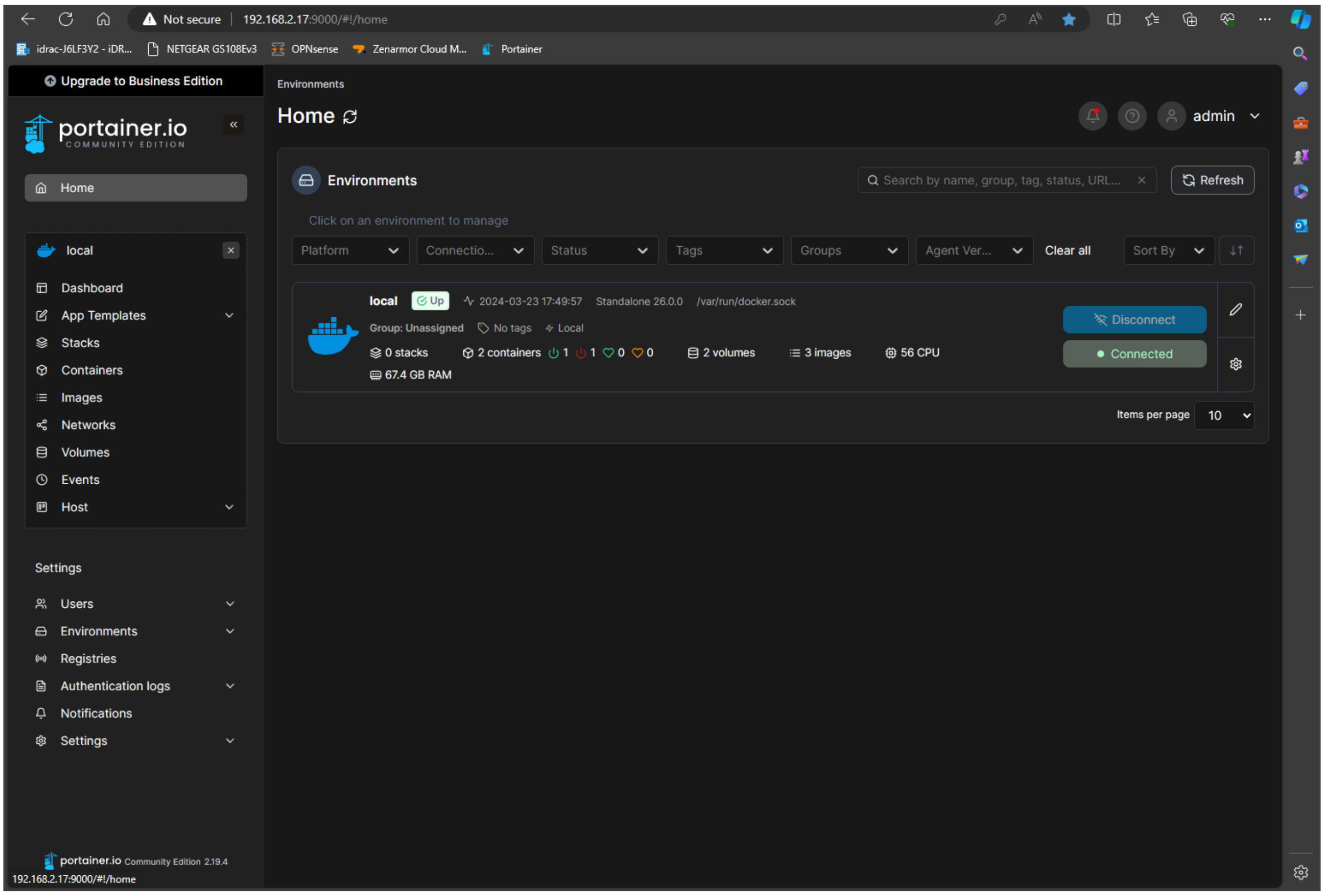
Docker Containers

Portainer

Web GUI to manage Docker containers being run on a Docker Engine hosted on an Ubuntu server

Portainer

GUI Preview



Once your docker container is successfully up and running, you can access your Portainer web GUI on port 9000 of your Ubuntu Server's static IP. The homepage above shows I have my docker environment currently running and connected, which means I can deploy containers and configure them however I'd like. Your next steps should be to connect your docker profile so that you can push images.

