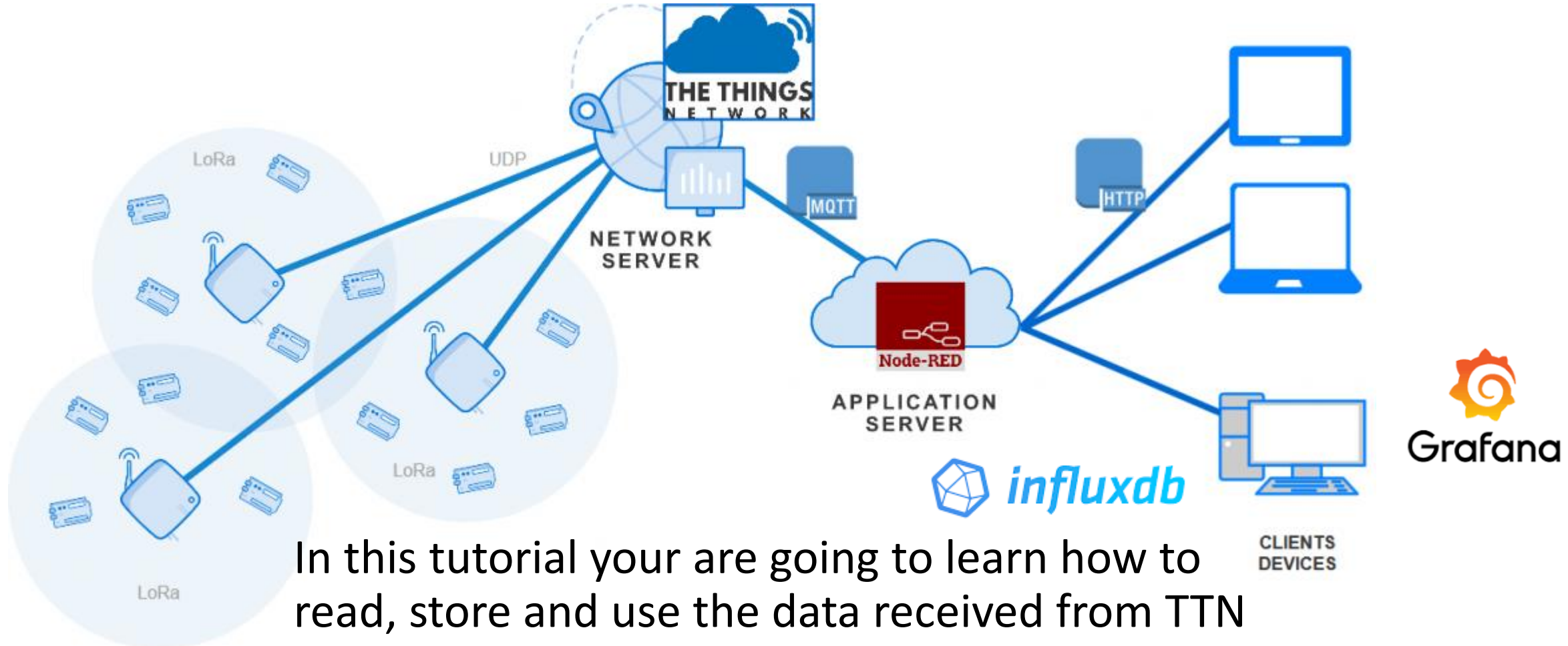


# IoT LoRa application service Tutorial

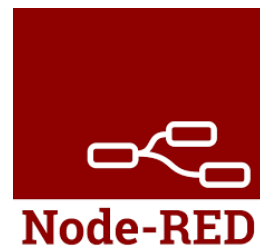
F. Ferrero



# Node Red – InFluxDB - GRAFANA



# 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



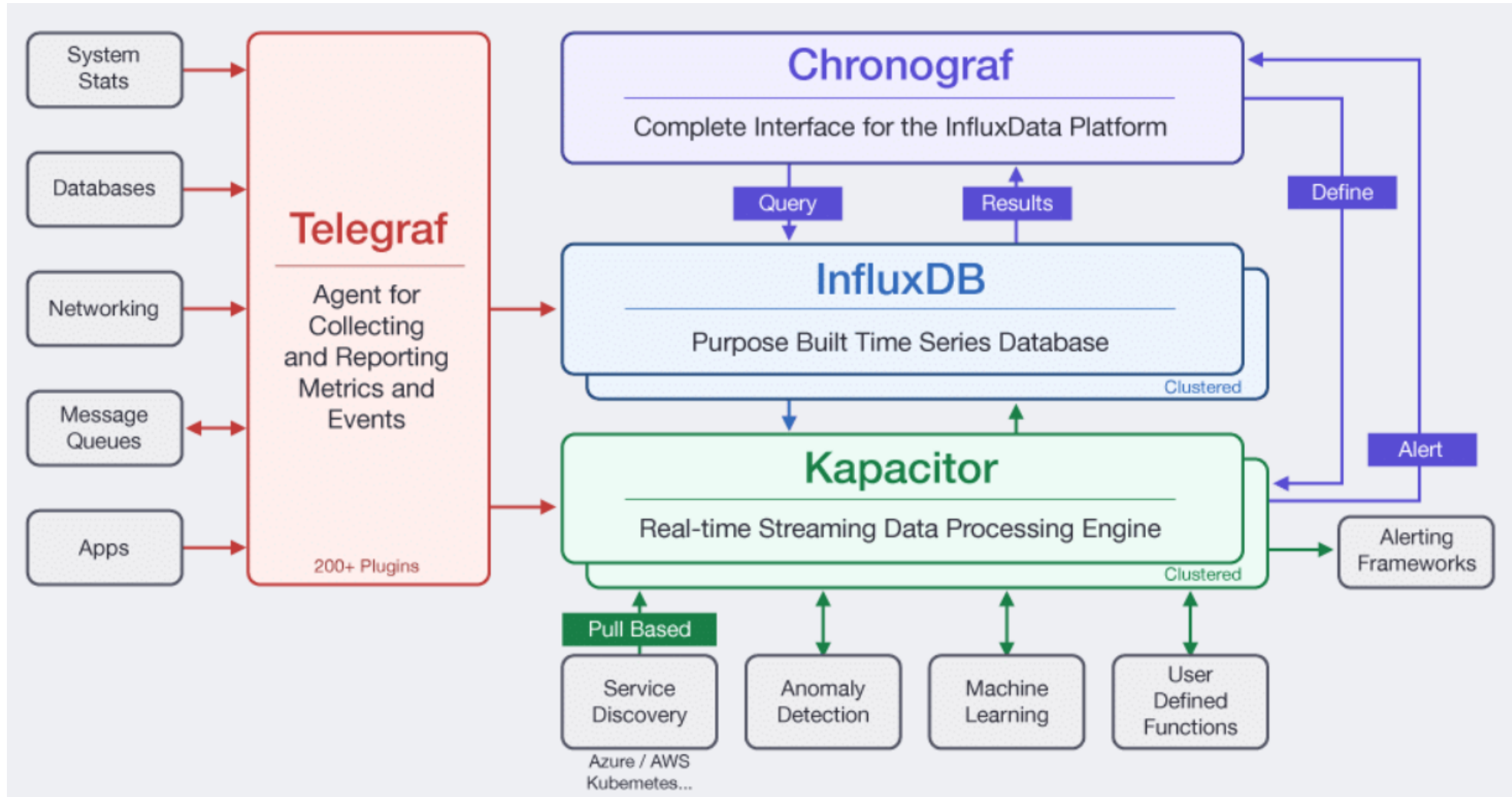
## InfluxDB

- 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

# Node-Red

- Please install Node Red and TTn lib

