

The background of the slide features a complex visualization of IoT data stacks. It consists of two main horizontal bands. The top band is a sparse, light-colored area with thin, vertical orange lines of varying heights, suggesting a low-frequency or low-amplitude data stream. The bottom band is a much denser, darker area filled with a multitude of thin, vertical lines in various colors, including orange, green, and blue. These lines represent a high-frequency, high-volume data stream. The overall effect is one of a large, active data environment.

IoT2024 - 06 - Data stacks for the IoT

Sebastian Büttrich 202203, update 202403

20:00 21:00 22:00 23:00 00:00 01:00 02:00 03:00 04:00 05:00 06:00 07:00

Agenda

Data stacks: transport, storage, presentation

Ingesting data

We know how to generate data and we know the physical network layers (and a bit of higher layers, e.g. MQTT) - now we need to complete the data flow

Storing data

(Timeseries) Databases for IoT Data, e.g. InfluxDB

Showing (and (pre-)processing) data

e.g. Grafana

Example



Ingesting data



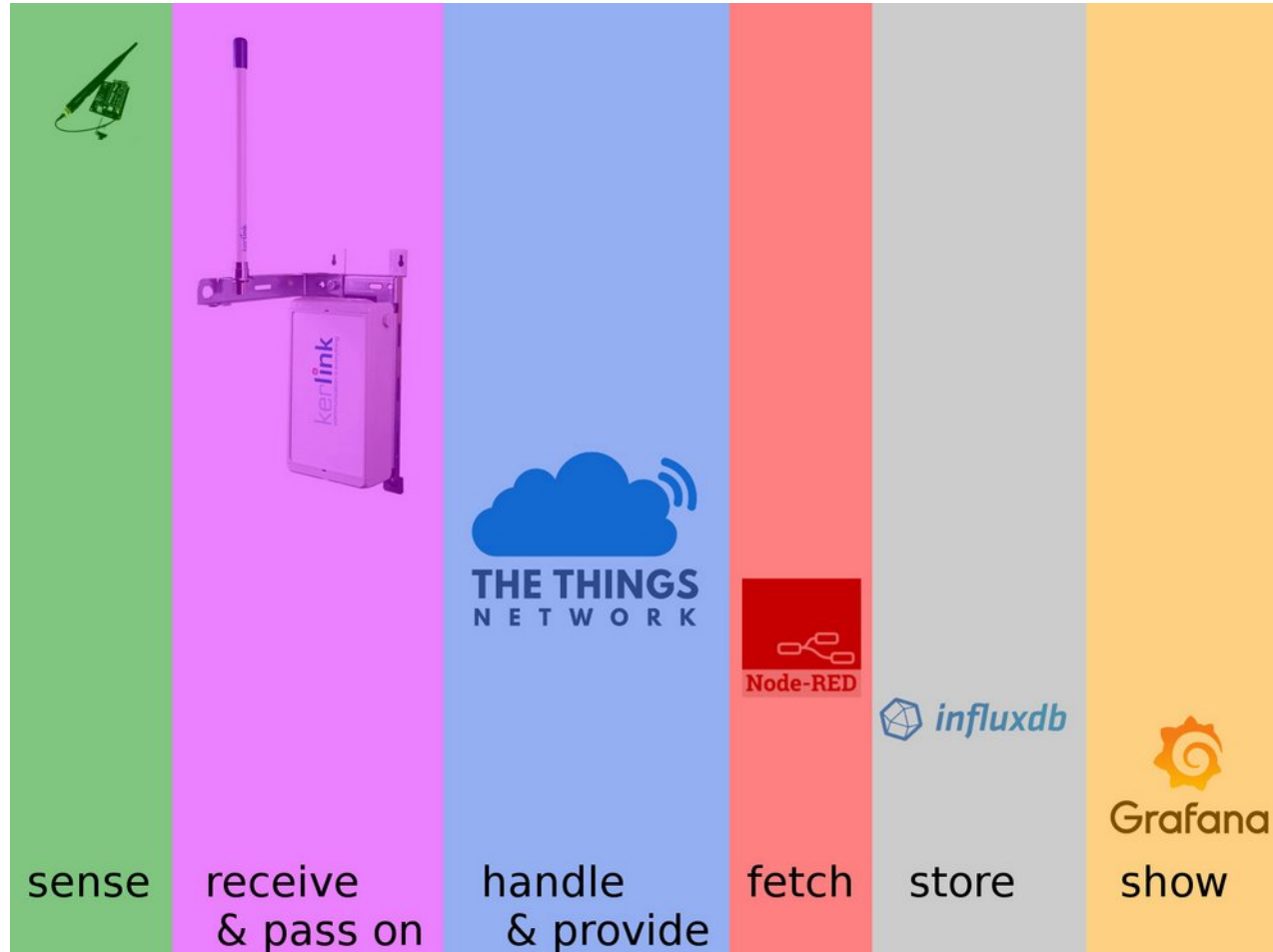
Storing data



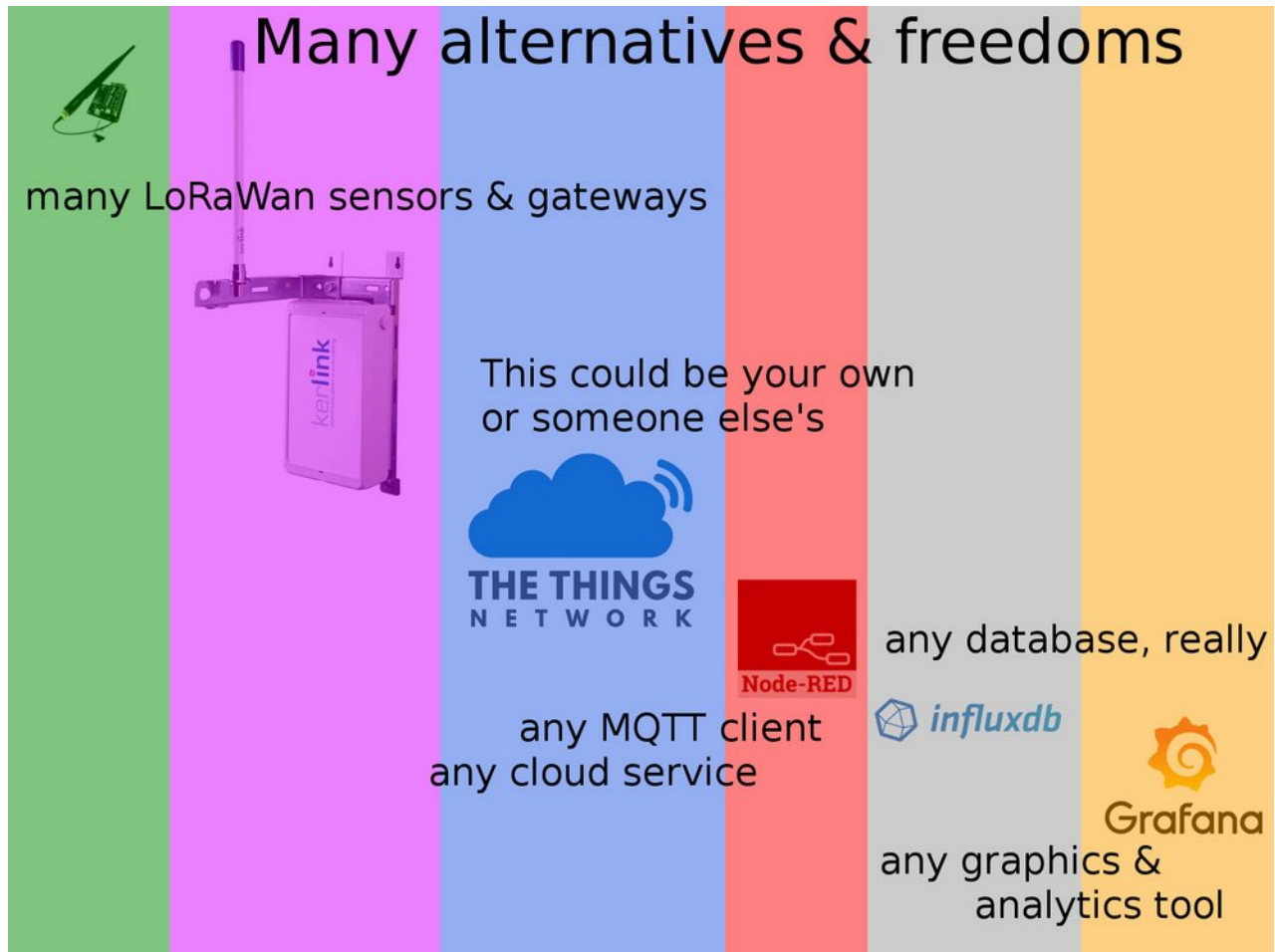
Grafana

Showing (and processing) data

Roles



Freedom Of Choice



Our current setup(s)



Ingesting data



Storing data



Grafana

Showing (and processing) data

Options & freedoms

**Though standardization in the IoT is still a big challenge,
the vast majority of IoT stacks agree on some open technologies
For connectivity and communication -**

e.g. LoRaWAN, ... http APIs, MQTT, json,

allowing us to combine and exchange systems quite freely.

