

# Explotació de dades

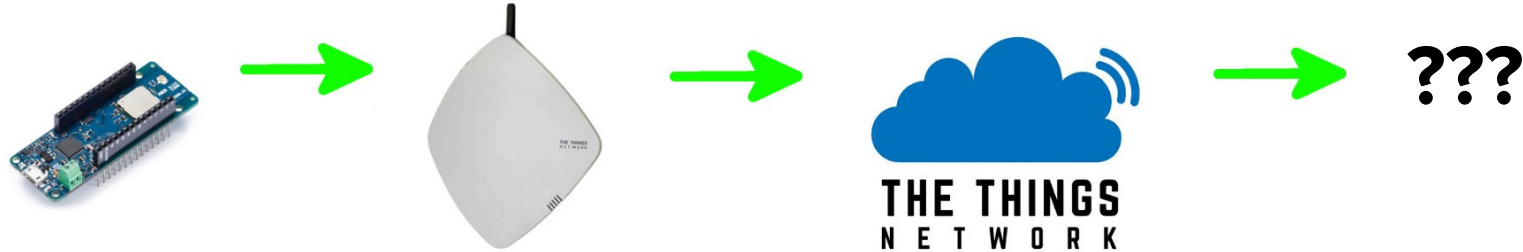
## Integració amb TTN



This work is licensed under a  
Creative Commons Attribution-ShareAlike 4.0  
International License

**Context**

# Què fem amb les dades?



**APPLICATION DATA** || pause 🗑 clear

Filters: uplink downlink activation ack error

	time	counter	port	
▲	23:04:19	3	10	payload: 03 00 00 E6 2AD341 D35B8D 42 CCAD7D 44 humidity: "71" pressure: "1014.72" t
▲	23:03:51	2	10	payload: 03 00 00 A3 51 D341 C0 398D 42 5FAD7D 44 humidity: "71" pressure: "1014.71" t
▲	23:03:23	1	10	payload: 03 00 00 61 78 D341 2F E6 8C 42 38AB7D 44 humidity: "70" pressure: "1014.68" t

# Plataformes

thethings.io



sentilo



Google Cloud Platform

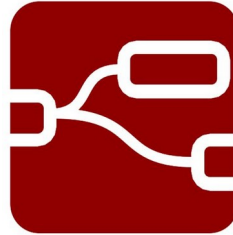


thingtia  
CLOUD



ThingSpeak

# Eines *on-premise*

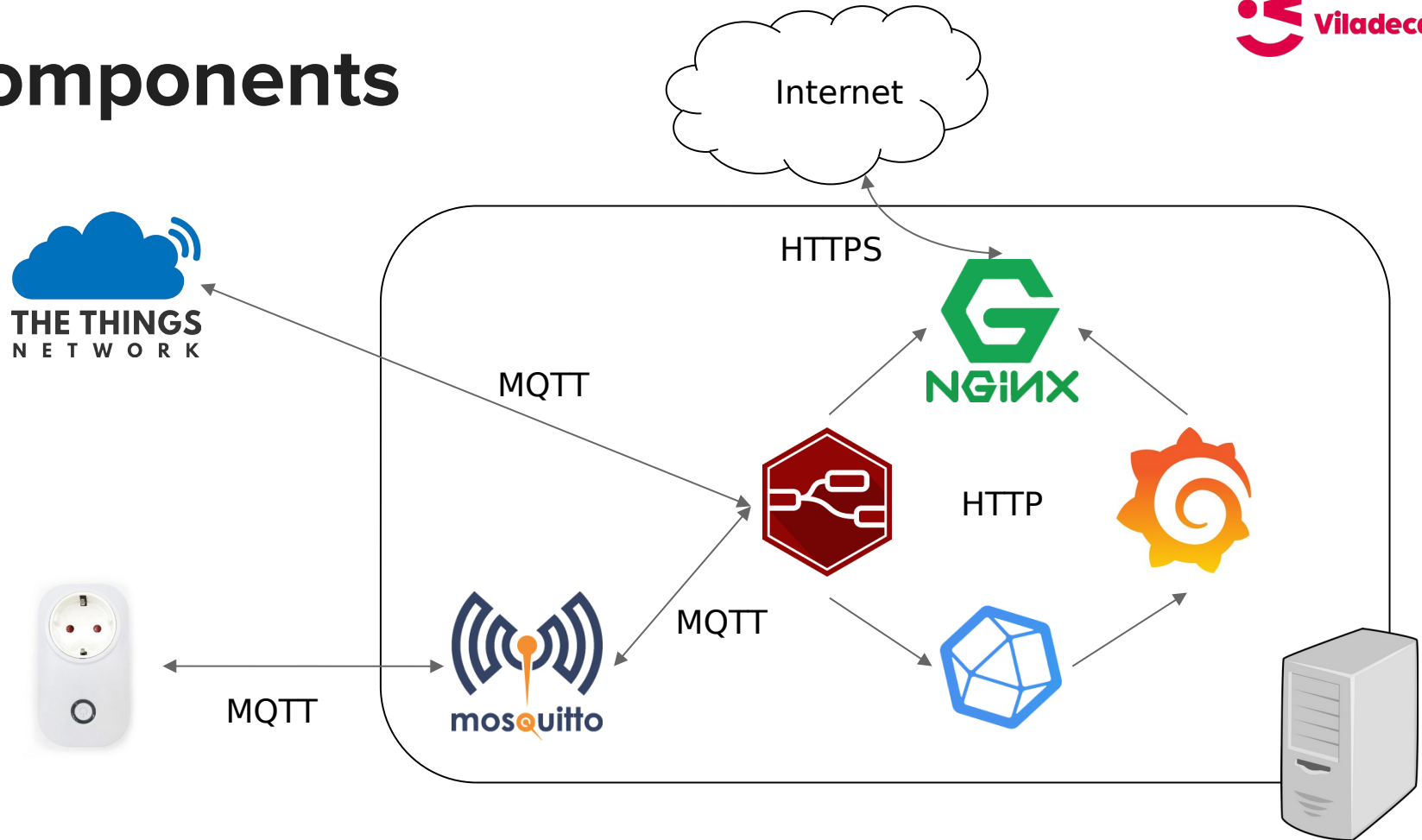


# Eines *on-premise* (domòtica)



**Arquitectura**

# Components





# Instal·lació manual

xoseperez / rpi3\_iot\_server.md

Last active 5 days ago

Code Revisions 48 Stars 67 Forks 21

Embed <script src="https://g...> Download ZIP

Raspberry Pi 3 with Mosquitto, Node-RED, InfluxDB, Grafana and Nginx (as a reverse proxy)

rpi3\_iot\_server.md Raw

## Raspberry Pi 3 IoT Home Server

### Presentation

[http://tinkerman.cat/rpi3\\_iot\\_server.pdf](http://tinkerman.cat/rpi3_iot_server.pdf) (Catalan)

