REMS IoT installation instructions [April 2021]

19 April 2021 12:11

Created by: Dima Maneuski, dima.maneuski@glasgow.ac.uk
Version: 1.0.0
Date: 19 April 2021

Introduction

This instruction serves as a guide for installation of IoT infrastructure around REMS. For details about REMS refer to Barts REMS_KIT.pdf

This instruction broadly covers installation of Grafana, InfluxDB and several Raspberry PI hacks related to COM port communication and automatic startup of the python code on RPi boot.

I recommend to not install Grafana and InfluxDB on Raspberry Pi and use it as a passive unit for data acquisition, local memory backup and upload to central server. Instructions below for Grafana and InfluxDB will be for the central server, which could be a computer in the lab. It needs access to the local area network to which Raspberry PI is connected (either by means of WiFi or ethernet cable).

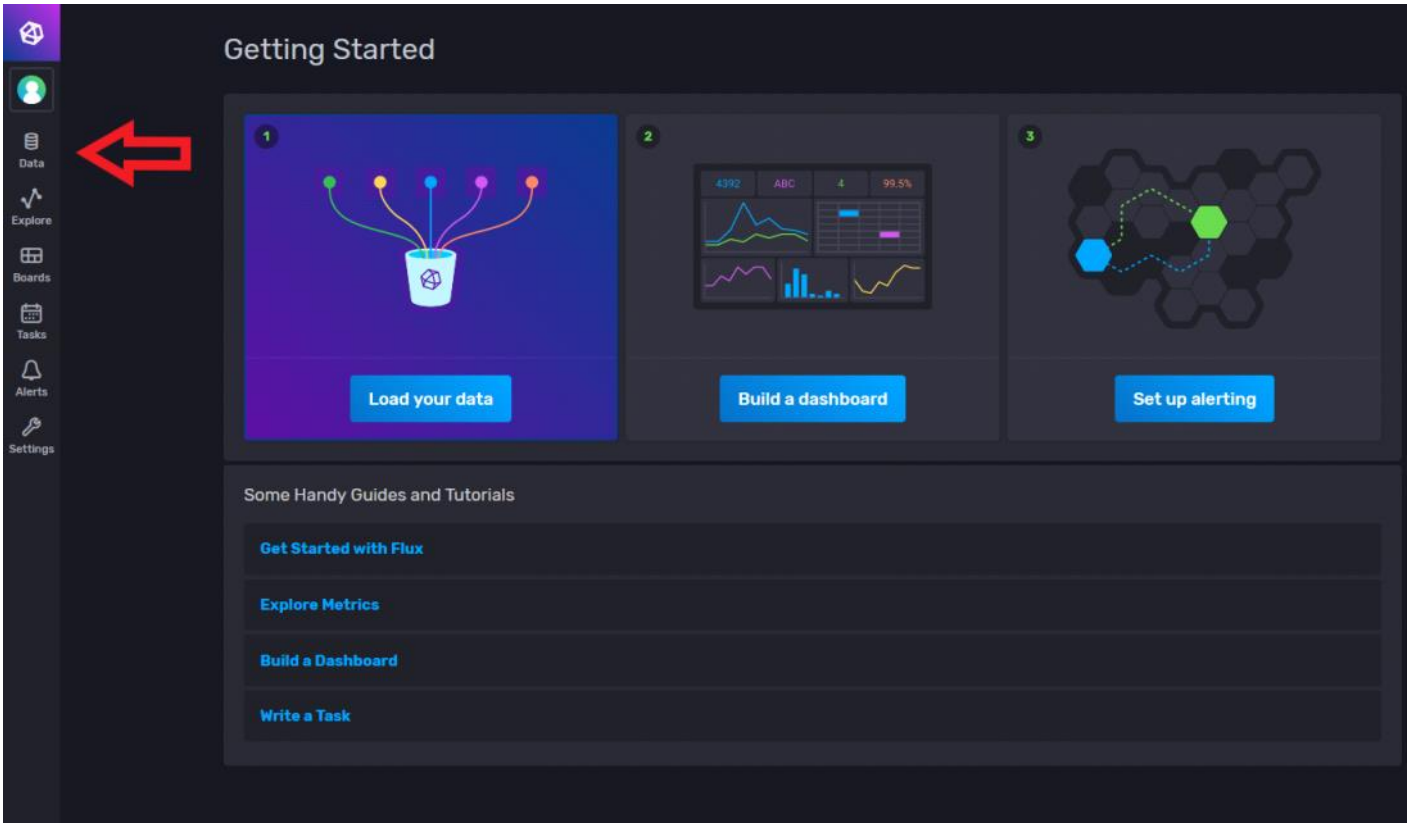
There are these broad steps in the installation and configuration:

- Install and configure InfluxDB
- Install and configure Grafana
- Configure python script on Raspberry Pi
- Configure Raspberry Pi OS