MQTT is everywhere - e.g. TTN

The Things Network's Application server, openly available in the community version, makes data available via MQTT.

It implements a 'tenant' structure which you find in many commercial IoT systems.

You can subscribe to the data feeds provided you have the right credentials for a given application.

MQTT is everywhere - e.g. TTN

The screenshot shows the 'Applications' page for 'co2-001' in the 'MQTT' section. The left sidebar contains navigation links: Overview, End devices, Live data, Payload formatters, Integrations (selected), MQTT (selected), Webhooks, Storage Integration, AWS IoT, and Azure IoT Hub. The main content area has a breadcrumb 'Applications > co2-001 > MQTT' and a title 'MQTT'. Below the title is a paragraph explaining MQTT as a publish/subscribe messaging protocol for IoT, noting that every application on TTS automatically exposes an MQTT endpoint and that a new API key must be created to connect. Further resources include links to the 'MQTT server' and the 'Official MQTT website'. The 'Connection information' section contains two rows: 'MQTT server host' with 'Public address' (eu1.cloud.thethings.network:1883) and 'Public TLS address' (eu1.cloud.thethings.network:8883). The 'Connection credentials' section contains 'Username' (dasya-co2-001@ttn) and 'Password' (with a 'Generate new API key' button and a 'Go to API keys' link).

THE THINGS STACK
Community Edition

Overview Applications Gateways Organizations

co2-001

Overview

End devices

Live data

Payload formatters

Integrations

MQTT

Webhooks

Storage Integration

AWS IoT

Azure IoT Hub

Hide sidebar

Applications > co2-001 > MQTT

MQTT

MQTT is a publish/subscribe messaging protocol designed for IoT. Every application on TTS automatically exposes an MQTT endpoint. In order to connect to 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.

Further resources

[MQTT server](#) | [Official MQTT website](#)

Connection information

MQTT server host

Public address: eu1.cloud.thethings.network:1883

Public TLS address: eu1.cloud.thethings.network:8883

Connection credentials

Username: dasya-co2-001@ttn

Password: [Generate new API key](#) [Go to API keys](#)

MQTT is everywhere - e.g. TTN

MQTT broker `eu1.cloud.thethings.network:8883`

user `application@ttn` `ttn = tenant`

password `API key generated by application server`

topics:

`v3/{application id}@{tenant id}/devices/{device id}/join`

`v3/{application id}@{tenant id}/devices/{device id}/up`

`v3/{application id}@{tenant id}/devices/{device id}/down/queued`

`v3/{application id}@{tenant id}/devices/{device id}/down/sent`

`v3/{application id}@{tenant id}/devices/{device id}/down/ack`

`v3/{application id}@{tenant id}/devices/{device id}/down/nack`

`v3/{application id}@{tenant id}/devices/{device id}/down/failed`

`v3/{application id}@{tenant id}/devices/{device id}/service/data`

`v3/{application id}@{tenant id}/devices/{device id}/location/solved`

MQTT is everywhere - e.g. TTN

```
^Csebastian@x2021:~$ mosquitto_sub -u dasya-co2-001@ttn -P NNSXS.MJIIGGBMBOTBYPFH4WKKAGYC2GOMTF3JNMRGJWQ.FCV46DXUDW2V4CCXV3
IRI6LUGOVSVJT2BYIJVJHLZLSPNI3FKQEQ -v --capath /etc/ssl/certs -h eu1.cloud.thethings.network -p 8883 -t "#"
v3/dasya-co2-001@ttn/devices/eui-70b3d54990564b35/up {"end_device_ids":{"device_id":"eui-70b3d54990564b35","application_ids
":{"application_id":"dasya-co2-001"},"dev_eui":"70B3D54990564B35","join_eui":"70B3D57ED003CEC0","dev_addr":"260B0DB0"},"cor
relation_ids":["as:up:01FXMZ90BEPJDSRVFJB8ANT3RK","gs:conn:01FXHXCX0EC5EZKBMTCT7FZB585","gs:up:host:01FXHXCXY5T0N5R7J863FFH9M
NM","gs:uplink:01FXMZ90502333RMD72XCC5B0Q","ns:uplink:01FXMZ9050E720JXQP2HKVDEEE","rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01
FXMZ9050XM13J6XW0RGGGRPV","rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FXMZ90BET12KGQKGABFT68WM"],"received_at":"2022-03-08T14:
35:05.966780657Z","uplink_message":{"session_key_id":"AX7d4Kd00GgGq7h2+290/Q==","f_port":2,"f_cnt":18501,"frm_payload":"AeB
mTRNJ","rx_metadata":[{"gateway_ids":{"gateway_id":"dasya-purple-01","eui":"B827EBFFFE51B07E"},"timestamp":370017707,"rssi"
:-23,"channel_rssi":-23,"snr":10.2,"location":{"latitude":55.65971490666779,"longitude":12.591437101364138,"altitude":30,"s
ource":"SOURCE_REGISTRY"},"uplink_token":"Ch0KGwoPZGFzeWetcHVycGxLLTAxEgi4J+v//lGwfHcri7iwARoMCJnTnZEGENK6q+oCIPjHmLbiwhc="
}]},"settings":{"data_rate":{"lora":{"bandwidth":125000,"spreading_factor":7},"coding_rate":"4/5","frequency":"868100000","
timestamp":370017707},"received_at":"2022-03-08T14:35:05.760496726Z","consumed_airtime":"0.051456s","locations":{"user":{"l
atitude":55.659605433634624,"longitude":12.591425404814808,"altitude":30,"source":"SOURCE_REGISTRY"},"network_ids":{"net_i
d":"000013","tenant_id":"ttn","cluster_id":"ttn-eu1"}}}
v3/dasya-co2-001@ttn/devices/eui-70b3d54990564b35/up {"end_device_ids":{"device_id":"eui-70b3d54990564b35","application_ids
":{"application_id":"dasya-co2-001"},"dev_eui":"70B3D54990564B35","join_eui":"70B3D57ED003CEC0","dev_addr":"260B0DB0"},"cor
relation_ids":["as:up:01FXMZCWDDBCQDNMKA42383TTCD","gs:conn:01FXHXCX0EC5EZKBMTCT7FZB585","gs:up:host:01FXHXCXY5T0N5R7J863FFH9M
NM","gs:uplink:01FXMZC6SKVM0BXQ4ZP6741KH","ns:uplink:01FXMZC6S9ZS3GZ6V69W57XPR","rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01
FXMZC6SCFYE69K14H2Z2A9C","rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FXMZC6WDA66EH7R18MGD23MQF"],"received_at":"2022-03-08T14:
37:13.003702798Z","uplink_message":{"session_key_id":"AX7d4Kd00GgGq7h2+290/Q==","f_port":2,"f_cnt":18502,"frm_payload":"AdZ
mTBNO","rx_metadata":[{"gateway_ids":{"gateway_id":"dasya-purple-01","eui":"B827EBFFFE51B07E"},"timestamp":497052315,"rssi"
:-19,"channel_rssi":-19,"snr":9,"location":{"latitude":55.65971490666779,"longitude":12.591437101364138,"altitude":30,"sour
ce":"SOURCE_REGISTRY"},"uplink_token":"Ch0KGwoPZGFzeWetcHVycGxLLTAxEgi4J+v//lGwfHcb1YHTARoMCJjUnZEGELT5k/oCIPjagdW7xhc=","c
hannel_index":4},"settings":{"data_rate":{"lora":{"bandwidth":125000,"spreading_factor":7},"coding_rate":"4/5","frequency"
:"867300000","timestamp":497052315},"received_at":"2022-03-08T14:37:12.793965590Z","consumed_airtime":"0.051456s","locatio
ns":{"user":{"latitude":55.659605433634624,"longitude":12.591425404814808,"altitude":30,"source":"SOURCE_REGISTRY"},"netwo
rk_ids":{"net_id":"000013","tenant_id":"ttn","cluster_id":"ttn-eu1"}}}
```

Integrations, webhooks, endpoints

**We know how to move payloads around -
either big ones on non-constrained networks (e.g. Wi-Fi, LTE, 5G),
or minimized ones on constrained ones (LPWAN, LowPAN).**

We still need the mechanism to parse and store those payloads.

