

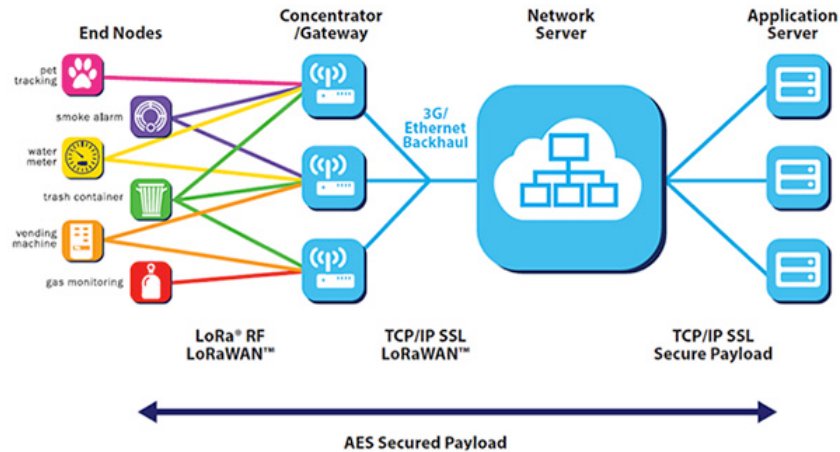
DCR*Pi*U: data center on a ~~R~~*p*~~i~~Ubuntu

Marco Zennaro, PhD
ICTP



LoRaWAN architecture

MQTT Broker!



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.

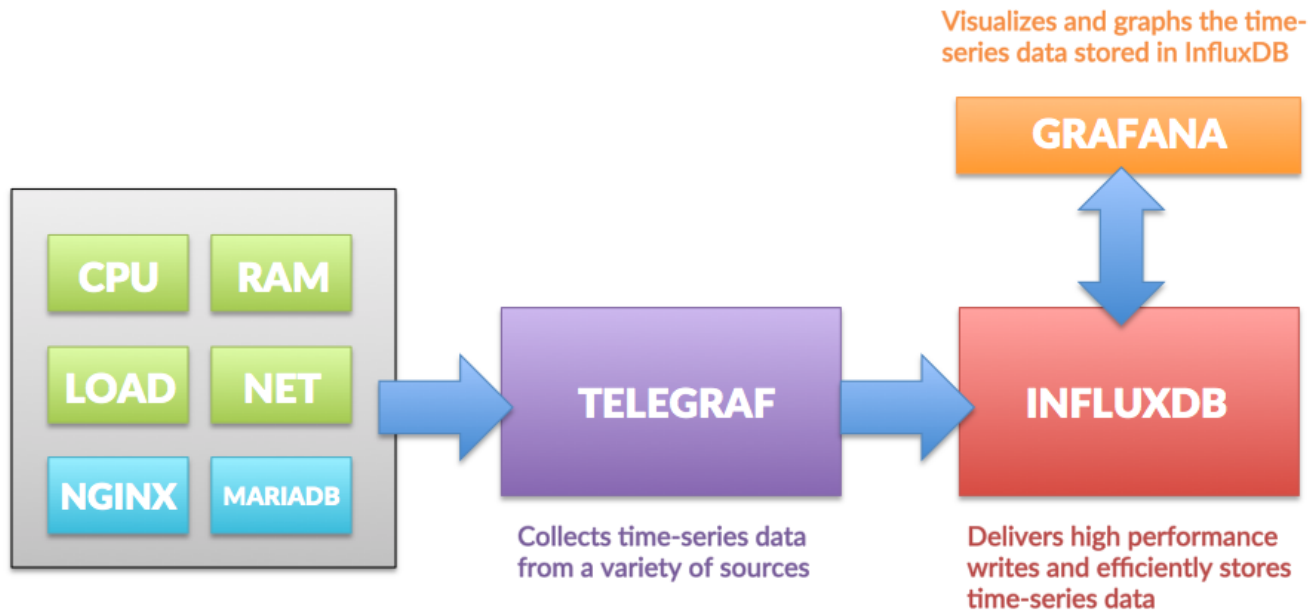


What is a time series?