<https://www.thethingsnetwork.org/docs/applications/nodered/quick-start.html>

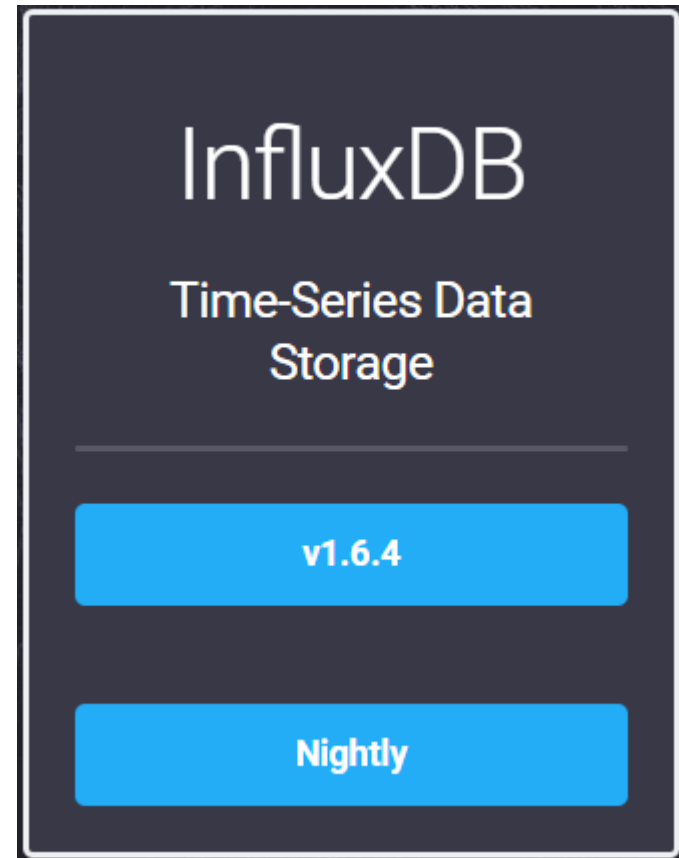
If git is not installed on your PC : <https://git-scm.com/downloads>

You will also need to install :  
node-red-contrib-influxdb

# Influx DB Data base

- Please install InfluxDB v1.6.4

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