A json from the LoRaWAN app server

```
v3/dasya-co2-001@ttn/devices/eui-70b3d54990564b35/up {"end_device_ids":{"device_id":"eui-70b3d54990564b35","application_ids":{"application_id":"dasya-co2-001"},"dev_eui":"70B3D54990564B35","join_eui":"70B3D57ED003CEC0","dev_addr":"260B0DB0"},"correlation_ids":["as:up:01FXQ33WZPWNGQ26QHZAACKK92","gs:conn:01FXHXCX0EC5EZKBMTC7FZB585","gs:up:host:01FXHXCXY5T0N5R7J863FFH9MNM","gs:uplink:01FXQ33WS1ZK5X8B2Z0AQ6XFN4","ns:uplink:01FXQ33WS2C870ZNAZ5BA08QB3"],"rpc:/ttn.lorawan.v3.GsNs/HandleUplink:01FXQ33WS26TVAH4X618ZEM0ZR","rpc:/ttn.lorawan.v3.NsAs/HandleUplink:01FXQ33WZPMR92PX4SZRE46QX7"},"received_at":"2022-03-09T10:20:41.846775520Z","uplink_message":{"session_key_id":"AX7d4KdO0GgGq7h2+290/Q==","f_port":2,"f_cnt":19061,"frm_payload":"AgdkUBMU","rx_metadata":[{"gateway_ids":{"gateway_id":"dasya-purple-01","eui":"B827EBFFFE51B07E"},"timestamp":2786380707,"rssi":-22,"channel_rssi":-22,"snr":9.8,"location":{"latitude":55.65971490666779,"longitude":12.591437101364138,"altitude":30,"source":"SOURCE_REGISTRY"},"uplink_token":"Ch0KGwoPZGFzeWEtcHVycGxILTAXEgi4J+v//lGwfHcJl9OwChoMCPn+oZEGEKi+ga4CILjp7YmM2Sc="}], "settings":{"data_rate":{"lora":{"bandwidth":125000,"spreading_factor":7},"coding_rate":"4/5","frequency":"868100000","timestamp":2786380707},"received_at":"2022-03-09T10:20:41.634389195Z","consumed_airtime":"0.051456s","locations":{"user":{"latitude":55.659605433634624,"longitude":12.591425404814808,"altitude":30,"source":"SOURCE_REGISTRY"},"network_ids":{"net_id":"000013","tenant_id":"ttn","cluster_id":"ttn-eu1"}}}}
```

A json from the LoRaWAN app server

```
"received_at": "2022-03-09T11:43:14.310905856Z",
"uplink_message": {
  "session_key_id": "AX7d4Kd00GgGq7h2+290/Q==",
  "f_port": 2,
  "f_cnt": 19100,
  "frm_payload": "AepkThLr",
  "rx_metadata": [
    {
      "gateway_ids": {
        "gateway_id": "dasya-purple-01",
        "eui": "B827EBFFFE51B07E"
      },
      "timestamp": 3443866836,
      "rssi": -20,
      "channel_rssi": -20,
      "snr": 8.8,
      "location": {
        "latitude": 55.65971490666779,
        "longitude": 12.591437101364138,
        "altitude": 30,
        "source": "SOURCE_REGISTRY"
      },
      "uplink_token": "Ch0KGwoPZGFzeWEtcHVycGx1LTxEgi4J+v//lGw",
      "channel_index": 7
    }
  ],
  "settings": {
    "data_rate": {
      "lorawan": {
        "bandwidth": 125000,
```

Integrations, webhooks, endpoints

= lots of metadata &

a tiny payload: AgdkUBMU

which is base64 encoded (not encrypted!).

use script or tool to

decode to hex: 02 07 64 50 13 14

online tool: <https://v2.cryptii.com/base64/hexadecimal>

Once we have the hex bytes

02 07 64 50 13 14

we of course need to understand
what these bytes meant.

Decoders may reside on the application server or on a receiving **endpoint**.

Webhooks are **URLs** to such **endpoints**.

Decoders by sensor companies, e.g. decentlab

master

decentlab-decoders / DL-CTD10 /

DL

decentlab

JavaScript, Docksters: do not add unit if not available (pub)

..

DL-CTD10.Docksters.json	JavaScript, Docksters: do not add unit if not available (pub)
DL-CTD10.ELEMENT-IoT.ex	remove empty unit defs (pub)
DL-CTD10.cs	remove empty unit defs (pub)
DL-CTD10.erl	erlang: fix typo (pub)
DL-CTD10.ex	remove empty unit defs (pub)
DL-CTD10.js	JavaScript, Docksters: do not add unit if not available (pub)
DL-CTD10.lua	Improve display names (pub)
DL-CTD10.php	remove empty unit defs (pub)
DL-CTD10.py	remove empty unit defs (pub)

source: <https://github.com/decentlab/decentlab-decoders/tree/master/DL-CTD10>

Summary: Integrations

A vast choice of integrations, connectors, decoders, libraries, APIs ...

Main challenge is to keep the overview :)

TIG stack

One possible and popular option - the TIG Stack
with alternatives

Telegraf

[Node-Red, any scripting language](#)

InfluxDB

SQL Dbs (any)

Timeseries DBs:

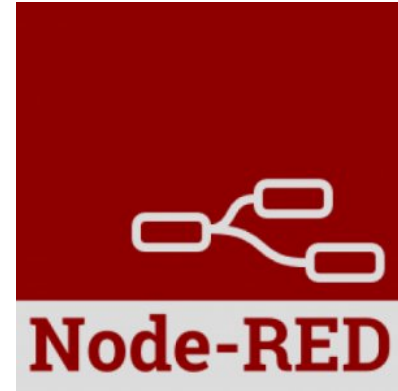
Timescale, Prometheus, ElasticSearch
plain text, custom formats

Grafana

Kibana, Tableau, Power BI, ...

Node-RED /1

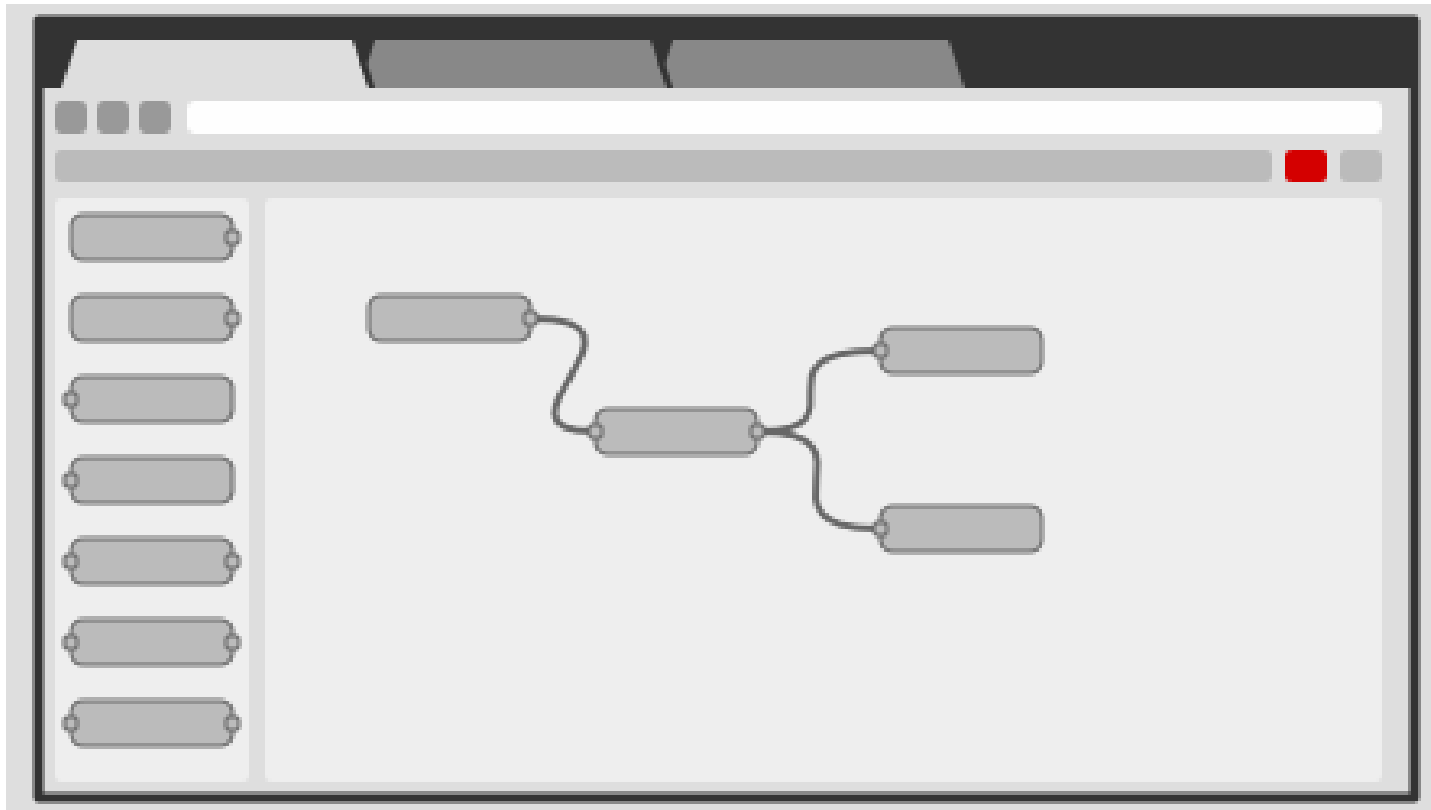
Node-RED is a
graphical tool for wiring together
hardware devices, APIs and online services
in new and interesting ways.



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**

Node-RED /2 principle



nodes flows

Node-RED /3 nodes

Examples of existing nodes:

Input/Output: tcp, udp, http, mqtt, ttn, ...

debug, status, inject, link, trigger

Functions: logic, analytics

Storage: e.g. databases

Social: e.g. tweet, mail

You can write your own nodes!

It is a bit like a DJs patchbay.



Node-RED /4 node config

Nodes are configured by double-clicking and editing the necessary info,

e.g.