### Get the latest image and flash the SD card

- download the latest image

```
$ wget --output-document=raspbian.img.zip https://downloads.raspberrypi.org/raspbian_lite_latest
```

- locate the destination volume

```
$ unzip -p raspbian.img.zip | sudo dd of=/dev/mmcblk0 bs=4M conv=fsync
```

- mount the SD card
- windows users might want to install <http://www.paragon-drivers.com/extfs-windows/> to read/write EXT4 partitions
- locate the `boot` and `rootfs` partitions (on bare Windows machines only `boot` partition is visible)
- enable ssh by default:

```
$ cd /media/SUSER/boot
$ touch ssh
```

- configure the wifi connection (optional):

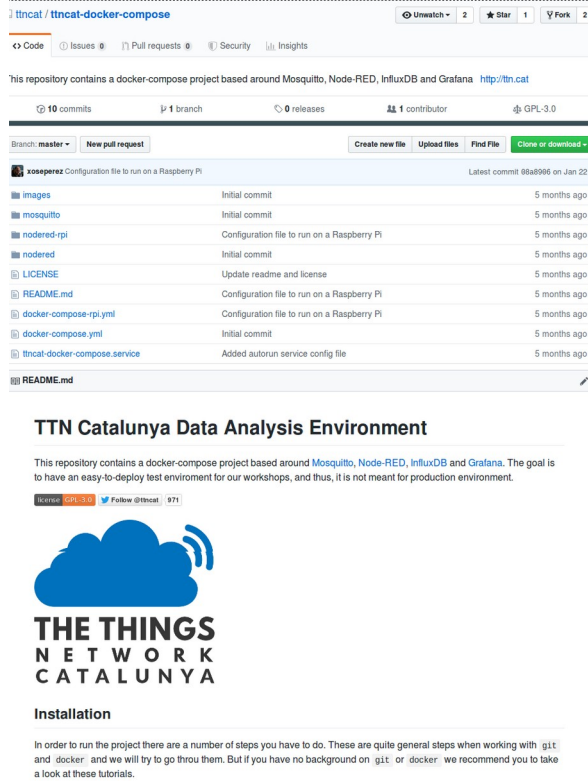
```
$ cd /media/SUSER/boot
```

Instruccions pas a pas:

<https://gist.github.com/xoseperez/e23334910fb45b0424b35c422760cb87>



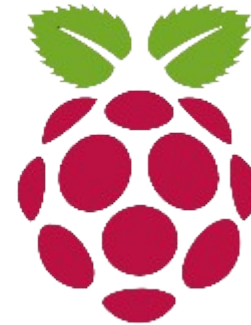
# Instal·lació amb Docker



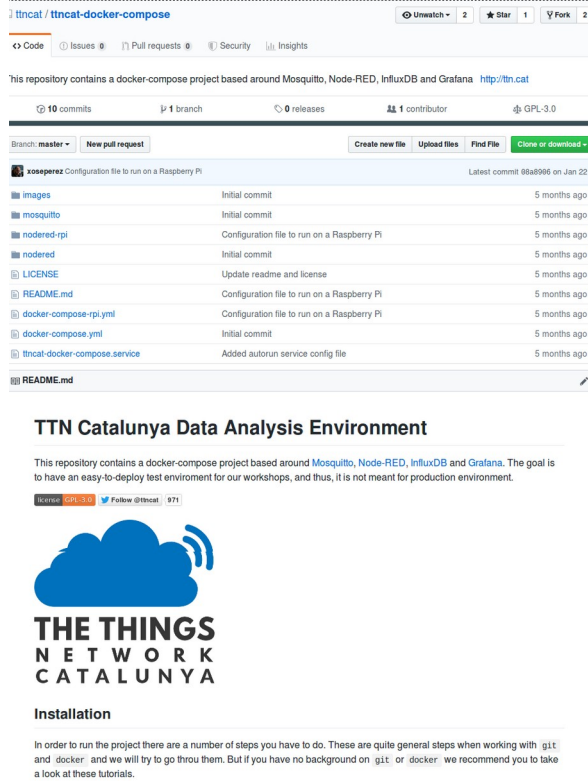
The screenshot shows the GitHub repository page for `ttn-cat/ttn-cat-docker-compose`. The repository is based around `Mosquitto`, `Node-RED`, `InfluxDB`, and `Grafana`. It contains a `docker-compose` project for running on a Raspberry Pi. The file list includes `images`, `mosquitto`, `nodered-rpi`, `nodered`, `LICENSE`, `README.md`, `docker-compose-rpi.yml`, `docker-compose.yml`, and `ttn-cat-docker-compose.service`. The README section is titled "TTN Catalunya Data Analysis Environment" and describes the goal of having an easy-to-deploy test environment. It includes a GitHub repository link and a social media follow button for @ttn\_cat.

<https://github.com/ttn-cat/ttn-cat-docker-compose>

```
$ git clone https://github.com/ttn-cat/ttn-cat-docker-compose.git
$ cd ttn-cat-docker-compose/docker
$ docker-compose up
```



# Instal·lació amb Vagrant

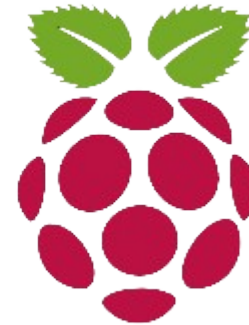


The screenshot shows the GitHub repository page for `ttn-cat/ttn-cat-docker-compose`. The repository is based around `Mosquitto`, `Node-RED`, `InfluxDB` and `Grafana`. The file list includes `images`, `mosquitto`, `nodered-rpi`, `nodered`, `LICENSE`, `README.md`, `docker-compose-rpi.yml`, `docker-compose.yml`, and `ttn-cat-docker-compose.service`. The `README.md` file is selected, showing the title "TTN Catalunya Data Analysis Environment" and a description: "This repository contains a docker-compose project based around `Mosquitto`, `Node-RED`, `InfluxDB` and `Grafana`. The goal is to have an easy-to-deploy test environment for our workshops, and thus, it is not meant for production environment." The installation section is partially visible, starting with "In order to run the project there are a number of steps you have to do. These are quite general steps when working with `git` and `docker` and we will try to go through them. But if you have no background on `git` or `docker` we recommend you to take a look at these tutorials."

