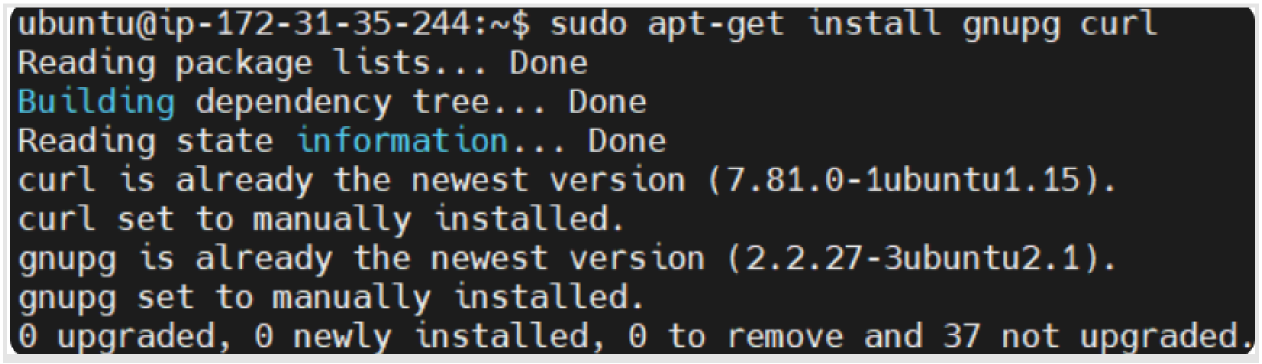
Step #1:Install MongoDB on Ubuntu

Update the system before start installation process.

```
sudo apt update
```

Then import the public key used by the package management system.

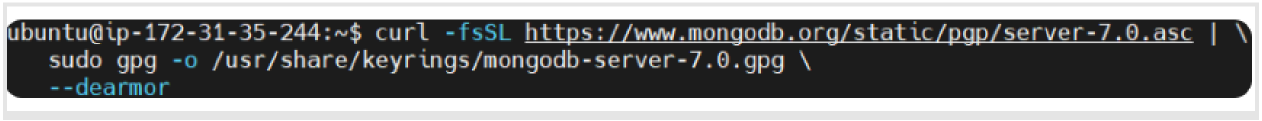
```
sudo apt-get install gnupg curl
```



```
ubuntu@ip-172-31-35-244:~$ sudo apt-get install gnupg curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.15).
curl set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
```

To import the MongoDB public GPG key, run the following command.

```
curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc | \
   sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \
   --dearmor
```



```
ubuntu@ip-172-31-35-244:~$ curl -fsSL https://www.mongodb.org/static/pgp/server-7.0.asc | \
   sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \
   --dearmor
```

Create the `/etc/apt/sources.list.d/mongodb-org-7.0.list` file for Ubuntu 22.04.

```
echo "deb [ arch=amd64,arm64
signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ]
https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/7.0 multiverse"
| sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list
```

Reload the local package database using following command.

```
sudo apt-get update
```

Install the latest stable version of MongoDB using following command.

```
sudo apt-get install -y mongodb-org
```

After this reload the daemon service and also enable the mongod and after that start the mongod.

```
sudo systemctl daemon-reload
sudo systemctl enable mongod
sudo systemctl start mongod
```

Check the status to if MongoDB is running successfully.

```
sudo systemctl status mongod
```

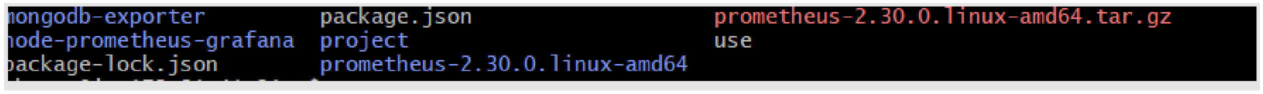
Step #2:Install Prometheus on Ubuntu

From the github repo, download the latest version of Prometheus using the following command.

```
wget
https://github.com/prometheus/prometheus/releases/download/v2.30.0/pr
ometheus-2.30.0.linux-amd64.tar.gz
```

Extract the downloaded archives after the completion of download.

```
tar xvfz prometheus-2.30.0.linux-amd64.tar.gz
```



```
nongodb-exporter         package.json                  prometheus-2.30.0.linux-amd64.tar.gz
node-prometheus-grafana  project                       use
package-lock.json        prometheus-2.30.0.linux-amd64
```

The command extracts the contents of the file `prometheus-2.30.0.linux-amd64.tar.gz` now let’s move into the extracted directory using following command.

```
cd prometheus-2.30.0.linux-amd64
```

navigate to the /etc/systemd/system, this is where typically systemd unit files are located, which are used for managing services on Linux systems.

```
cd /etc/systemd/system
```

now lets create a service named prometheus.service for Prometheus.

```
sudo vi prometheus.service
```

add the following block into it.

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
ExecStart=/home/ubuntu/prometheus-2.30.0.linux-amd64/prometheus
--config.file=/home/ubuntu/prometheus-2.30.0.linux-amd64/prometheus.y
ml
Restart=always

[Install]
WantedBy=default.target
```

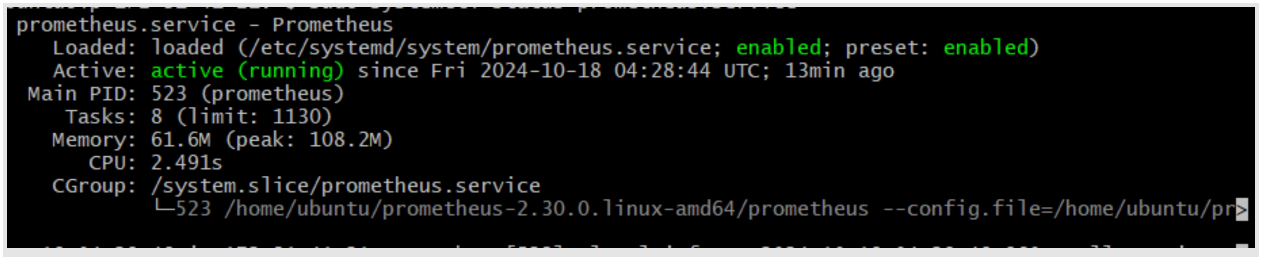
Now we’ve created our prometheus.service.

After this firstly reload the daemon service to verify our configuration file is correct then enable the Prometheus service and at last start the service.