MQTT topics,

http URLs,

TheThingsNetwork applications and security keys

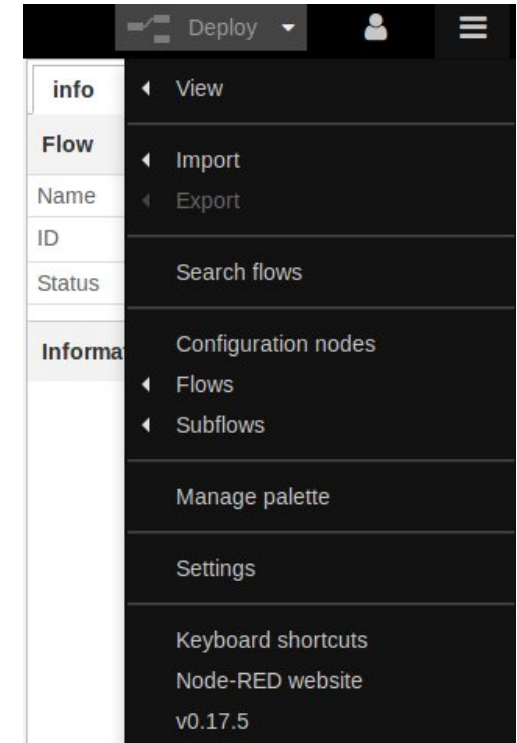
Node-RED /5 adding nodes

You may add nodes, e.g. the ttn node,
<https://flows.nodered.org/node/node-red-contrib-ttn>
via the command line, like so:

```
$ npm install node-red-contrib-ttn
```

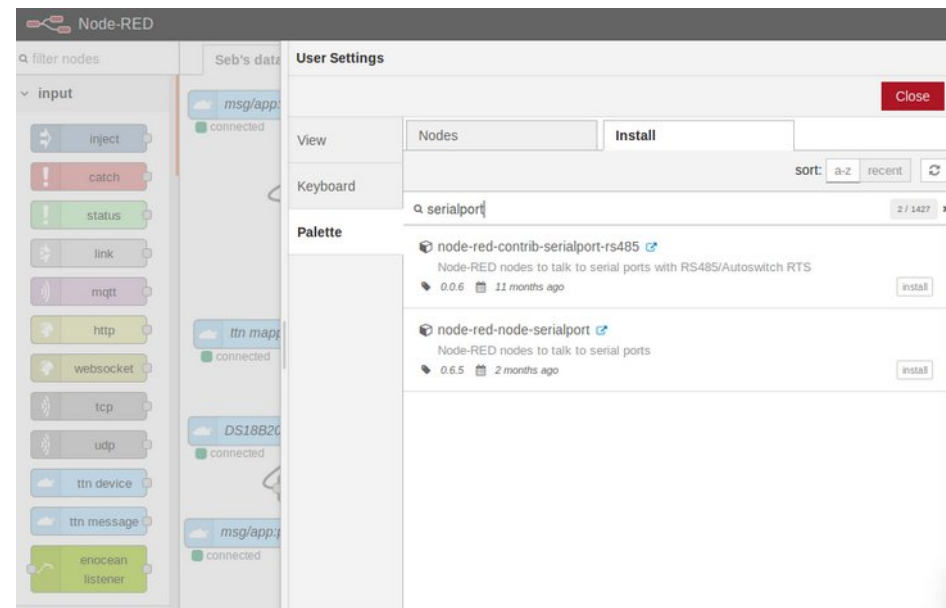
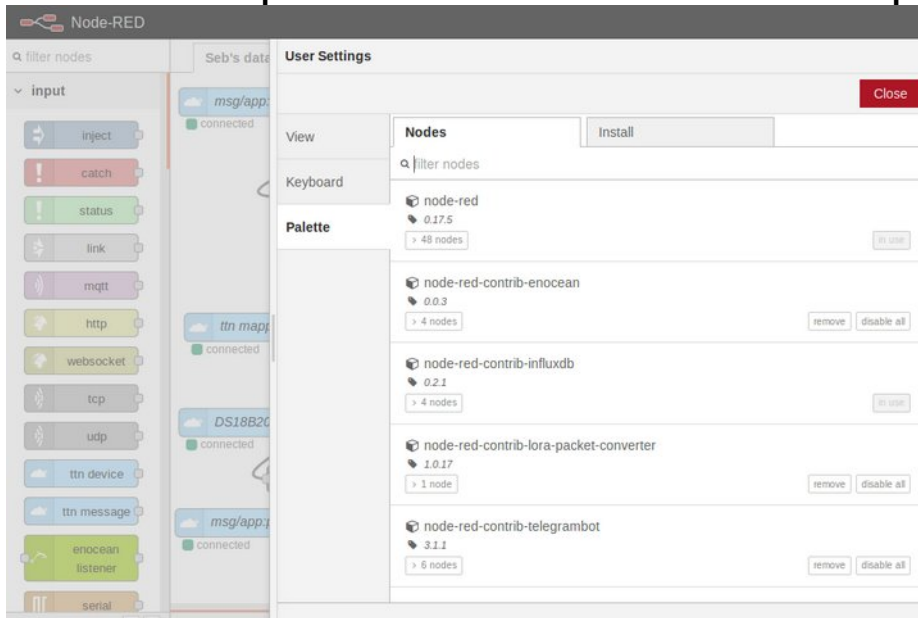
Or you can go to the node-RED menu ==>

and then ...



Node-RED /6 adding nodes

Use the palette manager to find and install new nodes, in this example,
A serial port node to read direct input over USB serial



Node-RED /7 node config ttn

Example of a TTN node:


Edit ttn device node

Delete


Cancel

Done


▼ node properties

 Name


dev/lopy-seb04

 App

pitlab-core

 Device ID

70B3D54990E351DA

 Event

up

info

debug

Node

Name	dev/lopy-seb04
Type	ttn device
ID	"e927e84.f6b3e18"

show more ▼

Information

A node to receive events from devices on The Things Network.

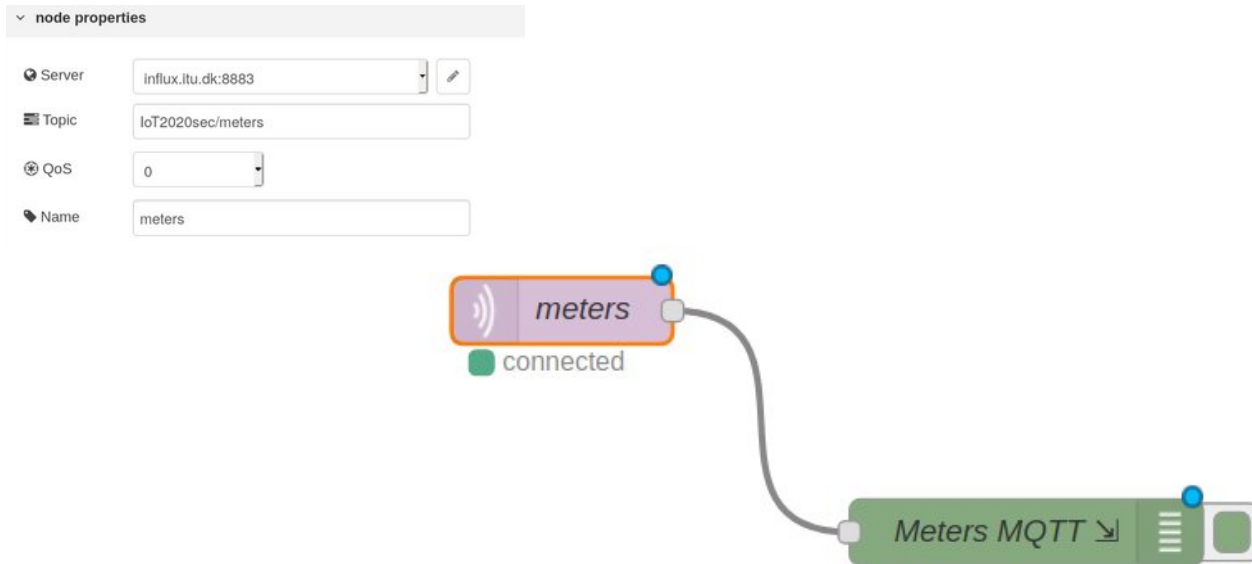
The application and event must be configured in the node. A device ID to filter on can also be configured.