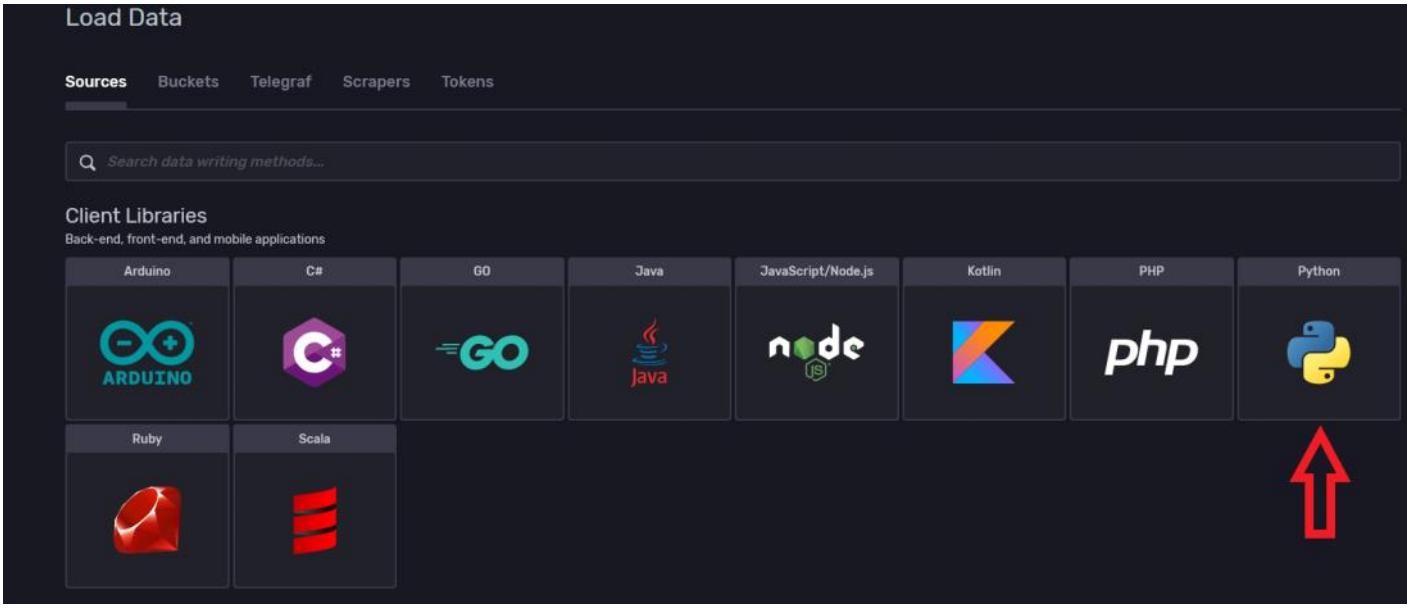
Install and configure InfluxDB
InfluxDB v.2.X is recommended for all new installs. This version is drastically different from v.1.X in all terms and two versions are not really compatible, learning v. 2.x from v. 1.x is like learning a completely new DB technology.
InfluxDB steps include installation of influxDB on your server, setting up Organization, setting up Bucket and getting API token for python scripts. Call your Bucket, REMS

Link here will guide you through InfluxDB installation and configuration. After completing steps there, you should be able to login to the InfluxDB web interface through your server IP address and port 8086, say https://192.168.1.55:8086

