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.

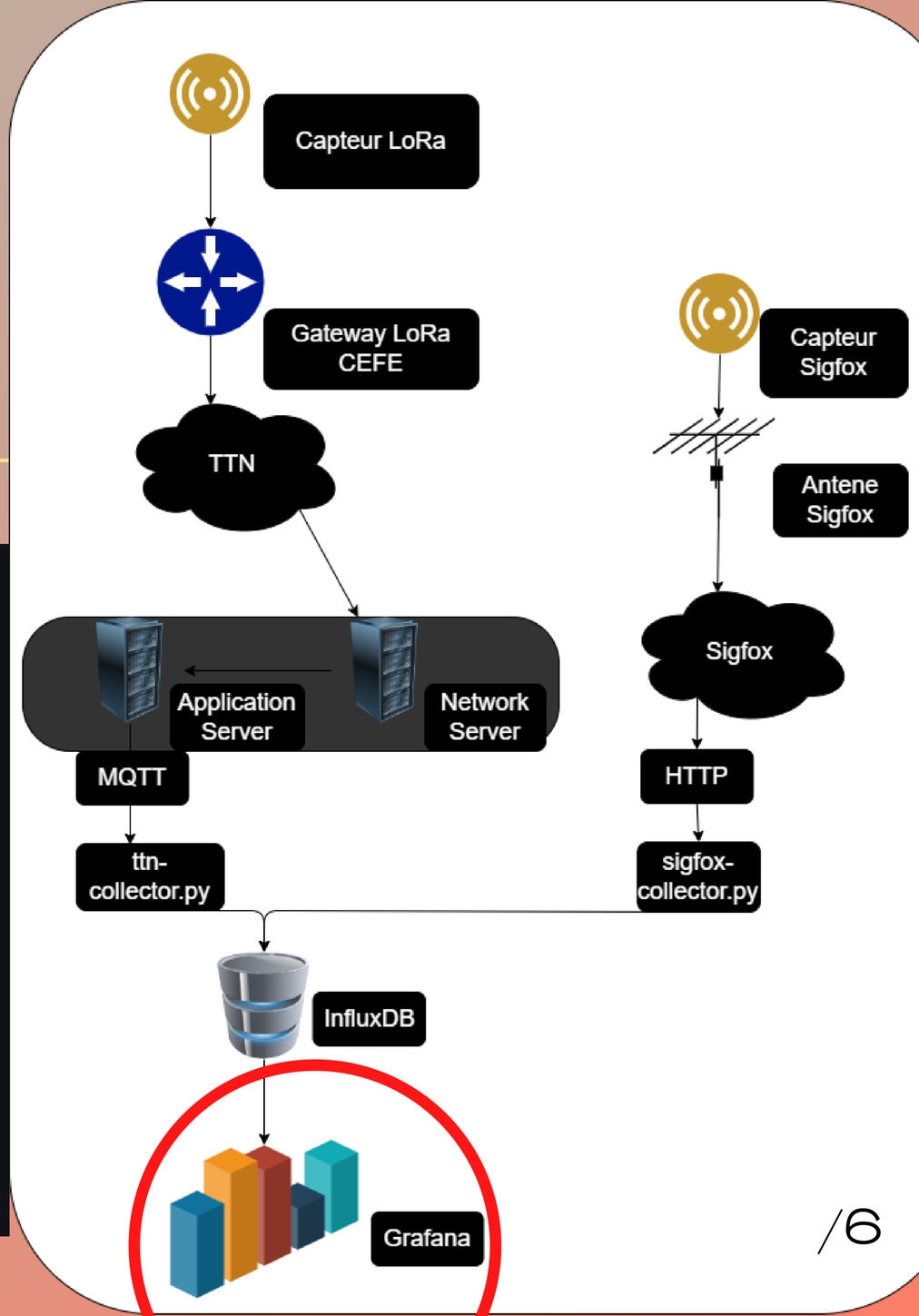
Contraintes

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

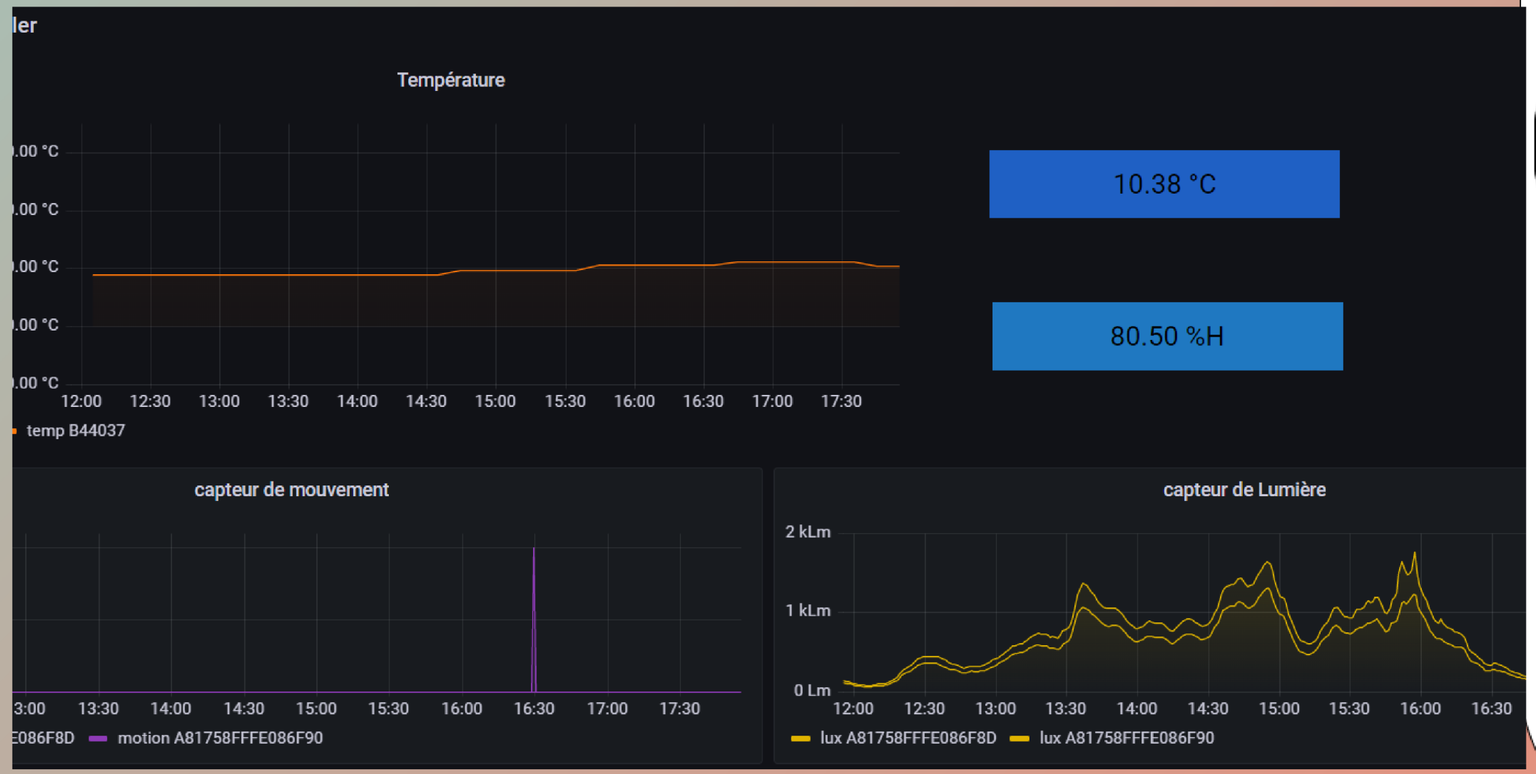
Poulailler connecté

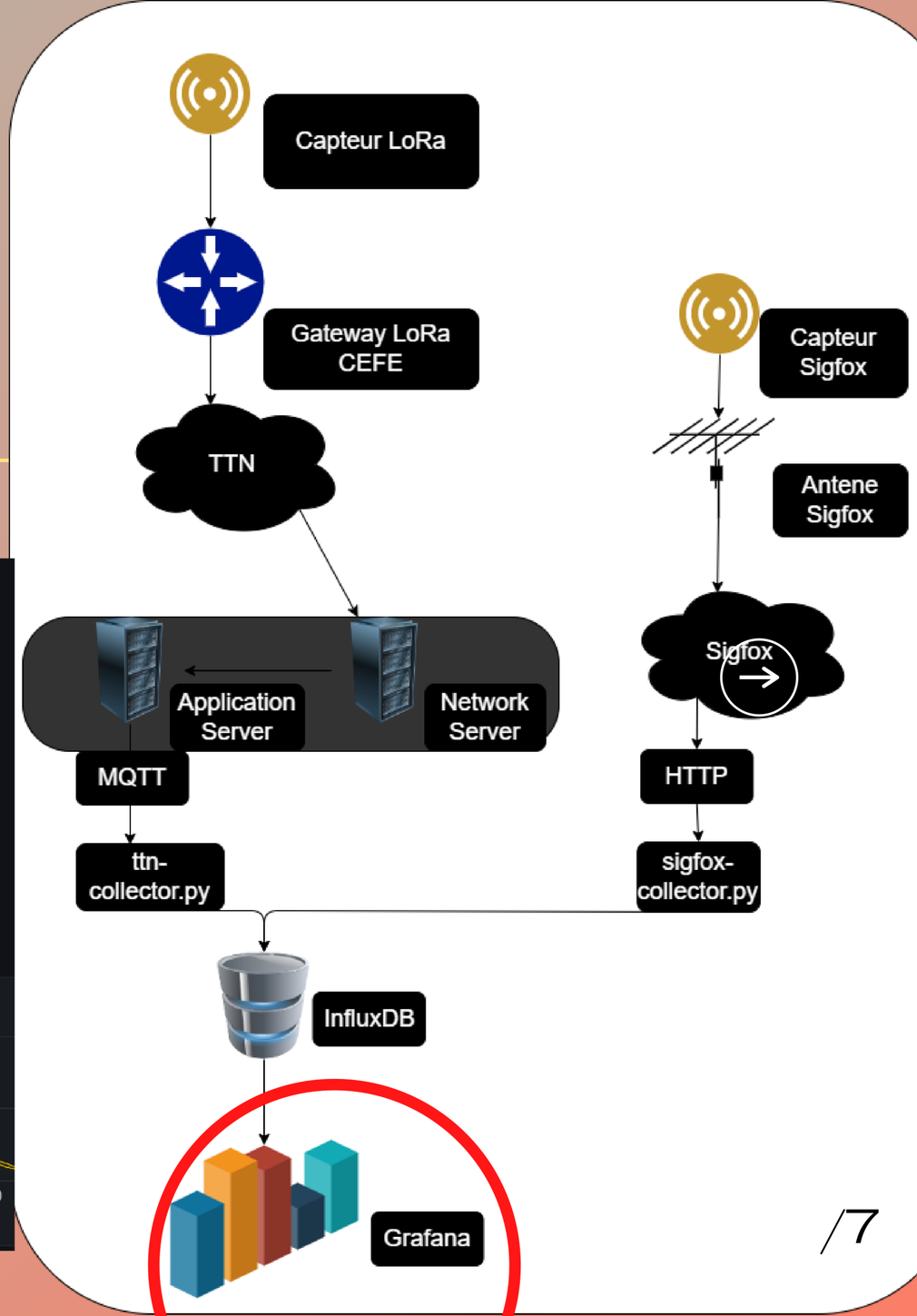
Le poulailler sur le site du CNRS, il est équipé de capteur de mouvement, de température, d'humidité, de luminosité et d'un lecteur de badge.