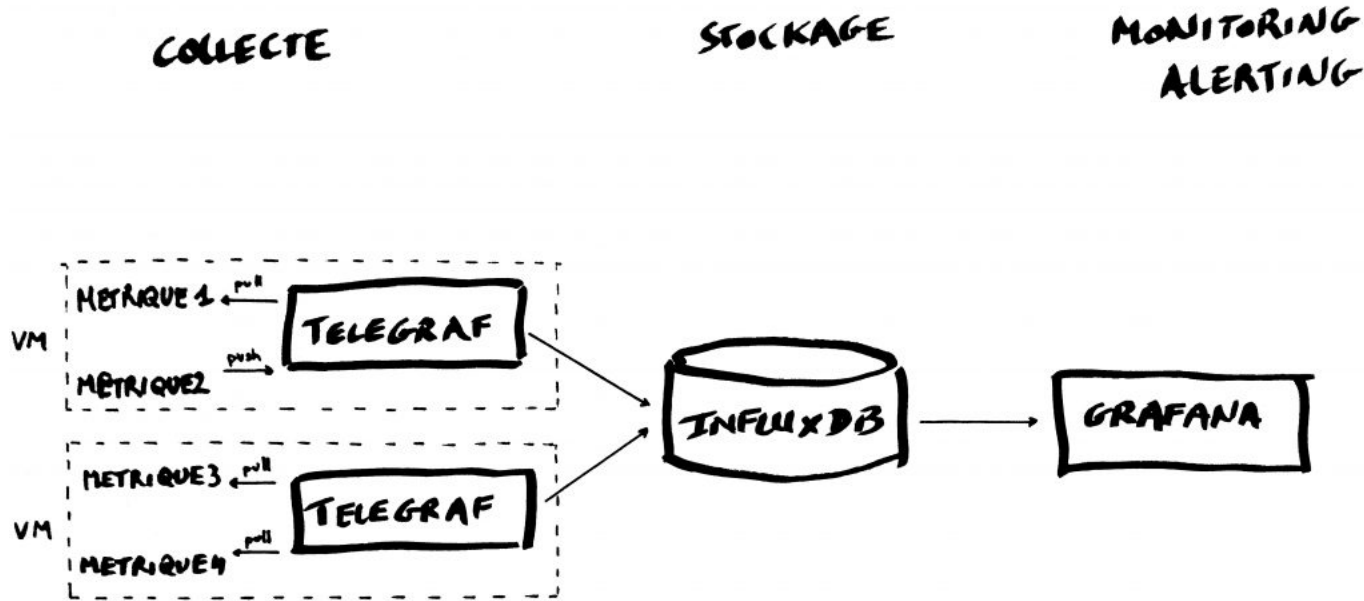
A time series is simply any set of values with a timestamp where time is a meaningful component of the data. The classic real world example of a time series is stock currency exchange price data.



What is the TIG Stack?



What is the TIG Stack?



What is the TIG Stack?

Telegraf is 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 free text panels.

Installing TIG on a Linux machine

For Ubuntu users, follow these commands
(<https://docs.influxdata.com/influxdb/v1.7/introduction/installation/>)

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

```
source /etc/lsb-release
```

```
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" |  
sudo tee /etc/apt/sources.list.d/influxdb.list
```

Installing TIG on a Linux machine

We can now install Telegraf and Influxdb:

```
sudo apt-get update
```

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

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

Installing TIG on a Linux machine

Starting from v5.2.0-beta1 Grafana introduced official support for arm64 linux platforms. For Ubuntu install it with:

```
wget https://dl.grafana.com/oss/release/grafana_6.2.5_amd64.deb
```

```
sudo dpkg -i grafana_6.2.5_amd64.deb
```

If you are running an old version of Ubuntu, you might have to install an older version of Grafana.



Installing TIG

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 database_name
```

```
> SHOW DATABASES
```

```
name: databases
```

```
name
```

```
_internal
```

```
database_name
```

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.



Retention Policy

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

```
> SHOW RETENTION POLICIES ON database_name
```

```
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 Telegraf configuration file to connect via MQTT to TTN.

