RAK7289 Supported LoRa Network Servers

AWS IoT Core for LoRaWAN

Execute the following steps to set up your AWS account and permissions:

Set Up Your AWS Account and Permissions

If you don't have an AWS account, refer to the AWS guide . The relevant sections are **Sign up for an AWS account** and **Create a user and grant permissions**.

Overview

The high-level steps to get started with AWS IoT Core for LoRaWAN are as follows:

1. Set up Roles and Policies in IAM
2. Add a Gateway (see section Add the Gateway to AWS IoT)
3. Add Devices (see section Add a LoRaWAN Device to AWS IoT)
   - Verify device and service profiles
   - Set up a Destination to which device traffic will be routed and processed by a rule.

These steps are discussed as you browse through this guide. For additional details, refer to the AWS LoRaWAN Developer Guide .

Add the Gateway to AWS IoT

Preparation

Refer to the online guide for steps required prior to onboarding your gateway. For more details, check the software section of the datasheet .

Frequency Band Selection and Role Setup

Refer to the online guide for information on selecting an appropriate frequency band.

> 📝 **NOTE**
>
> LoRa Frequency bands supported by RAK7289 are as follows:
>
> - RU864
> - IN865
> - EU868
> - US915
> - AU915
> - KR920
> - AS923
> - CN470
>
> You can select an appropriate frequency band from the RAK store .

Add the LoRaWAN Gateway

To register the gateway with AWS IoT Core for LoRaWAN, follow the steps in the online guide under the **Add a gateway using the console** section.

Add a LoRaWAN Device to AWS IoT

Preparation

- Go to the datasheet to learn more about the RAK4631 WisBlock LPWAN Module.
- Follow the steps in the online guide under the **Before onboarding your wireless device** section, then proceed to the **Add your wireless device to AWS IoT Core for LoRaWAN** section.

Verify Profiles

AWS IoT Core for LoRaWAN supports device profiles and service profiles. Device profiles contain the communication and protocol parameter values the device needs to communicate with the network server. Service profiles describe the communication parameters the device needs to communicate with the application server.

Some pre-defined profiles are available for device and service profiles. Before proceeding, verify that these profile settings match the devices you will be setting up to work with AWS IoT Core for LoRaWAN. For more details, refer to the online guide under the **Add profiles to AWS IoT Core for LoRaWAN** section.

> 📝 **NOTE**
>
> Proceed only if you have a device and service profile that will work for you.

Set Up a Destination for Device Traffic

Because most LoRaWAN devices don't send data to AWS IoT Core for LoRaWAN in a format that can be consumed by AWS services, traffic must first be sent to a Destination. A Destination represents the AWS IoT rule that processes a device's data for use by AWS services. This AWS IoT rule contains the SQL statement that selects the device's data and the topic rule actions that send the result of the SQL statement to the services that will use it.

For more information, refer to the online guide under sections **Add a destination using the console** and **Create an IAM role for your destinations**. Also, refer to **Create rules to process LoRaWAN device messages** section in the online guide .

Set Up the Gateway

- **Set Up the Gateway Hardware**
- **Set Up the Gateway Software**

For Additional Software References, check the following links:

- **FAQ**
- **Community Forum**

Configure the Gateway Device

RAK7289 With WisGate OS 1 LoRaGateway Setting

By default, the gateway will work in Wi-Fi AP Mode, which means you can find an SSID named "**RAK7289_XXXX**" on your PC's Wi-Fi Network List. "**XXXX**" is the last two bytes of the gateway MAC address.

1. To access the Web Management Platform, input the IP Address `192.168.230.1` in your web browser.

> 📝 **NOTE**
>
> No password is required to connect via Wi-Fi.

2. Using your preferred web browser, input the aforementioned IP Address and you should see the same login page shown in **Figure 1**. Login the credentials provided below:

- Username: **root**
- Password: **root**

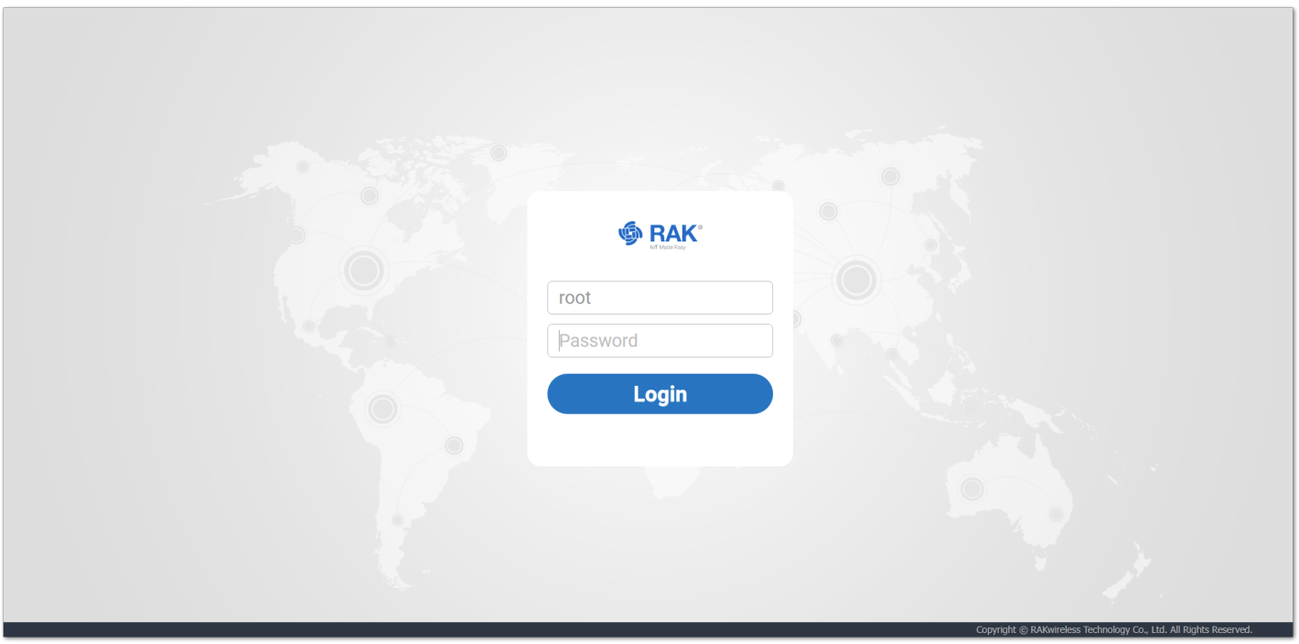

Figure 1: Login page

3. The first firmware version that supports AWS IoT Core for LoRaWAN is **1.2.0065_Release_r209**. It can be verified on **Status** -> **Overview** -> **System** -> **Firmware Version**.

