

Mission poulette

CAUBEL Aksel
PRUVOST Arnaud
DOMERGUE Mathys



Sommaire

- Cahier des charges et contraintes
 - Solution mise en place
 - Echange avec la classe et les enseignants
-

Cahier des charges

- Monitoring d'un poulailler (température et humidité).
- Déterminer la présence des poules dans le poulailler.
- Déterminer la présence d'animaux autour du poulailler.
- Pouvoir accéder aux différentes données.

Contraintes

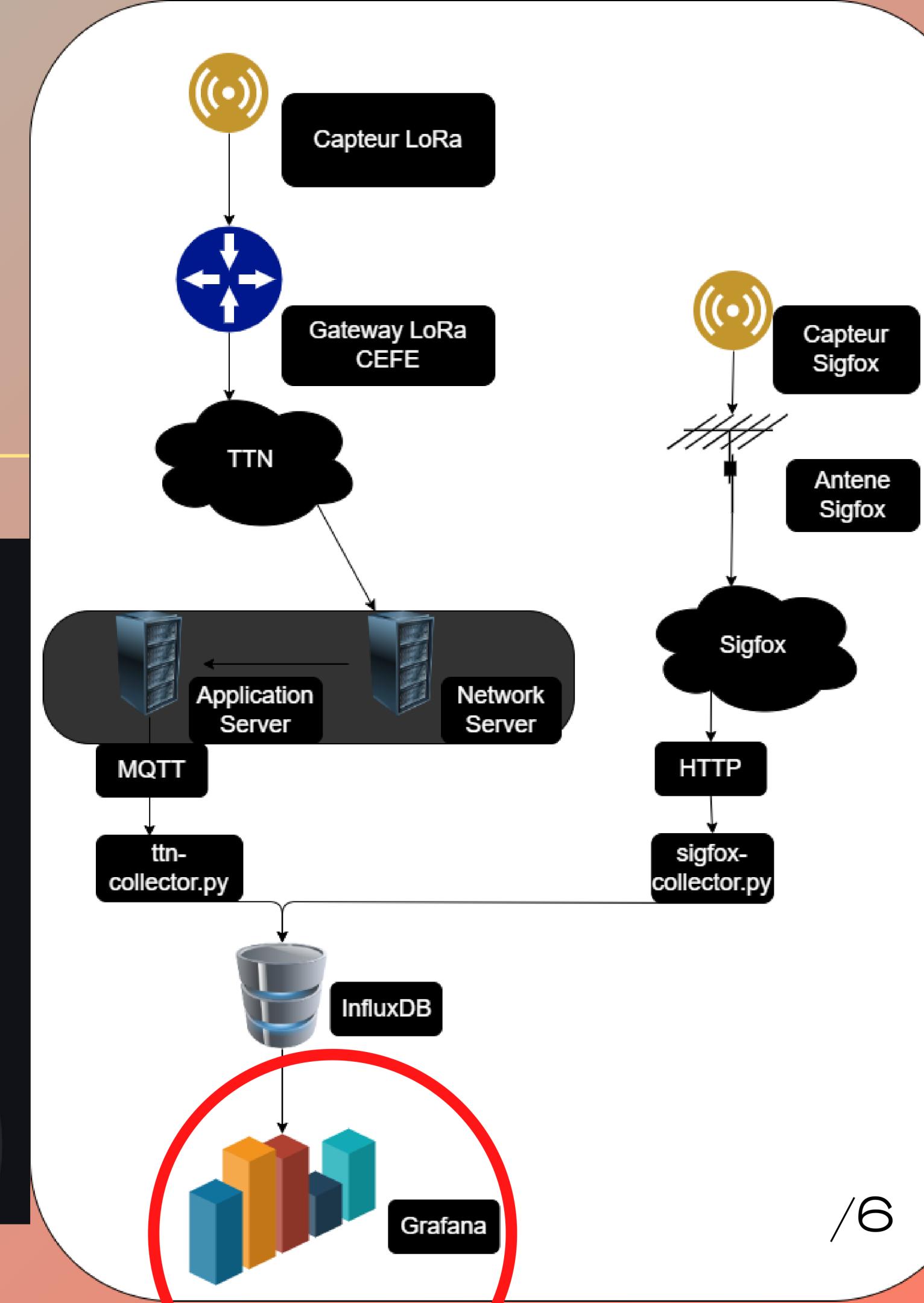
- 2 jours pour la préparation et la mise en place de l'infrastructure
- Trouver des solutions à partir des capteurs imposés
- Contraintes réseaux

Poulailler connecté

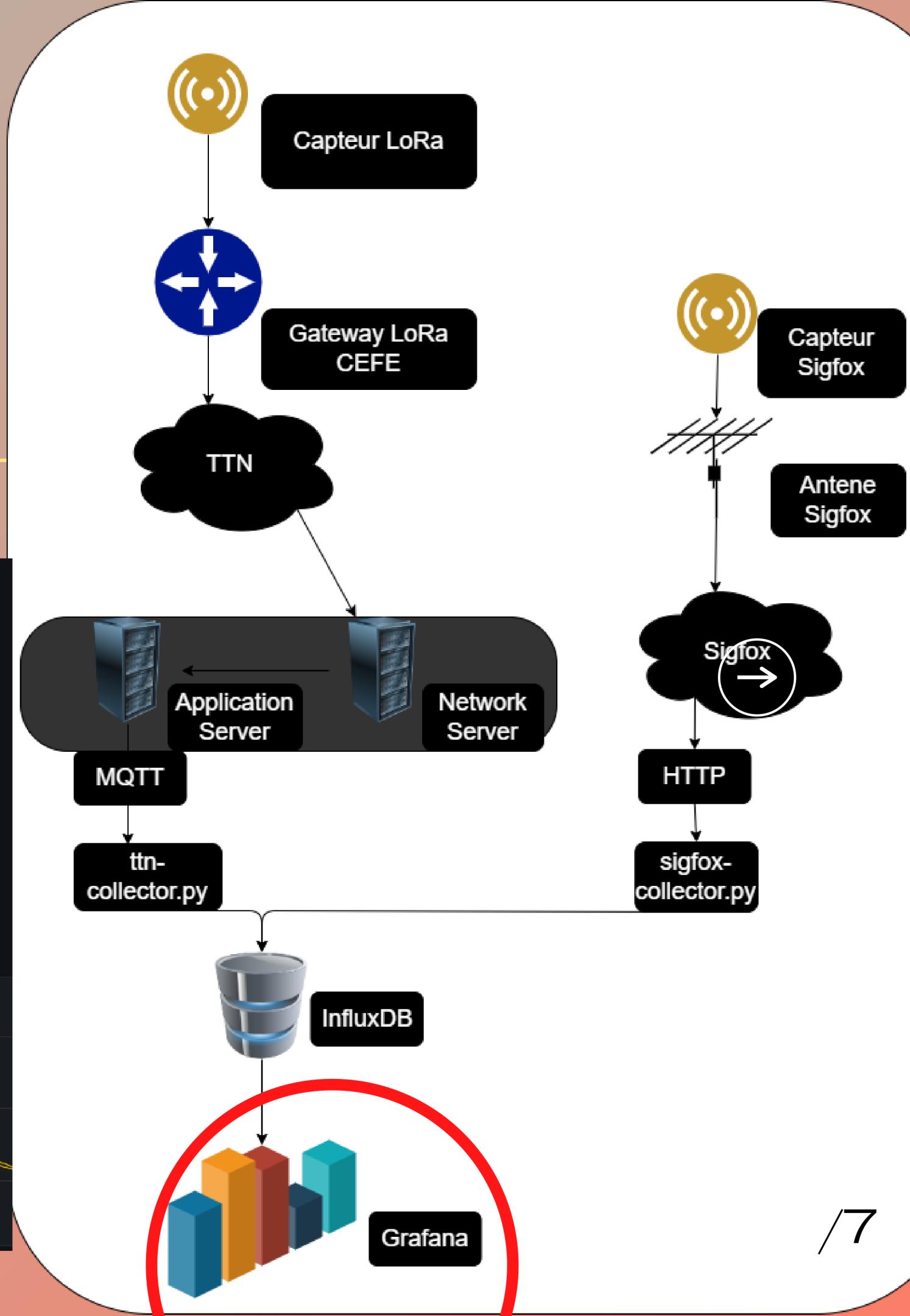
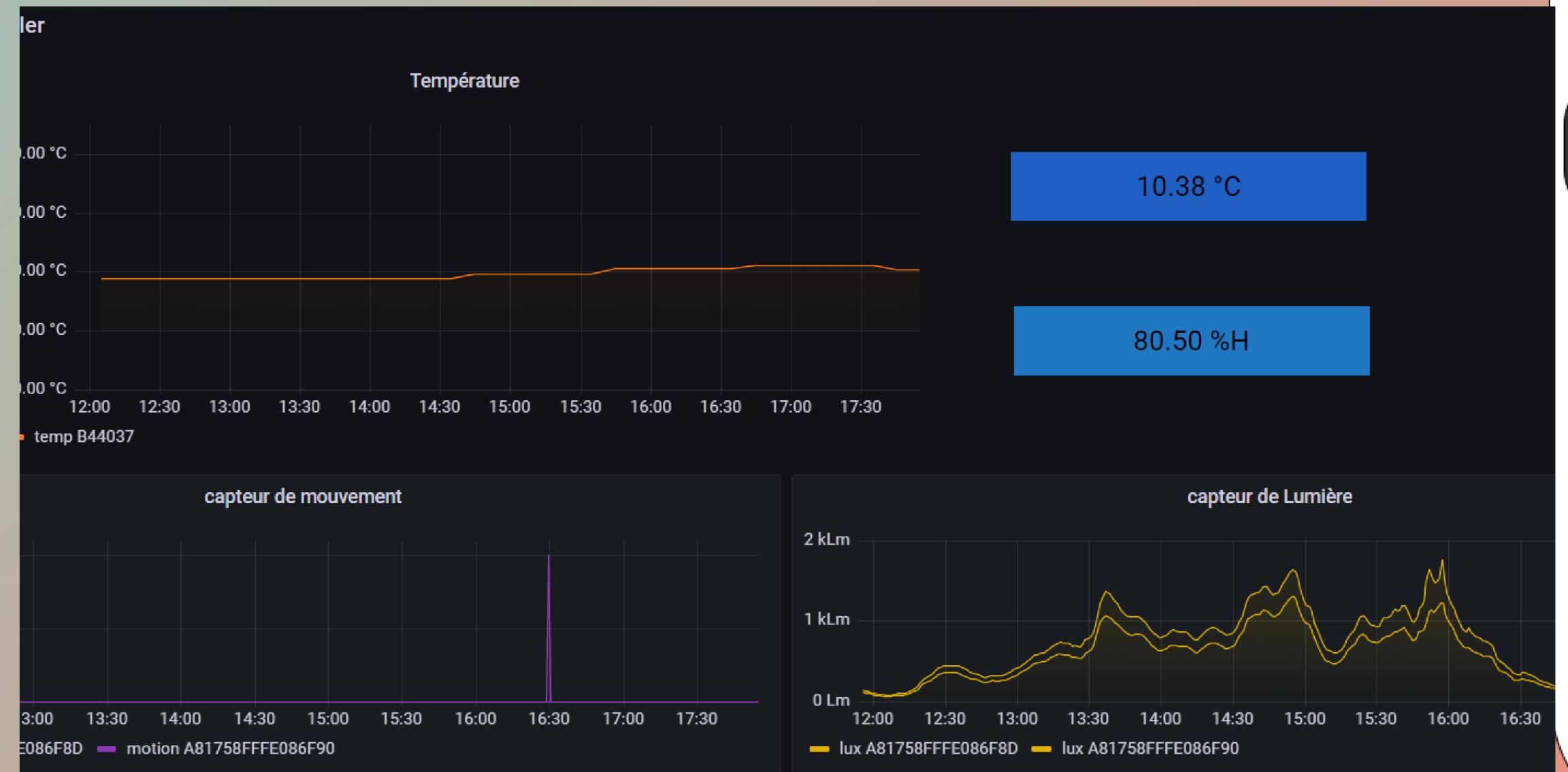