Deployment

To deploy the Portainer container, you'll need to first set up a couple of configurations. These configurations will create a persistent volume, expose ports, and enable the container to run at start up automatically.

Create Persistent Volume

In your docker engine CLI, run the following command:

-
  ```
  docker volume create portainer_data
  ```

    - Volume name is portainer_data

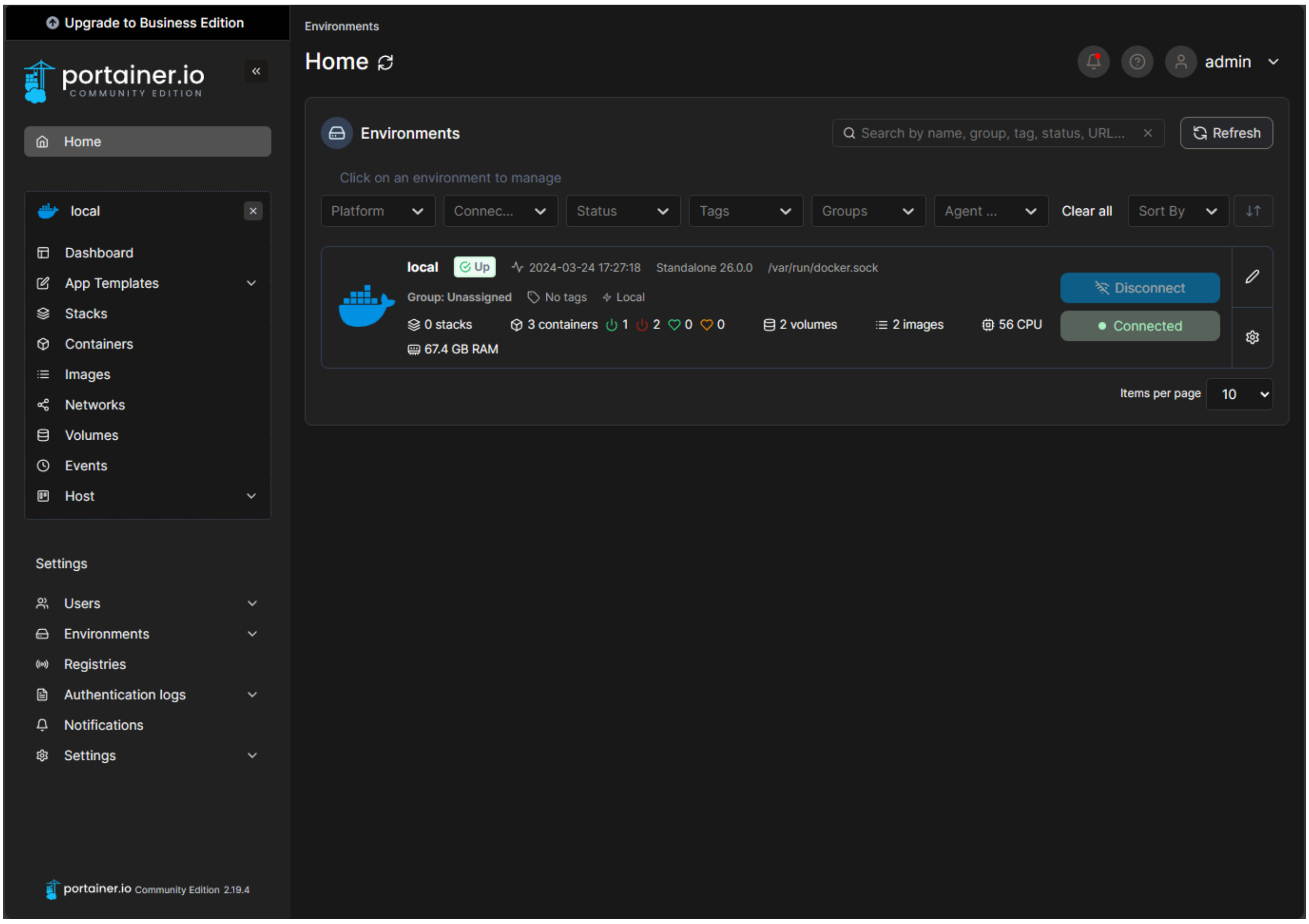
Run the Container

Run the container as you would any other container, with the following ports exposed: 8000 (Portainer Agent), 9000(Web GUI)

-
  ```
  docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always
  /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce
  ```