4. Navigate to **System** -> **Backup/Flash Firmware** -> **Flash new firmware image**, and upgrade the firmware.

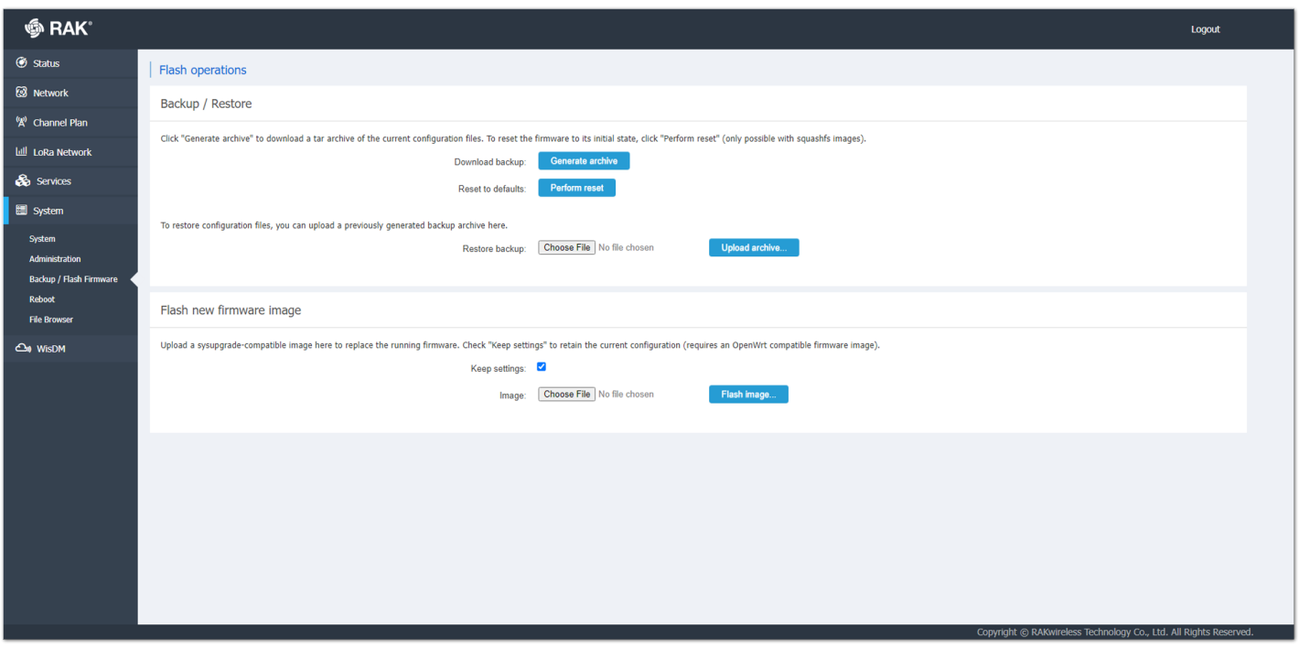

Figure 2: Flash new firmware image

4. Configure Network Mode to Basic Station.

- Navigate to **LoRa Network** -> **Network Settings**.
- Change **Mode** in LoRaWAN Network Settings to **Basic Station**.
- Select **LNS Server** from Server, then select **TLS Server and Client Authentication** from Authentication Mode.