Configuring Telegraf

First create a backup copy of the config file. In Ubuntu, enter the following command in the terminal:

```
mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf_original
```

Then edit the config file:

```
> sudo nano /etc/telegraf/telegraf.conf
```

Telegraf config 1/3

```
[agent]
```

```
hostname = "myserver"
```

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

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

Telegraf config 2/3

```
[[inputs.mqtt_consumer]]
```

```
servers = ["tcp://asia-se.thethings.network:1883"]
```

```
qos = 0
```

```
connection_timeout = "30s"
```

```
topics = [ "+/devices/+/up" ]
```

```
client_id = ""
```

```
username = "test-bsfrance"
```

```
password = "ttn-account-  
v2.TsFoWEWZe0xENIS_wjwTLuXavF3esk7tXME0ozwZCw8"
```

```
data_format = "json"
```

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

ACCESS KEYS

 [manage keys](#)

default key

devices

messages



ttn-account-v2.TsFoWEWZe0xENIS_wjwTLuXavF3esk7tXME0ozv

base64



password

Telegraf config 3/3

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

```
database = "database_name"
```

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

Save and exit the editor with control+X, Yes and Enter.

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 now check if the data is sent from Telegraf to InfluxDB:

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

Enter an InfluxQL query

```
> use database_name
```

Using database telegraf

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


Database is populated!

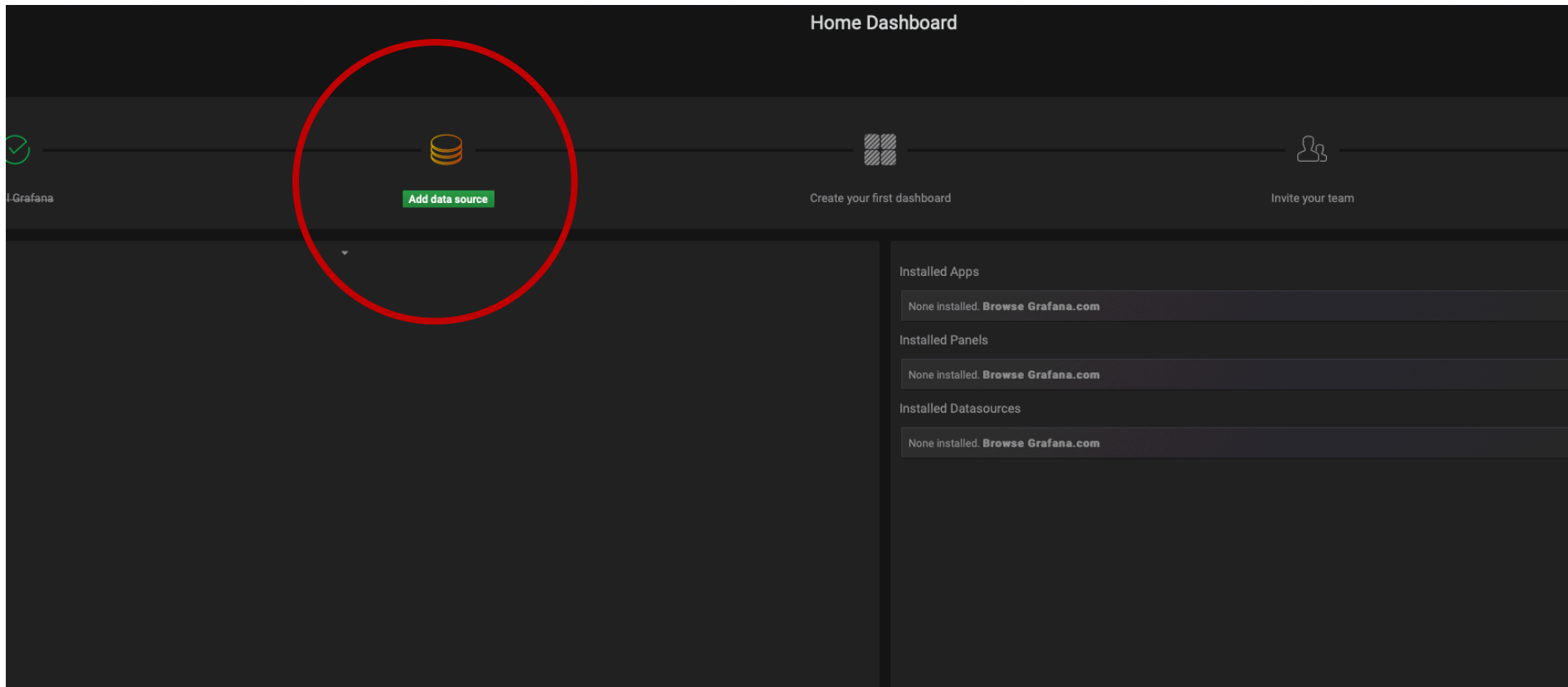
1557323990319369114	292	myserver	287744000	868.3	15	1
45.703526	13.72079		1	-112	-5.8	
294082396	0	0	1008.1	23.6		
45	0	2.92	7204	23.3	3.9	0
0	292	8459640	1	test-bsfrance/devices/bsfabp0001/up		

