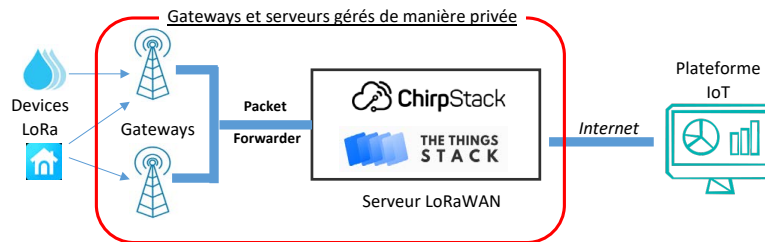


TP 2 SERVEUR LORAWAN PRIVÉ CHIRPSTACK

Objectifs :

- Installer et configurer un serveur Lorawan privé.
- Configurer la passerelle Lorawan.
- Enregistrer un device sur le serveur privé.

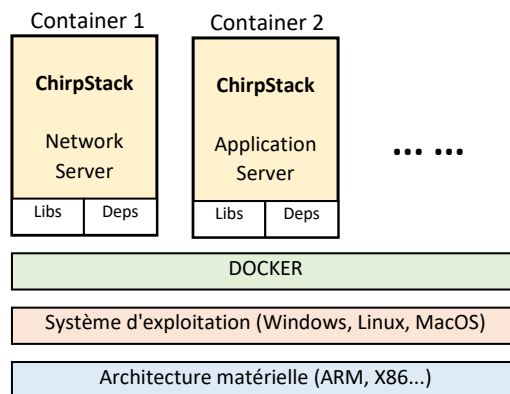


1. INSTALLATION DU SERVEUR CHIRPSTACK

1.1. Clonez le projet Chirpstack docker depuis son github

```
git clone https://github.com/chirpstack/chirpstack-docker.git
cd chirpstack-docker
```

1.2. Installation du projet avec docker et docker-compose



1.2.1. Saisissez la commande suivante qui permet la création des différents conteneurs constituant le serveur Lorawan.

```
docker-compose up
```

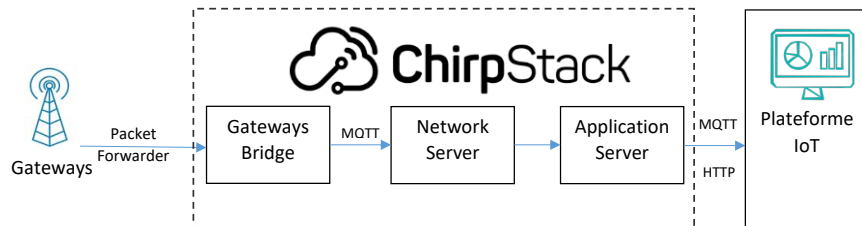
1.2.2. Saisissez dans un autre onglet du terminal la commande suivante pour vérifier les conteneurs créés :

```
docker ps
```

Parmi les conteneurs créés, Nous pouvons remarquer que le service Gateway Bridge écoute sur le port 1700/udp parce que nous utilisons le "Semtech UDP Packet forwarder". C'est le port sur lequel notre Gateway doit envoyer ses données. L'Application Server écoute sur le port 8080/tcp. C'est l'accès à l'interface web pour configurer le serveur LoRaWAN.

NOMS PORTS

```
chirpstack_chirpstack-application-server_1 0.0.0.0:8080->8080/tcp
chirpstack_chirpstack-network-server_1
chirpstack_chirpstack-gateway-bridge_1 0.0.0.0:1700->1700/udp
```