The output message:

- `dev_id`, the ID of the device that sent the message.
- `payload`, the original [MQTT events](#)

Node-RED /8 node config MQTT

Example of a MQTT node



3/29/2020, 7:21:01 PM node: Meters MQTT

IoT2020sec/meters : msg.payload : string[12]

"gF6A2PwAEA=="

3/29/2020, 7:21:01 PM node: Meters MQTT

IoT2020sec/meters : msg.payload : string[12]

"gV6A2PwG7Q=="

3/29/2020, 7:21:02 PM node: Meters MQTT

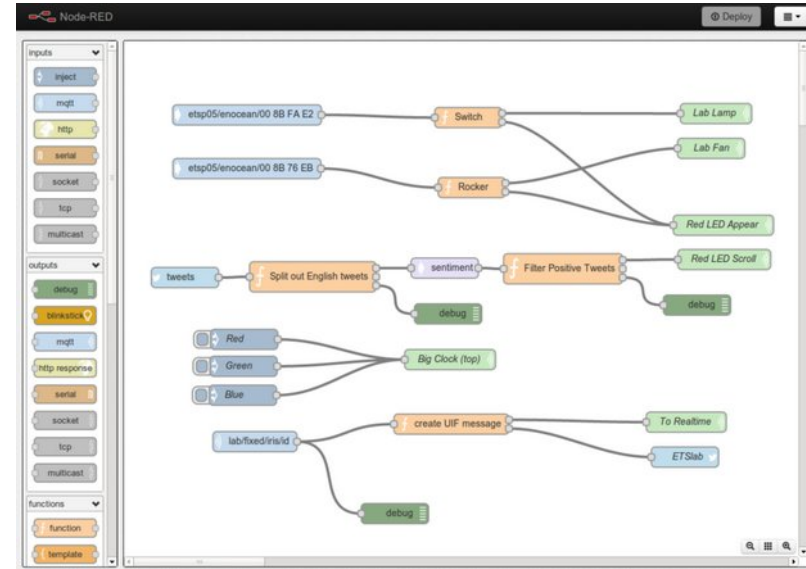
IoT2020sec/meters : msg.payload : string[12]

"g16A2PwFFw=="

Node-RED /9 flows

Flows are combinations of inputs, outputs, connections, Actions, etc which you can share and reuse.

e.g. receive sensor messages, do calculations on the values, keep averages or deltas, put them in a database, trigger an actuator, and inform the owner via messages.



Node-RED /10 text

While this is a graphical tool,
behind the scenes it s all text files

```
[{"id":"e7fb9fb.21ef46","type":"tab","label":"Seb's data flows","disabled":false,"info":"","id":"fe383a2e.538d78","type":"ttn app","z":"","appId":"pitlab-ds18b20","region":"eu","accessKey":"ttn-account-v2.07-uy5QHbEbvj0v65hV_D14chfWPAC0tWTGNQoiGYno"},{"id":"7714594.09e8ca8","type":"ttn app","z":"","appId":"pitlab-ds18b20","region":"eu","accessKey":"ttn-account-v2.07-uy5QHbEbvj0v65hV_D14chfWPAC0tWTGNQoiGYno"},{"id":"2e1a5f7.ecc9aa","type":"influxdb","z":"","hostname":"127.0.0.1","port":"8086","protocol":"http","database":"pit001","name":"","usetls":false,"tls":"","id":"132c2fb3.7d3c","type":"ttn app","z":"e7fb9fb.21ef46","appId":"","region":"","accessKey":"","id":"4e72e8d2.31e58","type":"ttn app","z":"","appId":"pitlab-core","region":"eu","accessKey":"ttn-account-v2.XLXDZg4cTdT0X-4obr4F3A5StKGLt49VeqBMQL-QIRo"},{"id":"c433206c.cb88b","type":"mqtt-broker","z":"","broker":"influx.itu.dk","port":"1883","clientId":"","usetls":false,"compatnode":true,"keepalive":"60","cleansession":true,"willTopic":"","willQos":"0","willPayload":"","id":"7ddf9c9c.690d7c","type":"ttn app","z":"","appId":"pitlab-test-seb-201804","region":"eu","accessKey":"ttn-account-v2.UNQlc1102r44inJVyb9fgkVPEgNlj80wN8tzY8Lrmg"},{"id":"bbe92aff.b8c38","type":"ttn app","z":"","appId":"pitlab-seb-temperature-humidity","region":"eu","accessKey":"ttn-account-v2.d1HGfBctUyngVrcXC0_08V2Qw4meJ_M7VC489qk5Kg"},{"id":"ebf65f2f.3c1748","type":"ttn app","z":"","appId":"dk-cph-itu-pitlab-01","region":"eu","accessKey":"ttn-account-v2.2ohV653kNtCAOPsm0F9u9K1BmAJP3Mq62Ywv6FYzdm"},{"id":"e927e84.f6b3e18","type":"ttn device","z":"e7fb9fb.21ef46","name":"dev/lopy-seb04","app":"4e72e8d2.31e58","devId":"7083D5499E3510A","event":"up","x":"96.53570556640625","y":"715.321533203125","wires":[["71231f80.fdba38"]]},{"id":"6e482785.6e1468","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"tindiestic","precision":"","retentionPolicy":"","x":"1180","y":"40","wires":[],out":"z":"e7fb9fb.21ef46","type":"ttn message","z":"e7fb9fb.21ef46","name":"msg/app:dk-cph-itu-pitlab-01","app":"ebf65f2f.3c1748","devId":"","field":"","x":"129","y":"31","wires":[["5ff3781c.a6152","2d37e02c.830c28","1db6ebe3.327dec"]]},{"id":"2d37e02c.830c28","type":"function","z":"e7fb9fb.21ef46","name":"function 3 - 3 sticks and waterTemp","func":"node.log (\\function 3 called\\)\\nvar msgAll = { payload: msg.payload};\\nvar msg0 = { payload: msg.payload[0] };\\nvar msg1 = { payload: msg.payload[1] };\\nvar msg2 = { payload: msg.payload[2] };\\nvar msg3 = { payload: msg.payload[3] };\\nvar msg4 = { payload: msg.payload[4] };\\nvar msg5 = { payload: msg.payload[5] };\\nvar msg6 = { payload: msg.payload[6] };\\nvar msg7 = { payload: msg.payload[7] };\\nvar msg8 = { payload: msg.payload[8] };\\nvar msg9 = { payload: msg.payload[9] };\\nvar msg10 = { payload: msg.payload[10] };\\nvar msg11 = { payload: msg.payload[11] };\\nvar msg12 = { payload: msg.payload[12] };\\nreturn [ msg0, msg1, msg2, msg3, msg4, msg5, msg6, msg7, msg8, msg9, msg10, msg11, msg12 ];\\n","outputs":"13","noerr":0,"x":"462.857177734375","y":"145.71426391601562","wires":[["109102f9.a35295","f464746b.2c73c"],["f6162812.22b968"],["2ce784ac.12558c"],["195b976d.dd29e9"],["f689d8d2.af84b8"],["12d1fc81.2209a3"],["62791b96.df9c84"],["863f73e.2f8189"],["10afid28.2aabcf"],["a23ae5df.b2d2c8"],["5e7d504e.3905e8"],["8c5135df.f6a148"],["f4314019.a81c58"]]},{"id":"687145e1.4c21bc","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"lopySeb02","precision":"","retentionPolicy":"","x":"1180","y":"80","wires":[],out":"z":"5ff3781c.a6152","type":"function","z":"e7fb9fb.21ef46","name":"function 04","func":"node.log (\\function 04 called\\)\\nvar msg1 = {payload: msg.payload[2]};\\nreturn [ msg1];\\n","outputs":"1","noerr":0,"x":"390.00080305175781","y":"24.28571319580078","wires":[[]]},{"id":"f6162812.22b968","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-01-moist","precision":"","retentionPolicy":"","x":"1272","y":"182","wires":[],out":"z":"109102f9.a35295","type":"influxdb out","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-01-temp","precision":"","retentionPolicy":"","x":"1173","y":"142","wires":[[]]},{"id":"7ce784ac.12558c","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-01-light","precision":"","retentionPolicy":"","x":"1257","y":"224","wires":[],out":"z":"f4314019.a81c58","type":"influxdb out","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-05-temp","precision":"","retentionPolicy":"","x":"1264","y":"622","wires":[[]]},{"id":"12d1fc81.2209a3","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-02-light","precision":"","retentionPolicy":"","x":"1257","y":"344","wires":[],out":"z":"689d8d2.af84b8","type":"influxdb out","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-02-moist","precision":"","retentionPolicy":"","x":"1264","y":"302","wires":[[]]},{"id":"195b976d.dd29e9","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-02-temp","precision":"","retentionPolicy":"","x":"1267","y":"264","wires":[],out":"z":"10afid28.2aabcf","type":"influxdb out","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-03-lab-h","precision":"","retentionPolicy":"","x":"1267","y":"464","wires":[[]]},{"id":"9c3c773a.7c6f80","type":"influxdb","z":"e7fb9fb.21ef46","influxdb":"2e1a5f7.ecc9aa","name":"","measurement":"003-03-lab-h","precision":"","retentionPolicy":"","x":"1267","y":"464","wires":[[]]}]
```

which you can access via shell, edit, export, import, ...

Node-RED /11 Try it!

Local install (best! requires nodejs)

or

free IBM instance

or

training.itu.dk (ask for access!)

Telegraf /1

Alternative to Node-Red:

Telegraf “is a popular open-source agent for collecting, processing, aggregating, and sending metrics and events to various monitoring systems. Telegraf is widely used and highly configurable.”

[ChatGPT, <https://chat.openai.com/c/31d15754-ed89-4e81-8b74-324cbf0af5be> prompt: “what are alternatives to telegraf?” 20240306]

Large library of plugins (100+) for data collection.