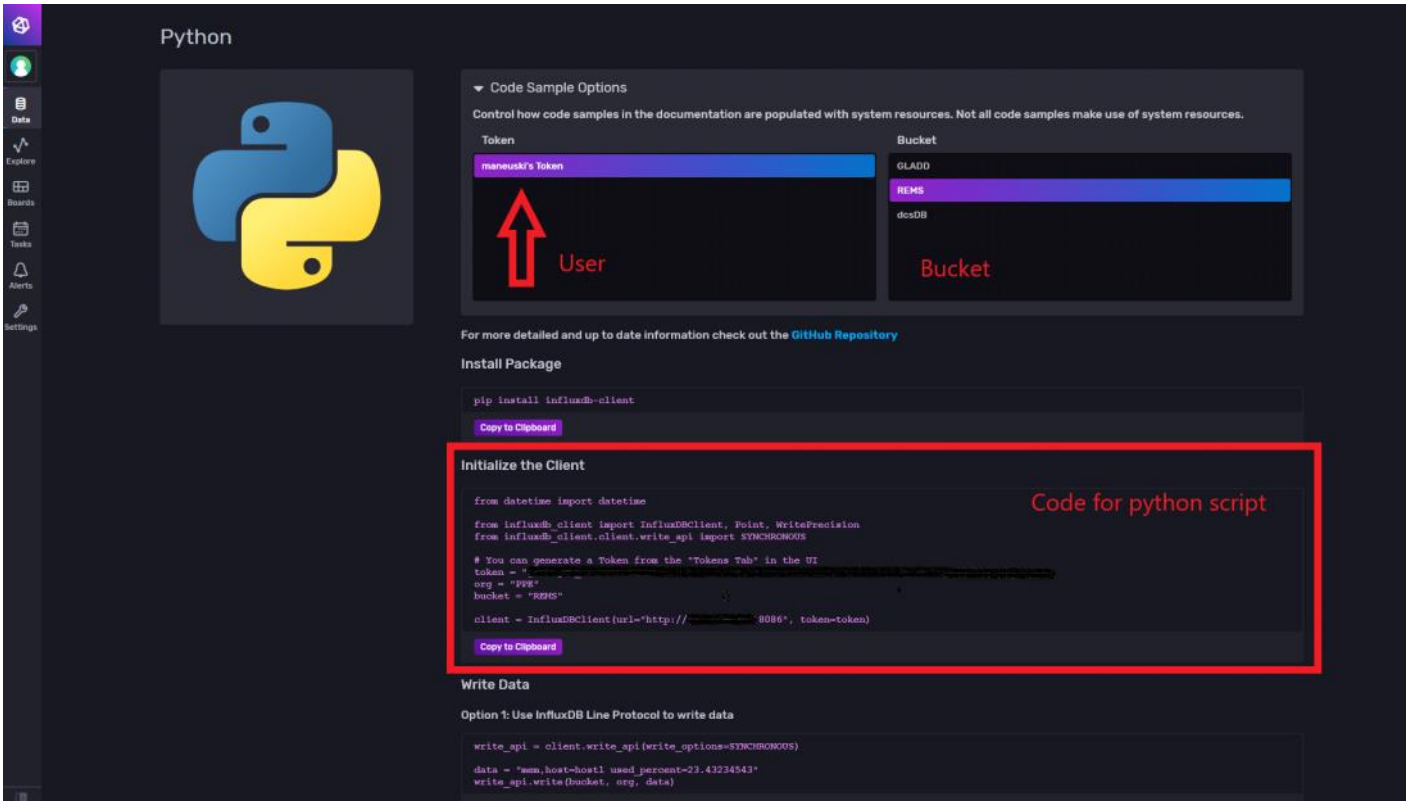
Go to left menu and select Data



Click on Python

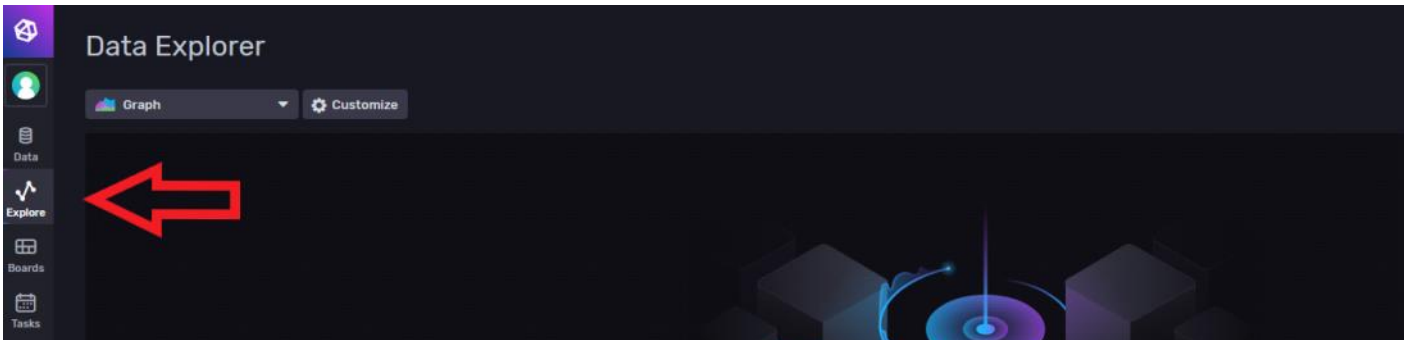


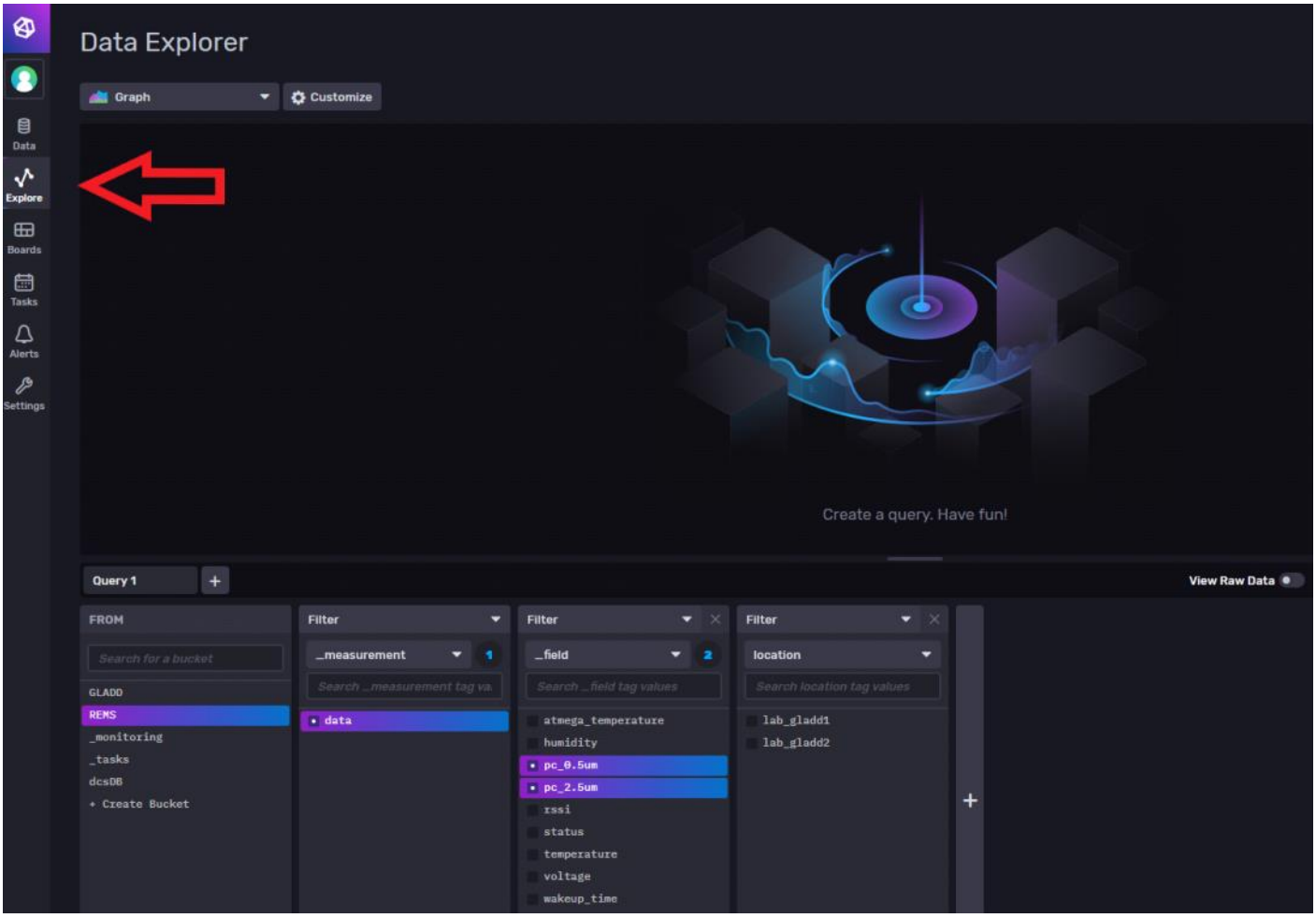
There you'll see user name, bucket and, in the Initialise the client area, token. You'll need this section for the python scripts. The page also contains code snippets for writing data into InfluxDB.