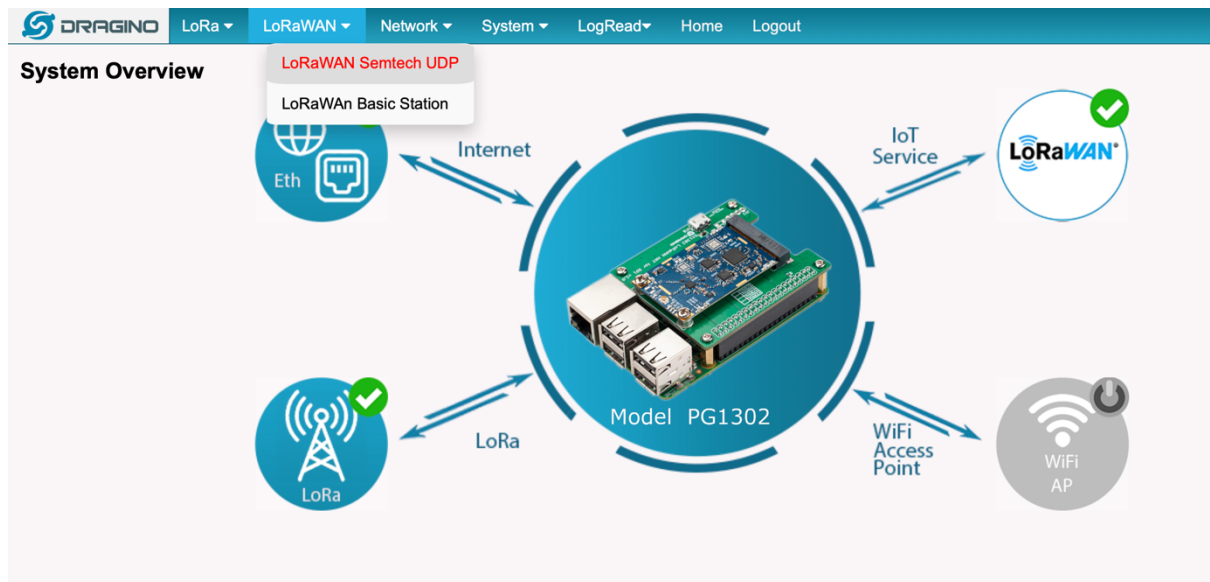


1.3. Importez les profils des principaux « device » depuis le répertoire chirpstack-docker


make import-lorawan-devices

2. CONFIGURATION DE LA PASSERELLE

2.1. Connectez-vous sur votre passerelle (gatewayX.local avec identifiants de connexion root/dragino). X dépend du raspberry utilisé.



2.2. Indiquez l'adresse IP de votre serveur Chirpstack et le port 1700 dans l'onglet LoRaWan/LoRaWan Semtech UDP de la passerelle.


LoRa ▼ LoRaWAN ▼ Network ▼ System ▼ LogRead▼ Home Logout


LoRaWAN Configuration

General Settings

Email

Gateway ID

Primary LoRaWAN Server

Service Provider 

Uplink Port

Server Address

Downlink Port

Packet Filter

Fport Filter ?

DevAddr Filter ?

Current Mode: **LoRaWAN Semtech UDP**

3. CONFIGURATION DE CHIRPSTACK


3.1. Configurez ChirpStack à partir de son interface web graphique disponible via le port 8080/tcp : <http://@IP-ChirpStack:8080>

Les noms d'utilisateur et les mots de passe par défaut sont les suivants :

- Nom d'utilisateur : admin
- Mot de passe : admin

ChirpStack login

* Username / email :

* Password : 

3.2. Ajoutez les informations de la gateway (Add gateway), donnez lui un nom et complétez l'EUI gateway (on le trouve sur l'interface de la gateway)

The screenshot shows the 'Add gateway' form in the ChirpStack interface. The left sidebar contains a menu with 'Network Server' and 'Tenant' sections. The 'Tenant' section is expanded, showing 'Dashboard', 'Users', 'API Keys', 'Device Profile Templates', 'Regions', 'Gateways' (highlighted), and 'Applications'. The main content area is titled 'Add gateway' and has tabs for 'General', 'Tags', and 'Metadata'. The 'General' tab is active. It contains the following fields: 'Name' (filled with 'Gateway TP1'), 'Description' (filled with 'Gateway TP'), 'Gateway ID (EUI64)' (filled with 'C0EE40FFFF2A3FED'), 'Stats interval (secs)' (set to '30'), and a 'Location' section with a map placeholder. The breadcrumb trail at the top reads 'Tenants / ChirpStack / Gateways / Add'.

