

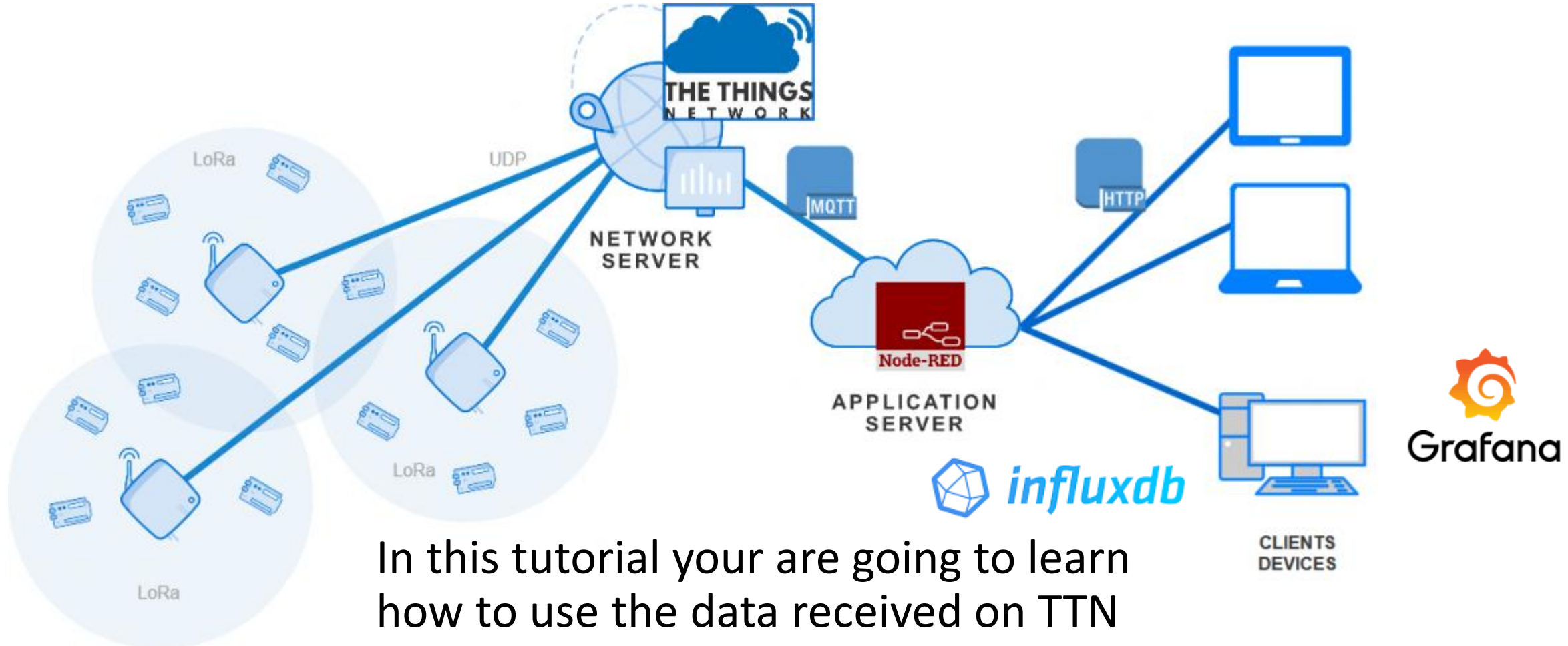
IoT LoRa application service Tutorial

F. Ferrero

V.1.1



Node Red – InFluxDB - GRAFANA

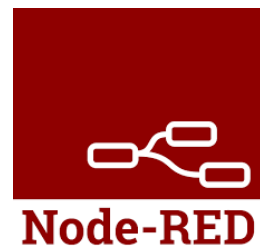


Outline

1/ Definition

2/ Tutorial

Node Red



- Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.
- It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.
- Built on Node.js
 - The light-weight runtime is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