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

Data Sources / Venice Weather Station
Type: InfluxDB

Settings

Name: Venice Weather Station Default ☒

Type: InfluxDB

HTTP

URL: http://localhost:8086

Access: Server (Default) [Help](#)

Auth

Basic Auth ☐ With Credentials ☐

TLS Client Auth ☐ With CA Cert ☐

Skip TLS Verification (Insecure) ☐

Name

Type: InfluxDB

Address

Add data source 2/2

InfluxDB database name

InfluxDB database username

InfluxDB database passwd

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`

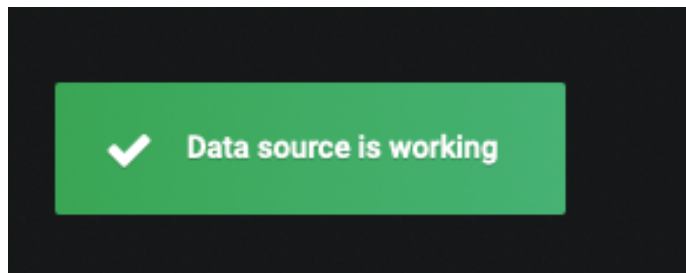
To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval

10s

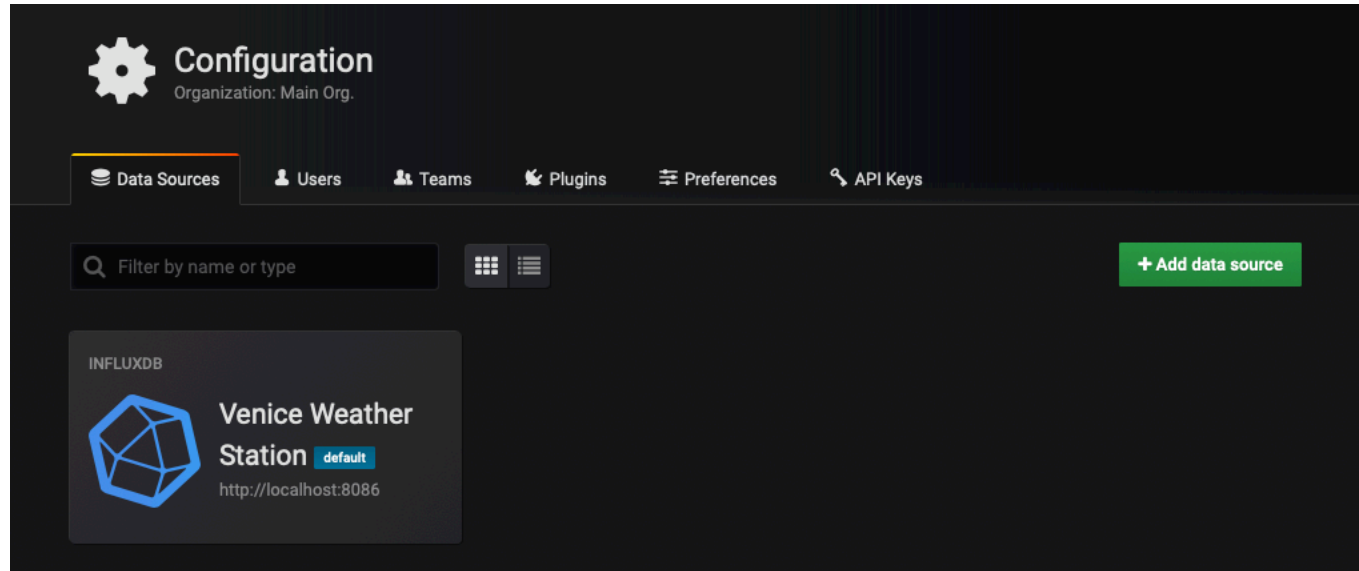
Add data source

If everything is fine you should see:

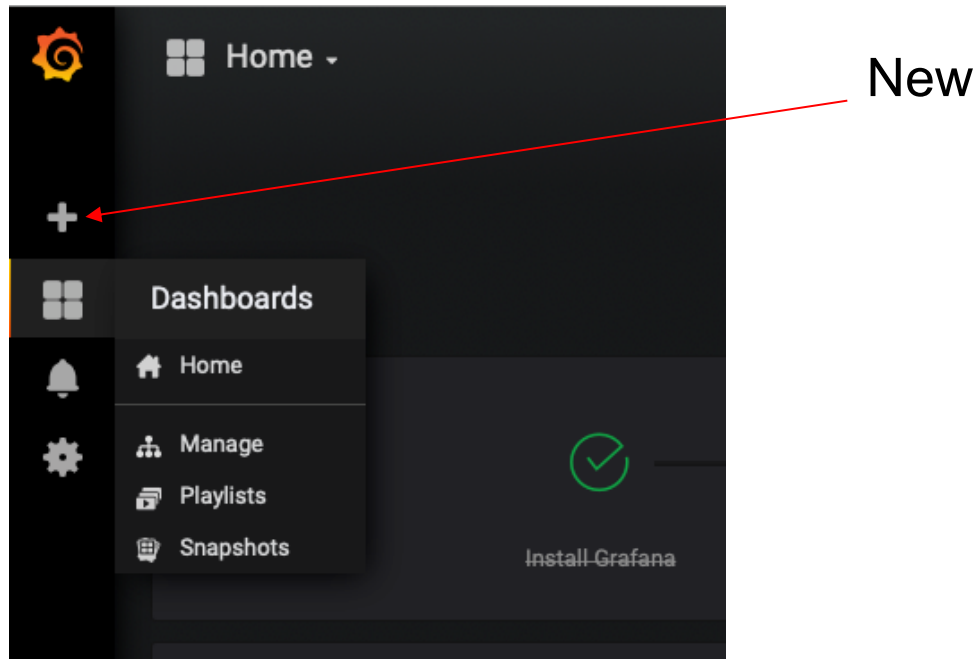


Add data source

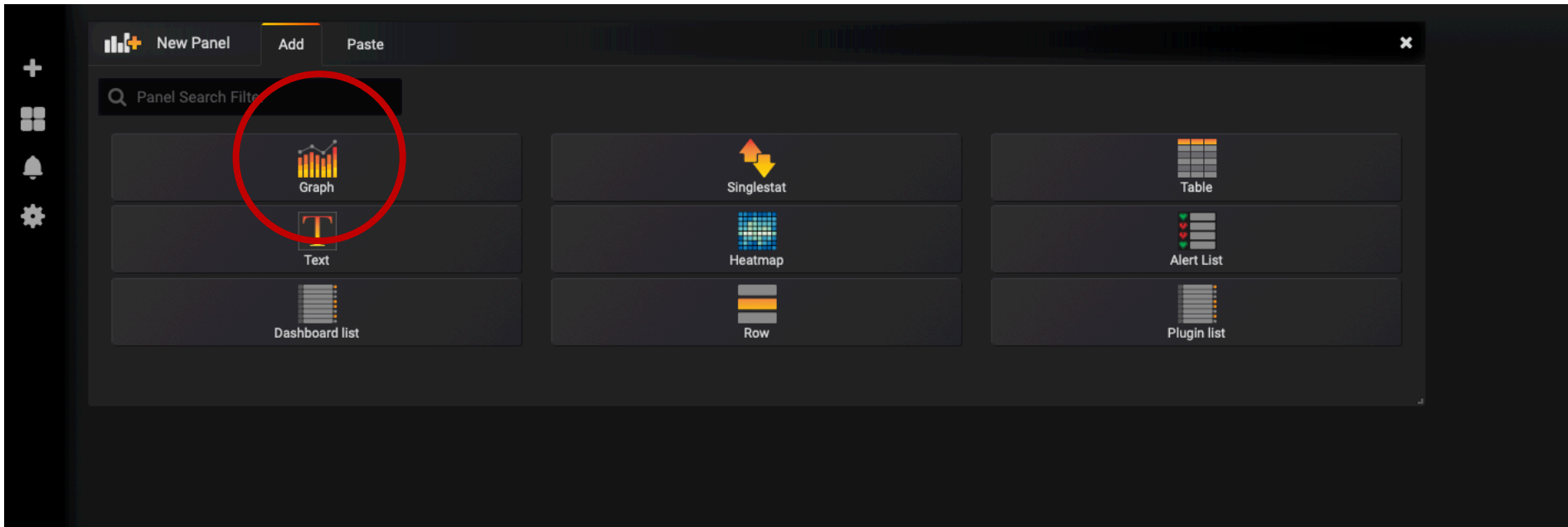
If everything is fine you should see:



Add Dashboard

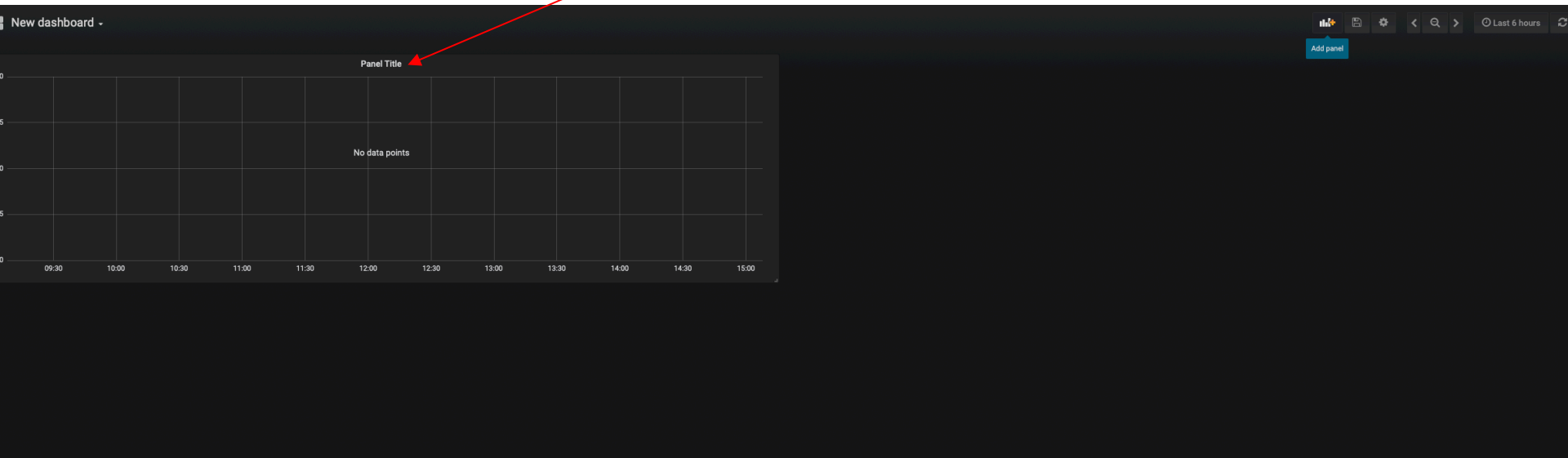


Add graph



Empty graph!

Select Edit



Add info to Graph: General

The screenshot shows a configuration interface for a graph panel. At the top, there is a horizontal menu with tabs: 'Graph', 'General', 'Metrics', 'Axes', 'Legend', 'Display', 'Alert', and 'Time range'. The 'General' tab is currently selected and highlighted with an orange underline. Below the menu, the interface is divided into two main sections. On the left, under the heading 'Info', there are three fields: 'Title' with the value 'Panel Title', 'Description' with the placeholder text 'Panel description, supports markdown & links', and 'Transparent' with an unchecked checkbox. On the right, under the heading 'Repeat', there is a dropdown menu labeled 'For each value of' with a downward arrow. At the bottom of the interface, there is a section labeled 'Drilldown / detail link' with a help icon, and a button labeled '+ Add link'.

Add Title and Description

Add info to Graph: Metrics

The screenshot shows the Grafana 'Metrics' configuration page. The 'Data Source' dropdown menu is open, displaying a list of available data sources: 'default', 'ruuvi', and 'Venice Weather Station'. A red arrow points from the text 'Your InfluxDB database name' to the 'ruuvi' option. The 'FROM' clause is set to 'ruuvi'. The 'SELECT' clause is set to 'time (\$__interval)'. The 'WHERE' clause is empty. The 'GROUP BY' clause is set to 'time (\$__interval)'. The 'FORMAT AS' dropdown is set to 'Time series'. The 'ALIAS BY' field is set to 'Naming pattern'. The 'Add Query' button is visible at the bottom left.

