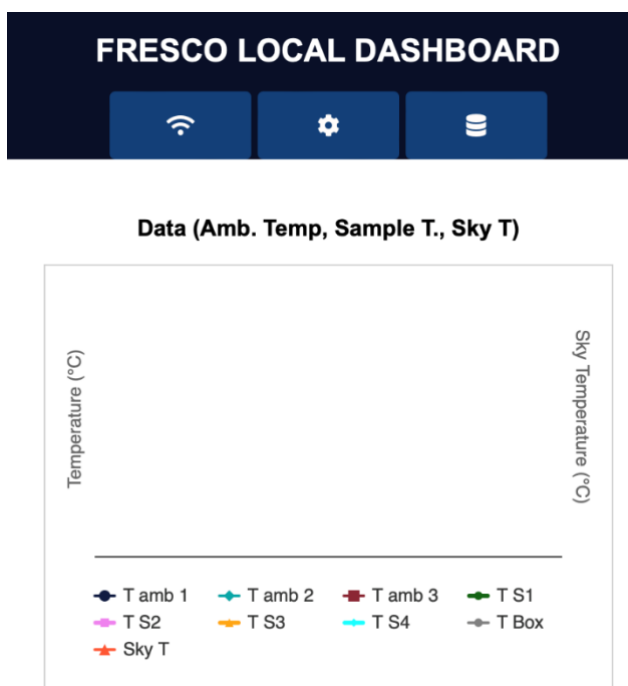


# FRESCO Board startup

## Software

### Index

<b>Integrated Web-App.....</b>	<b>1</b>
<b>WiFi manager (📶) and connection to external network .....</b>	<b>1</b>
<b>Usage of the Database (🗄️) .....</b>	<b>2</b>
<b>Settings (⚙️) .....</b>	<b>3</b>
<b>Cloud InfluxDb and Grafana configuration .....</b>	<b>5</b>
<b>Local InfluxDb and Grafana configuration .....</b>	<b>11</b>



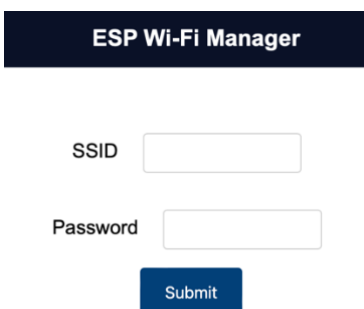
### Integrated Web-App

After power supplying the FRESCO board, using a smartphone or a laptop with a WiFi connection find the WiFi network with the SSID named “FRESCO-board” and use the password “123456789”. Once the device is connected navigate using your browser to the “local IP” displayed on the oled screen, commonly it is *192.168.4.1*. You will directly be redirected to the web app page and the following screenshot will be displayed.

The web app will directly display the collected data in the two double Y axis plots. Clicking on the labels you can choose to hide or not the selected measurements.

Using the three buttons placed at the top is possible to manage the WiFi connection (📶) over

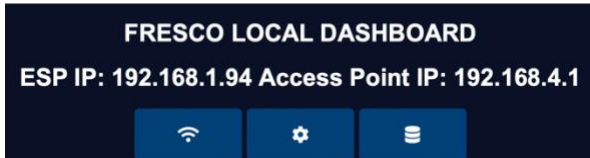
an external network, the ESP settings (⚙️) and to manage the database connection (🗄️).



### WiFi manager (📶) and connection to external network

The panel reported on the left, after clicking on the dedicated icon, allows including two entries for the external network connection. In the SSID field it will be indicated the name of the network selected for the connection and in the Password field the associated field.

Once these values will be submitted the ESP will restart and after a while it is important to be turned off and turned on again the entire FRESCO board, it is important to get a well alignment between the read and transferred data.



After that, the ESP will be connected to the external network, the main web page will display the IP where it is connected on the external network and the local IP as depicted in the Figure on the left.

### Influx and Grafana Manager

INFLUX\_URL

INFLUXDB\_ORG

INFLUXDB\_BUCKET

INFLUXDB\_TOKEN

Submit Clear

### Usage of the Database (🗄️)

Once the button 🗄️ is clicked you will be redirected to the following page and you can set the main input entries for InfluxDB V2. The next lines report the list of the entries that have to be written in the web app.

*INFLUX\_URL*

*INFLUXDB\_ORG* (organization ID)

*INFLUXDB\_BUCKET* (the database name)

*INFLUX\_TOKEN* (the password to connect the device to the database in a unique way.)

The following parameters can be collected by using the InfluxDB cloud or V2 installed on a local server. The next two sections will be devoted to explain step by step how to connect and setup the database and the dashboard to see the output data.

Once these values will be submitted the ESP will restart and after a while it is important to be turned off and turned on again the entire FRESCO board, it is important to get a well alignment between the read and transferred data.

## Settings (⚙️)

## Setting Parameters

## ON ESP8266

## NTP:

ntp.unifi.it

TZ\_INFO (TZ code available [here](#)):

CET-1CEST,M3.5.0,M10.5.0/3

## To be passed to Arduino

## NTC B Value:

3380

## Data Transfer Time (ms):

10000

## Saving Time (ms):

1000

## Displaying Time (ms):

5000

Irradiance cal. ([see manual](#)):

3.4

Submit

The setting web app page allows the user to manage the following entries:

Once the FRESCO device is connected to the WiFi, the NTP is needed to setup correctly the network time protocol useful for the certification to access and send data to InfluxDB (local and cloud version) but it is also important to align the RTC time on the device. According to the network should be necessary to change the NPT and it is possible thanks to the drop-down menu. The user can select the suitable NTP address from the list.

## NTP:

- ✓ ntp.unifi.it
- ntp1.inrim.it
- 0.pool.ntp.org
- at.pool.ntp.org**
- europe.pool.ntp.org
- oceania.pool.ntp.org
- asia.pool.ntp.org
- north-america.pool.ntp.org
- south-america.pool.ntp.org

After the NTP selection it is also important for a correct time alignment to select the time zone (TZ\_Info) and it is possible directly writing the unique code associated to the different TZ. The list of the codes is available at the following link or clicking in the webapp link.

[https://github.com/nayarsystems/posix\\_tz\\_db/blob/master/zones.csv](https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv)

for example:

Europe/Amsterdam CET-1CEST,M3.5.0,M10.5.0/3

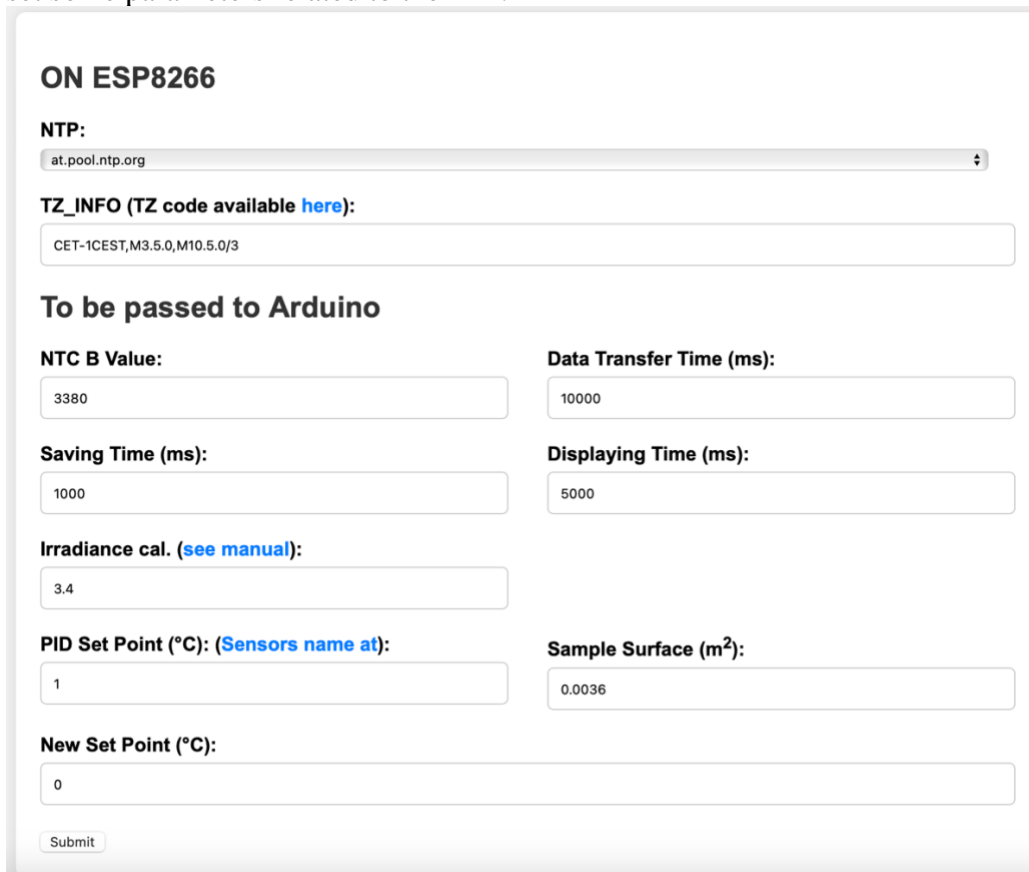
Indicates the Central European Summer Time (CEST) that take in account the winter and summer time shift, the Standard and Day Saving Time (DST) and automatically it allows aligning the RTC time avoiding any kind of delay.

In the section “*To be passed to Arduino*” is possible to set other entries such as:

- The Beta (also called K const) value related to the used NTC (pay attention to use 10k  $\Omega$  NTCs).
- “Data Transfer Time” The time to transfer the data over the ESP and the webapp or the Database.
- “Saving Time”: The time to save the data on the SD card (local database).
- “Displaying Time”: The time to display the information over the Oled display.
- “Irradiance cal.”: this value is used to calibrate the BH1750 with the mounted “dome”, if it results not well aligned with some weather station data can be used to re-align it.

Please pay attention that all the time have to be integer and are expressed in milli seconds, hence 1 second have to be written as 1000 ms.

FRESCO is now able to automatically detect if another shield such as PCool is mounted, in this case the settings, but also the main page, provide to the user other entries (or charts) and it is possible to set some parameters related to the PID.



**ON ESP8266**

**NTP:**

**TZ\_INFO (TZ code available [here](#)):**

**To be passed to Arduino**

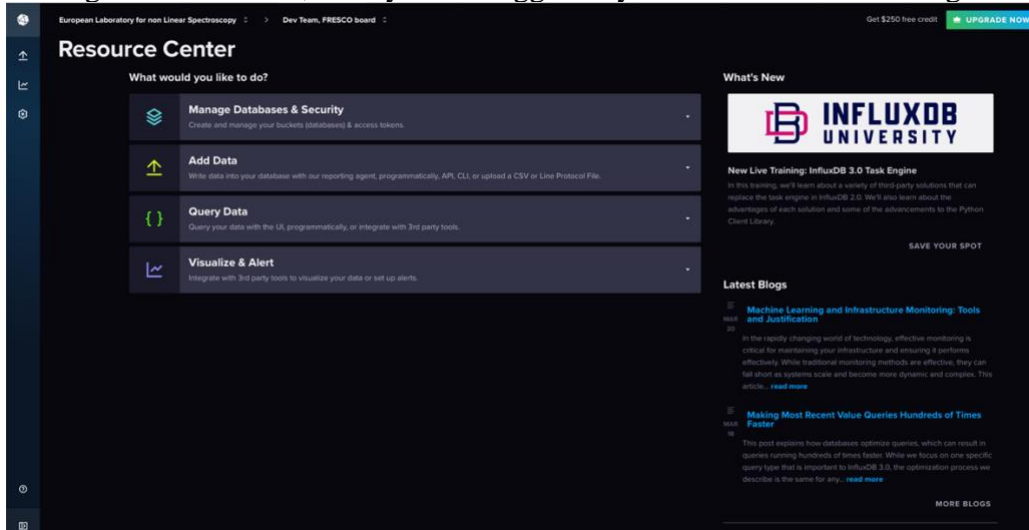
<b>NTC B Value:</b> <input type="text" value="3380"/>	<b>Data Transfer Time (ms):</b> <input type="text" value="10000"/>
<b>Saving Time (ms):</b> <input type="text" value="1000"/>	<b>Displaying Time (ms):</b> <input type="text" value="5000"/>
<b>Irradiance cal. (<a href="#">see manual</a>):</b> <input type="text" value="3.4"/>	
<b>PID Set Point (°C): (<a href="#">Sensors name at</a>):</b> <input type="text" value="1"/>	<b>Sample Surface (m<sup>2</sup>):</b> <input type="text" value="0.0036"/>
<b>New Set Point (°C):</b> <input type="text" value="0"/>	


In this case it is possible to set the **PID Set Point**, the **Sample Surface** used as substrate and to evaluate the cooling power density and in case a **New Set Point (°C)** to directly set the desired temperature to drive the PID. All the details about it's working principle are available at the following link