    - ***-d*** to run in background
    - ***-p*** to expose ports
    - ***--name*** to name container
    - ***--restart*** to make it run constantly
    - ***/var/run/docker.sock:/var/run/docker.sock*** to enable portainer to manage the docker registry of the local machine
    - ***-v*** declares a volume to use
    - ***portainer/portainer-ce*** refers to the community edition version of the container

You're all set! Run through the set-up wizard via the web GUI that can be accessed at your docker engine IP port 9000 and create your admin account. You can now view the following dashboard:



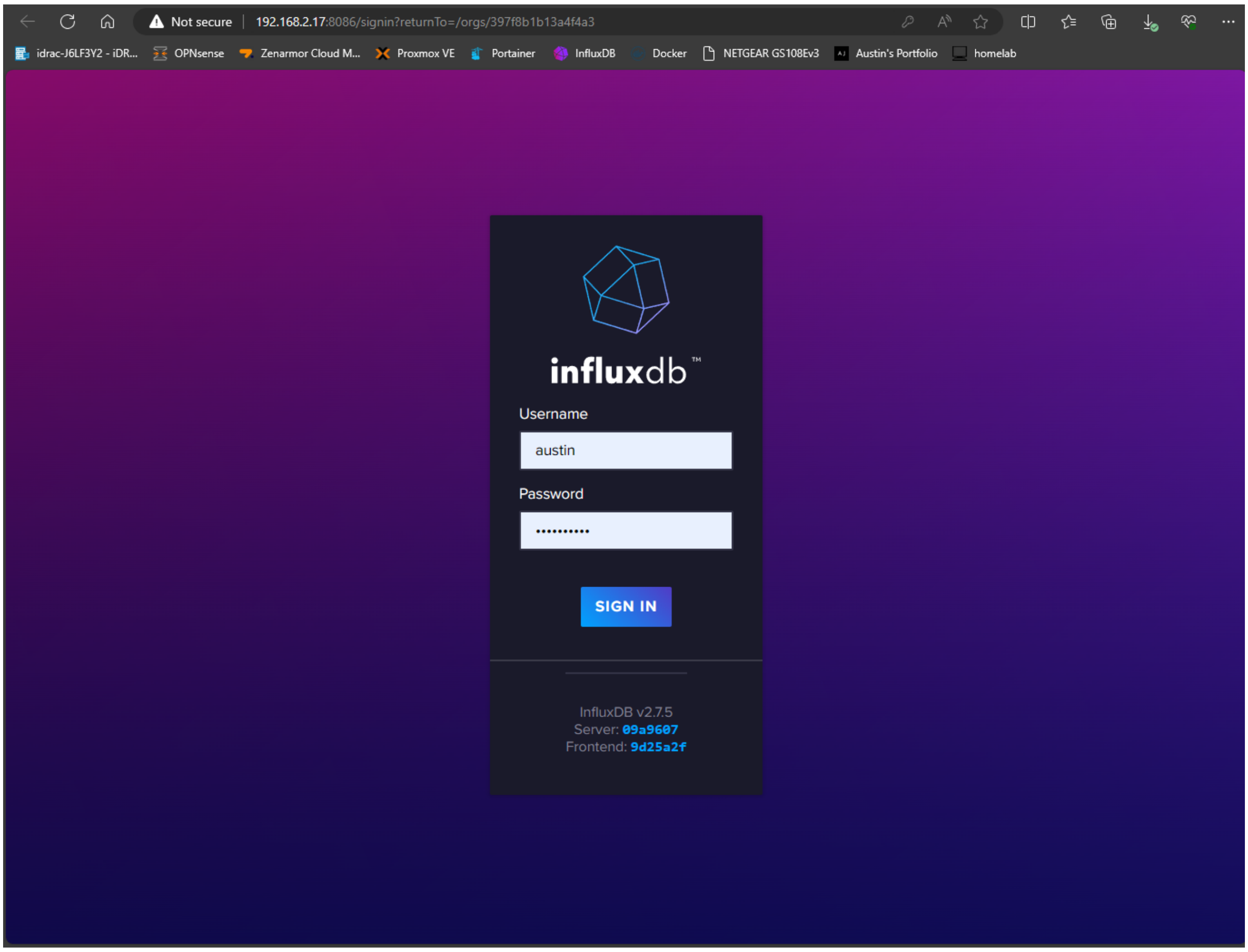
local Up 2024-03-24 17:27:18 Standalone 26.0.0 /var/run/docker.sock
Group: Unassigned No tags Local
0 stacks 3 containers 1 2 0 0 2 volumes 2 images 56 CPU
67.4 GB RAM