```
sudo systemctl daemon-reload
sudo systemctl enable prometheus.service
sudo systemctl start prometheus.service
```

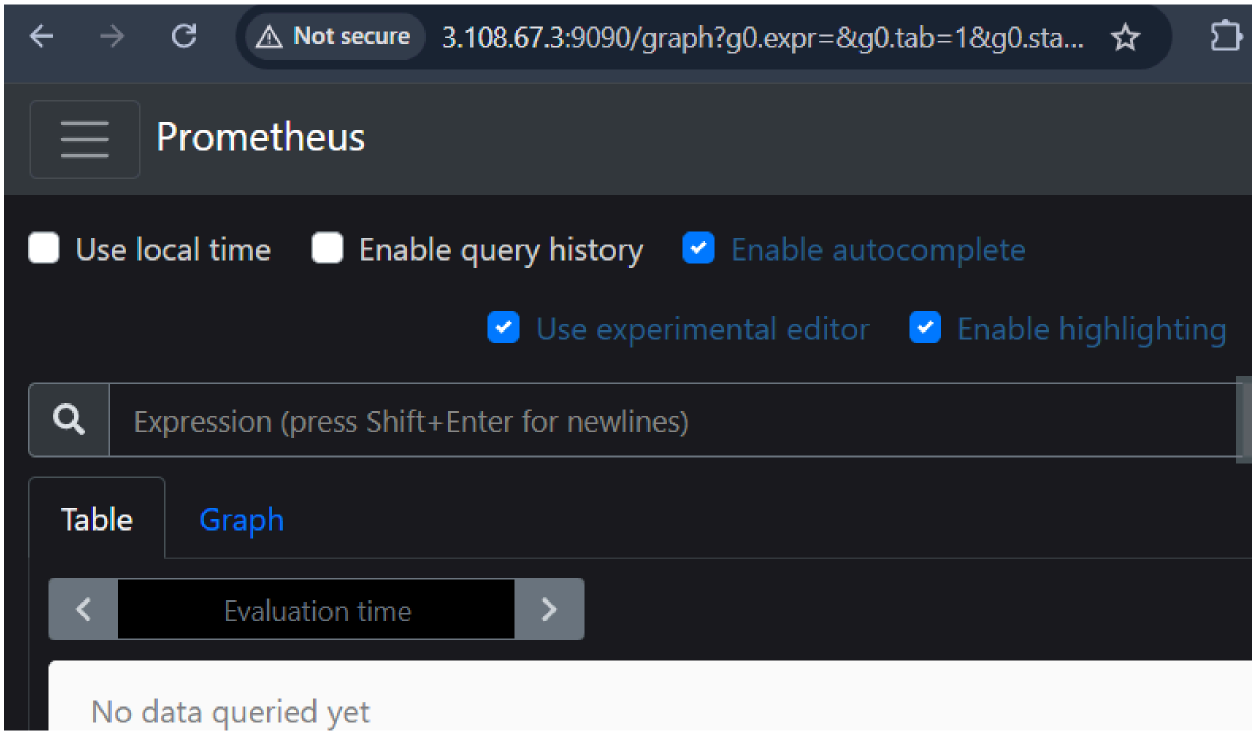
now check the status to see if the Prometheus service is running properly by running following command:

```
sudo systemctl status prometheus.service
```



```
prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-10-18 04:28:44 UTC; 13min ago
 Main PID: 523 (prometheus)
    Tasks: 8 (limit: 1130)
   Memory: 61.6M (peak: 108.2M)
      CPU: 2.491s
   CGroup: /system.slice/prometheus.service
           └─523 /home/ubuntu/prometheus-2.30.0.linux-amd64/prometheus --config.file=/home/ubuntu/pr>
```

If your service is running properly then you can run prometheus by running your public ip address with port 9090 which is default port for prometheus in url.


Not secure
3.108.67.3:9090/graph?g0.expr=&g0.tab=1&g0.sta...
Prometheus
Expression (press Shift+Enter for newlines)
Table
Graph
Evaluation time

Use local time ☐ · Enable query history ☐ · Enable autocomplete ✓ · Use experimental editor ✓ · Enable highlighting ✓

No data queried yet

Step #3:Install Grafana on Ubuntu

First import the GPG key used by the Grafana package.

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add
-
```

Add the Grafana repository to the APT sources.

```
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb
stable main"
```

Update the package lists before installing in Grafana.

```
sudo apt update
```

now lets install the grafana

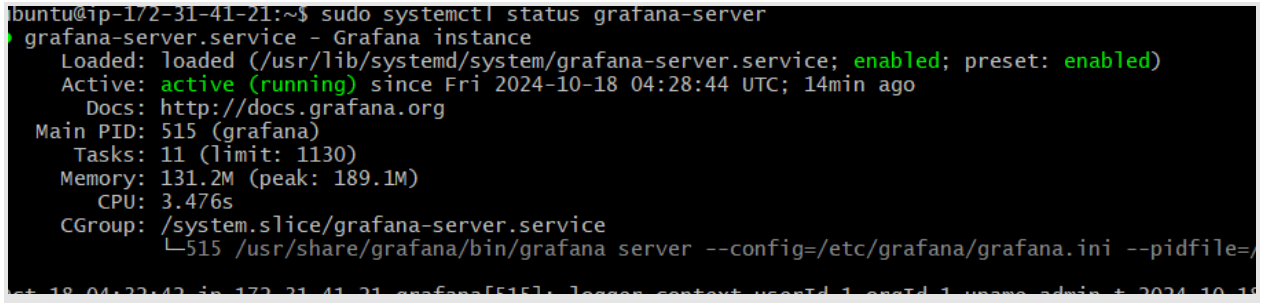
```
sudo apt install grafana
```

Start the Grafana service

```
sudo systemctl start grafana-server
```

then enable the grafana service.

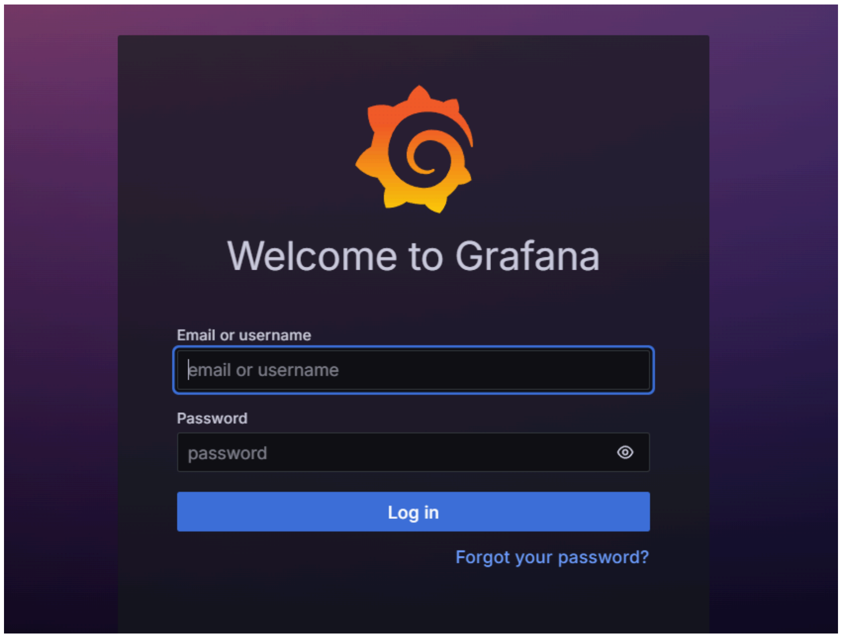
```
sudo systemctl enable grafana-server
```



```
ubuntu@ip-172-31-41-21:~$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-10-18 04:28:44 UTC; 14min ago
       Docs: http://docs.grafana.org
   Main PID: 515 (grafana)
      Tasks: 11 (limit: 1130)
     Memory: 131.2M (peak: 189.1M)
        CPU: 3.476s
     CGroup: /system.slice/grafana-server.service
             └─515 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/

ct 18 04:33:42 ip-172-31-41-21 grafana[515]: logger=context userId=1 orgId=1 uname=admin t=2024-10-18
```

