

# DCR*Pi*: data center on a R*Pi*

Marco Zennaro, PhD  
ICTP



# Lab alert

The number of variables in the lab settings is huge (computer operating system, firewall, device firmware version, code version, network, etc)

Things will go wrong :-)

Be patient, we will solve all issues!

Found a bug? Let me know! Feedback is welcome.



# Our Lab equipment

Raspberry Pi

SD with latest Raspbian OS

Keyboard

Mouse

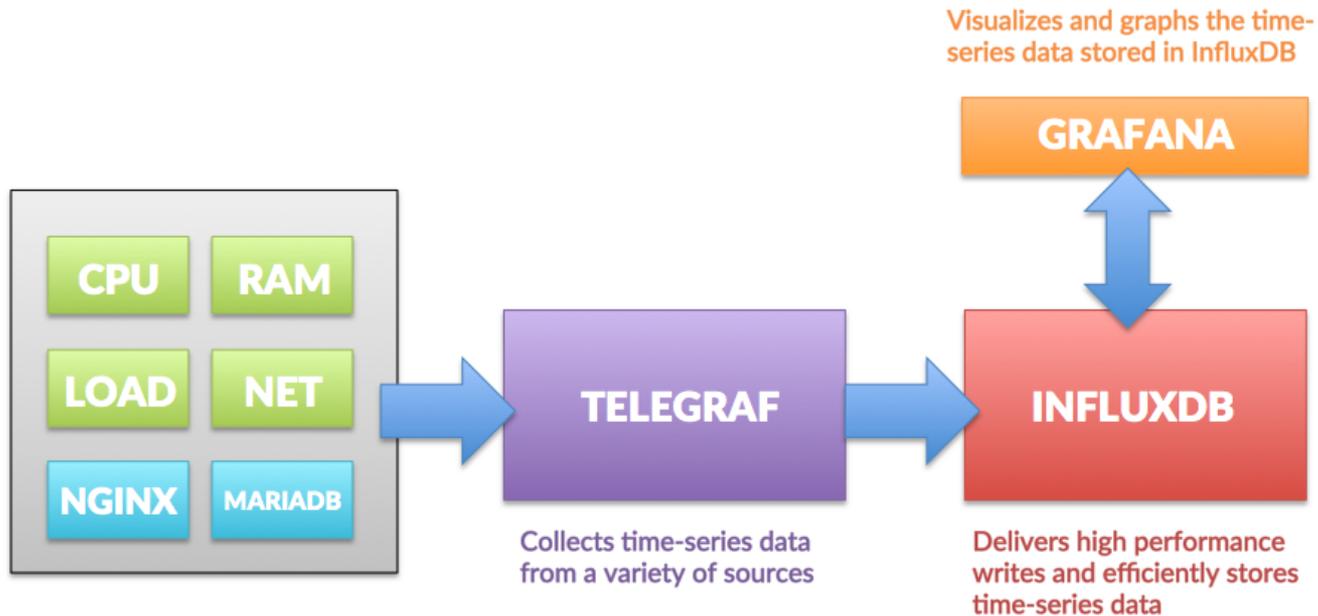
Screen

# What is the TIG Stack?

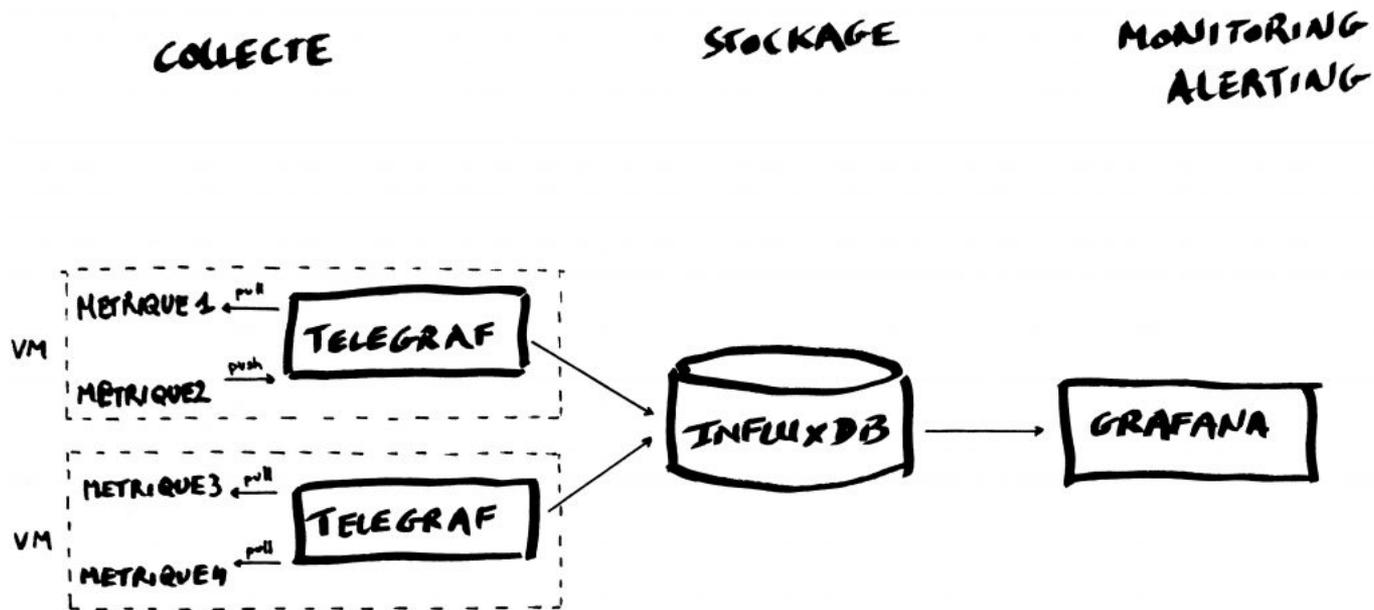
The **TIG Stack** is an acronym for a platform of open source tools built to make collection, storage, graphing, and alerting on **time series data** incredibly easy.