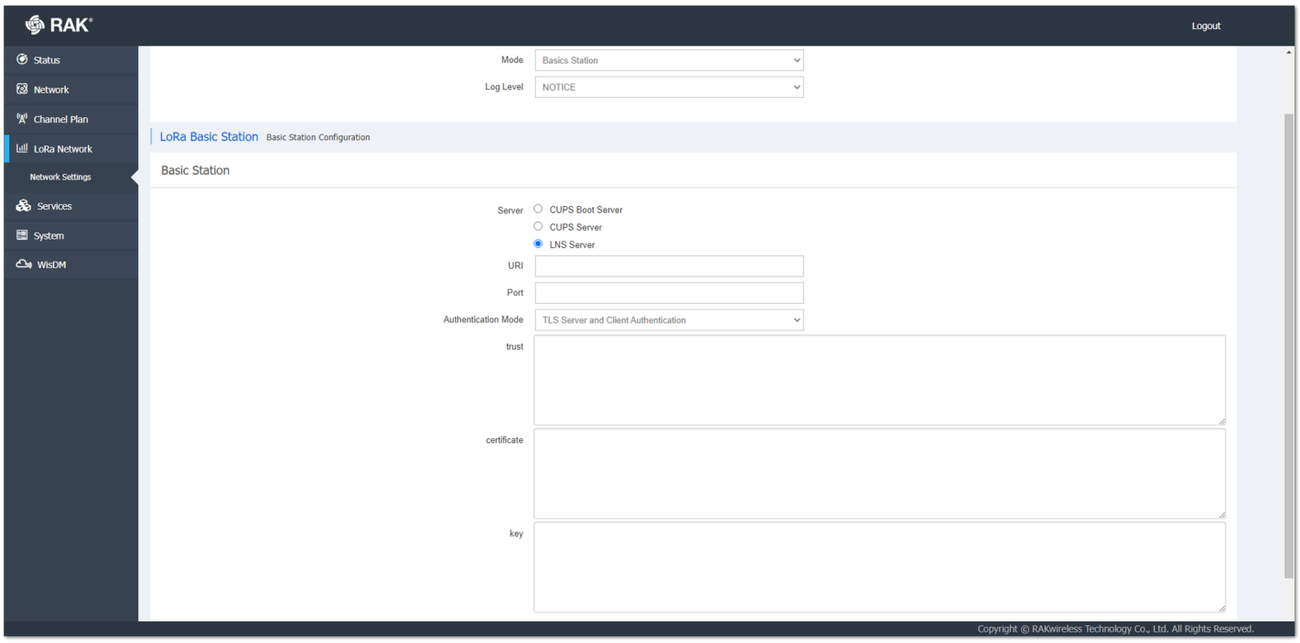

Figure 3: Configure network settings to Basic Station

5. Configure URI, Port and Authentication Mode.

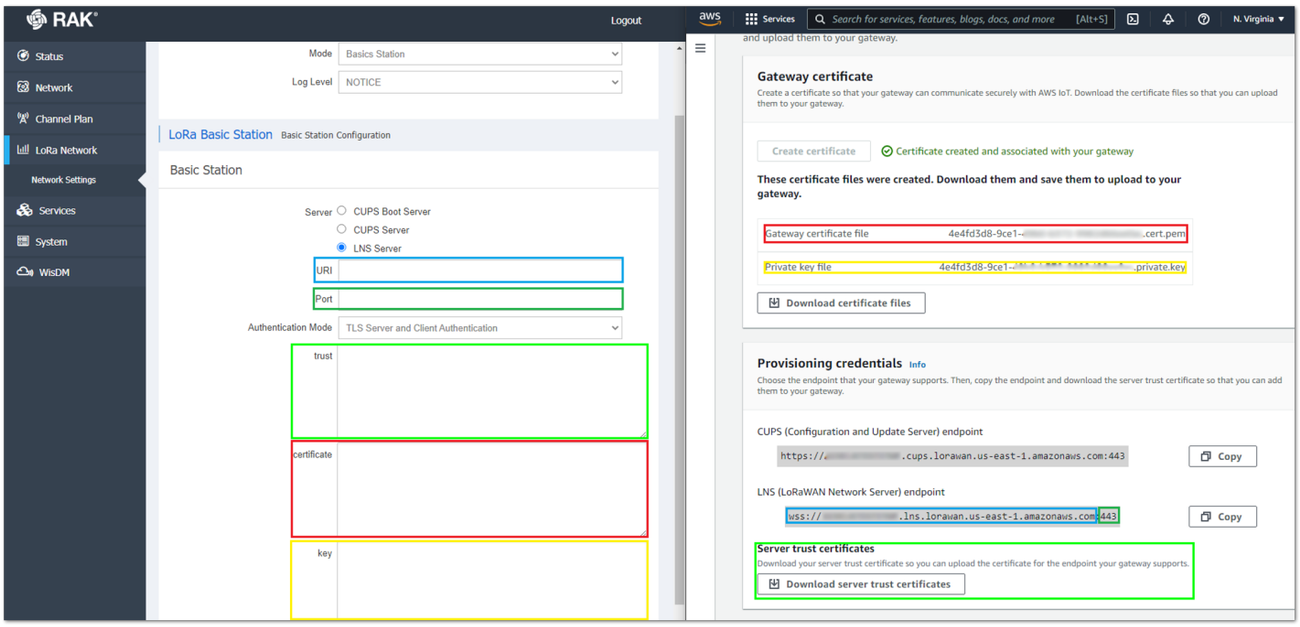

Figure 4: Configure network settings to Basic Station

RAK7289 With WisGate OS 2 LoRaGateway Setting

By default, the gateway will work in Wi-Fi AP Mode, which means that you can find an SSID named like "**RAK7289CV2_XXXX**" on your PC's Wi-Fi Network List. "**XXXX**" is the last two bytes of the Gateway MAC address.

1. To access the Web Management Platform, input the IP Address: `192.168.230.1` in your web browser.

> 📝 **NOTE**
>
> No password is required to connect via Wi-Fi.

2. Using your preferred web browser, input the aforementioned IP Address and you should see the same login page shown in **Figure 5**. If it is the first time to login, the system will let you reset the password.

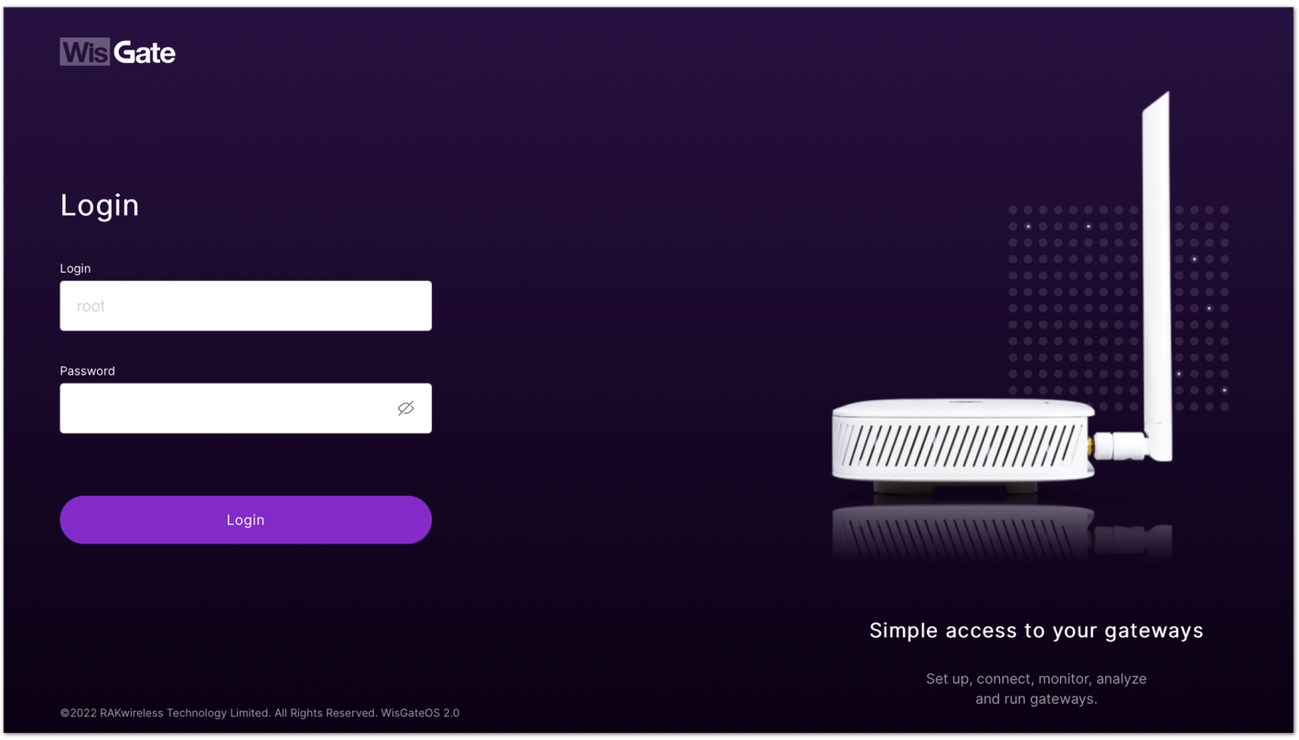

Figure 5: Logging in to WisGate OS 2

3. Navigate to LoRa, change **Work Mode** to **Basics Station**. Select **LNS Server** from Server, then select **TLS Server** and **Client Authentication** from Authentication Mode.

