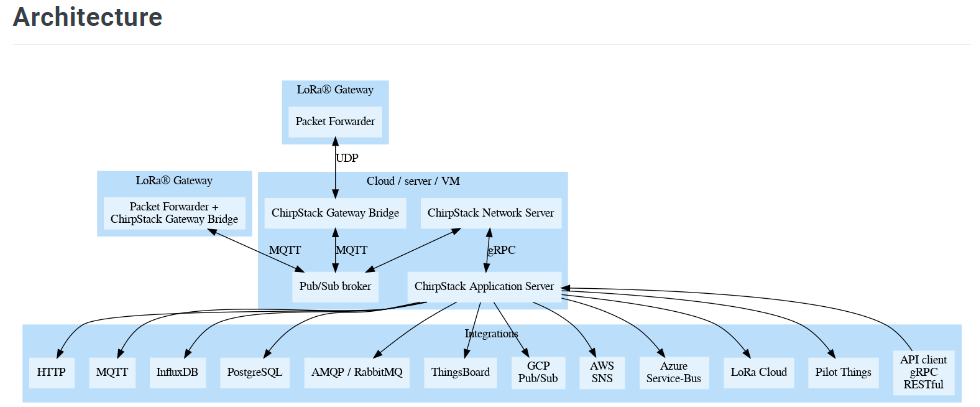
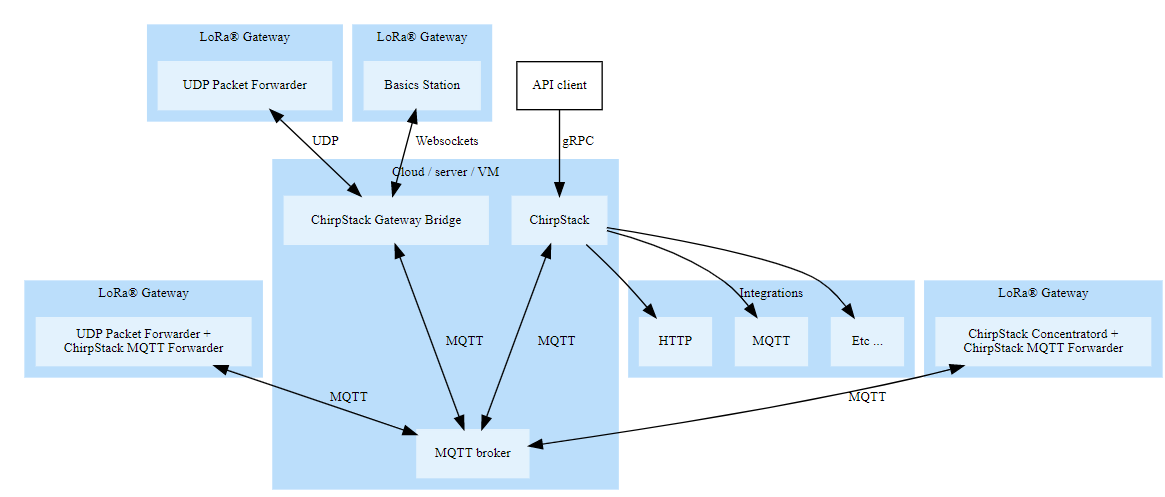
Chirpstack

# **1 Chirpstack**

Le serveur d'applications ChirpStack est un serveur d'applications LoRaWAN open source, qui fait partie de la pile de serveurs réseau LoRaWAN open source [ChirpStack.](https://www.chirpstack.io/)Il est responsable de la partie "inventaire" des nœuds d'une infrastructure LoRaWAN, de la gestion des charges utiles des applications reçues et de la file d'attente des charges utiles des applications en liaison descendante. Il est livré avec une interface Web et une API (RESTful JSON et gRPC) et prend en charge l'autorisation à l'aide de jetons JWT (facultatif). Les charges utiles reçues sont publiées sur MQTT et les charges utiles peuvent être mises en file d'attente à l'aide de MQTT ou de l'API.





**Backends pour le Packet Forwarder:**

Concentrateur Chirpstack (chirpstack concentratord)

Transmetteur de paquets UDP Semtech (Semtech UDP Packet Forwarder)

Transmetteur de paquets de la station de base (Basic Station Packet Forwarder)

**Chirpstack Concentratord**

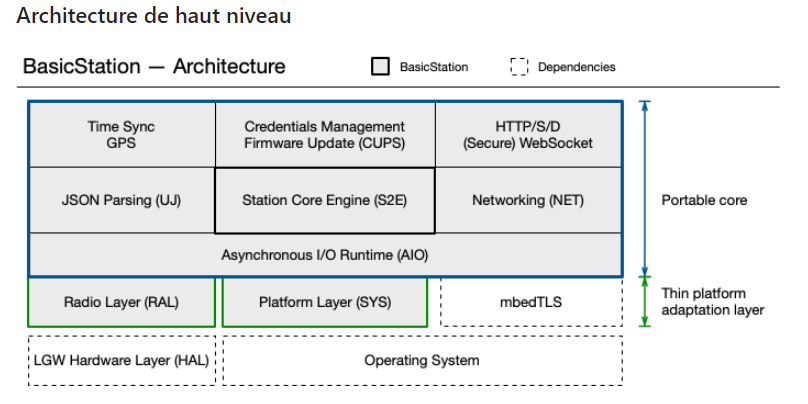
Concentrateur Chirpstack Concentratord est un démon de concentrateur LoRa open source. Il expose une API basée sur ZeroMQ qui est utilisée par une ou plusieurs applications (transitaires) pour interagir avec le matériel de la passerelle. En implémentant et en extrayant les spécificités matérielles dans un démon séparé et en les exposant via une API basée sur ZeroMQ, l'application de transfert de paquets peut être complètement découplée du matériel de la passerelle. Il permet également d'exécuter simultanément plusieurs applications de transfert de paquets.

**Paquet forwarder UDP Semtech**

L'expéditeur de paquets est un programme exécuté sur l'hôte d'une passerelle Lora qui transmet les paquets RF reçus par le concentrateur à un serveur via une liaison IP/UDP et émet les paquets RF qui sont envoyés par le serveur. Il peut également émettre un signal de balise synchrone GPS à l'échelle du réseau utilisé pour coordonner tous les nœuds du réseau.

Paquet forwarder de la station de base : est une implémentation de la passerelle LoRaWAN.

<https://github.com/lorabasics/basicstation>



Intégrations

Broker MQTT générique ou en cloud ou Azure.

**Intégrations :**

Les intégrations suivantes sont fournies

Broker MQTT générique (Generic MQTT broker)

Pont MQTT GCP Cloud Io Core (GCP cloud)

Pont MQTT Azure IoT Hub (Azure Hub)

**Chirpstack MQTT Forwarder**

Transitaire Chirpstack MQTT est un expéditeur Protobuf ou JSON MQTT opensource, qui peut utiliser le Semtech UDP Packet Forwarder ou le Chirpstack Concentratord comme backend de passerelle. Il est destiné à être installé sur chaque passerelle.

**Chirpstack Gateway Bridge**

C’est un service qui convertit les protocoles LoRa

Format de données commun (<https://github.com/chirpstack/chirpstack/blob/master/api/proto/gw/gw.proto>)

Le pont de passerelle Chirpstack est un pont open source qui convertit les messages reçus de l’expéditeur de paquets Semtech UDP ou de la station de base Semtech en MQTT. Il peut être installé sur la passerelle ou dans le cloud.

**Chirpstack**

Est un serveur de réseau LoRaWAN open source qui fournit une interface web pour la gestion des passerelles, des appareils et locataire et configure les intégrations des données. Chirpstack fournit une API basé sur gRPC pour intégrer ou étendre Chirpstack.

* [ChirpStack Gateway Bridge](https://www.chirpstack.io/gateway-bridge/) : gère la communication avec les passerelles LoRaWAN
* [ChirpStack Network Server](https://www.chirpstack.io/network-server/) : une implémentation de LoRaWAN Network Server
* [ChirpStack Application Server](https://www.chirpstack.io/application-server/) : une implémentation du serveur d'application LoRaWAN
* [ChirpStack Gateway OS](https://www.chirpstack.io/gateway-os/) : système d'exploitation basé sur Linux pour exécuter la pile ChirpStack (complète) sur une passerelle LoRa basée sur Raspberry Pi

**Stratégie de déploiement**

Le plus simple connecter toutes vos passerelles à une seule instance du pont de passerelle Chirpstack, solution simple. L’option la moins sécurisé. Le protocole UDP implémenté par la plupart des passerelles ne prend en charge aucune forme d’autorisation et vérifie que les données reçues sont authentiques. C’est le moyen le plus facile de commencer.

Puisque l’installation du pont passerelle Chirpstack sur la passerelle peut impliquer des étapes supplémentaires.

Pour des performances hautes, nous pouvons exécuter le bridge gateway chirpstack sur plusieurs serveurs, chacun se connectant au même courtier MQTT.

Le gateway bridge chirpstack installer sur chaque passerelle donne des étapes supplémentaires dans la configuration mais présente les avantages :

*MQTT (utilisant TCP)* au lieu de UDP lors de l’utilisation du backend de transfert de paquets Semtech UDP, la connexion devient plus fiable au cas où la perte de paquets serait courante.

*Authentification* possible de configurer des informations d’identification pour chaque passerelle de sorte que seules les passerelles avec des informations d’identification valides puissent ingérer des données.

*SSL/TLS* le protocole MQTT prend en charge SSL/TLS donc une connexion sécurisée entre vos passerelles et le courtier MQTT donc personne n’est en mesure d’intercepter des données et de les altérer.

**Chirpstack Network Server** : Une implémentation de LoRaWAN Network Server.

Pour la déduplication des trames lorawan reçues par les passerelles LoRa et gérer les trames collectées.

Authentification, couche mac lorawan et commande mac, communication avec le serveur d’applications chirpstack et ordonnancement des trames descendantes.

Configuration : /etc/chirpstack-network-server/chirpstack-network-server.oml

Il a besoin de postgres et utilise un courtier MQTT pour publier et recevoir des payloads de l’application.

Il utilise redis pour stocker les données de session de périphérique et les données non persistantes (verrous distribués, metadonnées)

**Backends**

* AMQP/LapinMQ
* Hub Iot Azure
* Pub/Sub GCP
* MQTT

Comme la livraison de message circulaires n’est pas possible avec (tous) les courtiers MQTT ce backend implémente sa propre déduplication pour s’assurer qu’un évènement de passerelle n’est géré qu’une seul fois.

**Chirpstack Application Server**

Inventaire et interface web et propose une API gRPC et Restful.

# **2 Installation de Chirpstack Ubuntu/Debian**

1. Installer les exigences Chirpstack avec le gestionnaire de package apt

> sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql

En supplément

postgresql-common postgresql-10 postgresql-client-10 postgresql-client-common postgresql-contrib

1. Configuration de PostgreSQL

> sudo -u postgres psql

:invite de postgresSQL

-- create role for authentication

> create role chirpstack with login password 'chirpstack';

-- create database

> create database chirpstack with owner chirpstack;

-- change to chirpstack database

> \c chirpstack

-- create pg\_trgm extension

> create extension pg\_trgm;

-- exit psql

> \q

1. Chirpstack fournit un référentiel Debian/Ubuntu pour installer la dernière version de Chirpstack

:Assurez-vous d’abord que dirmgnr et apt-transport-https sont installés :

--installer le repository de Chirpstack

> sudo apt install apt-transport-https dirmngr

--configurez la clé pour le dépôt Chirpstack

> sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00

--ajouter le dépôt à la liste des dépôts :

> sudo echo "deb https://artifacts.chirpstack.io/packages/4.x/deb stable main" | sudo tee /etc/apt/sources.list.d/chirpstack.list

--mettez à jour le cache du package apt

> sudo apt update

1. Installer le pont de passerelle Chirpstack (Chripstack gateway bridge)

Remarque : si vous avez l’intention d’installer le pont Chirpstack uniquement sur la ou les passerelles elles-même, vous pouvez ignorer cette étape.

-- installer le pont de passerelle chirpstack

> sudo apt install chirpstack-gateway-bridge

4 bis)

--Configuration du pont

Le fichier dans etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

--Mettre à jour [integration.mqtt] pour correspondre au préfixe de la région

Europe:

[integration.mqtt]

event\_topic\_template="eu868/gateway/{{ .GatewayID }}/event/{{ .EventType }}"

command\_topic\_template="eu868/gateway/{{ .GatewayID }}/command/#"

1. démarrer le service Chirpstack Gateway Bridge

> start chirpstack-gateway-bridge

> sudo systemctl start chirpstack-gateway-bridge

-- start chirpstack-gateway-bridge on boot

> sudo systemctl enable chirpstack-gateway-bridge

1. Installer Chirpstack

> apt install chirpstack

--la configuration /etc/chirpstack/chirpstack.toml « configuration de chirpstack

1. Démarrez le service Chirpstack

Exemple de configuration des region /etc/chirpstack/region\_eu868.toml

# start chirpstack

> sudo systemctl start chirpstack

#start chirpstack on boot

> sudo systemctl enable chirpstack

--Imprimez la sortie du journal Chirpstack

> sudo journalctl -f -n 100 -u chirpstack

# **3 Passerelle LoRa**

Le transmetteur de paquets installé sur votre passerelle et les étapes nécessaires pour installer le Chirpstack Gateway Bridge varient selon le fournisseur et le modèle de la passerelle et dans certain cas vous devez installer le Chirpstack Gateway Bridge sur la passerelle.

KERLINK

Passerelle kerlink istation 868

poe 48v/140ma

dc 42-57/170ma

productid:237CKe0539C2

boardid: 235CHi0901E0

@mac: 70:76:FF:03:81:25

EUI: 7076FF00560901E0

Chirpstack: <http://172.16.0.90:8080/> admin:admin

kerlink : <http://172.16.0.24/> login:admin password: pwd4admin ou

password :admin (admin :admin)

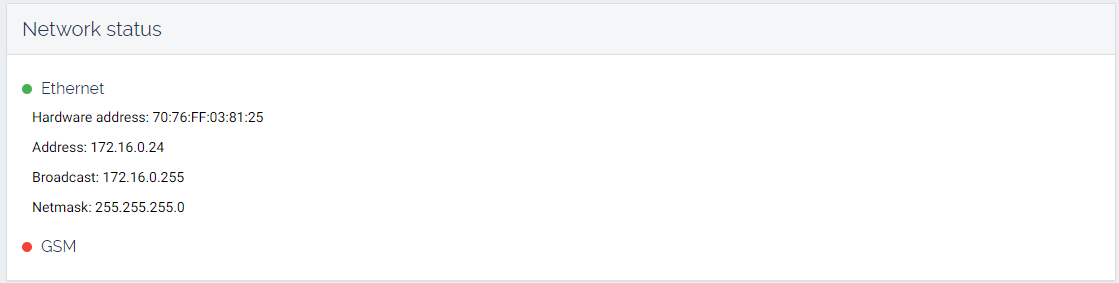
Kerlink/ssh : ssh [root@172.16.0.24](mailto:root@172.16.0.24) password : pdmk-0901E0

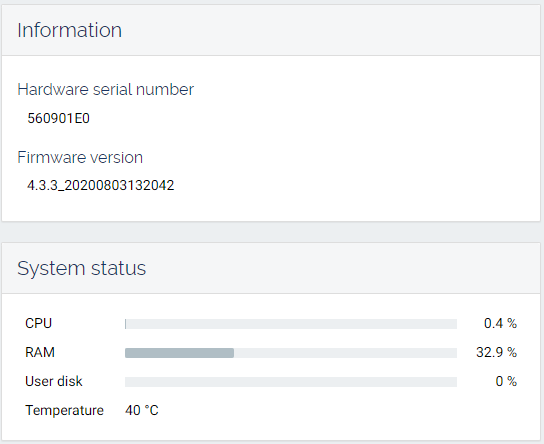
Documentation : <https://wikikerlink.fr/>

Documentation : <https://wikikerlink.fr/wirnet-productline/doku.php> login: uca password: ASRy0iOwn3

Documentation : <https://assistance.sphinxfrance.fr/home/Kerlink>

<http://172.16.0.24/> login:admin password: pwd4admin ou password :admin (admin :admin)





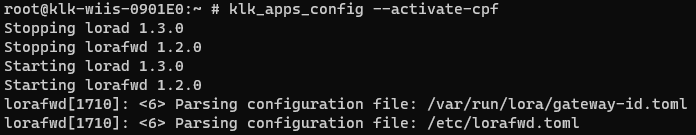
**Configuration de la passerelle**

>ssh [root@172.16.0.24](mailto:root@172.16.0.24) password : pdmk-0901E0

1) Activer Kerlink CPF

Par défaut kerlink Common Packet-Forwarder (CPF) est désactivé. Il faut l’activer :

> klk\_apps\_config –activate-cpf



2) Changer la fréquence de la passerelle

Dans /etc/lorad/ fichier json de fréquence en fonction des pays et des types de passerelles.