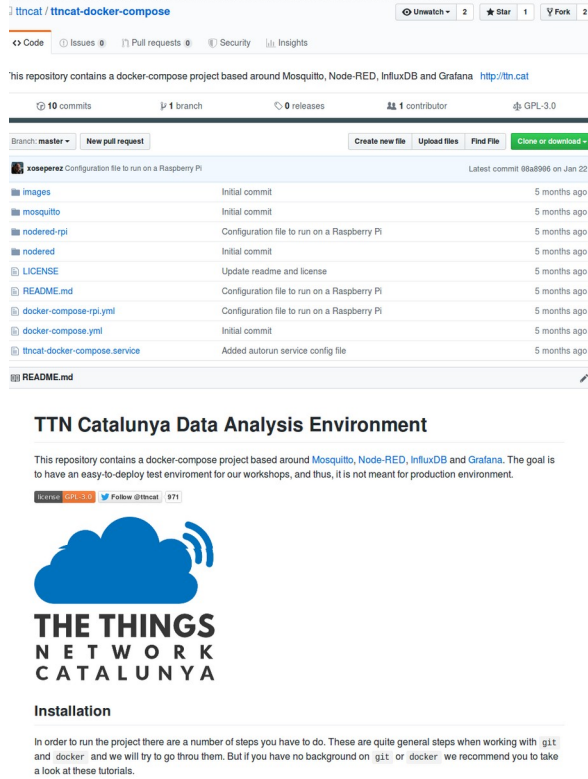
<https://github.com/ttn-cat/ttn-cat-docker-compose>

```
$ git clone https://github.com/ttn-cat/ttn-cat-docker-compose.git
$ cd ttn-cat-docker-compose/vagrant
$ vagrant up
```

Els serveis estaran disponibles sota 192,168,56,10



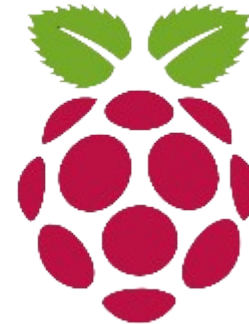
# Instal·lació amb VirtualBox



The screenshot shows the GitHub repository page for `ttn-cat/ttn-cat-docker-compose`. The repository is based around `Mosquitto`, `Node-RED`, `InfluxDB` and `Grafana`. It contains a `docker-compose` project for a Raspberry Pi. The file list includes `images`, `mosquitto`, `nodered-rpi`, `nodered`, `LICENSE`, `README.md`, `docker-compose-rpi.yml`, `docker-compose.yml`, and `ttn-cat-docker-compose.service`. The README section is titled "TTN Catalunya Data Analysis Environment" and describes the goal of having an easy-to-deploy test environment. It also includes a section for "Installation" with instructions on how to run the project.

<https://github.com/ttn-cat/ttn-cat-docker-compose>

Anar a assets i descarregar la darrera image (fitxer .OVA) per importar-la des de VirtualBox. Un cop arrencada la matge els serveis estaran disponibles a 192,168,56,10

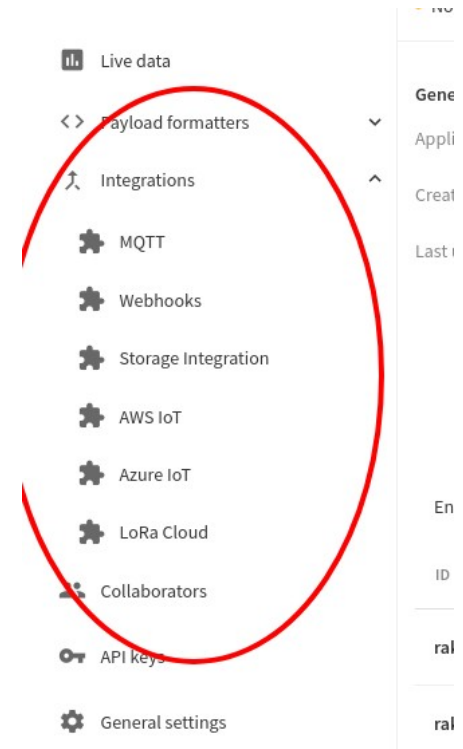


**Integració**

# Integracions des de TTN

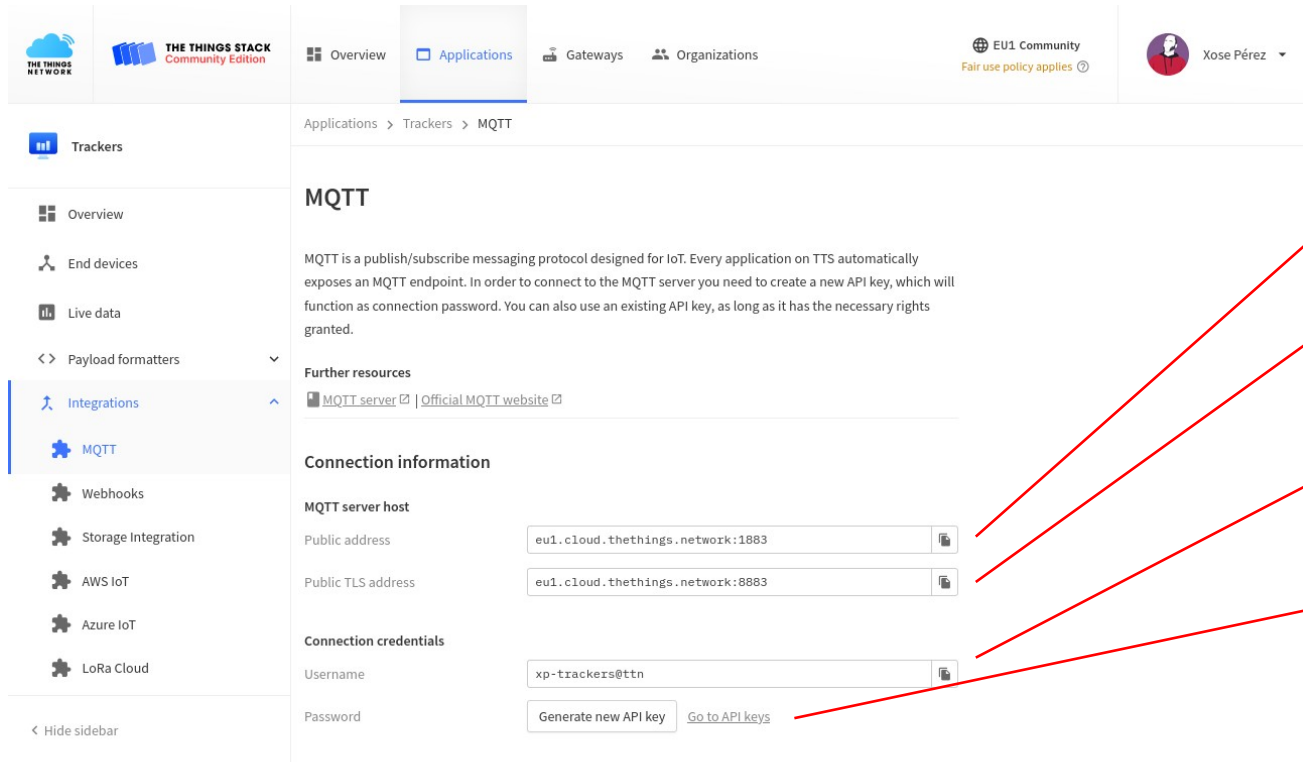
Per cada aplicatiu es poden definir 1 o més integracions:

- MQTT: publica missatges i events al broker local
- WebHooks: executa peticions HTTP(S)
- Storage Integration: habilita l'emmagatzemament temporal de dades (30 dies), consultable via API o gRPC
- AWS IoT: s'integra amb el packet IoT d'Amazon Web Services
- Azure IoT: s'integra amb el packet IoT de Microsoft Azure
- LoRa Cloud: envia les dades (i metadades) a LoRa Cloud de Semtech.



