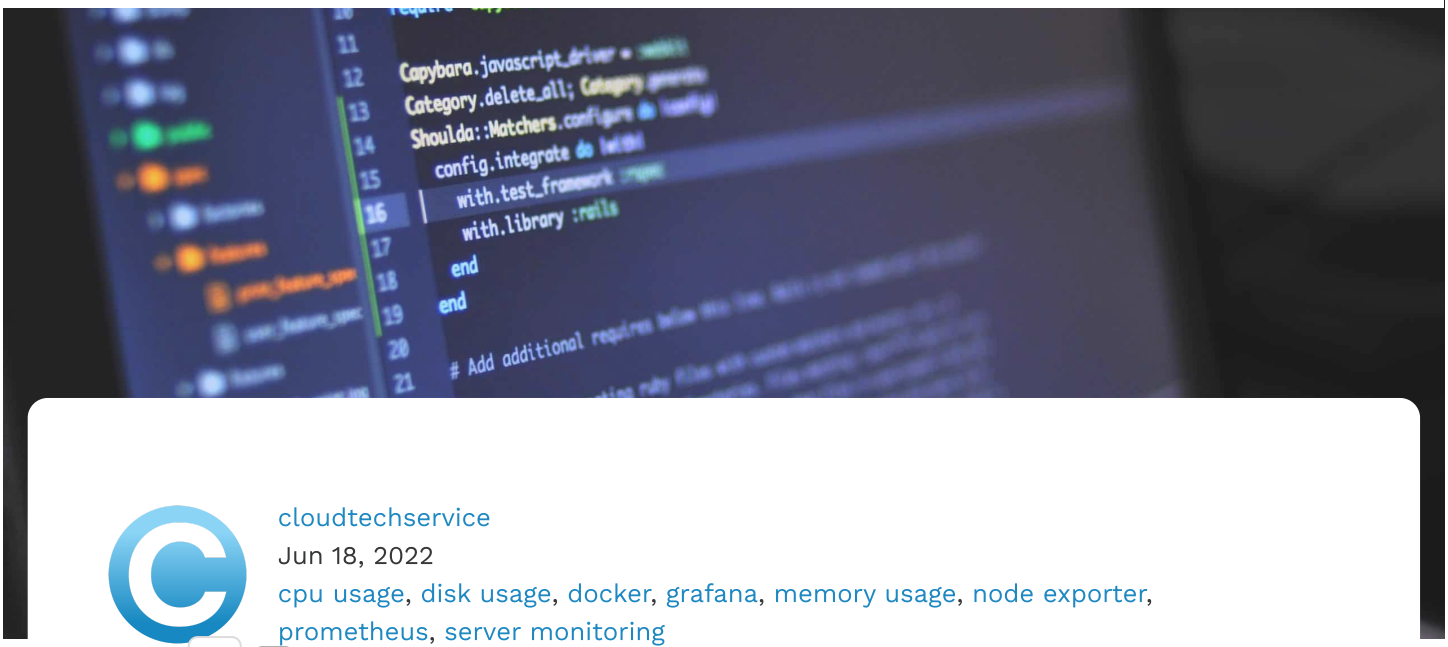


cloudtechservice
Jun 18, 2022
cpu usage, disk usage, docker, grafana, memory usage, node exporter, prometheus, server monitoring

Server Monitoring Guide: Using Prometheus, Grafana And Node_Exporter For Easy Server Monitoring

Share this

In this guide, we will learn how to install docker, docker-compose and learn the basics of docker-compose.yml file. We will also learn to configure prometheus, grafana and node_exporter in a container to monitor our server.

Server monitoring is really essential if we want to make sure that our applications are running smoothly. It is a basic need for every DevO engineer, System Administrator and Developers as well. It shows us

the health of our server and the insights provided by monitoring tool can prevent us from heading into a critical issue.

Prometheus and grafana are really popular monitoring tools that have a very large community. We will setup all the necessary tools in ubuntu server.

Page Contents

Part 1: Installing Docker

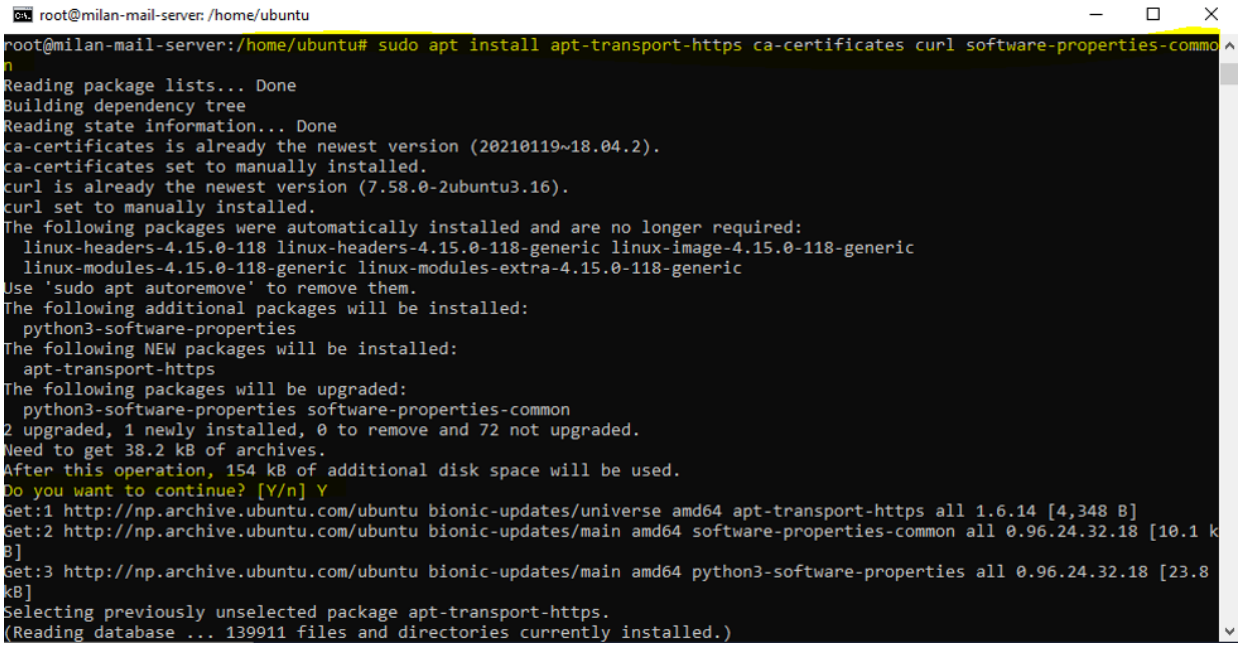
Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Use these commands and you can also check screenshots for reference:

```
sudo apt update
```