# What is the TIG Stack?



# What is the TIG Stack?



# What is a time series?

A time series is simply any set of values with a timestamp where time is a meaningful component of the data. The classic real world example of a time series is stock currency exchange price data.



# What is the TIG Stack?

**Telegraf:** A metrics collection agent. Use it to collect and send metrics to InfluxDB. Telegraf's plugin architecture supports collection of metrics from 100+ popular services right out of the box.

**InfluxDB** is a high performance Time Series Database. It can store hundreds of thousands of points per second. The InfluxDB SQL-like query language was built specifically for time series.



# What is the TIG Stack?

**Grafana** is an open-source platform for data visualization, monitoring and analysis. In Grafana, users can to create dashboards with panels, each representing specific metrics over a set time-frame. Grafana supports graph, table, heatmap and freetext panels.

# Installing TIG on a RPi

Let's start by adding the influxdb repositories:

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -  
echo "deb https://repos.influxdata.com/debian stretch stable" | sudo tee  
/etc/apt/sources.list.d/influxdb.list  
  
sudo apt-get update
```



# Installing TIG on a Rpi

We can now install Telegraf and Influxdb:

```
sudo apt-get install telegraf
```

```
sudo apt-get install influxdb
```

# Installing TIG on a RPi

Starting from v5.2.0-beta1 Grafana introduced official support for armv7 and arm64 linux platforms. Install it with:

```
sudo wget https://dl.grafana.com/oss/release/grafana-rpi_6.2.2_armhf.deb
```

```
sudo dpkg -i grafana-rpi_6.2.2_armhf.deb
```



# Installing TIG on a RPi

We can now activate all the services:

```
sudo systemctl enable influxdb
```

```
sudo systemctl start influxdb
```

```
sudo systemctl enable telegraf
```

```
sudo systemctl start telegraf
```

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

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



# Getting started with InfluxDB

InfluxDB is a time-series database compatible with SQL, so we can setup a database and a user easily. You can launch its shell with the *influx* command.

```
pi@raspberrypi:~ $ influx
```

# Creating a database

Next step is creating a database. Choose your name!

```
> CREATE DATABASE telegraf
```

```
> SHOW DATABASES
```

```
name: databases
```

```
name
```

```
_internal
```

```
telegraf
```



# Creating a user

Next step is creating a user and granting it full access to the database.

```
> CREATE USER telegraf WITH PASSWORD 'superpa$$word'
```

password is XXXX

```
> GRANT ALL ON telegraf TO telegraf
```

```
> SHOW USERS;
```

```
user  admin
```

```
telegraf false
```



# Retention Policy

A Retention Policy (RP) is the part of InfluxDB's data structure that describes for how long InfluxDB keeps data.

**InfluxDB** compares your local server's timestamp to the timestamps on your data and **deletes data that are older than the RP's DURATION**. A single database can have several RPs and RPs are unique per database.



# Installing TIG on a RPi

```
> CREATE RETENTION POLICY thirty_days ON telegraf DURATION 30d  
REPLICATION 1 DEFAULT
```

```
> SHOW RETENTION POLICIES ON telegraf
```

```
thirty_days 720h0m0s 1 TRUE
```

```
> exit
```



# Configuring Telegraf

Next, we have to configure the Telegraf instance to read from the TTN (The Things Network) server.

Luckily TTN runs a simple MQTT broker, so all we have to do is to edit the

`/etc/telegraf/telegraf.conf`

file to have the following section:



# Telegraf config 1/3