Notre solution pour répondre au cahier des charges

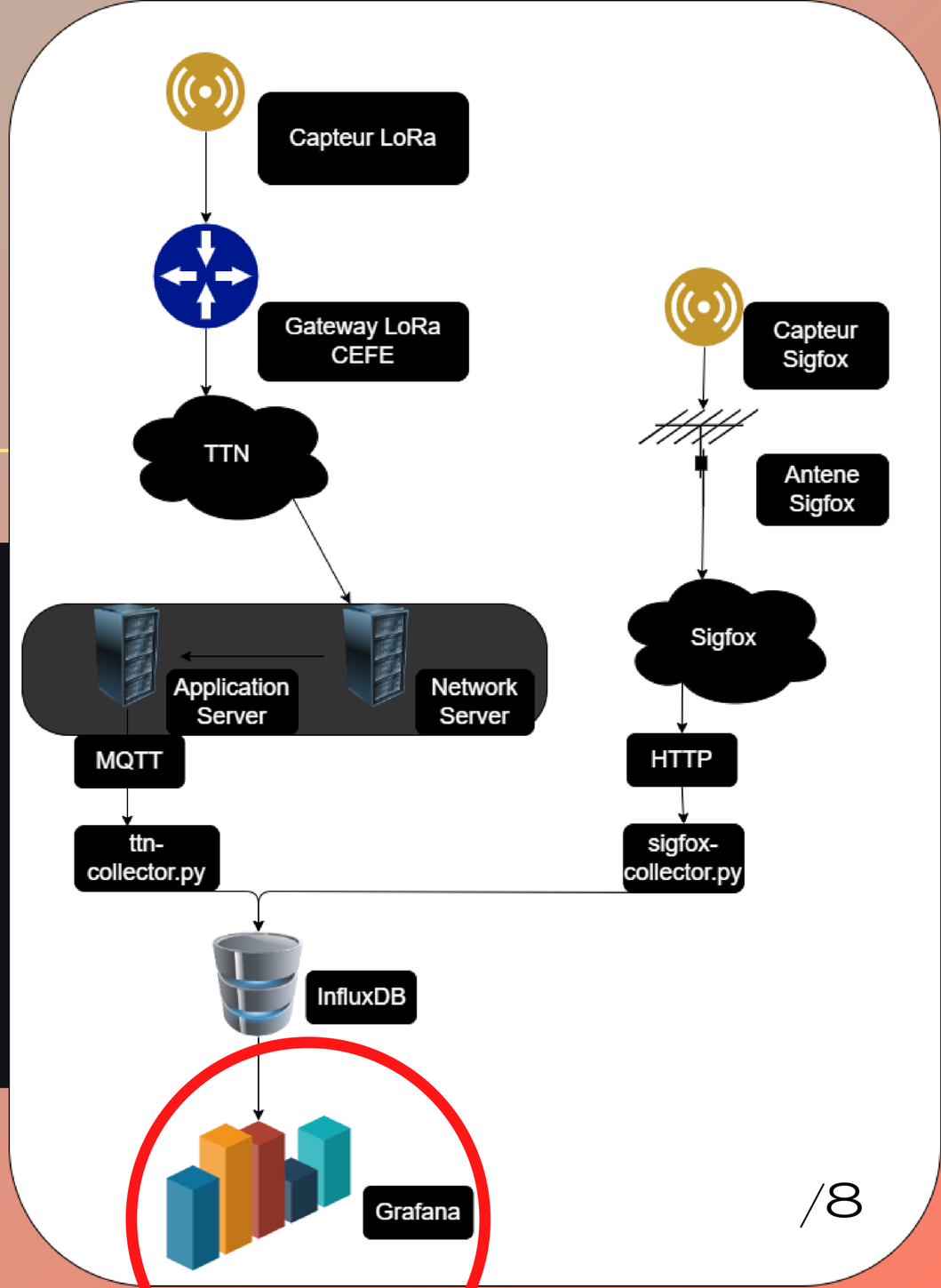


Notre solution pour répondre au cahier des charges

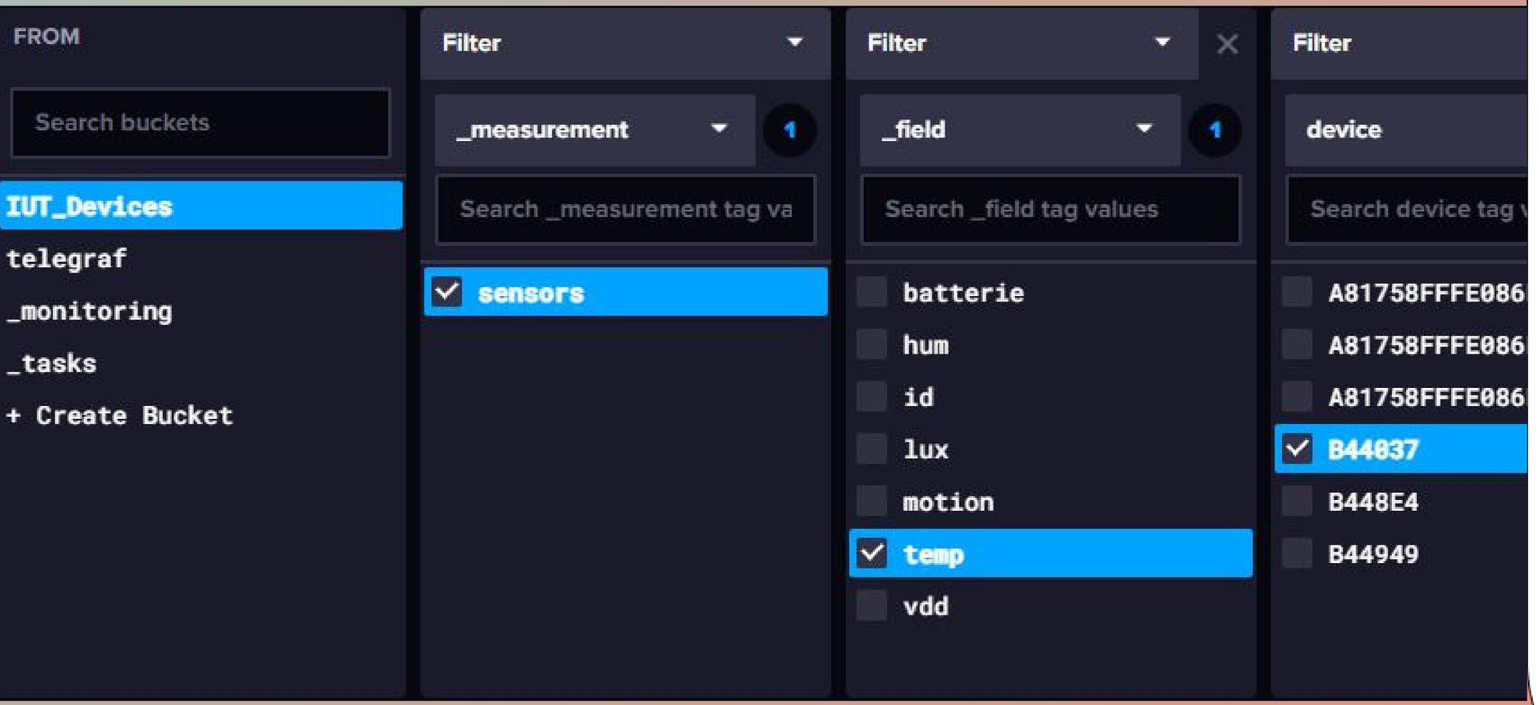


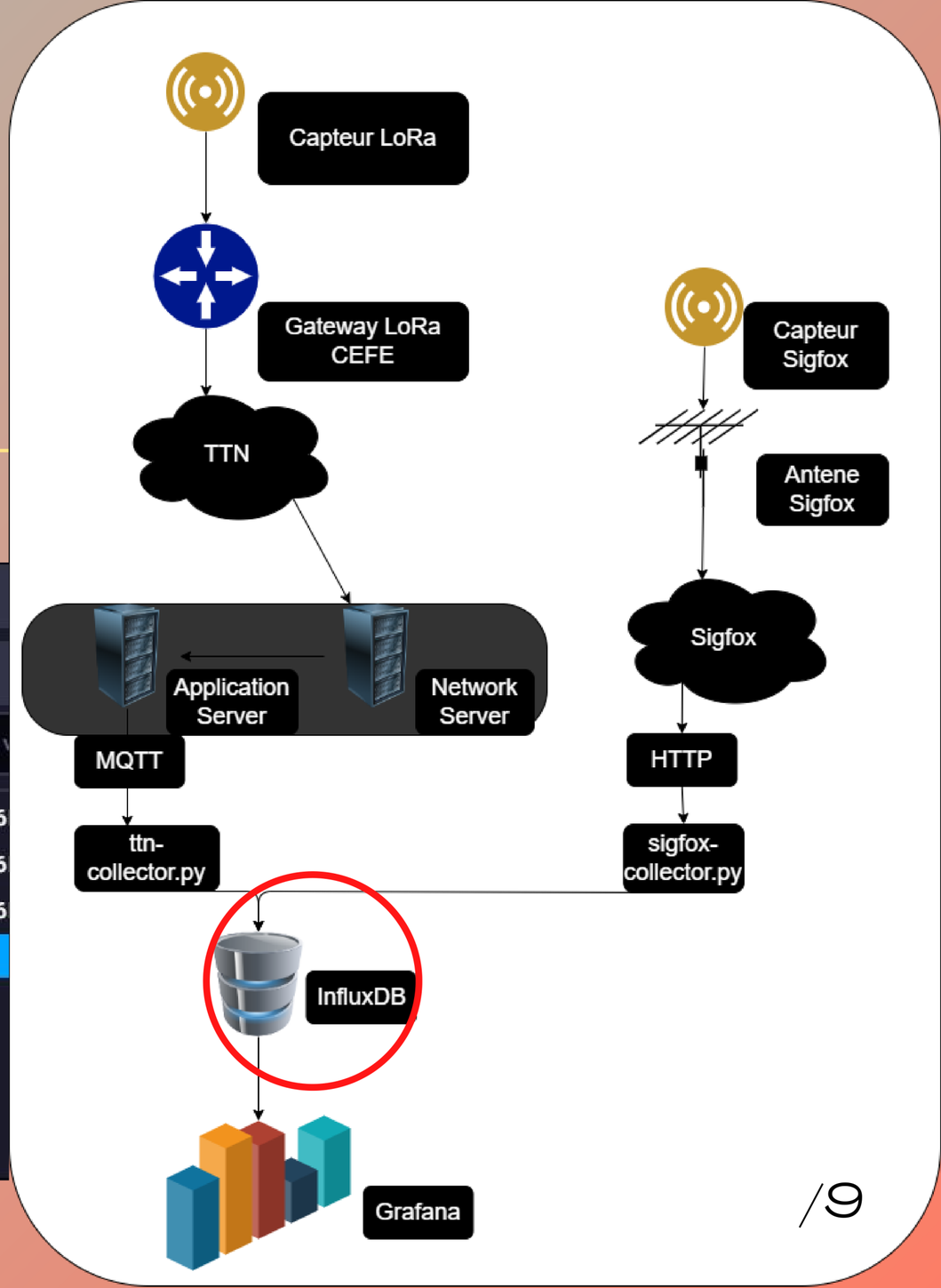


Notre solution pour répondre au cahier des charges

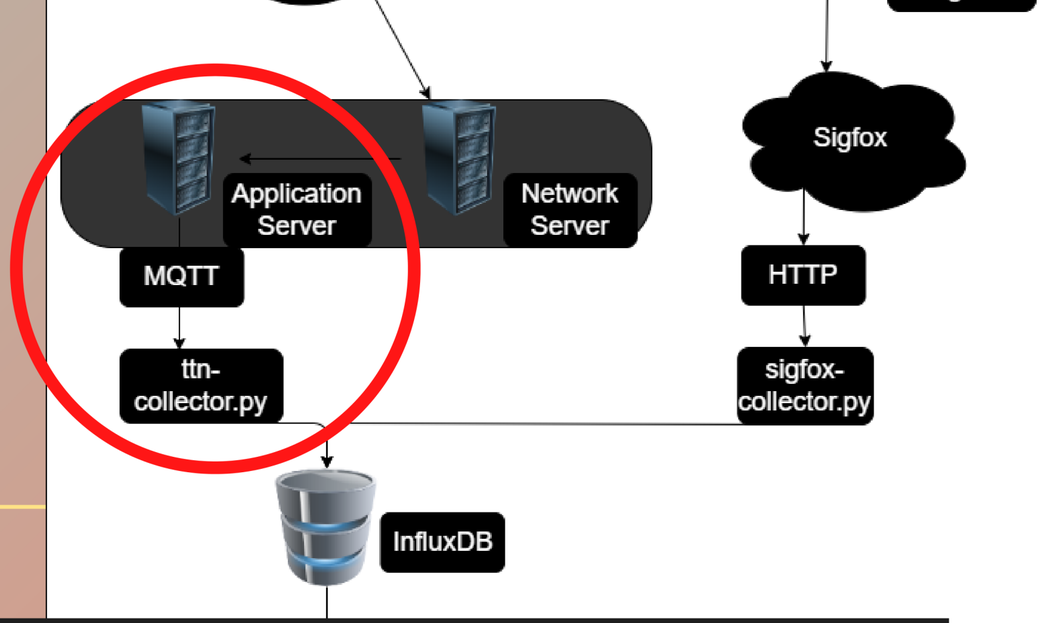