Le poulailler sur le site du CNRS, il est équipé de capteur de mouvement, de température, d'humidité, de luminosité et d'un lecteur de badge.

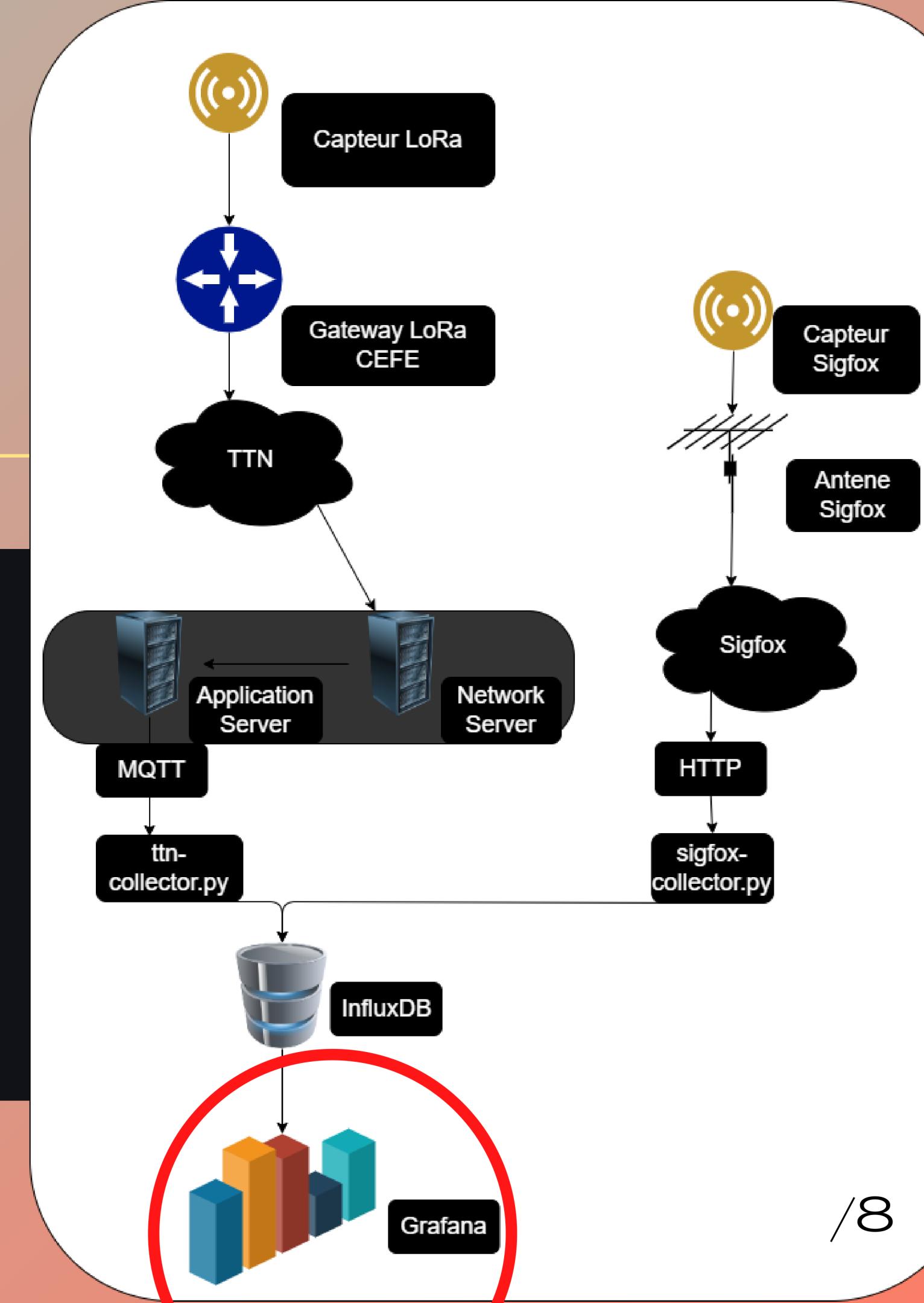
Notre solution pour répondre au cahier des charges



Notre solution pour répondre au cahier des charges



Notre solution pour répondre au cahier des charges



Notre solution pour répondre au cahier des charges

FROM

- Search buckets
- IUT_Devices
- telegraf
- _monitoring
- _tasks
- + Create Bucket