# Grafana

- Please 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.

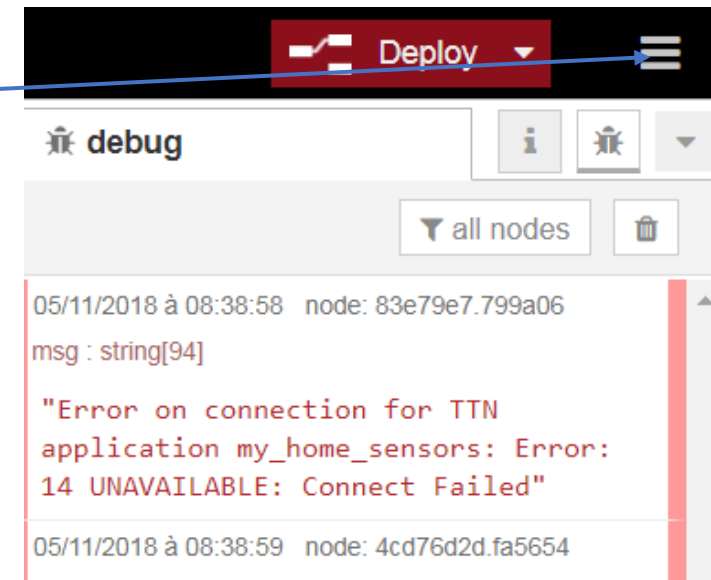
C:\Users\hp_sim>node-red
5 Nov 06:02:48 - [info]

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-influxdb



# Connecting to TTN

- You have the graphical Node-red editor
- Add mqtt in node
- Edit mqtt
- Choose « Add new mqtt-broker ... » in App and click on edit