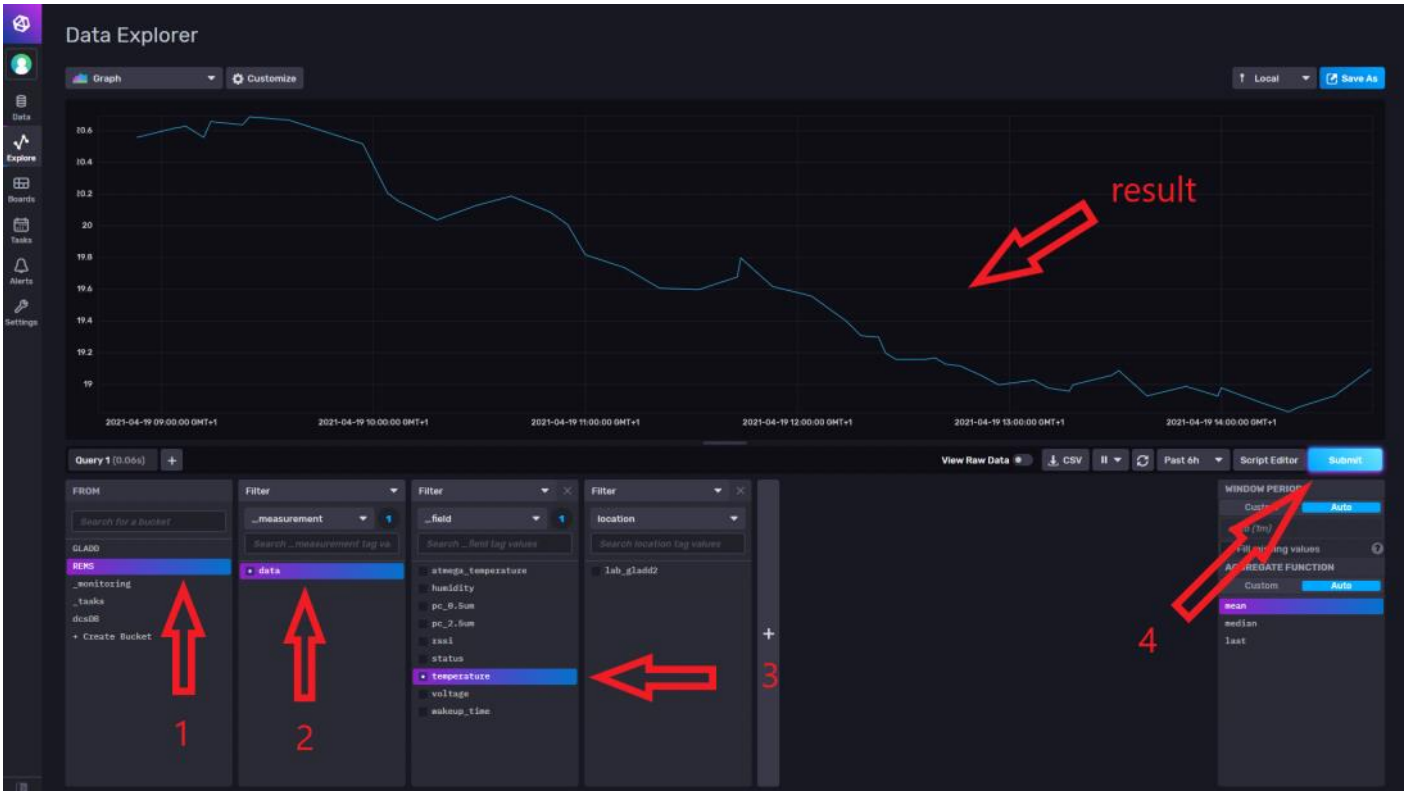
This concludes setting and configuration of InfluxDB for data storage. Having organisation, bucket and token at hand allows to write data to InfluxDB.

Once there is data in the database, it can be quickly visualised inside InfluxDB web GUI. This sometimes helps to debug problems. To visualise data in InfluxDB web GUI, go to Explore in the left hand side menu