4. CRÉATION DE NOTRE PREMIÈRE APPLICATION

4.1. Créez une application :

The screenshot shows the 'Add application' form in the ChirpStack interface. The left sidebar is identical to the previous screenshot, with 'Applications' highlighted in the 'Tenant' section. The main content area is titled 'Add application' and has tabs for 'General' and 'Tags'. The 'General' tab is active. It contains the following fields: 'Name' (filled with 'Station météo'), 'Description' (filled with 'station météo du lycée'), and a 'Submit' button. The breadcrumb trail at the top reads 'Tenants / ChirpStack / Applications / Add'.

4.2. Créez un device profiles

On importe un template (on choisit ici un Arduino MKR1310 en fréquence européenne)

The screenshot shows the ChirpStack interface with the 'Select device-profile template' dialog box open. The dialog lists various templates, including 'Arduino SA', 'Aqua-Scope Technologies', 'Arduino MKR WAN 1310', 'B Meters SRL', 'Baylan Water Meters', 'Beijing COTX Networks Technologies Co. Ltd.', and 'Reisaten Service GmbH'. The 'Arduino MKR WAN 1310' template is selected, showing its details: 'FW version: 1.2.0' and 'AU915'. The background shows the 'General' tab of the device profile configuration form, with fields for Name, Description, Region (EU868), MAC version (LoRaWAN 1.0.3), ADR algorithm (Default ADR algorithm (LoRa only)), and Flush queue on activate (checked).

The screenshot shows the ChirpStack interface with the 'General' tab of the 'Arduino MKR WAN 1310' device profile configuration form. The 'Region configuration' field is highlighted, showing 'EU868' and 'A'. A text box indicates that the region configuration should be completed and validated. The form also includes fields for Name, Description, MAC version, ADR algorithm, and Flush queue on activate. The 'Join (OTAA / ABP)' tab is also visible, showing 'Device supports OTAA' as checked.

The screenshot shows the ChirpStack interface with the 'Device profiles' table. The table lists the 'Arduino MKR WAN 1310' profile with its configuration details. The table has columns for Name, Region, MAC version, Revision, Supports OTAA, Supports Class-B, and Supports Class-C. The 'Arduino MKR WAN 1310' profile is listed with Region EU868, MAC version LoRaWAN 1.0.2, Revision A, Supports OTAA yes, Supports Class-B no, and Supports Class-C no.

Name	Region	MAC version	Revision	Supports OTAA	Supports Class-B	Supports Class-C
Arduino MKR WAN 1310	EU868	LoRaWAN 1.0.2	A	yes	no	no

Résultat du profil créé après validation

5. CRÉATION DU DEVICE

5.1. Importation du DEV EUI.

- 5.1.1. Connecter la carte Arduino MKR 1310 et lancer l'IDE Arduino.
- 5.1.2. Vérifier que la carte est reconnue par l'IDE (outils/board).
- 5.1.3. Vérifier que les ports.
- 5.1.4. Charger le sketch fichier/examples/MKRWAN/FirstConfiguration.
- 5.1.5. Téléverser le programme dans la carte et copier/coller votre device EUI.

```
FirstConfiguration.ino
1 //
2
3 #include <MKRWAN.h>
4
5 LoRaModem modem;
6
7 // Uncomment if using the Murata chip as a module
8 // LoRaModem modem(Serial1);
9
10 String appEui;
11 String appKey;
12 String devAddr;
13 String mskKey;
14 String appSKey;
15
16 void setup() {
17   // put your setup code here, to run once:
18   Serial.begin(115200);
19   while (!Serial);
20   Serial.println("Welcome to MKR WAN 1300/1310 first configuration sketch");
21   Serial.println("Register to your favourite LoRa network and we are ready to go!");
22   // change this to your regional band (eg. US915, AS923, ...)
23   if (!modem.begin(EU868)) {
24     Serial.println("Failed to start module");
25     while (1) {}
26   };
27   Serial.print("Your module version is: ");
28 }
```

Output Serial Monitor x

Message (Enter to send message to 'Arduino MKR WAN 1310' on '/dev/cu.usbmodem14401')

New Line 9600 baud

Message (Enter to send message to 'Arduino MKR WAN 1310' on '/dev/cu.usbmodem14401')

***** PLEASE READ THIS: THE DEVICE SHOULD BE ADDED AS A DEVICE *****

To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.