Notre solution pour répondre au cahier des charges



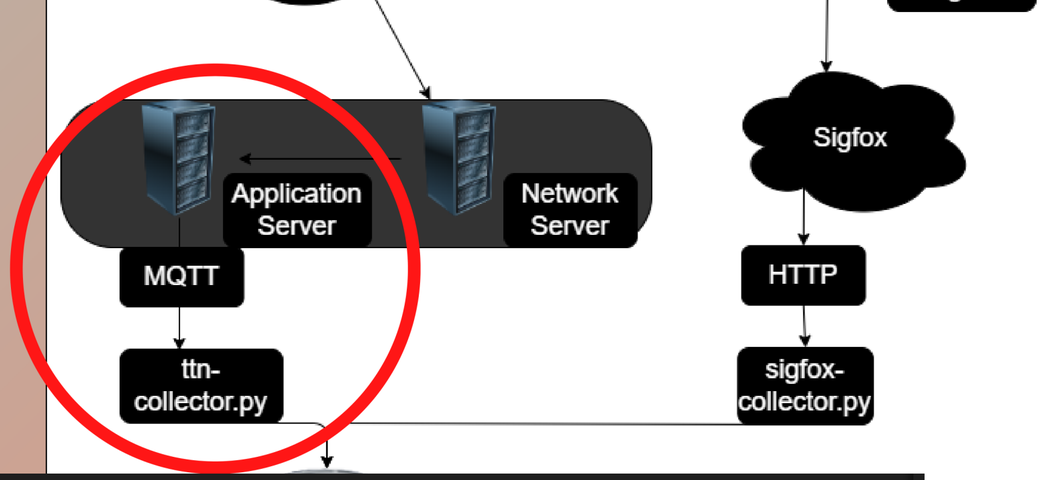


Notre solution pour répondre au cahier des charges



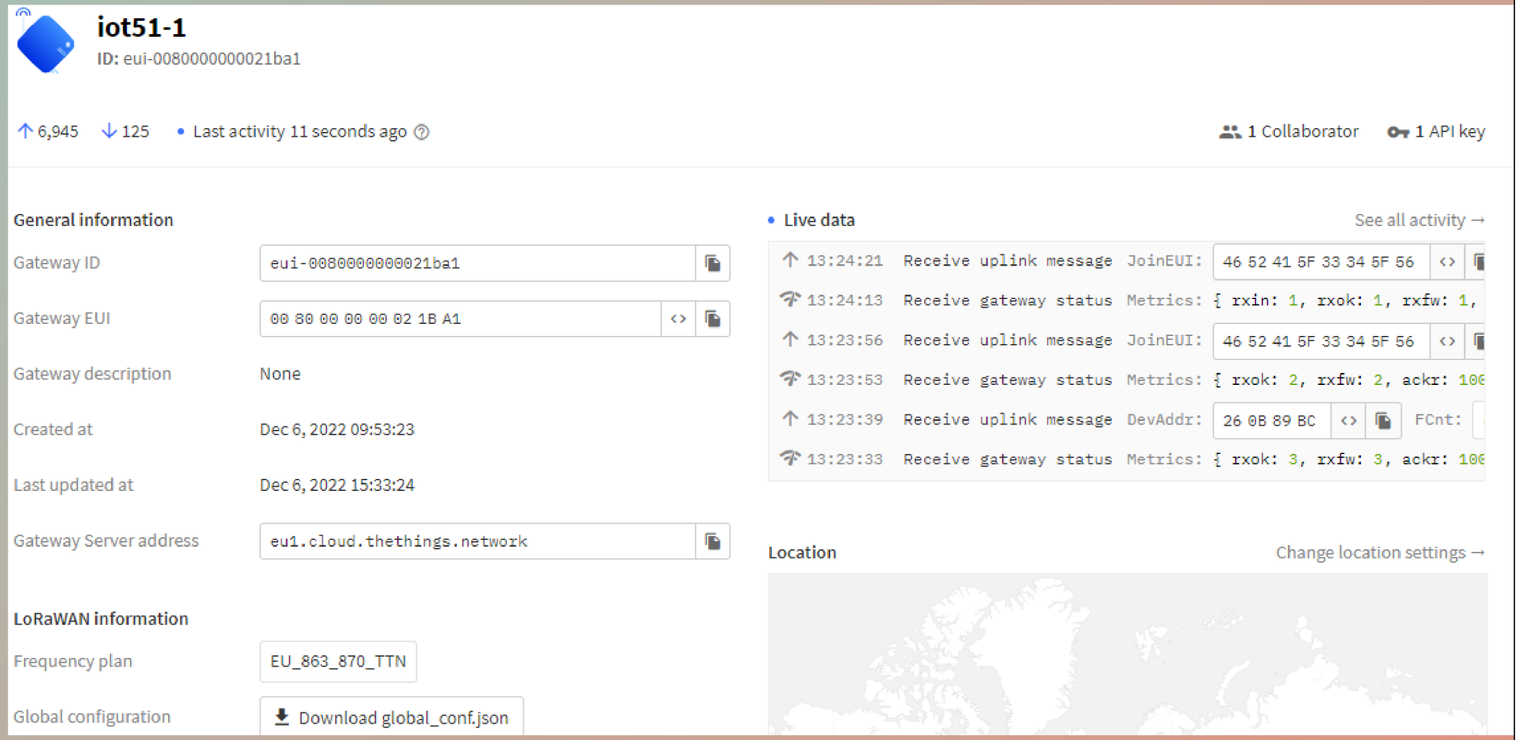

```python
for userId, ttnToken in ttnLogin:
    topic = f"v3/{userId}/devices/+/up"


    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