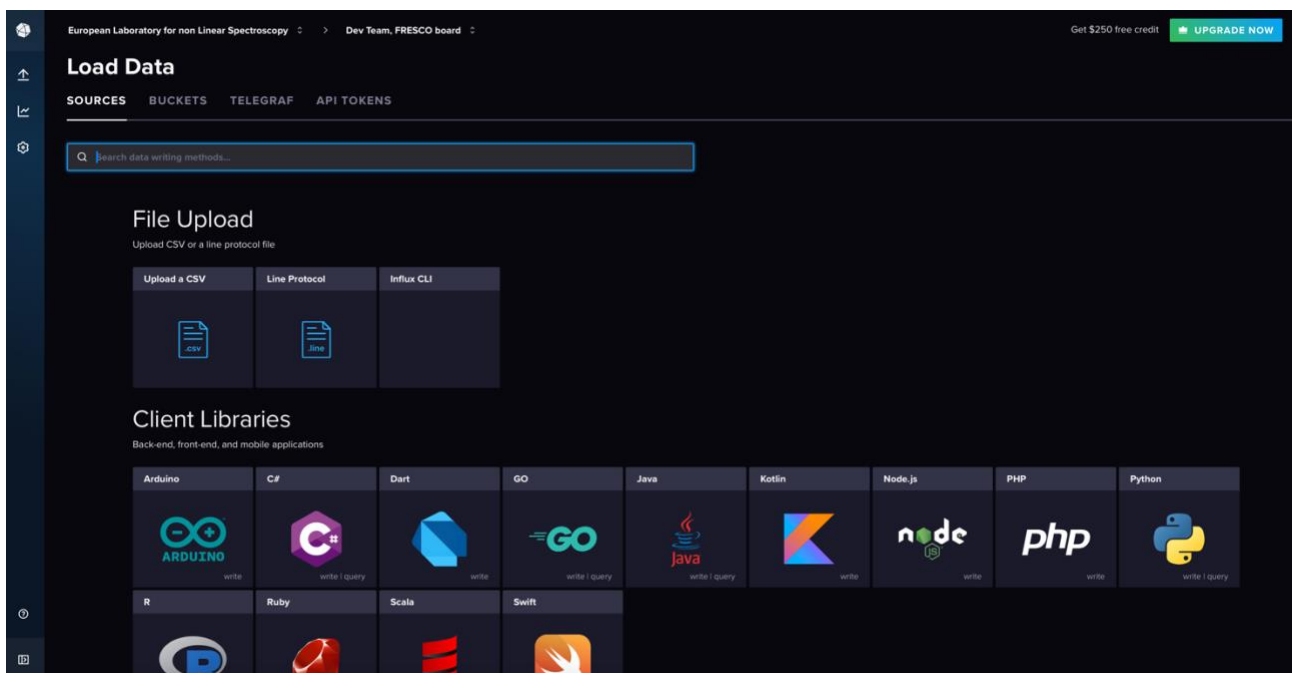
[https://github.com/GiuseppeELio/FRESCO-Board/blob/main/doc/FRESCO\\_IO\\_descript.pdf](https://github.com/GiuseppeELio/FRESCO-Board/blob/main/doc/FRESCO_IO_descript.pdf)

## Cloud InfluxDb and Grafana configuration

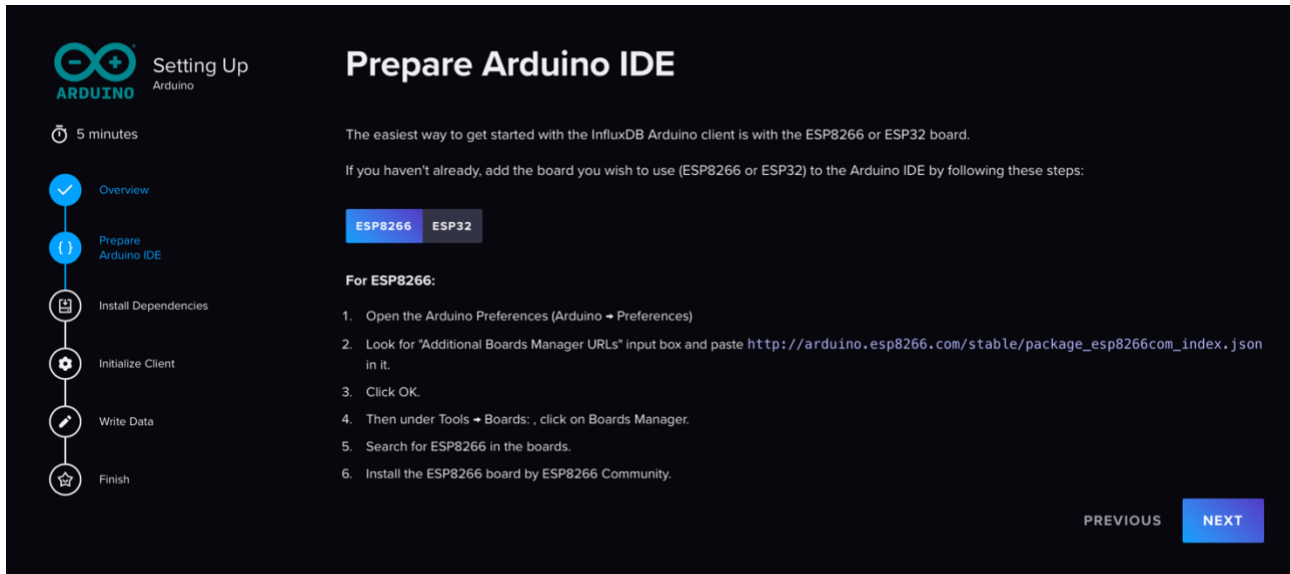
First of all to use the InfluxDB cloud it is necessary to go at the following link <https://cloud2.influxdata.com/signup> and register an account, once you are logged in you will find the following interface



Then you have to configure the database to work accordingly to a ESP8266 wifi and Arduino board, to to that go on sources () in the left menu and you will be directly moved on the following page



There you have to select “Arduino”, then it will start a guided procedure and make sure to select ESP8266 as reported in the next screenshot.



**Setting Up**  
Arduino

5 minutes

- Overview
- Prepare Arduino IDE**
- Install Dependencies
- Initialize Client
- Write Data
- Finish

## Prepare Arduino IDE

The easiest way to get started with the InfluxDB Arduino client is with the ESP8266 or ESP32 board.

If you haven't already, add the board you wish to use (ESP8266 or ESP32) to the Arduino IDE by following these steps:

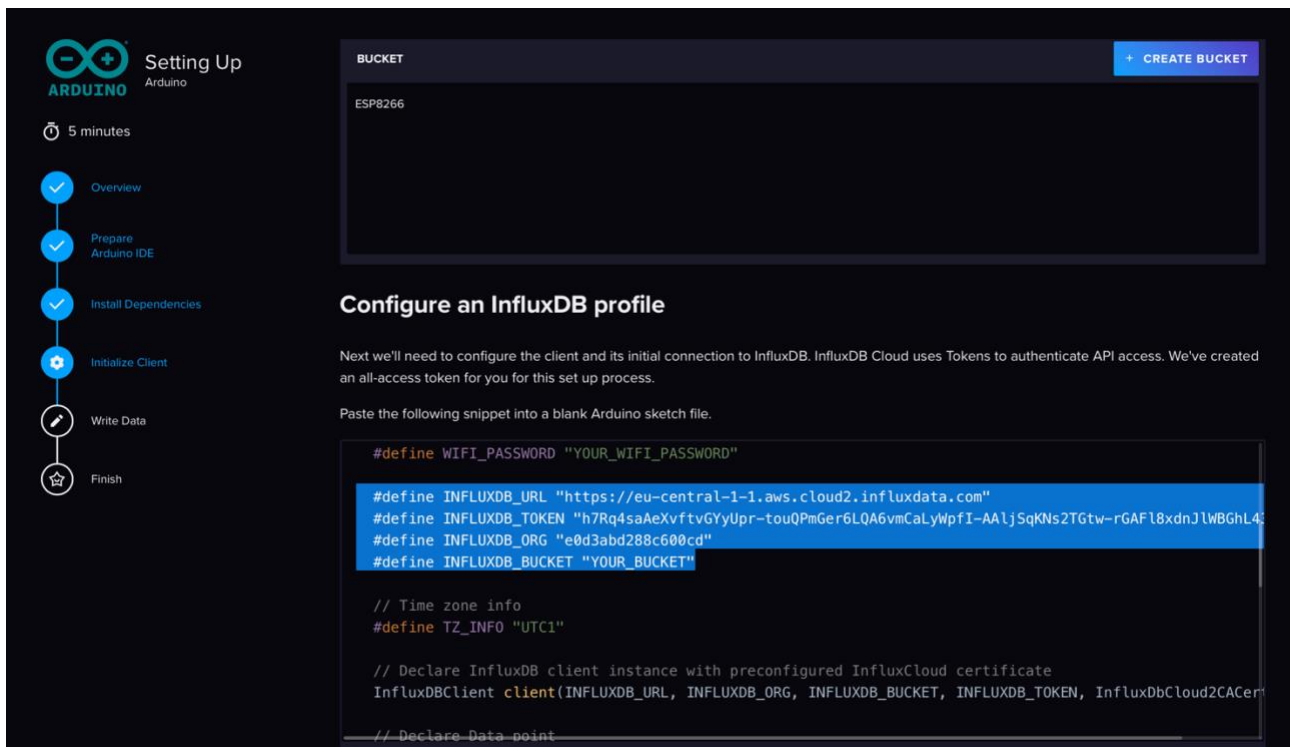
**ESP8266** **ESP32**

**For ESP8266:**

1. Open the Arduino Preferences (Arduino → Preferences)
2. Look for "Additional Boards Manager URLs" input box and paste [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) in it.
3. Click OK.
4. Then under Tools → Boards:, click on Boards Manager.
5. Search for ESP8266 in the boards.
6. Install the ESP8266 board by ESP8266 Community.

PREVIOUS **NEXT**

Then in “initialized client” you will define the name of your **Bucket** so you can left ESP8266 or you can change it as you prefer. Then, scrolling down you will arrive to the main important part, where it is displayed, and please remember that you will see them just now (if you lost the following credentials you have to do all the things again from a scratch.)



**Setting Up**  
Arduino

5 minutes

- Overview
- Prepare Arduino IDE
- Install Dependencies
- Initialize Client**
- Write Data
- Finish

## Configure an InfluxDB profile

Next we'll need to configure the client and its initial connection to InfluxDB. InfluxDB Cloud uses Tokens to authenticate API access. We've created an all-access token for you for this set up process.

Paste the following snippet into a blank Arduino sketch file.

```
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

#define INFLUXDB_URL "https://eu-central-1-1.aws.cloud2.influxdata.com"
#define INFLUXDB_TOKEN "h7Rq4saAeXvftvGYyUpr-touQPmGer6LQA6vmCaLyWpFI-AAljSqKns2TGtw-rGAF18xdnJlWBghL4"
#define INFLUXDB_ORG "e0d3abd288c600cd"
#define INFLUXDB_BUCKET "YOUR_BUCKET"

// Time zone info
#define TZ_INFO "UTC1"

// Declare InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACer

// Declare Data point
```

**BUCKET** **+ CREATE BUCKET**

ESP8266

The highlighted parts are the entries required in the database web app manager. Conclude the guided procedure until “Finish”. Once it has been created is time to include the parameters indicated before in the InfluxDB manager as following

## Influx and Grafana Manager

INFLUX\_URL

https://eu-central-1-1.aws.cloud2

Select InfluxDB

☒ Cloud  
INFLUXDB\_ORG

e0d3abd288c600cd

INFLUXDB\_BUCKET

ESP8266

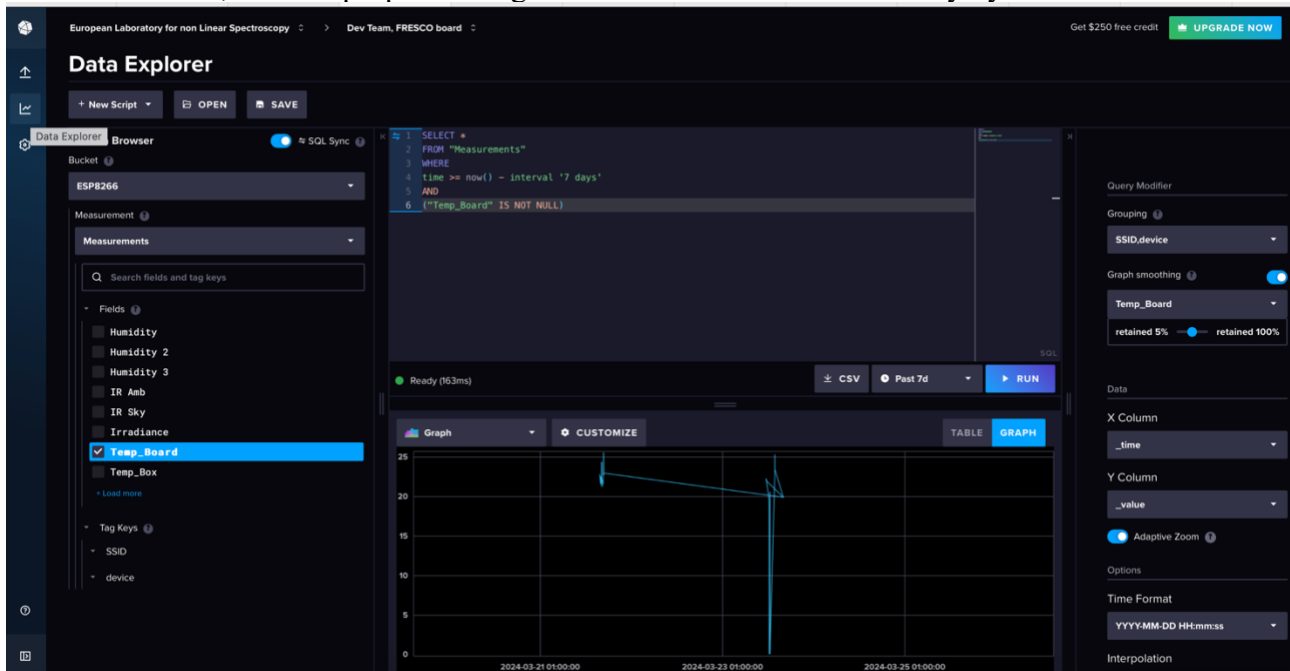
INFLUXDB\_TOKEN

s2TGtw-rGAFi8xdnJIWBGhL43E

Submit

Clear

Now it is necessary to use the InfluxDb webpage on the “Data Explorer” to see if the FRESCO board is transmitting the data, if it is yes on the left columns selecting the *Bucket* it will appear the field “Measurements”, this is a proper name given to the data collection directly by the FRESCO device.



Once the label “Measurements” has been selected all the contained data will be displayed in the menu placed below of it. Selecting for example Temp\_Board and the time interval in the center, under the code generator, and then Run a table of data or a graph will be plotted. The generated SQL code will be important for the next steps.