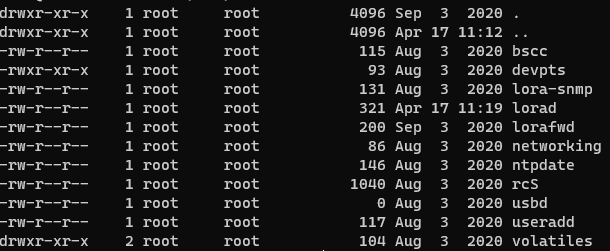
-- iStation: vi /etc/lorad/wiis/EU868-FR.json

-- iFemtocell: vi /etc/lorad/wifc/EU868-FR.json

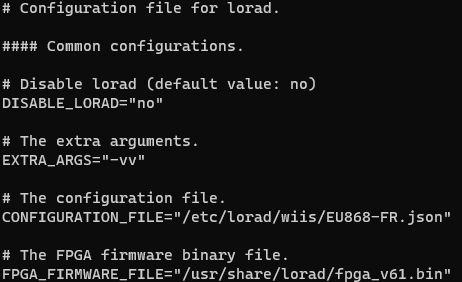
-- iFemtocell-Evolution: vi /etc/lorad/fevo/EU868-FR.json

Choix kerlink istation >cd /etc/lorad/wiis/EU868-FR.json

Dans /etc/default/ ls -al



>cd /etc/default/nano lorad



Origine : CONFIGURATION\_FILE= "configurer le chemin de la bonne fréquence du pays "

. (en vidéo DISABLE\_LORAD="no") <https://www.youtube.com/watch?v=mkuS5QUj5Js>

loradfwd desable=no si on est pas un transitaire à voir

3) Configurer la connexion au server chirpstack

> vi /etc/lorafwd.toml

[gateway]

Id = 0x7076FF00560901E0

[gwmp]

node = " 172.16.0.90" #ip du serveur chirpstack

service.uplink = 1700

service.downlink = 1700

Redémarrer le service : > monit restart lorafwd

4) les commandes de diagnostic de la passerelle.

Relancer le lora de la passerelle

>/etc/init.d/lorafwd restart

>/etc/init.d/lorad restart

>tail -f /var/log/lora.log

>grep EUI /tmp/board\_info.json



Commande de (re)démarrage et d’arrêt des services :

# status

> monit status chirpstack-gateway-bridge

# start

> monit start chirpstack-gateway-bridge

# stop

> monit stop chirpstack-gateway-bridge

# restart

> monit restart chirpstack-gateway-bridge

Ou mais ne fonctionne pas

/etc/init.d/ chirpstack-gateway-bridge stop

Commandes de log

> more /var/log/user.log

> tail -f /var/log/lora.log

La plate-forme utilise systemd ou init.d

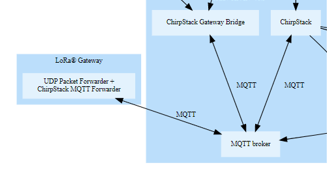
* Systemctl status chirpstack-gateway-bridge
* /etc/init.d/chirpstack-gateway-bridge status

Sortie du journal du pont de passerelle chirpstack pour vérifier que les données arrivent réellement :

Plate-forme systemd ou init.d

* journalctl -u chirpstack-gateway-bridge -f -n 50
* jail -f /var/log/chirpstack-gateway-bridge/chirpstack-gateway-bridge.log

5) Si nous voulons installer la gateway sur la passerelle au lieu de l’installer dans le server chirpstack



Installer la chirpstack gateway bridge dans la passerelle :

**A] Installation et configuration du Gateway Bridge**

Dans la doc connecs

Copier depuis le pc avec mobaxterm scp le chirpstack-gateway-bridge\_3.10.0-r1\_klkgw.ipk

> mkdir -p /user/.updates

> mv chirpstack-gateway-bridge\_3.10.0-r1\_klkgw.ipk /user/.updates

> sync

> kerosd -u

> reboot

Sinon télécharger depuis le serveur de chirpstack, dans la doc chirpstack :

Trouvez le dernier package sur <https://artifacts.chirpstack.io/vendor/kerlink/keros-gws/>

> cd /user/.updates

> wget https://artifacts.chirpstack.io/vendor/kerlink/keros-gws/chirpstack-gateway-bridge\_4.0.6-r1\_klkgw.ipk

Pour déclencher la passerelle pour installer/mettre à jour le package exécuter les commandes :

> sync

> kerosd -u

> reboot

La passerelle va vérifier le contenu du dossier /user/.updates et installer le package qu’il contient.

**B] après le reboot passerelle**

> vi /etc/ chirpstack-gateway-bridge/**chirpstack-gateway-bridge.toml**

Section [integration.mqtt.auth.generic]

Server= " tcp://172.16.0.90 :1883" #l’adresse du server tcp

Section[backend.semtech\_udp]

udp\_bind = " 0.0.0.0 :1700" #vous pouvez changer de port exemple : 2000

**C] Configurer la connexion MQTT ???**

Le fichier de configuration /etc/cat lorafwd.toml

[gateway]

L’identifiant de la passerelle, pour l’identifier à l’intérieur du réseau, de longueur 64bits il peut être exprimé en hexadécimal pour une meilleure lisibilité.

[filter]

Indique si un message de liaison montante avec un CRC valide sera transmis ou non.

Indique si une liaison descendante LoRaWan sera transmise ou non en tant que message de liaison montante.

S’il est défini, seuls les messages de liaison montante avec devaddr correspondant à ce netid seront transférés.

[database]

Si une base de données persistante stockait ou non les messages entrants jusqu’à ils seraient envoyés et reconnus.

Le nombre maximum de messages pouvant être stockés dans la base de données. Quand lce sera plein le message le plus récent remplacera le plus ancien.

Le délai minimum entre deux extractions de base de données. Pour autoriser les messages entrants à agréger avant d’être envoyé.

[gwmp]

L’hôte Internet auquel la passerelle doit se connecter, le nœud peut être soit un nom de domaine complet ou une adresse IP (IPV4 ou IPV6)

Les services GWMP peuvent être un nom de service (vois services 5) ou un entier et dans ce cas fait référence à un port réseau.

Le service où la passerelle doit extraire les messages de liaison descendante.

La période des battements de coeur. Utilisé pour garder le pare-feu ouvert.

La période des statistiques. Utilisé pour envoyer des statistiques.

Le nombre de messages expirés qui déclencheront automatiquement un réseau, redémarrage du socket. Utilisé pour surveiller la connexion.

La taille maximale du datagramme pour les messages de liaison montante. Le datagramme inclut le GWMP en-tête et charge utile.

Le point de terminaison pour contrôler le démon LoRa. Utilisé pour demander des statistiques.

[api]

L’API utilse ZeroMQ comme couche de transport plus d’informations sur ZeroMQ, le format de points de terminaison peut être trouvé ici : <http://api.zeromq.org/4-2:zmq-connect>

Les points de terminaison pour le canal de liaison montante. Utilisé pour recevoir des messages de liaison montante.

Les points de terminaison pour le canal de liaison descendante. Utilisé pour envoyer des messages de liaison descendante.

Les points de terminaison du canal de contrôle. Utilisé pour recevoir la demande de contrôle.

Les filtres pour le canal de liaison montante. Utilisé pour s’abonner aux messages de liaison montant.

Et les filtres peuvent gérer le binaire brut ou s’abonner à tous les messages entrants.

Configuration minimale

/etc/lorafwd.toml

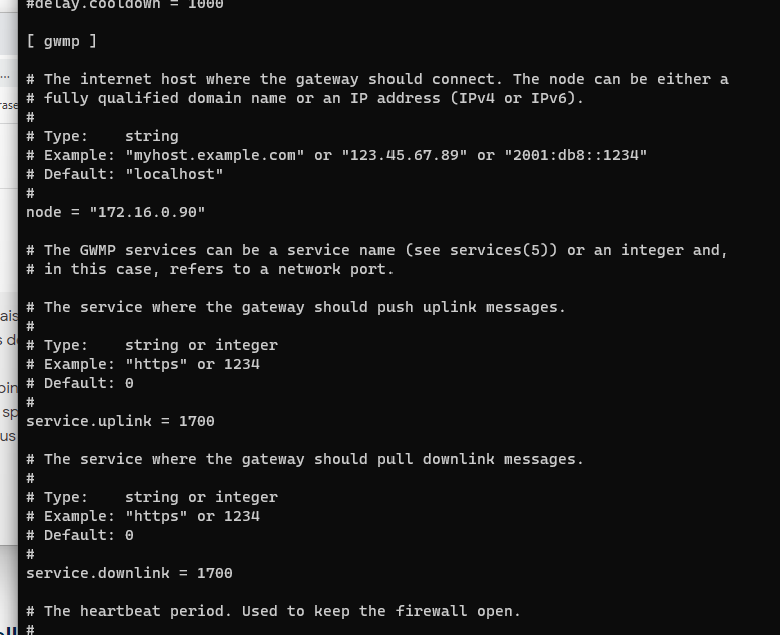
Restart les services

/etc/init.d/lorafwd restart

/etc/init.d/lorad restart

Afficher

tail -f /var/log/lora.log

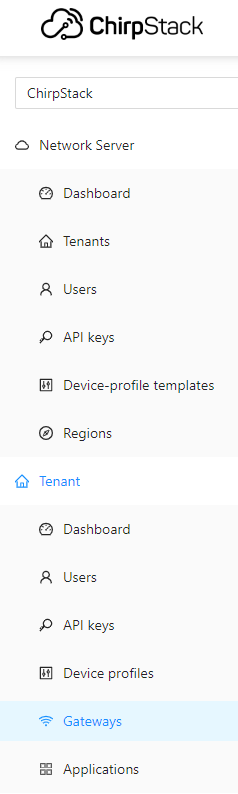


172.16.0.90 serveur chirpstack 172.16.0.91 passrelle kerlink

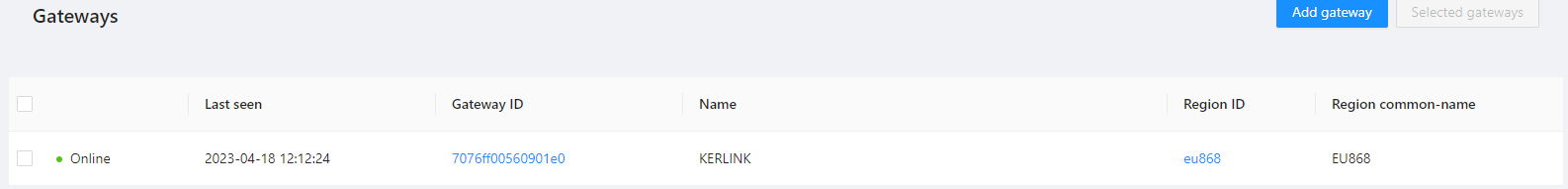
22/tcp ssh et 8080/tcp http-proxy 631/udp ipp et 5353/udp zeroconf 22/tcp ssh et 80/tcp http

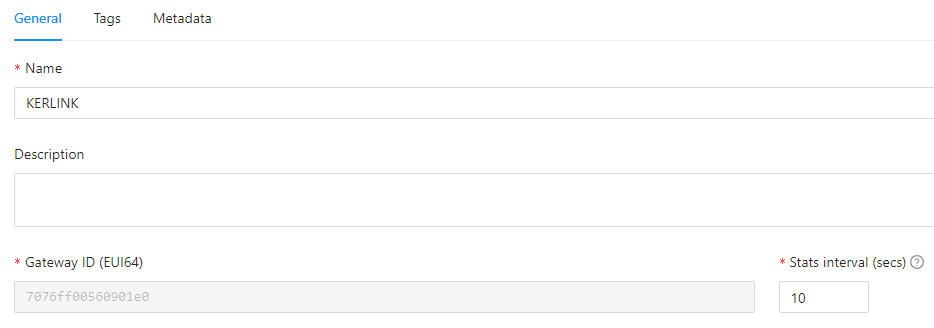
# **4 Configuration ChirpStack**

Menu:

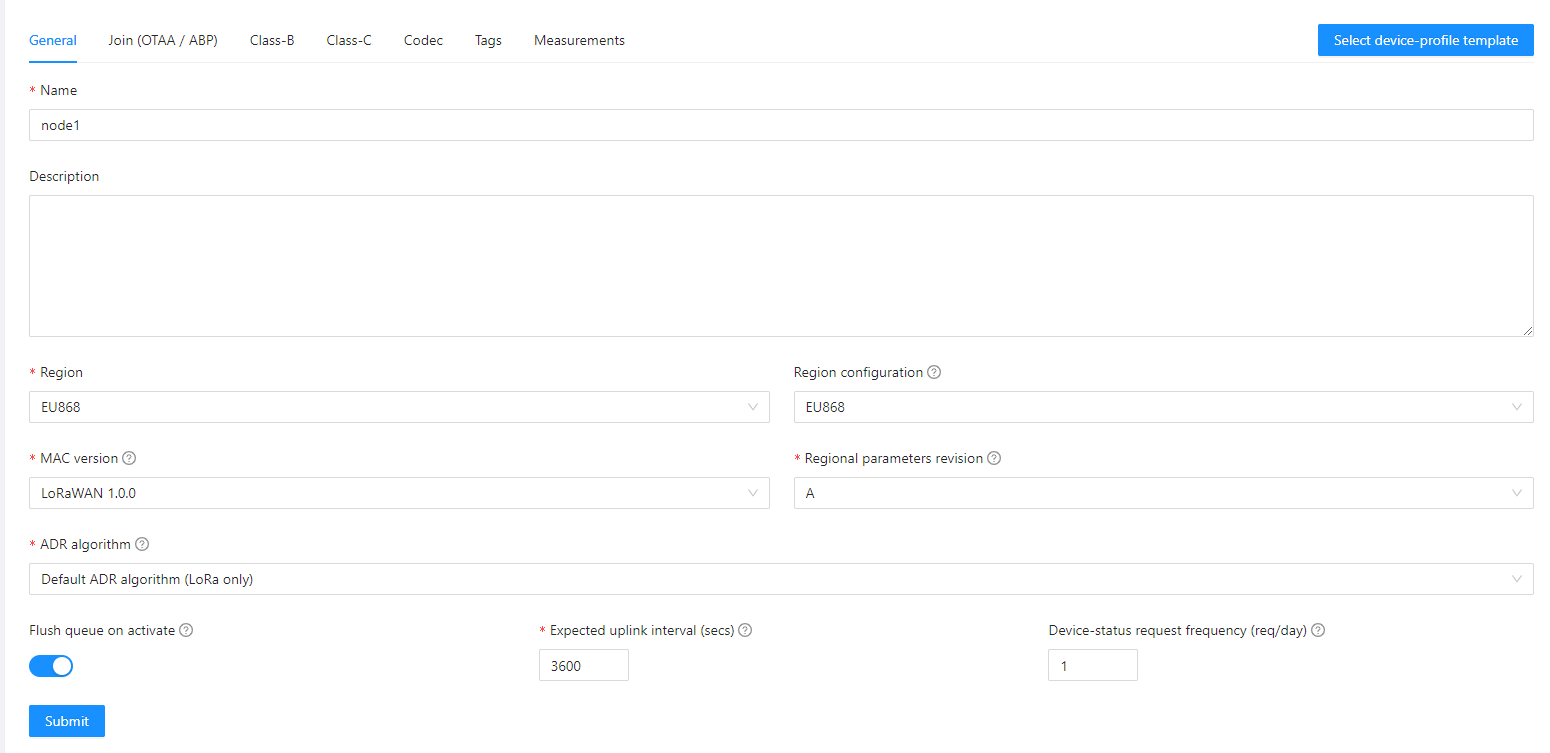


Gateway:

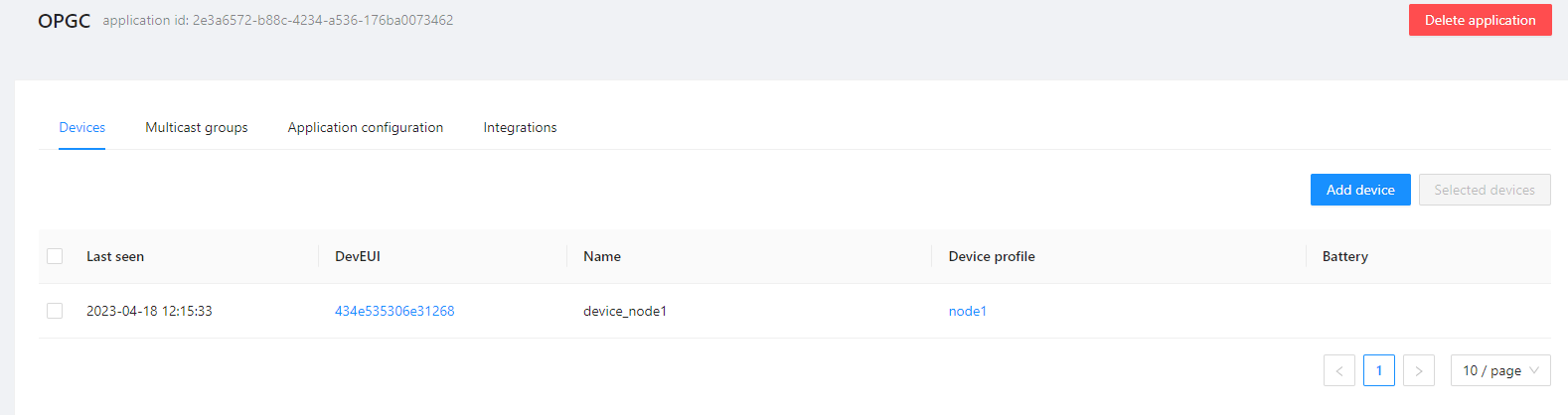


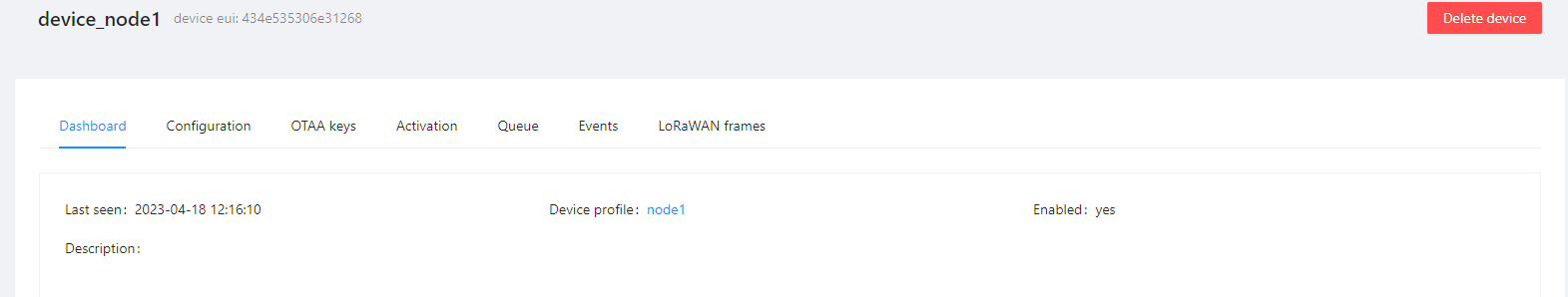


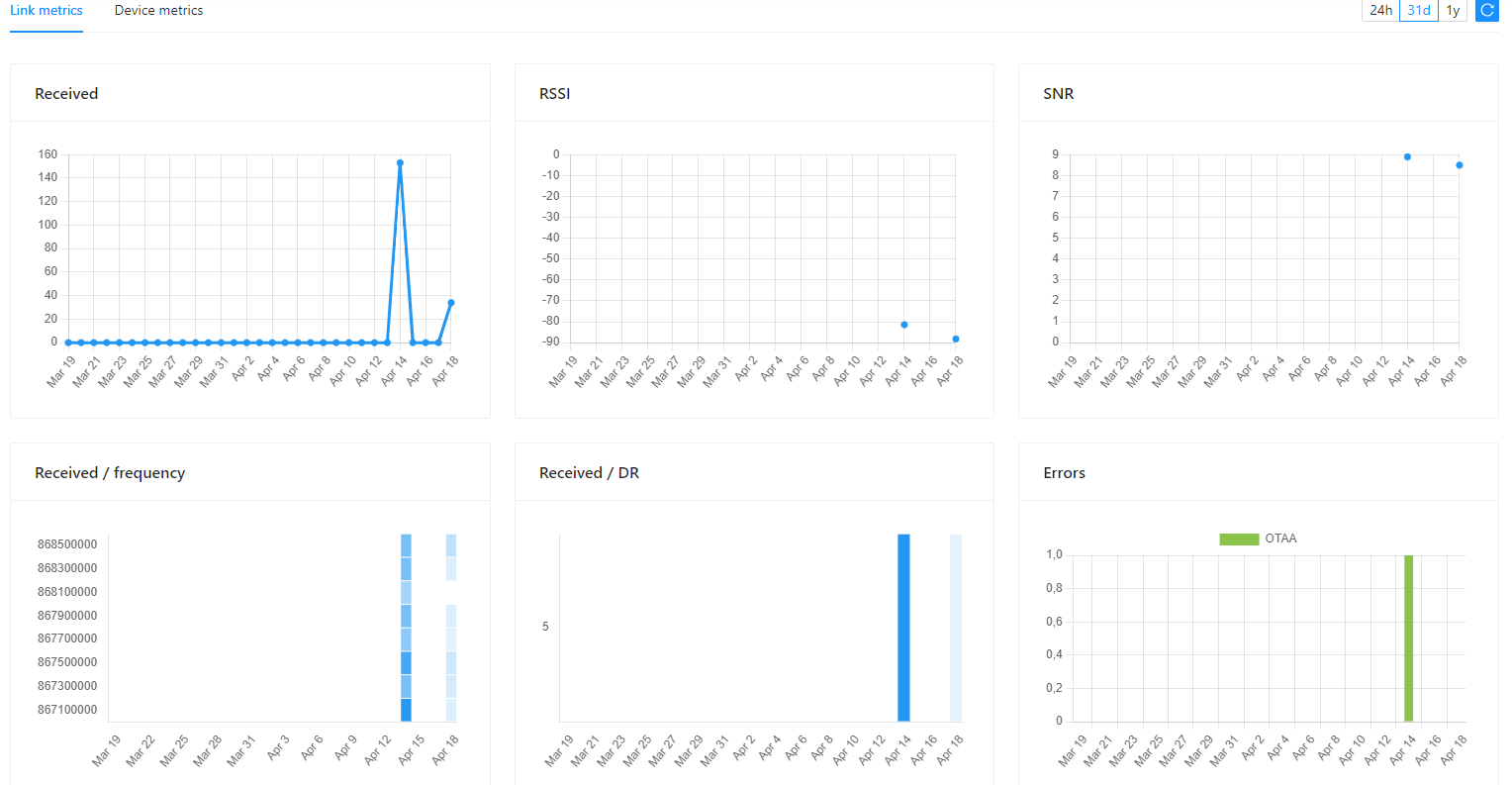
Device profiles

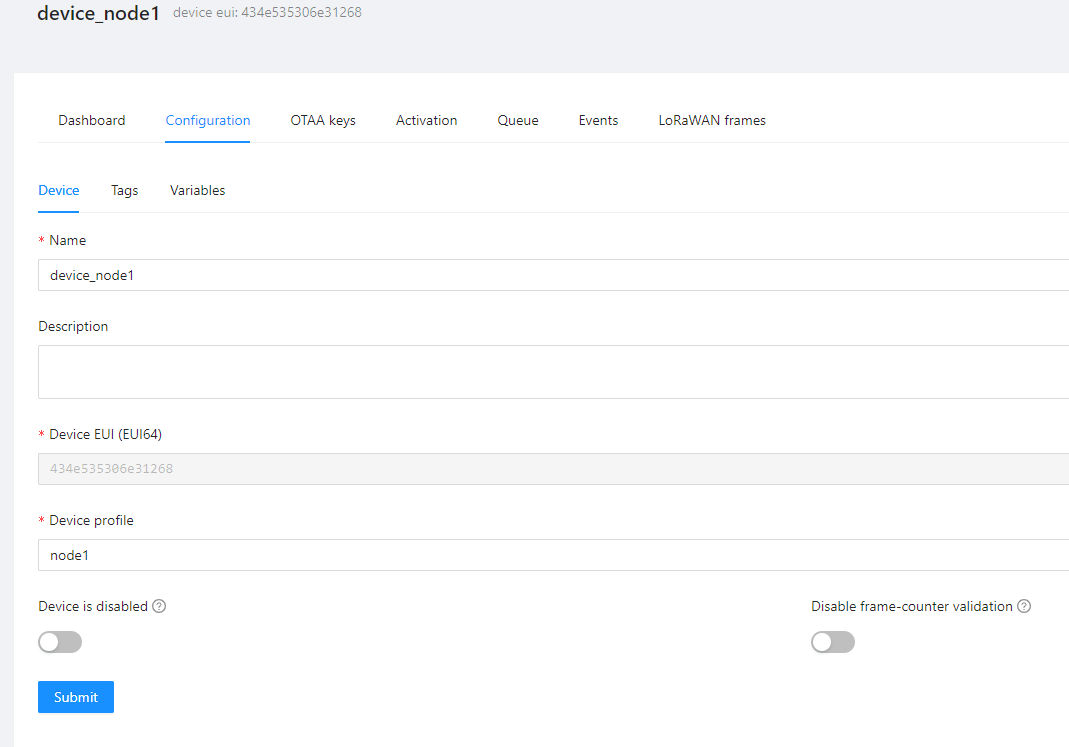


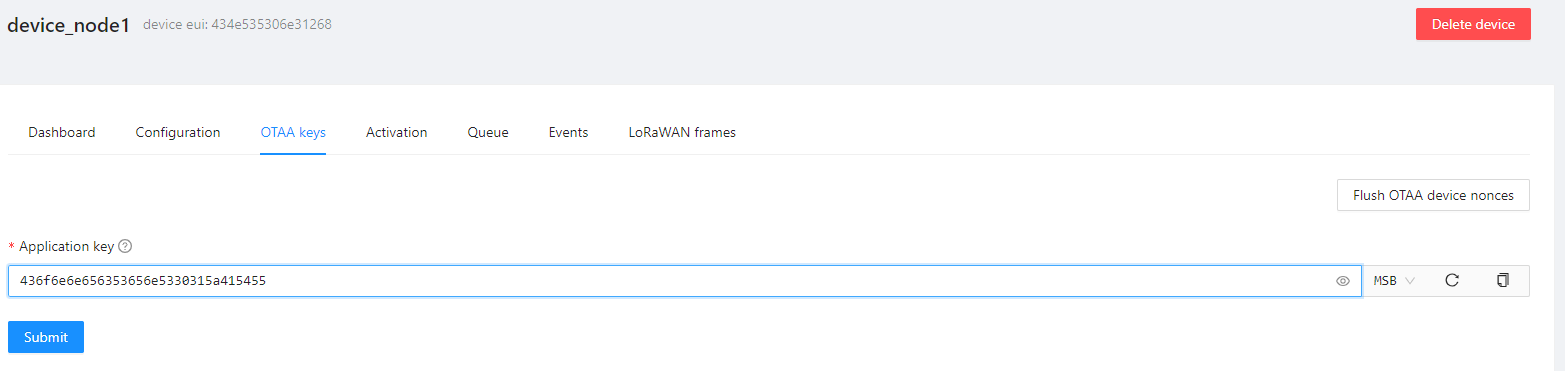
Applications:



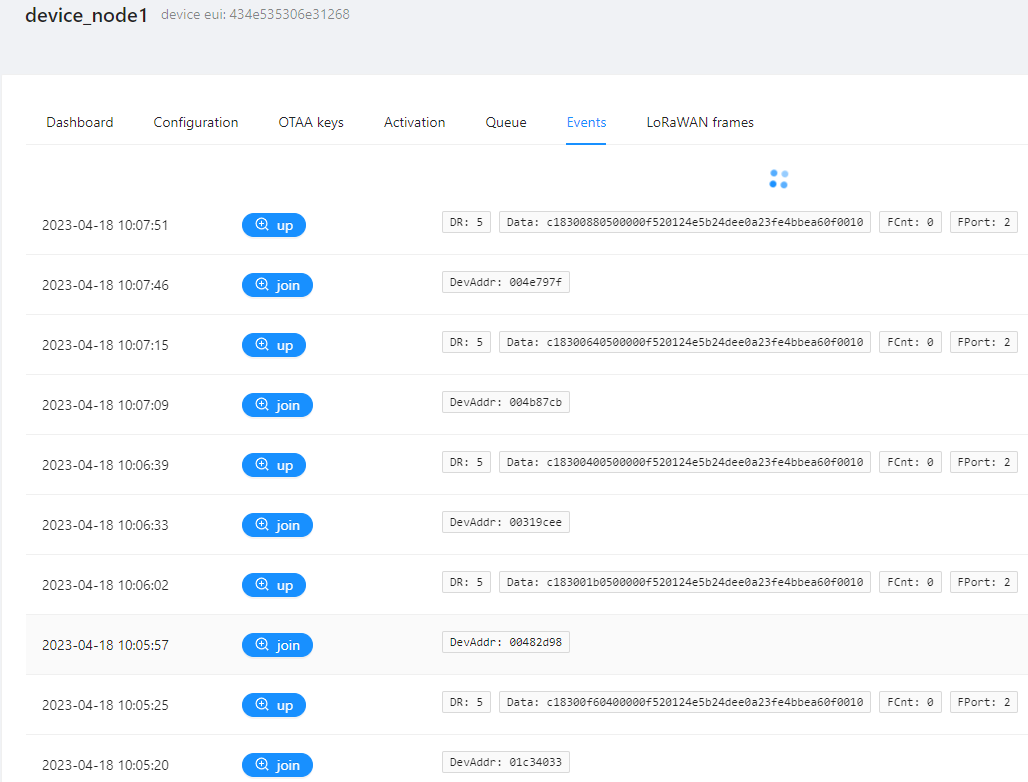


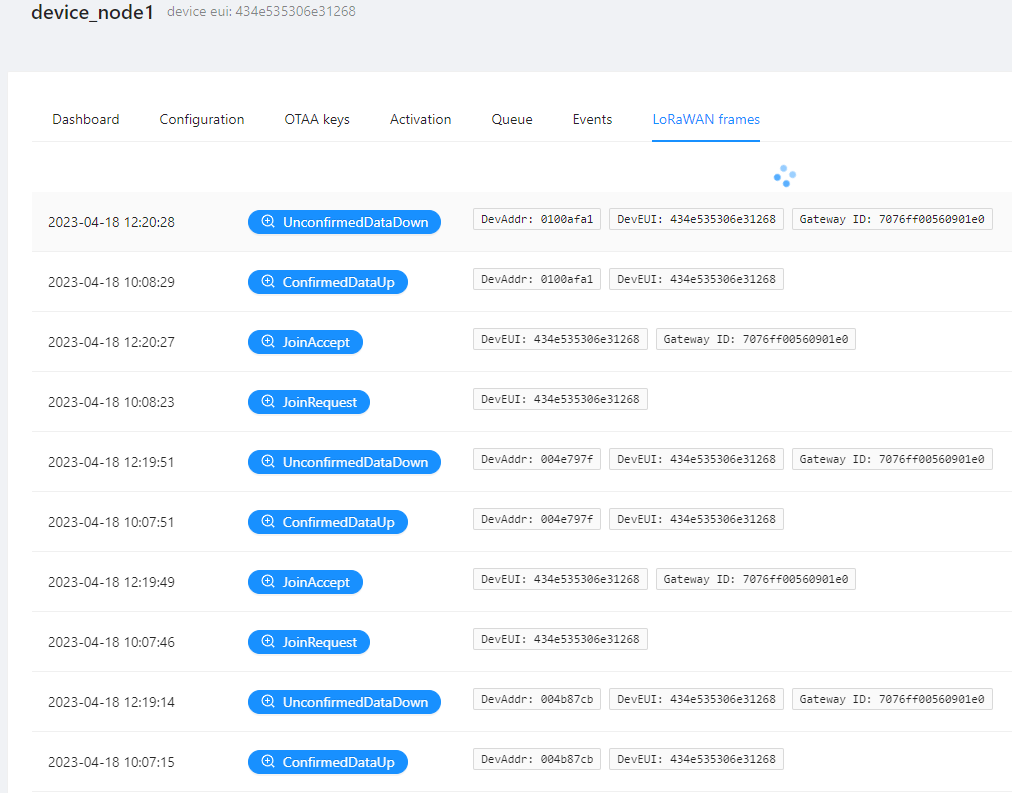












# **5 Node LoRa**

------------Config passerelle----------------

Passerelle kerlink istation 868

poe 48v/140ma

dc 42-57/170ma

productid:237CKe0539C2

boardid: 235CHi0901E0

@mac: 70:76:FF:03:81:25

EUI: 7076FF00560901E0

------------Config noeud----------------

"devEUI": "434E535306E31268", #unique au noeud

"appEUI": "A78F1729918331B4", #tous

"appKey": "436f6e6e656353656e5330315a415455", #tous

------------Config noeud----------------

"devEUI": "434E535306E31268", #unique au noeud

"appEUI": "A78F1729918331B4", #tous

"appKey": "436f6e6e656353656e5330315a415455", #tous

# **6 LoRaWAN**

DevEUI\_Identifie le end-device

AppEUI-Identifie l’application

GatewayEUI-Identifie la passerelle

DevAddr-Adresse du device sur le réseau

DevEUI (Extended Unique Identifier) comme l’adresse MAC du nœud.