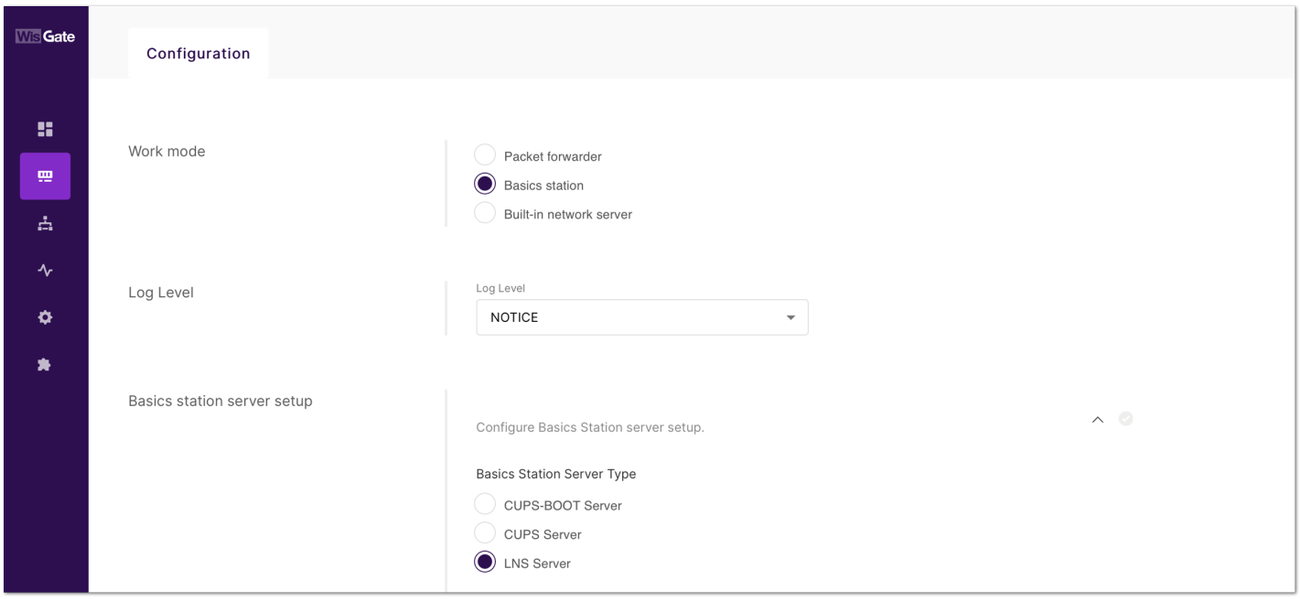

Figure 6: Basic station configuration

4. Configure **URI**, **Port**, and **Authentication Mode**.

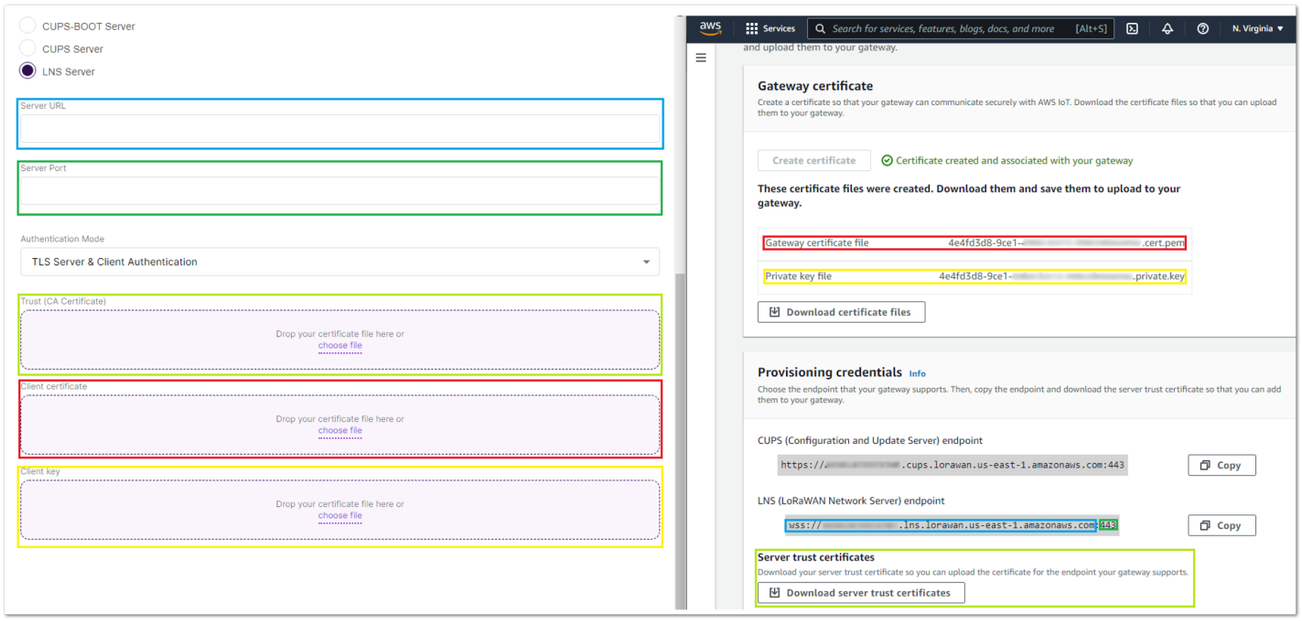

Figure 7: LNS server configuration

Connect the Gateway and Verify the Connection Status

Follow the instructions in the online guide to connect your gateway to AWS IoT Core for LoRaWAN.