Filter

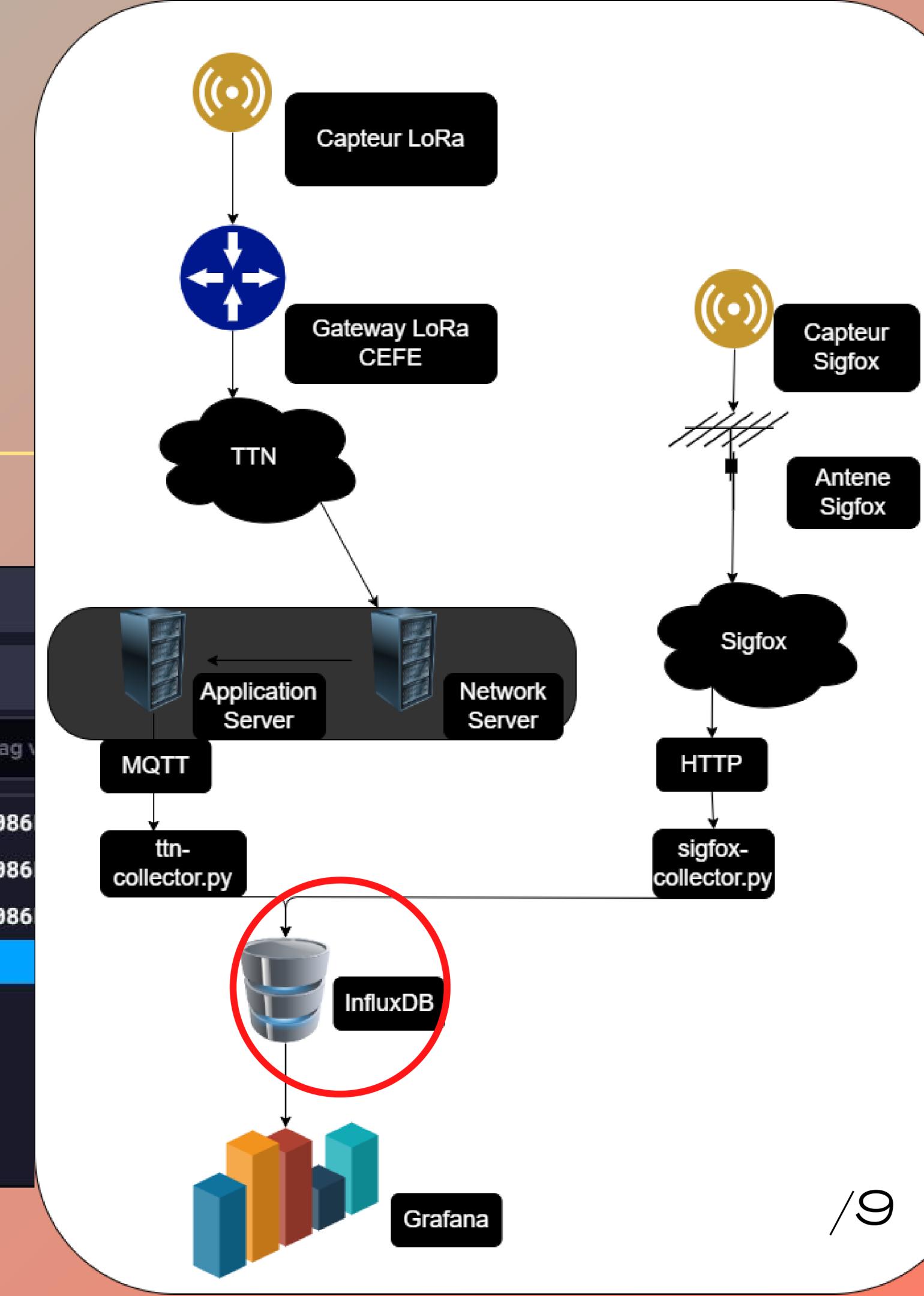
- _measurement
- _field
- device

Search _measurement tag values

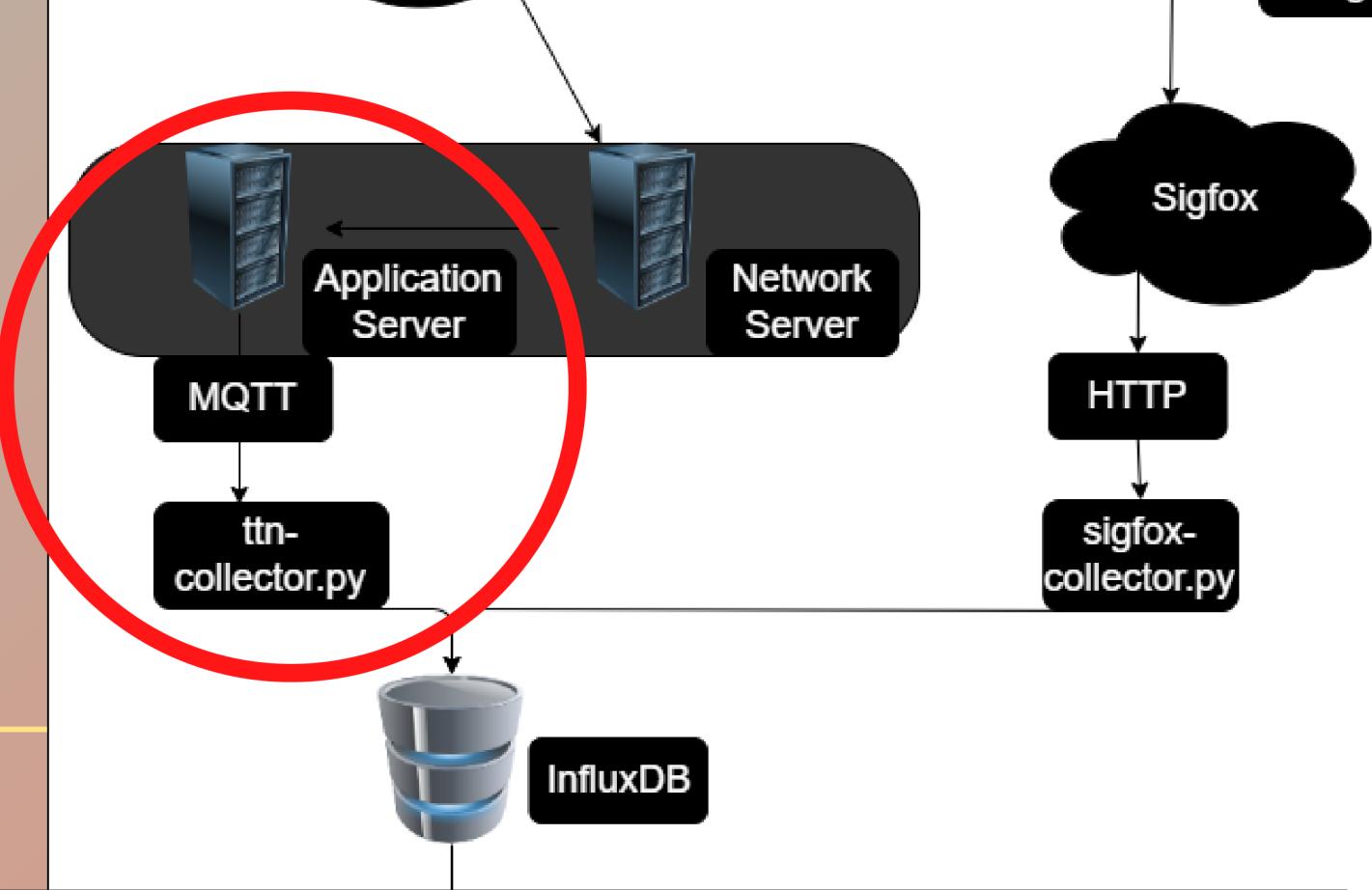
- batterie
- hum
- id
- lux
- motion
- temp
- vdd

Search _field tag values

- A81758FFFE086
- A81758FFFE086
- A81758FFFE086
- B44837
- B448E4
- B44949

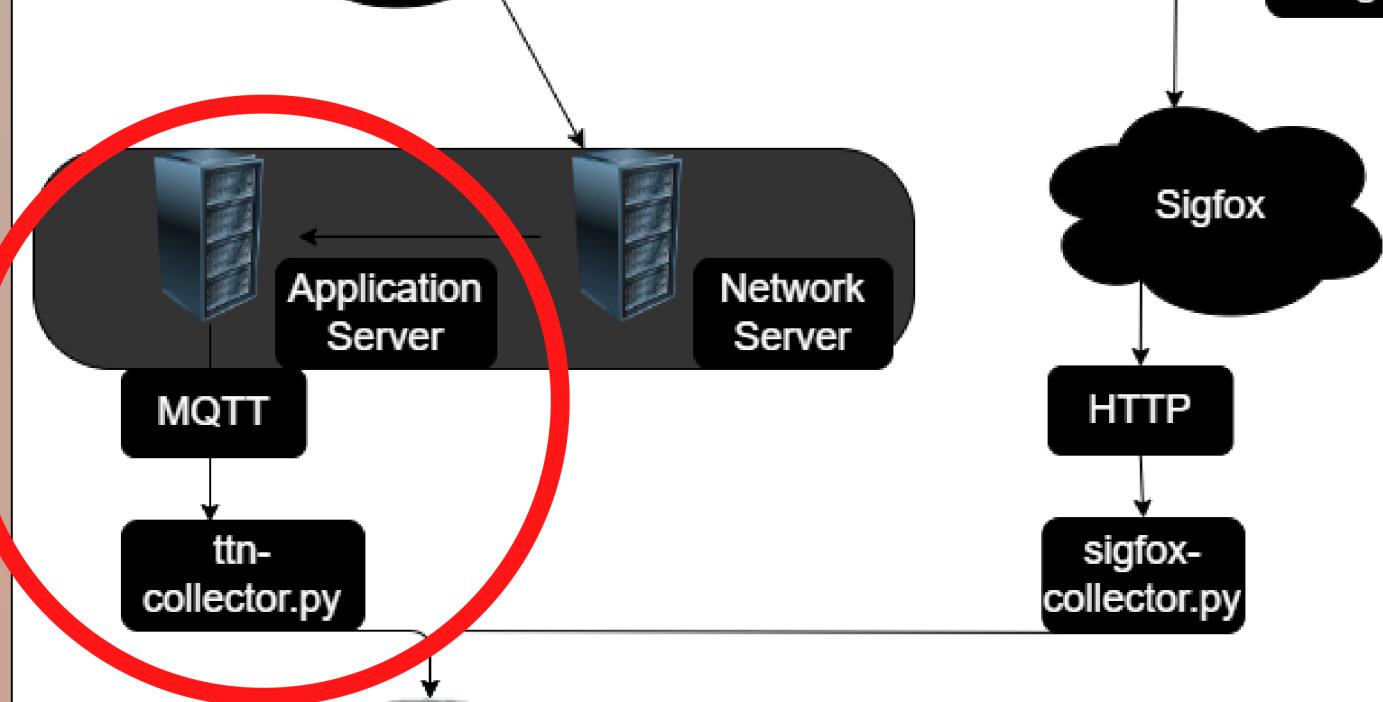


Notre solution pour répondre au cahier des charges



```
for userId, ttntoken in ttncard:  
    topic = f"v3/{userId}/devices/+/"  
  
    listener[userId] = mqtt.client()  
    listener[userId].username_pw_set(userId, password=ttntoken) # Set se login to the MQTT-Broker  
    listener[userId].connect(brokerIp,brokerPort,60)  
    listener[userId]._send_connect = print(f"listener {userId} : trying to connect...") # SIN  
    listener[userId].on_connect = print(f"listener {userId} : Connected on {brokerIp}:{brokerPort}") # SIN ACK  
    listener[userId].on_message = on_message # exec on_message when a message arrive  
    listener[userId].subscribe(topic) # listen on the topic  
    listener[userId].loop_start() # listen begin | Use a thread
```

Notre solution pour répondre au cahier des charges



```
def on_message(client, userdata, message):
    """Fonction appelé lors de l'arriver"""

    jsonData = json.loads(message.payload.decode('utf-8')) # Charge le payload sous forme JSON

    bucket = "IUT_Devices" # bucket Influx
    token = "NZuKx9nuCgkreX6PJ7jd2IdEGNVAosqmxcJUnJi944WTODCL-XJ1WXcAEHcMwmpEbML20G4P60QEr99YiU-xcg==" # Token Ecriture IUT_Devices

    client = InfluxDBClient(url="http://51.158.110.29:8086", token=token, org="iutbeziers") # Login to Influx
    write_api = client.write_api(write_options=SYNCHRONOUS) # Write mode + synchronise

    device = jsonData["end_device_ids"]["dev_eui"]

    if device == "A81758FFFE086F8D" or device == "A81758FFFE086F90" or device == "A81758FFFE086F91": # Capteur de mouvement poulailler 1 | 1 |

        hum,light,motion,temperature,vdd = jsonData.get('uplink_message').get('decoded_payload').values() # récupère les différentes valeurs

        # Init des différents relever
        h = Point("sensors").tag("device",device).field("hum",float(hum))
        l = Point("sensors").tag("device",device).field("lux",light)
        m = Point("sensors").tag("device",device).field("motion",motion)
        t = Point("sensors").tag("device",device).field("temp",float(temperature))
        v = Point("sensors").tag("device",device).field("vdd",vdd/1000)
```