Fournit par le fabricant de l’appareil, c’est plutôt un algorithme pour convertir les adresses MAC 48bits en EUI64. Algorithme déprécié car il peut entraîner des collisions avec d’autre DevEUI. Lors de l’activation en direct, un DevAddr est attribué à l’appareil. Ce DevAddr est ensuite utilisé dans le protocole LoRaWAN. Le DevEUI est envoyé non chiffré.

AppEUI reformulé par le JoinEUI est un ID d’application global dans l’espace d’adressage IEEE EUI64 identifiant le serveur de jointure lors de l’activation en direct. Cela signifie que chaque application a au moins un AppEUI et si vous avez vos propres AppEUI vous pouvez également les ajouter à votre application.

Pour les réseaux non privés, il correspond à un sous-domaine de joineuis.lora-alliance.org. Ce nom de serveur est utilisé pour trouver l’adresse IP un serveur de jointure via DNS.

L’APPEUI peut être différent pour chaque appareil ou il peut être le même pour tous les appareils. Cela dépend du type de serveur d’application Dans loraserver.io vous pouvez avoir une AppKey, AppEUI, DevEUI uniques mais dans TheThingsNetwork vous pouvez enregistrer plusieurs appareils pour une application ou AppEUI.

Mais AppKey et DevEUI doivent être uniques pour chaque nœud final.

GatewayEUI chaque passerelle fabriquées intègre un EUI qui est utilisé pour enregistrer la passerelle sur le réseau (Chirpstack / The things network). Si votre passerelle n’est pas fournie avec un EUI intégrée, vous pouvez utiliser une autre EUI que vous possédez ou en configurer une AppEUI qui est enregistrée sur votre compte. Vous pouvez également utiliser un redirecteur basé sur MQTT qui n’a besoin que d’un GatewayID (choisir vous-même) au lieu d’un GatewayEUI.

AppKey est une AES128 clé racine spécifique à l’appareil final. Chaque fois qu’un appareil final rejoint un réseau via l’activation sans fil (OAT), l’AppKey est utilisée pour dériver les clés de session NwkSKey et AppSKey spécifique à cet appareil final pour chiffrer et vérifier les communications réseau et les données d’application.

L’AppKey doit être unique. L’utilisateur peut dériver sa propre AppKey.

# **7 Application**

1. Accès :

Chirpstack: <http://172.16.0.90:8080/> admin:admin

kerlink : <http://172.16.0.24/> login:admin:pwd4admin

Kerlink/ssh : ssh [root@172.16.0.24](mailto:root@172.16.0.24) password : pdmk-0901E0

1. Server Chirpstack
2. Passerelle KERLINK

Passerelle kerlink istation 868

poe 48v/140ma

dc 42-57/170ma

productid:237CKe0539C2

boardid: 235CHi0901E0

@mac: 70:76:FF:03:81:25

EUI: 7076FF00560901E0

Documentation : <https://wikikerlink.fr/>

Documentation : <https://wikikerlink.fr/wirnet-productline/doku.php> login: uca password: ASRy0iOwn3

Documentation : <https://assistance.sphinxfrance.fr/home/Kerlink>

Interface web : <http://172.16.0.24/>

1. Diagnostique du réseau

monit restart lorafwd #prise de la configuration

!!pour le moment il faut le stopper car le chirpstack gateway bridge est installer sur le serveur et pas dans la passerelle.

monit status chirpstack-gateway-bridge #l’état

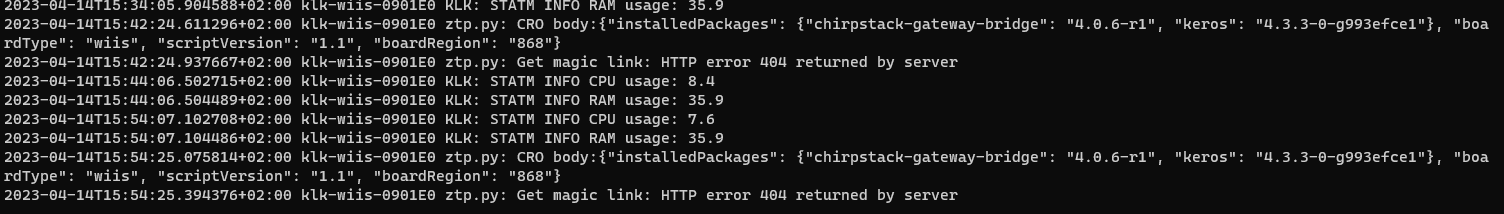
monit stop chirpstack-gateway-bridge

mais /etc/init.d/chirpstack-gateway-bridge stop #ne stoppe pas tout à fait le service

pour visualiser les logs

>tail -f /var/log/lora.log #visualise en continu [more|less]

> less /var/log/user.log



Savoir la connexion autant sur la passerelle que serveur

>netstat -antu #les ip et port

>netstat -atu

>netstat -antu | grep 1700

Remettre à l’heure de la passerelle

>date #remettre à l’heure la passerelle

/etc/init.d/ntpd stop

>ntpdate 193.54.52.113 #server de temps opgc

/etc/init.d/ntpd start

Côté serveur faire tout passer pour le firewall

Accepter tout sur iptable.

iptables -n -L

iptables -X

iptables -F

Visualiser les connexions en temps réelle. (capture)

>tcpdump -n -i eno1 host 172.16.0.91 #voir les connexion du client

>tcpdump -n -i eno1 #connexion de tout

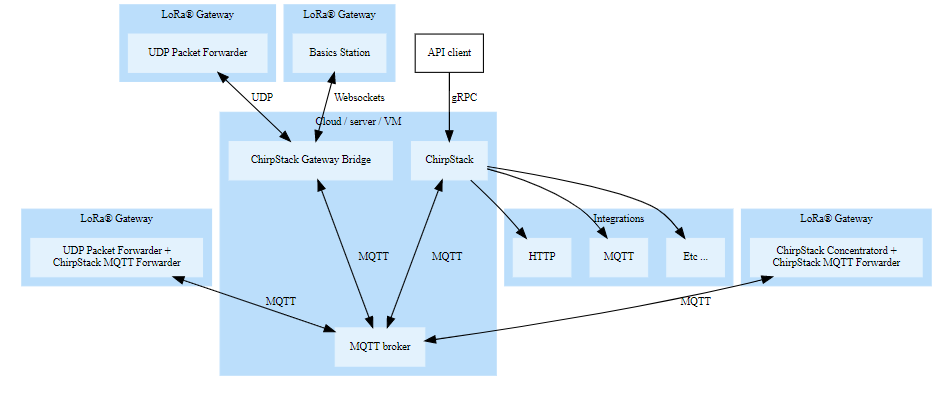
>tcpdump -n -i eno1 port 1700

Visualiser les topic mqtt su server

>/etc/chirpstack-gateway-bridge#

>mosquito\_sub -h localhost -t ‘#’ -F %j

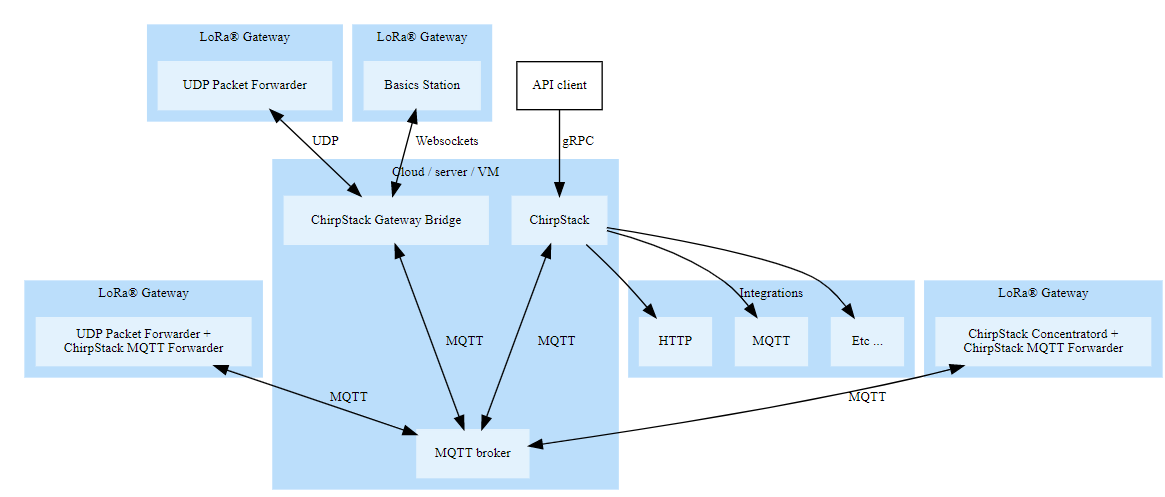
# **8 Chirpstack et récupérations des données**



**Conception de Connecs**

**2023 : Construction d’un serveur autour de l’architecture Chirpstack**

Architecture possible sur chirpstack afin de récupérer les données lora



1

Client 1 (Node JS)

API Grpc

Conception de client-serveur

2

Serveur Architecture libre

Postgres

Grafana

InfluxDB

Python

2

Serveur Chirpstack

ChirpStack

Broker MQTT

Chirpstack-Gateway-Bridge

OpenVPN

Site Extérieur

3

Passerelle

EUI: 7076FF00560901E0

Installation :

Chirpstack Gateway Bridge + VPN

Configuration connexion au broker MQTT

Node 1268 Envoi des données crypté

"devEUI": "434E535306E31268" #unique au noeud

"appEUI": "A78F1729918331B4" #tous

"appKey": "436f6e6e656353656e5330315a415455" #tous

2) Décodage des données

Python

Fichier JSON ./data/site/data20230427.json

Broker MQTT

Payloadcodecs.cnssrf.frame module oayloadcodecs

Mqttjsonflatterner.mqttjsonflattener module mqttjsonflattener

Inspire :

<https://www.emqx.com/en/blog/how-to-use-mqtt-in-python>

<https://munier.perso.univ-pau.fr/tutorial/iot/2022/20220510-mqtt_python/>

<https://sidik.my/lorawan-chirpstack-local/>

python :

<https://docs.python.org/fr/3/installing/index.html>

Utilisation de Mosquitto broker, le courtier le plus populaire, open source et facile à utiliser, écrit en C

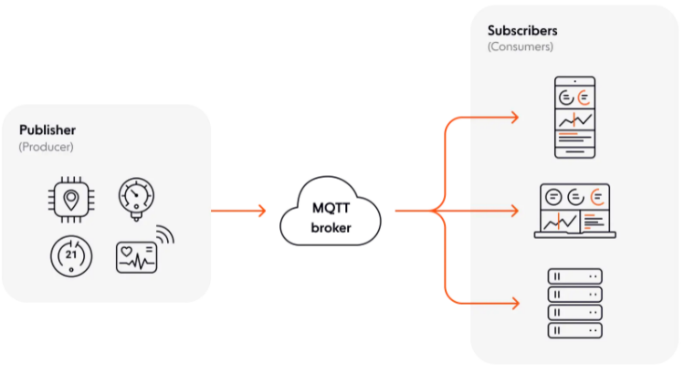
**ANNEXES**

**MQTT**

Protocol

**MQTT (Message Queuing Telemetry Transport)**

Protocol de communication léger utilisant le modèle de messagerie de publication/abonnement pour une communication de machine à machine sur des réseaux avec une bande passante limitée ou une connectivité imprévisible. Idéal pour les appareils de petite taille et limités avec une CPU, une mémoire et une autonomie de batterie limitées.



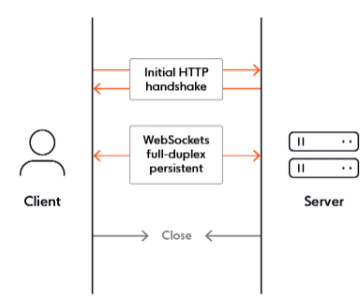
Semblable à WebSockets, MQTT s’exécute sur TCP contrairement aux WebSockets. MQTT fournit des flux de données un à un, un à plusieurs

Editeur (producteurs) et abonnés (consommateur)

Courtier (brocker) comme un serveur middleware MQTT, les messages sont stockés dans des rubriques (canaux).

**WebSocket**

Protocol en temps réel qui fournit un canal de communication en duplex intégral entre un client Web et un serveur Web via une seule connexion TCP. Idéal pour des systèmes nécessitant des mises à jour en temps réel ou des scénarios où la communication client et serveur est bidirectionnelle et n’appartient pas au schéma simple ‘demande-réponse’ (tickers financier, jeux multi-joueurs en temps réel, application de chat et mise à jour de géolocalisation en temps réel.



Le client envoie un GET et initie une poignée de main requête/réponse http client/serveur. Il met à niveau la connexion de http vers WebSockets et le serveur renvoie un http 101 Switching Protocols code de réponse pour que la connexion WebSocket soit établie. Après ils peuvent échanger des messages ensuite elle peut être interrompue via une poignée de main de fermeture.

Vous pouvez utiliser mqtt sur websocket pour consommer des messages MQTT dans un navigateur. Comme les deux protocoles sont bidirectionnels, vous pouvez envoyer des messages depuis des navigateurs Web (via WebSocket) au courtier MQTT qui peut ensuite les transmettre aux consommateur IoT.

**Comparaisons**

|  |  |  |
| --- | --- | --- |
| Critère | MQTT | WebSocket |
| Cas d’utilisation | MQTT convient aux appareils à faible puissance avec un processeur et mémoire limités et des réseaux peu fiables et faible bande passante. C’est le protocole incontournable pour les cas d’utilisation de l’IoT | Les WebSocket conviennent aux applications Web et aux appareils non limités par une faible bande passante. Convient aux cas d’utilisation tels que les téléscripteurs financiers, jeux multi-joueurs en ligne, le chat, les sondages en direct … |
| Architecture | MQTT utilise un modèle de messagerie de publication/abonnement | API bidirectionnelle de type socket. De nombreux frameworks et bibliothèque WebSocket ajoutent des fonctionnalités pub/sub en plus de WebSocket |
| Qualité de service de la messagerie | MQTT a trois options de QoS : 0 (au + 1 livraison), 1 (au moins 1 livraison) et 2 (exactement une fois) | Les WebSockets bruts ne fournissent pas d’options QoS |
| Protocole sous-jacent | TCP | TCP |
| Mode messagerie | Asynchrone, basé sur les évènements | Asynchrone, basé sur les évènements |
| Taille des messages | Maximum de 256Mo | 2^63 octets par trame |
| Connexions sécurisées | TLS | TLS |
| Frais généraux de message | Minimum de 2 octets | Minimum de 2 octets pour tous les messages à l’exception des trames masquées, qui ont un minimum de 6 octets. |
| File d’attente des messages | Les courtiers MQTT mettent en file d’attente les messages pour tous les clients déconnectés abonnés à une rubrique | Raw WebSocket ne prend pas en charge la mise en file d’attente des messages |
|  |  |  |

**Performances MQTT sur TCP et MQTT sur WebSocket**

MQTT sut TCP sont plus rapides cela est principalement dû à la surcharge de l’architecture MQTT sur WebSocket, cela est dû à l’échange initial d’en-tête plus complexes entre le client et le serveur et à la canalisation au déballage côté client et côté serveur.

**Les facteurs différenciant** MQTT et WebSocket de HTTP(Hypertext Transfert Protocol) est que MQTT et WebSocket sont des protocoles pilotés par les évènements, tandis que http est piloté par les requêtes.