Your InfluxDB
database name

Add info to Graph: Metrics

Graph General **Metrics** Axes Legend Display Alert Time range

Data Source Venice Weather Station

FROM default WHERE +

SELECT field (value)

GROUP BY time (\$__interval)

FORMAT AS Time series

ALIAS BY Naming

Add Query

- cpu
- disk
- diskio
- kernel
- mem
- mqtt_consumer**
- processes
- swap
- system

Select
mqtt_consumer

Add info to Graph: Metrics

The screenshot shows the 'Metrics' tab of a data visualization tool. The 'Data Source' is set to 'Venice Weather Station'. The query builder has the following configuration:

- FROM:** default, mqtt_consumer, WHERE, +
- SELECT:** field (), mean (), +
- GROUP BY:** time ()
- FORMAT AS:** Time
- ALIAS BY:** Nam

A dropdown menu is open under the 'SELECT' field, listing the following variables:

- 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

A red arrow points from the text 'Select the variable you want to graph' to the dropdown menu.

Select
the variable
you want to
graph

Add info to Graph: Metrics

Graph

General Metrics Axes Legend Display Alert Time range

Data Source Venice Weather Station

FROM default mqtt_consumer WHERE +

SELECT field (value) mean () +

GROUP BY time (\$__interval) min (value) +

FORMAT AS Time series

ALIAS BY Naming pattern

Add Query

Remove
mean()

Add info to Graph: Metrics

Graph General Metrics Axes Legend Display Alert Time range

Data Source Venice Weather Station

FROM default mqtt_consumer WHERE +

SELECT field (value) +

GROUP BY time (\$__interval) fill (null) +

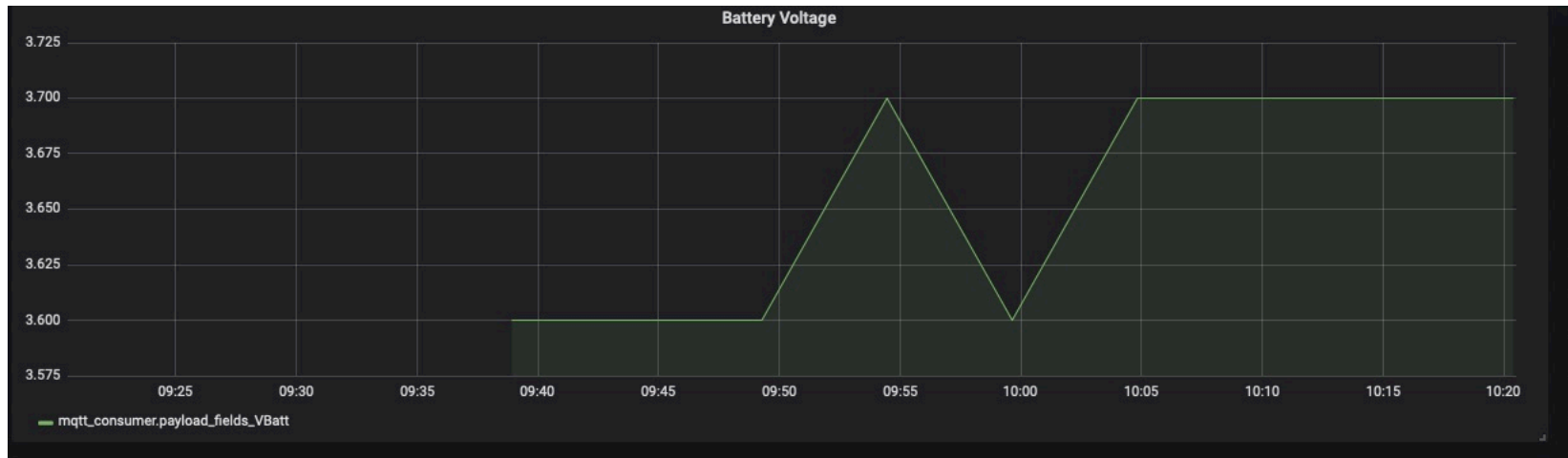
FORMAT AS Remove time series

ALIAS BY Naming pattern

B Add Query

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
```

If pip is not installed, install it first with:

```
$ sudo apt-get install pip
```

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('database_name')
```

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) Send some temperature and humidity data to InfluxDB via TTN. 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