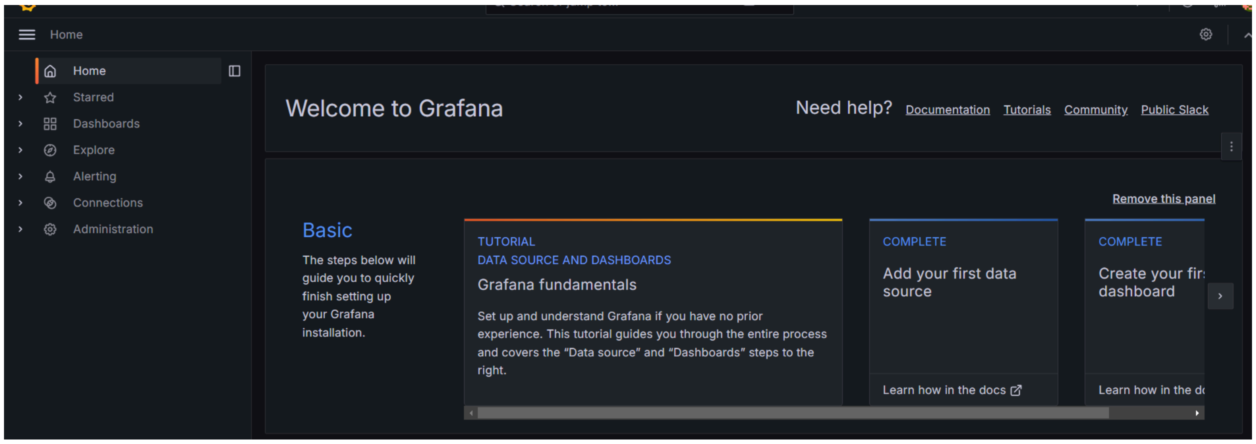
and if everything works fine and you see your service is running properly then run the grafana by running your public ip address and port number 3000 which is default port of grafana in searchbar.

you will see the login page of grafana (UI) user interface and Grafana has `admin` as default username and password



it will ask for changing the password you can change the password or skip it.

then you will see the welcome page of grafana.


Home
Home
Starred
Dashboards
Explore
Alerting
Connections
Administration
Welcome to Grafana
Need help? Documentation Tutorials Community Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your fir
dashboard
Learn how in the d

Step #4:Create User Group and User for Prometheus

Now lets create User Group and User for Prometheus after the installation of Prometheus and grafana is completed.

```
sudo groupadd --system prometheus
```

```
sudo useradd -s /sbin/nologin --system -g prometheus prometheus
```

Monitor MongoDB with Prometheus and Grafana

Step #5:Download and install MongoDB exporter

First create a directory for the exporter and navigate to it.

```
mkdir mongodb-exporter
```

```
cd mongodb-exporter
```

Download the binary file of the exporter from the github repo.

```
wget
https://github.com/percona/mongodb_exporter/releases/download/v0.7.1/
mongodb_exporter-0.7.1.linux-amd64.tar.gz
```

Now in your current folder extract the downloaded archive.

```
tar xvzf mongodb_exporter-0.7.1.linux-amd64.tar.gz
```

Finally, move the `mongodb_exporter` binary to `usr/local/bin/`.

```
sudo mv mongodb_exporter /usr/local/bin/
```

Next set up MongoDB authentication for the MongoDB exporter and create a user to monitor the cluster's metrics. Begin by connecting with `mongosh`.

```
mongosh
```

After connecting to your MongoDB instance with `mongosh`, switch to the `admin` database using the command.

```
use admin
```

Then, create an administrator account for your exporter with the `clusterMonitor` role.

```
db.createUser({user: "test",pwd: "testing",roles: [{ role:
"clusterMonitor", db: "admin" },{ role: "read", db: "local" }]})
```

Exit the shell after creating the user.

```
exit
```

Step #6:Configuring MongoDB Exporter Service.

Next, set your MongoDB URI environment variable with the appropriate authentication credentials.

```
export MONGODB_URI=mongodb://test:testing@localhost:27017
```

To check that the environment variable was set correctly, run the following command.

```
env | grep mongodb
```

Now lets create a service for exporter. First navigate to the `/lib/systemd/system` folder.

```
cd /lib/systemd/system/
```

create a new service file for the exporter.

```
sudo nano mongodb_exporter.service
```

```
[Unit]
Description=MongoDB Exporter
User=prometheus

[Service]
Type=simple
Restart=always
ExecStart=/usr/local/bin/mongodb_exporter
```

```
[Install]
WantedBy=multi-user.target
```

save and close file.

Next, restart your system daemon to reload the unit files, enable the service and finally start the service.

```
sudo systemctl daemon-reload
sudo systemctl enable mongodb_exporter.service
sudo systemctl start mongodb_exporter.service
```

lastly check the service is running or not by running the following command.

```
sudo systemctl status mongodb_exporter.service
```

to make sure it is running properly you can run it on your local machine by running your public ip address with the port number 9216 which is default port for exporter.

Step #7:Configuring the MongoDB Exporter as a Prometheus target.

here you will configure the exporter as prometheus target, first go to the directory where prometheus is present

```
cd prometheus-2.30.0.linux-amd64
```

open the file for editing.

```
sudo nano prometheus.yml
```

add **localhost:9216** as shown below.

```
# my global config
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" , "localhost:9216"]
```

9216 is default port for exporter. Save and close your file.

Restart Prometheus after adding the target.

```
sudo systemctl restart prometheus
```

Now go to Prometheus dashboard and click on status, select target, you can see our exporter are up and running.

Step #8:Setting up Grafana Dashboards for MongoDB Metrics

In this step we will create a dashboard for MongoDB exporter in the grafana in order to view and analyze the metrics.

Go to `Home` at up left corner, go into the `Connections`, in the `Data sources`.

Search for the prometheus and select it.

In the connections, give the prometheus server url on which our prometheus server is running.

Click on the `save and test` button you will see the successful message as shown below.

Go to the `+` icon and select the `New dashboard` from up right corner.

Here you can start your own new dashboard by adding a visualization.

Click on the `Add visualization` box.

You can also import dashboards.

Select the prometheus as a data source.

Now you can start running the queries.

In the query section add the query A

- Metric: mongodb_exporter_last_scrape_duration_seconds
- instance: localhost:9216

In the query section add the query B

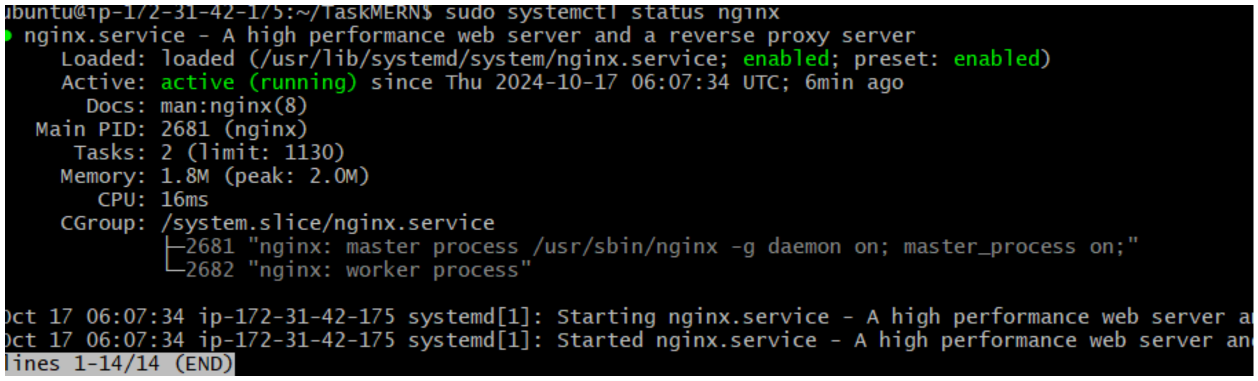
- Metric = mongodb_exporter_scrapes_total
- job = prometheus

Click on `run queries`.