# Enllaç MQTT

<https://www.thethingsindustries.com/docs/reference/root-certificates/>



The screenshot shows the 'MQTT' configuration page in the The Things Stack. The left sidebar contains navigation options: Trackers, Overview, End devices, Live data, Payload formatters, Integrations (selected), MQTT (selected), Webhooks, Storage Integration, AWS IoT, Azure IoT, and LoRa Cloud. The main content area is titled 'MQTT' and includes a description of the protocol, further resources, and connection information. The connection information section contains the following fields:

- MQTT server host:**
  - Public address:
  - Public TLS address:
- Connection credentials:**
  - Username:
  - Password:  [Go to API keys](#)

Servidor MQTT no segur

Servidor MQTT segur

Usuari

Mot d'accés

**Eines**





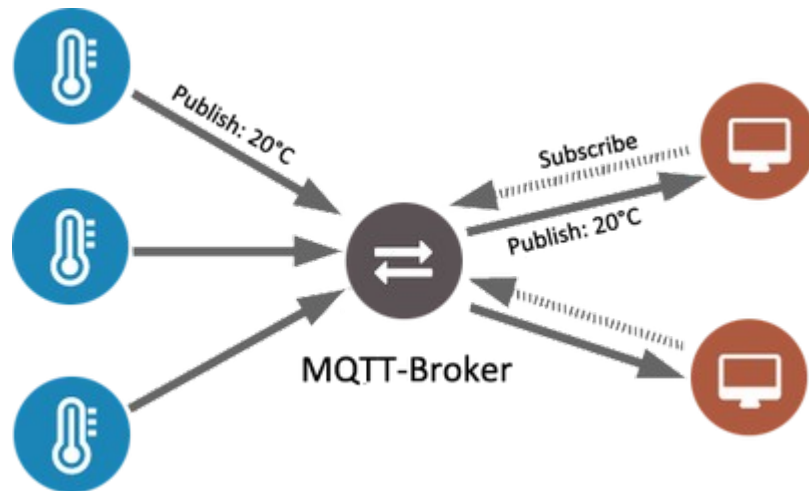
# Mosquitto

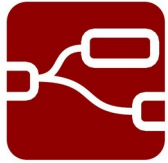
## MQTT

- *Message Queueing Telemetry Transport*
- Protocol de missatgeria especialment dissenyat per telemetria (**sensors**)
- Patró **publish/subscribe**
- Quality of Service (**QoS**)
- Distribuït (*bridging*)
- Open Specification

## Mosquitto

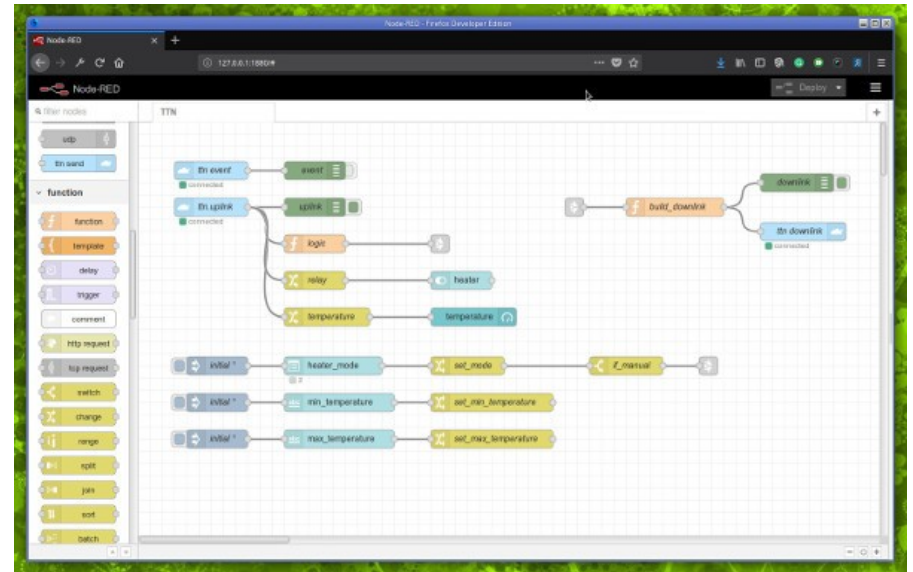
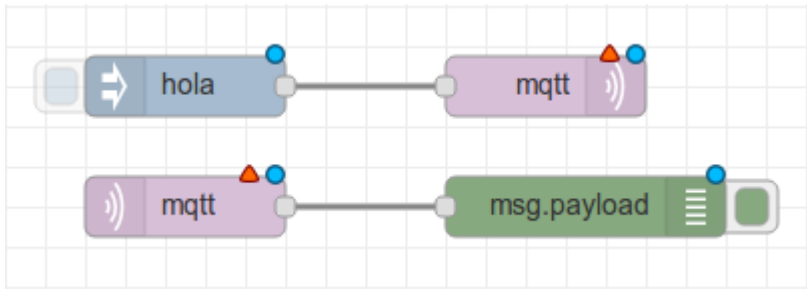
- Broker MQTT
- Open Source

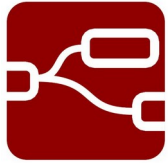




# Node-RED

- Aplicatiu BI visual (drag & drop)
- Lògica basada en **nodes i fluxes**
- Basat en node.js (~javascript)
- Suport MQTT per defecte
- Open source
- Comunitat gran i activa
- **Aplicatiu web**





# Node-RED



Node

Fluxe

Selector  
d'espai de  
treball

Cercador  
de nodes

Desplegament  
de canvis

Configuració

Biblioteca  
de nodes

Missatges  
de  
depuració

Informació  
del node  
seleccionat

Espai  
de  
treball

Zona  
d'informació i  
missatges de  
depuració

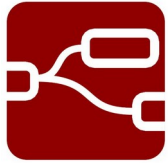
The screenshot shows the Node-RED web interface in a browser window. The main workspace contains a flow with the following nodes: a 'thermostat' node connected to a 'debug' node, which is then connected to a 'lògica' (logic) node. The logic node has two outputs: 'tancar calefacció' (close heating) and 'obrir calefacció' (open heating), both connected to MQTT publish nodes. On the left, the 'input' category of the node library is visible, including nodes like inject, catch, status, link, mqtt, http, websocket, tcp, udp, ttn event, and ttn uplink. The right sidebar shows the 'info' panel for the selected MQTT node, displaying its ID, name, type, and help information. The top navigation bar includes a search bar, tabs for different flows, and a 'Deploy' button.