```

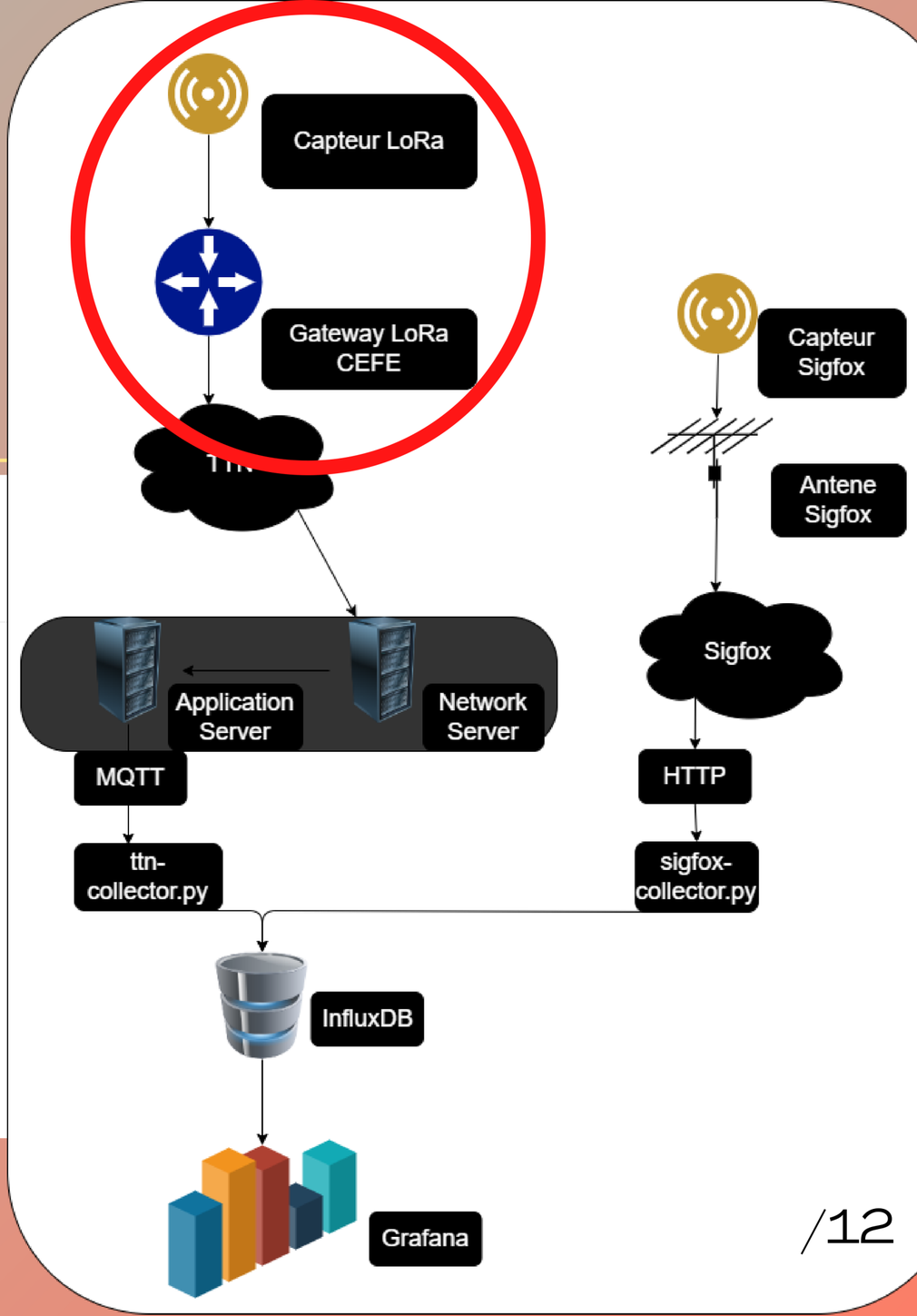
Notre solution pour répondre au cahier des charges



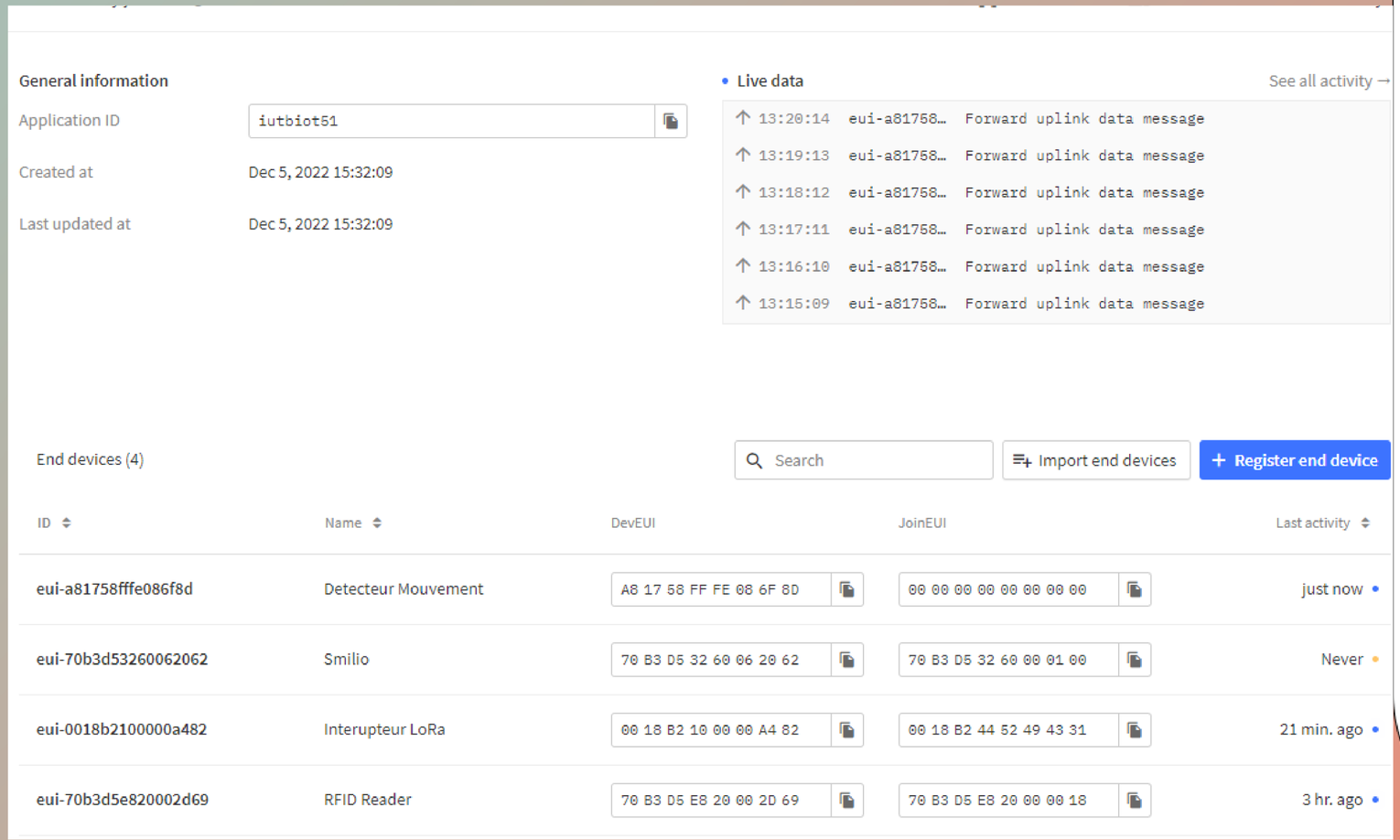

```python
def on_message(client, userdata, message):
    """Fonction appelé lors de l'arriver """

    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)
```