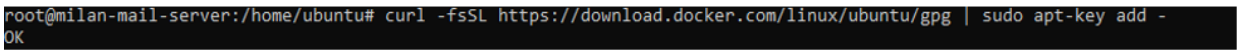
Next, install a few prerequisite packages which let apt use packages over HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```



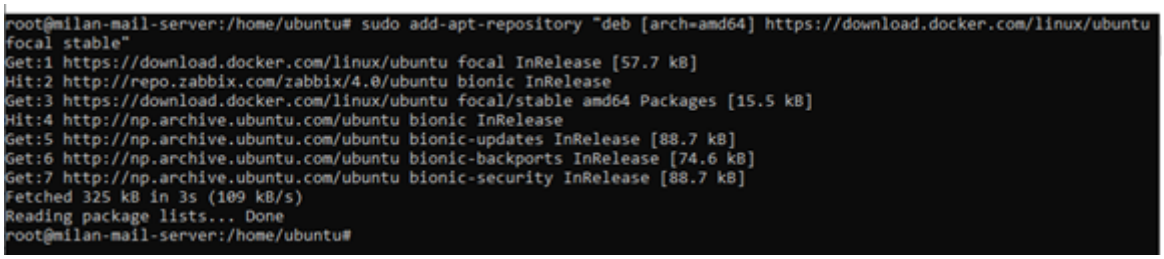
Then add the GPG key for the official Docker repository to your system:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```



Add the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```



This will also update our package database with the Docker packages from the newly added repo.

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

```
apt-cache policy docker-ce
```

```
root@milan-mail-server:/home/ubuntu# apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.14~3-0~ubuntu-focal
  Version table:
     5:20.10.14~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

Notice that docker-ce is not installed, but the candidate for installation is from the Docker.

Finally, install Docker:

```
sudo apt install docker-ce
```

```
root@milan-mail-server:/home/ubuntu# sudo apt install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.15.0-118 linux-headers-4.15.0-118-generic linux-image-4.15.0-118-generic
  linux-modules-4.15.0-118-generic linux-modules-extra-4.15.0-118-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
0 upgraded, 7 newly installed, 0 to remove and 72 not upgraded.
Need to get 96.4 MB of archives.
After this operation, 405 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io amd64 1.5.11-1 [22.9 MB]
Get:2 http://np.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:3 http://np.archive.ubuntu.com/ubuntu bionic/main amd64 libltdl7 amd64 2.4.6-2 [38.8 kB]
Get:4 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce-cli amd64 5:20.10.14~3-0~ubuntu-focal [41.0 MB]
Get:5 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce amd64 5:20.10.14~3-0~ubuntu-focal [20.9 MB]
Get:6 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce-rootless-extras amd64 5:20.10.14~3-0~ubuntu-focal [7,932 kB]
Get:7 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-scan-plugin amd64 0.17.0~ubuntu-focal [3,521 kB]
Fetched 96.4 MB in 10s (9,515 kB/s)
```

Check the status of docker:

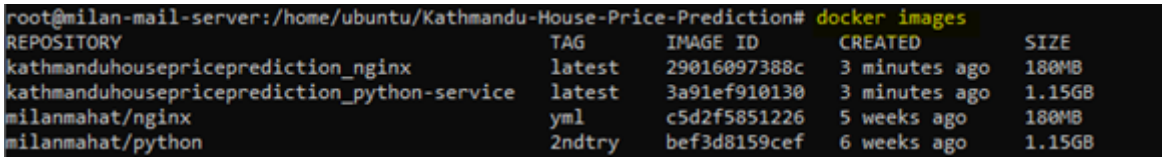
```
sudo systemctl status docker
```

```
root@milan-mail-server:/home/ubuntu# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-04-26 11:03:41 UTC; 1min 36s ago
     Docs: https://docs.docker.com
 Main PID: 6407 (dockerd)
    Tasks: 7
   CGroup: /system.slice/docker.service
           └─6407 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Apr 26 11:03:34 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:34.218950242Z" level=info msg="ClientConn switching balancer to \"pick_first\"" mo
Apr 26 11:03:37 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:37.815915291Z" level=warning msg="Your kernel does not support swap memory limit"
Apr 26 11:03:37 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:37.816583129Z" level=warning msg="Your kernel does not support CPU realtime schedu
Apr 26 11:03:37 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:37.817061822Z" level=info msg="Loading containers: start."
Apr 26 11:03:39 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:39.451687071Z" level=info msg="Default bridge (docker0) is assigned with an IP add
Apr 26 11:03:40 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:40.196002324Z" level=info msg="Loading containers: done."
Apr 26 11:03:41 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:41.764622636Z" level=info msg="Docker daemon" commit=87a90dc graphdriver(s)=overla
Apr 26 11:03:41 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:41.766243368Z" level=info msg="Daemon has completed initialization"
Apr 26 11:03:41 milan-mail-server systemd[1]: Started Docker Application Container Engine.
Apr 26 11:03:41 milan-mail-server dockerd[6407]: time="2022-04-26T11:03:41.814917490Z" level=info msg="API listen on /var/run/docker.sock"
lines 1-19/19 (END)
```

Initially you won't have any images and containers.

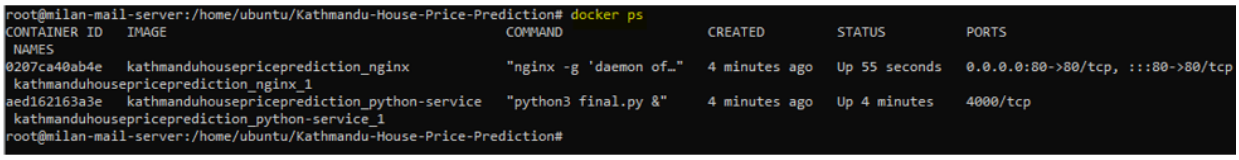
To view images:

```
sudo docker images
```



To view running containers:

```
sudo docker ps
```



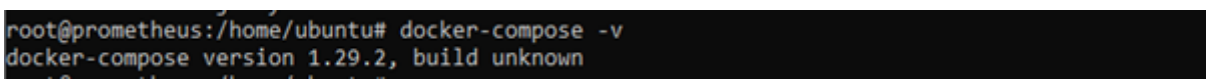
Part 2: Installing Docker-Compose

We don't need to do much while installing docker-compose. Just follow these steps:

```
sudo apt-get update -y
sudo apt-get install docker-compose -y
```

Check your docker-compose version after its installed:

```
sudo docker-compose --version
```



Part 3: Setting up prometheus and grafana:

After installing docker-compose we can easily setup prometheus and grafana using docker-compose.yml file. While we are doing this let's

try to monitor our host machine (server) as well. We need node_exporter for this.

Lets configure docker-compose.yml and start up our containers

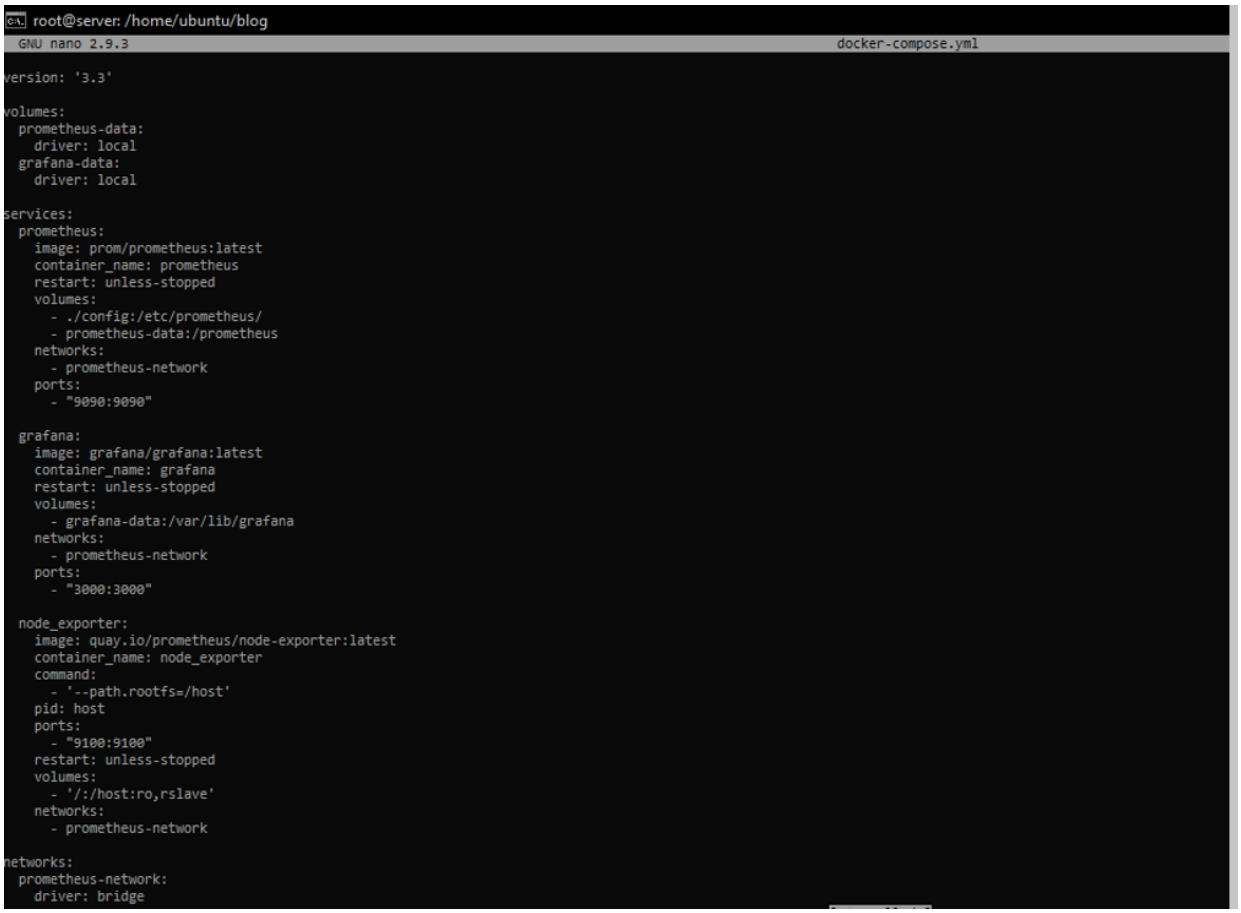
```
sudo mkdir prometheus

cd prometheus
sudo nano docker-compose.yml
```

Now insert the following code into it:

```
version: '3.3'
volumes:
  prometheus-data:
    driver: local
  grafana-data:
    driver: local
services:
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    restart: unless-stopped
    volumes:
      - ./config:/etc/prometheus/
      - prometheus-data:/prometheus
    networks:
      - prometheus-network
    ports:
      - "9090:9090"
  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    restart: unless-stopped
    volumes:
      - grafana-data:/var/lib/grafana
    networks:
      - prometheus-network
    ports:
      - "3000:3000"
  node_exporter:
    image: quay.io/prometheus/node-exporter:latest
    container_name: node_exporter
    command:
```