# Node-RED - Missatges

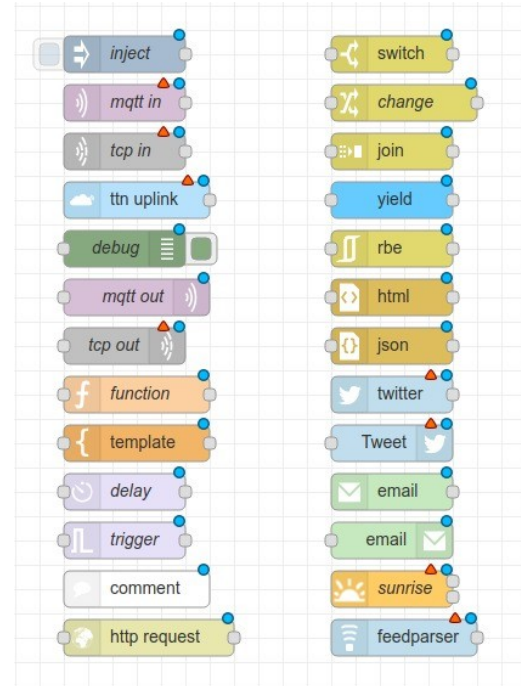
- objecte que es passa d'un node a un altre
- acostuma a estar en format **JSON**
- estructura i propietats arbitràries, però
- sovint presenta un **topic** i un **payload**
- de vegades conté informació de configuració pels nodes

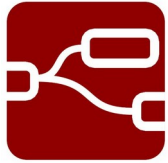
```
{  
  "topic": "/device/rfm69gw/rssi",  
  "payload": "-36",  
  "qos": 0,  
  "retain": false,  
  "_msgid": "6336dfbc.26b45"  
}
```



# Node-RED - Nodes

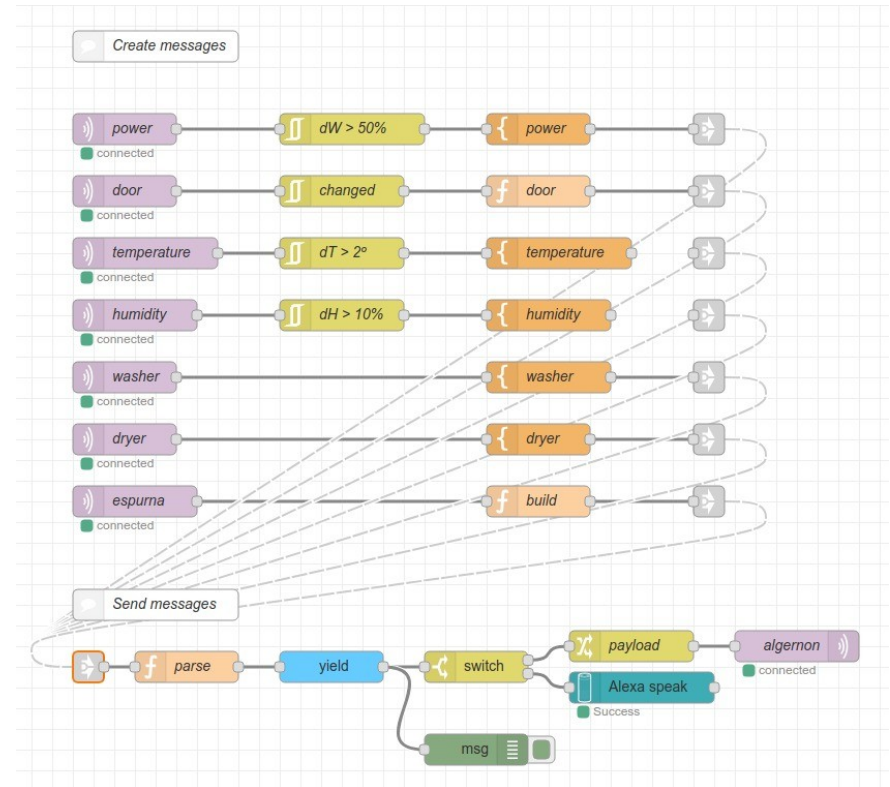
- rep un missatge i el processa
- una o cap entrada
- cap, una o més sortides
- pot descartar el missatge
- fa una única cosa
- es pot preconfigurar o
- pot agafar la configuració del missatge
- biblioteca de nodes precarregada
- milers d'extensions amb desenes de milers de nodes (<https://flows.nodered.org>)

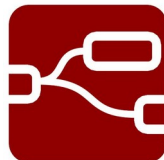




# Node-RED - Fluxe

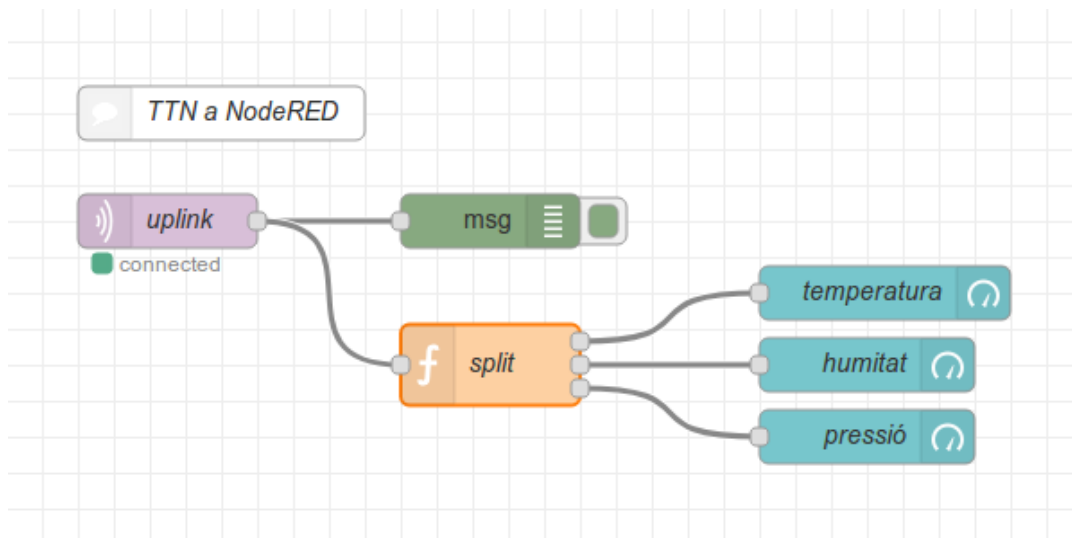
- conjunt de nodes connectats
- pot tenir múltiples ramificacions
- es pot dividir en diferents espais de treball amb nodes tipus “link”
- compte amb els bucles!