Your device EUI is: a8610a35301e8909

Are you connecting via OTAA (1) or ABP (2)?

Enter your APP EUI

Enter your APP KEY

Something went wrong; are you indoor? Move near a window and retry

Welcome to MKR WAN 1300/1310 first configuration sketch

Register to your favourite LoRa network and we are ready to go!

Your module version is: ARD-078 1.2.3

Please make sure that the latest modem firmware is installed.

To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.

Your device EUI is: a8610a35301e8909

Are you connecting via OTAA (1) or ABP (2)?

Device EUI nécessaire pour la suite

5.2. Reconnectez-vous sur votre application Chirpsatck et créez un device.

ChirpStack

Tenants / ChirpStack / Applications / Station météo

Station météo application id: a24ec754-0d17-45dd-b416-61e35aacd628

Delete application

Devices Multicast groups Relays Application configuration Integrations

Add device Selected devices

Last seen	DevEUI	Name	Device profile	Battery
No data				

5.3 Donnez un nom au device, choisissez le device profile créer précédemment, recopiez le device EUI précédent et complétez le join EUI (App EUI avec la valeur 00 00 00 00 00 00 00 00).

Tenants / ChirpStack / Applications / Station météo / Add device

Add device

Device Tags Variables

* Name
MKR1310

Description
MKR1310 Temp et Hum

* Device EUI (EUI64)
a8610a3233428e03 MSB C

Join EUI (EUI64)
0000000000000000 MSB C

* Device profile
Arduino MKR WAN 1310

Device is disabled ☐

Disable frame-counter validation ☐

Submit

5.4 Après soumission du formulaire précédent, la fenêtre de création de l'application key apparaît, générez cette clé comme ci-dessous :

