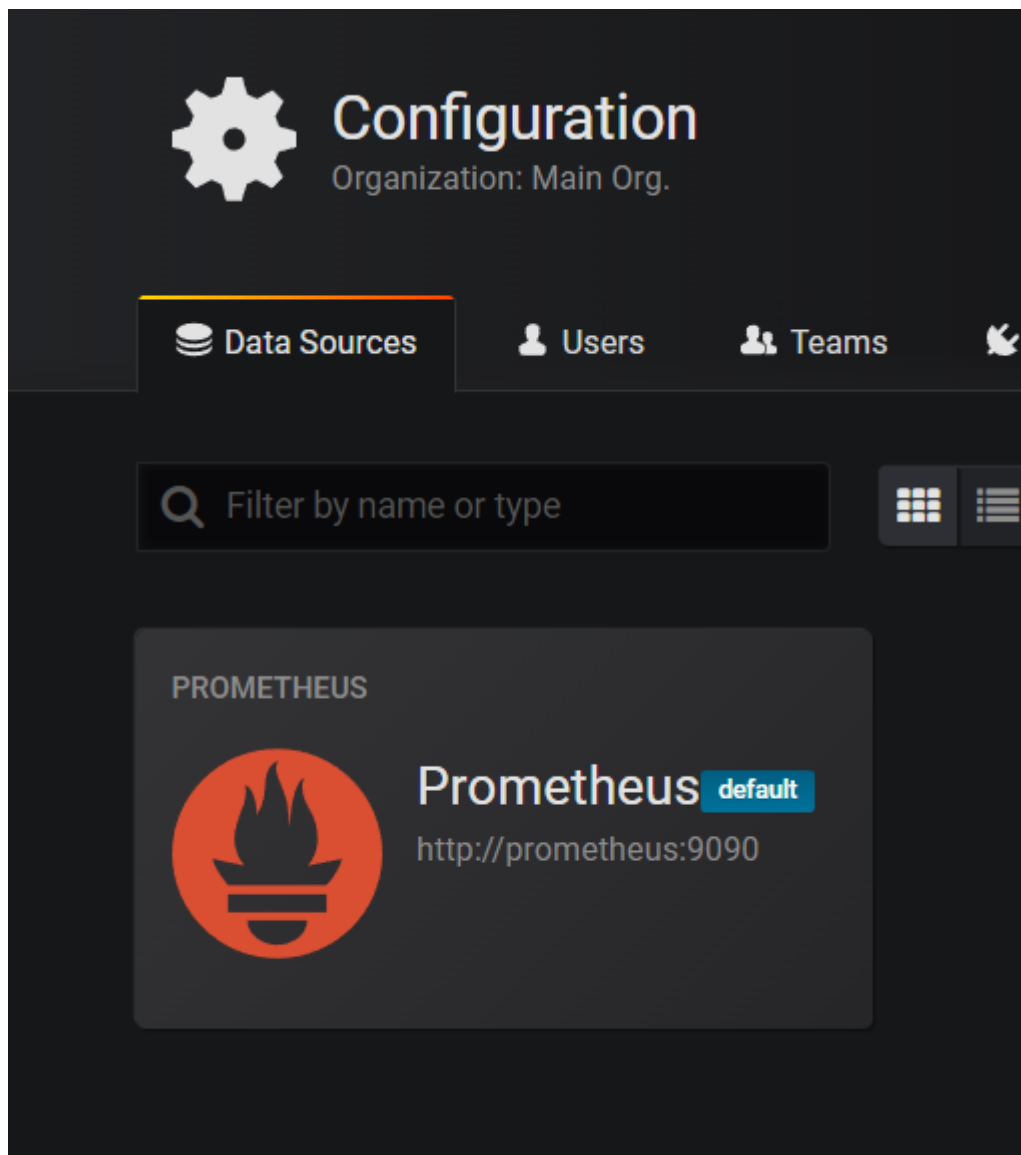
7. Head to configuration ->data sources from the side menu.