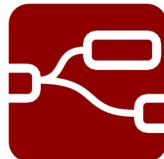


`http://<ip>:1880`

- El node MQTT rep el missatge de TTN i el node JSON converteix el payload de string a json.
- El podem veure amb el node “debug”
- “Split” en un node “function” que separa el payload en valors discrets
- I es passen individualment a nodes tipus “chart”



# TTN - Node-RED



Server: ttn  
Topic: +/devices+/up  
QoS: 2  
Output: a parsed JSON object  
Name: uplink

Name: ttn  
Connection: Security  
Username: ttnctl-taller  
Password: .....

Connection Security Messages  
Server: eu.thethings.network Port: 8883  
 Enable secure (SSL/TLS) connection  
TLS Configuration: TLS configuration  
Client ID: Leave blank for auto generated  
 Keep alive time (s): 60  Use clean session  
 Use legacy MQTT 3.1 support

Use key and certificates from local files

Certificate: Upload

Private Key: Upload

Passphrase: private key passphrase (optional)

CA Certificate: Upload mqtt-ca.pem

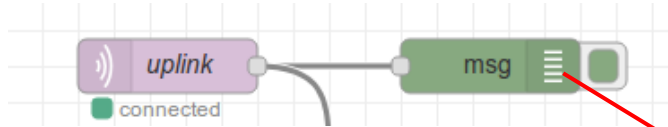
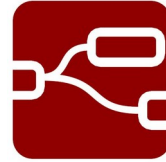
Verify server certificate

Server Name: for use with SNI

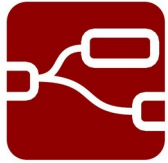
Name: Name

ACCESS KEYS		manage keys
default key	devices messages	base64
curs_upc_cim	messages	base64
nodered	messages	base64





```
{
  "topic": "ttncat-taller/devices/wcbn2-01/up",
  "payload": {
    "app_id": "ttncat-taller",
    "dev_id": "wcbn2-01",
    "hardware_serial": "3834313933085223",
    "port": 10,
    "counter": 1,
    "payload_raw": "AwAARprSQb8Sjk+s31E",
    "payload_fields": {
      "humidity": "71",
      "pressure": "1014.80",
      "temperature": "26.33"
    }
  },
  "metadata": {
    "time": "2019-09-17T21:14:50.777512035Z",
    "frequency": 868.1,
    "modulation": "LORA",
    "data_rate": "SF8BW125",
    "airtime": 123392000,
    "coding_rate": "4/5",
    "gateways": [...]
  }
},
"qos": 0,
"retain": false,
"_msgid": "43dbeaa.c538214"
}
```



# Node-RED - Dashboard



**Edit function node**

Delete Cancel Done

▼ node properties

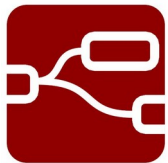
Name  
split

Function

```
1 var out = [];  
2 out.push({"payload": parseFloat(msg.payload.temperature)});  
3 out.push({"payload": parseFloat(msg.payload.pressure)});  
4 out.push({"payload": parseFloat(msg.payload.humidity)});  
5 return out;
```

⌘ Outputs 3

See the Info tab for help writing functions.



# Node-RED - Dashboard

**Edit chart node**

Delete Cancel Done

node properties

Group [BME280] Temperature

Size auto

Label Last hour

Type Line chart  enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss

Y-axis min 10 max 20

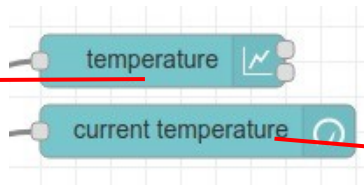
Legend None Interpolate linear

Series Colours

Blank label display this text before valid data arrives

Use deprecated (pre 2.5.0) data format.

Name temperature



**Edit gauge node**

Delete Cancel Done

node properties

Group [BME280] Temperature

Size auto

Type Gauge

Label Current

Value format {{value}}

Units °C

Range min 0 max 30

Colour gradient

Sectors 0 ... 15 ... 25 ... 30

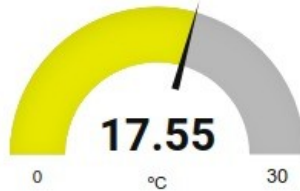
Name current temperature



# Node-RED - Dashboard

## Temperature

Current



17.55

0 °C 30

Last hour



## Pressure

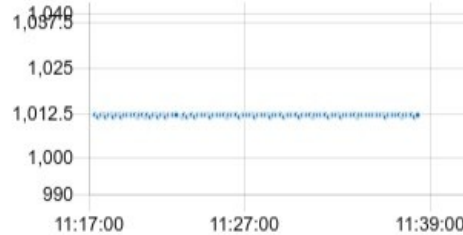
Current



1011.8

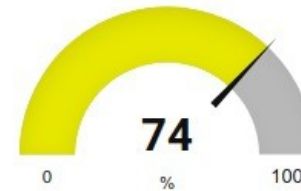
980 hPa 1040

Last hour



## Humidity

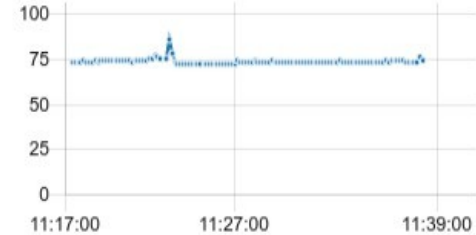
Current



74

0 % 100

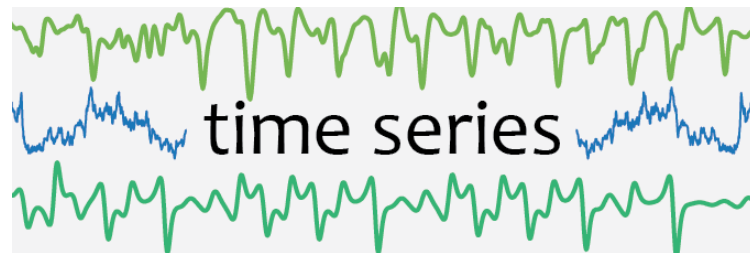
Last hour





# Influxdb

- Base de dades
- Específica per **sèries temporals**
- **Sense estructura**
- Taules => Measurements/Series
- Camps => Tags/Fields
- **API HTTP**
- **Retention policies**
- **Continuous queries**
- Open source





# Influxdb

```
$ influx -precision "rfc3339"
```

```
InfluxDB shell 0.10.0
```

```
> create database ttncat
```

```
> use ttncat
```

```
Using database ttncat
```

```
> select * from "ttncat-taller"
```