Les WebSockets sont avec état et la connexion entre client/serveur reste active jusqu’à la fermeture de client ou du serveur.

Le MQTT est également avec état et fonctionne différemment de http, il utilise une communication pub/sub basée sur push au lieu de GET/POST dans http

Contrairement à WebSocket et MQTT, http est sans état et nécessite une nouvelle connexion à chaque requête (gourmand en ressource) dit aussi d’un protocole client/serveur asymétrique (requête et réponse)

**Utilisation**

MQTT pour les communications M2M pour les capteurs, application IoT à faible bande passante et latence élevée.

WebSocket communication bidirectionnelle en duplex intégral dans un environnement non contraint comme des applications de chat, jeux multi-joueurs temps réel.

MQTT sur WebSocket si vous souhaitez envoyer des messages MQTT directement à partir d’appareils et de capteurs à faible puissance vers des navigateurs Web.

Les brokers

Mosquitto broker, le courtier le plus populaire, open source et facile à utiliser, écrit en C

HiveMQ-CE plus intéressés par les grandes entreprise BMW, NetFlix, Audi …, il y a une version gratuite HiveMQ-CE disponible dans un conteneur docker mais manque d’options de personnalisation.

MQTT + Python

Import sys #fournite des fonctions et variables permettant d’interagir avec l’interpréteur Python.

Import json #encodage et décodage json

Import argparse #alanlyse arguments et commande génère des messages d’aide et d’utilisation et erreur argv

Import inspect #inspect les objets modules, classes, méthodes ou récupère le code source d’une méthode

Import os #une façon portable d’utiliser les fonctionnalités dépendantes du système d’exploitation.

Import subprocess #lande de nouveau processus, connecter e/s/er

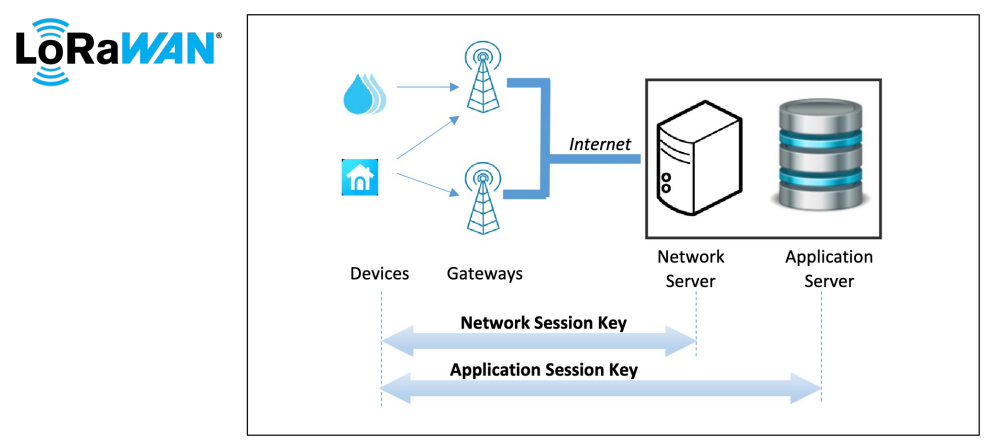
Import site

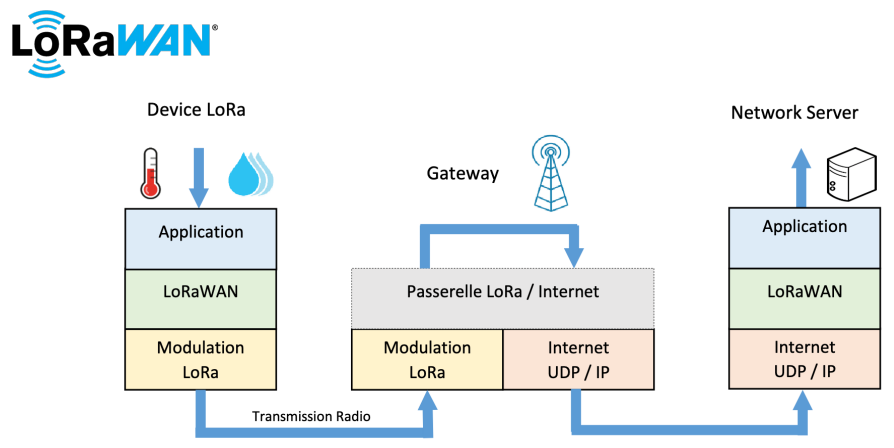
Import importlib

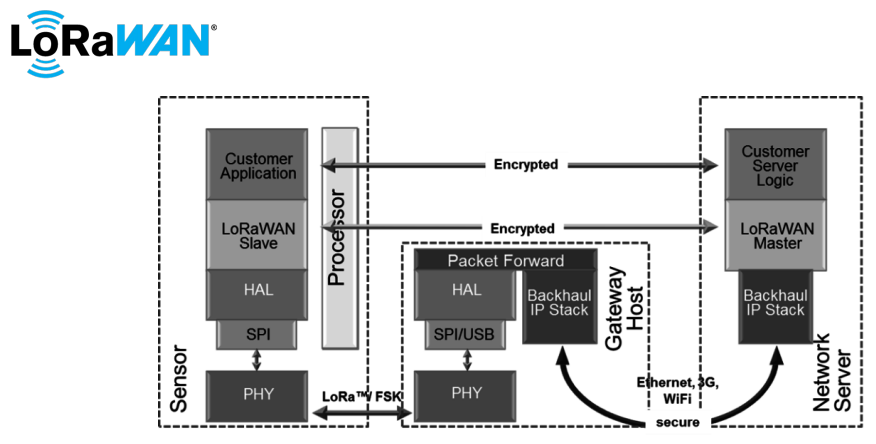
Frome importlib import reload #un module peut contenir des instructions exécutables généralement pour l’initialisation un module est importé donc exécuté une seul fois par session d’interprétation cela permet donc de l’exécuter une autre fois.