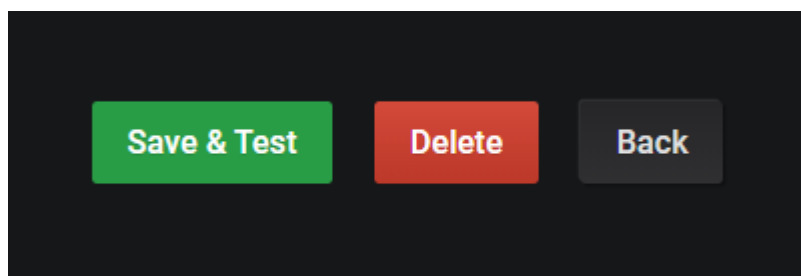
Here you can add any data source that Grafana supports, but Prometheus will already be added as we had configured it earlier in our config file for grafana.



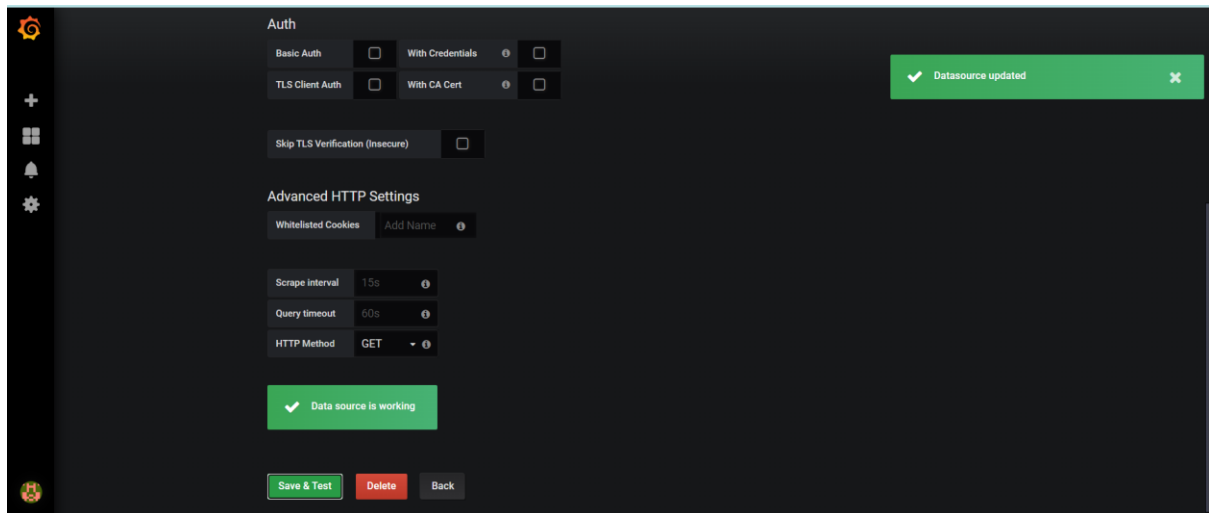
Click on Prometheus (default)