Name

**Connection** | Security | Messages

Server  Port

☐ Enable secure (SSL/TLS) connection

Client ID

☒ Keep alive time (s)  ☒ Use clean session

☐ Use legacy MQTT 3.1 support

The image shows the Node-RED graphical editor interface. On the left, a sidebar contains a palette of nodes. Under the 'network' category, the 'mqtt in' node is highlighted. In the main workspace, an 'mqtt' node is placed on a grid. A context menu is open over the 'mqtt' node, showing options 'Delete', 'Cancel', and 'Done'. Below the workspace, the 'Edit mqtt in node' configuration panel is visible. It has tabs for 'Properties', 'Messages', and 'Security'. The 'Properties' tab is active, showing fields for 'Server' (a dropdown menu with 'Add new mqtt-broker...' selected), 'Topic' (a text input with 'Topic'), 'QoS' (a dropdown menu with '2' selected), 'Output' (a dropdown menu with 'auto-detect (string or buffer)' selected), and 'Name' (a text input with 'Name').

# Connecting to TTN

- Update security and topic:  
discovery.thethingsnetwork.org:1900

- Go to you application in TTN
- Copy past the User name and keys

v3/uca-project@ttn/devices/**device\_name**/up

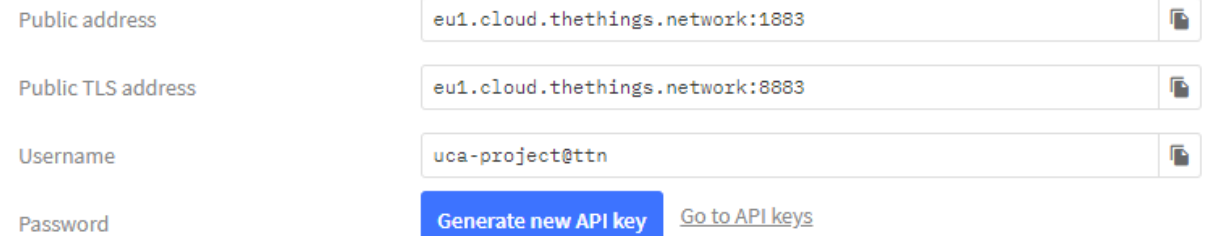


A screenshot of the TTN application configuration page. At the top, there is a 'Name' field with a small icon to its left. Below this, there are three tabs: 'Connection', 'Security' (which is selected and highlighted), and 'Messages'. Under the 'Security' tab, there are two fields: 'Username' and 'Password'. The 'Password' field contains a series of dots, indicating it is masked.

## MQTT

The Application Server exposes an MQTT server to work with streaming events. In order to use the MQTT server you need to create a new API key, which will function as connection password. You can also use an existing API key, as long as it has the necessary rights granted. Use the connection information below to connect.

### Connection credentials



A screenshot of the 'Connection credentials' section. It contains four fields: 'Public address' with the value 'eu1.cloud.thethings.network:1883', 'Public TLS address' with the value 'eu1.cloud.thethings.network:8883', 'Username' with the value 'uca-project@ttn', and 'Password'. The 'Password' field is empty, but there is a blue button labeled 'Generate new API key' and a link labeled 'Go to API keys' next to it. Each of the first three fields has a small icon to its right.

# 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. Below the debug window, a 'my\_home\_sensors' node is shown with a 'connected' status. A blue arrow points from the 'Click here' instruction to the 'Device ID' field in the 'Edit ttn uplink node' dialog. The 'Edit ttn uplink node' dialog shows the 'node properties' section with the following fields: Name (my\_home\_sensors), App (my\_home\_sensors), Device ID (50ff1a0000010004), and Field (empty). A blue arrow points from the 'Device ID' field in the dialog to the 'Device ID' field in the 'DEVICE OVERVIEW' section. The 'DEVICE OVERVIEW' section shows the Application ID (my\_home\_sensors), Device ID (50ff1a0000010004), Description (Light test), and Activation Method (OTAA).