Notre solution pour répondre au cahier des charges

iot51-1
ID: eui-0080000000021ba1

↑ 6,945 ↓ 125 • Last activity 11 seconds ago

1 Collaborator 1 API key

General information

Gateway ID: eui-0080000000021ba1

Gateway EUI: 00 80 00 00 00 02 1B A1

Gateway description: None

Created at: Dec 6, 2022 09:53:23

Last updated at: Dec 6, 2022 15:33:24

Gateway Server address: eu1.cloud.thethings.network

LoRaWAN information

Frequency plan: EU_863_870_TTN

Global configuration: Download global_conf.json

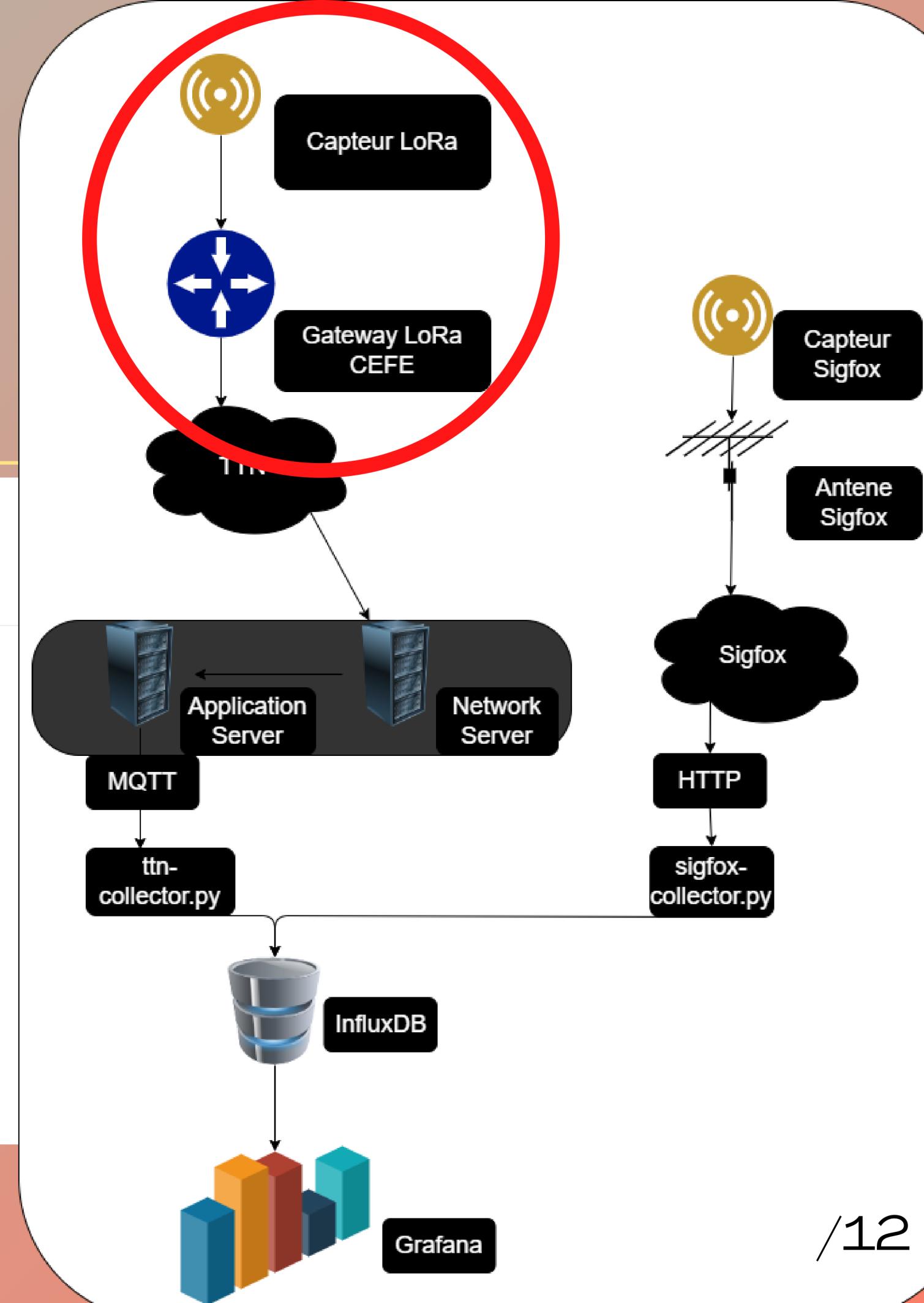
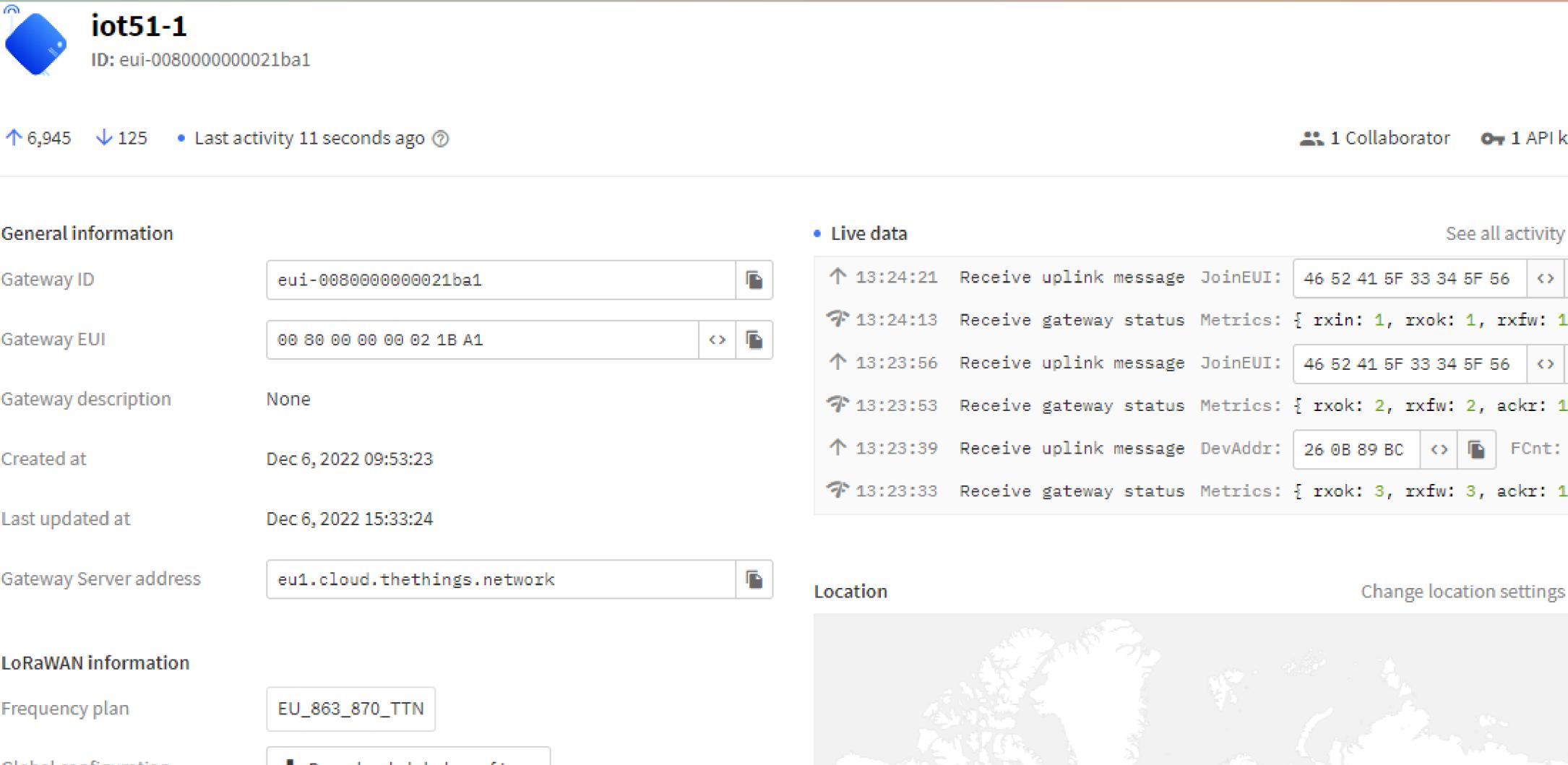
Live data

See all activity →

- ↑ 13:24:21 Receive uplink message JoinEUI: 46 52 41 5F 33 34 5F 56
- ↗ 13:24:13 Receive gateway status Metrics: { rxin: 1, rxok: 1, rxfw: 1,
- ↑ 13:23:56 Receive uplink message JoinEUI: 46 52 41 5F 33 34 5F 56
- ↗ 13:23:53 Receive gateway status Metrics: { rxok: 2, rxfw: 2, ackr: 100
- ↑ 13:23:39 Receive uplink message DevAddr: 26 0B 89 BC
- ↗ 13:23:33 Receive gateway status Metrics: { rxok: 3, rxfw: 3, ackr: 100

Location

Change location settings →



Notre solution pour répondre au cahier des charges

General information

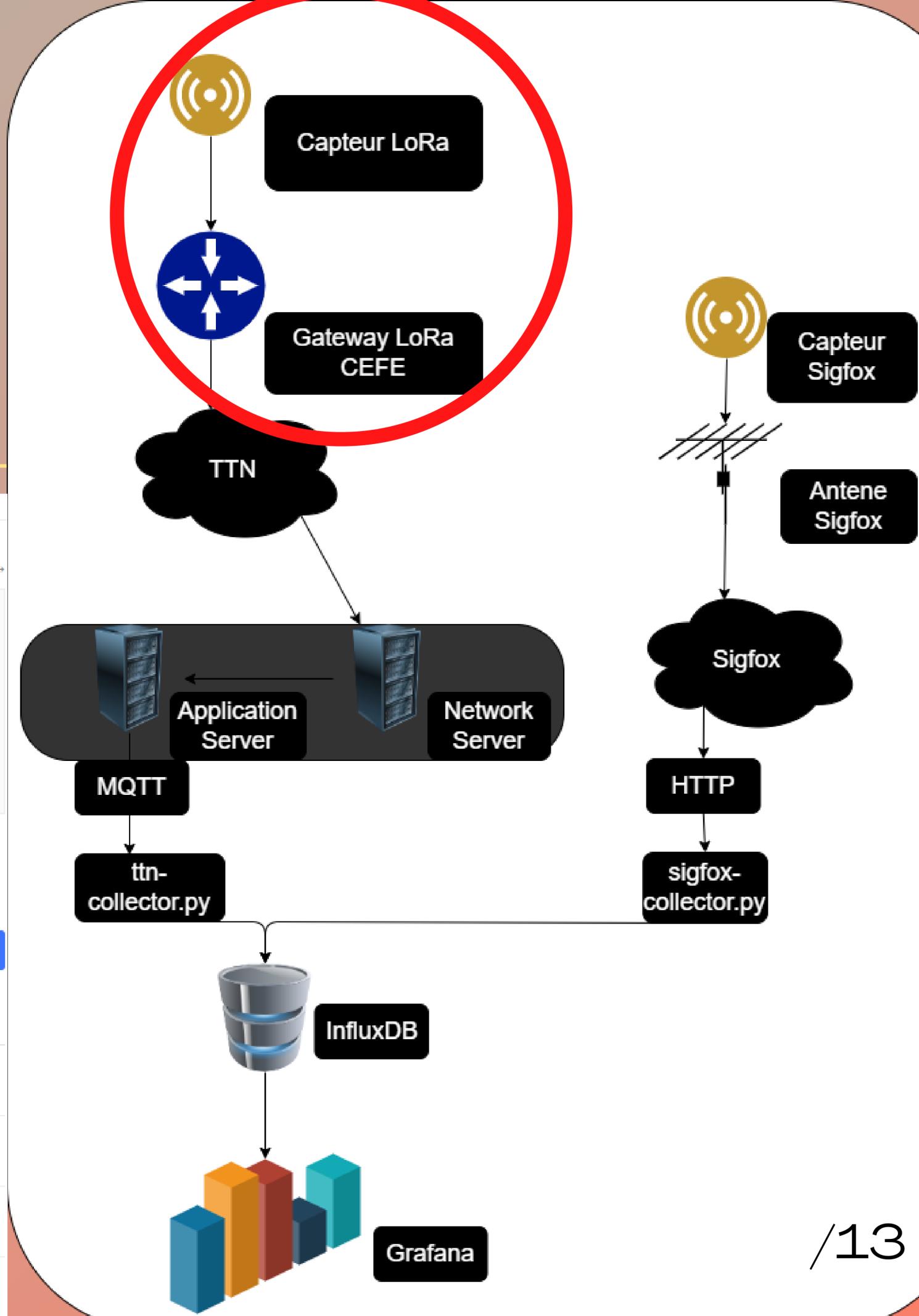
Application ID	iutbiot51
Created at	Dec 5, 2022 15:32:09
Last updated at	Dec 5, 2022 15:32:09