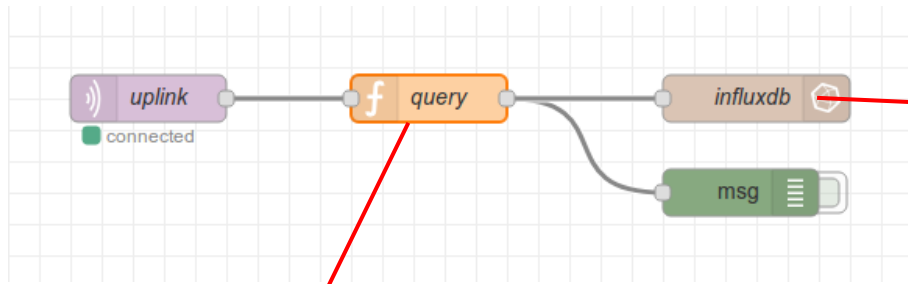
```
name: ttncat-taller
```

```
-----
```

time	device	humidity	port	pressure	temperature
2019-09-17T21:29:54.721268783Z	wcbn2-01	71	10	1014.95	26.46
2019-09-17T21:30:21.520116020Z	wcbn2-01	71	10	1014.95	26.44
2019-09-17T21:30:49.184982791Z	wcbn2-01	71	10	1015.00	26.44
2019-09-17T21:31:17.147908115Z	wcbn2-01	71	10	1015.00	26.46
2019-09-17T21:31:44.521738735Z	wcbn2-01	71	10	1015.01	26.47



# Node-RED - Influxdb



```
msg.measurement = msg.payload.app_id;
msg.payload = [
  msg.payload.payload_fields,
  {
    "device": msg.payload.dev_id,
    "port": msg.payload.port
  }
];
return msg;
```

Server: influxdb:8086/ttncat

Measurement:

Advanced Query Options

Name: influxdb

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

Host: influxdb Port: 8086

Database: ttncat

Username:

Password:

Enable secure (SSL/TLS) connection

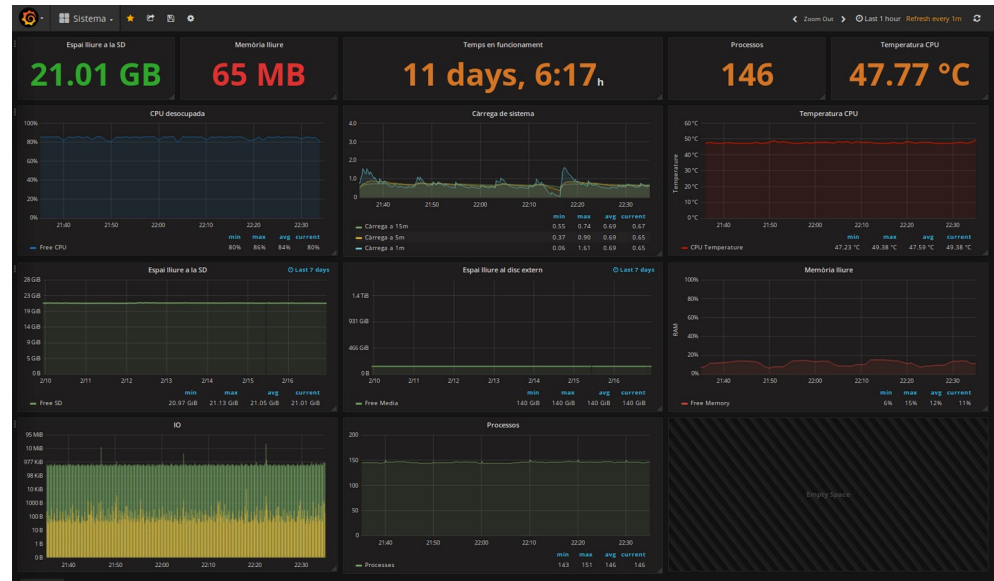
Name: Name



# Grafana



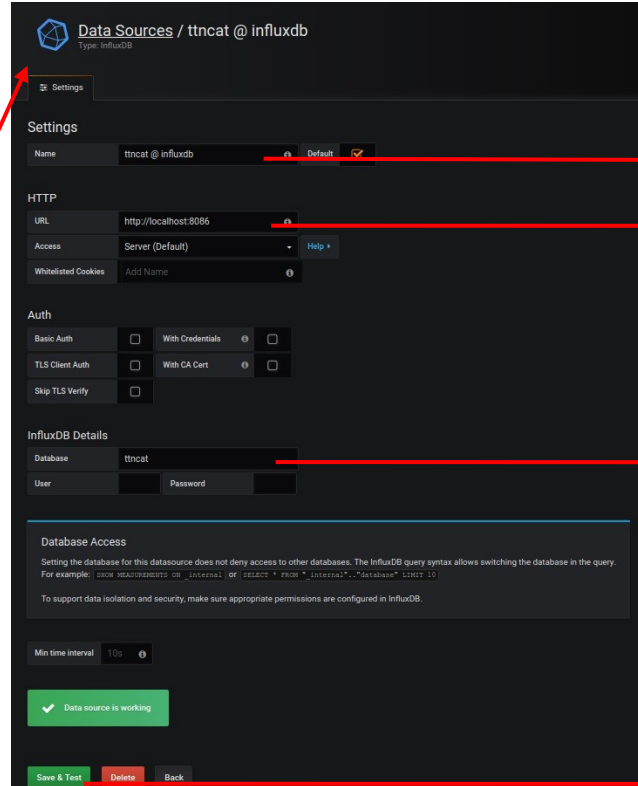
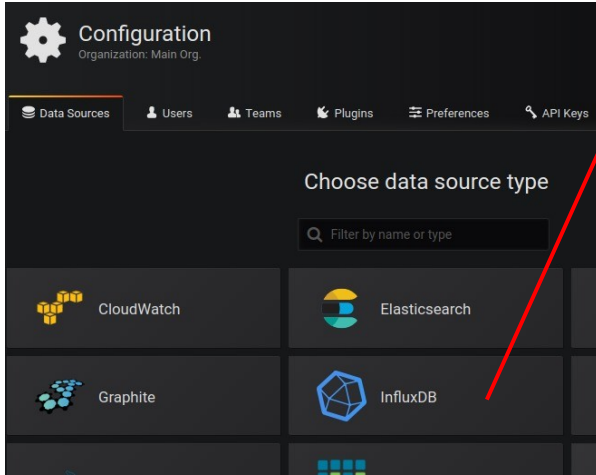
- Eina de **graficat i analítica**
- Especialment dissenyada per **dades temporals**.
- **Orígens de dades** (data sources):  
Elasticsearch, Graphite, Prometheus, MySQL, PostgreSQL, InfluxDB,...
- Aplicatiu web
- Open source