InfluxDB

Database to store data from OPNsense and Proxmox. Will be used to populate Grafana Dashboards

GUI Preview

Once your docker container is configured and running, you should be able to access the Web GUI:

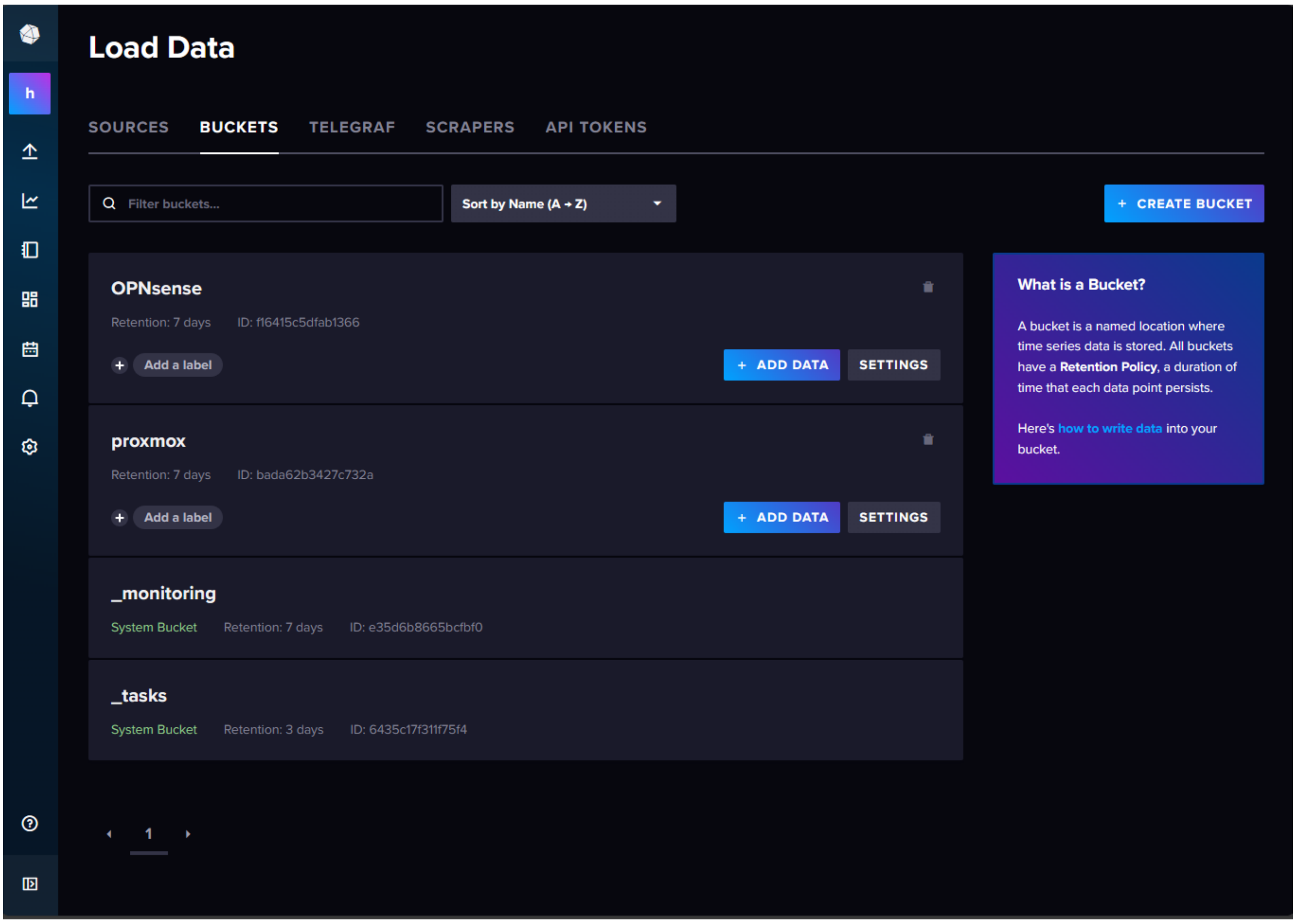


Username: austin

Password: ••••••••••

InfluxDB v2.7.5
Server: 09a9607
Frontend: 9d25a2f

