

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

Marco Zennaro, PhD
ICTP



Labs

- 1/3 Ready to use, tested examples
- 1/3 Exercise based on the examples
- 1/3 Your imagination → create new applications

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.

Hands-on sessions

"Be excellent to each other", asking / helping is OK.

Google error messages to fix issues.

Coping blindly does not lead to new insight.

Reading other people's code helps a lot.

Our Lab equipment

Raspberry Pi

SD with latest Raspbian OS

Keyboard

Mouse

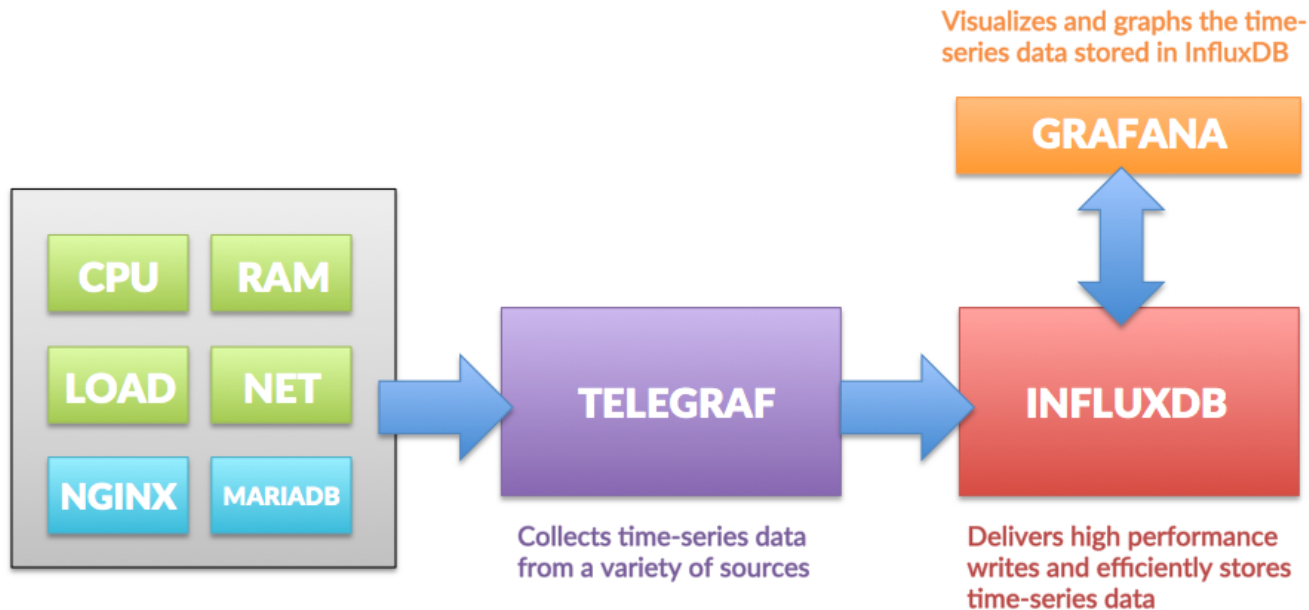
Screen

What is the TIG Stack?