# Influxdb - Grafana



Anomena la connexió  
URL del servidor  
[http://influxdb:8086]

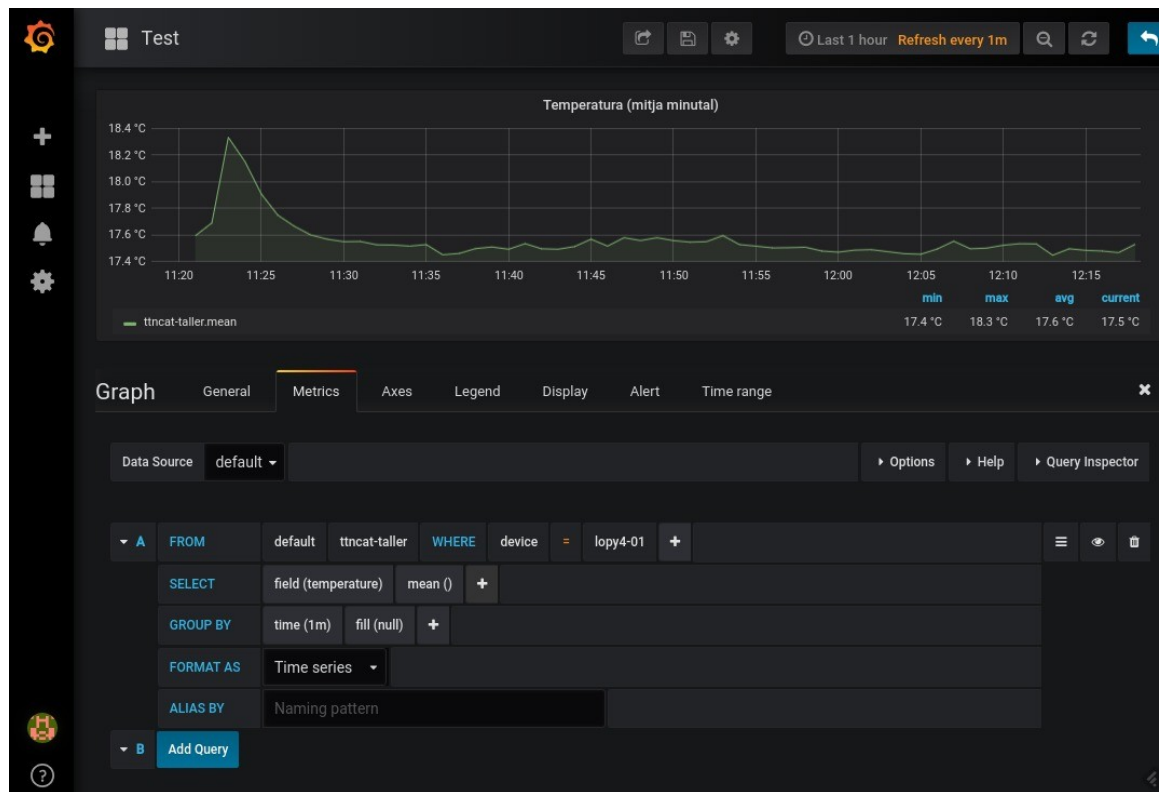
Nom de la base de dades  
[ttnocat]

Comprovar i desar

http://<ip>:3000



# Grafana







# Grafana





# Telegram Bot (@botfather)

-  **Xose** 11:50:32 PM  
/newbot
-  **BotFather** 11:50:32 PM  
Alright, a new bot. How are we going to call it? Please choose a name for your bot.
-  **Xose** 11:50:46 PM  
taller-ttncat
-  **BotFather** 11:50:46 PM  
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris\_bot.
-  **Xose** 11:51:00 PM  
taller\_ttncat\_bot

-  **BotFather** 11:51:00 PM  
Done! Congratulations on your new bot. You will find it at [t.me/taller\\_ttncat\\_bot](https://t.me/taller_ttncat_bot). You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
- Use this token to access the HTTP API:  

- Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.
- For a description of the Bot API, see this page:  
<https://core.telegram.org/bots/api>



Write a message...



SEND



# Telegram Bot (@myid\_bot)

Wednesday, September 18, 2019



**Xose**

/start

12:02:19 AM



**My ID**

Hola, este bot te dirá tu id de Telegram. Escribe /id para ver tu ID de usuario y /chatid para ver el ID del chat.

12:02:19 AM



**Xose**

/chatid

12:02:29 AM



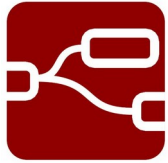
**My ID**

The chat id is XXXXXXXXXX

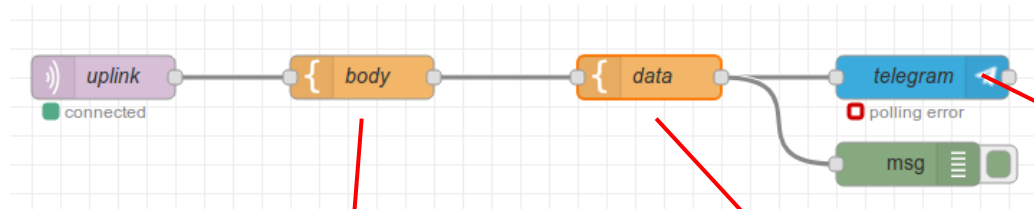
12:02:29 AM



SEND



# Node-RED - Telegram



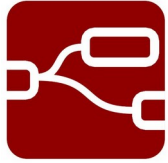
Bot-Name

Token

Sensor BME280:  
Temperatura: `{{ payload.payload_fields.temperature }}`C  
Humitat: `{{ payload.payload_fields.humidity }}`%  
Pressió: `{{ payload.payload_fields.pressure }}`hPa

```
{  
  "chatId": xxxxxxxx,  
  "type": "message",  
  "content": "{{ payload }}"  
}
```

Output as "parsed JSON"!!



# Node-RED - Telegram



Wednesday, September 18, 2019



**TTNCat Bot**

12:18:15 AM

Sensor BME280:

Temperatura: 26.48C

Humitat: 71%

Pressió: 1014.45hPa



Write a message...



SEND



# Grafana - Telegram



Alert Rules | Notification channels

## New Notification Channel

Name: Telegram

Type: Telegram

Send on all alerts

Include image

Disable Resolve Message

Send reminders

### Telegram API settings

BOT API Token:

Chat ID:

[Save](#) [Send Test](#) [Back](#)

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

## Alert Config

Alert Config

Notifications (1): Name: Temperatura (mitja minatal) alert

State history: Evaluate every: 1m For: 5m

Delete

### Conditions

WHEN avg () OF query (A, 5m, now) IS ABOVE 17.6

+

If no data or all values are null SET STATE TO No Data

If execution error or timeout SET STATE TO Alerting

[Test Rule](#)