Telegraf /2 Configuration

```
[agent]
hostname = "localhost"
flush_interval = "15s"
interval = "15s"
[[inputs.mqtt_consumer]]
servers = ["tcp://eu.thethings.network:1883"]
qos = 0
connection_timeout = "30s"
topics = [ "+/devices/+/up" ]
client_id = ""
username = "your application name"
password = "ha! not gonna tell you"
data_format = "json"
[[outputs.influxdb]]
database = "your database"
urls = [ "http://localhost:8086" ]
```



Ingestion, custom /1

Any modern scripting or programming language
has the integrations, libraries etc
needed to connect -

e.g.

In: http, MQTT in

Out: to database.

python:

MQTT paho-mqtt **<https://pypi.org/project/paho-mqtt/>**

influxdb **<https://github.com/influxdata/influxdb-client-python>**

Time Series Databases (TSDB) /0

Time Series Databases are a form of NoSQL databases, non-relational databases

though in fact many are very near-SQL



Time Series Databases (TSDB) /1

Time series databases are different from relational databases -

Note the benefits, but also fundamental trade-offs!

- Schema-less: no definition of "table" schemes - just write! Any *write* creates a "table"
- Like an SQL database table where the primary key is pre-set by the system and is always time.
- Time stores timestamps, in RFC3339 UTC, precision might vary
- Time is assumed to be ascending – and only ascending!
- Time series data is predominantly new data that is never updated. UPDATE and DELETE are very limited / impossible.

Time Series Databases (TSDB) /1

How time series databases are different from relational databases:

- To simplify conflict resolution and increase write performance, data sent multiple times is seen as duplicate data. Identical points aren't stored twice.
 - If temperature at 15:00 is 21 degrees, it s not 23 degrees!
- Provide time specific functions, such as now()
 - `SELECT * FROM "meters" WHERE time > now() - 1h`

InfluxDB

- ▶ Time Series database
- ▶ “time” is a special value, has special meaning
- ▶ Applies special logic to “time”

SMART!

MySQL

- ▶ Generic relational database
- ▶ “time” is a generic data value
- ▶ Doesn't apply special logic

DUMB!

InfluxDB

```
“SELECT count(bar1) FROM foo  
WHERE bar1 > 0 AND bar2 > 0 AND  
time > now() - 7d GROUP BY time(1h)”
```

<https://www.slideshare.net/PhilipWernersbach/grafana-and-mysql-benefits-and-challenges>

MySQL

```
“SELECT time, count(bar1) FROM foo  
WHERE bar1 > 0 AND bar2 > 0 AND  
time > NOW(6) - INTERVAL 7 DAY  
GROUP BY YEAR(time), MONTH(time),  
DAY(time), HOUR(time) ORDER BY time  
ASC”
```

<https://www.slideshare.net/PhilipWernersbach/grafana-and-mysql-benefits-and-challenges>

InfluxQL != SQL

- ▶ InfluxQL is SQL-like, but different enough that it can't be passed through to MySQL

<https://www.slideshare.net/PhilipWernersbach/grafana-and-mysql-benefits-and-challenges>

NoSQL – Timeseries - InfluxDB

- ▶ “time” is SELECT’d automatically implicitly in InfluxDB
- ▶ “SELECT bar FROM foo” → “SELECT time, bar FROM FOO”
- ▶ GROUP’ing on “time” is smart in InfluxDB, and dumb in MySQL
- ▶ See slides 16 and 17
- ▶ “time” in epoch format with millisecond precision is a float in MySQL
- ▶ “FROM UNIXTIME(1444667802.145)”

<https://www.slideshare.net/PhilipWernersbach/grafana-and-mysql-benefits-and-challenges>

NoSQL – Timeseries – InfluxDB: v1 vs v2

Note that the previous comments refer to InfluxQL, which – while still available in InfluxDB v2 for backwards compatibility – is being replaced by the Flux scripting language for data query, and classical SQL is offered as an alternative.*

<https://docs.influxdata.com/influxdb/v2.1/query-data/>

~~*At the time of writing (March 2022), ITU InfluxDB instances are still v1.*~~

At the time of writing (March 2024), ITU InfluxDB runs v 2.7.

Documentation: <https://docs.influxdata.com/influxdb/v2/>

NoSQL – Timeseries – InfluxDB: v1 vs v2 vs v3

** ... is being replaced by the Flux scripting language
or so we thought :) ... announcement 2023: →*

The future of Flux

Flux is going into maintenance mode. You can continue using it as you currently are without any changes to your code.

Flux is going into maintenance mode and will not be supported in InfluxDB 3.0. This was a decision based on the broad demand for SQL and the continued growth and adoption of InfluxQL. We are continuing to support Flux for users in 1.x and 2.x so you can continue using it with no changes to your code. If you are interested in transitioning to InfluxDB 3.0 and want to future-proof your code, we suggest using InfluxQL.

For information about the future of Flux, see the following:

- [The plan for InfluxDB 3.0 Open Source](#)
- [InfluxDB 3.0 benchmarks](#)

SHOW LESS

Time Series Databases (TSDB) /2

Popular TSDBs:

- InfluxDB
 - near-SQL
- Cassandra
 - Mozilla
- Timescale
 - Postgres-based, extension to SQL
- Prometheus

In what follows, we focus on InfluxDB as example.

Comparison/Choice of TSDBs:

- Often, choices tend to be culturally determined rather than tech-driven
- Depending on choice, you will model data in different ways
 - Example: room booking – how is this best represented?
- Integration with analytics / machine learning concerns may be a determining factor

Time Series Databases (TSDB) /4

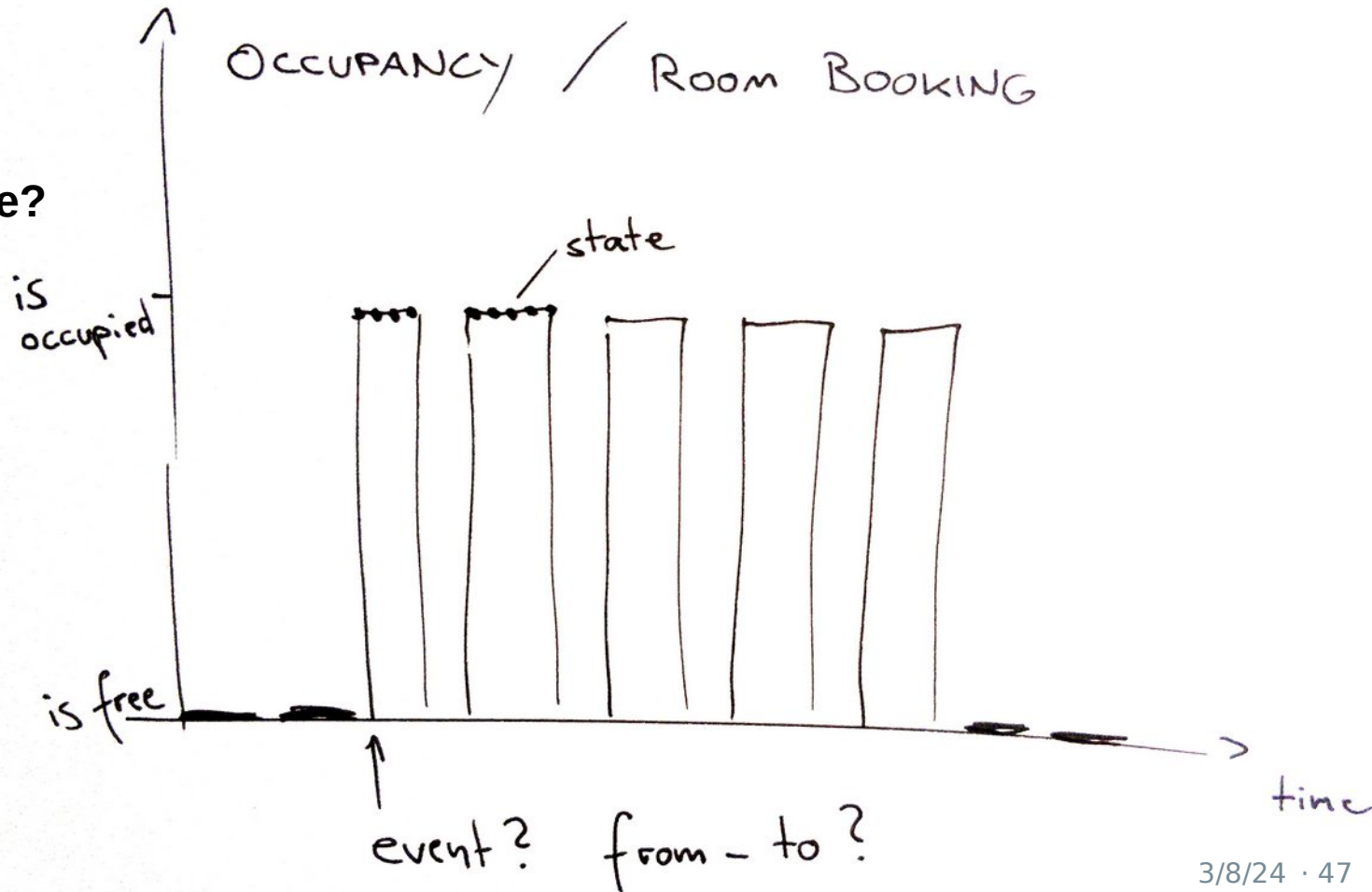
Example:

prioritize

... performance?