InfluxDB



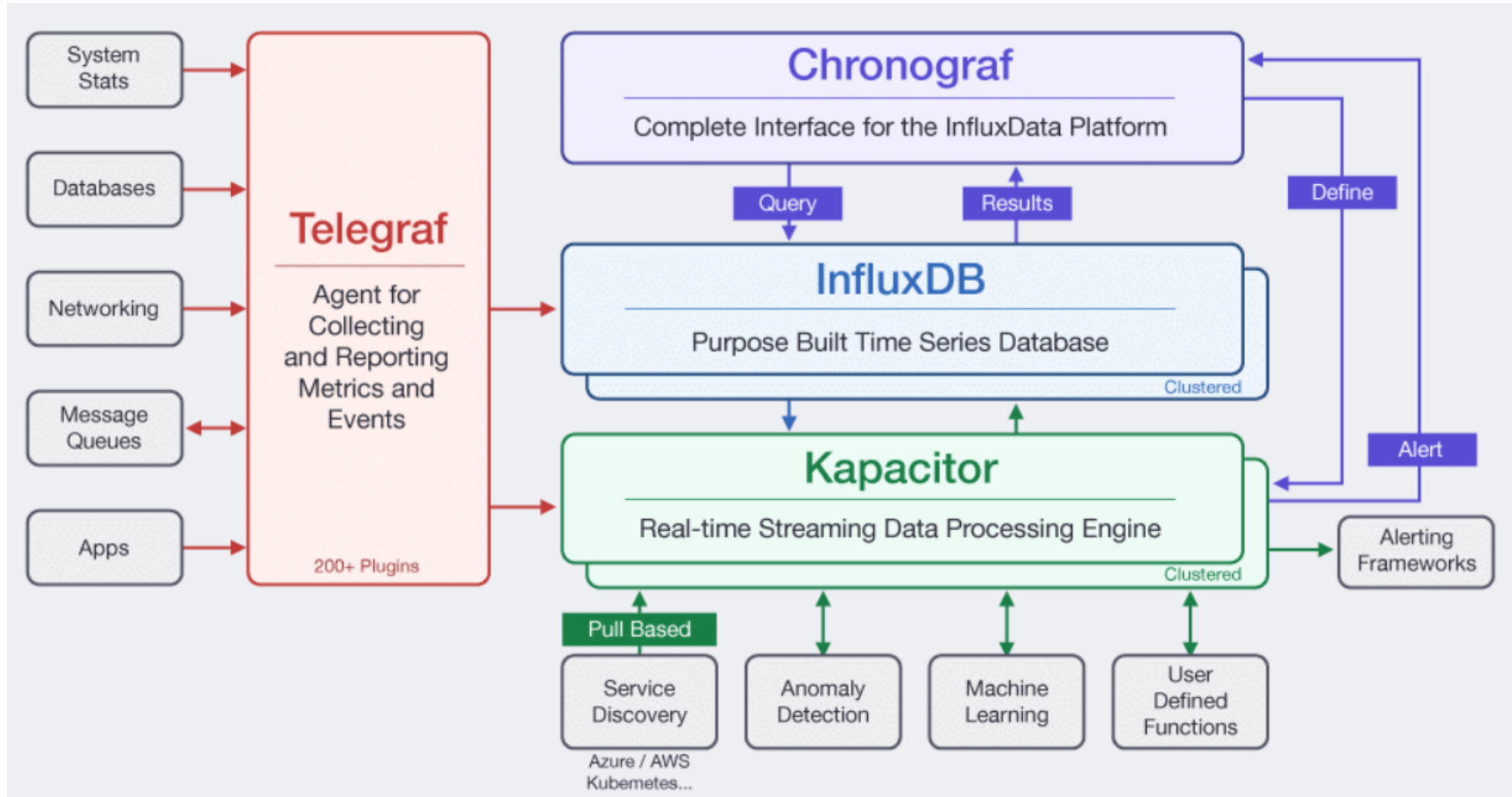
InfluxDB

- InfluxDB is an open source distributed time series database developed by InfluxData. The main advantage of InfluxDB is its capacity to aggregate values in time buckets on-the-fly without any manual intervention.
- InfluxDB can be accessed by software like Grafana, which is a powerful front-end tool providing visualisation features for time series data. Each point consists of varied key-value pairs called fieldset and timestamp. Points are indexed by their time and tagset. InfluxDB stores data via HTTP, TCP and UDP.

Features

- Purely written in the Go programming language and facilitates compilation into a single binary with no external dependencies.
- High performance customised data store written especially for time series data. The TSM engine of InfluxDB allows efficient and high speed data storage and compression.
- In-built Web front-end tool for database and user administration.
- Competent in merging multiple series together.
- **Official website:** <https://www.influxdata.com/>

InfluxDB



Additionally to the database, influx data provide interesting applications

Grafana



- Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data driven culture.
- Grafana includes a built in Graphite query parser that takes writing graphite metric expressions to a whole new level. Expressions are easier to read and faster to edit than ever.
- Click on any metric segment to change it
- Quickly add functions (search, typeahead)
- Click on a function parameter to change it
- Move function order to the left or right
- Direct link to Graphite function documentation
- Rich templating support

Outline

1/ Definition

2/ Tutorial

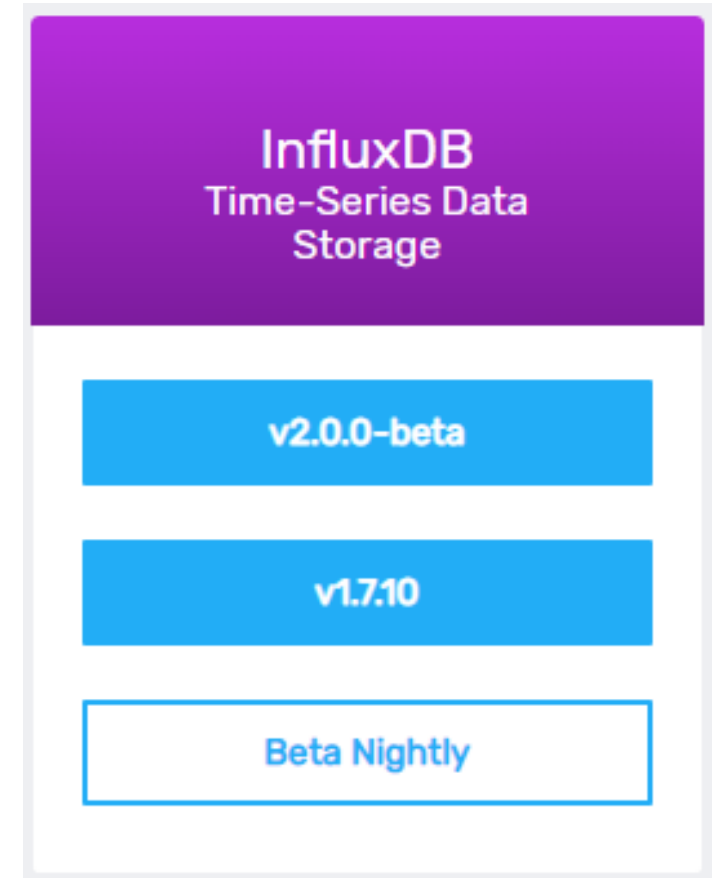
Node-Red

- Install git : <https://git-scm.com/downloads>
- Then install Node Red, follow this tutorial :
<https://www.thethingsnetwork.org/docs/applications/nodered/quick-start.html>
- Install package in Node Red :
 - node-red-contrib-ttn
 - node-red-contrib-influxdb