Live data

Time	Device	Action
↑ 13:20:14	eui-a81758...	Forward uplink data message
↑ 13:19:13	eui-a81758...	Forward uplink data message
↑ 13:18:12	eui-a81758...	Forward uplink data message
↑ 13:17:11	eui-a81758...	Forward uplink data message
↑ 13:16:10	eui-a81758...	Forward uplink data message
↑ 13:15:09	eui-a81758...	Forward uplink data message

End devices (4)

ID	Name	DevEUI	JoinEUI	Last activity
eui-a81758ffe086f8d	Detecteur Mouvement	A8 17 58 FF FE 08 6F 8D	00 00 00 00 00 00 00 00	just now •
eui-70b3d53260062062	Smilio	70 B3 D5 32 60 06 20 62	70 B3 D5 32 60 00 01 00	Never •
eui-0018b2100000a482	Interupteur LoRa	00 18 B2 10 00 00 A4 82	00 18 B2 44 52 49 43 31	21 min. ago •
eui-70b3d5e820002d69	RFID Reader	70 B3 D5 E8 20 00 2D 69	70 B3 D5 E8 20 00 00 18	3 hr. ago •



Notre solution pour répondre au cahier des charges

```
while True: # toute les 10mins

    for device in devices:

        response = requests.get(f"https://backend.sigfox.com/api/v2/devices/{device}/messages",
                               auth=authentication) # questionne l'API REST Sigfox

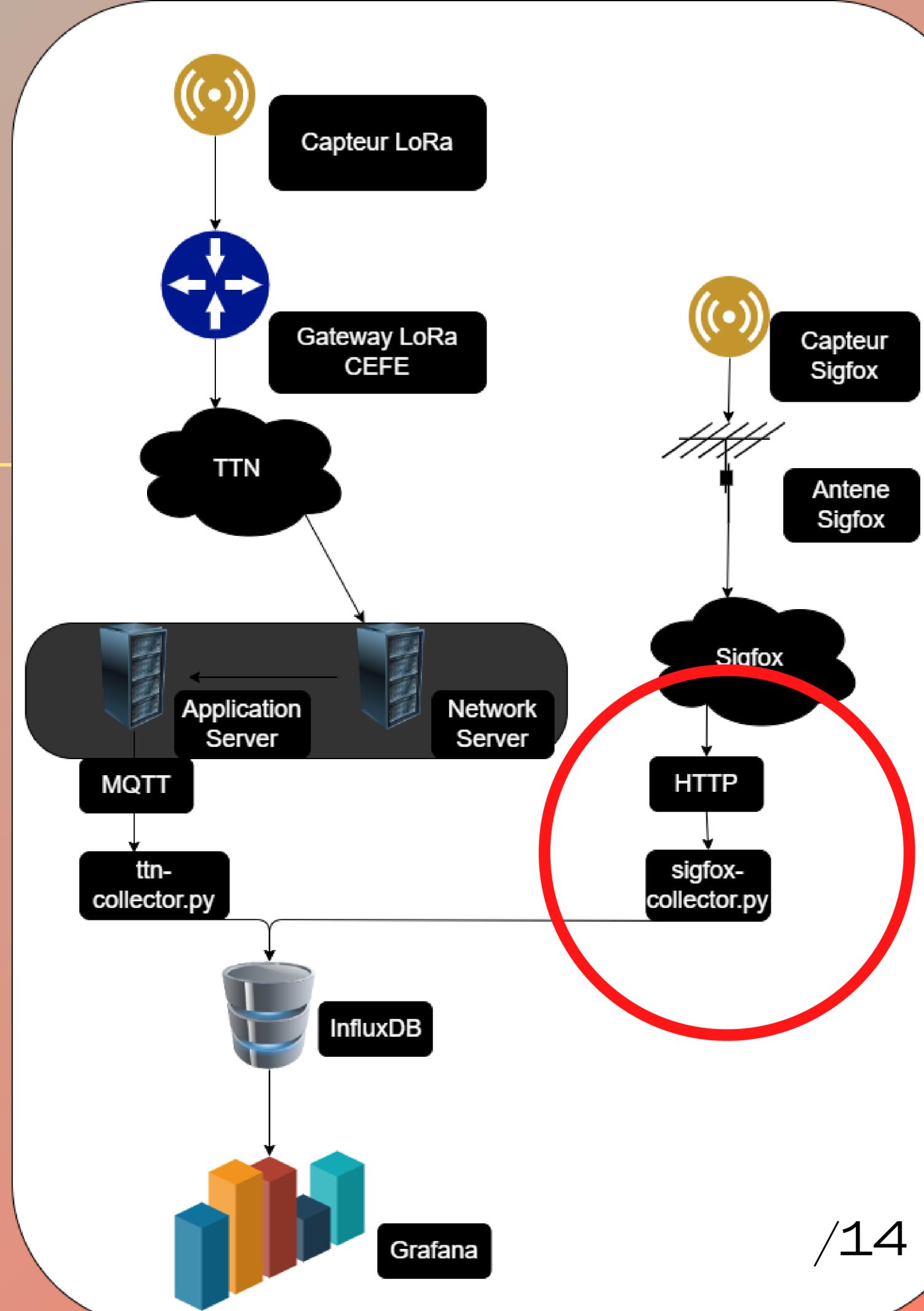
        jsonData = json.loads(response.text) # transforme le Str en JSON
        lastData = jsonData["data"][-1] # prend la dernière valeurs

        my_hexdata = lastData['data'] # extrait le payload ( Hexa )

        scale = 16 ## nb caractère

        num_of_bits = len(my_hexdata)*4

        binData = bin(int(my_hexdata, scale))[2:].zfill(num_of_bits) # hexa -> bin
```



Notre solution pour répondre au cahier des charges

```
""" Parcing payload avec information Sigfox """

batterie = (int(binData[:5],2)* 0.05) + 2.7

reserved = binData[5:8]
dataType = binData[8:13]

flag = binData[13:14]

dataTmp = (int(binData[14:24],2)-200)/8

dataHum = int(binData[24:32],2)/2

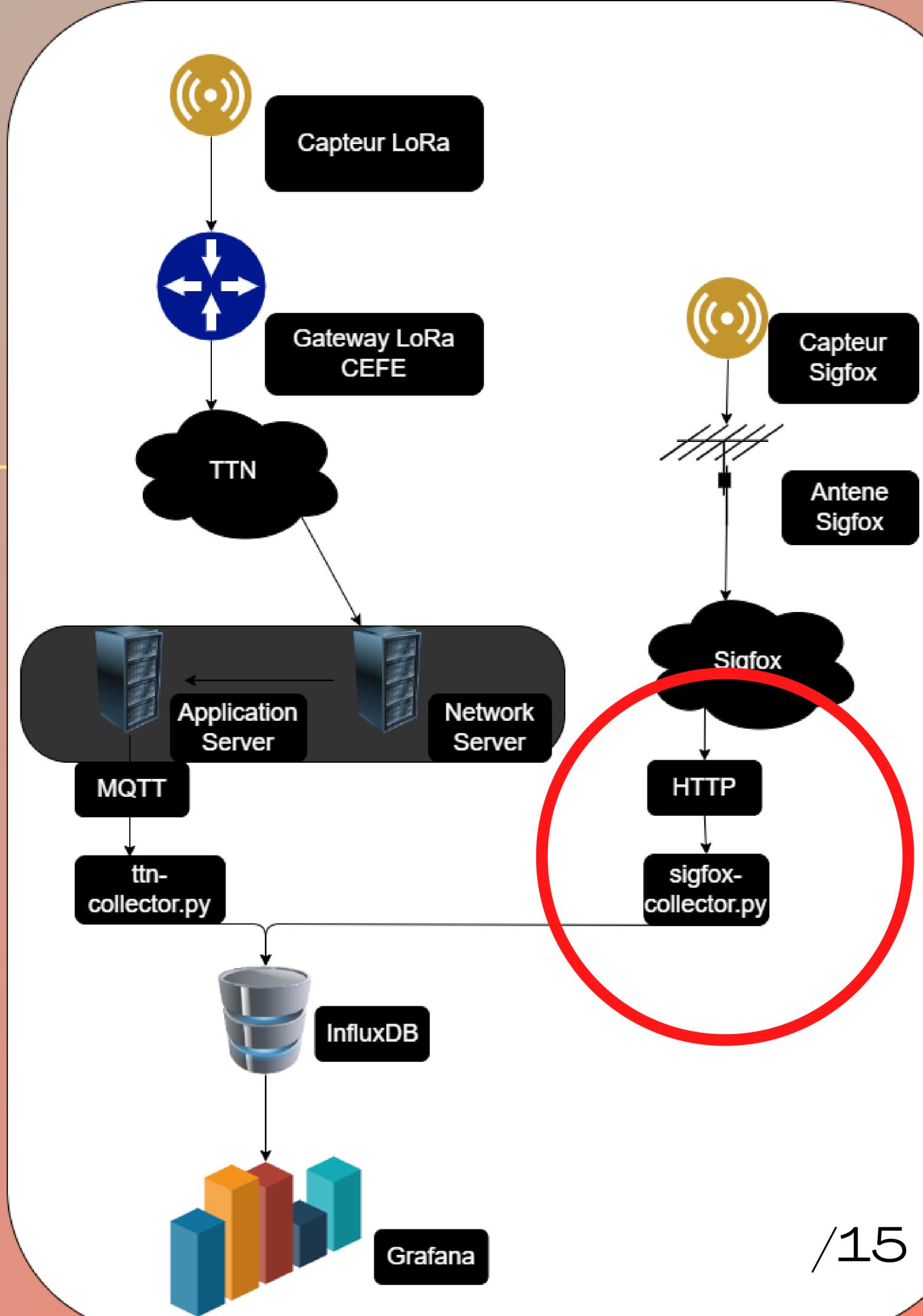
bucket = "IUT_Devices" # Bucket influxDB
token = "NZuKx9nuCgkreX6PJ7jd2IdEGNVAosqmxJUnJi944WTODCL-XJ1WXcAEHcMwmpEbML20G4P60QEr99YiU-xcg==" # Token Bucket

client = InfluxDBClient(url="http://51.158.110.29:8086", token=token, org="iutbeziers") # Login to Influx
write_api = client.write_api(write_options=SYNCHRONOUS) # Write mode + synchronise

t = Point("sensors").tag("device",device).field("temp",dataTmp) # create the influx point
h = Point("sensors").tag("device",device).field("hum",dataHum) # create the influx point
v = Point("sensors").tag("device",device).field("vdd",batterie)

write_api.write(bucket=bucket, record=t) # write
write_api.write(bucket=bucket, record=h) # write
write_api.write(bucket=bucket, record=v) # write

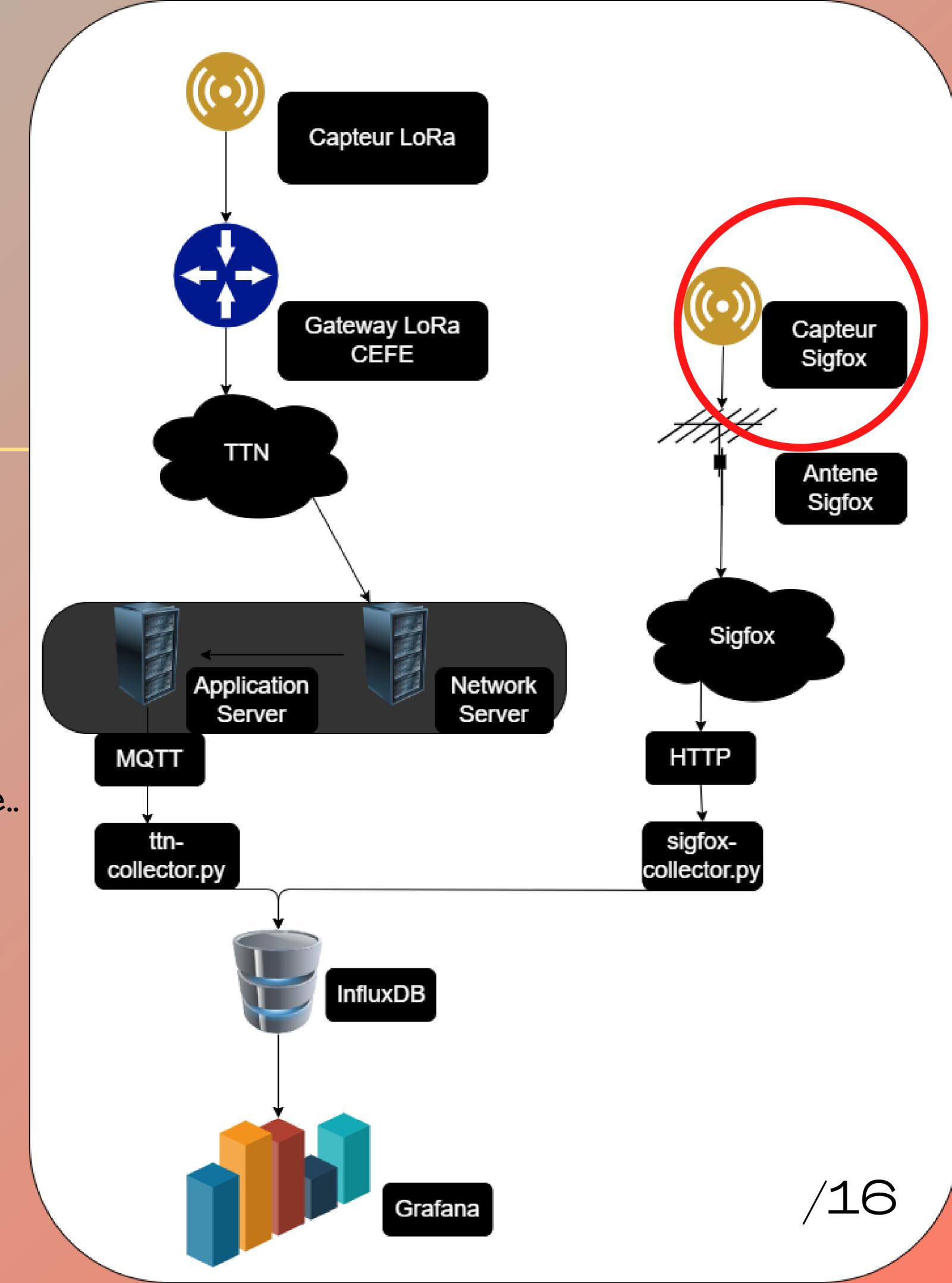
print(f"Data push : tmp = {dataTmp} | hum = {dataHum}")
client.close() # ferme la connexion client
```