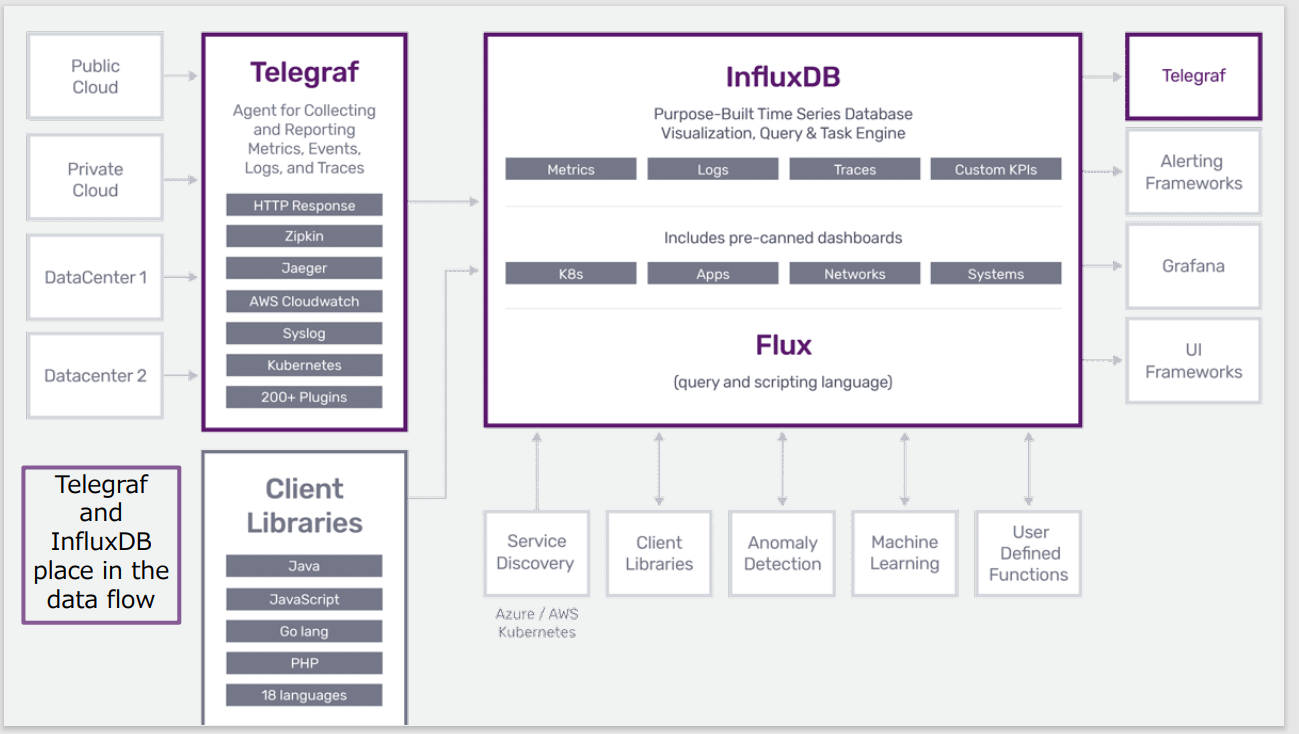
# **10 Annexes**

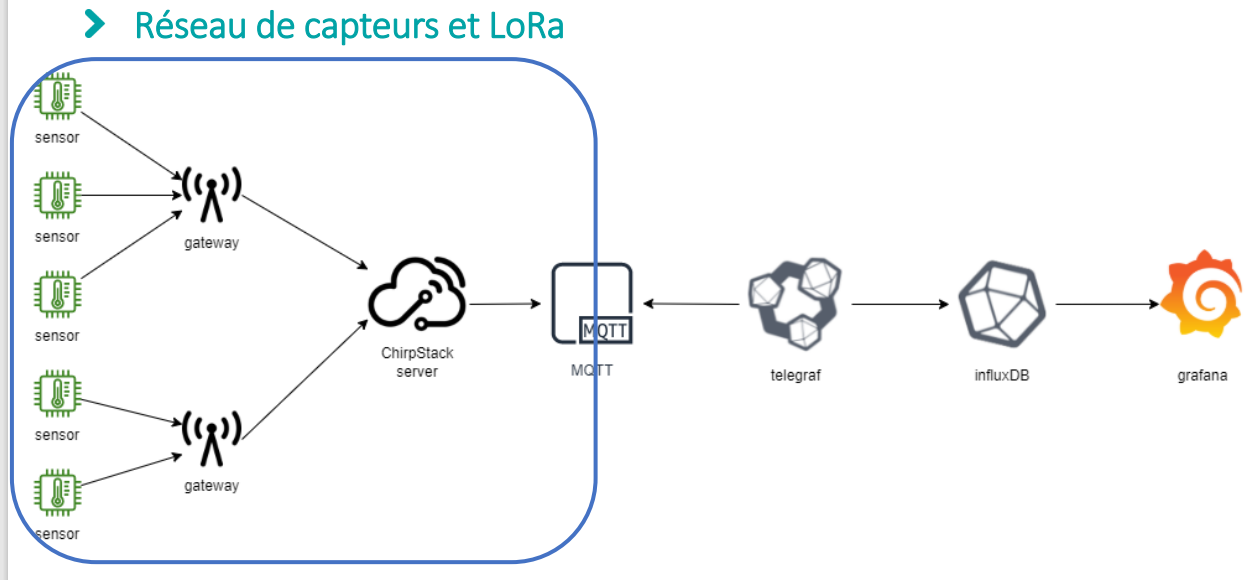
**ANNEXES**

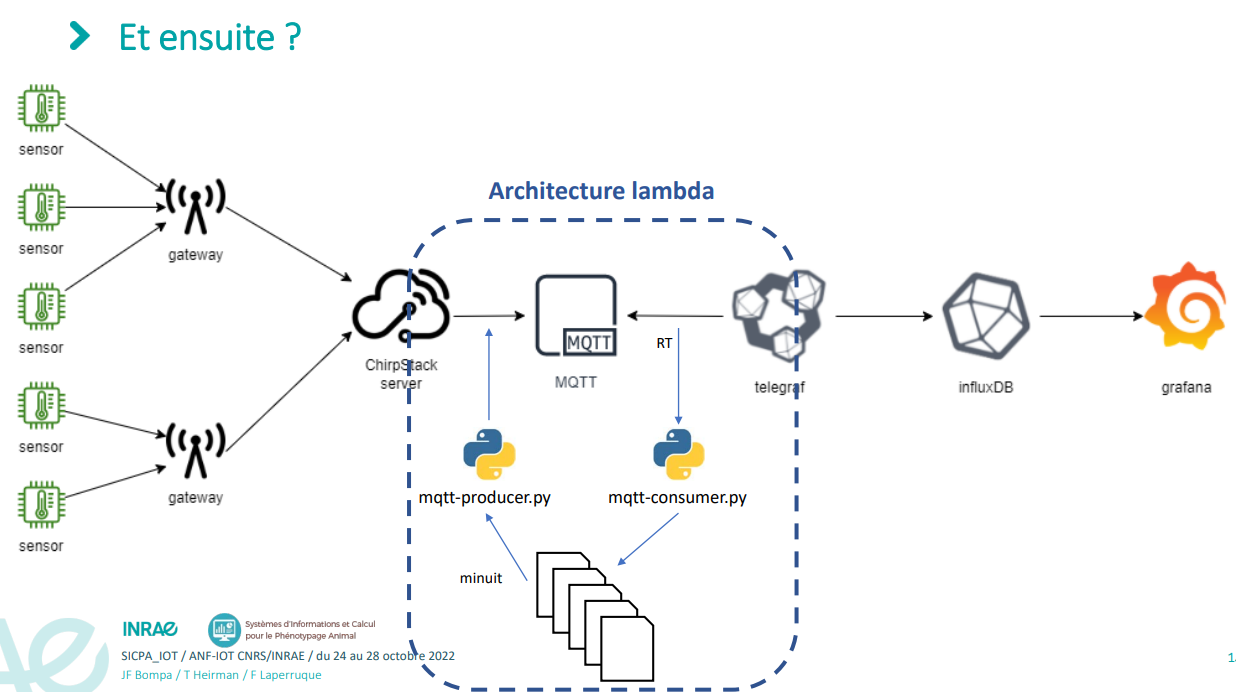


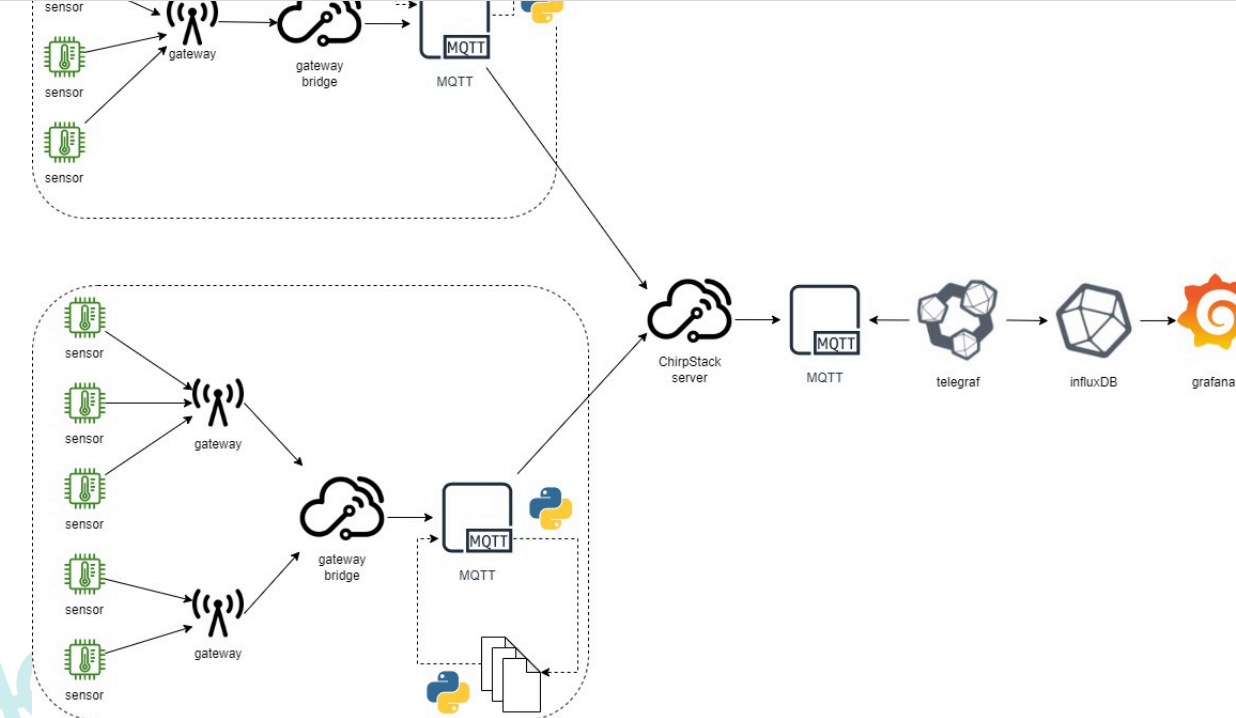


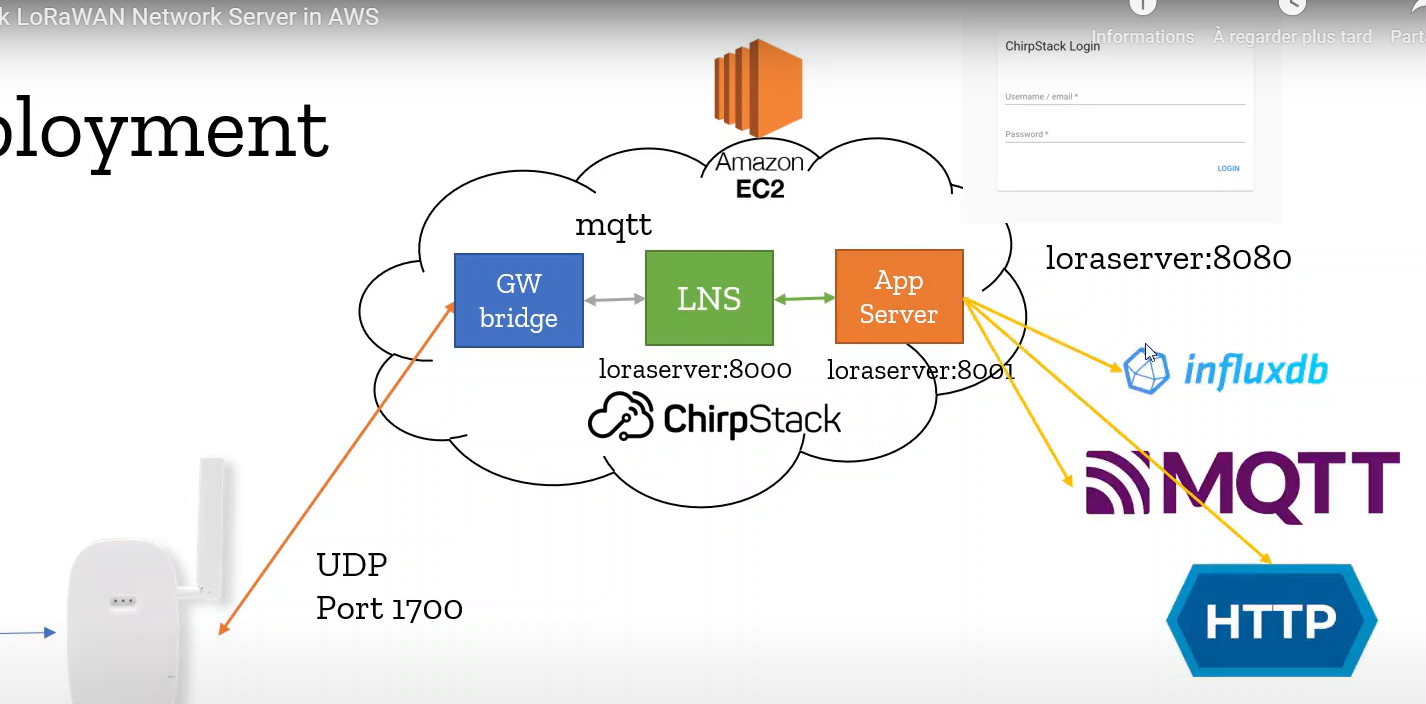






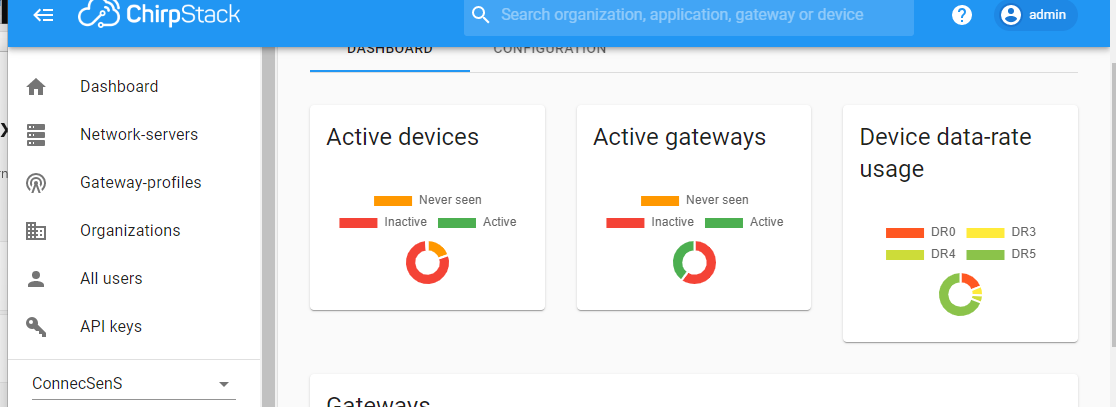


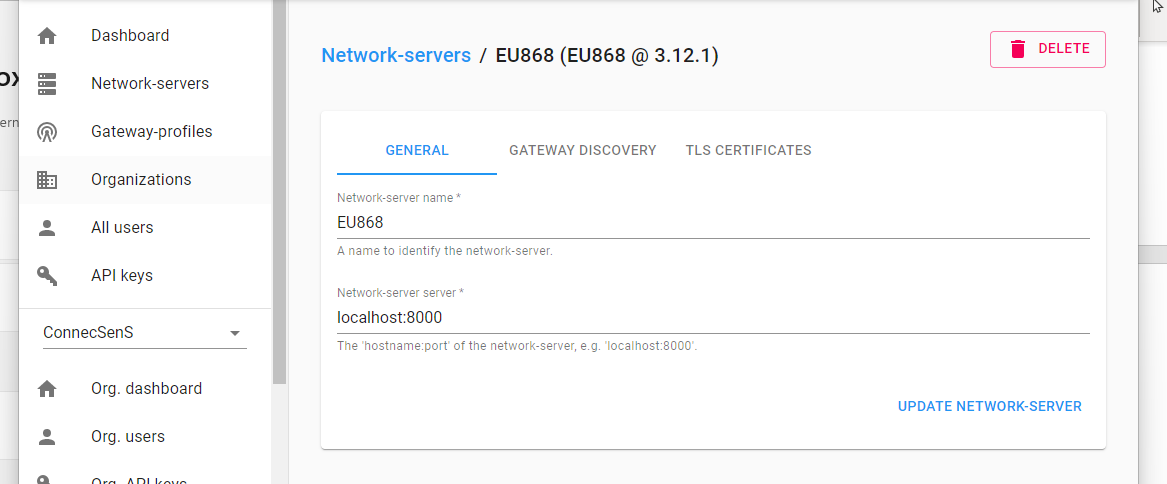


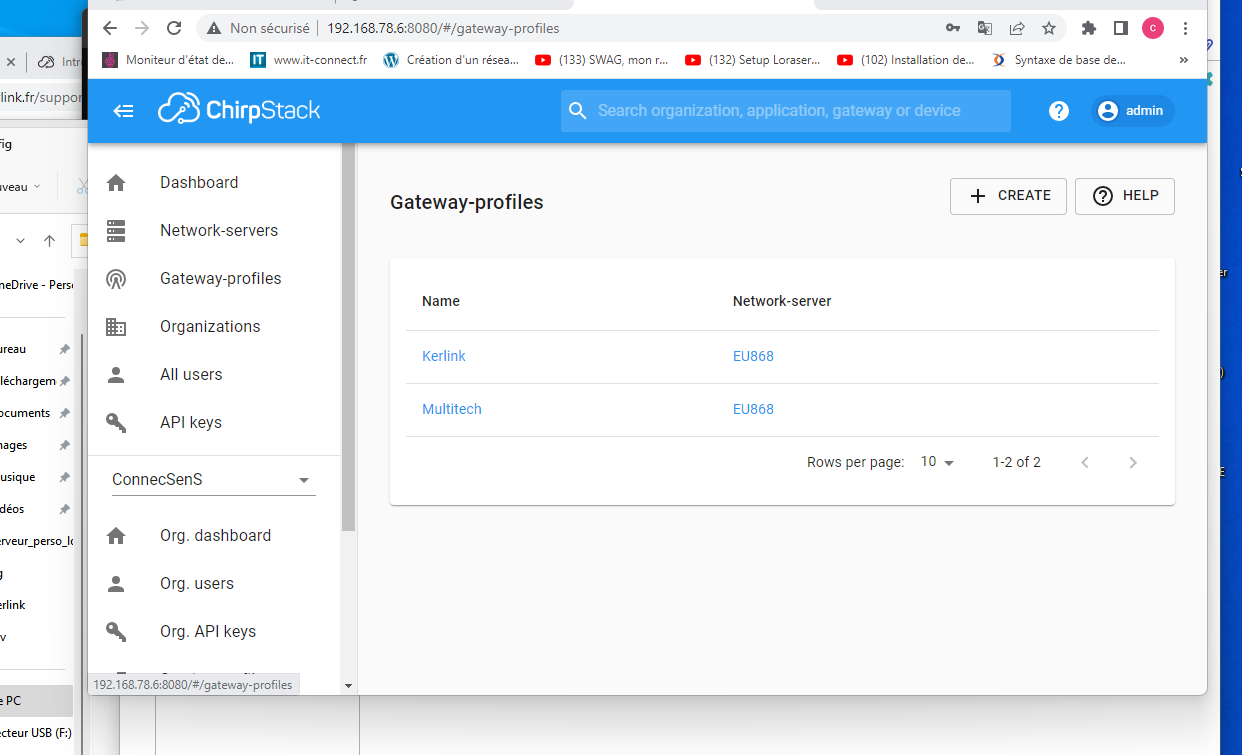


**Connecs Chirpstack**

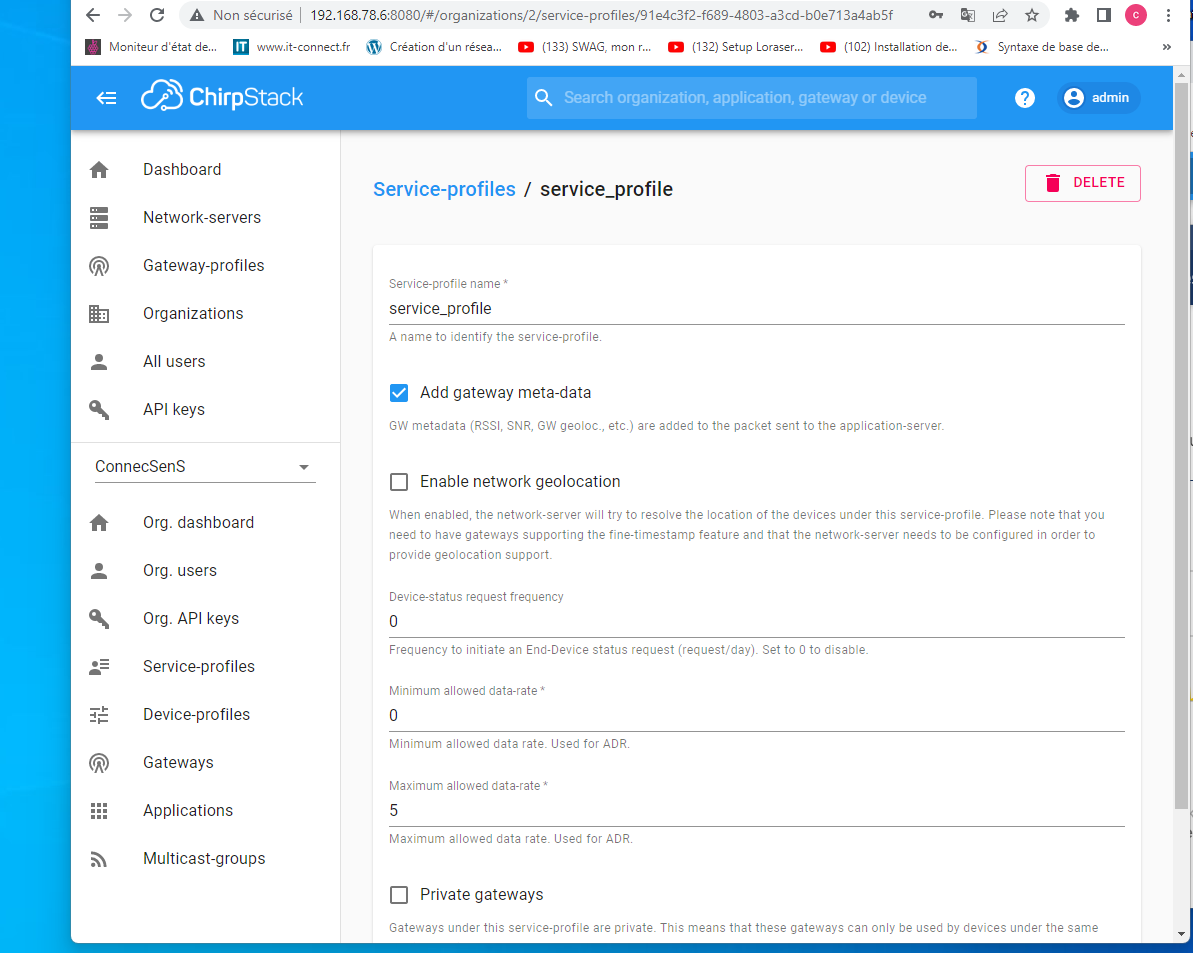
Chirpsatck connecsens

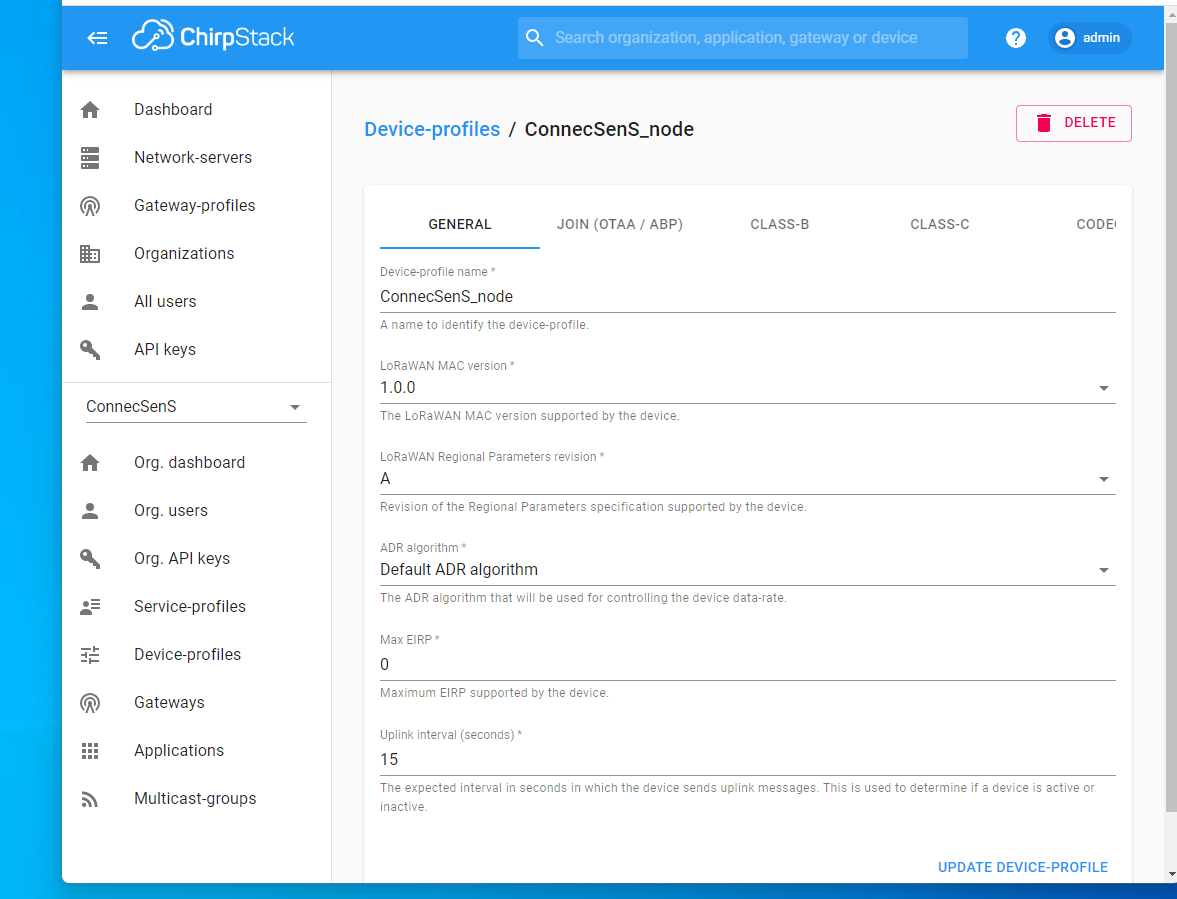


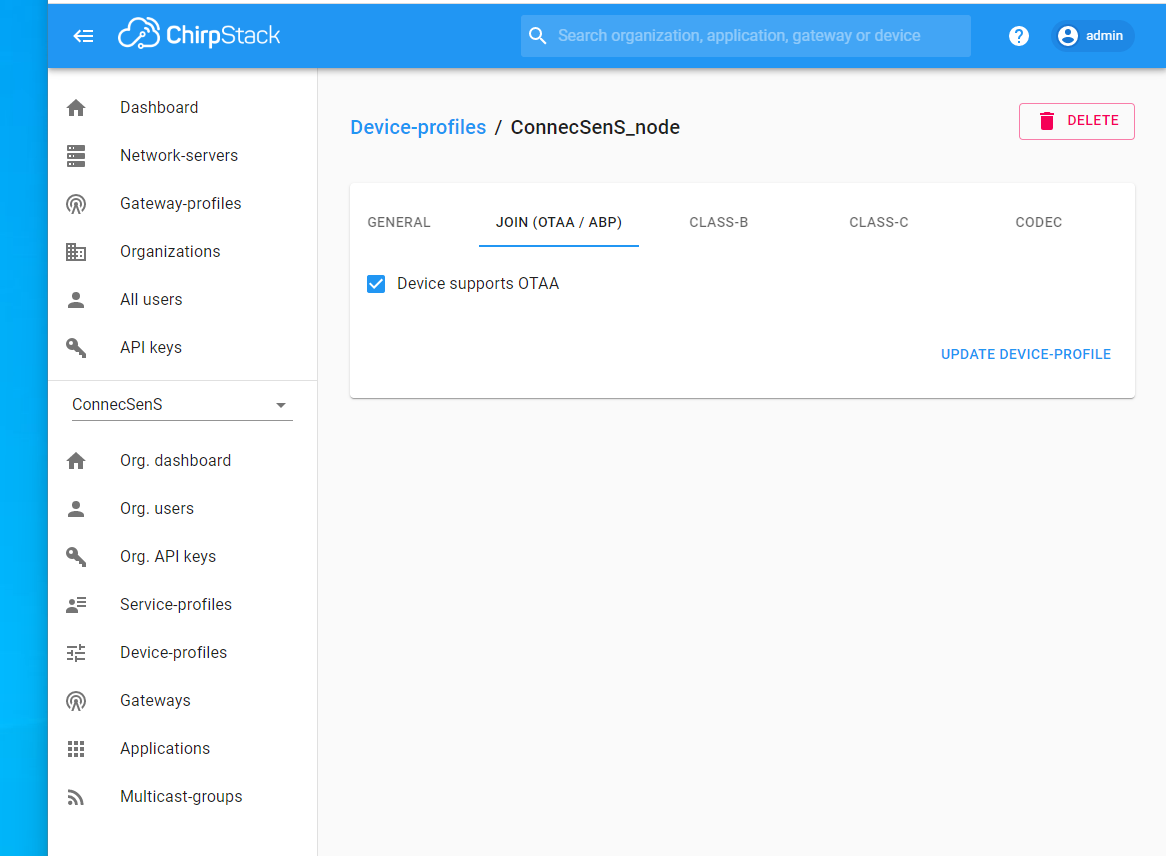


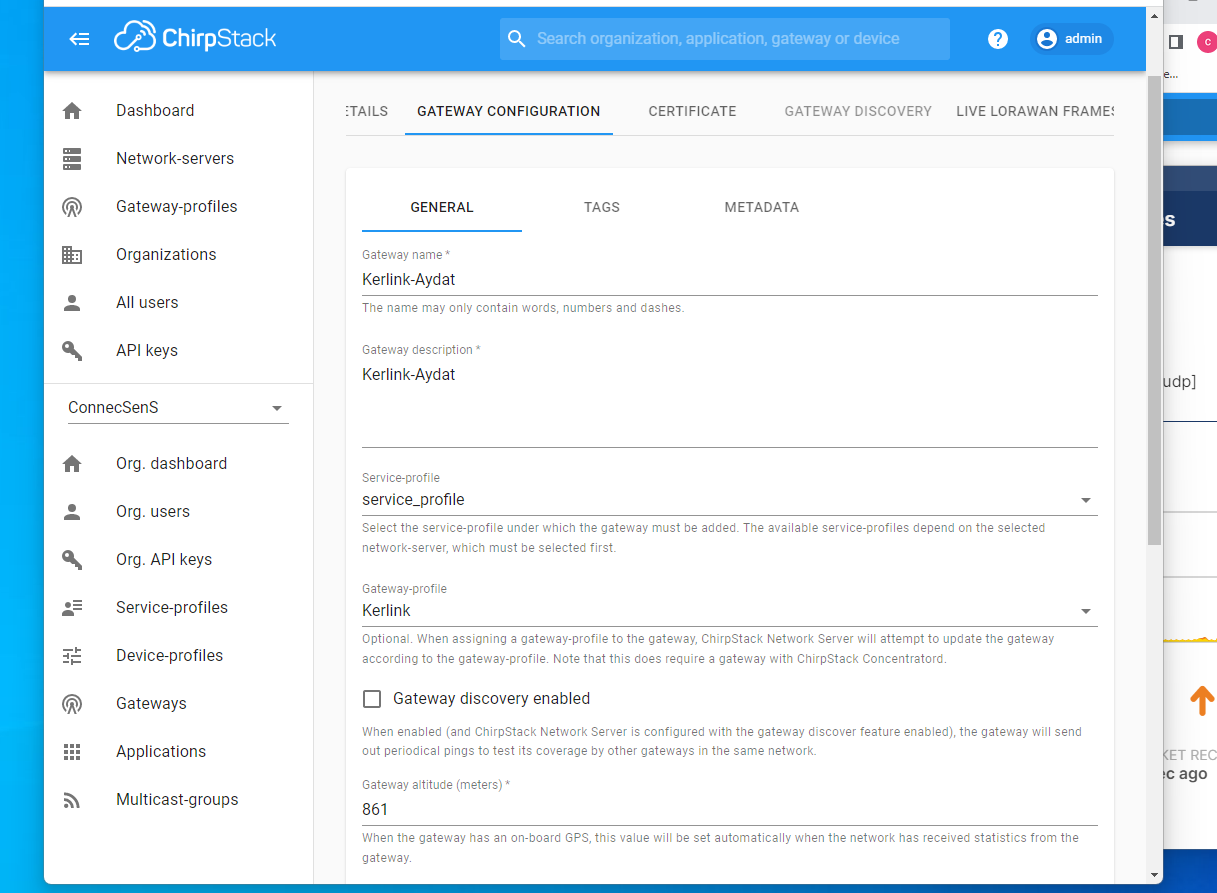


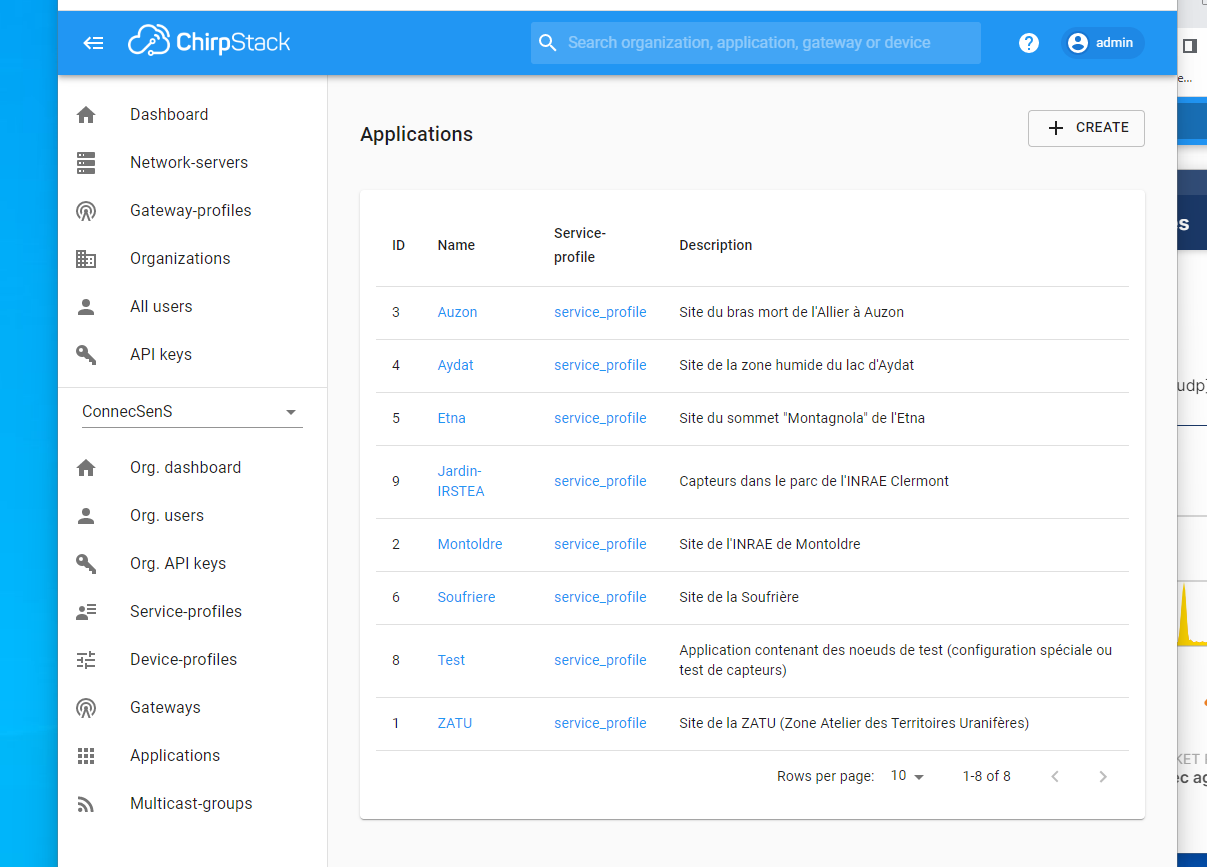


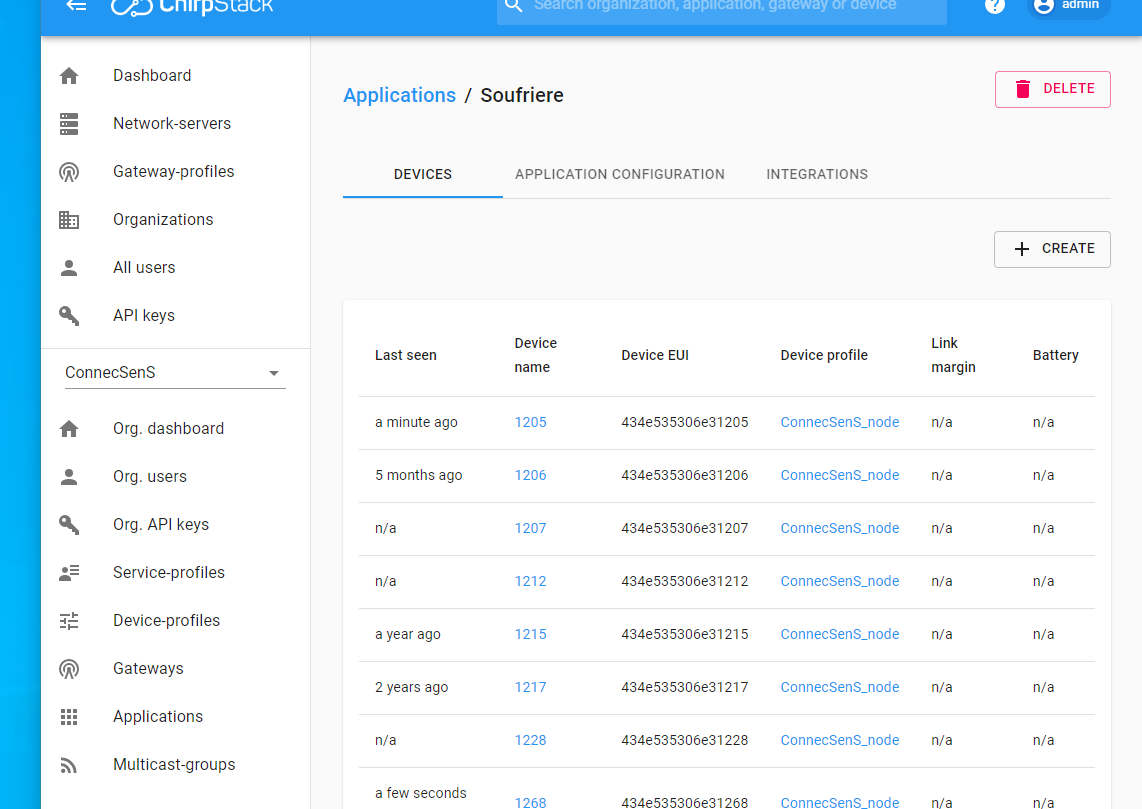


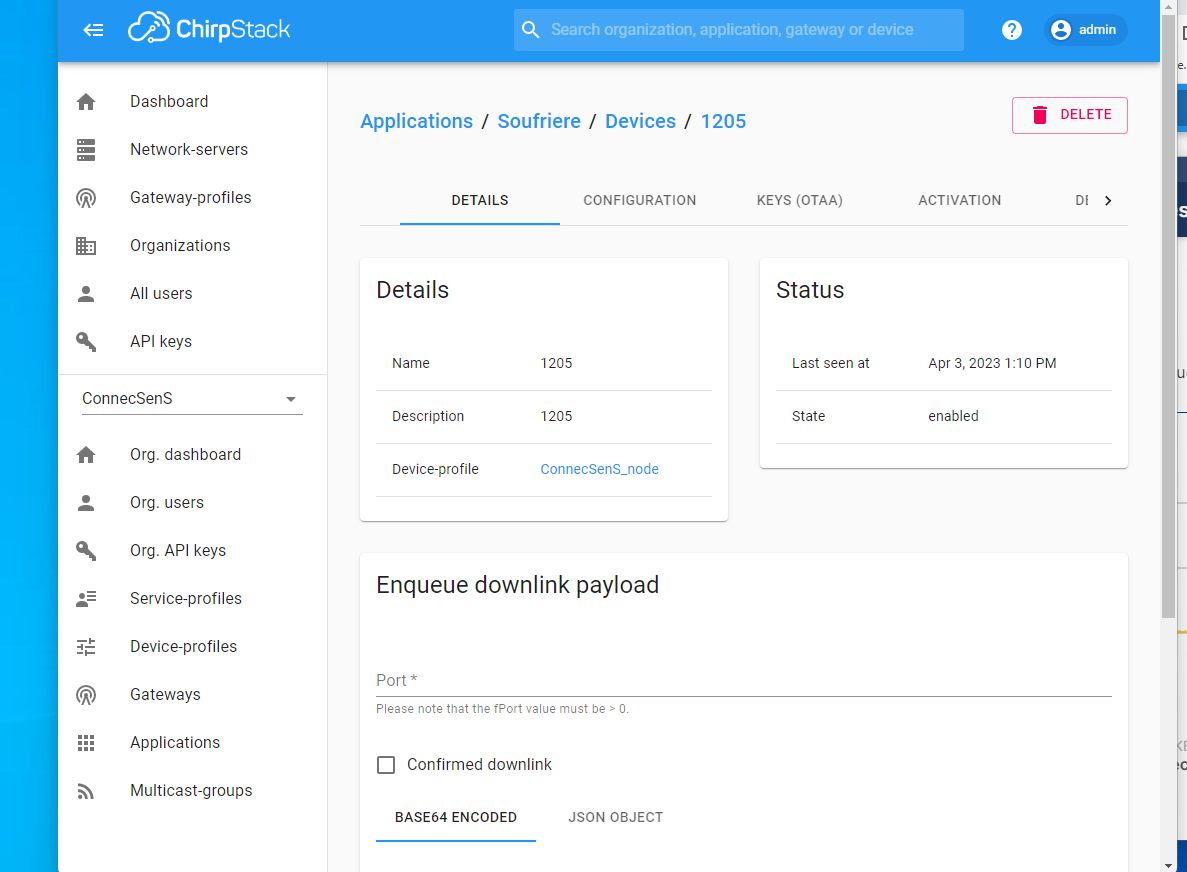


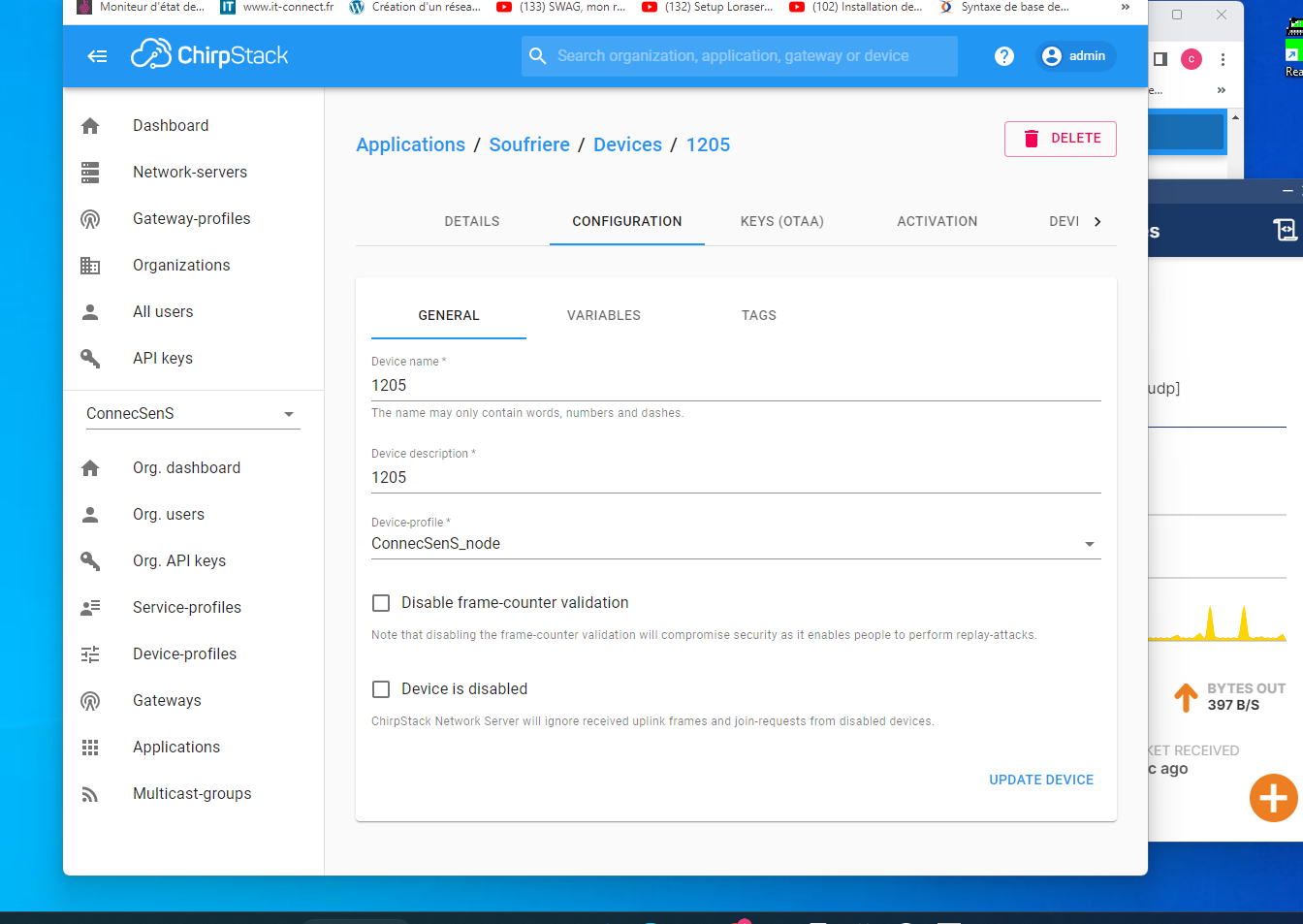


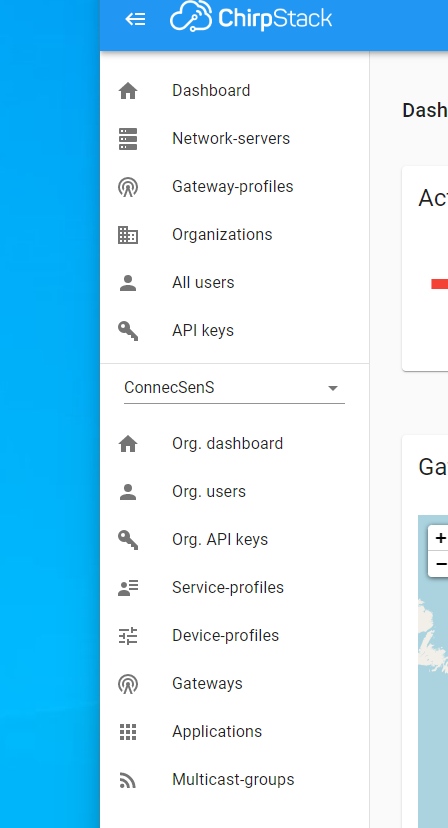












**Connecs Configuration des passerelles**

**Configuration des services façons connecs**

**A] Configuration de Packer Forwarder**

> iStation: vi /etc/lorad/wiis/EU868-FR.json

> iFemtocell: vi /etc/lorad/wifc/EU868-FR.json

> iFemtocell-Evolution: vi /etc/lorad/fevo/EU868-FR.json

Dans EU868-FR.json

Lorawan\_public : false

Copier la valeur du champ EUI (EUI64) > cat /tmp/board\_info.json #EUI64 : 7076FF00560901E0

> vi /etc/**loradfwd.toml**

Id = 0x7076FF00560901E0

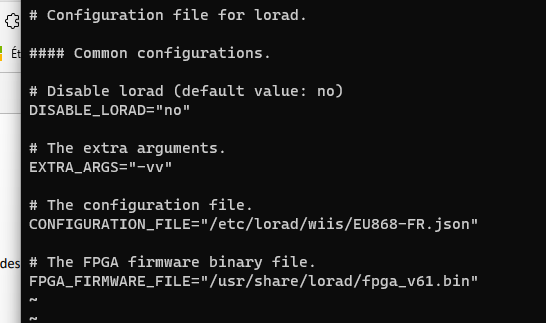
[gwmp]

node="127.0.0.1"

service.uplink=1700

service.downlink=1700

> Vi /etc/default/**lorad**



Origine : CONFIGURATION\_FILE= "configurer le chemin de la bonne fréquence du pays " + Disable\_lorad = no

**B] Installation et configuration du Gateway Bridge**

Dans la doc connecs

Copier depuis le pc avec mobaxterm scp le chirpstack-gateway-bridge\_3.10.0-r1\_klkgw.ipk

> mkdir -p /user/.updates

> mv chirpstack-gateway-bridge\_3.10.0-r1\_klkgw.ipk /user/.updates

> sync

> kerosd -u

> reboot

Dans la doc chirpstack :

Trouvez le dernier package sur <https://artifacts.chirpstack.io/vendor/kerlink/keros-gws/>

> cd /user/.updates

> wget https://artifacts.chirpstack.io/vendor/kerlink/keros-gws/chirpstack-gateway-bridge\_4.0.6-r1\_klkgw.ipk

Pour déclencher la passerelle pour installer/mettre à jour le package exécuter les commandes :

> sync

> kerosd -u

> reboot

La passerelle va vérifier le contenu du dossier /user/.updates et installer le package qu’il contient.