Influx DB Data base

- Install InfluxDB v1.7.10

<https://portal.influxdata.com/downloads>



Grafana

- Install Grafana

<https://grafana.com/get>



Connecting to TTN

- Start NODE.js command prompt
- Run : node-red
- Open your web browser and go to <http://127.0.0.1:1880>

```
node-red
Your environment has been set up for using Node.js 10.13.0 (x64) and npm.

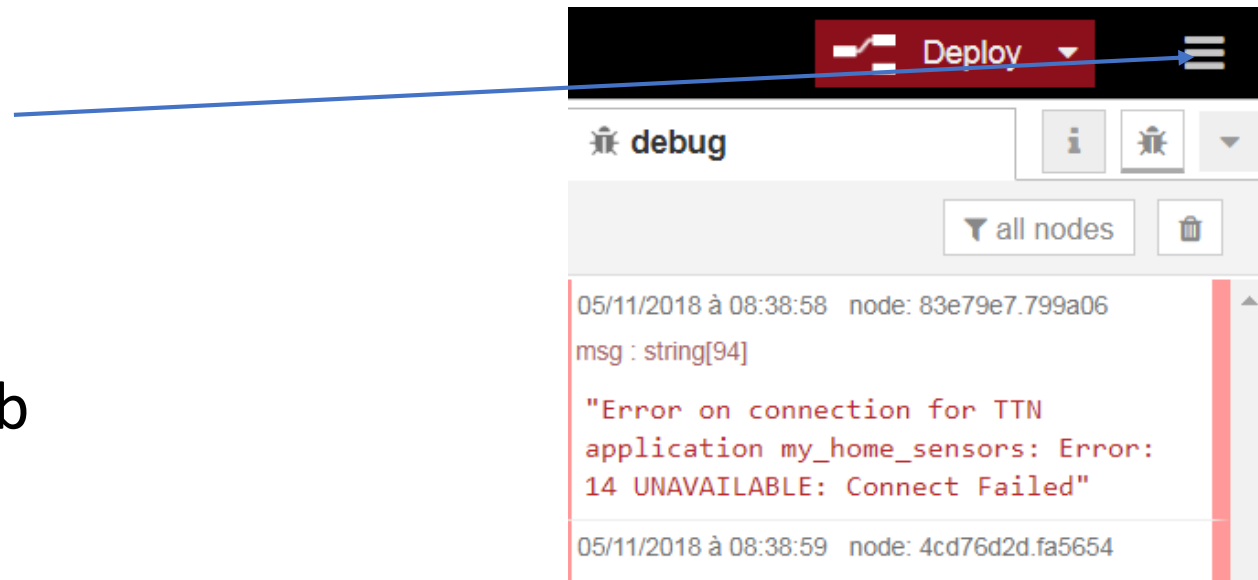
C:\Users\hp_sim>node-red
5 Nov 06:02:48 - [info]

Welcome to Node-RED
=====

5 Nov 06:02:48 - [info] Node-RED version: v0.19.5
5 Nov 06:02:48 - [info] Node.js version: v10.13.0
5 Nov 06:02:48 - [info] Windows_NT 6.1.7601 x64 LE
5 Nov 06:02:50 - [info] Loading palette nodes
5 Nov 06:02:52 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
5 Nov 06:02:52 - [warn] -----
5 Nov 06:02:52 - [warn] [node-red/tail] Not currently supported on Windows.
5 Nov 06:02:52 - [warn] -----
5 Nov 06:02:52 - [info] Settings file : \Users\hp_sim\.node-red\settings.js
5 Nov 06:02:52 - [info] Context store : 'default' [module=memory]
5 Nov 06:02:52 - [info] User directory : \Users\hp_sim\.node-red
5 Nov 06:02:52 - [warn] Projects disabled : editorTheme.projects.enabled=false
5 Nov 06:02:52 - [info] Flows file : \Users\hp_sim\.node-red\flows_hp_sim-HP
.json
5 Nov 06:02:52 - [warn] -----
```

- On the editor, click here
- And go to palette editor
Install :