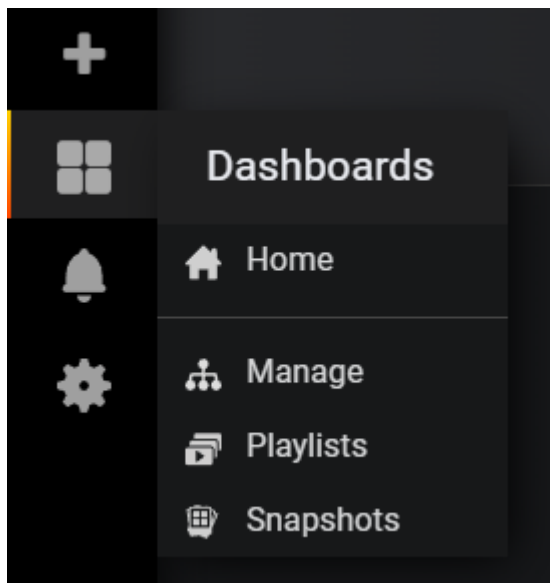
Click on save and test



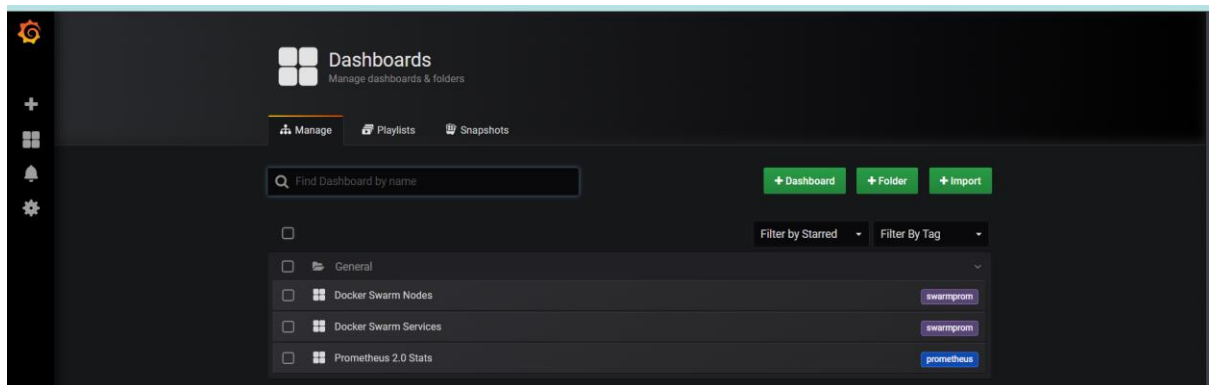
So our data source is working



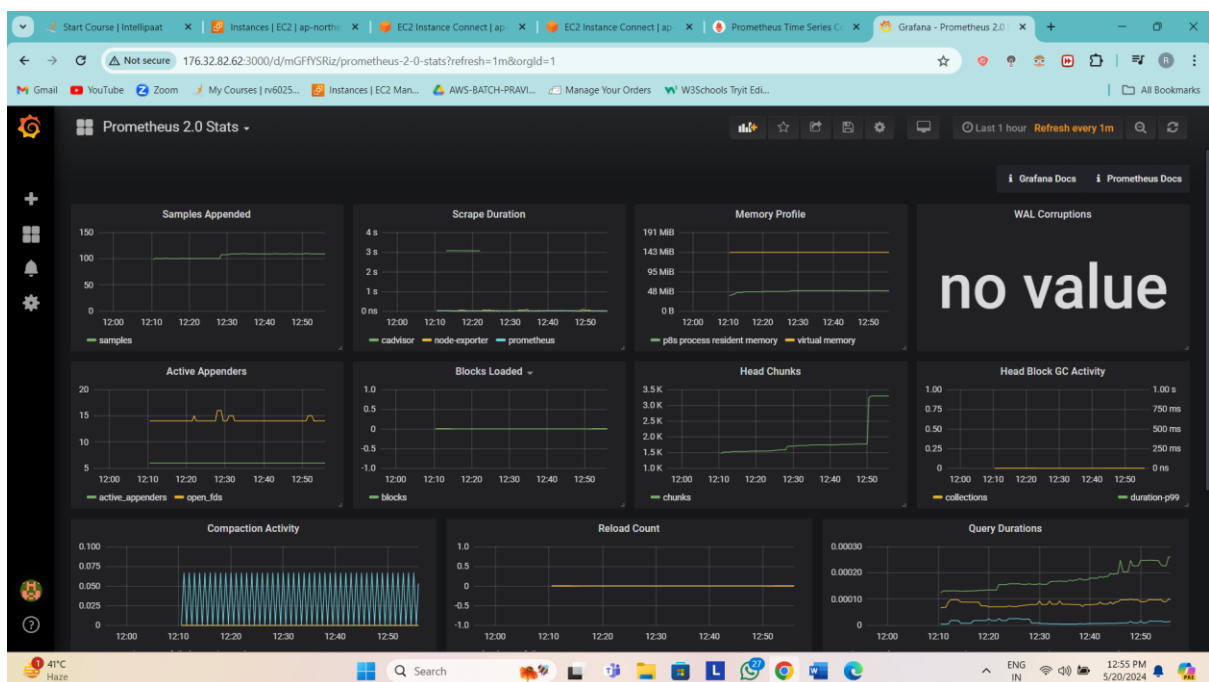
8. now go click on back and Head to dashboards – manage