To verify the connection status, refer to the instructions in the **Check gateway connection status using the console** section.

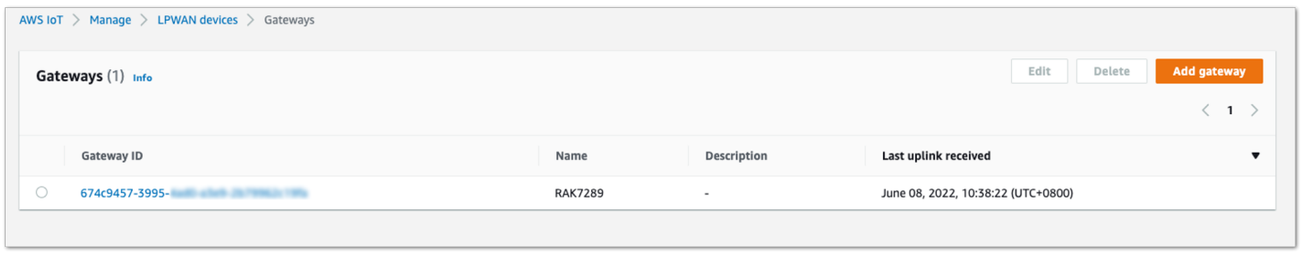

Figure 8: LNS server configuration

Add End-Devices

Refer to RAK4631 Quick Start Guide to enable communication with the gateway.

Connect the Device and Verify the Connection Status

Follow the instructions in the online guide to connect your device to AWS IoT Core for LoRaWAN.

To verify the connection status, refer to the instructions in the **Check device connection status using the console** section. You can also check **View format of uplink messages sent from LoRaWAN devices** .

Verifying Operation

Once setup is completed, provisioned OTAA devices can join the network and start to send messages. Messages from devices can then be received by AWS IoT Core for LoRaWAN and forwarded to the IoT Rules Engine.

Instructions for a sample Hello World application are given below, assuming that the device has joined and is capable of sending uplink traffic.

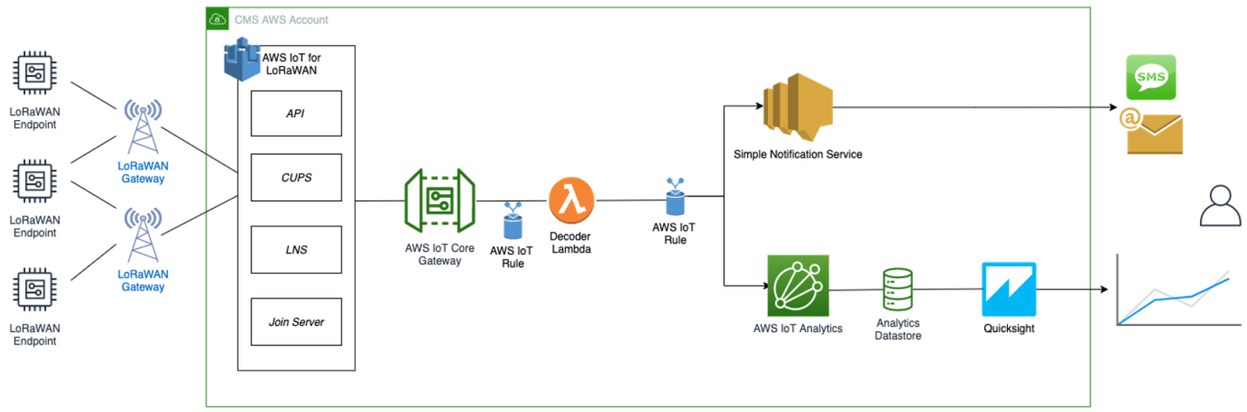

Figure 9: Sending Uplink Architecture

Create a Lambda Function for Destination Rule

Create the lambda function to process device messages processed by the destination rule.

1. Go to the AWS Lambda console .
2. In the navigation pane, click on **Functions**, then **Create function**.
3. Select **Author from scratch**.
4. Under **Basic Information**, enter the function name and choose ***Runtime Python 3.8***. from the drop-down under **Runtime**.
5. Click **Create function**.
6. Under **Function code**, paste the copied code into the editor under the ***lambda_function.py*** tab.

```py
import base64
import json
import logging
import ctypes
import boto3

# define function name
FUNCTION_NAME = "RAK-HelloWorld"

# Second Byte in Payload represents Data Types
# Low Power Payload Reference: https://developers.mydevices.com/cayenne/docs/lora/
DATA_TYPES = 1

# Type Temperature
TYPE_TEMP = 0x67

# setup iot-data client for boto3
client = boto3.client('iot-data')

# setup logger
logger = logging.getLogger(FUNCTION_NAME)
logger.setLevel(logging.INFO)

def decode(event):
  data_base64 = event.get("PayloadData")
  data_decoded = base64.b64decode(data_base64)

  result = {
      "devEui": event.get("WirelessMetadata").get("LoRaWAN").get("DevEui"),
      "fPort": event.get("WirelessMetadata").get("LoRaWAN").get("FPort"),
      "freq": event.get("WirelessMetadata").get("LoRaWAN").get("Frequency"),
      "timestamp": event.get("WirelessMetadata").get("LoRaWAN").get("Timestamp")
  }

  if data_decoded[DATA_TYPES] == TYPE_TEMP:
      temp = (data_decoded[DATA_TYPES + 1] << 8) | (data_decoded[DATA_TYPES + 2])
      temp = ctypes.c_int16(temp).value
      result['temperature'] = temp / 10

  return result

def lambda_handler(event, context):
  data = decode(event)
  logger.info("Data: %s" % json.dumps(data))

  response = client.publish(topic = event.get("WirelessMetadata").get("LoRaWAN").get("DevEui") +

  return response
```

7. Once the code has been pasted, choose **Deploy** to deploy the lambda code.
8. Click the **Configuration** tab of the lambda function and head to the **Permissions** menu.
9. Change the **Lambda Role Policy** permission.
   - Under **Execution role**, click on the hyperlink under **Role name**.
   - On the **Permissions tab**, find the policy name and select it.
   - Choose **Edit policy**, and choose the **JSON** tab.
   - Append the following to the Statement section of the policy to allow publishing to AWS IoT.

```json
{
"Effect":"Allow",
"Action":[
    "iot:Publish"
],
"Resource":[
    "*"
]
}
```

- Choose **Review Policy**, then Save changes.