Notre solution pour répondre au cahier des charges



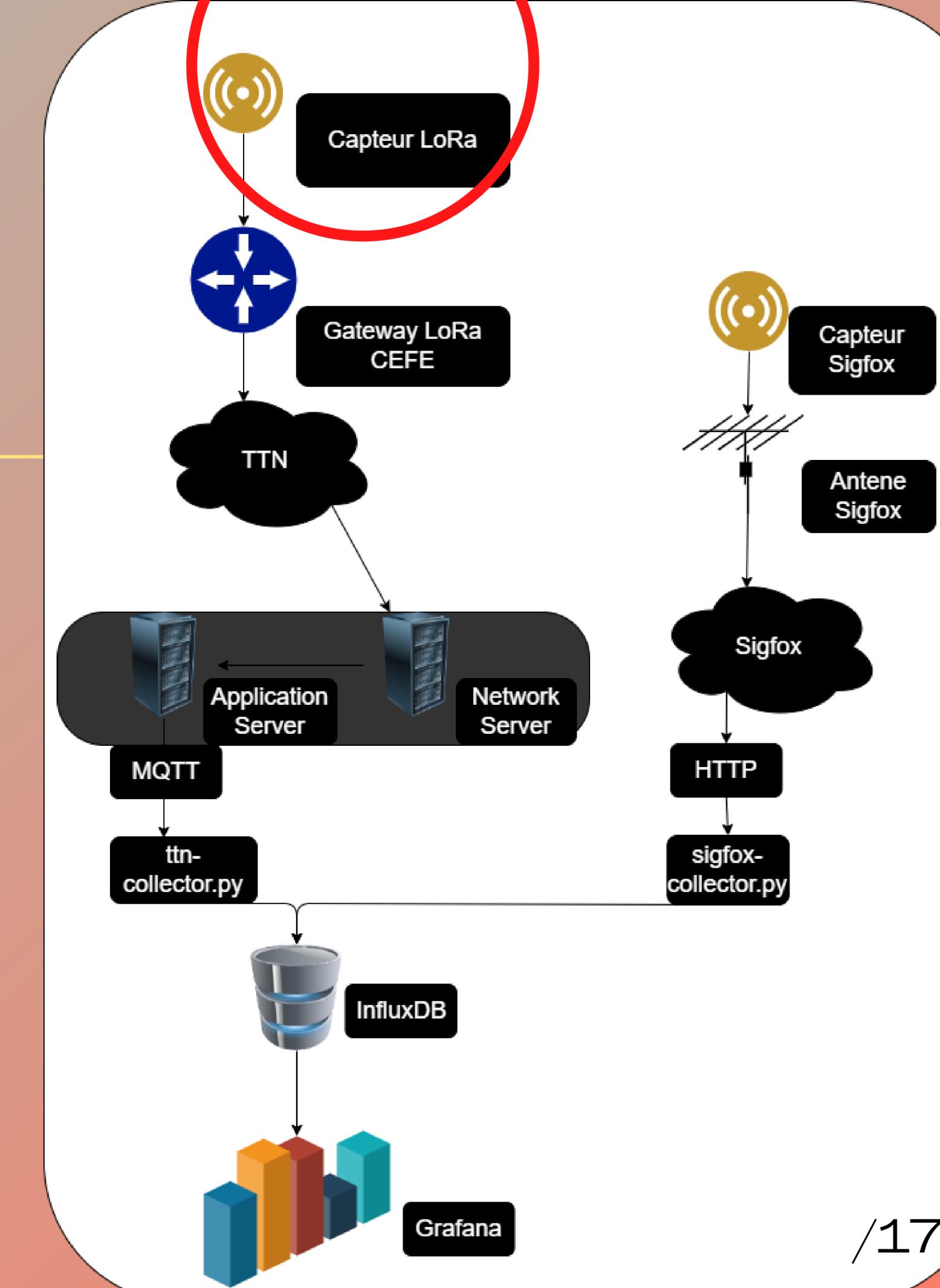
- Température & Humidité.
- Luxmètre.
- détection ouverture de porte..
- Vibration.
- magnétomètre.



Notre solution pour répondre au cahier des charges



- Lecteur de tag RFID.
- Décodage manuel.

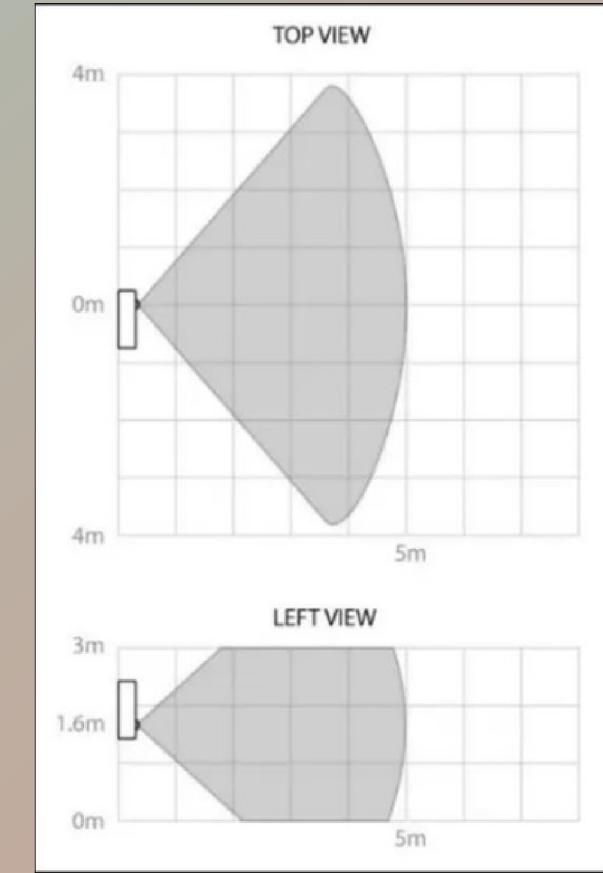


- Présentation du capteur
- Mise en place et configuration
- Récupération des données



Plusieurs utilité :