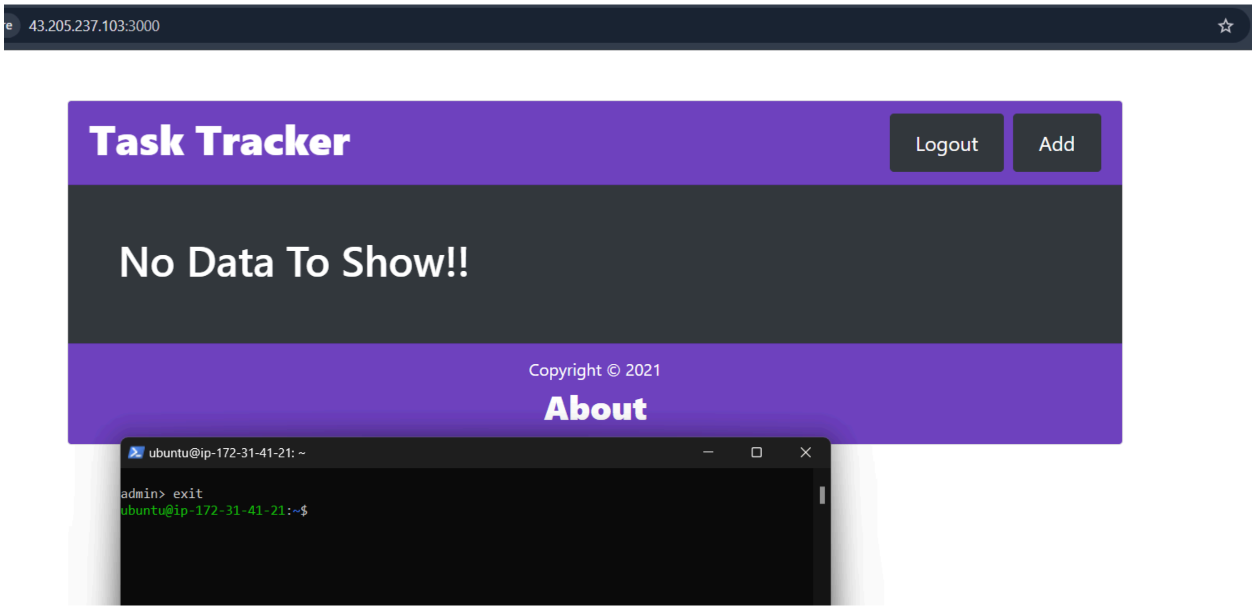
You will see the following output.

It's in the `Gauge` visualization, but there are many visualization option like `time series, stats, bar graph etc.`

Click `ctrl+s` to save the file. Give an appropriate Title for your dashboard the save it.



```
ubuntu@ip-172-31-42-175:~/TaskMERN$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-10-17 06:07:34 UTC; 6min ago
       Docs: man:nginx(8)
   Main PID: 2681 (nginx)
      Tasks: 2 (limit: 1130)
     Memory: 1.8M (peak: 2.0M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─2681 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2682 "nginx: worker process"

Oct 17 06:07:34 ip-172-31-42-175 systemd[1]: Starting nginx.service - A high performance web server a
Oct 17 06:07:34 ip-172-31-42-175 systemd[1]: Started nginx.service - A high performance web server an
lines 1-14/14 (END)
```

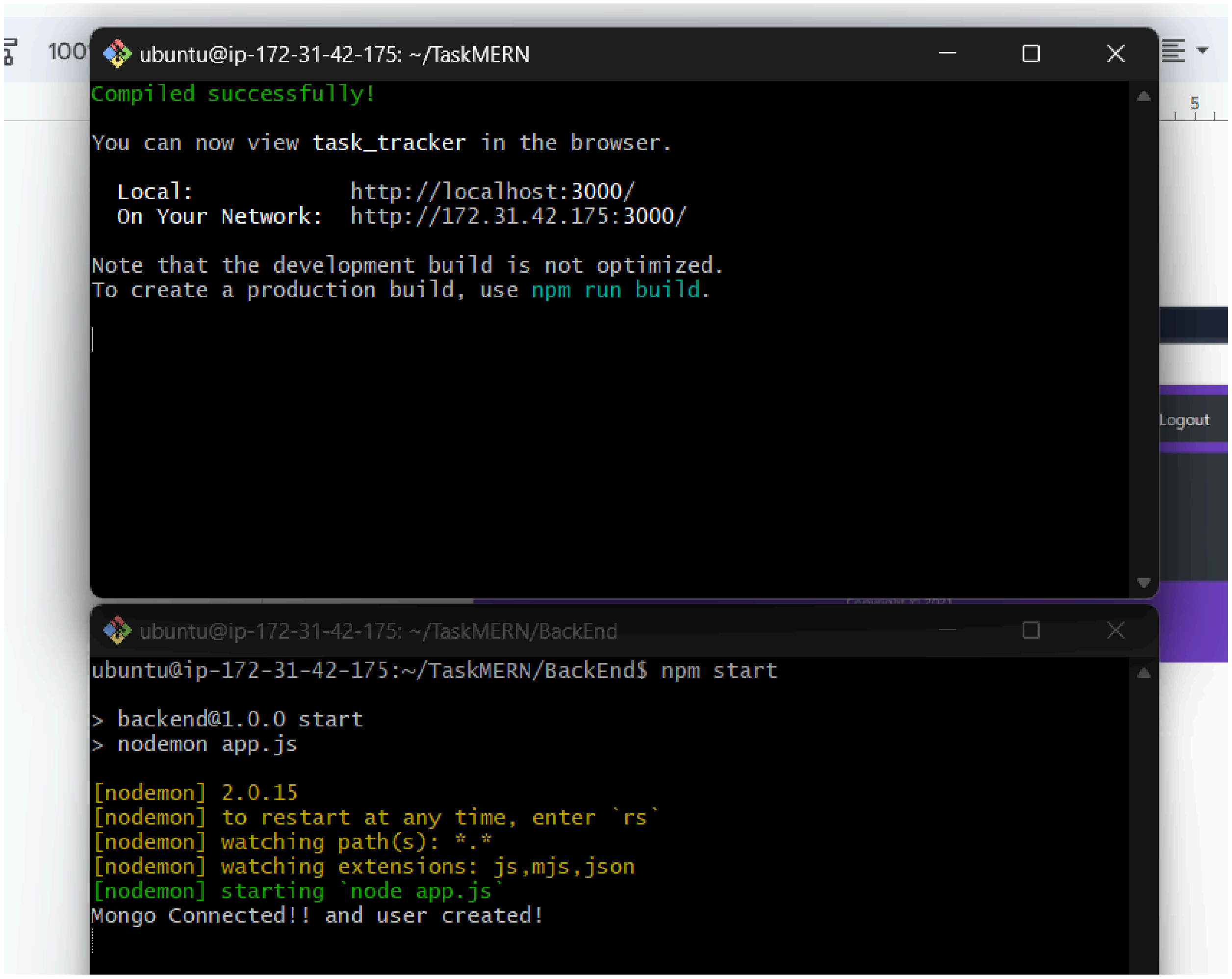


No Data To Show!!