**C] après le reboot passerelle**

> vi /etc/ chirpstack-gateway-bridge/**chirpstack-gateway-bridge.toml**

Section [integration.mqtt.auth.generic]

Server= " tcp://172.16.0.90 :1883" #l’adresse du server tcp

Section[backend.semtech\_udp]

udp\_bind = " 0.0.0.0 :1700" #vous pouvez changer de port exemple : 2000

***C] Supplément utilisation d’un VPN sur la passerelle avec l’utilisation d’une carte sim.***

Le VPN installé par défaut sur la passerelle ne fonctionne pas avec notre configuration.

Il est donc nécessaire de le remplacer par un OpenVPN compatible n’utilisant pas libressl.

– Si cela n’a pas déjà été fait, copiez sur la passerelle le contenu du dossier « OpenVPN » approprié au modèle de votre passerelle depuis votre PC sur cette dernière.

1 – Tapez les commandes suivantes :

cd ~

rm /usr/sbin/openvpn

chmod +x OpenVPN/openvpn

mv OpenVPN/openvpn /usr/sbin/

tar -xzvf OpenVPN/openvpn.tar.gz

rm /etc/openvpn/bscc.conf

mv /openvpn/\* /etc/openvpn/

rm /etc/init.d/openvpn

mv OpenVPN/etcinitd/openvpn /etc/init.d/

chmod +x /etc/init.d/openvpn

update-rc.d openvpn default

2 – Tapez la commande suivante : vi/etc/firewall.d/00\_default.rules

A la fin du fichier avant COMMIT ajoutez

#VPN

-I OUTPUT 1 -p tcp --dport 1194 -j ACCEPT

-I INPUT 1 -p tcp -s 193.55.252.23 -j ACCEPT

Après l’enregistrement:

3 – dans /etc/init.d/openvpn

Enlever individuellement tous les « ^M » du fichier

ATTENTION : Il faudra vérifier tous les fichiers copiés depuis le pc sur la passerelle, ils sont tous susceptible d’avoir comme caractère à la fin de chaque ligne « ^M ». J’ai également remarqué qu’un début d’entête commentée avait été supprimée en partie dans le fichier « Network\_functions ». Le fichier commençait alors par un début de phrase incompréhensible par la passerelle, ce qui engendrait un problème de lecture du fichier en question, qui est appelé par un autre fichier. Be careful ! Commande linux pour chercher et remplacer une chaine de caractères dans un fichier : Dos2unix “chemin/fichier”

4 – Vérification de la fonctionnalité de la passerelle

0 – Si ce n’est pas déjà fait, loggez-vous sur la passerelle en SSH.

1 – Tapez la commande « ifconfig ». Le résultat doit être le suivant.

Eth0 : Interface Ethernet, présente par défaut sur les passerelles, non utilisée ici.