Tenants / ChirpStack / Applications / Station météo / Devices / MKR1310

MKR1310 device eui: a8610a3233428e03 Delete device

Dashboard Configuration OTAA keys Activation Queue Events LoRaWAN frames

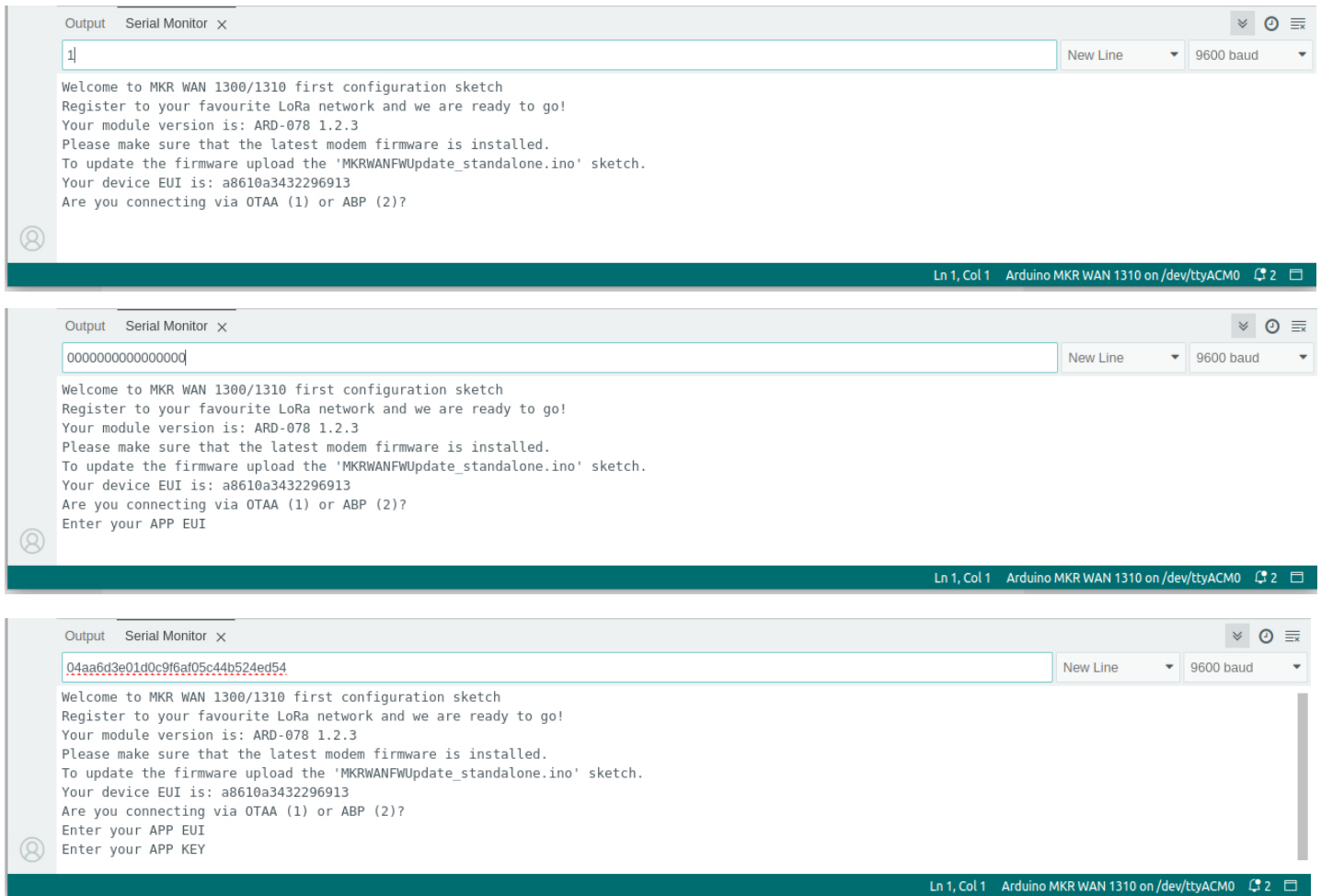
* Application key MSB C

Submit

Générer l'AppKey

5.5 Copiez cette clé dans le presse-papier et validez la création du device avec submit .

5.6 Depuis l'IDE Arduino, choisissez la connexion via OTAA (1), recopier l'app EUI, l'app key copiée précédemment dans le presse-papier.



```
Output Serial Monitor x
Welcome to MKR WAN 1300/1310 first configuration sketch
Register to your favourite LoRa network and we are ready to go!
Your module version is: ARD-078 1.2.3
Please make sure that the latest modem firmware is installed.
To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.
Your device EUI is: a8610a3432296913
Are you connecting via OTAA (1) or ABP (2)?

Ln 1, Col 1 Arduino MKR WAN 1310 on /dev/ttyACM0 9600 baud
```

```
Output Serial Monitor x
Welcome to MKR WAN 1300/1310 first configuration sketch
Register to your favourite LoRa network and we are ready to go!
Your module version is: ARD-078 1.2.3
Please make sure that the latest modem firmware is installed.
To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.
Your device EUI is: a8610a3432296913
Are you connecting via OTAA (1) or ABP (2)?
Enter your APP EUI

Ln 1, Col 1 Arduino MKR WAN 1310 on /dev/ttyACM0 9600 baud
```

```
Output Serial Monitor x
04aa6d3e01d0c9f6af05c44b524ed54
Welcome to MKR WAN 1300/1310 first configuration sketch
Register to your favourite LoRa network and we are ready to go!
Your module version is: ARD-078 1.2.3
Please make sure that the latest modem firmware is installed.
To update the firmware upload the 'MKRWANFWUpdate_standalone.ino' sketch.
Your device EUI is: a8610a3432296913
Are you connecting via OTAA (1) or ABP (2)?
Enter your APP EUI
Enter your APP KEY

Ln 1, Col 1 Arduino MKR WAN 1310 on /dev/ttyACM0 9600 baud
```