- Mouvement
- Température
- Humidité
- Luxmètre



Spécificité

- Capteur infrarouge
- Distance de 5m
- Angle de détection de 90°

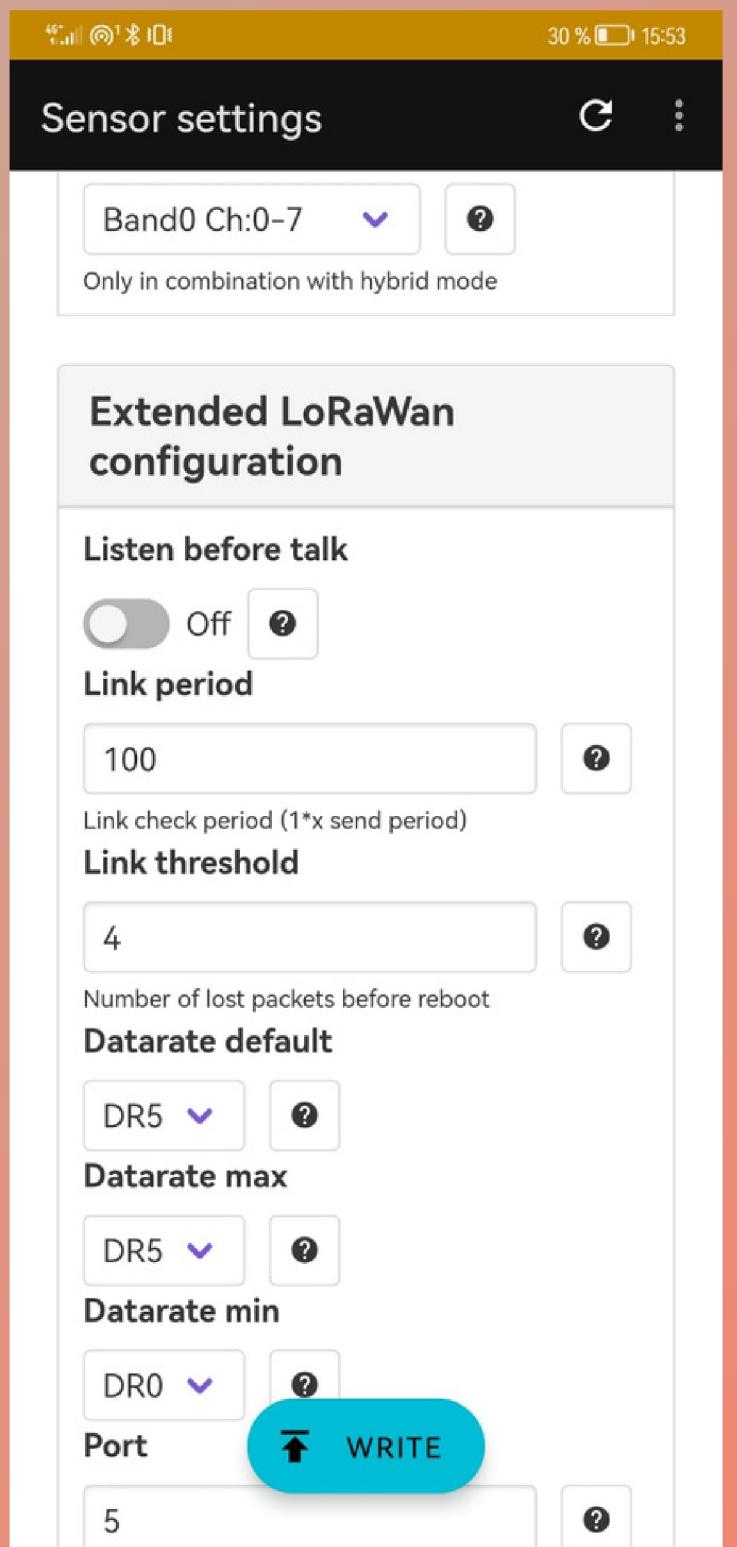
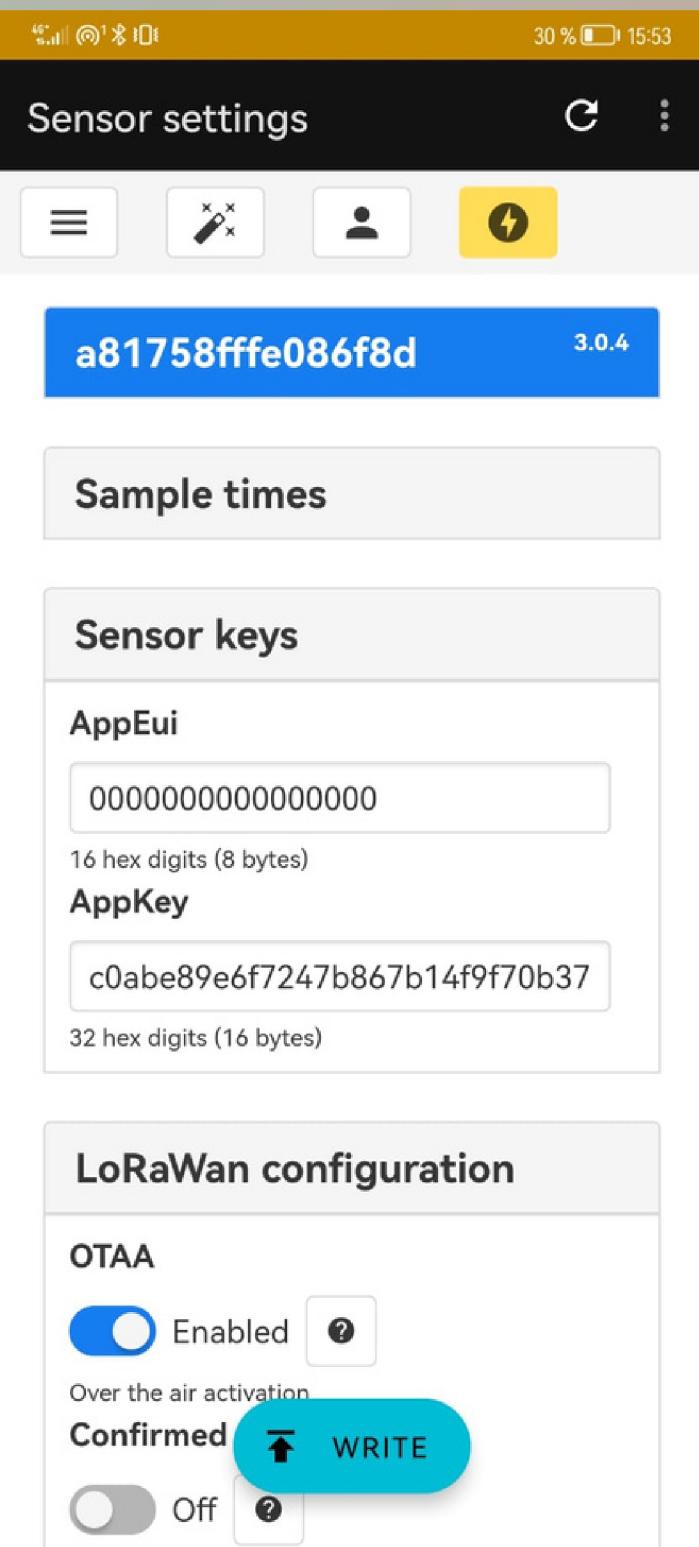
Mise en place du capteur

Necessiter :

- Deux piles
- L'application Sensor Settings



Configuration



Récupération des données

- Récupération des payloads
- Décodage des payloads
- Envoie des infos vers la base de données

Configuration de la gateway

Enregistrer sur la gateway

Register gateway

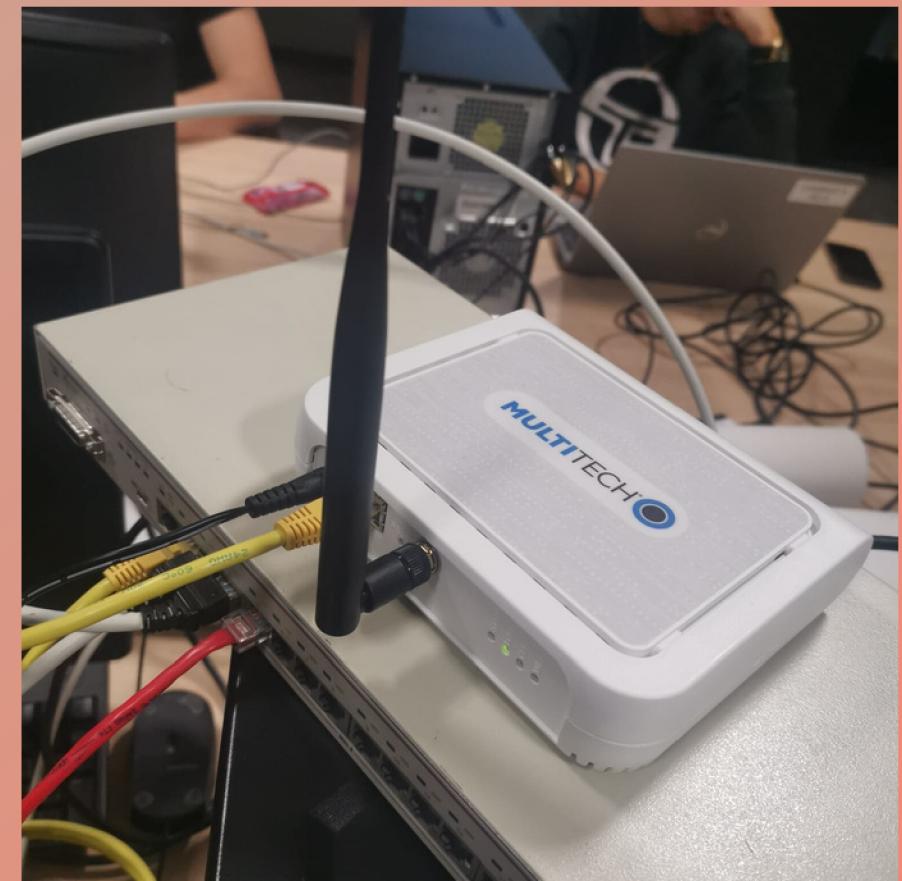
Register your gateway to enable data traffic between nearby end devices and the network.

Learn more in our guide on [!\[\]\(26debdd08de6dadb3b1430770cea622d_img.jpg\) Adding Gateways !\[\]\(4ae72a306054dbd6e24ae8eb29da4b3f_img.jpg\)](#).