```
      - '--path.rootfs=/host'
    pid: host
    ports:
      - "9100:9100"
    restart: unless-stopped
    volumes:
      - '/:/host:ro,rslave'
    networks:
      - prometheus-network
networks:
  prometheus-network:
    driver: bridge
```

Our configuration file looks like this:



In our docker-compose file:

version: Version 3.3 is specified.

volumes: Two volumes prometheus-data and grafana-data is created locally. It is created automatically in directory /var/lib/docker/volumes/

services: Two services prometheus and grafana is created; image and container name is specified for them respectively.

restart: restart is set to unless stopped i.e. if the container was running before the reboot, the container would be restarted once the system restarted.

volumes:

For Prometheus: – ./config:/etc/prometheus/ : A folder named config (inside our current folder of host/server) is mapped with /etc/Prometheus (inside the container prometheus) – prometheus-data:/prometheus : Our previously created volume prometheus-data is mapped with /Prometheus (inside the container prometheus)

For grafana: – grafana-data:/var/lib/grafana : Our previously created volume grafana-data is mapped with /var/lib/grafana (inside the container grafana)

For node_exporter: – '/:/host:ro,rslave' : Root directory of the host i.e. '/' is mapped as '/host' in readonly mode as a slave (inside the container node_exporter).

pid: pid of node_exporter is shared with host. This is done for process monitoring.

command: In case of node_exporter. Command '–path.rootfs=/host' is specified and it is executed while our container starts . It is done for monitoring the root disk of our host.

networks: All containers reside in same network i.e. prometheus-network

ports: Port mapping is done from host port to container port.

networks: A network named prometheus-network is defined whose driver mode is bridge.

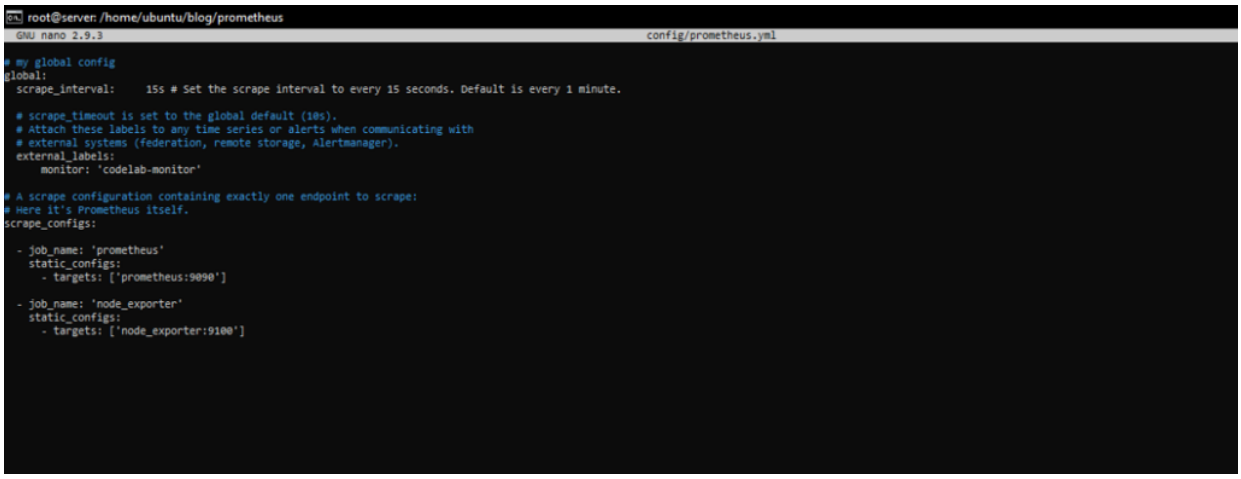
Also, we need a config file for Prometheus. So let's create it.

```
mkdir config
```

```
sudo nano config/prometheus.yml
```

Copy the following code inside it:

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15
seconds. Default is every 1 minute.

  # scrape_timeout is set to the global default (10s).
  # Attach these labels to any time series or alerts when
communicating with
  # external systems (federation, remote storage, Alertmanager).
  external_labels:
      monitor: 'codelab-monitor'
# A scrape configuration containing exactly one endpoint to
scrape:
# Here it's Prometheus itself.
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['prometheus:9090']
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['node_exporter:9100']
```



Now lets start our containers:

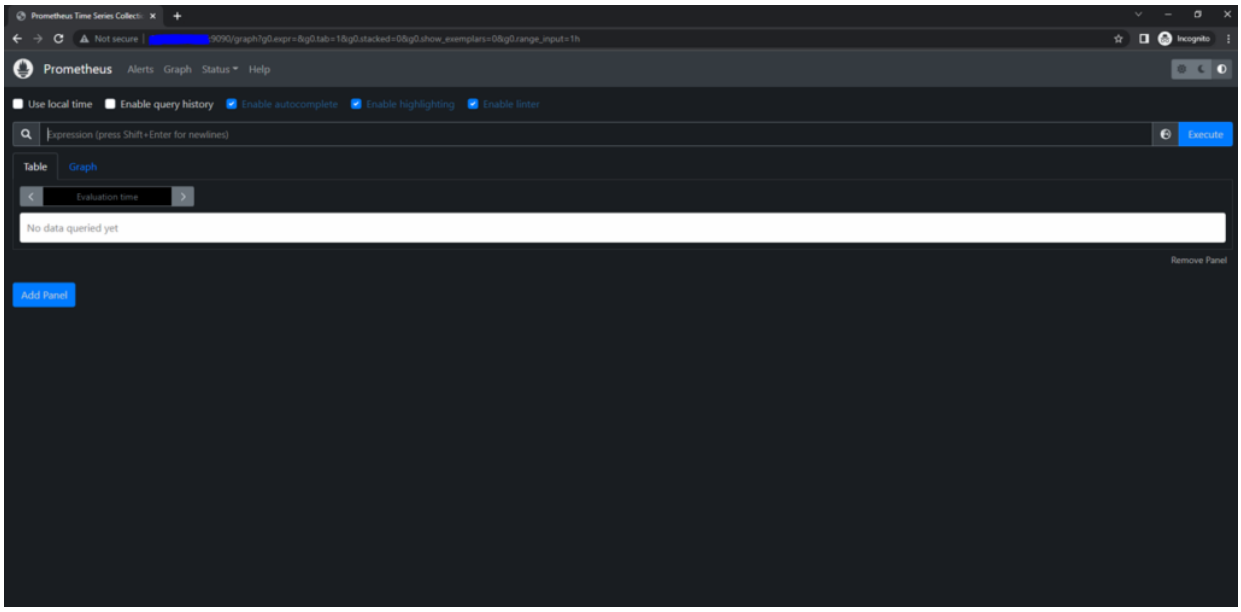
```
docker-compose up --build -d
```

and check the status:

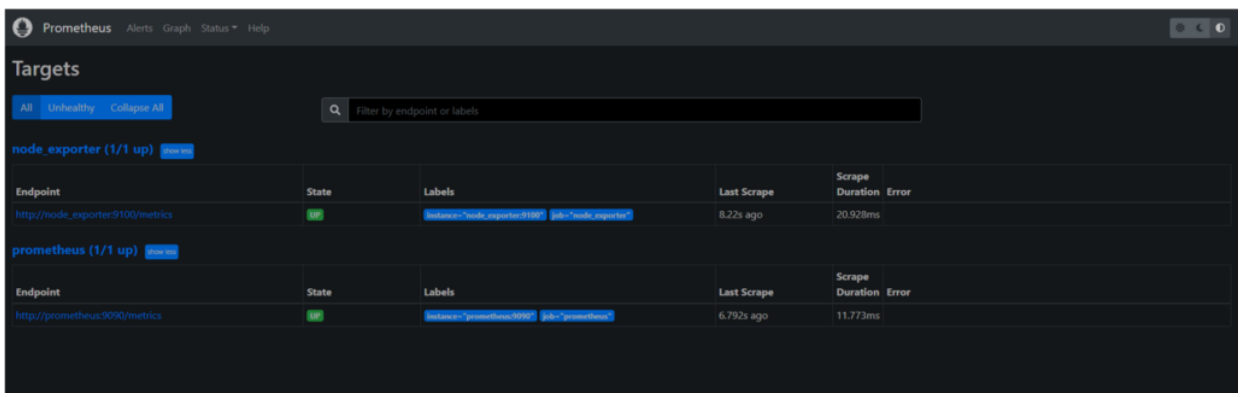
```
docker-compose ps
```

```
root@server: /home/ubuntu/blog/prometheus
root@server:/home/ubuntu/blog/prometheus# docker-compose up --build -d
Creating network "prometheus_prometheus-network" with driver "bridge"
Creating prometheus    ... done
Creating node_exporter ... done
Creating grafana       ... done
root@server:/home/ubuntu/blog/prometheus# docker-compose ps
    Name                  Command               State                    Ports
-------------------------------------------------------------------------------------------------
grafana         /run.sh                         Up      0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
node_exporter   /bin/node_exporter --path. ...   Up      0.0.0.0:9100->9100/tcp,:::9100->9100/tcp
prometheus      /bin/prometheus --config.f ...   Up      0.0.0.0:9090->9090/tcp,:::9090->9090/tcp
root@server:/home/ubuntu/blog/prometheus#
```

Looks like everything is running fine. Now go to http://yourserverip:9090/ to access Prometheus:



Go to Status->targets.



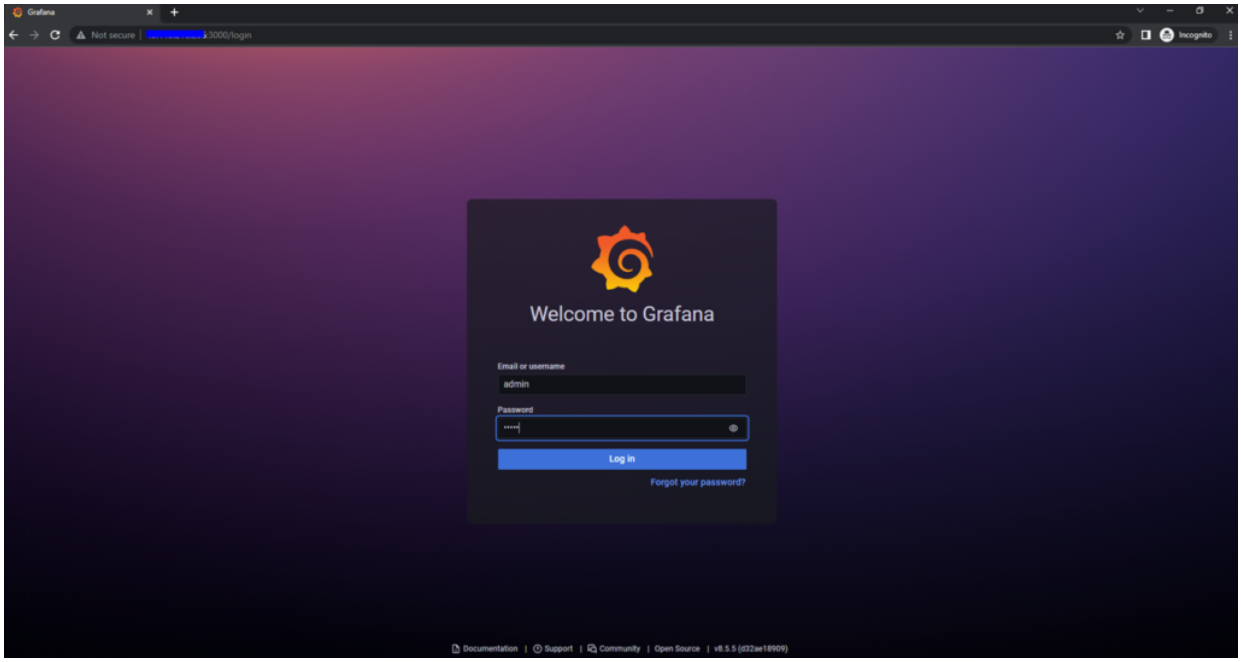
Here we can see that both our targets prometheus and node_exporter is up.

It means that prometheus can scrape data from these endpoints.

However, we won't be discussing about prometheus for now. We will be using grafana to display the metrics that is scraped by prometheus.