... data size?

... structure?



InfluxDB is an **open-source time series database** developed by InfluxData.

It is written in **Go** and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, **Internet of Things sensor data**, and real-time analytics.

Open Source: Each component (except for some ...) of the InfluxData platform or TIG Stack is on github and available via a simple download.

The company Influxdata is venture funded.



SQL-like language with builtin **time-centric** functions for querying a data structure composed of measurements, series, and points.

Each point consists of several **key-value pairs** called the fieldset and a **timestamp**. When grouped together by a set of key-value pairs called the tagset, these define a series. Finally, series are grouped together by a string identifier to form a **measurement**.

```
measurement(,tag_key=tag_val)* field_key=field_val(,field_key_n=field_value_n)*  
(nanoseconds-timestamp)
```

Values can be 64-bit integers, 64-bit floating points, strings, and booleans.

Main differences to a classical SQL database or NoSQL database:

It s for **time series** - nothing else!

A "table" (called measurement here) has 2 “columns”, not more *:
a **timestamp** and the **value** for that point in time.
(You can add optional “tags” to create some structure).

You would not keep an address database or such in InfluxDB.

* it s actually possible, but not advised to do so: <https://stackoverflow.com/questions/45368535/influxdb-single-or-multiple-measurement#45545405>

InfluxDB /4 terminology in-depth

Relational

vs

TSDB

database

database / “bucket”

table

measurement

index

tag key

column

field key (or tag)

field key - field value

row

points

data record = a set of

timestamp, field key, field tag and values

series: all data within one measurement

sharing the same tag set

InfluxDB /5 tags

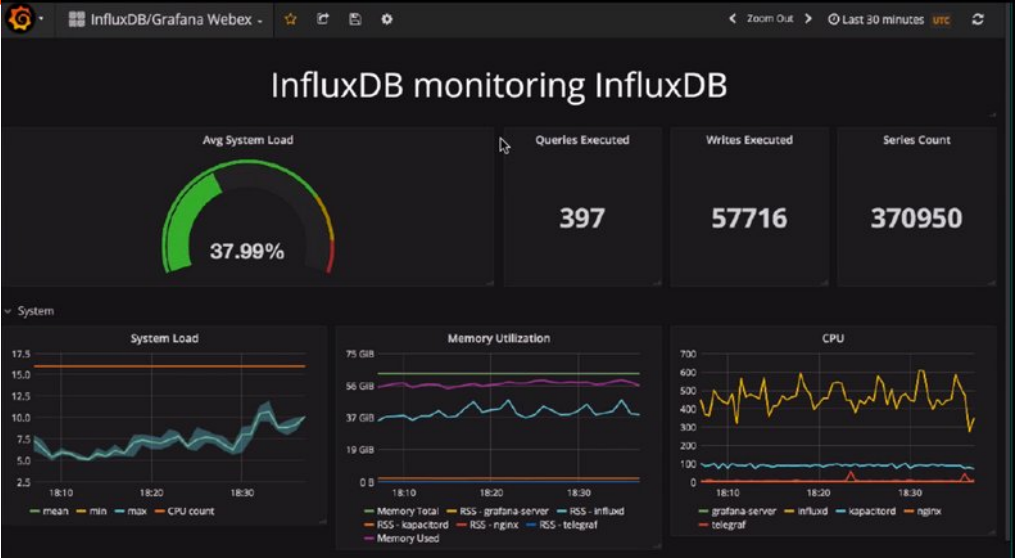
Tags are optional.

You don't need to have tags in your data structure, but it's generally a good idea to make use of them because, unlike fields, tags are indexed. This means that queries on tags are faster and that tags are ideal for storing commonly-queried metadata.

Properties that you often query for should best be tags:

e.g. all values for meter with ID=3, all sensors from room=3A52

InfluxDB /5.a example



```
adeunis_sb
adeunismapper
> select * from adeunismapper
name: adeunismapper
time                RSSI_DL SF      SNR_DL application  counter geohash  lat                lon                name
-----
1567082773119995678 -47      SF7BW125 7      adeunismapper  16      gcmzu7j54 53.410583333333335 -2.970333333333335 adeunisarf8123aa
1567082775811795699 -54      SF7BW125 7      adeunismapper  17      gcmzu7j54 53.410583333333335 -2.970333333333335 adeunisarf8123aa
1567082778335478066 -57      SF7BW125 7      adeunismapper  18      gcmzu7j54 53.410566666666667 -2.970333333333335 adeunisarf8123aa
1567082782067048465 -57      SF7BW125 7      adeunismapper  18      gcmzu7j54 53.410566666666667 -2.970333333333335 adeunisarf8123aa
1567082785251096765 -55      SF7BW125 8      adeunismapper  19      gcmzu7j5n 53.41055          -2.9701666666666666 adeunisarf8123aa
```

InfluxDB /6 additional

aggregations time-specific, such as medians, averages, etc

batches batches of points, for mass processing

retention how long we are keeping data

InfluxDB automatically creates the autogen retention policy with an infinite duration, replication factor = 1

```
> SHOW RETENTION POLICIES
```

name	duration	shardGroupDuration	replicaN	default
----	-----	-----	-----	-----
autogen	0s	168h0m0s	1	true

InfluxDB /8 shard

A **shard** contains the actual encoded and compressed data, and is represented by a TSM file on disk.

Every shard belongs to one and only one shard group. Multiple shards may exist in a single shard group. Each shard contains a specific set of series.

The default shard is one week of data.

TSM = Time structures merge tree

Details on TSM, storage:

https://docs.influxdata.com/influxdb/v1.7/concepts/storage_engine/

InfluxDB /9 structure on disk

```
root@influxus:/var/lib/influxdb/data/plantower# tree
```

```
.
├─ autogen
│   ├── 716
│   │   ├── 000000004-000000002.tsm
│   │   └─ fields.idx
│   ├── 725
│   │   ├── 000000001-000000001.tsm
│   │   └─ fields.idx
│   ├── 737
│   │   ├── 000000001-000000001.tsm
│   │   └─ fields.idx
│   ├── 749
│   │   ├── 000000005-000000002.tsm
│   │   └─ fields.idx
│   └─ 762
├─ ..
└─ _series
    ├── 00
    │   └─ 0000
    ├── 01
    │   └─ 0000
    ├── 02
    │   └─ 0000
    ├── 03
    │   └─ 0000
    ├── 04
    │   └─ 0000
    ├── 05
    │   └─ 0000
```


InfluxDB /10 surprises!

you will encounter surprises! (at least I did ...):

“Disappearing” of data

You **can not store more than one point** with the same timestamp in a series! Assumption: if the same data is sent multiple times, it is the exact same data that a client just sent several times.

PRO no conflicts, ever!

CON you ll have to separate (tag) to store multiple

Deleting and updating is hard! only whole series, shards

InfluxDB /11 look inside: v1 shell: show databases

```
InfluxDB shell 0.10.0
```

```
> show databases
```

```
name: databases
```

```
-----
```

```
name
```

```
_internal
```

```
pit001
```

```
pit002
```

```
pit003
```

InfluxDB /12 look inside: v1 shell: show measurements

```
> use pit001
```

```
Using database pit001
```

```
> show measurements
```

```
name: measurements
```

```
-----
```

```
name
```

```
003-01-light
```

```
003-01-moist
```

```
003-01-temp
```

```
003-02-light
```

InfluxDB /13 look inside: v1 shell: select

```
> select * from "004-01-temp"
```

```
1525096049108783758 50
```

```
1525096124247923458 49
```

```
1525096199476486703 49
```

```
1525096274435089220 50
```

```
1525096349502370874 50
```

```
1525096424982100010 50
```

```
1525096499794455963 49
```

InfluxDB /14 look inside: v1 shell: create database

- > CREATE DATABASE meters
- > CREATE USER smartie WITH PASSWORD 'youWish'
- > GRANT ALL ON meters TO smartie

InfluxDB /15 look inside: v1 shell

```
> select * from "meters-script" where "id"='0'
```

```
name: meters-script
```

time	fakeness	id	reading
----	-----	--	-----
1585471260000000000	8	0	850
1585471320000000000	8	0	845
1585471380000000000	8	0	272
1585471440000000000	8	0	165
1585471500000000000	8	0	302

InfluxDB /16 access

Typically, you would access data via network rather than direct -

InfluxDB accepts data via **HTTP, TCP, and UDP**.

For example you could connect from

Visualizing, Monitoring, Analyzing, Querying, Alerting



As of right now, there are

41 data sources, 30 panels, 17 apps and 857 dashboards available. (2021! check now!)

Data sources include: influxDB, MySql, Postgres, Azure, ...

Grafana /2



Dashboards, Graphs, Queries:

Setting up a new one is mainly click, drag and drop.

The screenshot displays the Grafana 'New dashboard' interface. A 'New Panel' modal is open, showing a grid of panel types: Graph, Singlestat, Table, Text, Heatmap, Alert List, Dashboard list, Row, and Plugin list. The 'Graph' panel type is selected, and its configuration panel is shown below. The configuration panel has tabs for 'General', 'Metrics', 'Axes', 'Legend', 'Display', 'Alert', and 'Time range'. The 'Metrics' tab is active, showing a 'Data Source' dropdown set to 'default' and a query editor with the query: `SELECT "value" FROM "008-01-temp"`.