Copyright © 2021

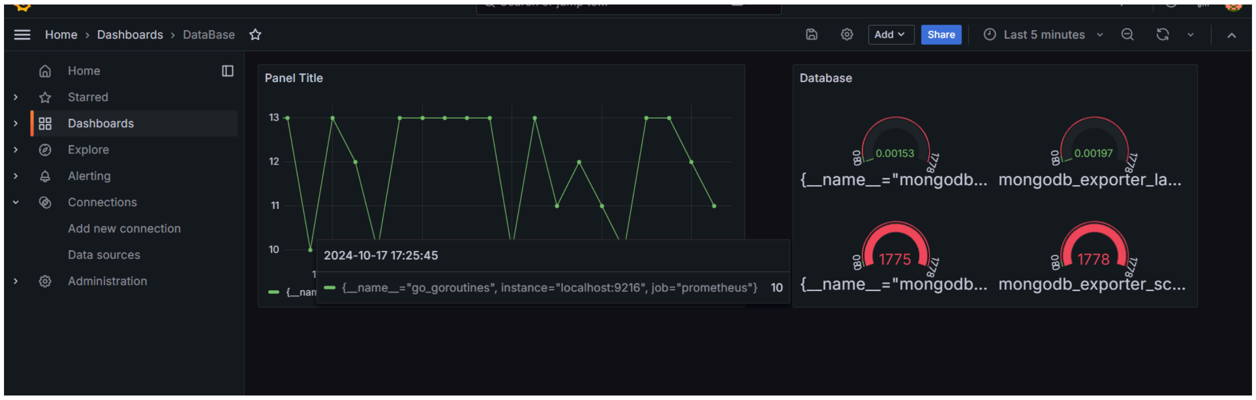
About

```
admin> exit
ubuntu@ip-172-31-41-21:~$
```



```
ubuntu@ip-172-31-42-175: ~/TaskMERN
Compiled successfully!

You can now view task_tracker in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://172.31.42.175:3000/

Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
ubuntu@ip-172-31-42-175: ~/TaskMERN/BackEnd
ubuntu@ip-172-31-42-175:~/TaskMERN/BackEnd$ npm start

> backend@1.0.0 start
> nodemon app.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Mongo Connected!! and user created!
```



2024-10-17 17:25:45
{__name__="go_goroutines", instance="localhost:9216", job="prometheus"} 10

Steps to Set Up Node.js Exporter with Grafana:

1. Install the Prometheus Client in the Node.js Application

First, install the `prom-client` package in your Node.js application. This package exposes metrics in a format Prometheus can scrape.

```
npm install prom-client
```

2. Expose Metrics in Your Node.js Application

3. Configure Prometheus to Scrape the Node.js Application

You need to configure Prometheus to scrape the /metrics endpoint of your Node.js app. Modify the prometheus.yml configuration file to add a new scrape target.

```
yaml

scrape_configs:
  - job_name: 'nodejs-app'
    static_configs:
      - targets: ['<your-nodejs-app-ip>:3000']
```

4. Restart Prometheus

After updating the configuration, restart Prometheus so it can start scraping metrics from your Node.js app.

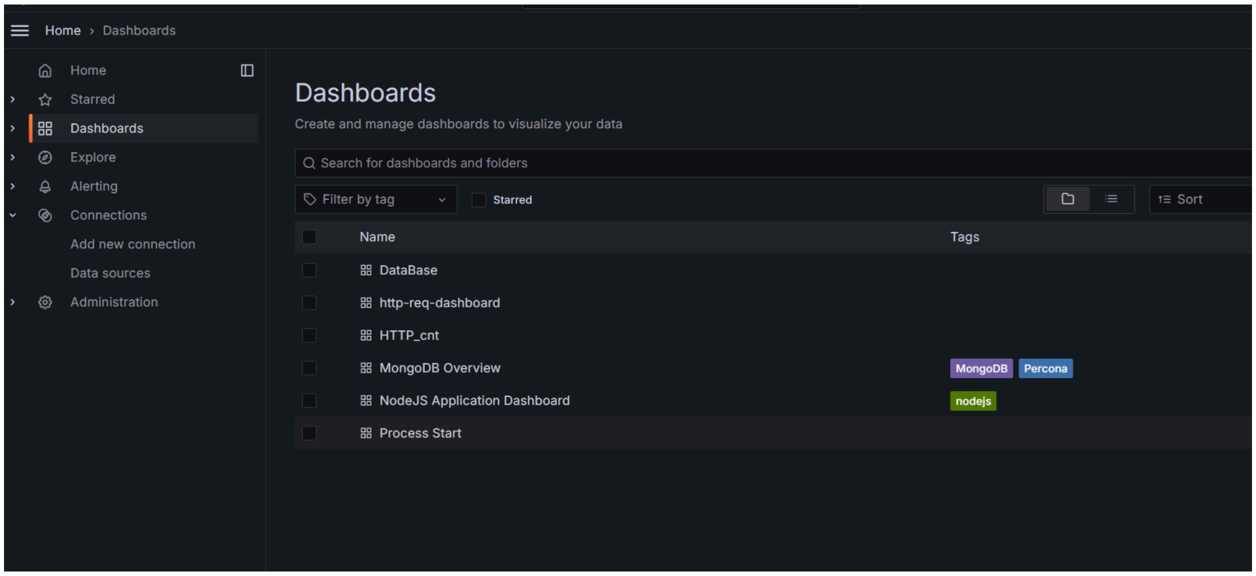
```
sudo systemctl restart prometheus
```

5. Add Prometheus as a Data Source in Grafana

- Open Grafana (running on VM 1).
- Go to **Configuration** > **Data Sources**.
- Add **Prometheus** as a new data source.
- Set the URL to your Prometheus server (http://<your-prometheus-ip>:9090).

Reference Link:
https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-grafana-on-ubuntu-20-04

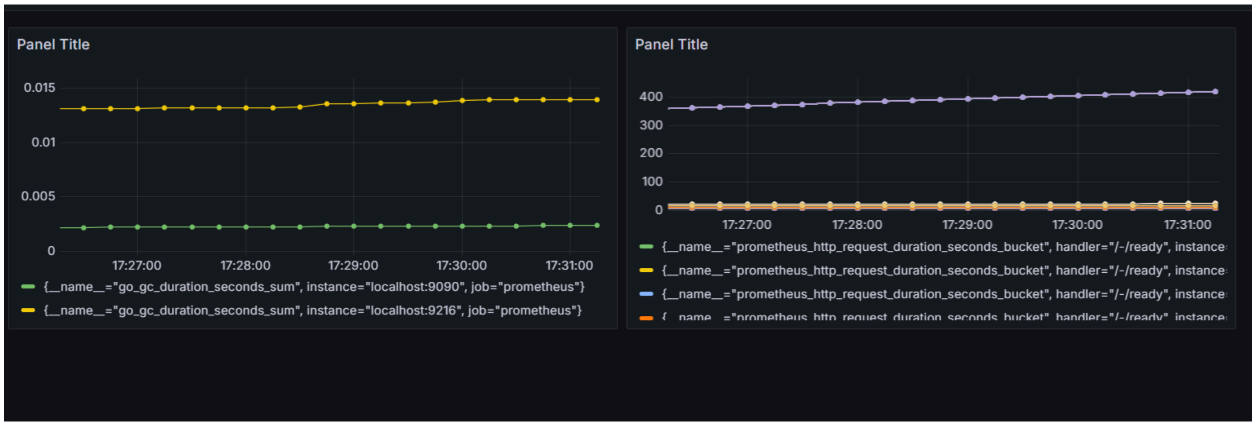
https://www.tothenew.com/blog/step-by-step-setup-grafana-and-prometheus-monitoring-using-node-exporter/