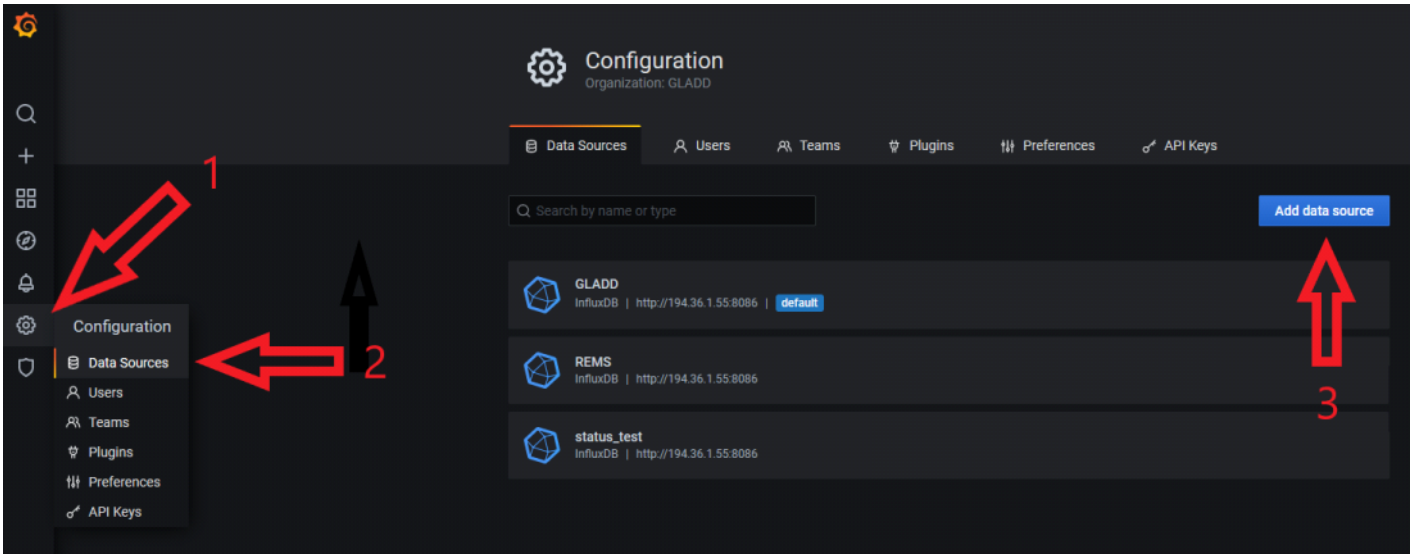


At the bottom of the page select bucket REMS, apply a filter (tag in database data point) and press Submit. You should see a graph on the screen above. An example from REMS device is shown below.