- node-red-contrib-ttn
- node-red-contrib-influxdb



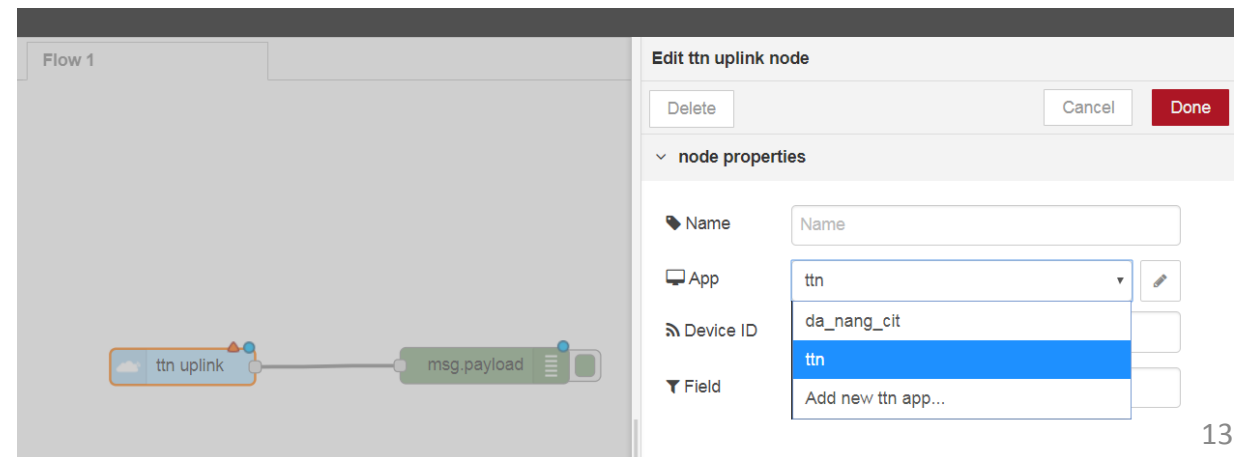
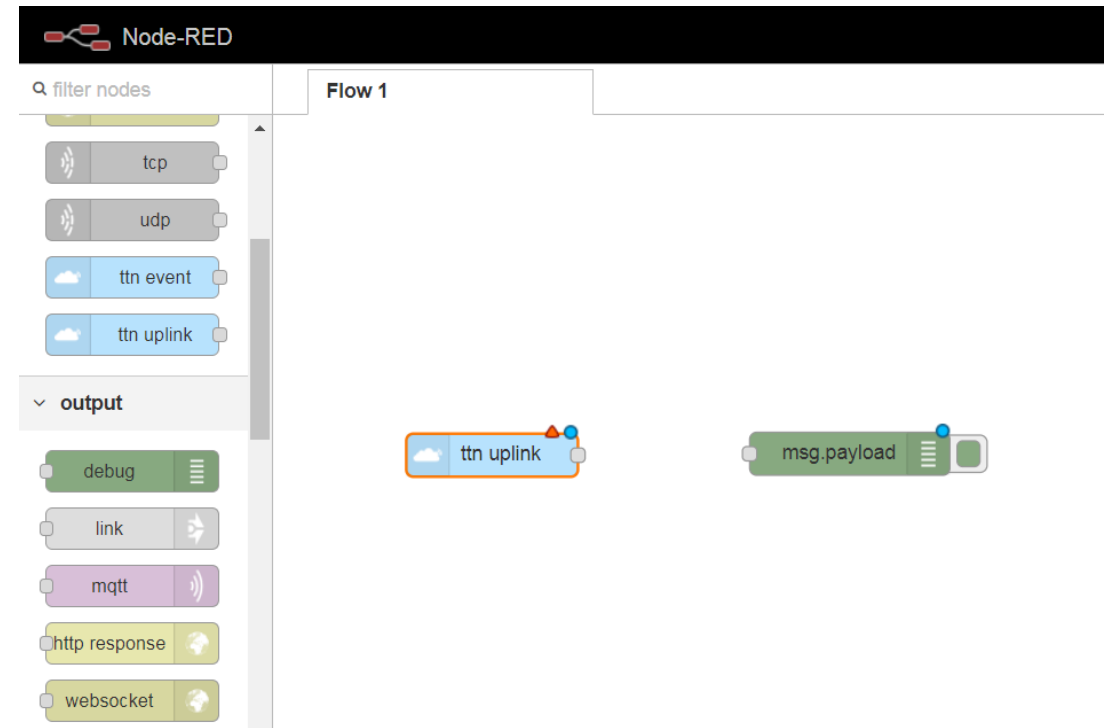
Connecting to TTN

- You have the graphical Node-red editor
- Add ttn uplink and a debug output
- Edit TTN uplink
- Choose « Add new ttn app ... » in App and click on edit

App ID

Access Key

Discovery address



Connecting to TTN

- You need :

- App ID :
- Access Key :
- Discovery adress :

discovery.thethingsnetwork.org:1900

- Go to you application in TTN
- Copy paste the Application ID and Access Key

Edit ttn uplink node > Edit ttn app node

Delete Cancel Update

App ID 70B3D57ED0013EAD

Access Key

Discovery address discovery.thethingsnetwork.org:1900

Applications > my_home_sensors

Overview Devices Payload Formats Integrations Data Settings

APPLICATION OVERVIEW

Application ID my_home_sensors documentation

Description My Home sensors

Created 7 months ago

Handler ttn-handler-eu (current handler)