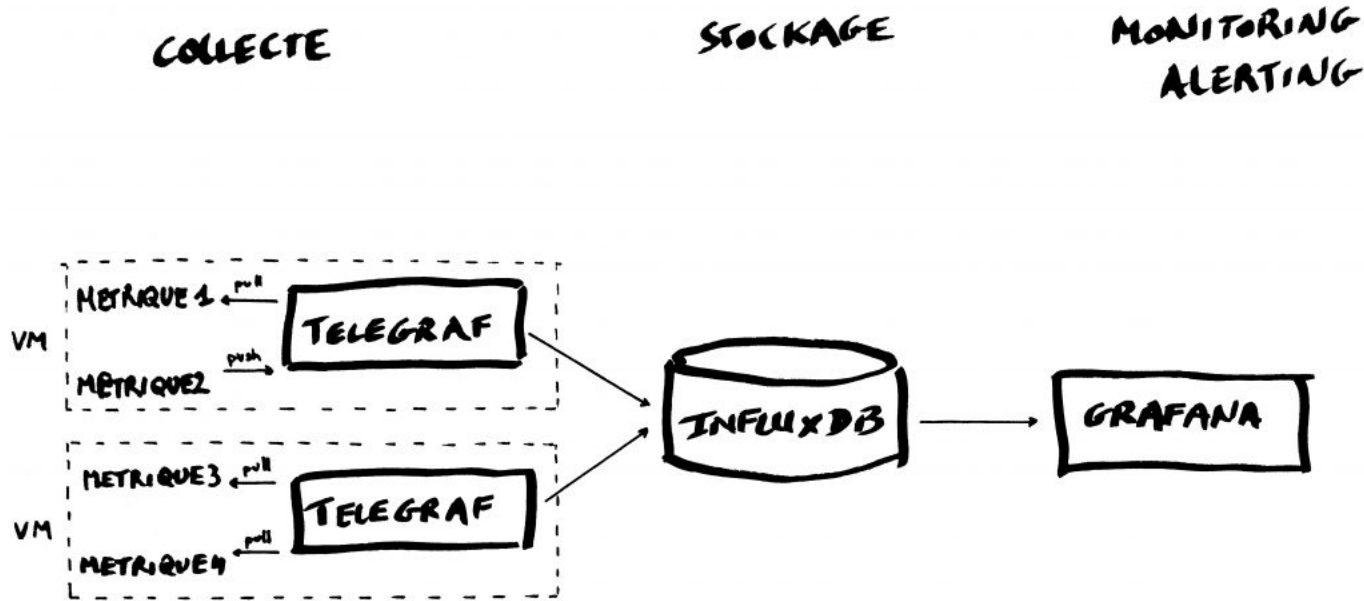
The **TIG Stack** is an acronym for a platform of open source tools built to make collection, storage, graphing, and alerting on **time series data** incredibly easy.



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 Linux machine

For Linux users, follow the instructions here (different for different distributions):

<https://docs.influxdata.com/influxdb/v1.7/introduction/installation/>

Installing TIG on a Rpi/Linux

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://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_5.2.0-beta1_armhf.deb
```

```
sudo dpkg -i grafana_5.2.0-beta1_armhf.deb
```

Installing TIG on a Linux

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

```
wget
```

```
https://dl.grafana.com/oss/release/grafana\_6.2.0\_amd64.deb
```

```
sudo dpkg -i grafana_6.2.0_amd64.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 an MQTT broker.