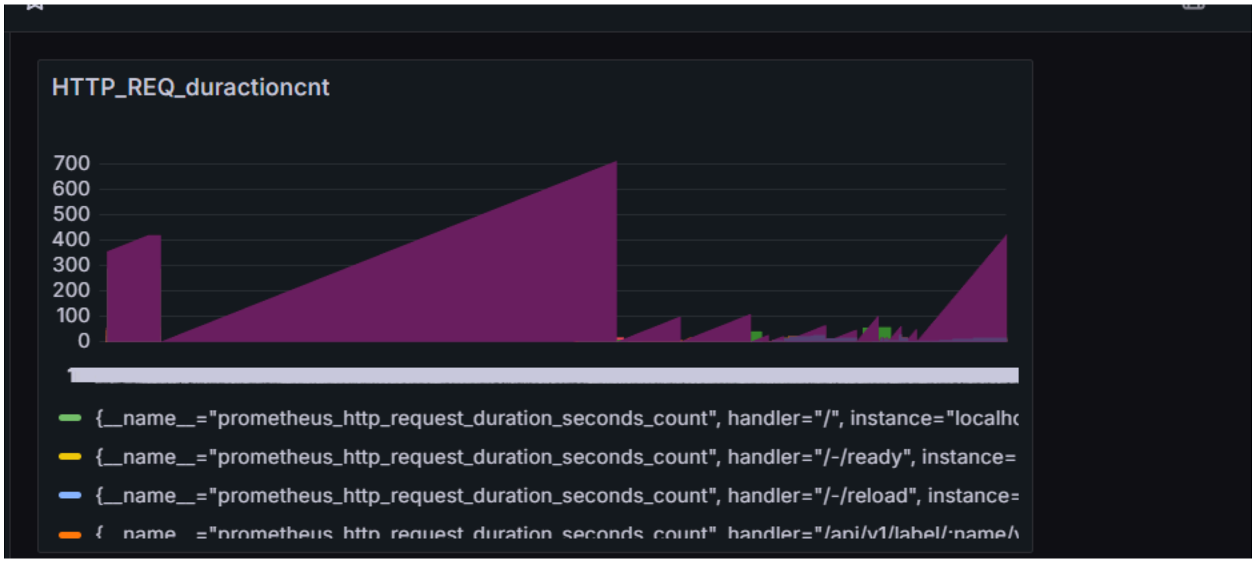


Dashboards

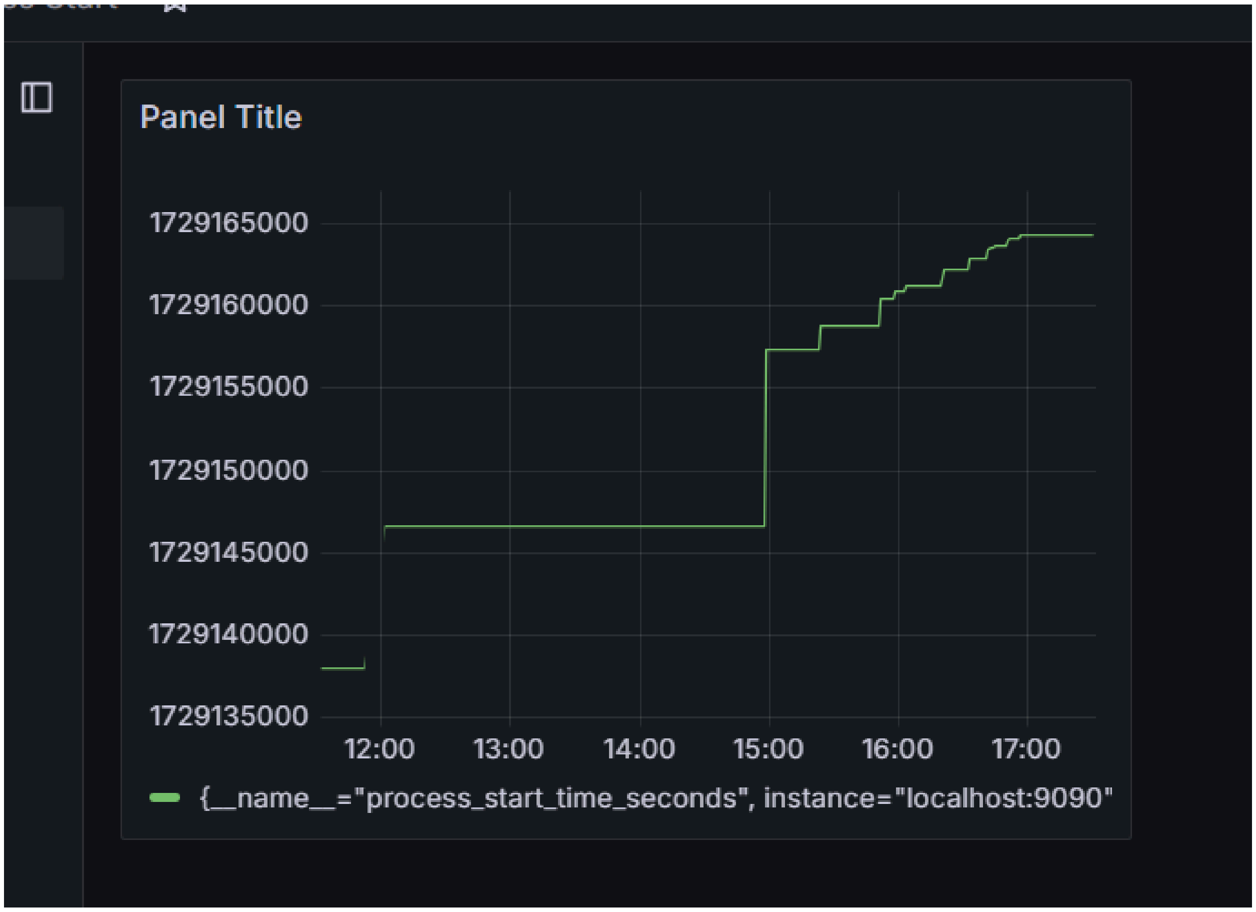
Create and manage dashboards to visualize your data

Name	Tags
DataBase	
http-req-dashboard	
HTTP_cnt	
MongoDB Overview	MongoDB Percona
NodeJS Application Dashboard	nodejs
Process Start	