**DEVICE OVERVIEW**

Application ID **my\_home\_sensors**

Device ID **50ff1a0000010004**

Description **Light test**

Activation Method **OTAA**

**Edit ttn uplink node**

Delete Cancel Done

node properties

Name **my\_home\_sensors**

App **my\_home\_sensors**

Device ID **50ff1a0000010004**

Field

# 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 nodes: 'my\_home\_sensors' (blue) and 'msg.payload' (green). A blue arrow points from the 'msg.payload' node 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 }
```

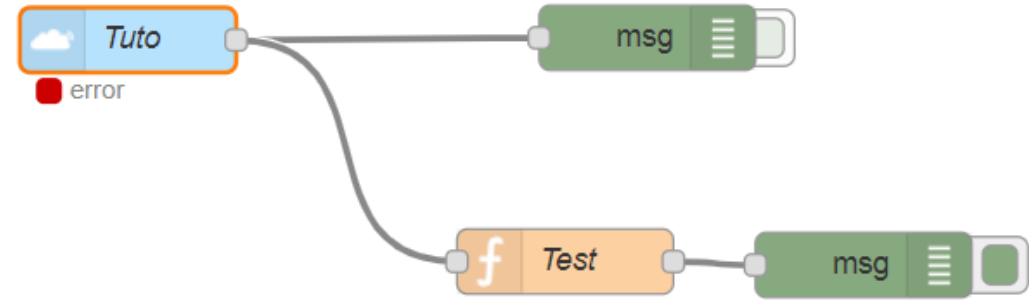
Below the node configuration, a code editor shows the following JSON payload:

```
{
  "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



```
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),  
  
};
```