```
[agent]
```

```
hostname = "myserver"
```

```
flush_interval = "15s"
```

```
interval = "15s"
```

# Telegraf config 2/3

```
[[inputs.mqtt_consumer]]
servers = ["tcp://as.thethings.network:1883"]
qos = 0
connection_timeout = "30s"
topics = [ "+/devices/+/up" ]
client_id = ""
username = "test-bsfrance"
password = "ttn-account-
v2.TsFoWEWZe0xENIS_wjwTLuXavF3esk7tXME0ozwZCw8"
data_format = "value"
```

# Telegraf config 2/3

## APPLICATION OVERVIEW

**Application ID** test-bsfrance

**Description** test bsfrance lora device

**Created** 11 months ago

**Handler** ttn-handler-eu (current handler)

username



password



## ACCESS KEYS

 [manage keys](#)

default key

devices

messages



ttn-account-v2.TsFoWEWZe0xENIS\_wjwTLuXavF3esk7tXME0ozi

base64



# Telegraf config 3/3

```
[[outputs.influxdb]]
```

```
database = "telegraf"
```

```
urls = [ "http://localhost:8086" ]
```

```
username = "telegraf"
```

```
password = "superpa$$word"
```

# Restart Telegraf

Then we can restart telegraf and the metrics will begin to be collected and sent to InfluxDB.

```
pi@raspberrypi:~ $ service telegraf restart
```



# Check database

We can check if the data is sent from Telegraf to InfluxDB:

```
pi@raspberrypi:~ $ influx
```

Enter an InfluxQL query

```
> use telegraf
```

Using database telegraf

```
> select * from "mqtt_consumer"
```



# Database is populated!

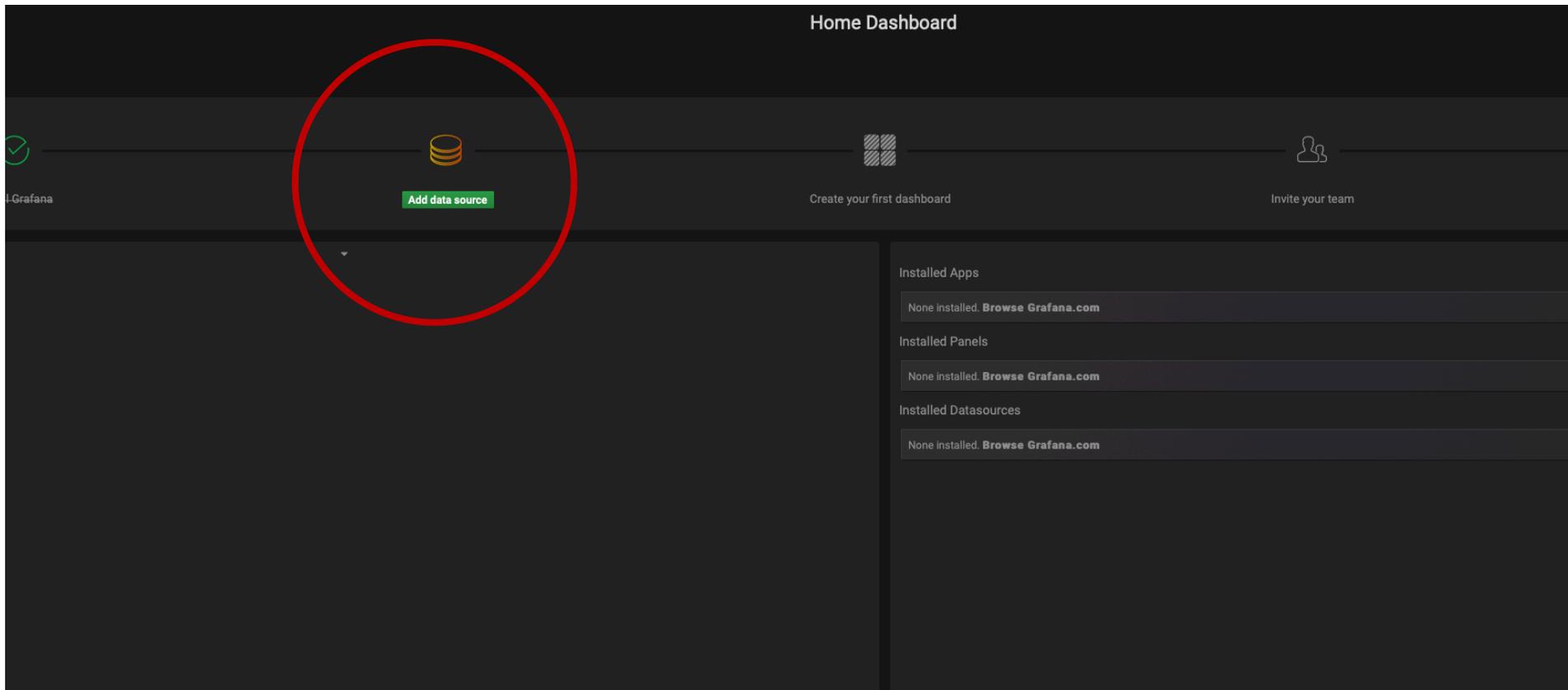
1557323990319369114	292	myserver	287744000	868.3	15	1
45.703526	13.72079		1	-112	-5.8	
294082396	0	0	1008.1	23.6		
45	0	2.92	7204	23.3	3.9	0
0	292	8459640	1	test-bsfrance/devices/bsfabp0001/up		

1557324301943104151	293	myserver	287744000	868.5	15	2
45.703526	13.72079		1	-112	-6.2	
605705244	0	0	1008.1	23.5		
45	0	2.92	7204	23.3	3.9	0
0	293	8482785	1	test-bsfrance/devices/bsfabp0001/up		

# Log into Grafana

- Address: <http://127.0.0.1:3000/login>
- Username: admin
- Password: admin

# Add data source



# Add data source 1/2

Name

Type: InfluxDB

The screenshot shows the configuration interface for a data source named 'Venice Weather Station'. The page title is 'Data Sources / Venice Weather Station' with a sub-label 'Type: InfluxDB'. A 'Settings' menu is visible. The configuration is organized into sections: 'Name' (Venice Weather Station, marked as 'Default'), 'Type' (InfluxDB), 'HTTP' (URL: http://localhost:8086, Access: Server (Default)), and 'Auth' (Basic Auth, TLS Client Auth, and Skip TLS Verification (Insecure) options).

Name	Venice Weather Station	Default	<input checked="" type="checkbox"/>
Type	InfluxDB		
<b>HTTP</b>			
URL	http://localhost:8086		
Access	Server (Default)	Help	
<b>Auth</b>			
Basic Auth	<input type="checkbox"/>	With Credentials	<input type="checkbox"/>
TLS Client Auth	<input type="checkbox"/>	With CA Cert	<input type="checkbox"/>
Skip TLS Verification (Insecure)	<input type="checkbox"/>		

Address



# Add data source 2/2

InfluxDB database name

InfluxDB database username

InfluxDB database passwd