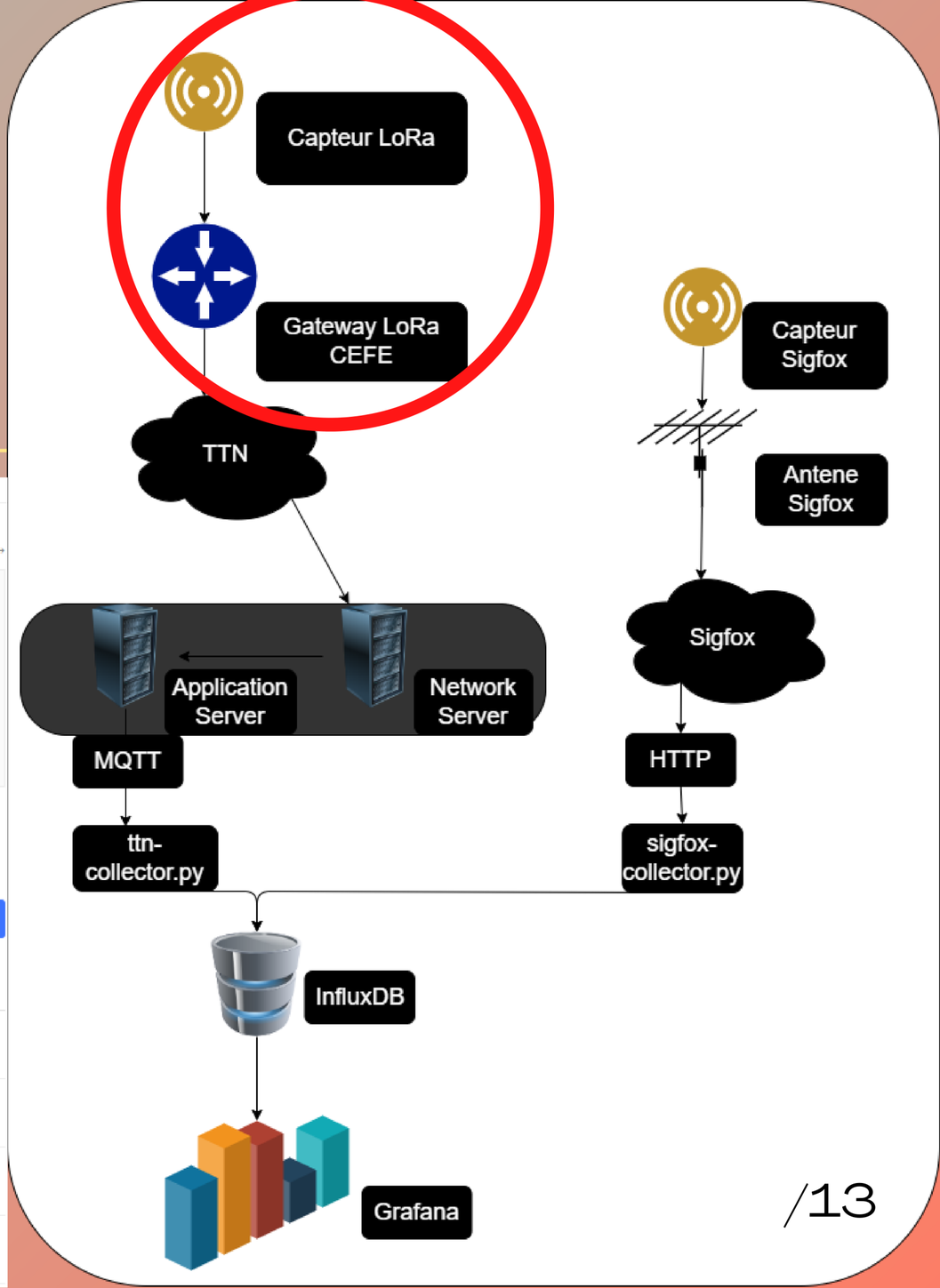
Notre solution pour répondre au cahier des charges





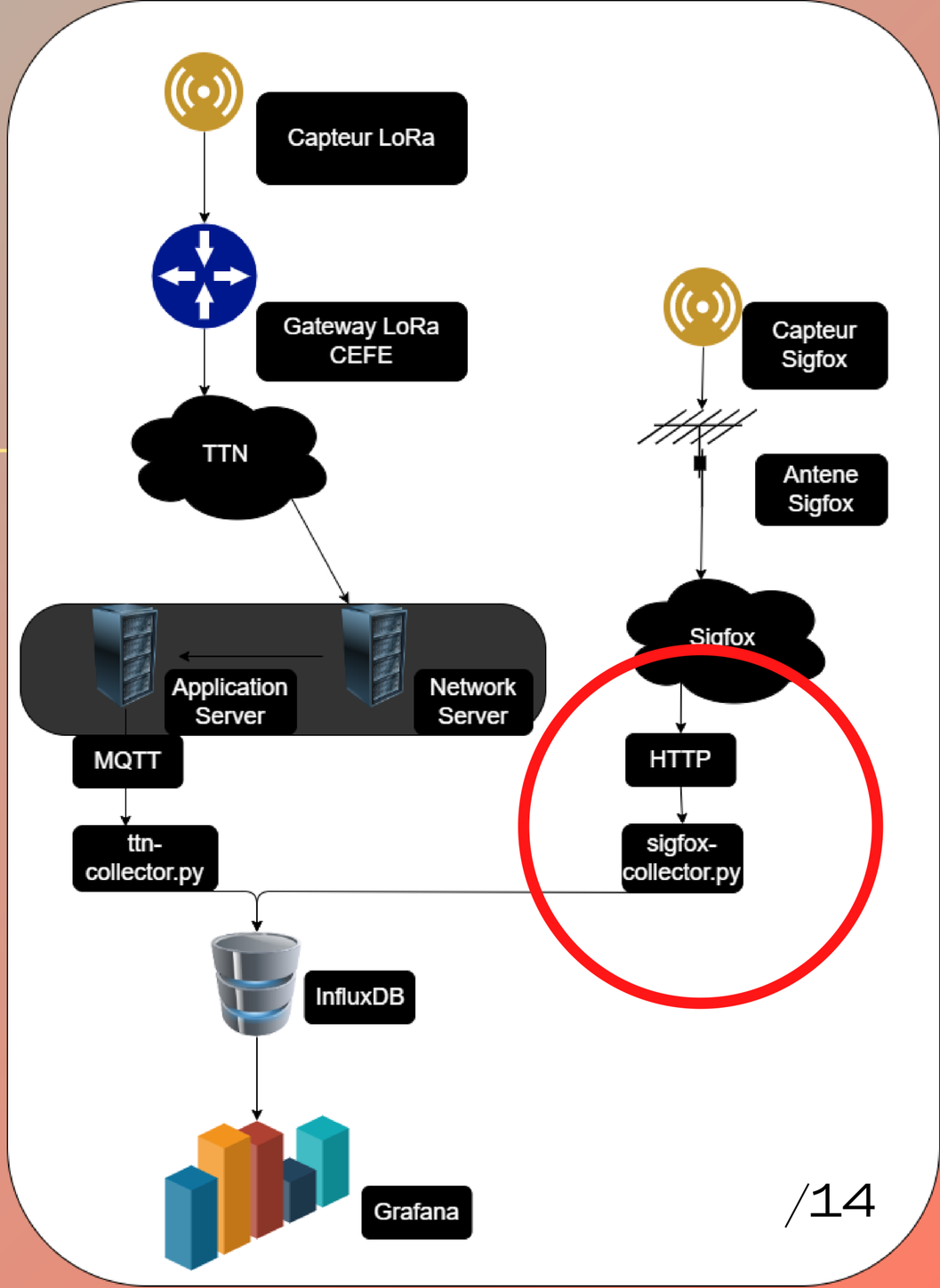
Notre solution pour répondre au cahier des charges