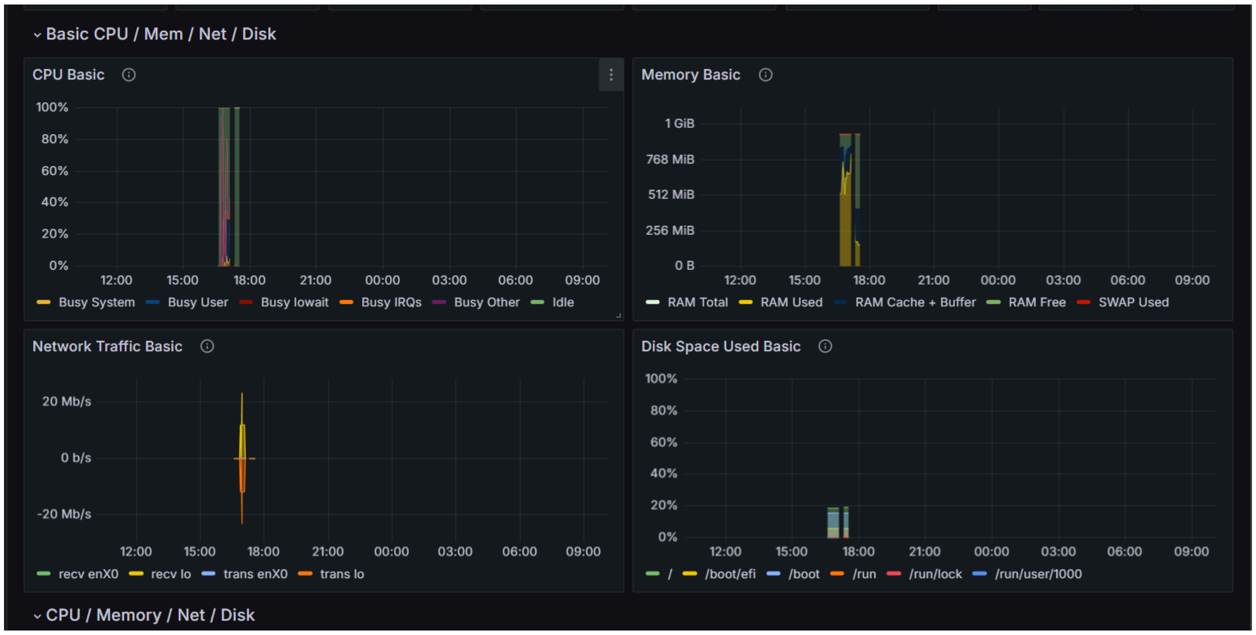


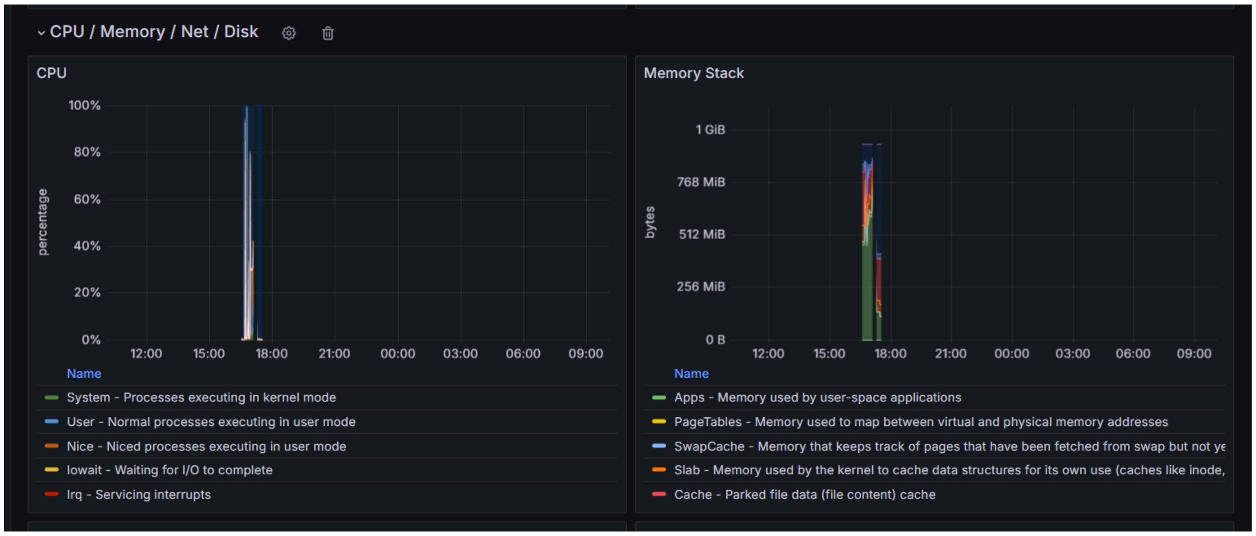


{__name__="prometheus_http_request_duration_seconds_count", handler="/", instance="localhc
{__name__="prometheus_http_request_duration_seconds_count", handler="/-/ready", instance=
{__name__="prometheus_http_request_duration_seconds_count", handler="/-/reload", instance=
{__name__="prometheus_http_request_duration_seconds_count", handler="/api/v1/label/:name/v



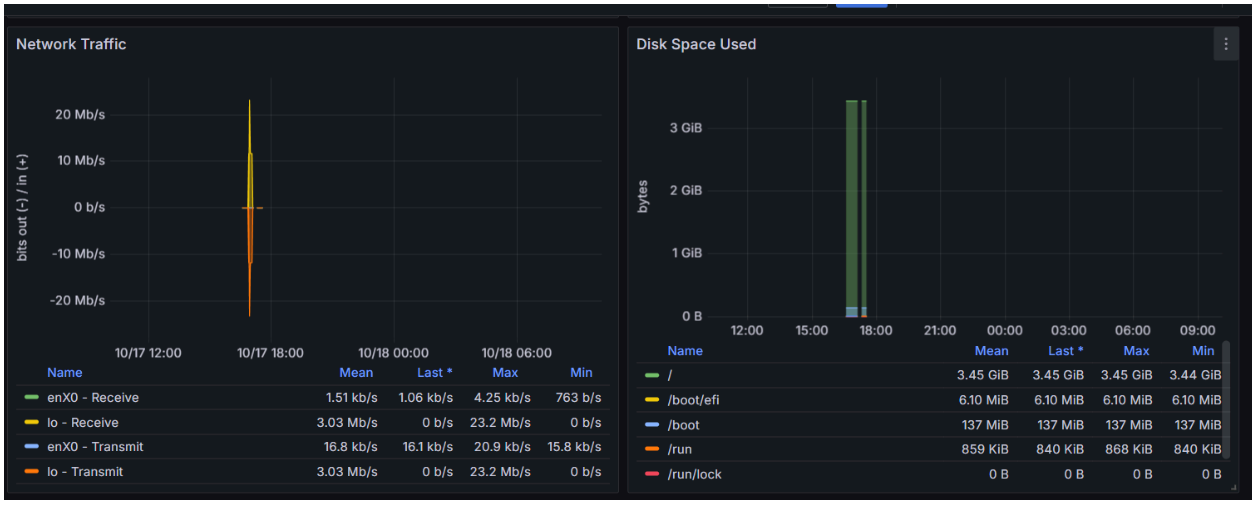
{__name__="process_start_time_seconds", instance="localhost:9090"





Name
- System - Processes executing in kernel mode
- User - Normal processes executing in user mode
- Nice - Niced processes executing in user mode
- Iowait - Waiting for I/O to complete
- Irq - Servicing interrupts

Name
- Apps - Memory used by user-space applications
- PageTables - Memory used to map between virtual and physical memory addresses
- SwapCache - Memory that keeps track of pages that have been fetched from swap but not ye
- Slab - Memory used by the kernel to cache data structures for its own use (caches like inode,
- Cache - Parked file data (file content) cache