Create buckets to store and query data via the Data Explorer tab:



Load Data

OPNsense
Retention: 7 days ID: f16415c5dfab1366

proxmox
Retention: 7 days ID: bada62b3427c732a

_monitoring
System Bucket Retention: 7 days ID: e35d6b8665bcfbf0

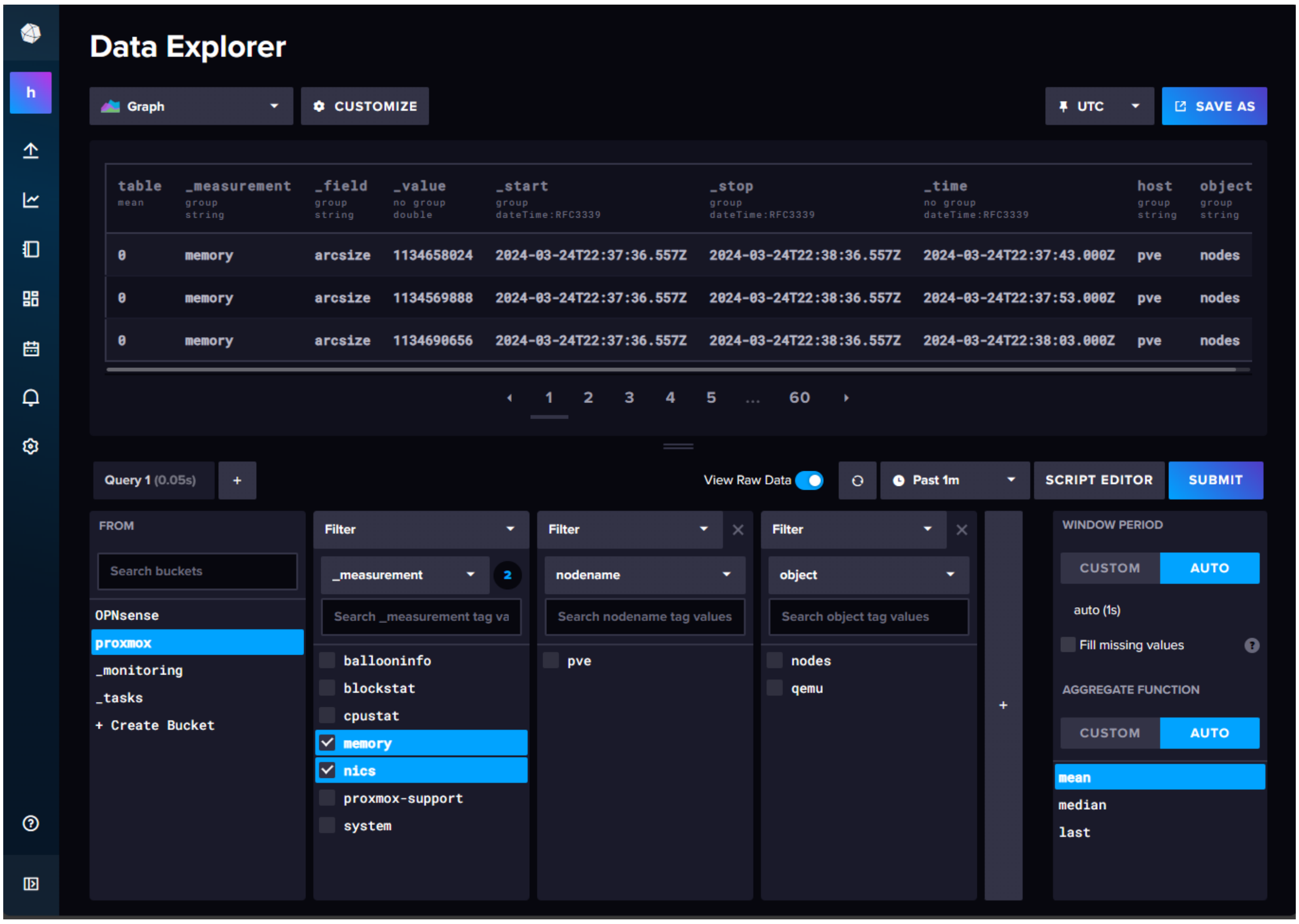
_tasks
System Bucket Retention: 3 days ID: 6435c17f311f75f4