APPLICATION EUIs manage euis

<> 70 B3 D5 7E D0 00 81 B4

DEVICES register device manage devices

11 registered devices

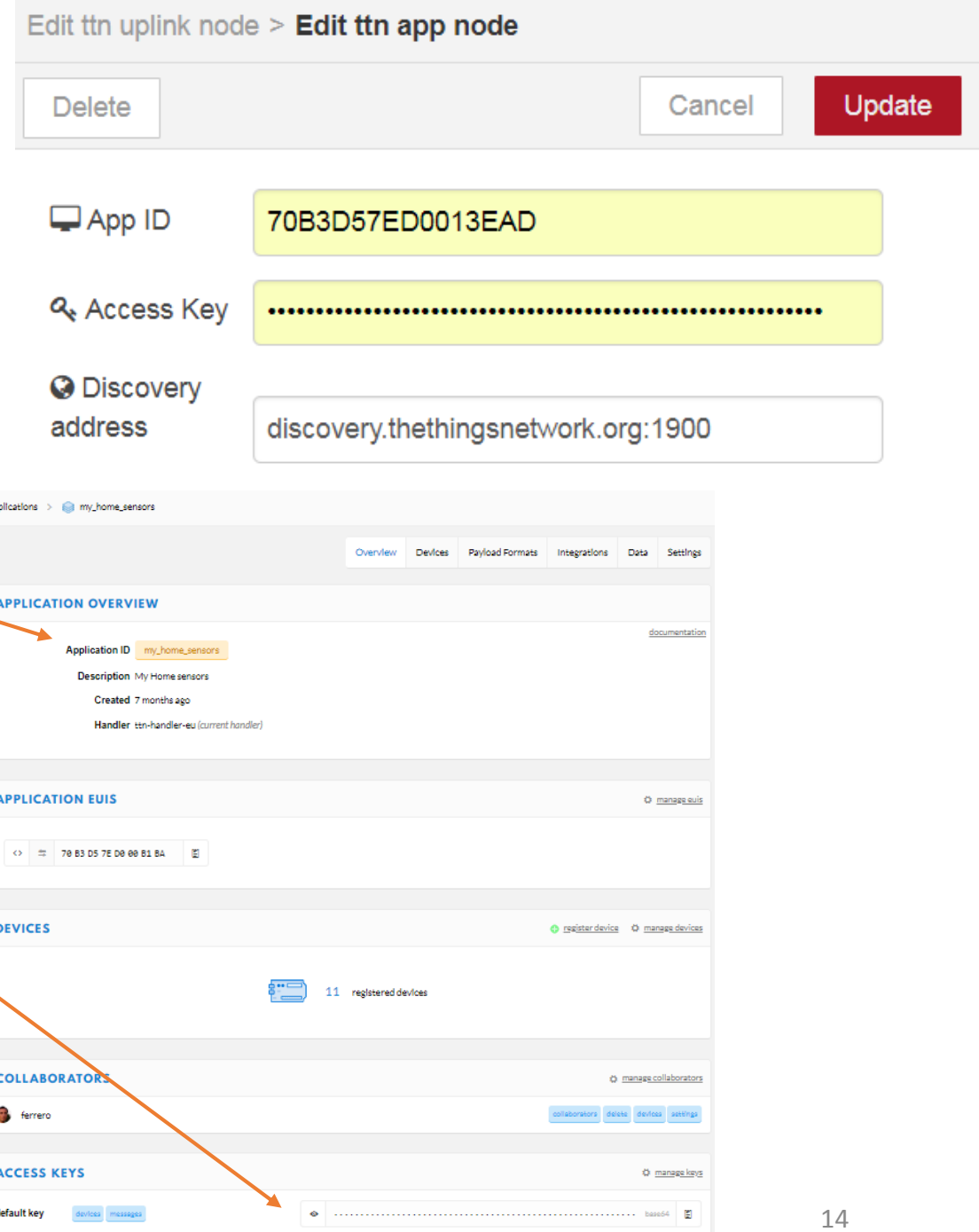
COLLABORATORS manage collaborators

ferro collaborators delete devices settings

ACCESS KEYS manage keys

default key devices messages

base64



Connecting to TTN

- Click on Deploy
- You uplink TTN should be connected
- Click on debug window
- You will receive the packet of the application
- If you want to filter only your device, add your device ID
- Click here :

The screenshot displays the TTN (The Things Network) console interface. At the top, a black bar contains a 'Deploy' button. Below this, a 'debug' window is open, showing a message from node 'ba3a9ad9.e50308' with a payload object: `{ analog_in_3: 5.14, digital_out_4: 0, relative_humidity_2: 79, temperature_1: 21.2 }`. A blue arrow points from the 'debug' window to the 'Click here' instruction in the list.

In the center, a diagram shows a node labeled 'my_home_sensors' connected to a 'msg.payload' block. A green 'connected' status indicator is shown below the node.

At the bottom left, the 'DEVICE OVERVIEW' section shows the following details:

- Application ID: my_home_sensors
- Device ID: 50ff1a0000010004
- Description: Light test
- Activation Method: OTAA

A blue arrow points from the 'Device ID' field to the 'Click here' instruction in the list.

At the bottom right, the 'Edit ttn uplink node' dialog is open, showing the following fields:

- Name: my_home_sensors
- App: my_home_sensors
- Device ID: 50ff1a0000010004
- Field: (empty)

A blue arrow points from the 'Device ID' field in the dialog to the 'Device ID' field in the 'DEVICE OVERVIEW' section.

Connecting to TTN

- Click here :
- Choose « complete msg object »
- And Deploy
- You have now more information of your uplink

The screenshot displays the TTN console interface. At the top, a black bar contains a 'Deploy' button. Below this, a node configuration is shown with two blocks: 'my_home_sensors' (blue) and 'msg.payload' (green). A blue arrow points from the 'msg.payload' block to the 'debug' window on the right. The 'debug' window shows the following output:

```
05/11/2018 à 06:17:40 node: ba3a9ad9.e50308
msg.payload : Object
  ▶ { analog_in_3: 5.14, digital_out_4: 0,
    relative_humidity_2: 79, temperature_1: 21.2 }
```

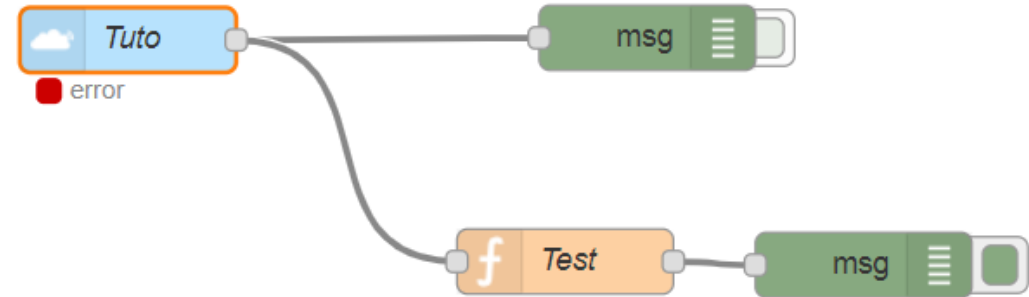
Below the node configuration, a code editor shows the following JSON payload:

```
{
  "app_id": "my_home_sensors",
  "dev_id": "my_cellar_sensor",
  "hardware_serial": "50FF1A0000010003",
  "port": 1,
  "counter": 32781,
  "payload_raw": "buffer[14]",
  "payload_fields": "object",
  "metadata": "object",
  "payload": "object",
  "_msgid": "fc884851.257408"
}
```

At the bottom right, the 'Edit debug node' dialog is open, showing the 'Output' dropdown set to 'complete msg object', the 'To' dropdown set to 'debug window', and the 'Name' field empty.