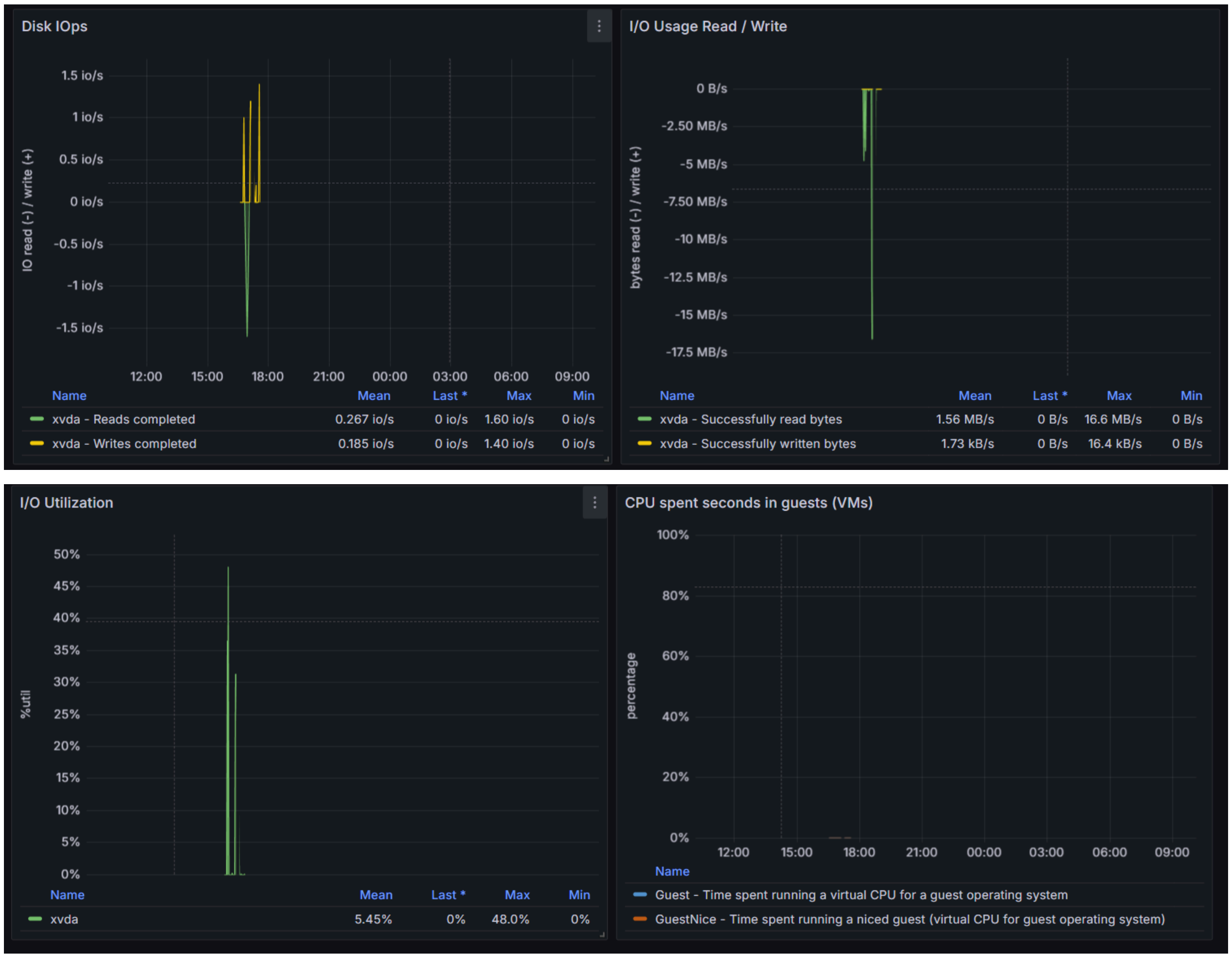


Network Traffic

	Name	Mean	Last *	Max	Min
🟩	enX0 - Receive	1.51 kb/s	1.06 kb/s	4.25 kb/s	763 b/s
🟨	lo - Receive	3.03 Mb/s	0 b/s	23.2 Mb/s	0 b/s
🟦	enX0 - Transmit	16.8 kb/s	16.1 kb/s	20.9 kb/s	15.8 kb/s
🟧	lo - Transmit	3.03 Mb/s	0 b/s	23.2 Mb/s	0 b/s

Disk Space Used

	Name	Mean	Last *	Max	Min
🟩	/	3.45 GiB	3.45 GiB	3.45 GiB	3.44 GiB
🟨	/boot/efi	6.10 MiB	6.10 MiB	6.10 MiB	6.10 MiB
🟦	/boot	137 MiB	137 MiB	137 MiB	137 MiB
🟧	/run	859 KiB	840 KiB	868 KiB	840 KiB
🟥	/run/lock	0 B	0 B	0 B	0 B



Disk IOps

	Name	Mean	Last *	Max	Min
🟩	xvda - Reads completed	0.267 io/s	0 io/s	1.60 io/s	0 io/s
🟨	xvda - Writes completed	0.185 io/s	0 io/s	1.40 io/s	0 io/s

I/O Usage Read / Write

	Name	Mean	Last *	Max	Min
🟩	xvda - Successfully read bytes	1.56 MB/s	0 B/s	16.6 MB/s	0 B/s
🟨	xvda - Successfully written bytes	1.73 kB/s	0 B/s	16.4 kB/s	0 B/s

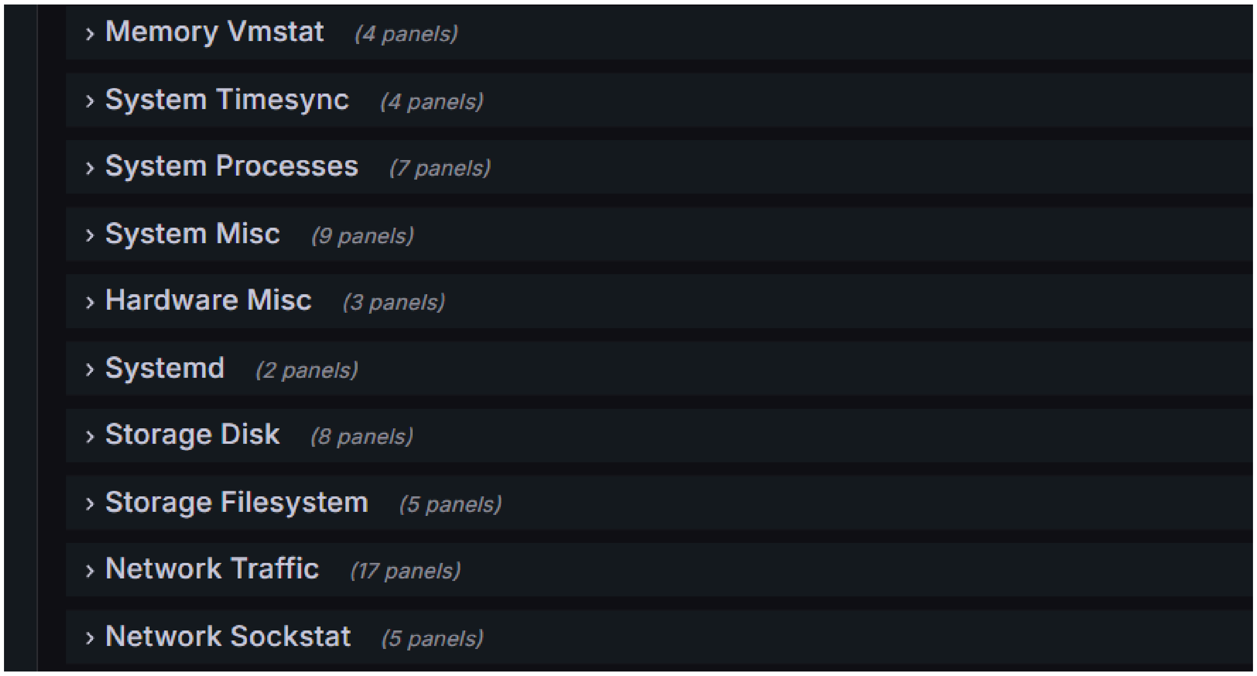
I/O Utilization

	Name	Mean	Last *	Max	Min
🟩	xvda	5.45%	0%	48.0%	0%

CPU spent seconds in guests (VMs)

	Name
🟦	Guest - Time spent running a virtual CPU for a guest operating system
🟧	GuestNice - Time spent running a niced guest (virtual CPU for guest operating system)

This all are the List of Dashboard I have designed.



› Memory Vmstat (4 panels)

› System Timesync (4 panels)

› System Processes (7 panels)

› System Misc (9 panels)

› Hardware Misc (3 panels)

› Systemd (2 panels)

› Storage Disk (8 panels)

› Storage Filesystem (5 panels)

› Network Traffic (17 panels)

› Network Sockstat (5 panels)