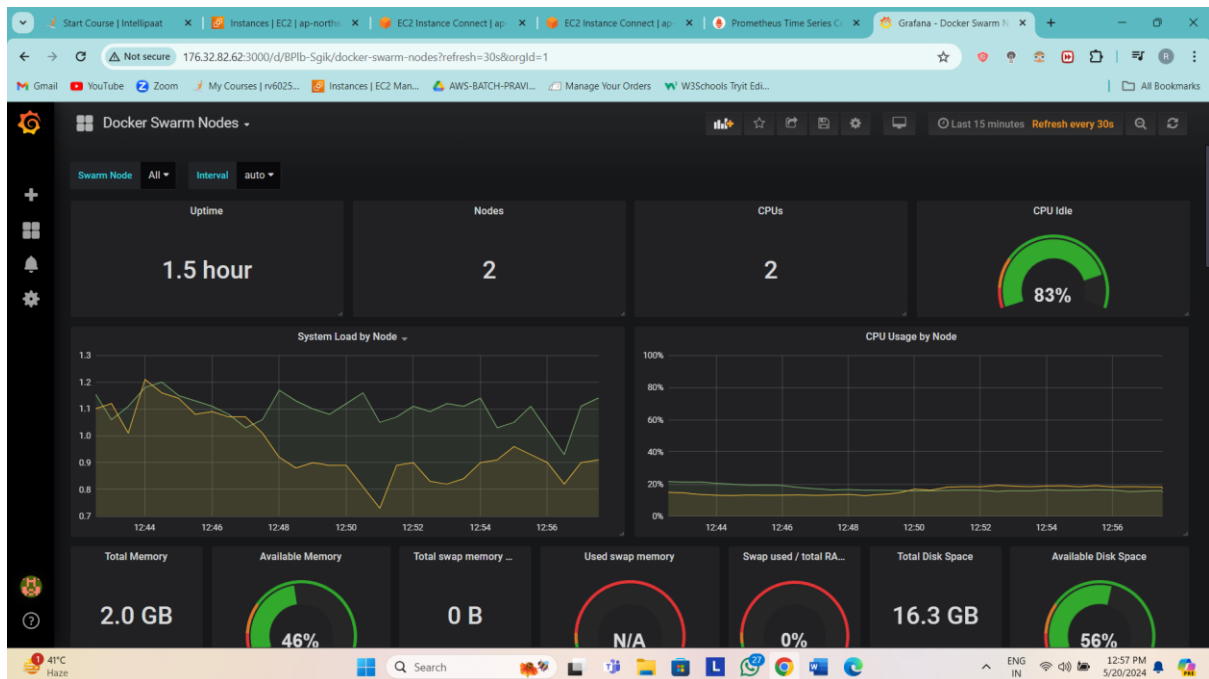
Here you will see three dashboards, normally you would have to create your own but we had already configured our Grafana to include these. Check all of them out.



9. This is the graph for Prometheus metrics.



10. This is the dashboard for Docker swarm Nodes



11. This is the Dashboard for docker swarm services