The screenshot shows a configuration form for an InfluxDB data source. It includes a 'Database' field with the value 'telegraf', a 'User' field with 'telegraf', and a 'Password' field with masked characters. Below the form is a 'Database Access' section with explanatory text and a query example. At the bottom, there is a 'Min time interval' field set to '10s'.

**InfluxDB Details**

Database	telegraf		
User	telegraf	Password	....

**Database Access**

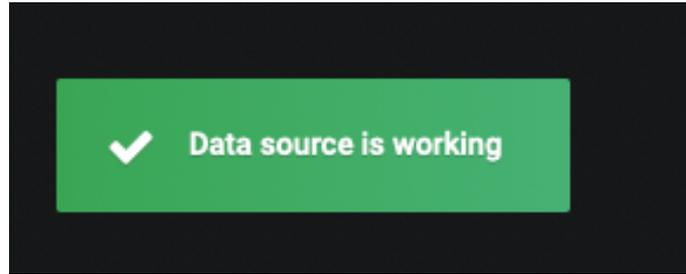
Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval 10s ⓘ

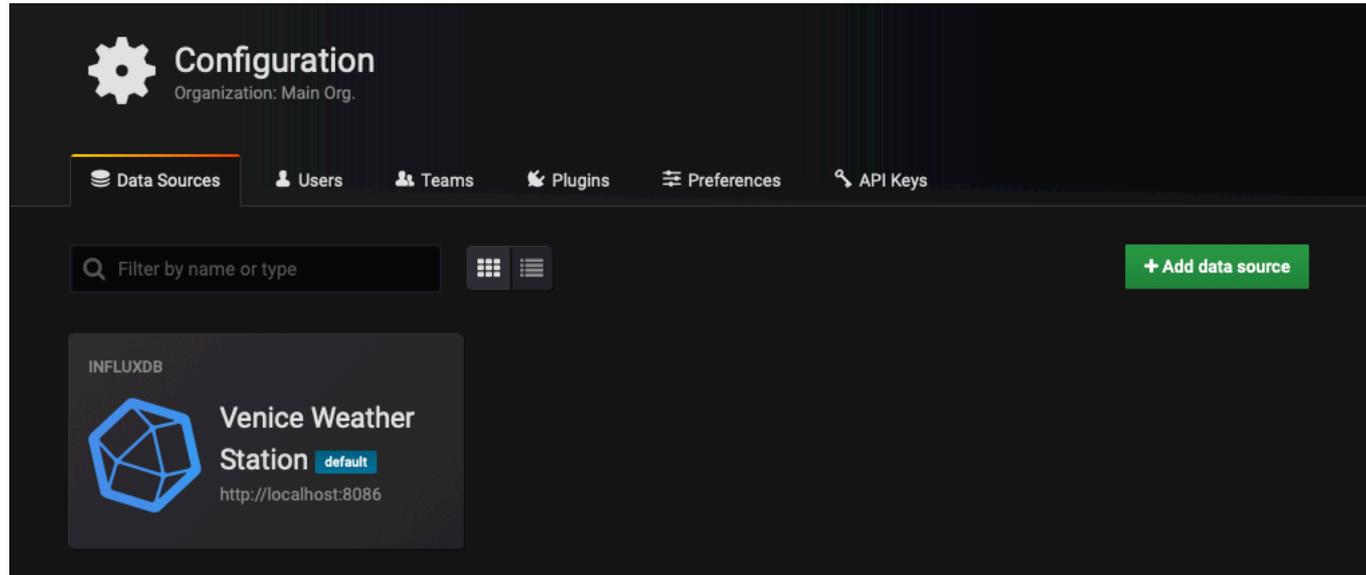
# Add data source

If everything is fine you should see:

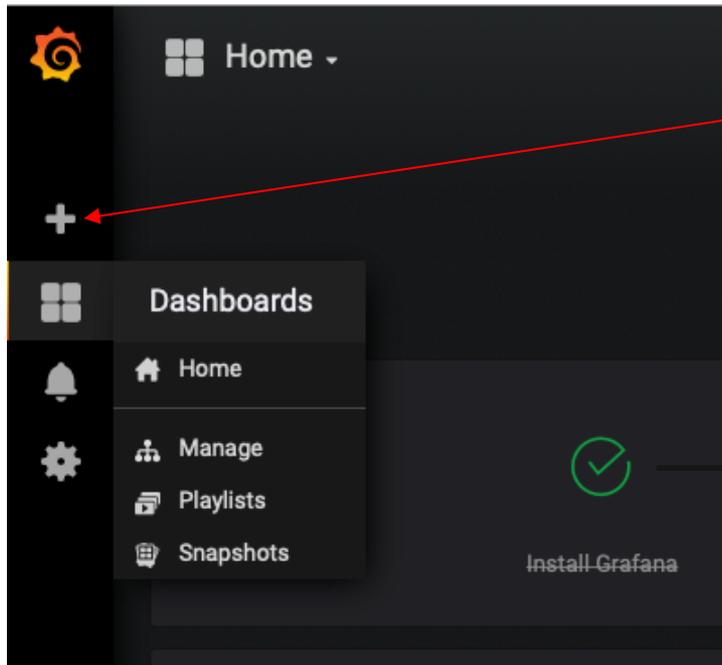


# Add data source

If everything is fine you should see:

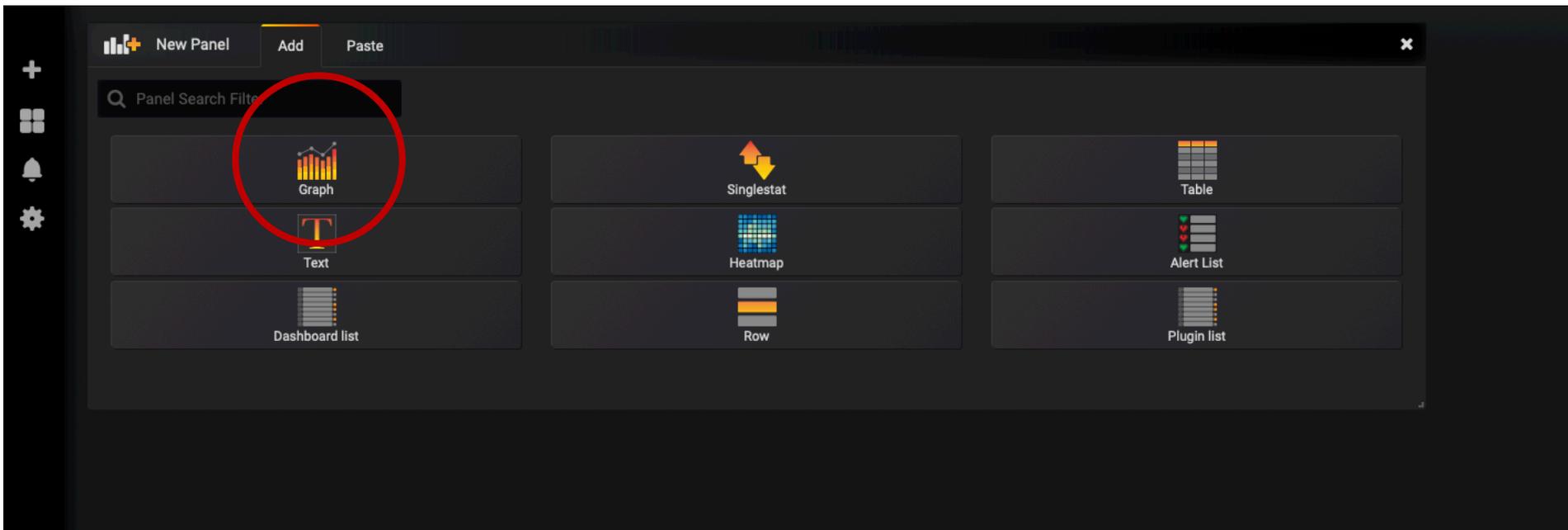


# Add Dashboard



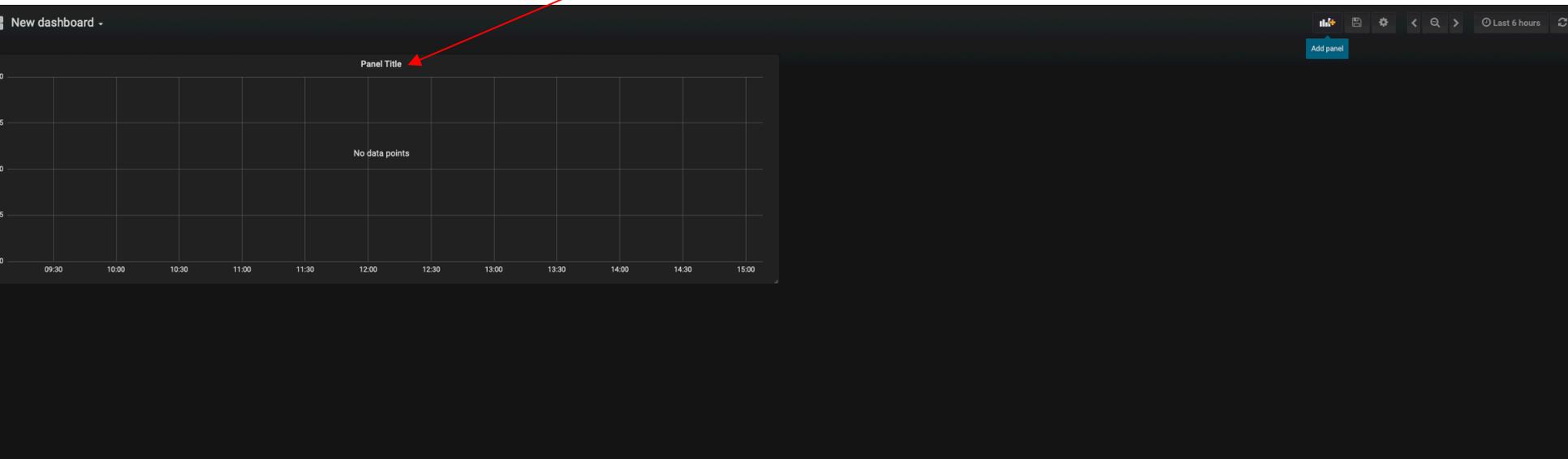
New

# Add graph



# Empty graph!

Select Edit



# Add info to Graph: General

Graph

General Metrics Axes Legend Display Alert Time range

Info

Title Panel Title

Description Panel description, supports markdown & links

Transparent

Drilldown / detail link ⓘ

+ Add link

Repeat

For each value of

## Add Title and Description

# Add info to Graph: Metrics

The screenshot shows the 'Metrics' configuration panel in Grafana. The 'Data Source' field is empty and highlighted with a blue border. Below it, the 'FROM' dropdown menu is open, showing options: 'default', 'ruuvi', and 'Venice Weather Station'. A red arrow points from the text 'Your InfluxDB database name' to the 'ruuvi' option. The 'SELECT' dropdown shows '- Grafana -' and '- Mixed -'. The 'GROUP BY' field contains 'time (\$\_\_interval)' and 'fill (null)'. The 'FORMAT AS' dropdown is set to 'Time series'. The 'ALIAS BY' field contains 'Naming pattern'. At the bottom left, there is an 'Add Query' button.

Your InfluxDB  
database name