What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

Here's how to write data into your bucket.

Once you create connections from buckets to your systems using the API tokens, select the bucket. You should be able to view the raw data being collected. After setting up my Proxmox bucket to receive data from my Proxmox VE to the influxDB, you can see it working below:



Data Explorer

Graph | CUSTOMIZE | UTC | SAVE AS

table mean	_measurement group string	_field group string	_value no group double	_start group dateTime:RFC3339	_stop group dateTime:RFC3339	_time no group dateTime:RFC3339	host group string	object group string
0	memory	arcsize	1134658024	2024-03-24T22:37:36.557Z	2024-03-24T22:38:36.557Z	2024-03-24T22:37:43.000Z	pve	nodes
0	memory	arcsize	1134569888	2024-03-24T22:37:36.557Z	2024-03-24T22:38:36.557Z	2024-03-24T22:37:53.000Z	pve	nodes
0	memory	arcsize	1134690656	2024-03-24T22:37:36.557Z	2024-03-24T22:38:36.557Z	2024-03-24T22:38:03.000Z	pve	nodes

1 2 3 4 5 ... 60

Query 1 (0.05s) | + | View Raw Data | Past 1m | SCRIPT EDITOR | SUBMIT

FROM
Search buckets
OPNsense
proxmox
_monitoring
_tasks
+ Create Bucket

Filter
_measurement 2
Search _measurement tag va
balllooninfo
blockstat
cpustat
✓ memory
✓ nics
proxmox-support
system