Here it is a simple SQL example to recall the data that we would like to visualize

```
SELECT time, "Temp Board"
FROM "Measurements"
WHERE time >= now() - interval '15 minutes'
AND ("Temp_Board" IS NOT NULL)
```

The highlighted parts are referred to the variable that we would like to see and the interval time. It is important to know that more than one variable for instance can be shown, it is enough to write for example

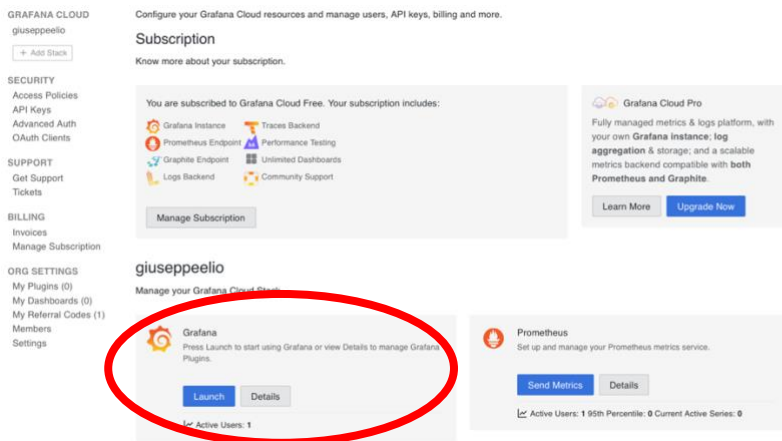
```
SELECT time, "Temp Board", "Temp S1", "Temp S2"
```

And to change the time interval can be minutes, hours or days. If you are referring to a single minute, 1 hour or 1 day use the single form of such word.

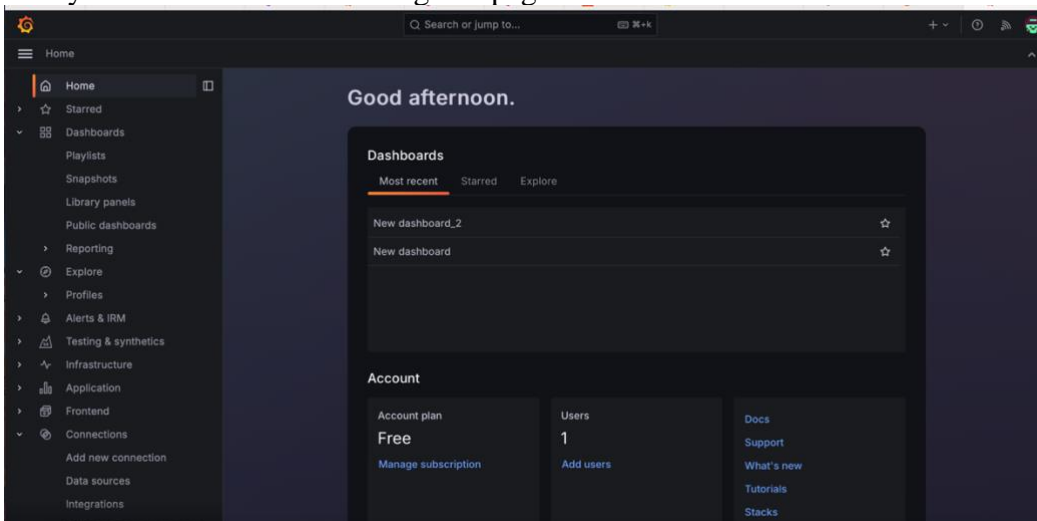
## GRAFANA

Now is time to use the stetted-up database and SQL codes for the **Grafana Dashboard** a simple and efficient way to visualize the collected data in real-time and whenever you want.

Start going to the following web page and create your [GRAFANA Free account](#). Then you can go directly to “My account”, and click on “Launch” your Grafana

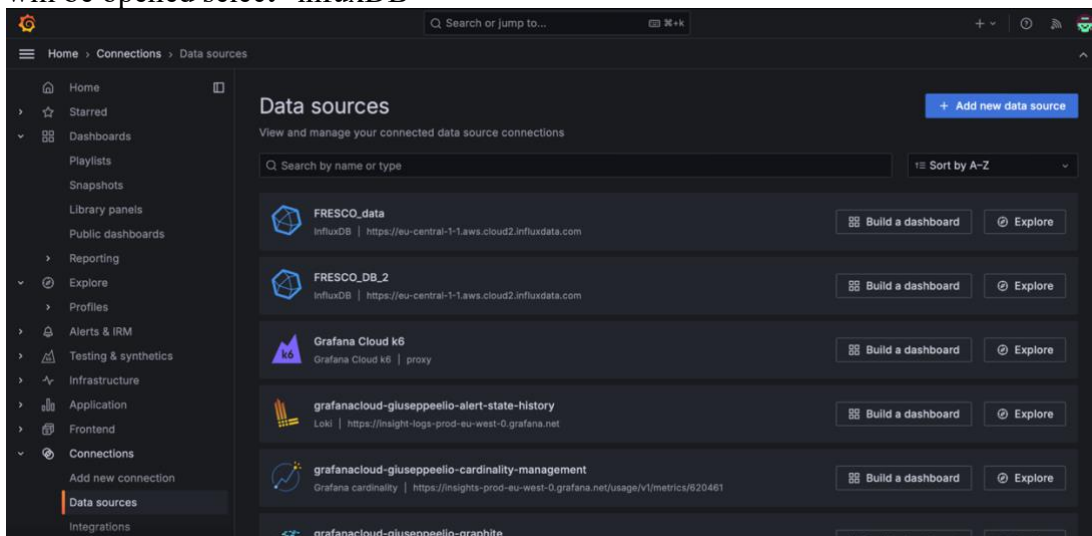


And you will reach the following webpage



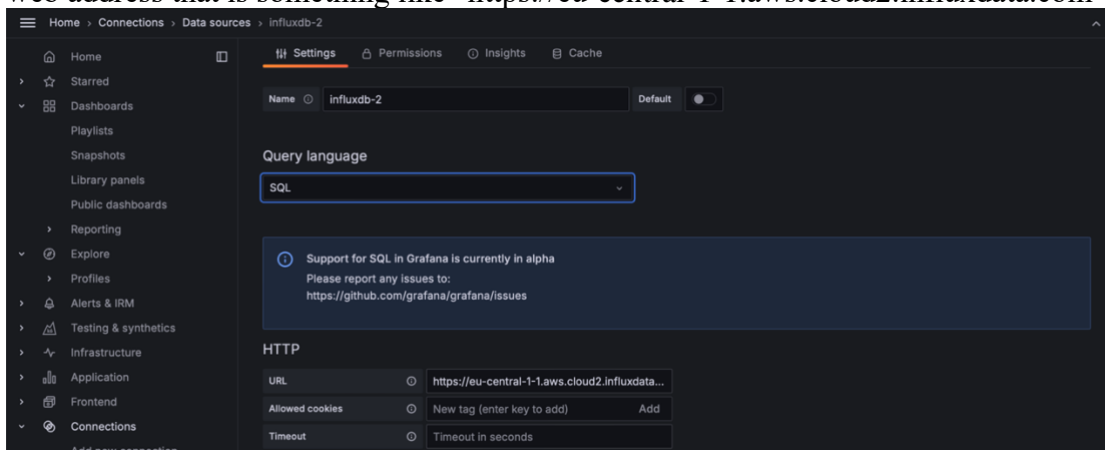


Here it is important to go on the left menu and scroll down until you are at the item “**Connections**” and then “**Data sources**”. Once you are here you have to press “+ add data source”, in the menu that will be opened select “influxDB”



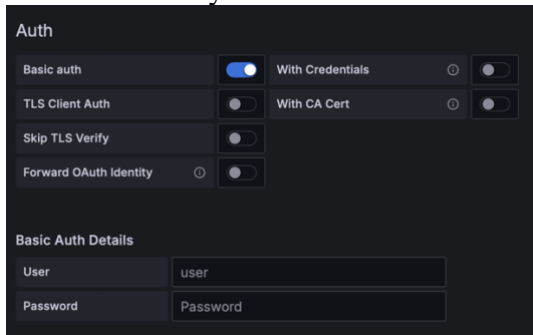
Following the next steps you will be able to connect Grafana to the database.