Connecting to TTN

- If you want to extract only 1 data,
- As an exemple the RSSI (received signal Strength indicator
- Use a function to extract the wanted data



```
var gateways = msg.metadata.gateways;

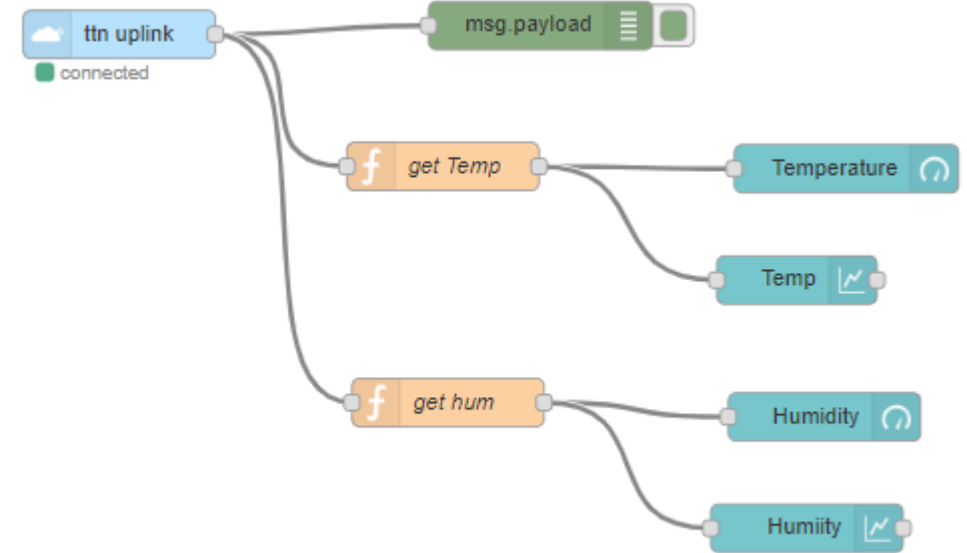
return {
  // Some fields from the metadata freq:
  msg.metadata.frequency,
  cr: msg.metadata.cr,
  dr: msg.metadata.dr,

  // Combine RSSI and SNR of all gateways into two arrays:
  rssi: gateways.map(gw => gw.rssi),
  snr: gateways.map(gw => gw.snr),

};
```

Add a Dashboard

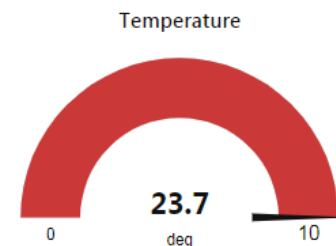
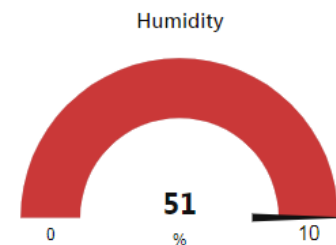
- Go to Manage Palette, select Install
- Install : node-red-dashboard
- Add a function to extract sensor values (Temp, Hum, luminosity...)
- Add Gauge and Graph for Dashboard section
- Go to : <http://127.0.0.1:1880/ui/>



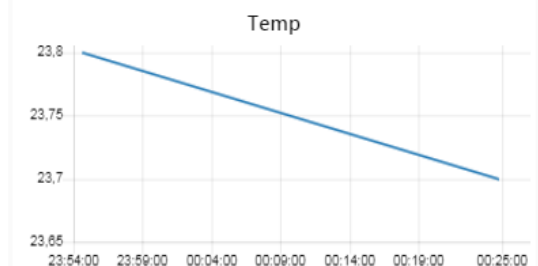
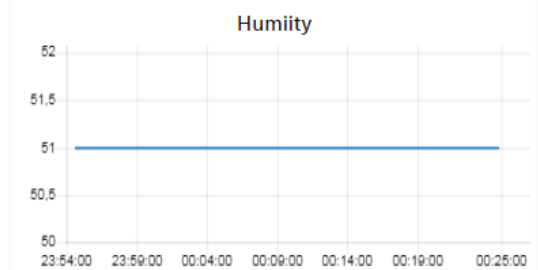
Function

```
1 var temp = {}  
2 temp.payload = msg.payload.temperature_4;  
3 return temp;
```

test



Test



InfluxDB

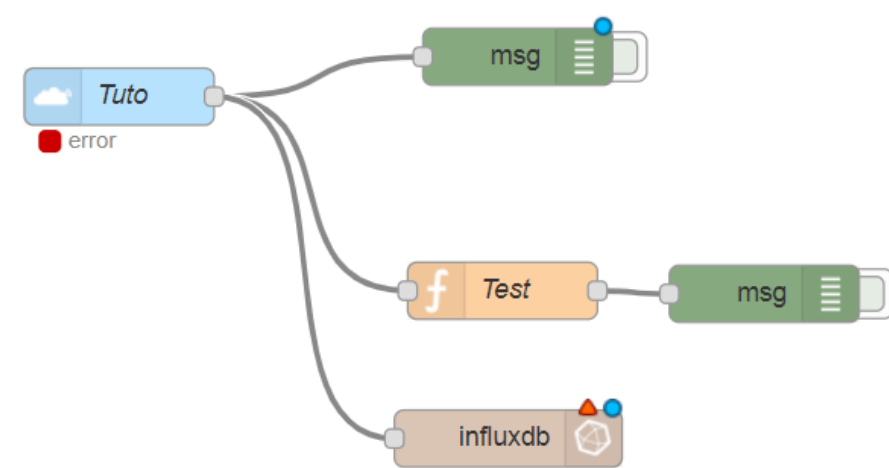
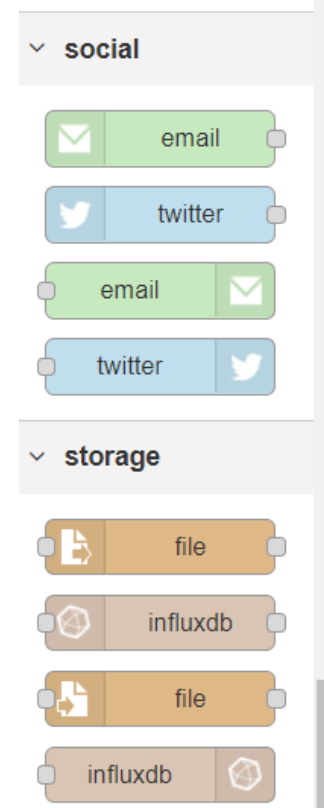
- Run « influxd.exe », it will start the database
- Run « influx.exe », it will open a shell
- Write : « CREATE DATABASE mySensor »
- Then write : « SHOW DATABASE »