# Add info to Graph: Metrics

The screenshot shows the Grafana query editor interface. The 'Metrics' tab is selected. The data source is 'Venice Weather Station'. The query editor shows a query with the following fields:

Field	Value
FROM	default
SELECT	field (value)
GROUP BY	time (\$__interval)
FORMAT AS	Time series
ALIAS BY	Naming scheme

A dropdown menu is open for the 'SELECT' field, listing the following metrics:

- cpu
- disk
- diskio
- kernel
- mem
- mqtt\_consumer
- processes
- swap
- system

The 'mqtt\_consumer' metric is highlighted, and a red arrow points to it from the text 'Select mqtt\_consumer' on the right.

Select  
mqtt\_consumer

# Add info to Graph: Metrics

The screenshot shows the Grafana 'Graph' panel configuration interface. The 'Metrics' tab is selected. The 'Data Source' is set to 'Venice Weather Station'. The 'FROM' clause is 'default' and 'mqtt\_consumer'. The 'WHERE' clause is empty. The 'SELECT' clause is 'field ()' with a dropdown menu open showing various metrics. The 'GROUP BY' clause is 'time ()', 'FORMAT AS' is 'Time', and 'ALIAS BY' is 'Name'. An 'Add Query' button is visible at the bottom left.

Clause	Value
FROM	default mqtt_consumer
WHERE	
SELECT	field ()
GROUP BY	time ()
FORMAT AS	Time
ALIAS BY	Name

Available metrics in the dropdown menu:

- counter
- metadata\_airtime
- metadata\_frequency
- metadata\_gateways\_0\_altitude
- metadata\_gateways\_0\_channel
- metadata\_gateways\_0\_latitude
- metadata\_gateways\_0\_longitude
- metadata\_gateways\_0\_rt\_chain
- metadata\_gateways\_0\_rssi
- metadata\_gateways\_0\_snr
- metadata\_gateways\_0\_timestamp
- payload\_fields\_ActiveRain

Select  
the variable  
you want to  
graph

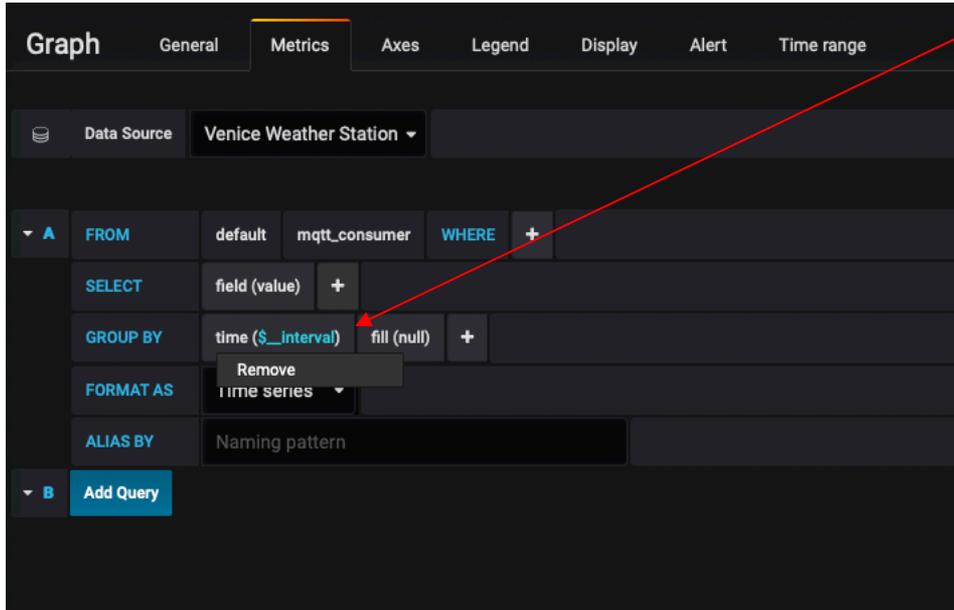
# Add info to Graph: Metrics

Remove  
mean()