10. Return to the **Code** tab and create a test event that will allow you to test the functionality of the lambda function.
    - In the **Test** drop-down menu, choose **Configure test events**.
    - Enter a name for the test event under the **Event name**.
    - Paste the following sample payload in the area under Event name:

```json
  {
  "WirelessDeviceId": "65d128ab-90dd-4668-9556-fe47c589610b",
  "PayloadData": "AWf/1w==",
  "WirelessMetadata": {
  "LoRaWAN": {
  "DataRate": "4",
  "DevEui": "0000000000000088",
  "FPort": 1,
  "Frequency": "868100000",
  "Gateways": [
          {
  "GatewayEui": "80029cffXXXXXXXX",
  "Rssi": -109,
  "Snr": 5
          }
      ],
  "Timestamp": "2021-02-08T04:00:40Z"
      }
  }
  }
```

11. Choose **Create** to save the event.
12. Navigate to the AWS IoT console, choose **Test** on the navigation pane, and select **MQTT client**.
13. Configure the MQTT client to subscribe to "#" (all topics).
14. Click on **Test** in the Lambda function page to generate the test event you just created.
15. Verify the published data in the AWS IoT Core MQTT Test client:
    - Open another window. Go to **AWS IoT Console**, select **Test** under Subscription Topic, **enter #** and select to **Subscribe to topic**.
    - The output should look similar to this:

```json
0000000000000088/project/sensor/decoded      February 09, 2021, 14:45:29 (UTC+0800)
{
    "devEui": "0000000000000088",
    "fPort": 1,
    "freq": "868100000",
    "timestamp": "2021-02-08T04:00:40Z",
    "temperature": -4.1
}
```

Create the Destination Rule

In this step, create the IoT rule that forwards the device payload to your application. This rule is associated with the destination created earlier in Set up a Destination for device traffic section.

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act**, then select **Rules**.
3. On the Rules page, choose **Create**.
4. On the Create a rule page, enter as follows:
   - Name: **LoRaWANRouting**
   - Description: **Any description of your choice**.

> 📝 **NOTE:**
>
> The **Name of your Rule** is the information needed when you provision devices to run on AWS IoT Core for LoRaWAN.

5. Leave the default Rule query statement '**SELECT * FROM 'iot/topic**' unchanged. This query has no effect at this time, as traffic is currently forwarded to the rules engine based on the destination.
6. Under **Set one or more actions**, choose **Add action**.
7. On the Select an action page, choose **Republish a message to an AWS IoT topic**. Scroll down, and choose **Configure action**.
8. On the Configure action page, for Topic, enter ***project/sensor/decoded***.The AWS IoT Rules Engine will forward messages to this topic.
9. Under **Choose or create a role to grant AWS IoT access to perform this action**, select **Create Role**.
10. For Name, enter a name of your choice.
11. Choose **Create role** to complete the role creation. You will see a "**Policy Attached**" tag next to the role name, indicating that the Rules Engine has been permitted to execute the action.
12. Choose **Add action**.
13. Add one more action to invoke the Lambda function. Under **Set one or more actions**, choose **Add action**.
14. Choose **Send a message to a Lambda function**.
15. Choose **Configure action**.
16. Select the Lambda function created earlier and choose **Add action**.
17. Then, choose **Create rule**.
18. A "**Success**" message will be displayed at the top of the panel, and the destination has a rule bound to it.

You can now check that the decoded data is received and republished by AWS by triggering a condition or event on the device itself.

- Go to the AWS IoT console. In the navigation pane, select **Test**, and choose **MQTT client**.
- Subscribe to the wildcard topic # to receive messages from all topics.
- Send message from endDevice using AT command: `at+send=1:01670110`.
- You should see traffic similar as shown below:

```json
393331375d387505/project/sensor/decoded          February 09, 2021, 14:47:21 (UTC+0800)
{
"devEui": "393331375d387505",
"fPort": 1,
"freq": "867100000",
"timestamp": "2021-02-09T06:47:20Z",
"temperature": 27.2
}
```

```json
project/sensor/decoded    February 09, 2021, 14:47:21 (UTC+0800)
  {
    "WirelessDeviceID": "6477ec22-9570-31d5981da021",
    "PayloadData": "AWcBEA==",
    "WirelessMetadata": {
        "LoRaWAN": {
            "DataRate": "4",
            "DevEui": "393331375d387505",
            "FPort": 1,
            "Frequency": "867100000",
            "Gateways": [
                {
                    "GatewayEui": "ac1ff09fffe014bd5",
                    "Rssi": -103,
                    "Snr": 8.5
                }
            ],
            "Timestamp": "2021-02-09T06:47:20Z"
        }
    }
  }
```

Configuring Amazon SNS

You will be using the Amazon Simple Notification Service to send text messages (SMS) when certain conditions are met.

1. Go to the Amazon SNS console .
2. Click on the menu to open the navigation pane.
3. Select **Text Messaging** (SMS) and choose **Publish text message**.
4. Under the **Message** type, select **Promotional**.
5. Enter your phone number (phone number that will receive text alerts).
6. Enter **Test message** for the Message and choose **Publish** message.
7. If the phone number you entered is valid, you will receive a text message and your phone number will be confirmed.
8. Create an Amazon SNS Topic as follows:
   - In the navigation pane, choose **Topics**.
   - Select **Create topic**.
   - Under **Type**, select **Standard**.
   - Enter a name of your choice. Here, you will use `text_topic` .
   - Choose **Create topic**.
9. Create a subscription for this topic:
   - In the page for the newly created `text_topic` , choose the **Subscriptions** tab.
   - Choose **Create subscription**.

- In **Topic ARN**, choose the topic you have created earlier.
- Select **Protocol** as SMS from the drop-down.
- Under **Endpoint**, enter the previously validated phone number to receive the SMS alerts.
- Choose **Create subscription**. You should see a "***Subscription to text_topic created successfully***" message.

Add a Rule for Amazon SNS Notification

Now, add a new rule to send an Amazon SNS notification when certain conditions are met in a decoded message.