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://eu.thethings.network:1883"]  
qos = 0  
connection_timeout = "30s"  
topics = [ "mytopic/value/" ]  
client_id = ""  
data_format = "json"
```

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!

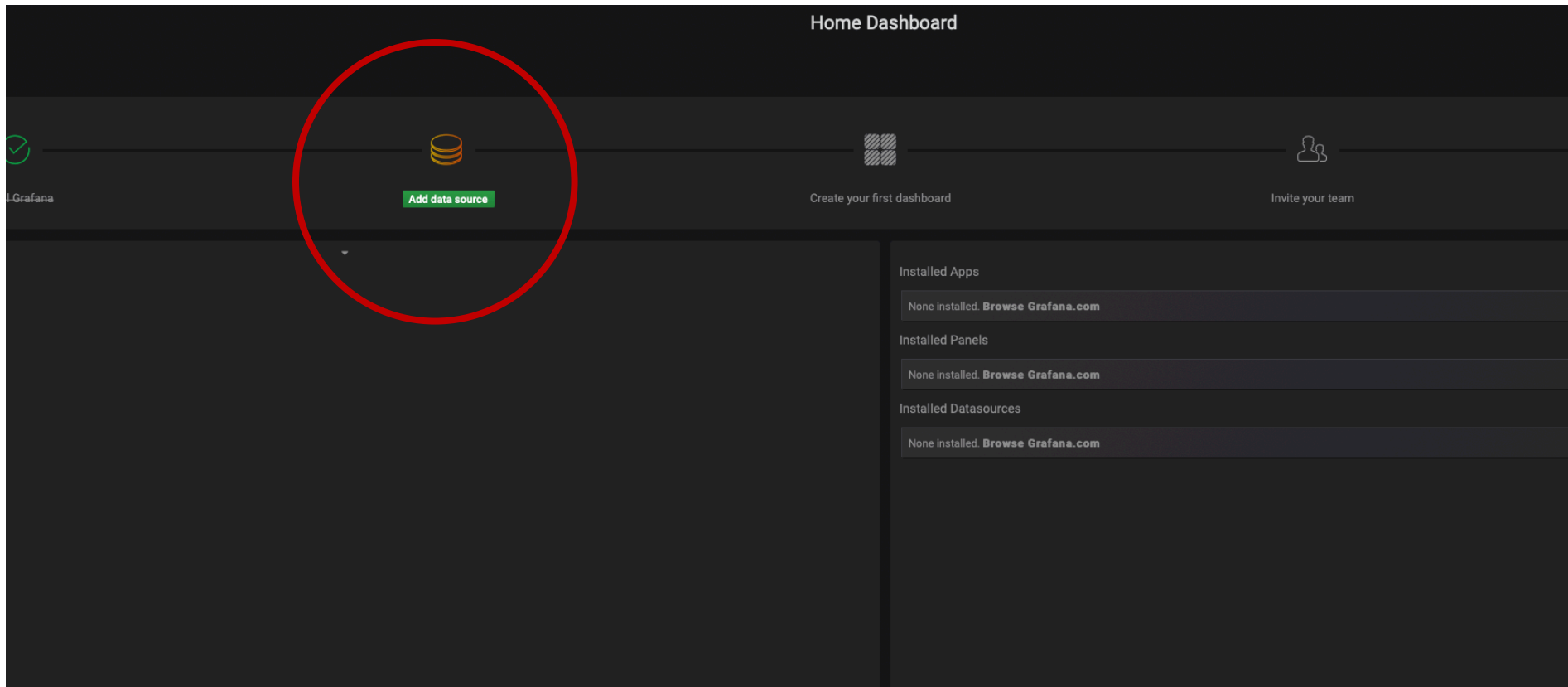
| | | | | | | |
|---------------------|----------|----------|-----------|-------------------------------------|------|---|
| 1557323990319369114 | 292 | myserver | 287744000 | 868.3 | 15 | 1 |
| 45.703526 | 13.72079 | | 1 | -112 | -5.8 | |
| 294082396 | 0 | 0 | | 1008.1 | 23.6 | |
| 45 | 0 | 2.92 | 7204 | 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 | | |

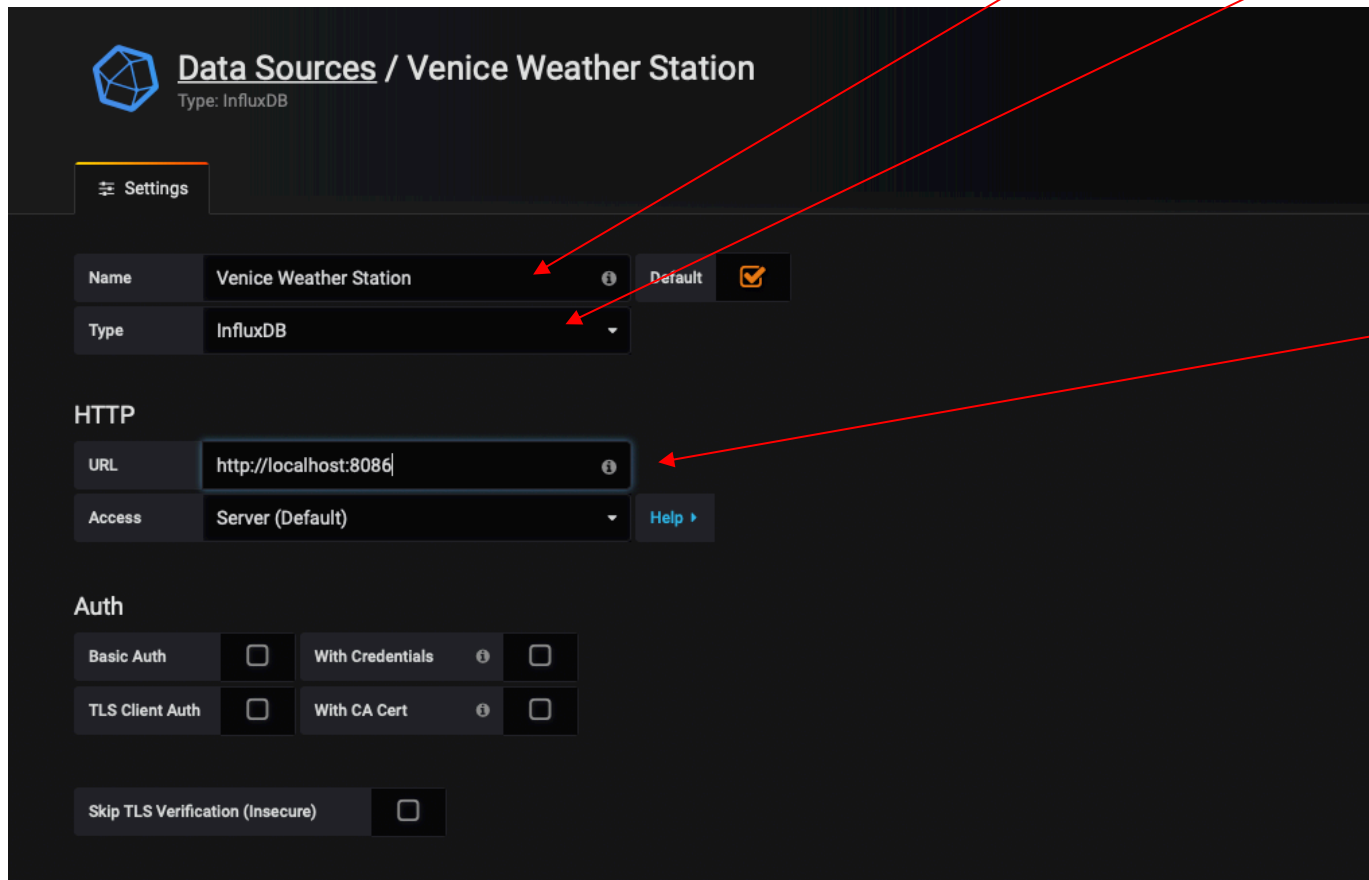
Log into Grafana

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

Add data source



Add data source 1/2



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`