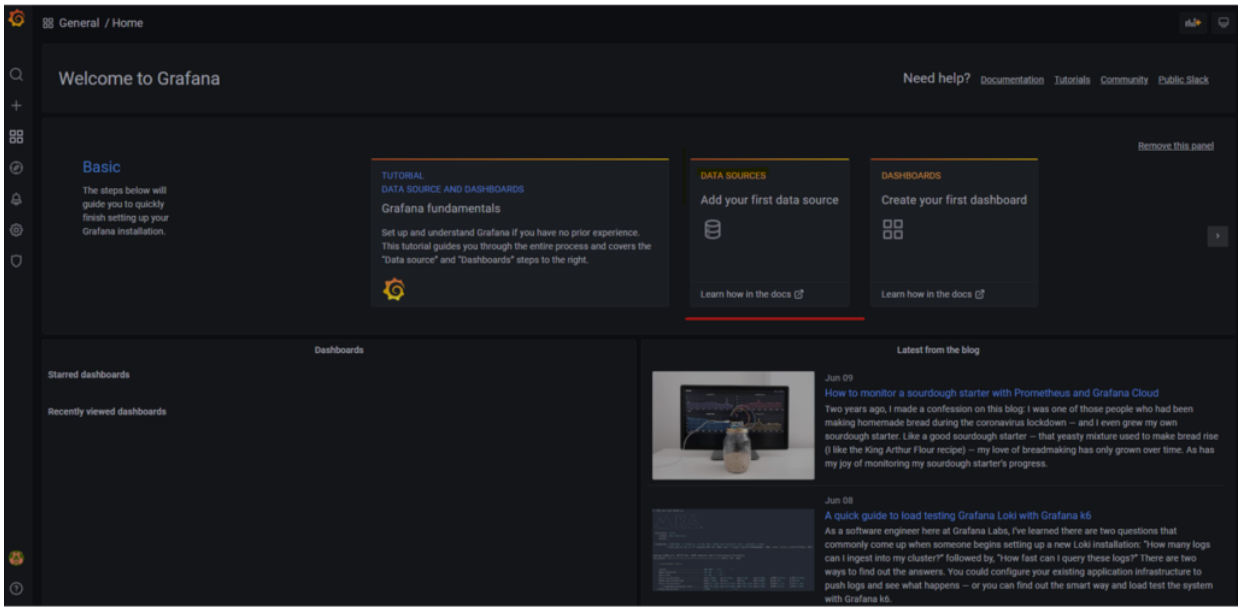
Let's login to our grafana server.

Goto http://yourserverip:3000/

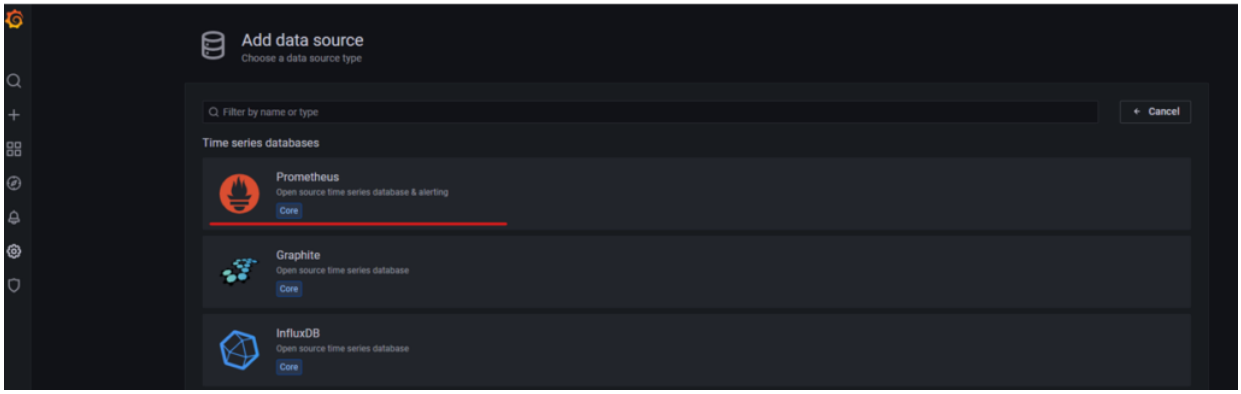


Default username and password is admin.

After logging in click on Add your first data source:



Select Prometheus:

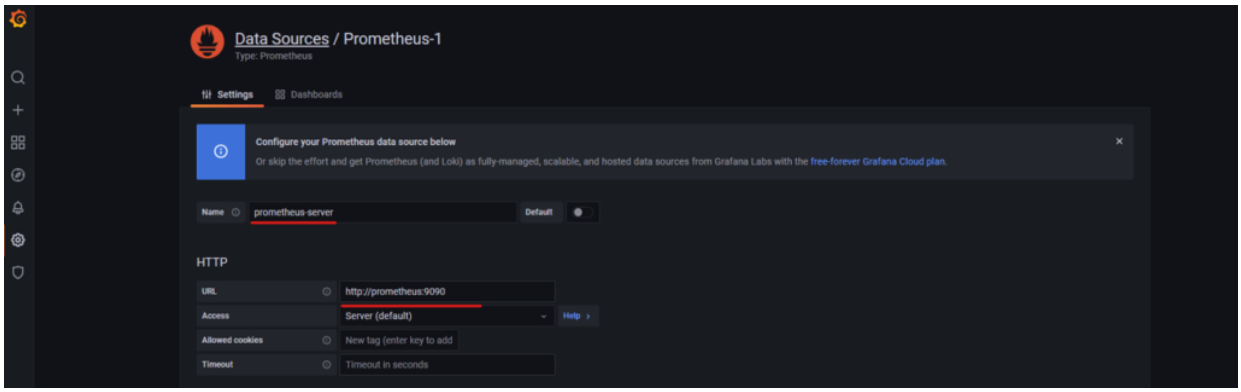


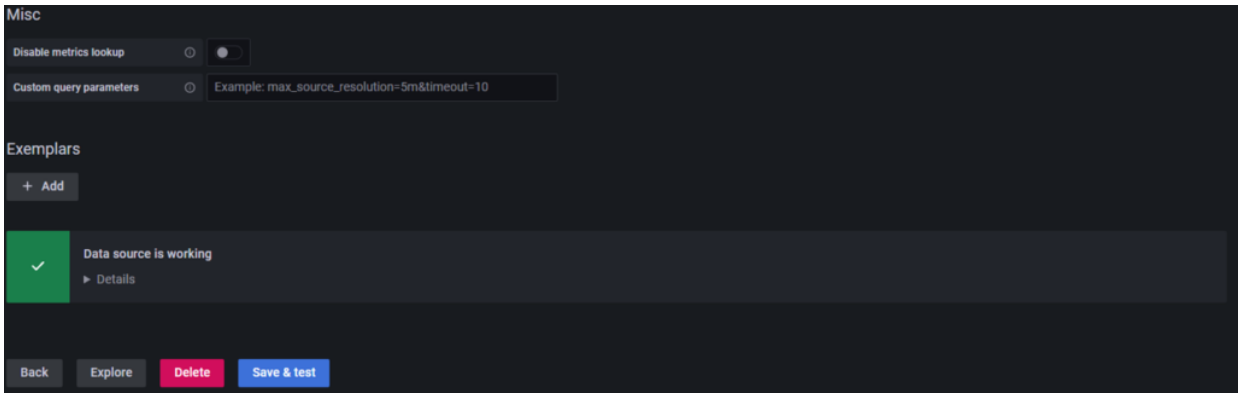
Set your data source name and enter url for making connection to prometheus.