Lo : Interface locale, présente par défaut sur les passerelles, non utilisée ici.

Usb0 / Usb1 : Interfaces USB, utilisées avec la sonde de debug Kerlink, non utilisée ici

Wwan0 : Interface GSM, sa présence indique que la passerelle a réussi à se connecter au réseau de l’opérateur.

Tun0 : Interface VPN, sa présence indique que la passerelle a réussi à rentrer sur le réseau du mésocentre.

5 – Monitoring des services

La commande "monit" est installé par défaut sur la passerelle et gère tous les services à l’exception d’openvpn.

Vous pouvez consulter l’état d’un service en tapant la commande :

> monit status [nom\_du\_service] (lorad, loradfw, chirpstack-gateway-bridge) pour les plus courant.

(status, stop, start, restart)

Pour ajouter le monitoring d’openvpn

1. Dans le fichier /etc/monitrc vérifier la ligne à la fin sous includes : include /etc/monit.d/\*
2. Dans le répertoire /etc/monit.d, rajouter un fichier nommé openvpn contenant ces qqs lignes :

CHECK PROCESS openvpn PIDFILE /var/run/openvpn.client-openvpn.pid

start program = "/etc/init.d/openvpn start"

stop program = "/etc/init.d/openvpn stop"

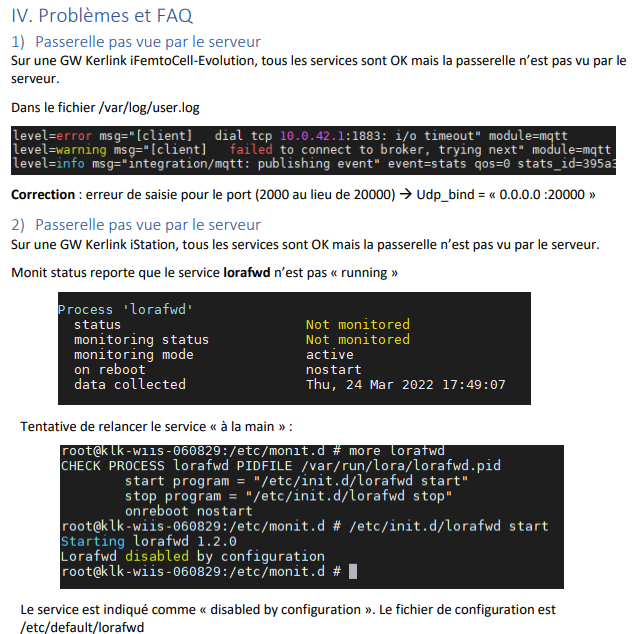
monit venant executer les scripts dans le répertoire /etc/monit.d, ce nouveau script sera pris en compte automatiquement après un reboot ou la commande monit restart. Vérifier qu’il est bien actif avec monit status openvpn.

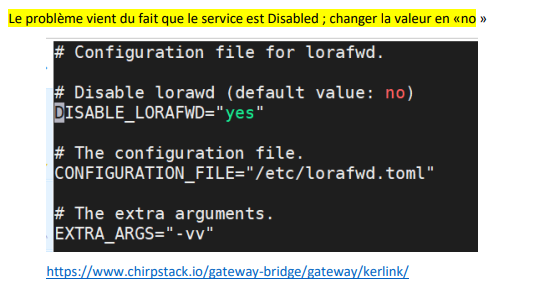
6 – Localisation des logs

Contrairement aux autres modèles de passerelle, les Kerlink iStation / iFemtocell / iFemtocell-Evolution utilisent le SSH comme connexion de debug. Les logs des services qui en fournissent (openvpn et chirpstackgateway-bridge) apparaitront donc sur la sortie Standard, sans besoin de consulter un fichier spécifique.

Le reste des logs est situé dans le fichier /var/log/message (pour GSM notamment)

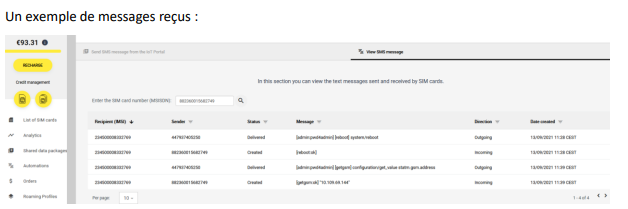
**Connecs Problèmos**

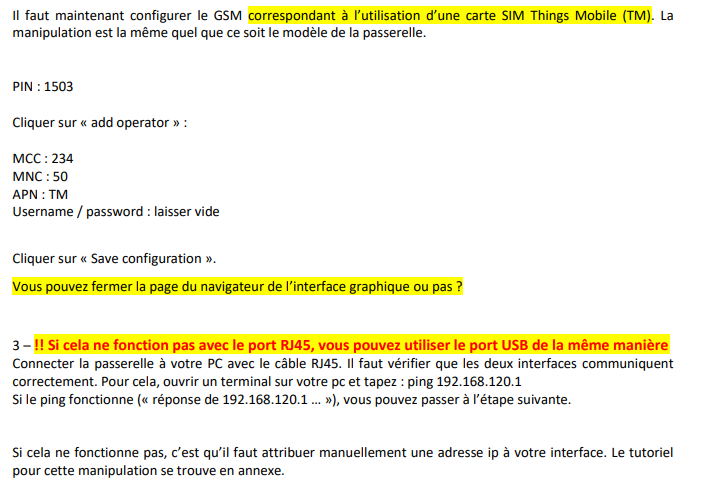












**Connecs Architecture**

MQTT/chirpStack

Chripstack/chirpsatck

VPN: 193.55.252.23 :11794

Passerelle

LoRa/4G

Gateway/

Bridge/

Chirpstack



Serveur-mqtt : tcp://172.16.0.90 :1883

PostgreSQL

Serveur Join

c

Afficher l’image source

Service web

Python

Grafana

user

ElasticSearch

Nœud

ConnecSenS

Fichier json

Loraserver : 192.168.78.6

/home/loraserver/data/fichier json