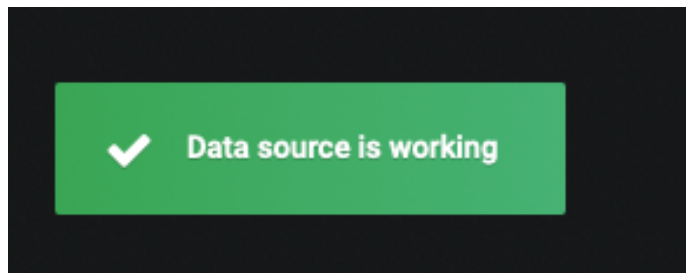
To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval

10s ⓘ

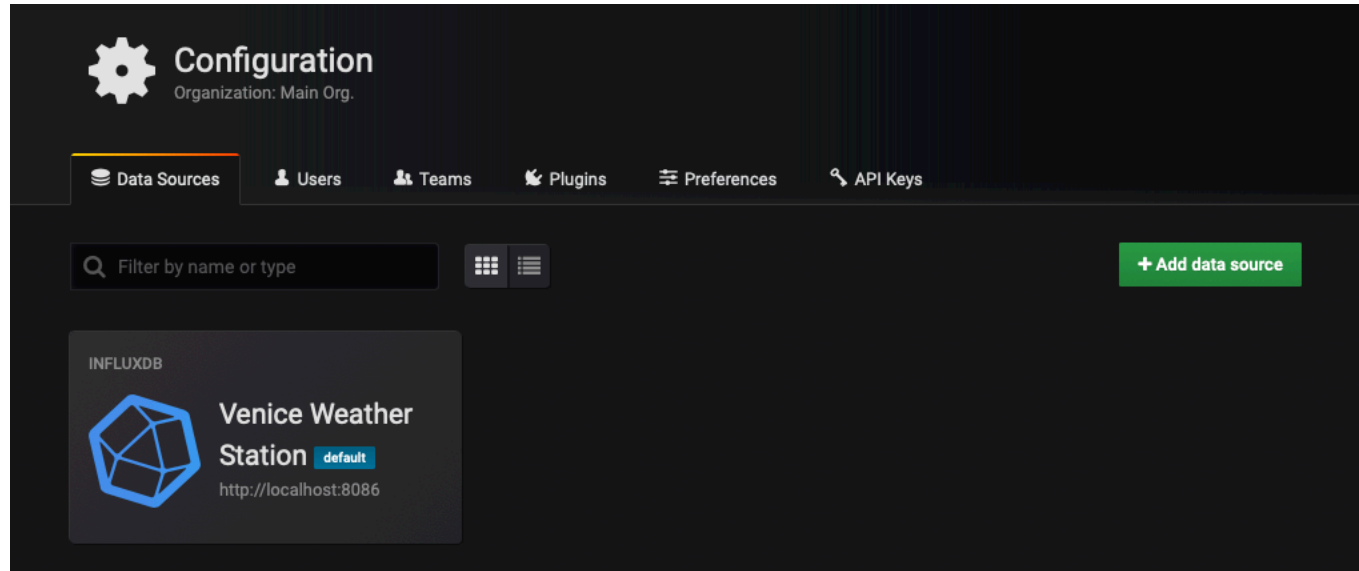
Add data source

If everything is fine you should see:

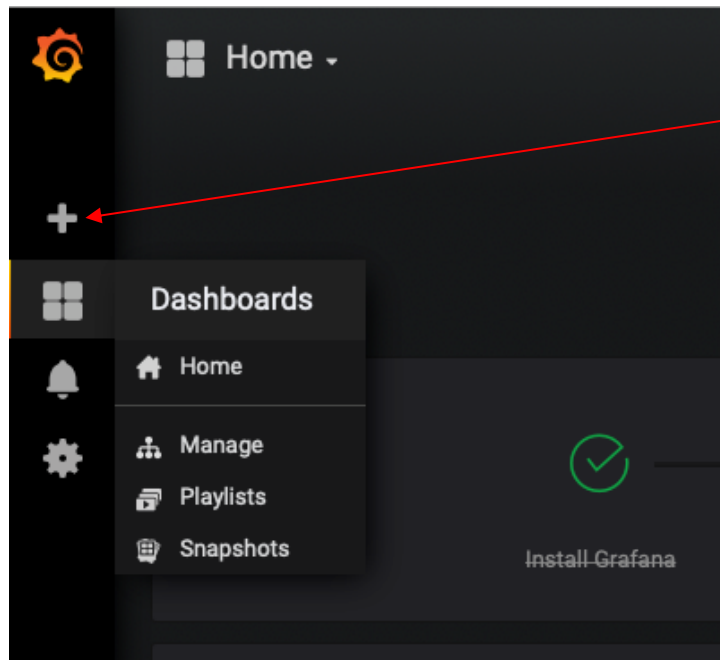


Add data source

If everything is fine you should see:

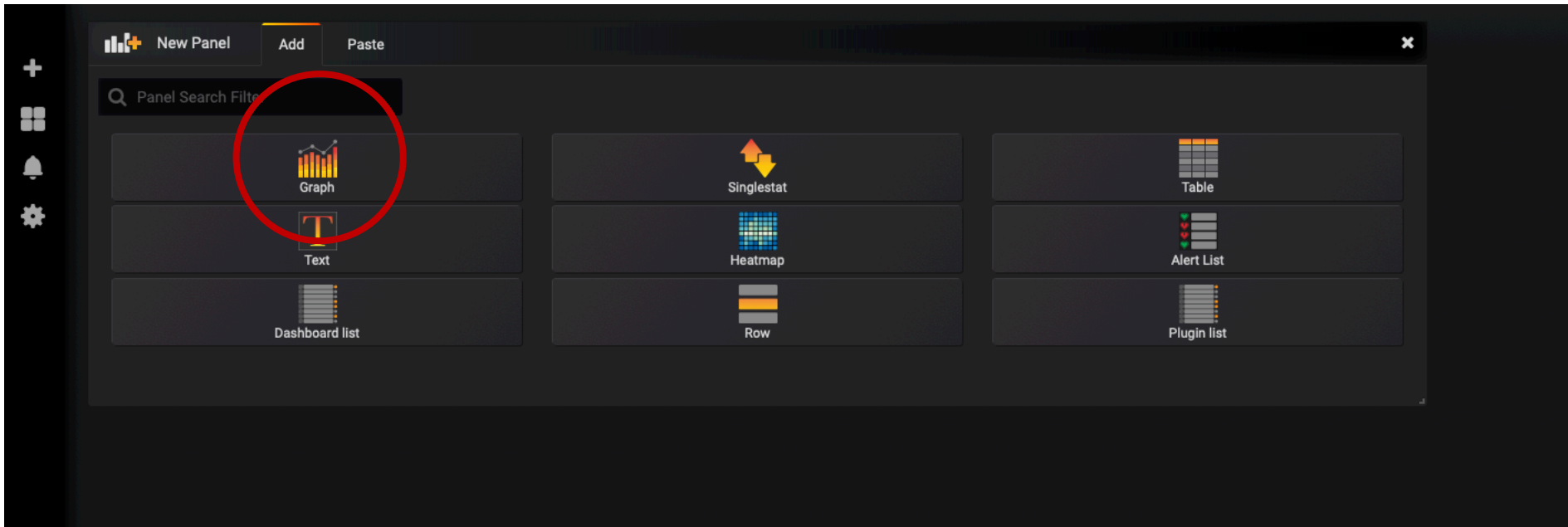


Add Dashboard



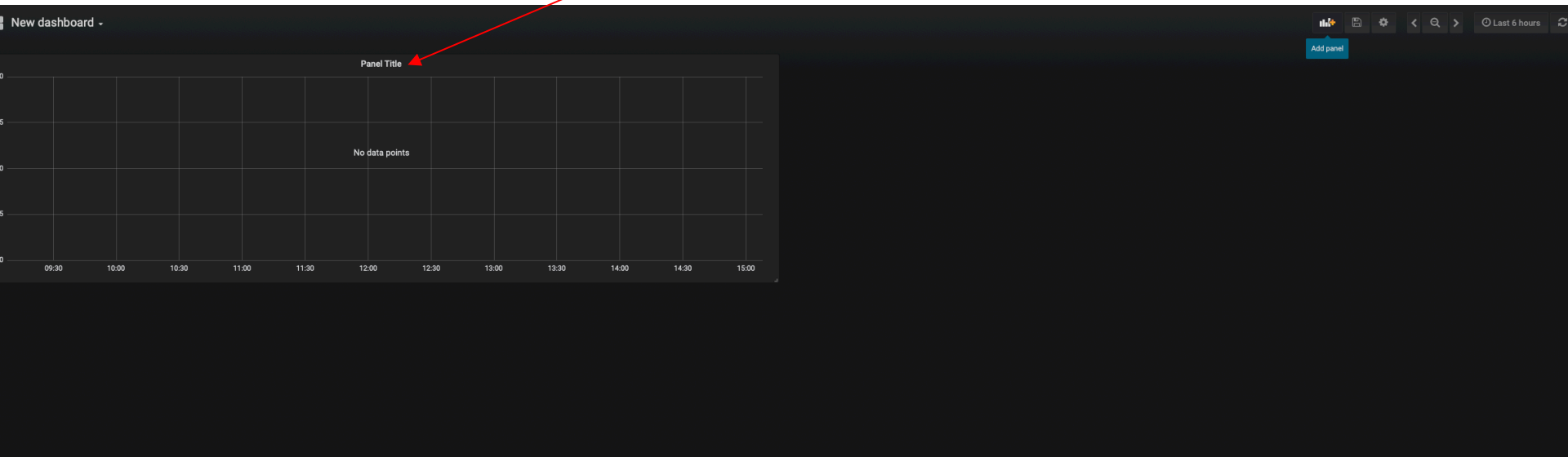
New

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. The left section, titled 'Info', contains three fields: 'Title' with the value 'Panel Title', 'Description' with the placeholder text 'Panel description, supports markdown & links', and a 'Transparent' toggle switch which is currently turned off. The right section, titled 'Repeat', contains 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'.

| Graph Configuration - General Tab | |
|-----------------------------------|--|
| Info | |