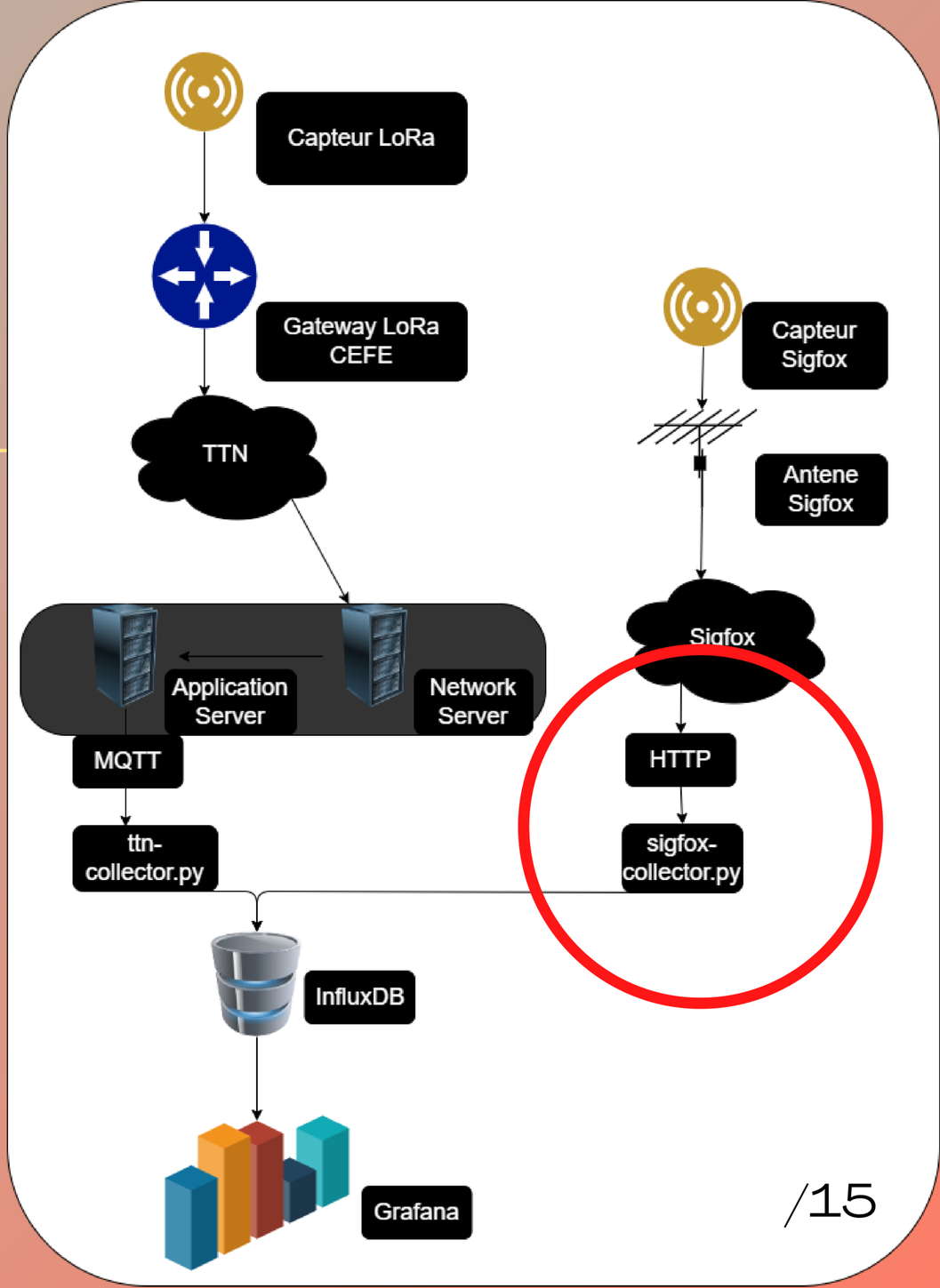
Notre solution pour répondre au cahier des charges

```python
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"][0] #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
```



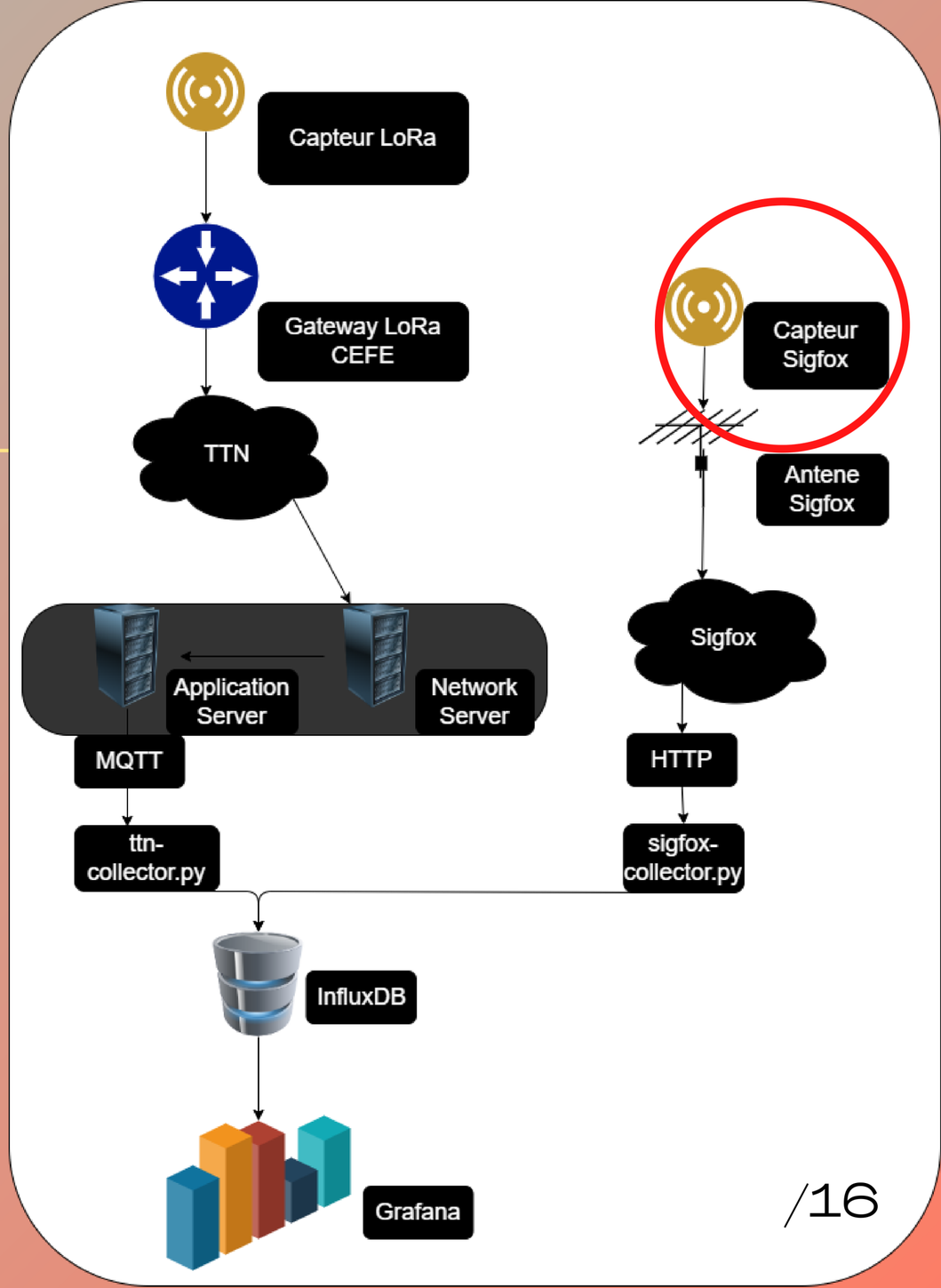
Notre solution pour répondre au cahier des charges

```python
""" 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 = "NZuKx9nuCgkreX6PJ7jd2IdEGNVAosqmxcJUnJi944WTODCL-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
```