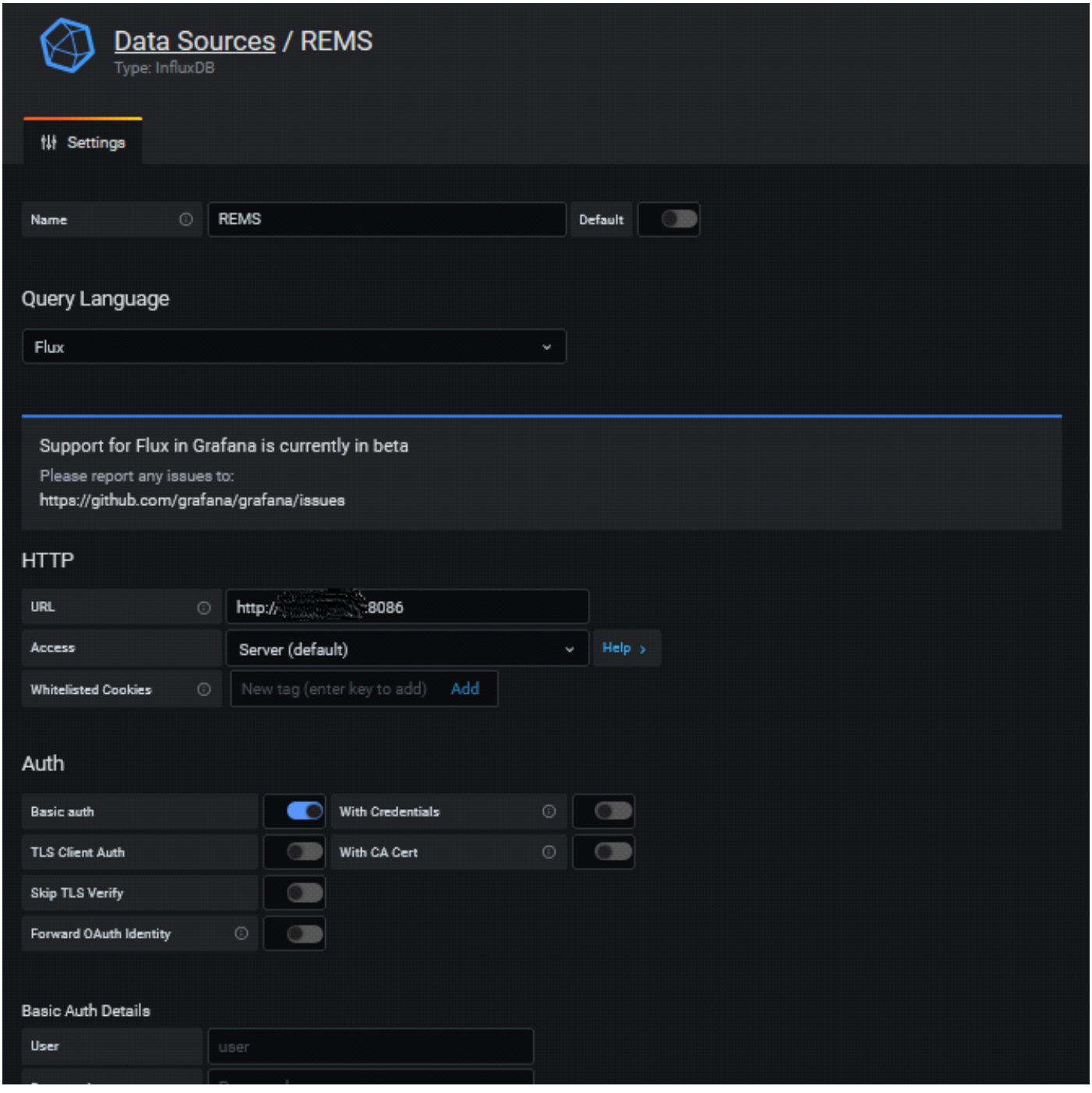


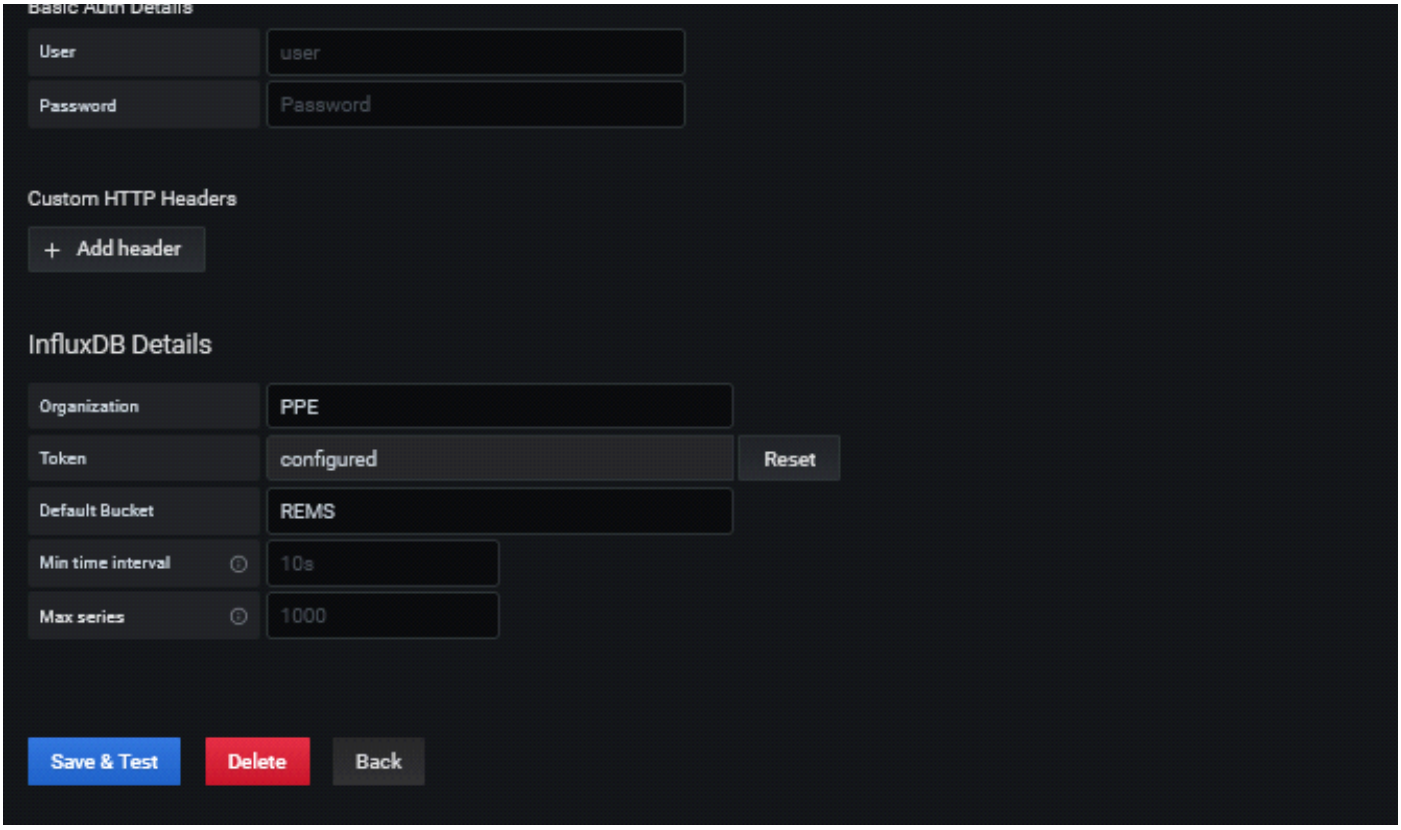
Install and configure Grafana
Good installation guide is provided here: https://grafana.com/docs/grafana/latest/installation/

Now we need to add data source to Grafana, in this case it will be REMS data from InfluxDB.



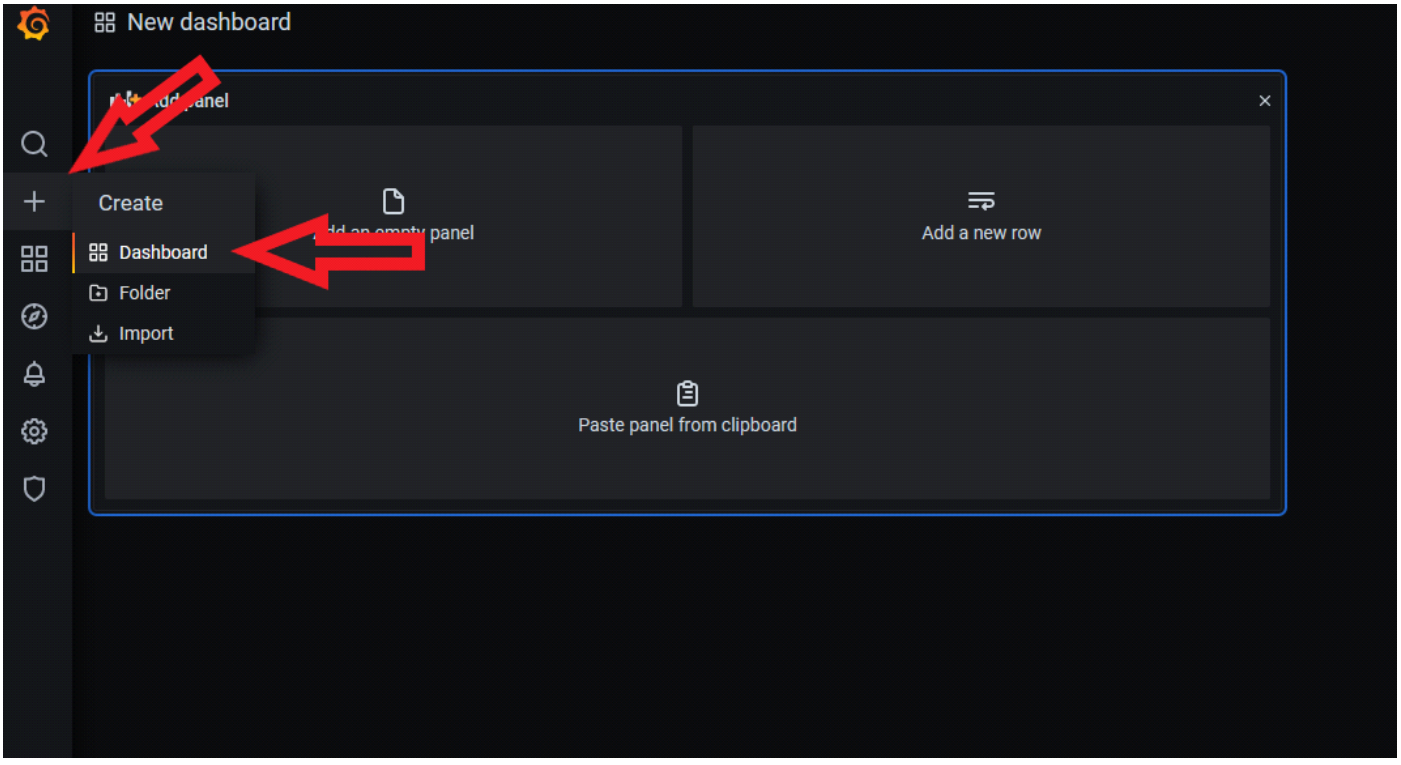
Select InfluxDB in the list.
Change settings and input data as shown in the screenshot below