Filter
nodename
Search nodename tag values
pve

Filter
object
Search object tag values
nodes
qemu

WINDOW PERIOD
CUSTOM [✓ AUTO]
auto (1s)
Fill missing values

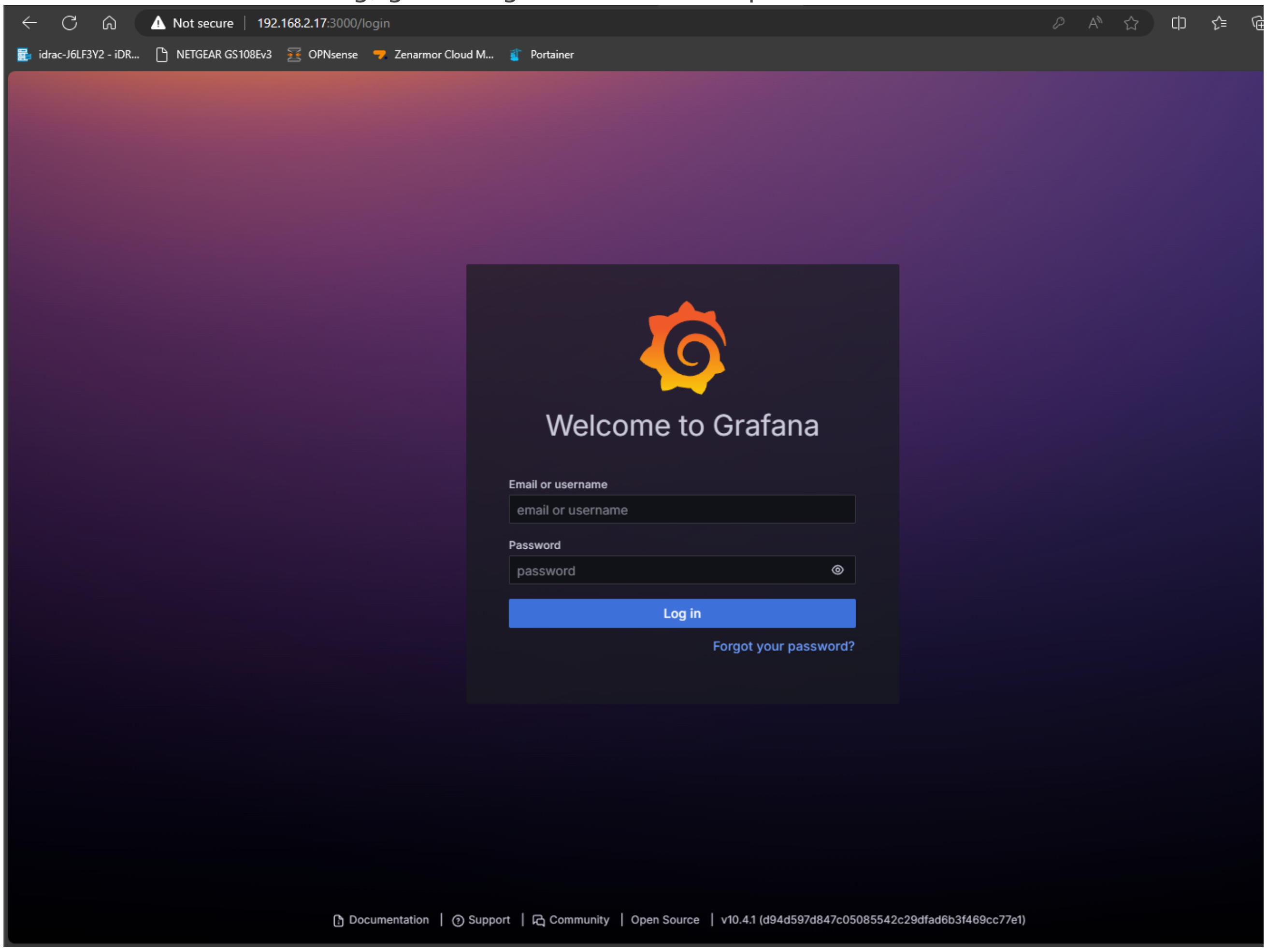
AGGREGATE FUNCTION
CUSTOM [✓ AUTO]
mean
median
last

Grafana

Used to build dashboards!

GUI Preview

Download and run the Grafana container. You can get more details about how to do so @ DockerHub - Grafana. Or, you can just run the following command:

-
  ```
  docker run -d --name=grafana -p 3000:3000 grafana/grafana
  ```
  - 3000:3000 specifies it'll be accesible from port 3000 of the docker engine, which in my case is 192.168.2.17.
- Once the container is running, go through the initial set up wizard via the Web GUI:
  - 