5.7 On peut vérifier les requêtes de connexion OTAA sur l'onglet LoRaWAN frames ;

Dashboard

Configuration

OTAA keys

Activation

Queue

Events

LoRaWAN frames

Download

2024-02-14 09:33:19

UnconfirmedDataDown

DevAddr: 005791db

DevEUI: a8610a3233428e03

Gateway ID: c0ee40ffff2a3fed

2024-02-14 09:33:19

ConfirmedDataUp

DevAddr: 005791db

DevEUI: a8610a3233428e03

2024-02-14 09:33:06

JoinAccept

DevEUI: a8610a3233428e03

Gateway ID: c0ee40ffff2a3fed

2024-02-14 09:33:06

JoinRequest

DevEUI: a8610a3233428e03

5.8 Vérifiez le message reçu dans l'onglet Events.

Dashboard Configuration OTAA keys Activation Queue **Events** LoRaWAN frames

2024-02-14 10:24:17 up DR: 0 Data: 48654c6f524120776f726c6421 FCnt: 0 FPort: 3

```
deduplicationId: "d5bce951-0798-46d5-8aa0-b6b6b2366e9f"
time: "2024-02-14T09:24:17.086452021+00:00"
▼ deviceInfo: {} 10 keys
  tenantId: "52f14cd4-c6f1-4fbd-8f87-4025e1d49242"
  tenantName: "ChirpStack"
  applicationId: "a24ec754-0d17-45dd-b416-61e35aacd628"
  applicationName: "Station météo"
  deviceProfileId: "c9e620d9-23ef-4bfa-90e6-548592b027cc"
  deviceProfileName: "Arduino MKR WAN 1310"
  deviceName: "MKR1310"
  devEui: "a8610a3233428e03"
  deviceClassEnabled: "CLASS_A"
  tags: {} 0 keys
  devAddr: "00c07f42"
  adr: true
  dr: 0
  fCnt: 0
  fPort: 3
  confirmed: true
  data: "SGVMb1JBIHdvcmxklQ=="
  ▼ object: {} 1 key
    ledState: null
  ▼ rxInfo: [] 1 item
    ▼ 0: {} 10 keys
      gatewayId: "c0ee40ffff2a3fed"
      uplinkId: 38391
      nsTime: "2024-02-14T09:24:16.865183285+00:00"
      rssi: -49
      snr: 8.2
      channel: 7
      location: {} 0 keys
      context: "FShuJA=="
      ▼ metadata: {} 2 keys
        region_config_id: "eu868"
        region_common_name: "EU868"
        crcStatus: "CRC_OK"
      ▼ txInfo: {} 2 keys
```

C'est du Base64.
Converti en caractère, on
reçoit : *HeLoRA world!*

6. RECEVOIR DES DONNÉES SUR LE SERVEUR CHIRPSTACK : UPLINK

6.1. Ouvrir le sketch LoraSendAndReceive.ino

On constate qu'il est associé à un fichier `arduino_secrets.h`. Ce fichier contiendra l'APP EUI et l'APP KEY.

6.2. Recopier ces identifiants sur le fichier et téléverser le sketch :



6.3. Saisir un message à envoyer.

```
Output Serial Monitor x
Hello Lora

Your module version is: ARD-078 1.2.3
Your device EUI is: a8610a35301e8909

Enter a message to send to network
(make sure that end-of-line 'NL' is enabled)
```

```
Output Serial Monitor x
Message (Enter to send message to 'Arduino MKR WAN 1310' on '/dev/cu.usbmodem14401')

Your module version is: ARD-078 1.2.3
Your device EUI is: a8610a35301e8909

Enter a message to send to network
(make sure that end-of-line 'NL' is enabled)

Sending: Hello Lora - 48 65 6C 6C 6F 20 4C 6F 72 61
Message sent correctly!
No downlink message received at this time.

Enter a message to send to network
(make sure that end-of-line 'NL' is enabled)
```

6.4. Dans l'onglet Events, on constate la réception du message

Dashboard Configuration OTAA keys Activation Queue **Events** LoRaWAN frames

2024-02-14 10:47:07 🔍 up DR: 0 Data: 48656c6c6f204c6f7261 FCnt: 0 FPort: 2

fPort: 2
confirmed: true
data: "SGVsbG8gTG9yYQ=="