| Title | Panel Title |
| Description | Panel description, supports markdown & links |
| Transparent | <input type="checkbox"/> |
| Repeat | |
| For each value of | [Dropdown Menu] |
| Drilldown / detail link ⓘ | |
| + Add link | |

Add Title and Description

Add info to Graph: Metrics

The screenshot shows the 'Metrics' tab in the Grafana configuration interface. 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. Below the dropdown, 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

The screenshot shows the Grafana 'Metrics' configuration page. The 'Data Source' is 'Venice Weather Station'. The 'FROM' dropdown is open, displaying a list of metrics. The 'mqtt_consumer' metric is highlighted by a red arrow. The 'SELECT' dropdown is set to 'field (value)', 'GROUP BY' is set to 'time (\$__interval)', 'FORMAT AS' is set to 'Time series', and 'ALIAS BY' is set to 'Naming'. The 'Add Query' button is visible at the bottom left.

| FROM | SELECT | GROUP BY | FORMAT AS | ALIAS BY |
|---------|---------------|---------------------|-------------|----------|
| default | field (value) | time (\$__interval) | Time series | Naming |

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

Select
mqtt_consumer

Add info to Graph: Metrics

The screenshot shows the 'Metrics' tab in a graphing interface. The 'Data Source' is 'Venice Weather Station'. The 'FROM' clause is 'mqtt_consumer'. The 'SELECT' clause is 'field ()' with a dropdown menu open showing various metadata fields. A red arrow points from the text 'Select the variable you want to graph' to the dropdown menu.

| Tab | Label | Value |
|-------------|----------|------------------------|
| Graph | General | Metrics |
| Data Source | | Venice Weather Station |
| FROM | default | mqtt_consumer |
| WHERE | | |