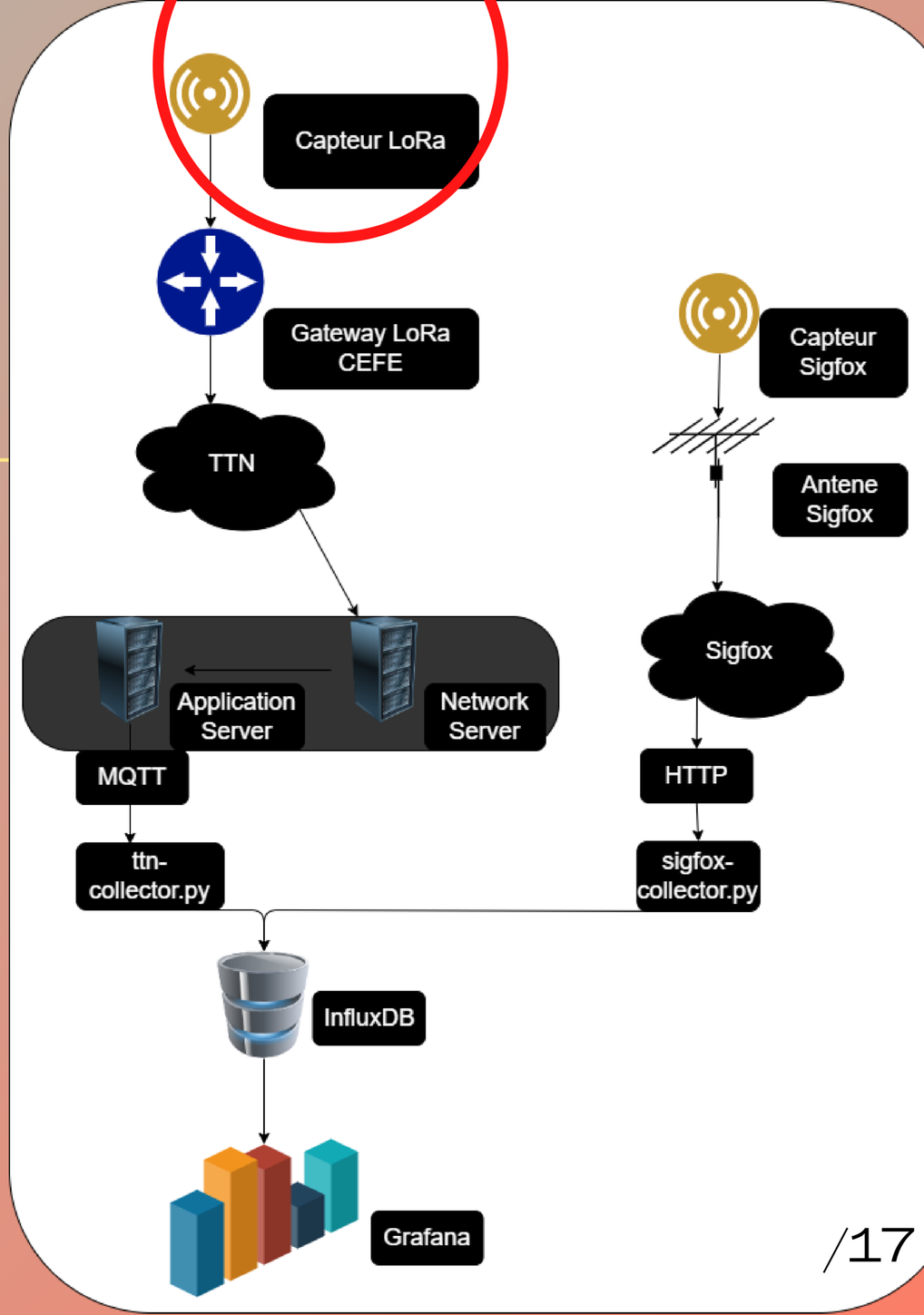
Notre solution pour répondre au cahier des charges

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



Notre solution pour répondre au cahier des charges

- Lecteur de tag RFID.
- Décodage manuel.

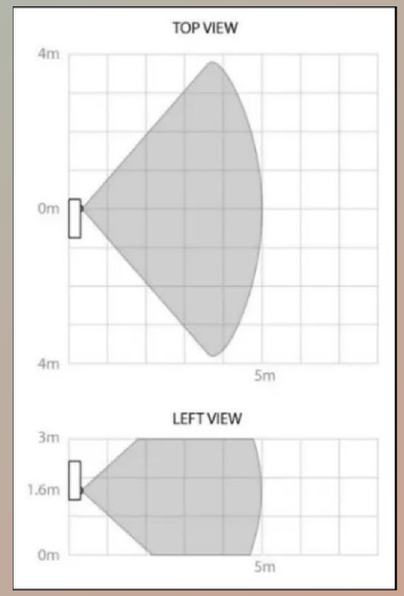


ERS2

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

Plusieurs utilité :

- Mouvement
- Température
- Humidité
- Luxmètre



Spécificité

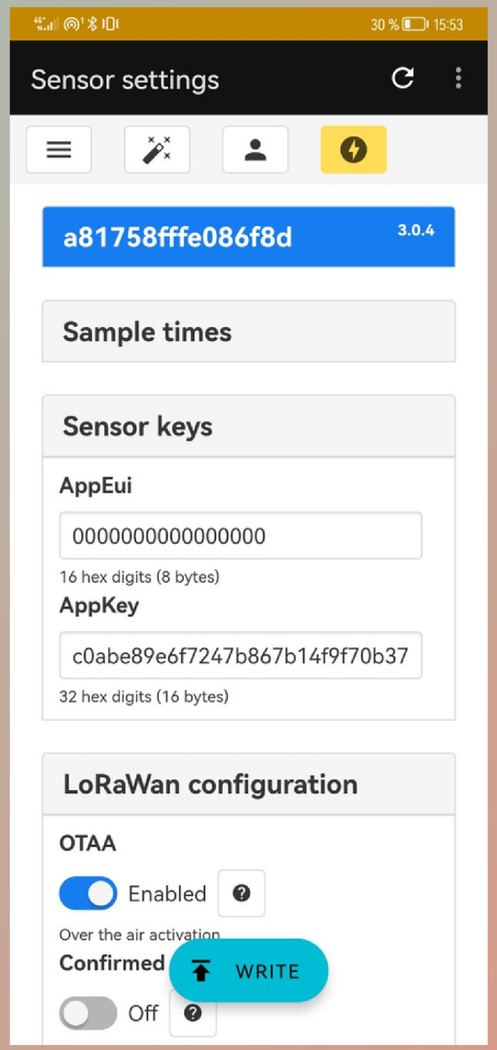
- Capteur infrarouge
- Distance de 5m
- Angle de détection de 90°

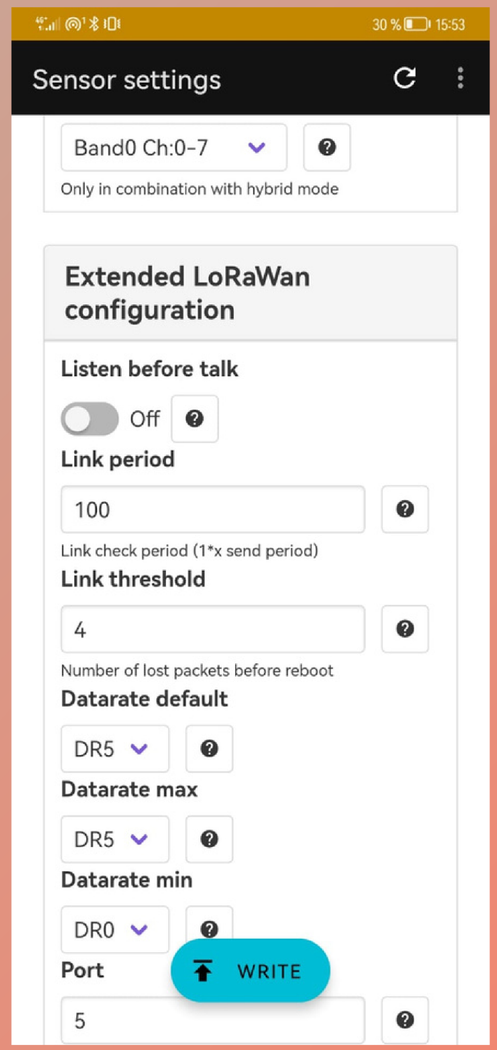
Mise en place du capteur

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 Adding Gateways.