Message en base64 qui donne Hello Lora

7. TRANSMETTRE DES DONNÉES DEPUIS LE SERVEUR CHIRPSTACK

5.9 On programme l'envoi de données depuis l'onglet Queue.

Dashboard Configuration OTAA keys Activation **Queue** Events LoRaWAN frames

Enqueue

Confirmed: ☐ FPort: Is encrypted ☐

HEX BASE64 JSON

Enqueue

5.10 On peut vérifier la programmation de l'envoi du message ci-dessous :

ID	Pending	Encrypted	Frame-counter	Confirmed	FPort	Data (HEX)
3fbd6778-c507-41f1-b745-57307396b775	no	no		no	1	aa

5.11 La transmission sera déclenchée lorsque le serveur aura reçu un message, on transmet donc depuis le device arduino.

Après 10 secondes, le message est reçu sur le device.

```
Message (Enter to send message to 'Arduino MKR WAN 1310' on '/dev/serial0')
Enter a message to send to network
(make sure that end-of-line 'NL' is enabled)

Sending: test - 74 65 73 74
Message sent correctly!
Received: AA
```

6 UPLINK ET DOWNLINK VIA MQTT

UPLINK :

7.1. Depuis le terminal d'un PC, générez la commande `mqtt subscribe` avec `mosquitto`. Elle est de la forme :

```
mosquitto_sub -h ip_du_serveur_lorawan -t
"application/id_application/device/dev_eui/event/up"
```

exemple :

```
mosquitto_sub -h 192.168.1.29 -t "application/a24ec754-0d17-45dd-b416-61e35aacd628/device/a8610a3233428e03/event/up"
```

7.2. Depuis le device Arduino, générez un flux uplink en émettant un message.

```
Enter a message to send to network
(make sure that end-of-line 'NL' is enabled)

Sending: Hello Lora - 48 65 6C 6C 6F 20 4C 6F 72 61
Message sent correctly!
```

7.3. Vérifiez le message reçu, il ressemble au message suivant :

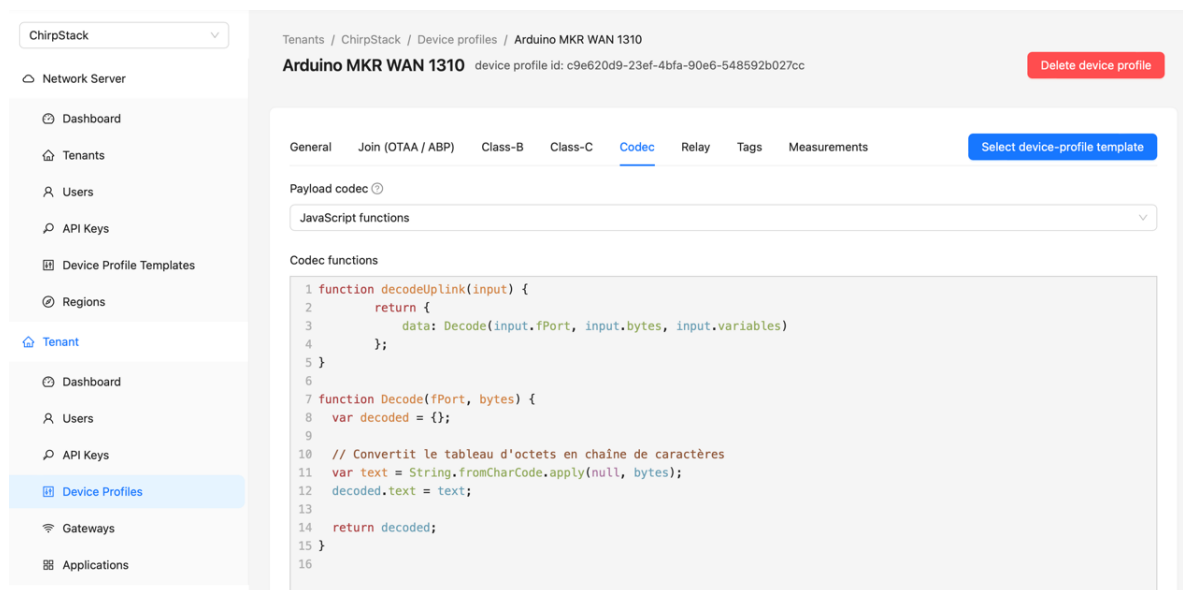
```
4fbd-8f87-4025e1d49242", "tenantName": "ChirpStack", "applicationId": "a24ec754-0d17-45dd-b416-61e35aacd628", "applicationName": "Station météo", "dev  
iceProfileId": "c9e620d9-23ef-4bfa-90e6-548592b027cc", "deviceProfileName": "Arduino MKR WAN 1310", "deviceName": "MKR1310", "devEui": "a8610a3233428e  
03", "deviceClassEnabled": "CLASS_A", "tags": {}, "devAddr": "00504025", "adr": true, "dr": 0, "fCnt": 0, "fPort": 2, "confirmed": true, "data": "SGVsbG8gTG9yYQ==",  
"rxInfo": [{"gatewayId": "c0ee40ffff2a3fed", "uplinkId": 48272, "nsTime": "2024-02-15T15:14:50.102679928+00:00", "rssi": -28, "snr": 9.8, "rfChain": 1,  
"location": {}, "context": "GKyjFA=", "metadata": {"region_config_id": "eu868", "region_common_name": "EU868"}, "crcStatus": "CRC_OK"}], "txInfo": {"frequ  
ency": 868100000, "modulation": {"lora": {"bandwidth": 125000, "spreadingFactor": 12, "codeRate": "CR_4_5"}}}]
```