As reported in the following screenshot you can choose the name of the data source, and the Query language, be careful to select SQL. Then in the HTTP you have to indicate the InfluxDb Cloud web address that is something like “https://eu-central-1-1.aws.cloud2.influxdata.com”



Once all these fields are filled the next one, available scrolling the page, are:

Toggle on the use of Basic Authentication and in the related field use as User and Password the same credentials that you use for the InfluxDB Cloud.



In the next two fields it is necessary to indicate the Bucket and the Token again and then press “**Save & test**”.

### InfluxDB Details

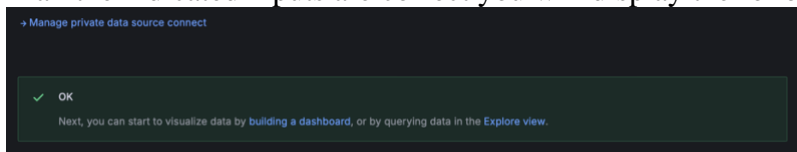
Database	
Token	
Insecure Connection	<input type="checkbox"/>
Max series	1000

Private data source connect

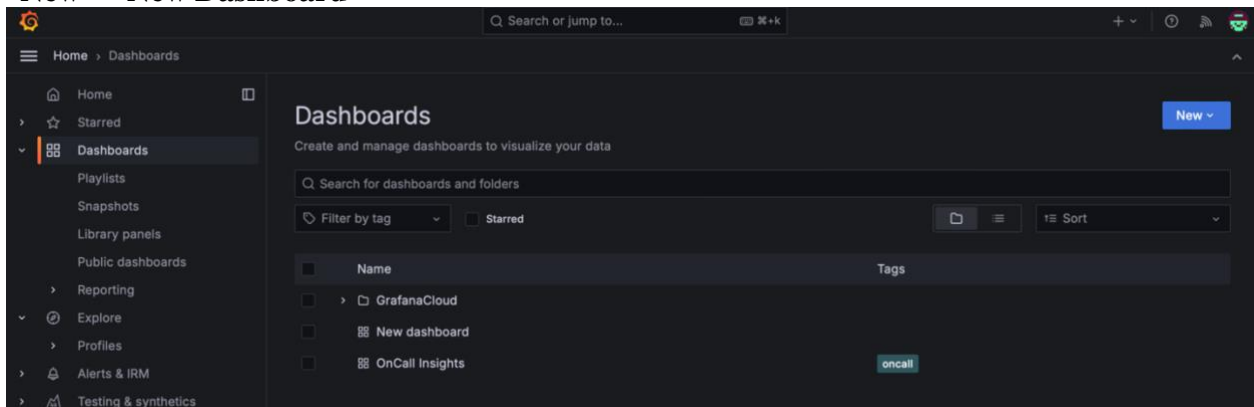
Private data source connect

[→ Manage private data source connect](#)

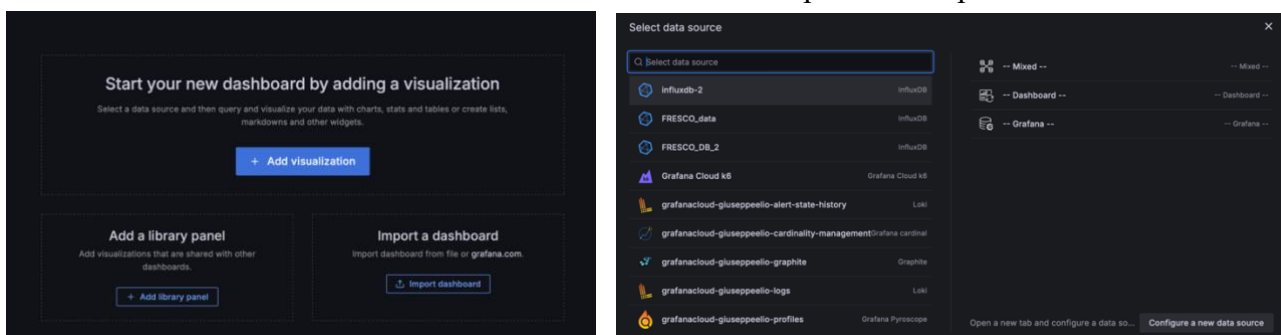
If all the indicated inputs are correct you will display the following banner.



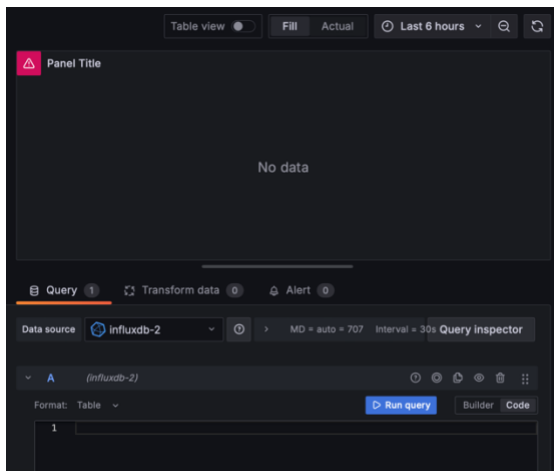
So now is time to go back to the Grafana Dashboard creation, in the “**Dashboard**” section press on “**New**”>”**New Dashboard**”



And you will display the following screen view and here you have to press on “**+ Add visualization**” and select the “**Data source**” and select the one created in the previous steps.



Then you will be redirected to the following interface and here you have to select in the Query A the voice related to “**code**”, now, here we can use the SQL code to see the data

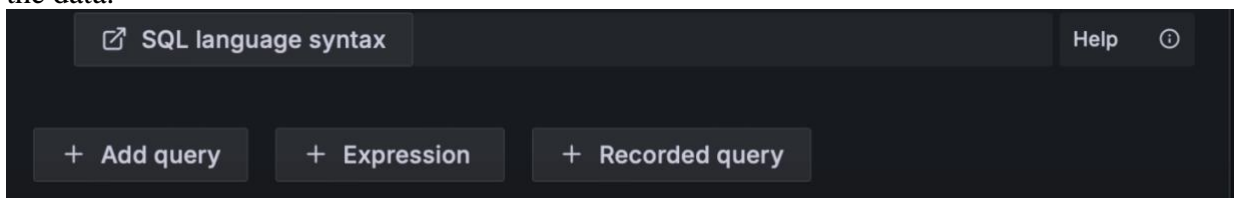


For example, including the following SQL code:

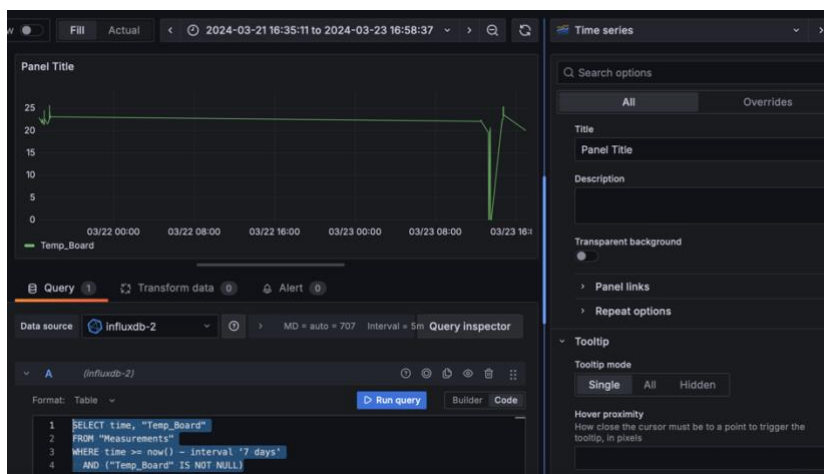
```
SELECT time, "Temp_Board"
FROM "Measurements"
WHERE time >= now() - interval '7 days'
AND ("Temp_Board" IS NOT NULL)
```

And pressing “**Run query**” it will display the collected data in the last 7 days, of course you can modify the time as you prefer. In the same chart is possible to add more Query just going on the bottom part of the chart and pressin “+ **Add query**”, then repeating the same procedure is possible to add another curve plotted in the same chart. In any case as already discussed before is

possible to add multiple items modifying the script as follow e.g SELECT time, "Temp\_Board", "Temp\_S1", "Temp\_S2". Different queries are useful for double Y axis or different way to visualize the data.



Once these values will be submitted the ESP will restart and after a while *it is important to turned off and turned on again the entire FRESCO board*, it is important to get a well alignment between the read and transferred data.



Now using the menu on the right side you can tailor the plots as you want, you can add more items in the same chart, change color lines, change the kind of plot from Time series, to histograms or gauges or what the user prefer. In any case Grafana is very flexible for all the usage and to be customized according to the user requirements.

The menu reported on the right side, accessible using the right side column in the previous interface, allows selecting different kind of charts or plots. It is possible to select the Time series, commonly used for data that change during the time, but also bar chart, statistical one that reports a banner with the last value measured, gauge meters, table to list the values, Pie chart or histograms and others.

It is also important that a *dashboard* can be easily *imported* using the related *Json structure file*. This allows to duplicate or manage an already existing Dashboard in Grafana and make it suitable for the selected purposes.

## Local InfluxDb and Grafana configuration

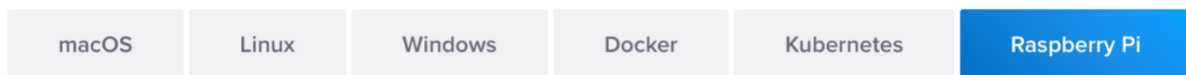
This section will be devoted on the procedure to set up a local server able to host a personal InfluxDb and Grafana. It will be expanded in the next time with all the advice related to prepare a Raspberry Pi (3 or 4 all models).

About the procedure to be followed to install InfluxDB V2 on a Raspberry or a Ubuntu (Linux) PC see the following guide

<https://docs.influxdata.com/influxdb/v2/install/>

As reported on this web page it is recommended to install a Ubuntu supported version on the Raspberry Pi

### 1. Download and install InfluxDB v2.

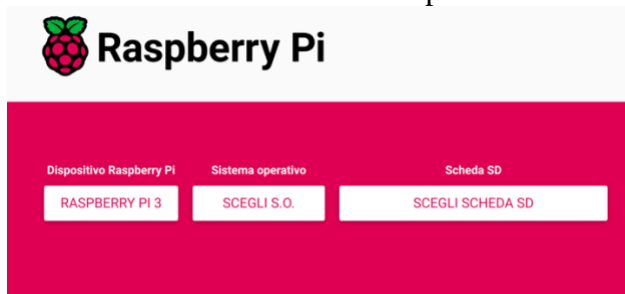


## Requirements

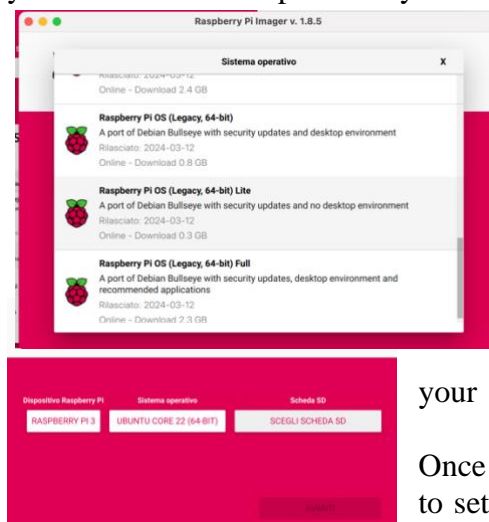
To run InfluxDB on Raspberry Pi, you need:

- a Raspberry Pi 4+ or 400
- a 64-bit operating system. We recommend installing a [64-bit version of Ubuntu](#) of Ubuntu Desktop or Ubuntu Server compatible with 64-bit Raspberry Pi.

Details about this procedure are reported here <https://ubuntu.com/download/raspberry-pi> you need to download the Raspberry Pi image <https://www.raspberrypi.com/software/> and install the Ubuntu stable version for IOT for 64 bit processor.



Once you have selected the you can choose the Operative system



your

Once this operation have been done, you can perform the procedure to set a static IP on your server as reported in the following guide

RaspBerry model that you posses

From this interface you have to select Raspberry Pi OS (Other), and you will go to the following page and select Raspberry Pi Os (Legacy 64-Bit) Lite (Bullseye distribution).

Finally, you can select the SD card used to startup and install the OS in Raspberry Pi.



<https://phoenixnap.com/kb/raspberry-pi-static-ip> it works for RaspbianOS but also for other the Linux distributions.

If you posses a Static IP follow this guide to set up the static IP on your Raspberry

<https://phoenixnap.com/kb/raspberry-pi-static-ip>

At this point the important poit is the use of the following command line

```
sudo nano /etc/dhcpd.conf
```

It allows opening the dhcpd configuration file and set the suitable values at the following lines:

```
interface [interface-name]
static ip_address=[ip-address]/[cidr-suffix]
static routers=[router-ip-address]
static domain_name_servers=[dns-address]
```

pay attention about the cidr-suffix, it depends on the subnet mask of your network, you can find the appropriate values at the following [website](#). The common value for cidr-suffix is 24.

be careful also to modify the item that are out of comment, in the conf file the commented lines are identified by #.

A possible example is where, of course, X,A,B,C,D,E,F are numbers

```
interface eth0
static ip_address=XXX.XXX.XXX.XXX/24
static routers=XXX.XXX.AAA.BBB
static domain_name_servers=XXX.XXX.CCC.DDD., XXX.XXX.EEE.FFF
```

Then, save the file and exit (normally is required to follow the instructions reported below in the console/ terminal).