In our case set this url:

http://prometheus:9090

This above url works only because grafana and prometheus are in same network in docker.

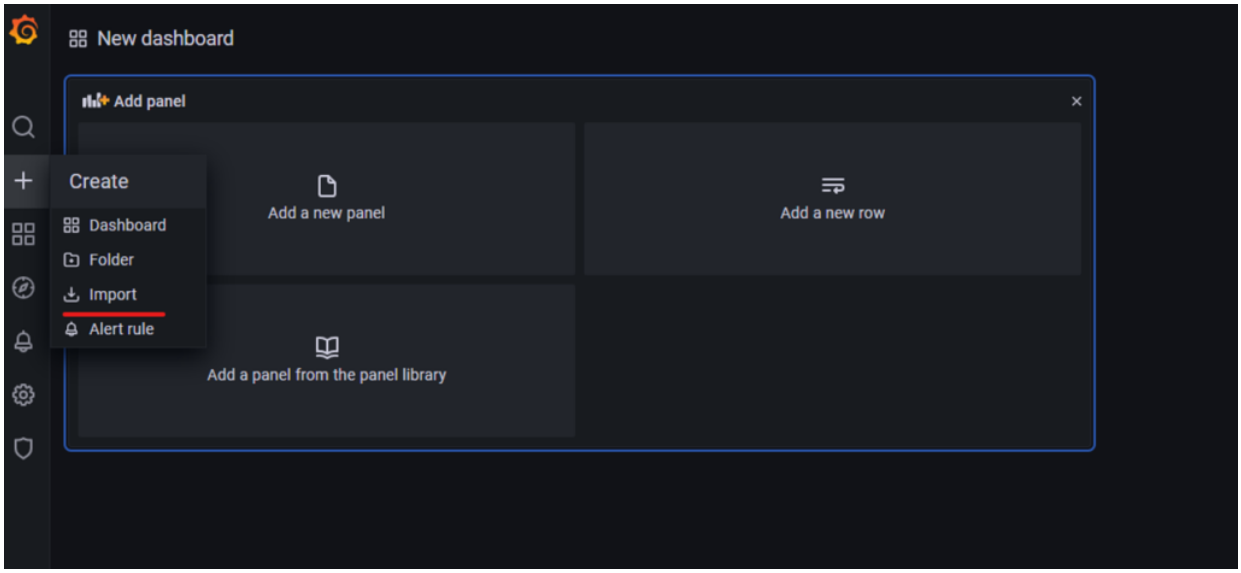




Click on Save and test.

Now let’s setup our dashboard for host monitoring.

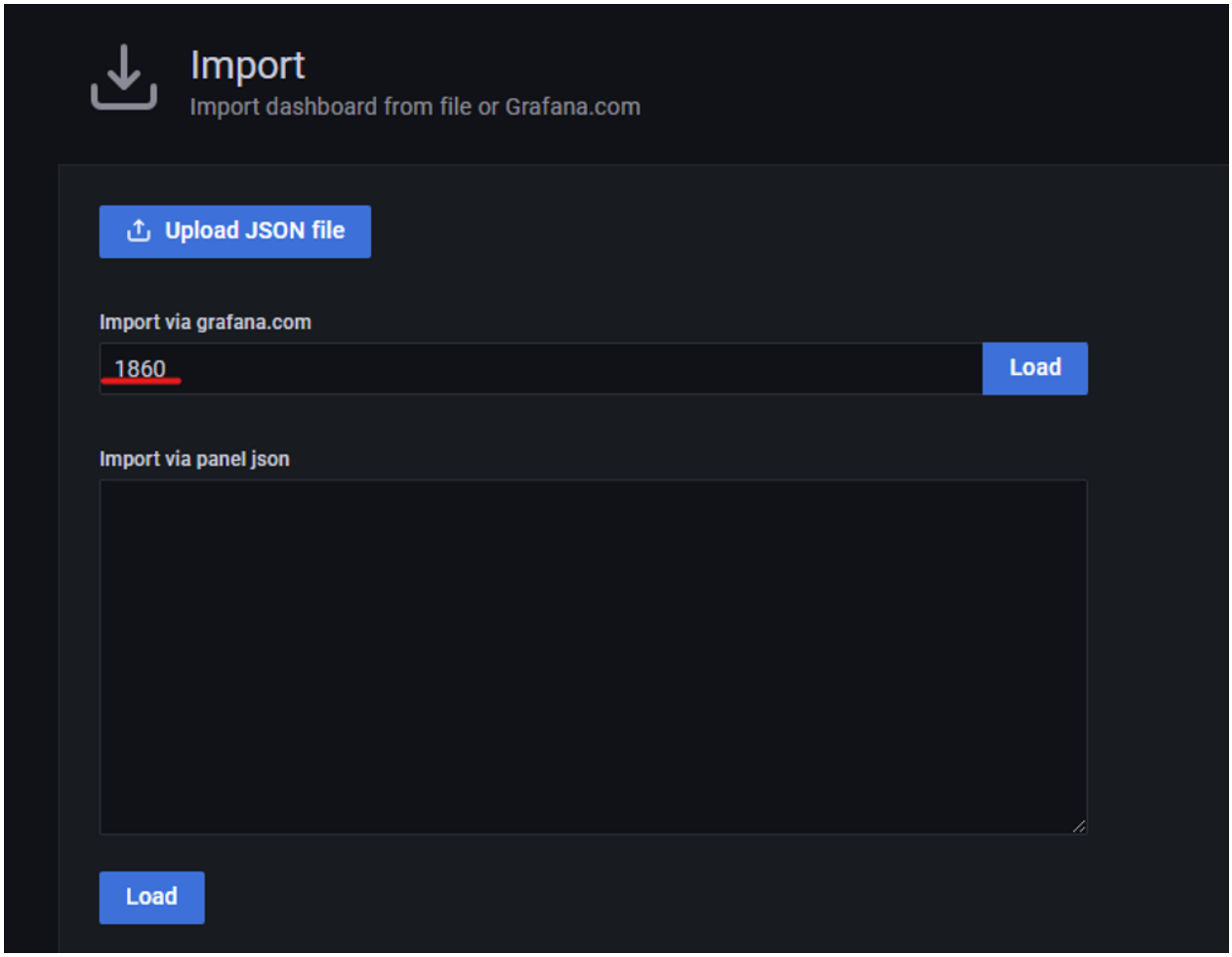
Go to import:



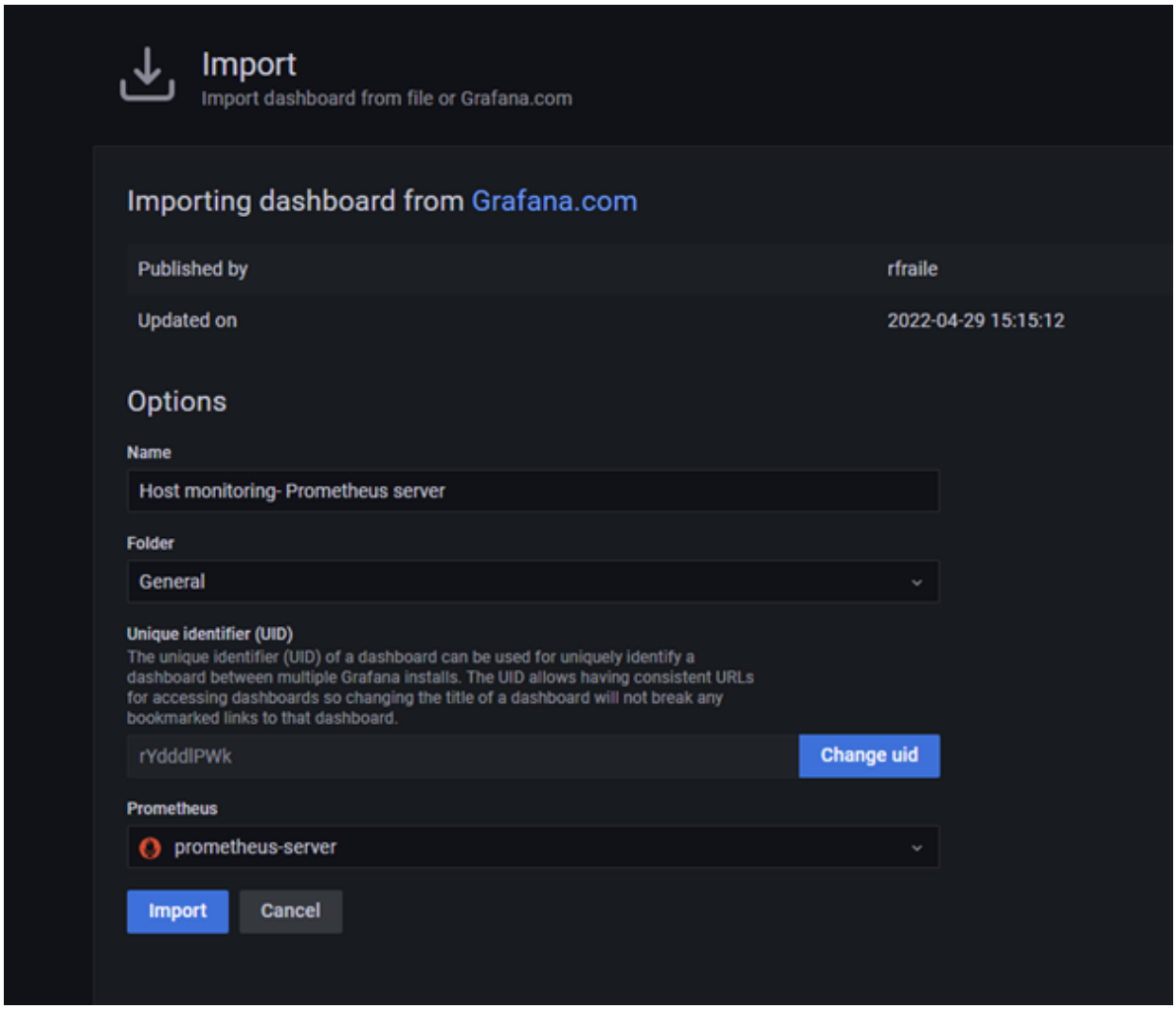
Now paste 1860 and click on Load

This ID helps us to export full node_exporter dashboard from grafana dashboard library. You can browser more dashboards on https://grafana.com/grafana/dashboards/.

Click on load.



Set name, set the correct preometheus and click on import:



After importing you can see a dashboard. This dashboard is readymade dashboard and contains a lot of information.

Your dashboard has been loaded along with various metrics such as CPU utilization, memory utilization, network traffic utilization, etc. There are a lot of various metrics that you can explore and customize as you need.

Thats it!! Its really easy to monitor your server using this method.

Leave a Reply

You must be logged in to post a comment.

Previous: Project Data Analysis: Best Way To Analyze Project Data

Next: Container Monitoring Guide: Using Cadvisor, Prometheus And Grafana For Easy Docker Contaner Monitoring

Enrollment Center

Payment Solution

Billing Solution

Electronic File Transfer

Ticketing Module

Commission Module

Communication Center

Education Management Software (EMS)

Services

Web Solutions

Mobile Application

Seo/Analytics

Managed Services/Hosting

Data Integration

Data Quality

Master Data

Connect with us

Facebook

LinkedIn

Tech