Les données reçues sont en base64 : SGVsbG8gTG9yYQ== qui donne Hello Lora.

7.4. On peut décoder le message et le transférer au format json en ajoutant un script js comme ci-dessous :

Source :

<https://github.com/bouhenic/FormationIOT/blob/main/journée3/TP2Chirpstack/DecodePayload.js>



8.5 Transmettez de nouveau un message depuis le device Arduino :

Sending: Hello Lora JS - 48 65 6C 6C 6F 20 4C 6F 72 61 20 4A 53
Message sent correctly!

7.5. Le message reçu est le suivant :

```
{"deduplicationId": "f08a4c8e-d2e8-41c7-b3b2-16be5ccf80b2", "time": "2024-02-15T15:39:10.889674134+00:00", "deviceInfo": {"tenantId": "52f14cd4-c6f1-  
4fbd-8f87-4025e1d49242", "tenantName": "ChirpStack", "applicationId": "a24ec754-0d17-45dd-b416-61e35aacd628", "applicationName": "Station météo", "dev  
iceProfileId": "c9e620d9-23ef-4bfa-90e6-548592b027cc", "deviceProfileName": "Arduino MKR WAN 1310", "deviceName": "MKR1310", "devEui": "a8610a3233428e  
03", "deviceClassEnabled": "CLASS_A", "tags": {}, "devAddr": "01ef35da", "adr": true, "dr": 0, "fCnt": 0, "fPort": 2, "confirmed": true, "data": "SGVsbG8gTG9yYS  
BKUw==", "object": {"text": "Hello Lora JS"}, "rxInfo": [{"gatewayId": "c0ee40ffff2a3fed", "uplinkId": 53028, "nsTime": "2024-02-15T15:39:10.669807126+00  
:00", "rssi": -25, "snr": 9.2, "channel": 7, "location": {}, "context": "b7vWLA=", "metadata": {"region_config_id": "eu868", "region_common_name": "EU868"}, "c  
rcStatus": "CRC_OK"}], "txInfo": {"frequency": 867900000, "modulation": {"lora": {"bandwidth": 125000, "spreadingFactor": 12, "codeRate": "CR_4_5"}}}]
```

On voit maintenant les données en base 64 et au format json.

DOWNLINK :

6.3 Depuis le terminal d'un PC, publier un message vers le device. La commande mosquitto est de la forme :

```
mosquitto_pub -h IP -t "application/a24ec754-0d17-45dd-b416-61e35aacd628/device/a8610a3233428e03/command/down" -m '{"devEui": "a8610a3233428e03", "confirmed": false, "fPort": 10, "data": "AQ=="}'
```

Exemple :

```
mosquitto_pub -h 192.168.1.29 -t "application/a24ec754-0d17-45dd-b416-61e35aacd628/device/a8610a3233428e03/command/down" -m '{"devEui": "a8610a3233428e03", "confirmed": false, "fPort": 10, "data": "AQ=="}'
```