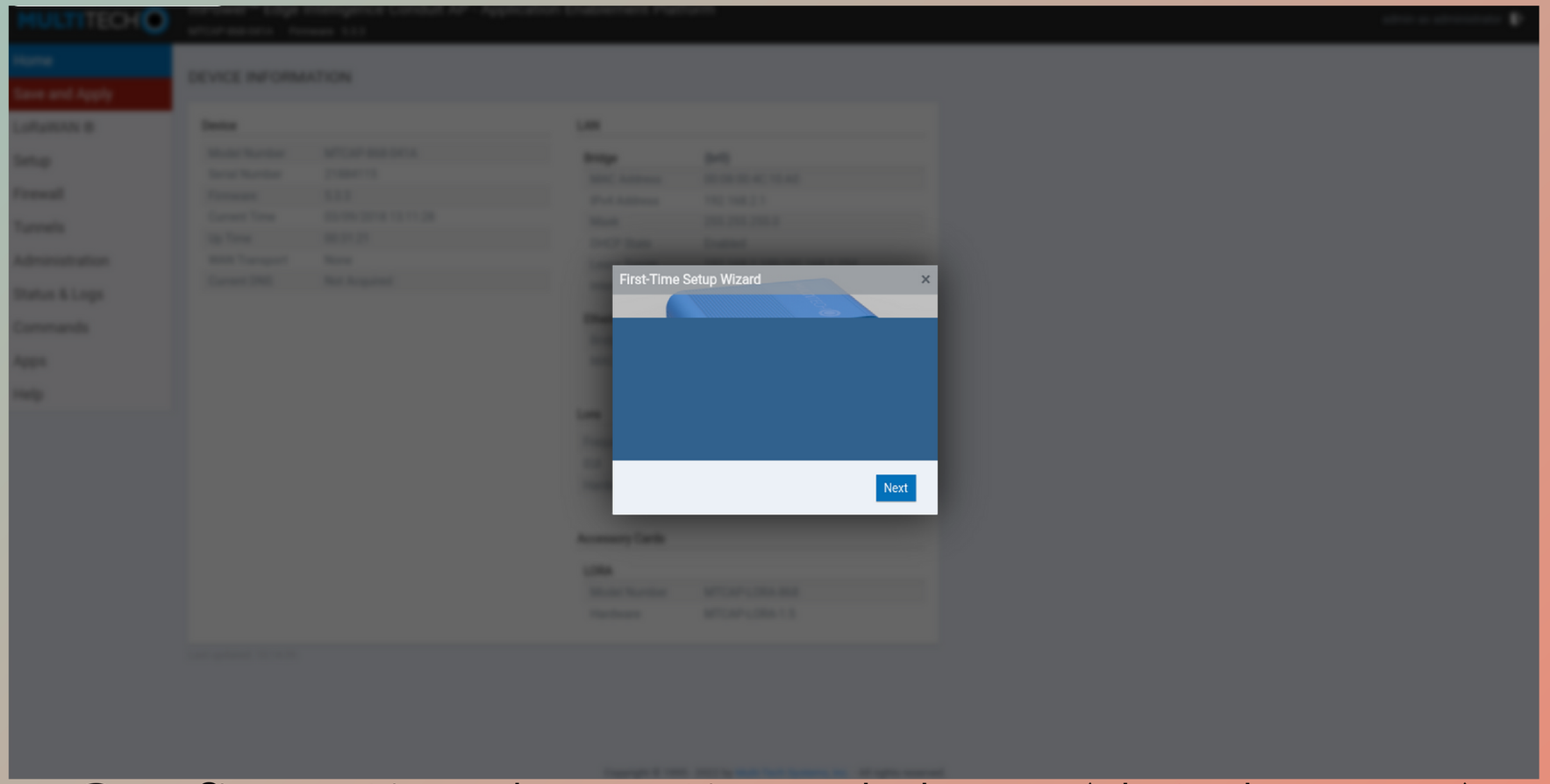
Gateway EUI

Gateway EUI | Continue without EUI

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

Configuration de la gateway

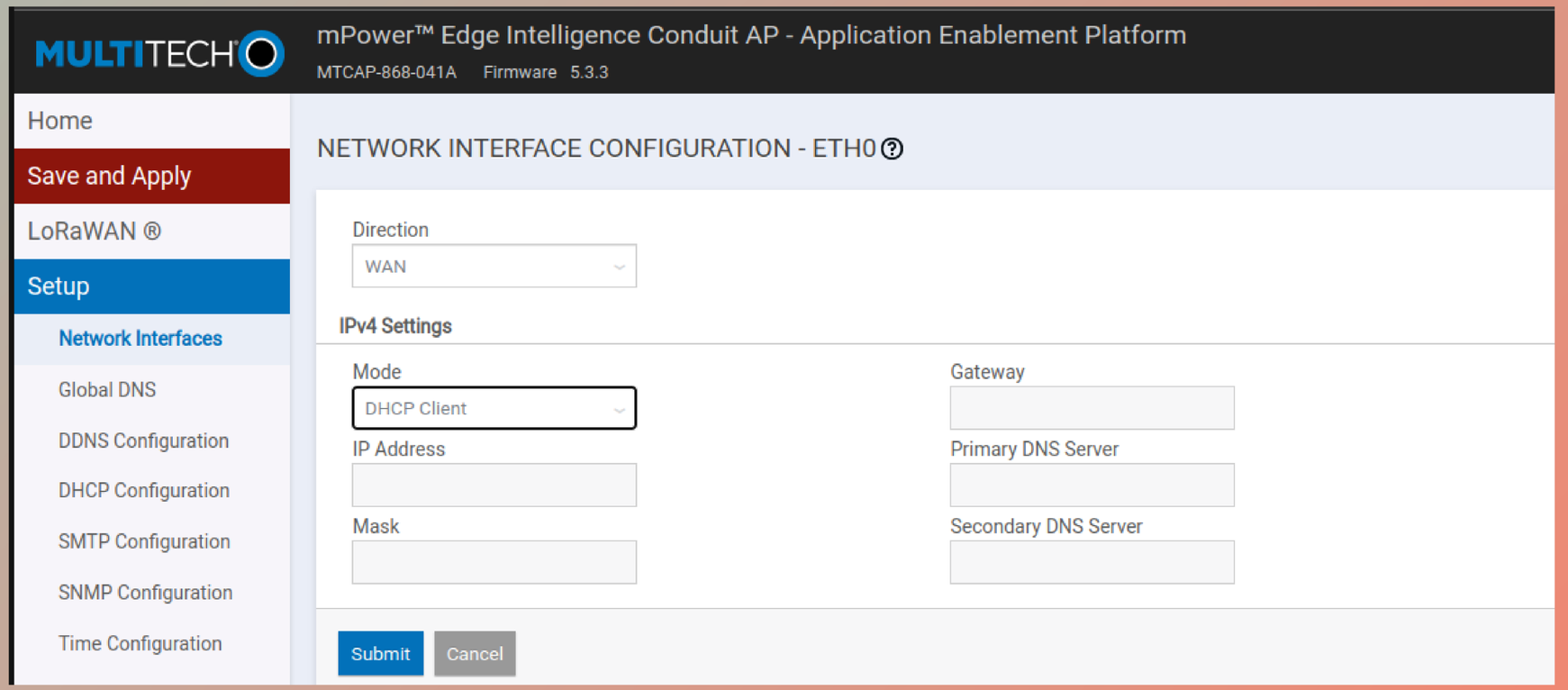
Connexion à la gateway sur le site avec l'ip 192.168.2.1



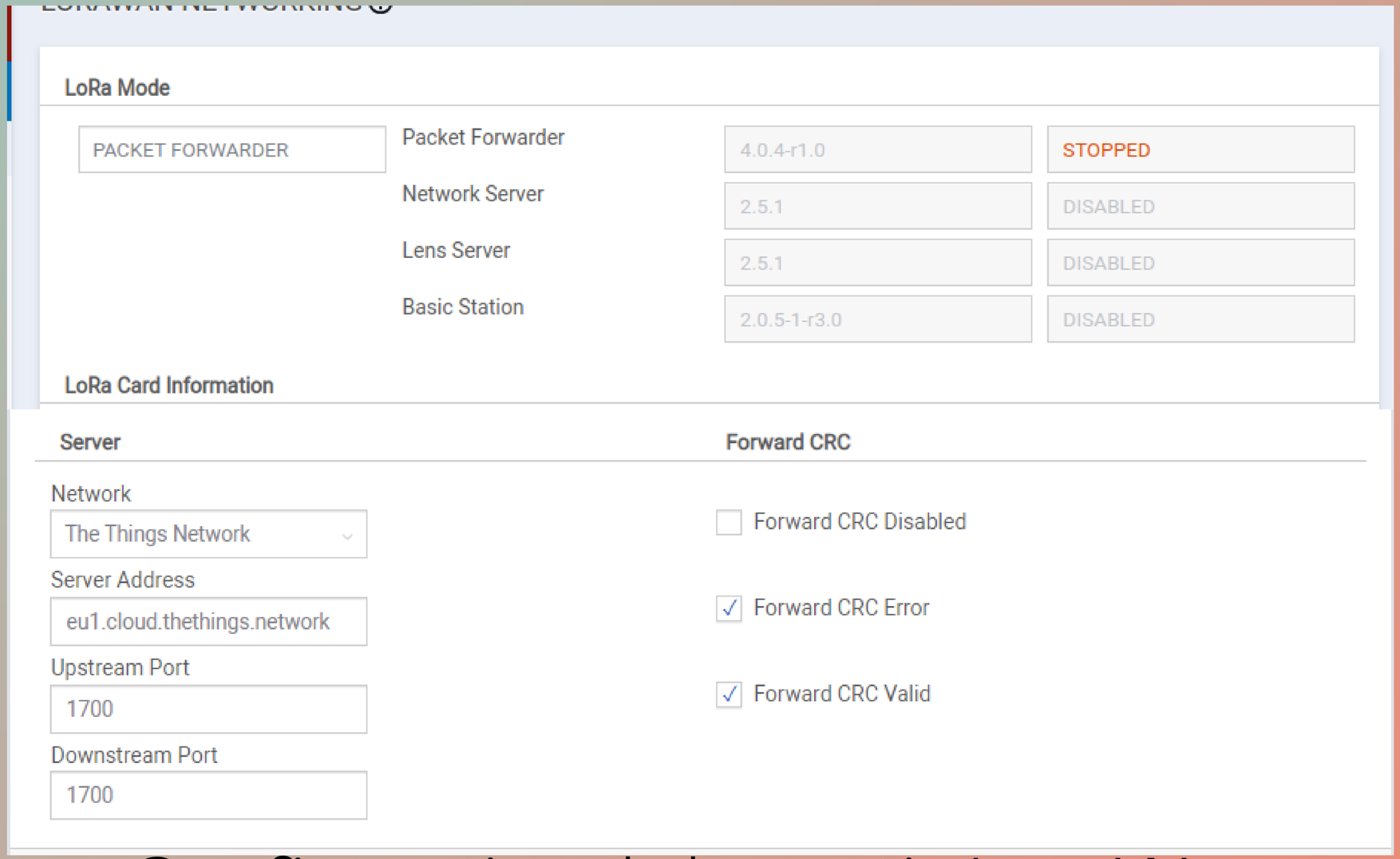
Configuration des options de base (date,heure ...)

Configuration de la gateway

Passage de la gateway en adressage dynamique



Configuration de la gateway



Configuration de la partie LoraWan

Système d'alerte

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

```python
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 a une alerte dans votre poulailler.

    Ci-dessous les relever pouvant être a 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 ?

```python
tempOk = True

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

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

    if tempOk == True :
        print("temOK 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)
```

```python
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
```