Finally, reboot Raspberry Pi by typing:

```
sudo shutdown -r now
```

## InfluxDB V2 installation

After these initial steps to prepare the Raspberry Pi to host the InfluxDB V2 you can follow these simple steps to install it

In the terminal – console use the following commands:

```
curl -O https://dl.influxdata.com/influxdb/releases/influxdb2_2.7.5-1_arm64.deb
sudo dpkg -i influxdb2_2.7.5-1_arm64.deb
```

Then start the service using the command line

```
sudo service influxdb start
```

and check the status using

```
sudo service influxdb status
```

If successful, the output is the following:

```
● influxdb.service - InfluxDB is an open-source, distributed, time series database
   Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
  enable>
   Active: active (running)
```

If successful, you can view the InfluxDB UI at <http://localhost:8086>, in case you use a static IP you can view it at <http://IP:8086> or <http://domain.something.com:8086>

Arrived to this point, the procedure to setup your influxDB V2 local version is the same of the Cloud one as already described in the previous section. So also in local is necessary to generate the Token, the Bucket and to get the ORG ID code. The difference is in the URL that now is <http://localhost:8086>, in case you use a static IP you can view it at <http://IP:8086> or <http://domain.something.com:8086>

## Grafana

To install Grafana open the terminal/ console and insert the following command:

Install the prerequisite packages:

```
sudo apt-get install -y apt-transport-https software-properties-common wget
```

Then, import the GPG key

```
sudo mkdir -p /etc/apt/keyrings/  
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee  
/etc/apt/keyrings/grafana.gpg > /dev/null
```

To add a repository for stable releases, run the following command:

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com  
stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

To add a repository for beta releases, run the following command:

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com beta  
main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

Run the following command to update the list of available packages:

```
# Updates the list of available packages  
sudo apt-get update
```

To install Grafana OSS, run the following command:

```
# Installs the latest OSS release:  
sudo apt-get install grafana
```

## Reboot Raspberry

```
sudo reboot
```

or

```
sudo shutdown -r
```

## Start the Grafana server

Follow this procedure to *start the Grafana server with systemd*

Complete the following steps to start the Grafana server using systemd and verify that it is running.

To start the service, run the following commands:

```
sudo systemctl daemon-reload  
sudo systemctl start grafana-server  
sudo systemctl status grafana-server
```

To verify that the service is running, run the following command:

```
sudo systemctl status grafana-server
```

On the other hand, you can *Start the Grafana server using init.d*

Complete the following steps to start the Grafana server using init.d and verify that it is running:

```
sudo service grafana-server start
sudo service grafana-server status
```

To verify that the service is running, run the following command:

```
sudo service grafana-server status
```

[Optional procedure]

On the other hand you can also configure the Grafana server to start at boot using init.d

To configure the Grafana server to start at boot, run the following command:

```
sudo update-rc.d grafana-server defaults
```

Restart the Grafana server using init.d

To restart the Grafana server, run the following commands:

```
sudo service grafana-server restart
```

If successful, you can view the InfluxDB UI at <http://localhost:3000>, in case you use a static IP you can view it at <http://IP:3000> or <http://domain.something.com:3000>

The port 3000 allows acceding to the Grafana interface, register your account, please initially it has the username to access “admin” and the password “admin”, this password will be required to be changed immediately after the first access.

## Configuration of Influx Data Source on Grafana

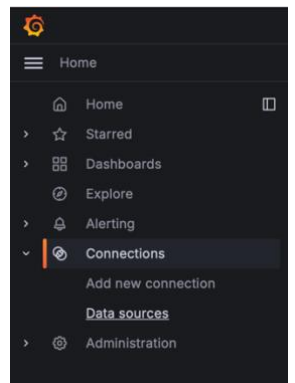
Finally, the local configuration of Grafana is **quite similar** to the one reported in the previous section for Grafana Cloud, the exceptions are the following:

- 1) in the “Data source” connection the Data Base *INFLUX URL* is the one used for the local InfluxDB V2.
- 2) The Data source connection on local Grafana uses the **FLUX** language instead of the *SQL* one, so follow the following steps to set it up.

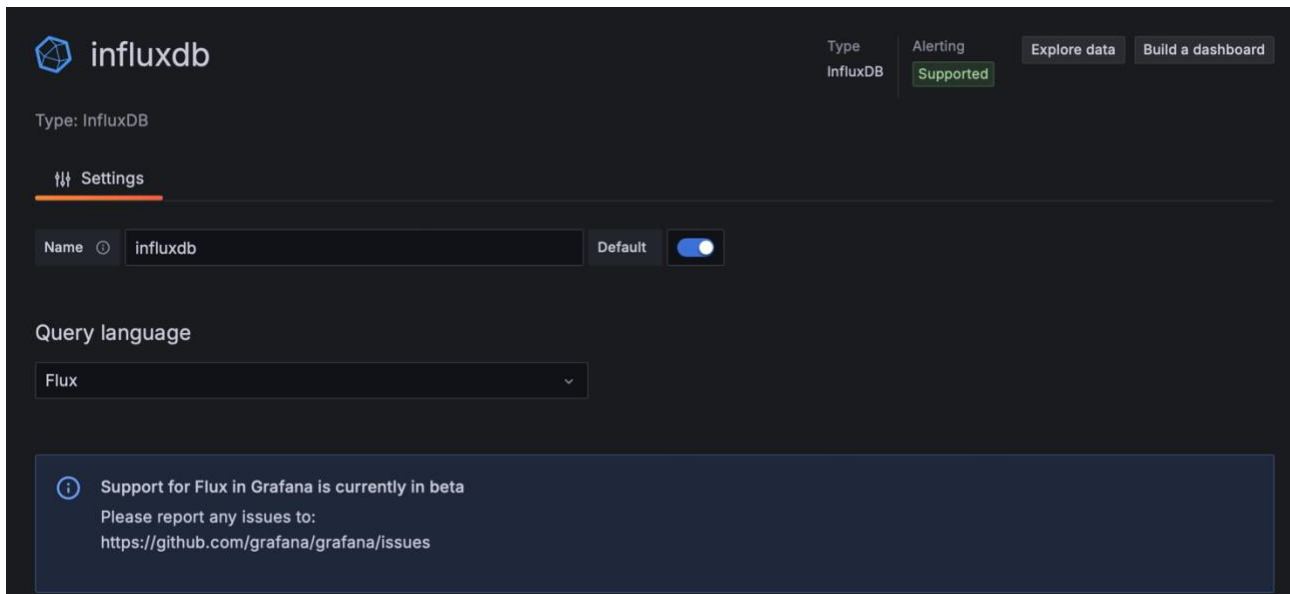
Now you can go on the left menu and scroll down until you are at the item “**Connections**” and then “**Data sources**”. Once you are here you have to press “+ **add Data sources**”, in the menu that will be opened select “influxDB”

Once the InfluxDB Data source has been addend, the following window will appear

Be sure to select as Query language **Flux**



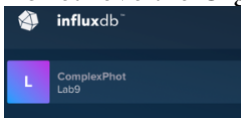




The other parameters and setting remains equal to the one used in the Cloud version, except for the fact that now is required the Organization name, a parameter that you have already setted when you start up the local InfluxDB.



To retrieve the Organization name you can go on InfluxDB page and find it under the account name.



Once everything has been done, you can move back to configure the Dashboard on Grafana, in this case the Query language to be included will have the following form

```
from(bucket: "ESP8266")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Measurements")
  |> filter(fn: (r) => r["_field"] == "Temp_amb" or r["_field"] == "Humidity")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

And it is retrieved by InfluxDB again using the Data Explorer section.

The dashboard and its Json code are available at the following [link](#).