| SELECT | field () | mean () |
| GROUP BY | time () | |
| FORMAT AS | Time | |
| ALIAS BY | Name | |
| Buttons | | Add Query |

Dropdown menu options for 'field ()':

- 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

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) +

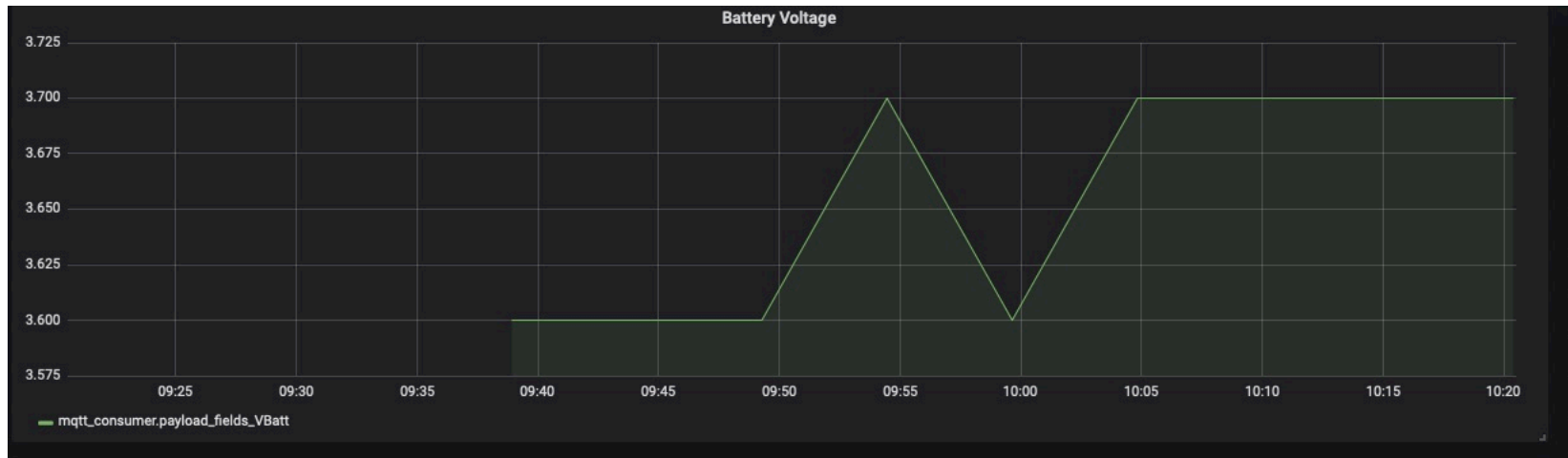
FORMAT AS Remove time series

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 Graphana!

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) Send some temperature data to InfluxDB via MQTT.
- 2) Save the data as csv (comma separated values) using Python and InfluxDB.
- 3) Produce a graph of the last 20 temperature measurements using Python and InfluxDB.

Summary

We learned how to install Telegraf, InfluxDB and Graphana.

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

We learned how to interact with InfluxDB using Python.



Feedback?

Email mzennaro@ictp.it