# 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 »

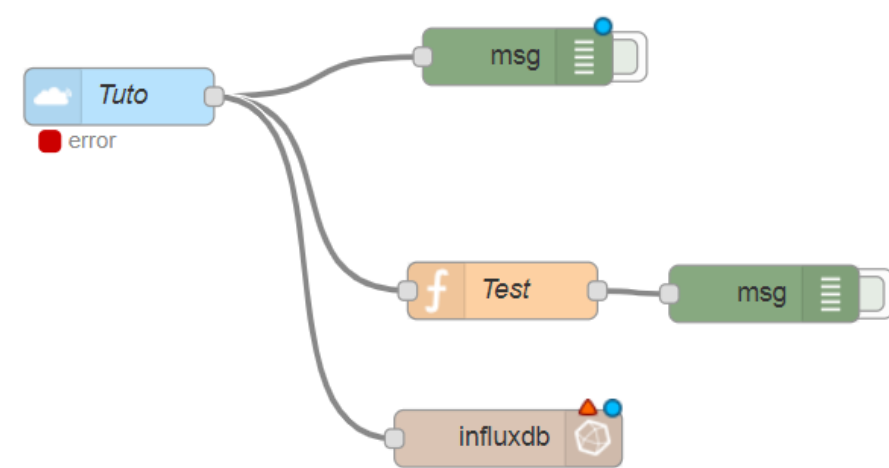
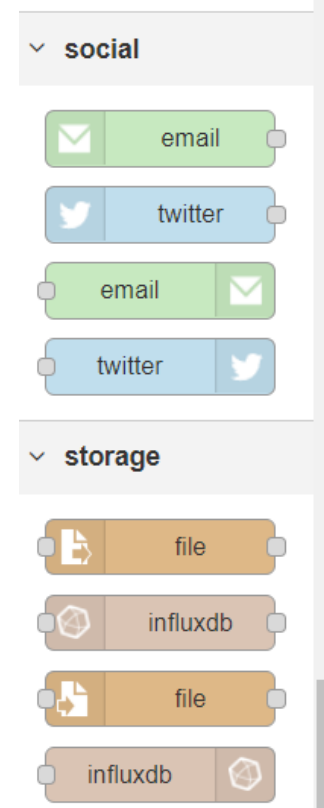
Your database is created

```
> 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

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

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) ☐

InfluxDB Details

Database: mySensor

User: Password:

✓ Datasource updated

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.

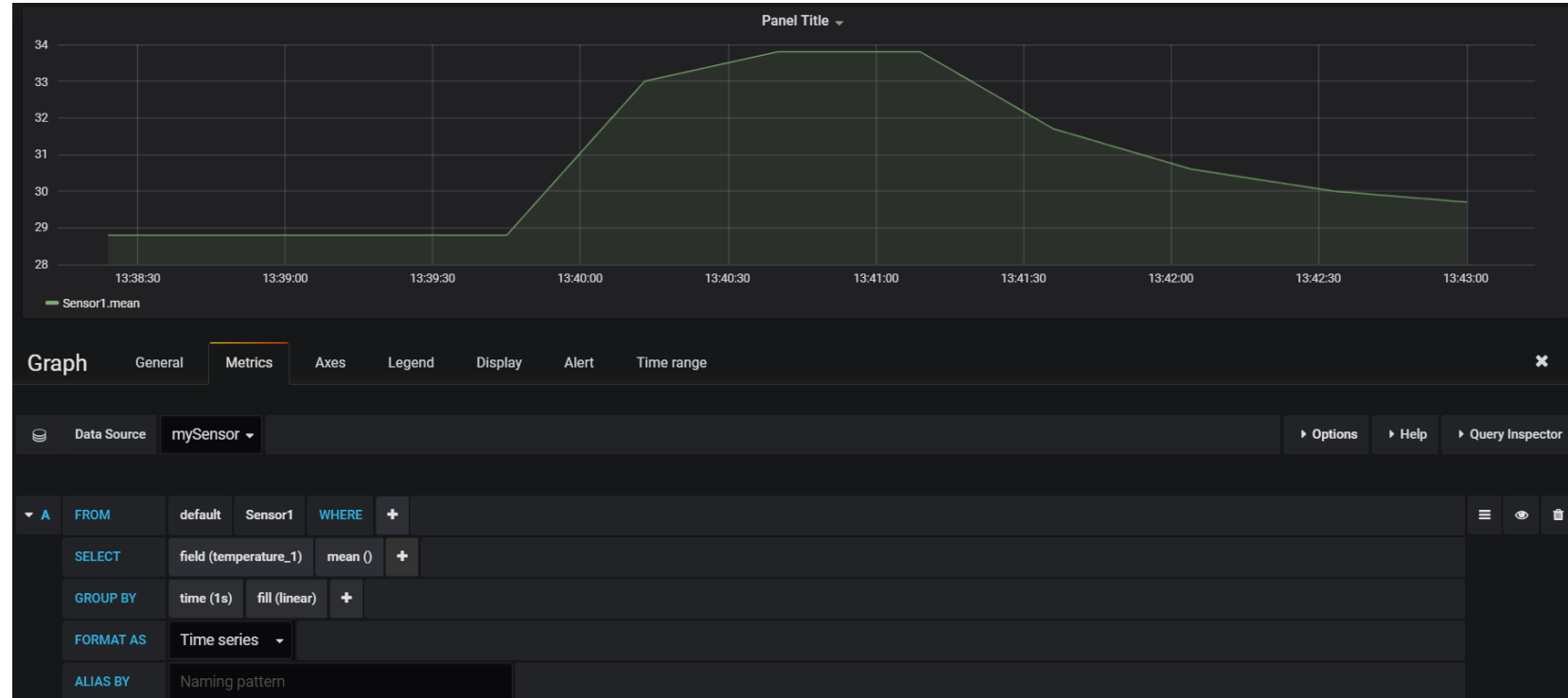
Min time interval: 10s

✓ Data source is working

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 !