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act**, then **Rules**.
3. On the Rules page, choose **Create**.
4. Enter the Name as `text_alert` and provide an appropriate **Description**.
5. Under the **Rule query statement**, enter the following query:

```
SELECT devEui as device_id, "Temperature exceeded 25" as message, temperature as temp, timestamp
```

6. Choose **Add action**.
7. Choose **Send a message as an SNS push notification**.
8. Choose **Configure action**.
9. Under SNS target, select `text_topic` from the drop-down.
10. Select **RAW** under **Message format**.
11. Under **Choose or create a role to grant AWS IoT access to perform this action**, choose **Create role**.
12. Enter a name for the role, and choose **Add action**.
13. Select **Create rule**. You should see a "**Success**" message, indicating that the rule has been created.

Test the Rule for Amazon SNS Notification

After adding the rule for Amazon SNS notification, you should receive a text message when hitting the event.

Send message from end-device using AT command: `at+send=1:01670110`. Here is the message from mobile after sending an uplink message:

```json
{
    "device_id": "393331375d387505",
    "message": "Temperature exceeded 25",
    "temp": 27.2,
    "time": "2021-02-22T07:58:54Z"
}
```

Send Downlink Payload

This section shows how to send downlink payload from AWS IoT LoRaWAN Server to end-device.

Install the AWS SAM CLI

Follow the instruction in the online guide to install the AWS SAM CLI.

Deploy the SAM Template to AWS

Follow the instruction in the Deploy SAM template to AWS GitHub.

Send Payload to End-Device

1. Go to the AWS IoT console.
2. In the navigation pane, select **Test**, and choose **MQTT client**.
3. Subscribe to the wildcard topic # to receive messages from all topics.
4. Specify the topic to `cmd/downlink/{WirelessDeviceId}` and a base64-encoded message.

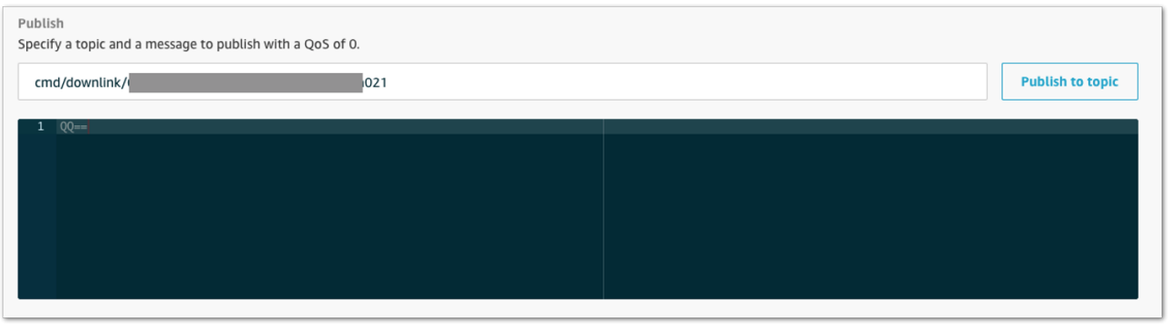

Figure 10: Specifying a topic

5. You should see traffic on AWS similar as shown below:

```json
downlink/status/6477ec22-9570-4fea-9668-31d5981da021   February 09, 2021, 15:09:29 (UTC+0800)
{
    "sendresult": {
        "status": 200,
        "RequestId": "4f1d36e1-8316-4436-8e9d-2207e3711755",
        "MessageId": "60223529-0011d9f5-0095-0008",
        "ParameterTrace": {
            "PayloadDate": "QQ==",
            "WirelessDeviceId": "6477ec22-9570-4fea-9668-31d5981da021",
            "Fport": 1,
            "TransmitMode": 1
        }
    }
}
```

Figure 11: Traffic on AWS

5. You should see traffic on your console of end-device similar as shown below:

```
SYSLOG:4:LoRa rX : 41 - 14
SYSLOG:4:LoRa Tx :
```

IoT Analytics

You will use IoT Analytics to visually display data via graphs if there is a need in the future to do further analysis.

Create an IoT Analytics Rule

Create a Rule First

1. Navigate to the AWS IoT console .
2. In the navigation pane, choose **Act** and then, choose **Rules**.
3. On the Rules page, choose **Create**.
4. Enter the Name as **Visualize** and provide an appropriate **Description**.
5. Under the **Rule query statement**, enter the following query:

```
SELECT * FROM 'project/sensor/decoded'
```

6. Choose **Add action**.
7. Select **Send a message to IoT Analytics**.
8. Choose **Configure Action**.
9. Choose **Quick Create IoT Analytics Resources**.
10. Under **Resource Prefix**, enter an appropriate prefix for your resources, such as *LoRa Choose Quick Create*.
11. Once the **Quick Create Finished** message is displayed, choose **Add action**.
12. Choose **Create rule**. You should see a Success message, indicating that the rule has been created.

Configure AWS IoT Analytics

Set Up AWS IoT Analytics

1. Go to the AWS IoT Analytics console .
2. In the navigation panel, choose **Data sets**.
3. Select the data set generated by the Quick Create in **Create an IoT Analytics Rule**.
4. In the Details section, edit the **SQL query**.
5. Replace the query with as follows:

```
SELECT devEui as device_id, temperature as temp, timestamp as time FROM LoRa_datastore
```

6. Under **Schedule**, choose **Add schedule**.
7. Under **Frequency**, choose **Every 1 minute**, and then click **Save**.

Configure Amazon QuickSight

Amazon QuickSight lets you easily create and publish interactive BI dashboards that include Machine Learning-powered insights.

1. Go to AWS Management console .
2. From the management console, enter **QuickSight** in the "*Search for services, features..*" search box.
3. Click on **QuickSight** in the search results.
4. If you haven't signed up for the service before, go ahead and sign up, as there is a free trial period.
5. Select the **Standard Edition**, and choose **Continue**.
6. Enter a unique name in the field **QuickSight account name**.
7. Fill in the **Notification email address**.