Your database is created

```
> CREATE DATABASE mySensor
> SHOW DATABASES
name: databases
name
----
_internal
tuto
mySensor
>
```

InfluxDB – Node Red

- How to store data in your database ?
- Add an influxdb storage and connect it to your uplink
- Define a server, just add the Database name : mySensor
- Add
- In measurement field, add a name for your device : device1
- Go to InfluxDB shell
- Run : SHOW SERIES ON mySensor



Edit influxdb out node > Add new influxdb config node

Cancel Add

Host 127.0.0.1 Port 8086

Database mySensor

Username

Password

☐ Enable secure (SSL/TLS) connection

Name Name

Edit influxdb out node

Delete Cancel Done

node properties

Server Add new influxdb...

Measurement

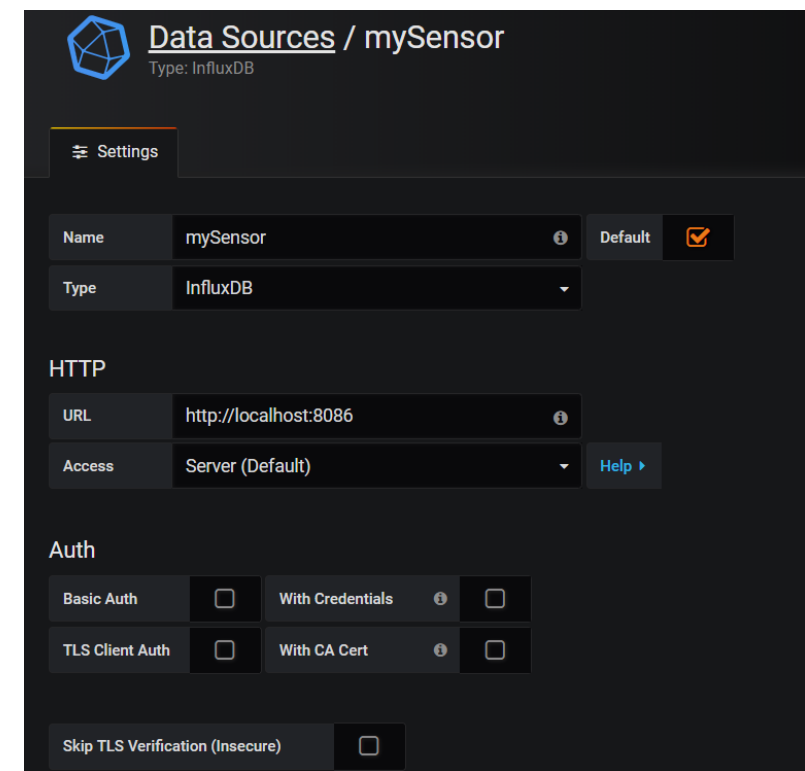
☐ Advanced Query Options

Name Name

Tip: If no measurement is specified, ensure msg.measurement contains the measurement name

Grafana

- Go to your unzip Grafana directory/bin
- Start grafana-server.exe
- Go to : `http://127.0.0.1:3000`
- User name and password is : admin
- Provide a new password
- Click « Add data source »
- Add a name
- Choose InfluxDB type
- Define Database name « mySensor »
- Click on Save and Test