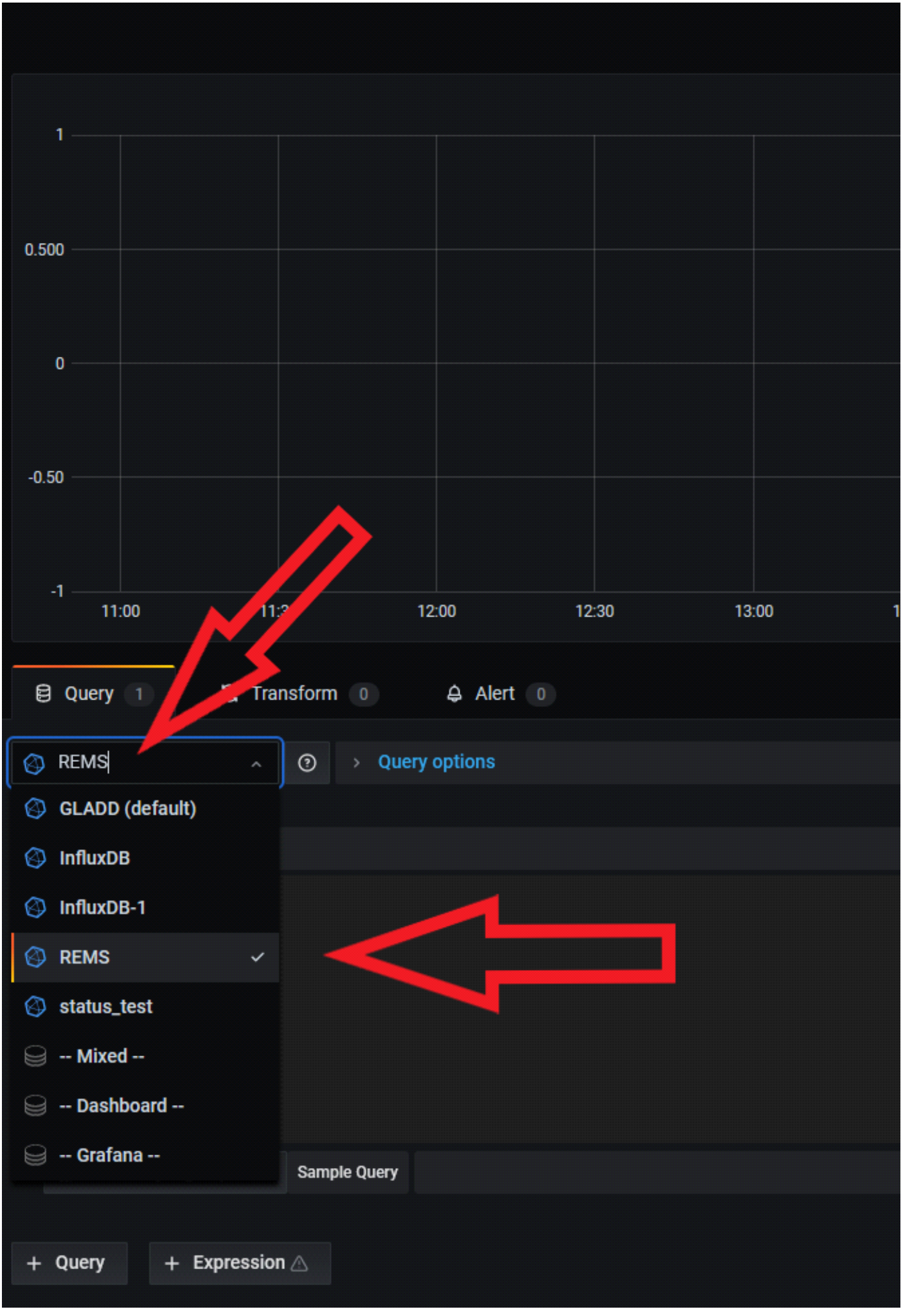


Press save and test. If Grafana managed to connect and query InfluxDB, it will flash green. Now we are ready to visualise data from REMS database.