Not secure | 192.168.2.17:3000/login

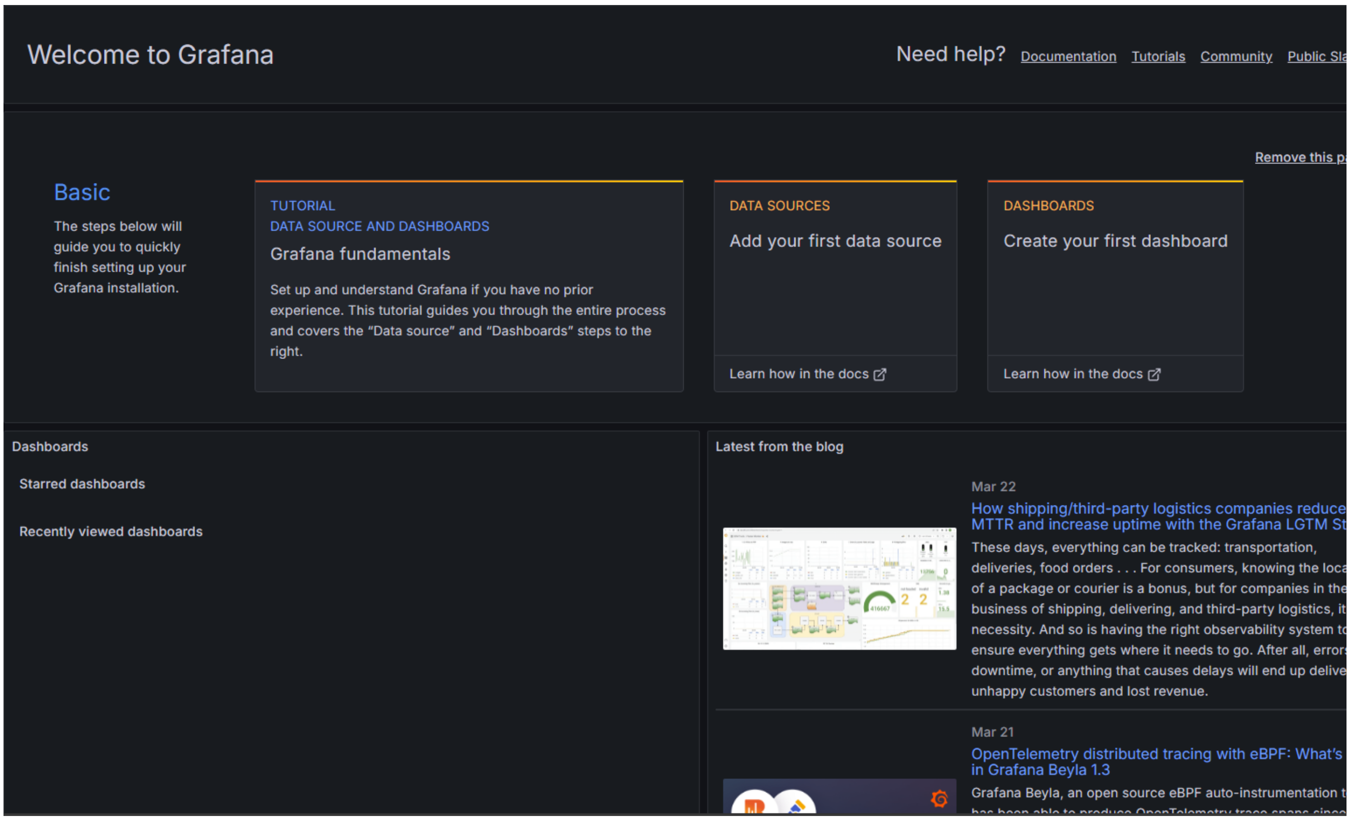
Welcome to Grafana

Email or username: email or username

Password: password

Log in
Forgot your password?
Documentation | Support | Community | Open Source | v10.4.1 (d94d597d847c05085542c29dfad6b3f469cc77e1)

With your Grafana container running, it can be accessed via port 3000 of your localhost. The default login username and password is **admin**. Once logged in, you should see a home page like the one below:

- 

Welcome to Grafana

Need help? Documentation Tutorials Community Public Sla

Remove this p

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.

DATA SOURCES
Add your first data source

Learn how in the docs

DASHBOARDS
Create your first dashboard

Learn how in the docs

Dashboards

Starred dashboards

Recently viewed dashboards

Latest from the blog

Mar 22
How shipping/third-party logistics companies reduce MTTR and increase uptime with the Grafana LGTM St

These days, everything can be tracked: transportation, deliveries, food orders . . . For consumers, knowing the loca of a package or courier is a bonus, but for companies in the business of shipping, delivering, and third-party logistics, it necessity. And so is having the right observability system to ensure everything gets where it needs to go. After all, errors downtime, or anything that causes delays will end up delive unhappy customers and lost revenue.

Mar 21
OpenTelemetry distributed tracing with eBPF: What's in Grafana Beyla 1.3

Grafana Beyla, an open source eBPF auto-instrumentation t