Gateway EUI 

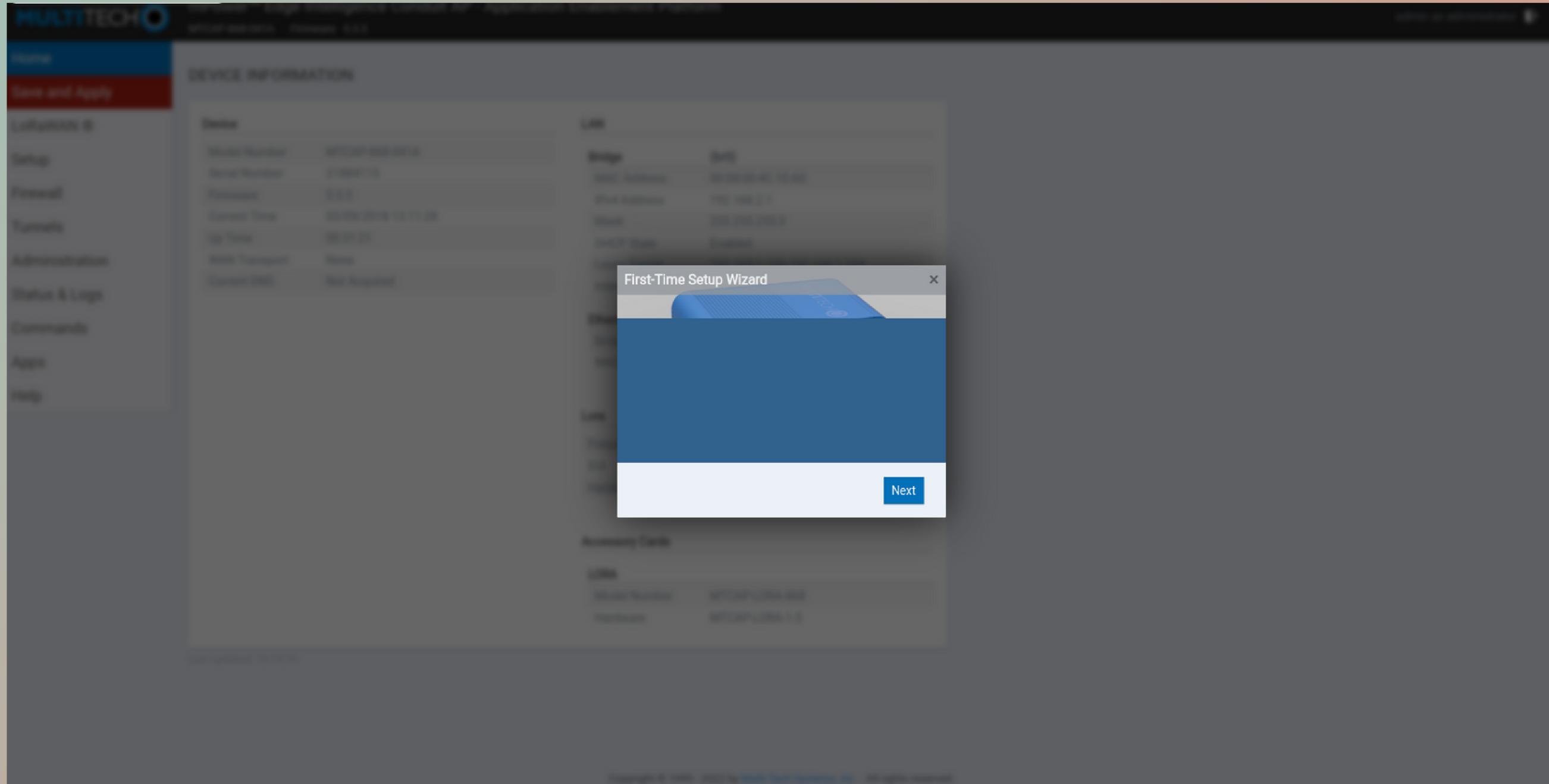
[Gateway EUI](#) [Continue without EUI](#)

To continue, please confirm the Gateway EUI so we can determine onboarding options



Configuration de la gateway

Connexion à la gateway sur le site avec l'ip 192.168.2.1



Configuration des options de base (date,heure ...)

Configuration de la gateway

Passage de la gateway en adressage dynamique

mPower™ Edge Intelligence Conduit AP - Application Enablement Platform
MTCAP-868-041A Firmware 5.3.3

Home
Save and Apply
LoRaWAN ®
Setup
Network Interfaces
Global DNS
DDNS Configuration
DHCP Configuration
SMTP Configuration
SNMP Configuration
Time Configuration

NETWORK INTERFACE CONFIGURATION - ETH0 ⓘ

Direction
WAN

IPv4 Settings

Mode
DHCP Client

IP Address

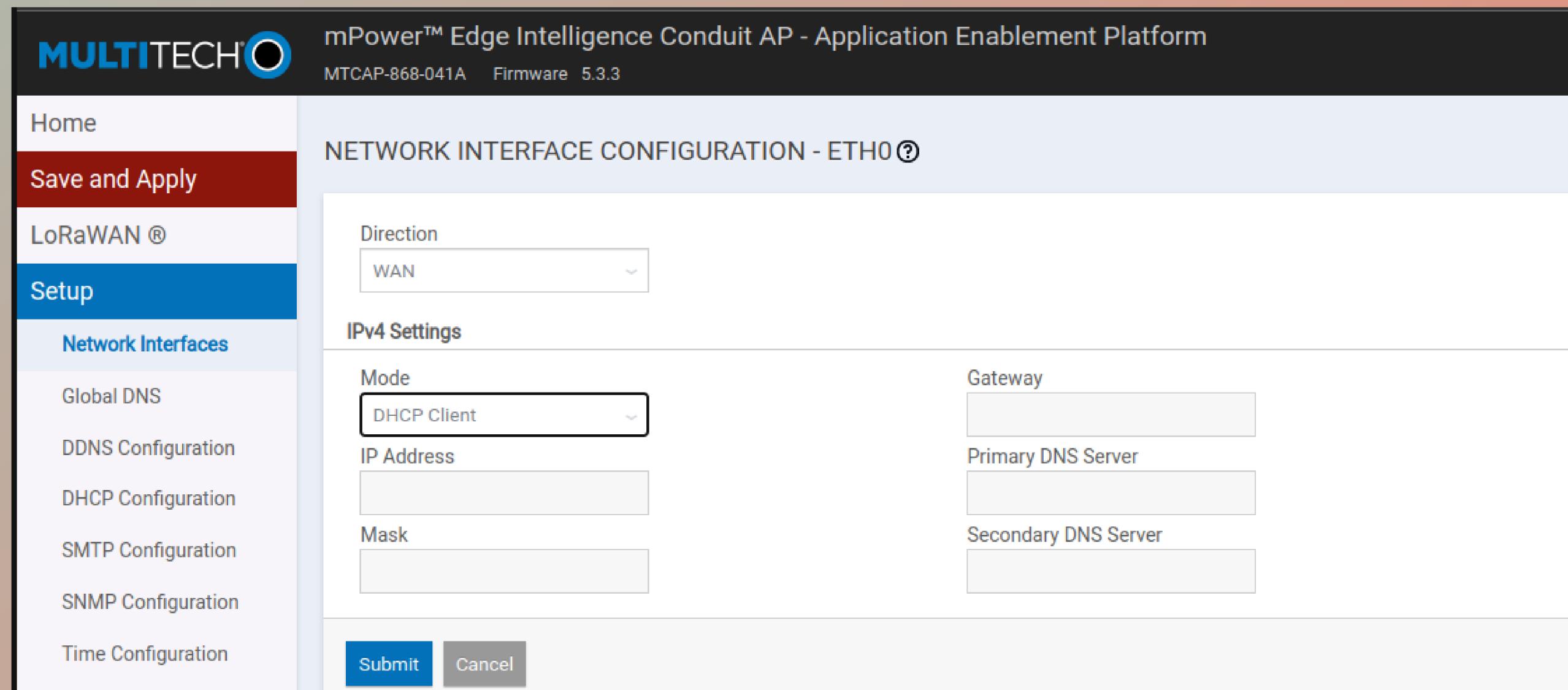
Mask

Gateway

Primary DNS Server

Secondary DNS Server

Submit Cancel



Configuration de la gateway

The screenshot shows the configuration interface for a gateway in LoRa Mode. It includes sections for LoRa Card Information, Server settings, and Forward CRC options.

Component	Version	Status
Packet Forwarder	4.0.4-r1.0	STOPPED
Network Server	2.5.1	DISABLED
Lens Server	2.5.1	DISABLED
Basic Station	2.0.5-1-r3.0	DISABLED

LoRa Card Information

Server	Forward CRC
Network: The Things Network	<input type="checkbox"/> Forward CRC Disabled
Server Address: eu1.cloud.thethings.network	<input checked="" type="checkbox"/> Forward CRC Error
Upstream Port: 1700	<input checked="" type="checkbox"/> Forward CRC Valid
Downstream Port: 1700	

Configuration de la partie LoraWan

Système d'alerte

- Alerte en fonction de la température
- Donne la Température
- Donne l'Humidité

```
def sendMailAlert(temp,hum,presence):  
    email_sender = 'alerte.poulailler51@gmail.com'  
    email_password = 'ptaoelxvsbrniany'  
    email_receiver = 'aksel.paulet@gmail.com'  
  
    subject = 'ALERTE Poulailler CNRS'  
    body = f"""Bonjour,  
  
Ce message survient à une alerte dans votre poulailler.  
  
Ci-dessous les relevés pouvant être à l'origine :  
  
Température : {temp} °c  
Humidité : {hum} %  
{"PRESENCE DANS L'ENCLO" if presence else "aucune présence détectée"}  
....  
  
em = EmailMessage()  
em['From'] = email_sender  
em['To'] = email_receiver  
em['Subject'] = subject  
em.set_content(body)  
# Add SSL (layer of security)  
print("creating ssl context")  
context = ssl.create_default_context()  
# Log in and send the email  
print("try login send")  
with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:  
    print("try")  
    smtp.login(email_sender, email_password)  
    print("login")  
    smtp.sendmail(email_sender, email_receiver, em.as_string())  
    print("send")
```

Système d'alerte

- Comment ça marche ?

```
tempOk = True

while True:
    hum ,temp = get_humTemp("B448E4")

    print(f"Hum = {hum} | temp = {temp}")

    if tempOk == True :
        print("tempOK True")
        if temp <= 14.5 or temp >= 21.5:
            print("TempOK going to False")
            tempOk = False
            print("mailAlertCall")
            sendMailAlert(temp,hum,False)
            print("mailAlertSend")

    else: # tempok = False

        if temp >= 15.5 and temp <= 20.5:
            tempok = True

            sendMailok(temp,hum,False)
            sleep(600)
```

```
def get_humTemp(devEUI):

    client = InfluxDBClient(
        url="http://51.158.110.29:8086",
        token="VXDczS854yyBIi11-_bQKsuYSi4Pik6W-FR3Cnf3rzNRAr",
        org="iutbeziers"
    )

    query_api = client.query_api()

    query = f'from(bucket: "IUT_Devices")\
    |> range(start: -1d)\
    |> last()\\
    |> filter(fn: (r) => r["device"] == "{devEUI}")'

    result = query_api.query(org="iutbeziers", query=query)
    client.close()
    value:list[int] = []

    for table in result:
        for element in table :
            value.append(round(element.get_value(),2))

    return value
```