Add new dashboard



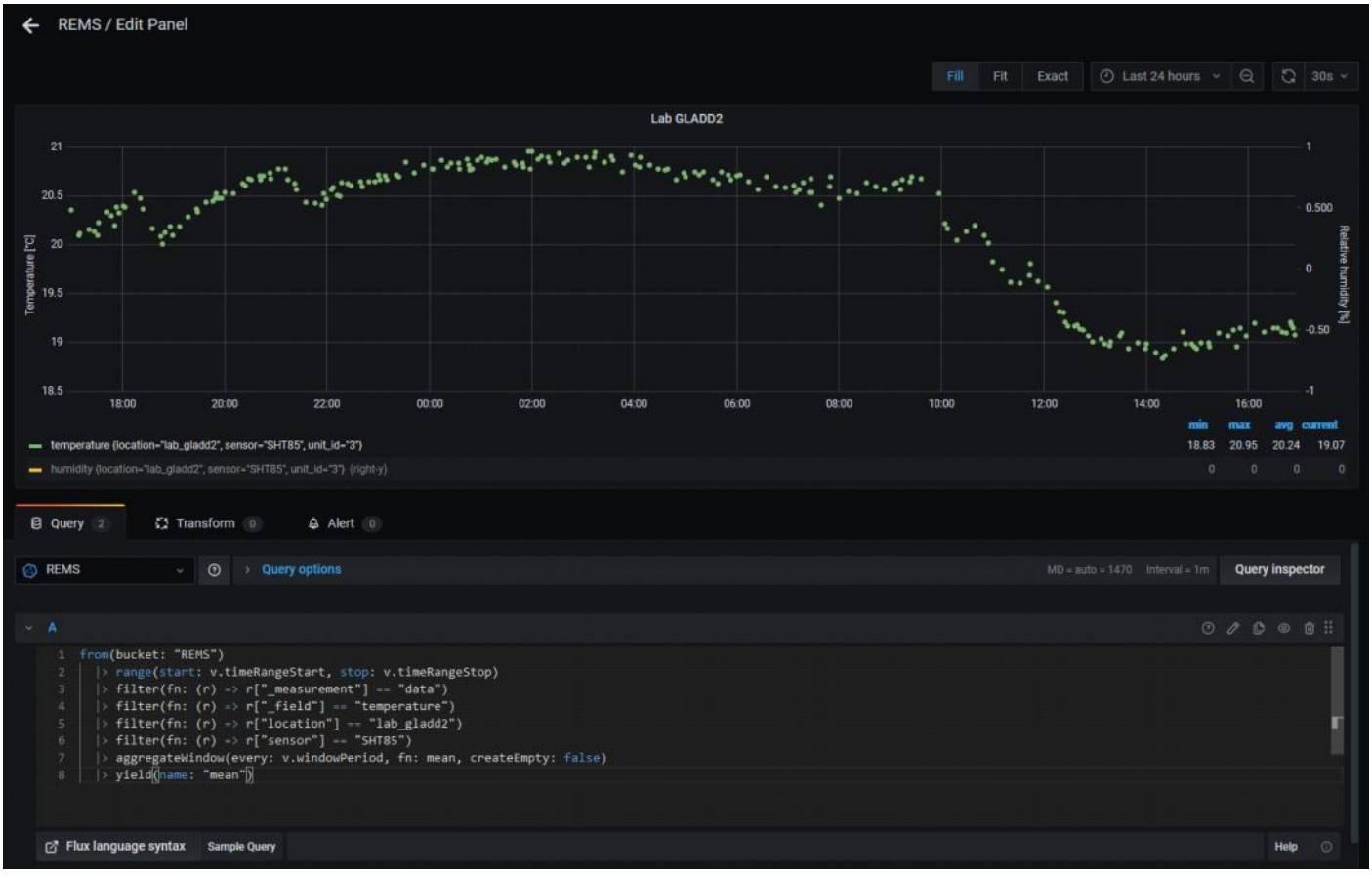
Click on add empty panel.
Choose REMS from the list as below.


1
0.500
0
-0.50
-1
11:00
12:00
12:30
13:00
Query 1
Transform 0
Alert 0
REMS
Query options
GLADD (default)
InfluxDB
InfluxDB-1
REMS
status_test
-- Mixed --
-- Dashboard --
-- Grafana --
Sample Query
+ Query
+ Expression

In the query filed under A enter flux query expression. For example:

```
from(bucket: "REMS")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "data")
  |> filter(fn: (r) => r["_field"] == "temperature")
  |> filter(fn: (r) => r["location"] == "lab_gladd2")
  |> filter(fn: (r) => r["sensor"] == "SHT85")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

This should show screen similar to the one below:



How to make Raspberry PI never complain about USB device permissions
Create empty file called: usb.rules
Add this to the file:

```
SUBSYSTEM=="ttyUSB", MODE="0666"
SUBSYSTEM=="usb", MODE="0666"
SUBSYSTEM=="ttyACM", MODE="0666"
```

Copy the file as sudo to /etc/udev/rules.d/

```
sudo cp usb.rules /etc/udev/rules.d/
```

Reboot

How run a python script on Raspberry PI at boot

```
sudo systemctl --full --force edit rems.service
```

Add this to the file:

```
[Unit]
Description=Dylos particle counter logger
After=multi-user.target
[Service]
```

```
[Unit]
Description=Dylos particle counter logger
After=multi-user.target
[Service]
Type=idle
User=pi
ExecStart=/usr/bin/python3 /home/pi/Documents/eclipse/pydev/gladd-iot/dylos/rems_gladd_iot_service.py &
[Install]
WantedBy=multi-user.target
```

Save the file. It will be saved here: /etc/systemd/system/rems.service
Now enable service:

```
sudo systemctl enable dylos.service
```

Add exec permissions to rems_gladd_iot_service.py file:

```
chmod +x rems_gladd_iot_service.py
```

Some useful commands:

```
systemctl status rems.service
systemctl start rems.service
systemctl stop rems.service
systemctl restart rems.service
```

Example rems_gladd_iot_service file can be found here