The screenshot shows a configuration interface for a graph. The 'Metrics' tab is active. The 'Data Source' is set to 'Venice Weather Station'. The 'FROM' clause includes 'default' and 'mqtt\_consumer'. The 'SELECT' clause contains 'field (value)' and 'mean ()'. A 'Remove' button is positioned over the 'mean ()' function. The 'GROUP BY' clause is 'time (\$\_interval)'. The 'FORMAT AS' is 'Time series'. The 'ALIAS BY' is 'Naming pattern'. There is an 'Add Query' button at the bottom left.

Section	Field	Value
FROM	default	mqtt_consumer
WHERE		
SELECT	field (value)	mean ()
GROUP BY	time (\$_interval)	
FORMAT AS		Time series
ALIAS BY		Naming pattern

# Add info to Graph: Metrics



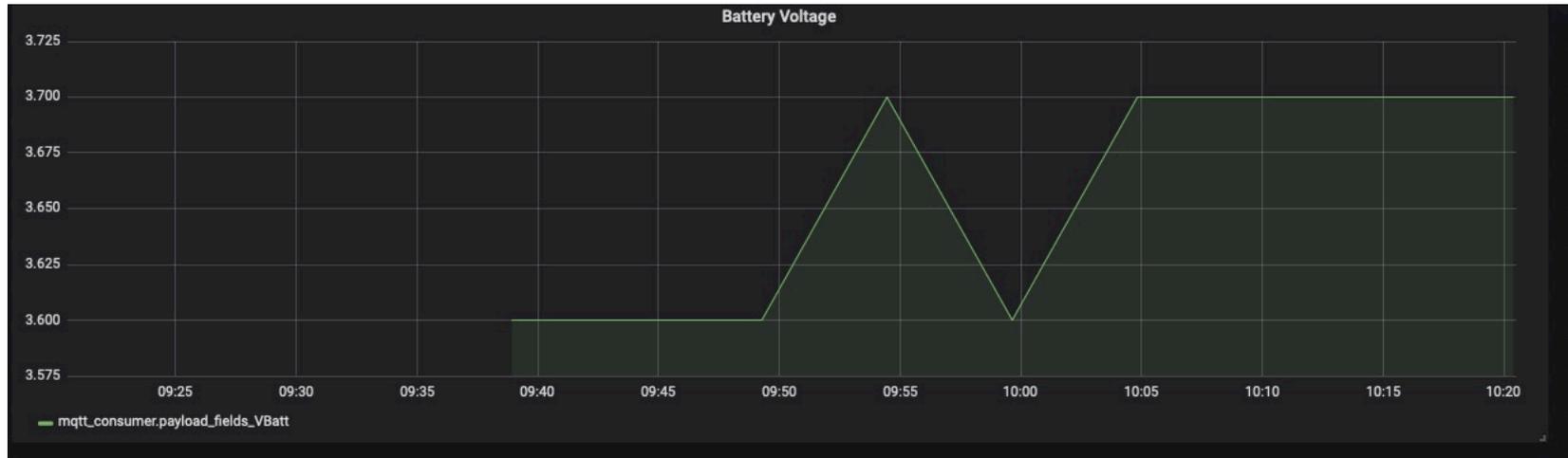
The screenshot shows the Grafana Metrics configuration panel for a query. The 'Metrics' tab is selected. The data source is 'Venice Weather Station'. The query is configured as follows:

Section	Field	Value
FROM	default	mqtt_consumer
WHERE		
SELECT	field (value)	
GROUP BY	time (\$_interval)	fill (null)
FORMAT AS	Remove	time series
ALIAS BY		Naming pattern

An 'Add Query' button is visible at the bottom left. A red arrow points from the text 'Remove time(\$\_interval)' to the 'time (\$\_interval)' field in the GROUP BY section.

Remove  
time(\$\_interval)

# Final result



# Final result

- You can add as many variables as you want to the same Dashboard
- You can add users and different users can have access to different Dashboards
- You can export Dashboards
- Have fun exploring Grafana!

# InfluxDB and Python

- You can interact with your Influx database using Python
- You need to install a library called *influxdb*
- Complete instructions are here:  
<https://www.influxdata.com/blog/getting-started-python-influxdb/>

# InfluxDB and Python

Like many Python libraries, the easiest way to get up and running is to install the library using pip:

```
$ python3 -m pip install influxdb
```

Now let's launch Python and import the library:

```
>>> from influxdb import InfluxDBClient
```



# InfluxDB and Python

Next we create a new instance of the `InfluxDBClient` with information about the server that we want to access.

```
>>> client = InfluxDBClient(host='localhost', port=8086)
```

If Influx has username and password then:

```
>>> client = InfluxDBClient(host='mydomain.com', port=8086,  
username='myuser', password='mypass' ssl=True,  
verify_ssl=True)
```



# InfluxDB and Python

Finally, we will list all databases and set the client to use a specific database:

```
>>> client.get_list_database()
```

```
>>> client.switch_database('telegraf')
```

# InfluxDB and Python

Let's try to get some data from the database:

```
>>> client.query('SELECT * from "mqtt_consumer"')
```

The `query()` function returns a `ResultSet` object, which contains all the data of the result along with some convenience methods. Our query is requesting all the measurements in our database.



# InfluxDB and Python

You can use the `get_points()` method of the `ResultSet` to get the measurements from the request, filtering by tag or field:

```
>>> points=results.get_points()
```

```
>>> for item in points:
```

```
    print(item['time'])
```

# InfluxDB and Python

You can get mean values, number of items, etc:

```
>>> client.query('select count(payload_fields_Rainfall) from  
mqtt_consumer')
```

```
>>> client.query('select mean(payload_fields_Rainfall) from  
mqtt_consumer')
```

```
client.query('select * from mqtt_consumer WHERE time >  
now() - 7d')
```



# Influx and Python: Exercises

- 1) Save the data as csv (comma separated values) using Python and InfluxDB.
- 2) Produce a graph of the last 20 temperature measurements using Python and InfluxDB.

# Summary

We learned how to install Telegraf, InfluxDB and Grafana.

We learned how to use Grafana to visualize data coming from an IoT network.

We learned how to interact with InfluxDB using Python.



# Feedback?

Email [mzennaro@ictp.it](mailto:mzennaro@ictp.it)