Alerts

let you define flexible alert conditions

and
may trigger

emails,
telegram messages,
slack messages, ...

Alert Config

Name	DS18B20 - temp only - in halfCelsius :) al...	Evaluate every	60s
------	---	----------------	-----

Conditions

WHEN	last ()	OF	query (A, 10s, now)	IS ABOVE	60	
------	---------	----	---------------------	----------	----	--

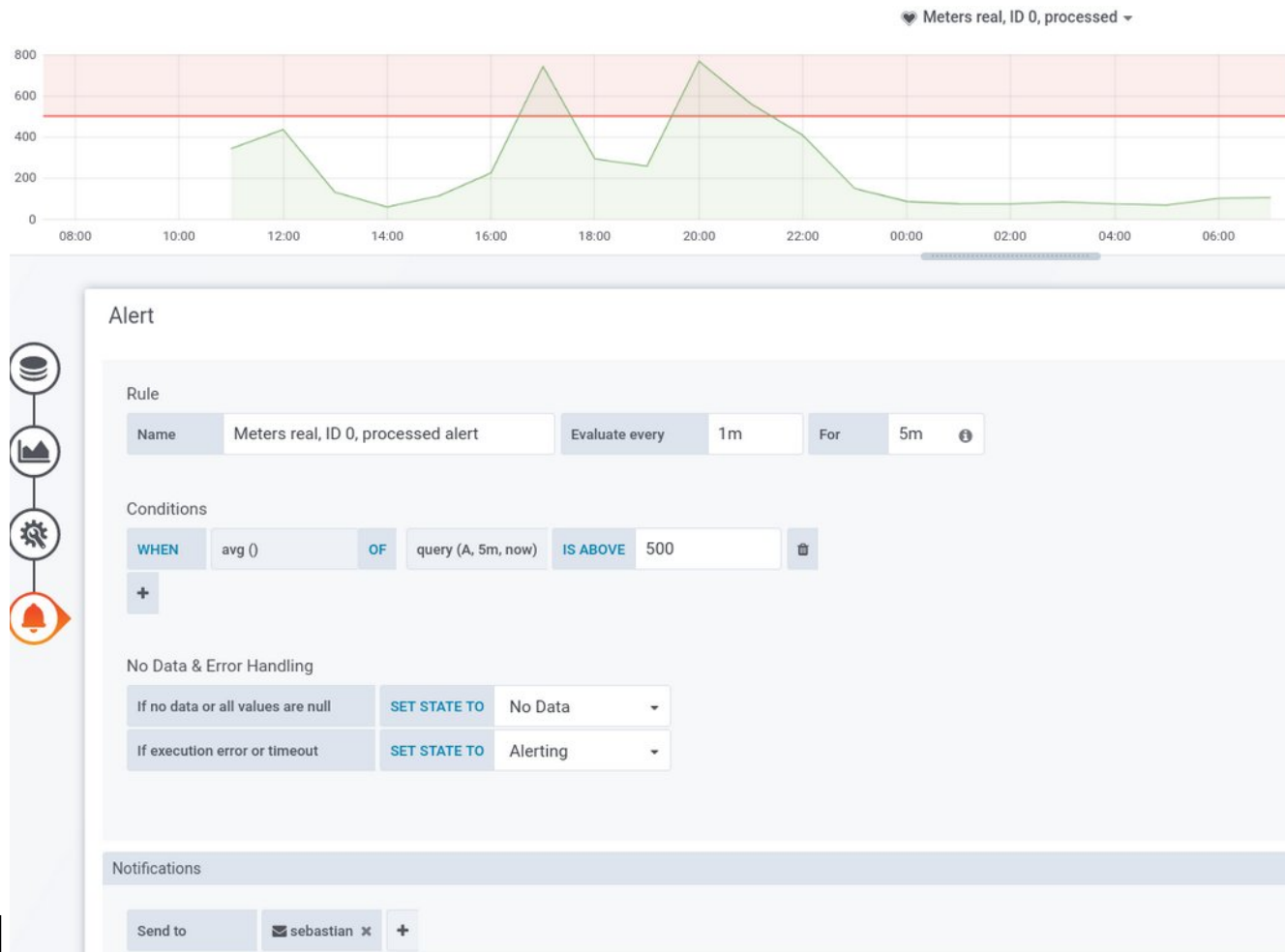
Edit Notification Channel

Name	sebAlert
Type	Telegram
Send on all alerts	<input checked="" type="checkbox"/>
Include image	<input checked="" type="checkbox"/>

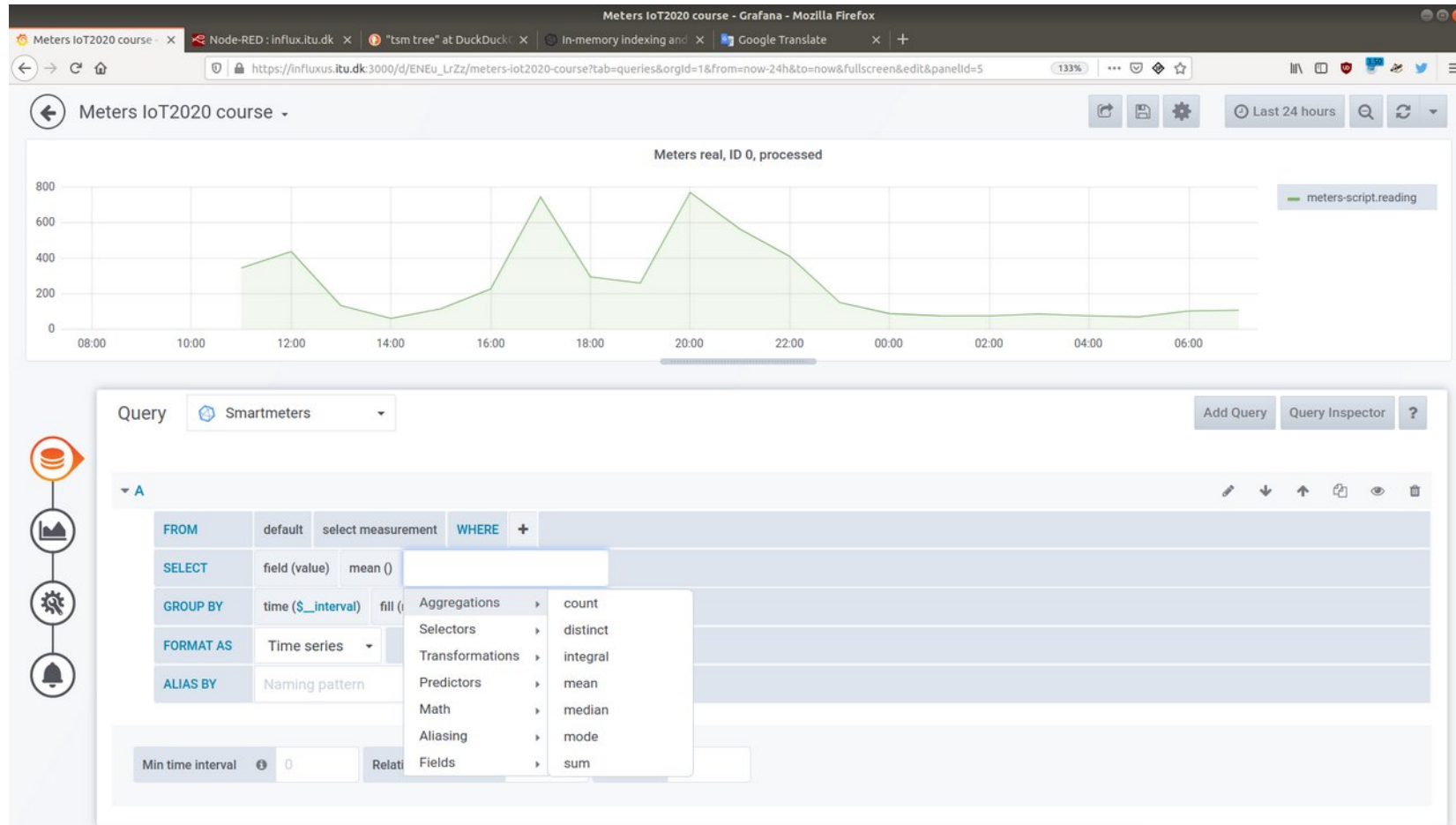
Telegram API settings

BOT API Token	539
Chat ID	923

Grafana /5 alerts



Grafana /6 aggregations, maths, predictions



Some concluding remarks

- Choices for interoperability and integrations are many ... to the extent of being overwhelming.
- Challenge: long term management
- Practical challenge for this course/exercise:
to deploy components like scripts/decoders in stable places.
- Your laptop is a great starting point, but likely not on public IP and not 24/7 available → cloud deployment

Take-aways

- Data transport
- Integrations, decoders, webhooks, endpoints
- Tools: TIG stack, node-red, ...
- Timeseries Databases and their special features
 - benefits / trade-offs
- Grafana as Data Visualizer