Data Sources / mySensor
Type: InfluxDB

Settings

Name	mySensor	Default	<input checked="" type="checkbox"/>
Type	InfluxDB		

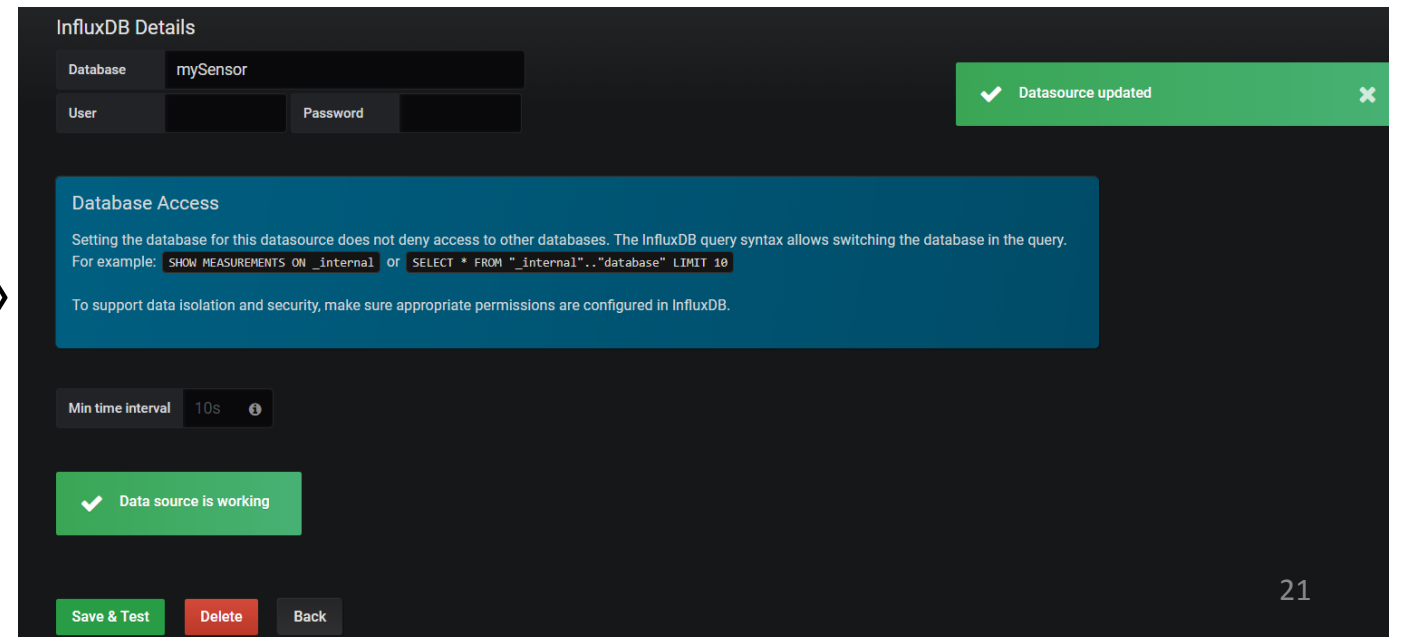
HTTP

URL	http://localhost:8086	
Access	Server (Default)	Help

Auth

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)



InfluxDB Details

Database	mySensor		
User		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

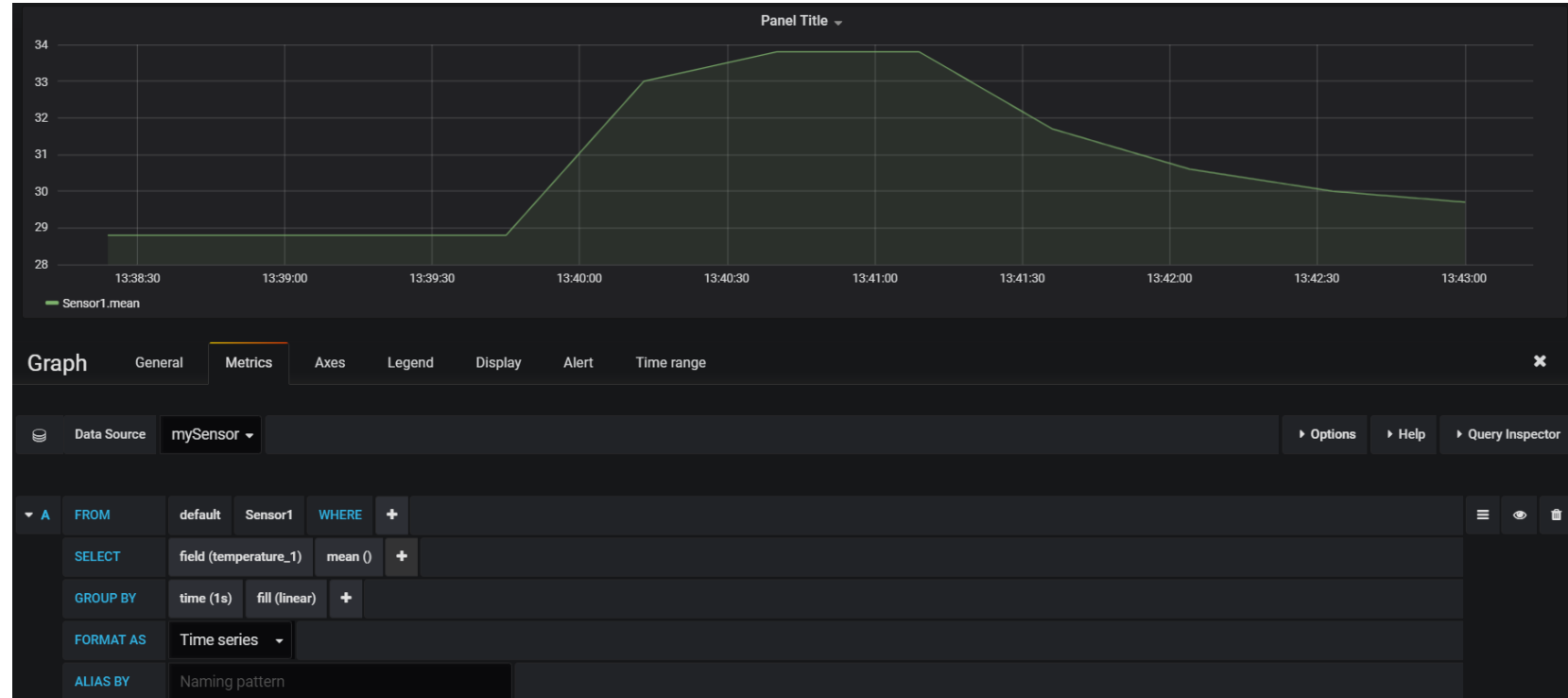
✓ Data source is working

✓ Datasource updated

[Save & Test](#) [Delete](#) [Back](#)

Grafana

- Create a new dashboard
 - Click on Graph
 - Panel Title / Edit
 - Select your data source and measurement, field temperature, time 1s, fill linear
 - Change to the last 5mn
 - Put your finger on the sensor
 - Look at your curve
-
- To speed your measurement :
In Arduino code, change Tx interval to be 20s for SF7



```
case DR_SF7: debugPrintLn(F("Datarate: SF7"));  
             TX_INTERVAL = 20;
```

Good luck for you projects !