8. Review the other checkbox options and change them as necessary. The **AWS IoT Analytics** option must be selected.
9. Choose **Finish**. You will see a confirmation message.
10. Choose **Go to Amazon QuickSight**.
11. Select **Datasets**.
12. Select **New dataset**.
13. Select **AWS IoT Analytics**.
14. Under Select an AWS IoT Analytics data set to import, choose the data set created in **Create an IoT Analytics Rule**.
15. Choose **Create data source**, and then choose **Visualize**.
16. Select the dataset created, then select **Refresh** or **Schedule Refresh** for a periodic refresh of the dataset.

Testing Your "Hello Word" Application

Using your device, create a condition to generate an event such as a high-temperature condition. If the temperature is above the configured threshold, then you will receive a text alert on your phone. This alert will include key parameters about the alert.

You can also visualize the data set as follows:

1. Go to the AWS IoT Analytics console .
2. Choose **Data sets**.
3. Select the dataset created earlier.
4. Select **Content** and ensure there are at least few uplink entries available in the data set.
5. Go to the **QuickSight console** .
6. Choose **New analysis**.
7. Choose the dataset created in **Create an IoT Analytics Rule**.
8. To see a chart of your dataset, select the following values:
   - **Time** on the X-axis
   - **Value** as temp (Average)
   - **Color** as device_id.

Debugging

If you experience any issues, you can check the logs located in the `/var/log/` directory.

Troubleshooting

- Unable to see the web login:
  - Check that your wifi is connected to **RAK7Wireless_XXXX**.
  - Try ping **192.168.230.1**.

The Things Network v3 (TTNv3)

In this section, it will be shown how to connect RAK7289 WisGate Edge Pro to TTNv3.

To login into the TTNv3, head on here . If you already have a TTN account, you can use your The Things ID credentials to log in.

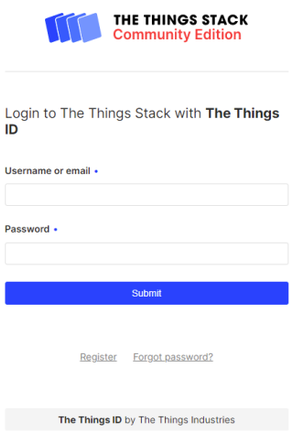

Figure 12: The Things Stack Home Page

> 📝 **NOTE**
>
> This tutorial is for the EU868 Frequency band.

Registering the Gateway

1. To register a commercial gateway, choose **Register a gateway** (for new users that do not already have a registered gateway) or go to **Gateways** > **+ Add gateway** (for users that have registered gateways before).

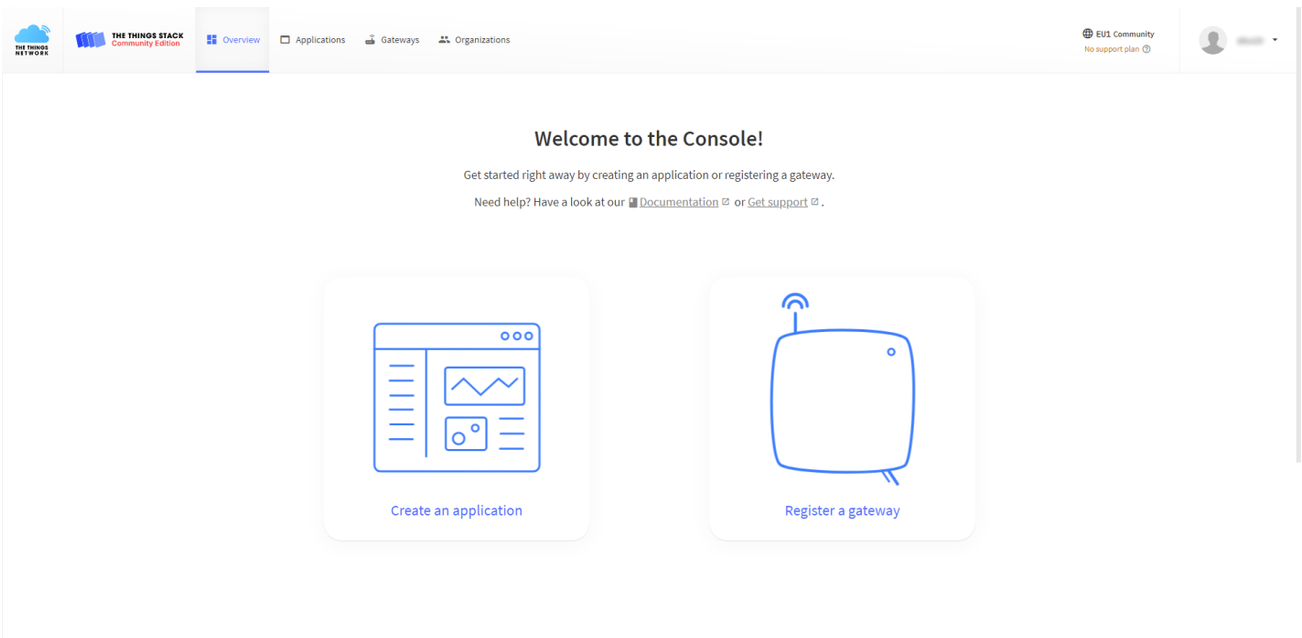

Figure 13: Console Page after successful login

2. Fill in the needed information:

- **Owner** – Automatically filled by The Things Stack, based on your account or created Organization.
- **Gateway ID** – This will be the unique ID of your gateway in the Network. Note that the ID must contain only lowercase letters, numbers, and dashes (-).
- **Gateway EUI** - A 64 bit extended unique identifier for your gateway. The gateway's EUI can be found either on the sticker on the casing or by going to the **LoRa Network Settings** page in the **LoRa Gateway** menu accessible via the Web UI. Instructions on how to access your gateway via Web UI can be found in the product's Quickstart Guide .

- **Gateway name** – A name for your gateway.
- **Gateway description (optional)** - Optional gateway description; can also be used to save notes about the gateway.
- **Gateway Server address** - The address of the Gateway Server to connect to.

> 📝 **NOTE**
>
> This tutorial is based on using the EU868 frequency band, so the server address will be: eu1.cloud.thethings.network.

- **Frequency plan** - The frequency plan used by the gateway.

> 📝 **NOTE**
>
> For this tutorial, we will use Europe 863-870 MHz (SF12 for RX2 - recommended).

- The other settings are optional and can be changed to satisfy your requirements.

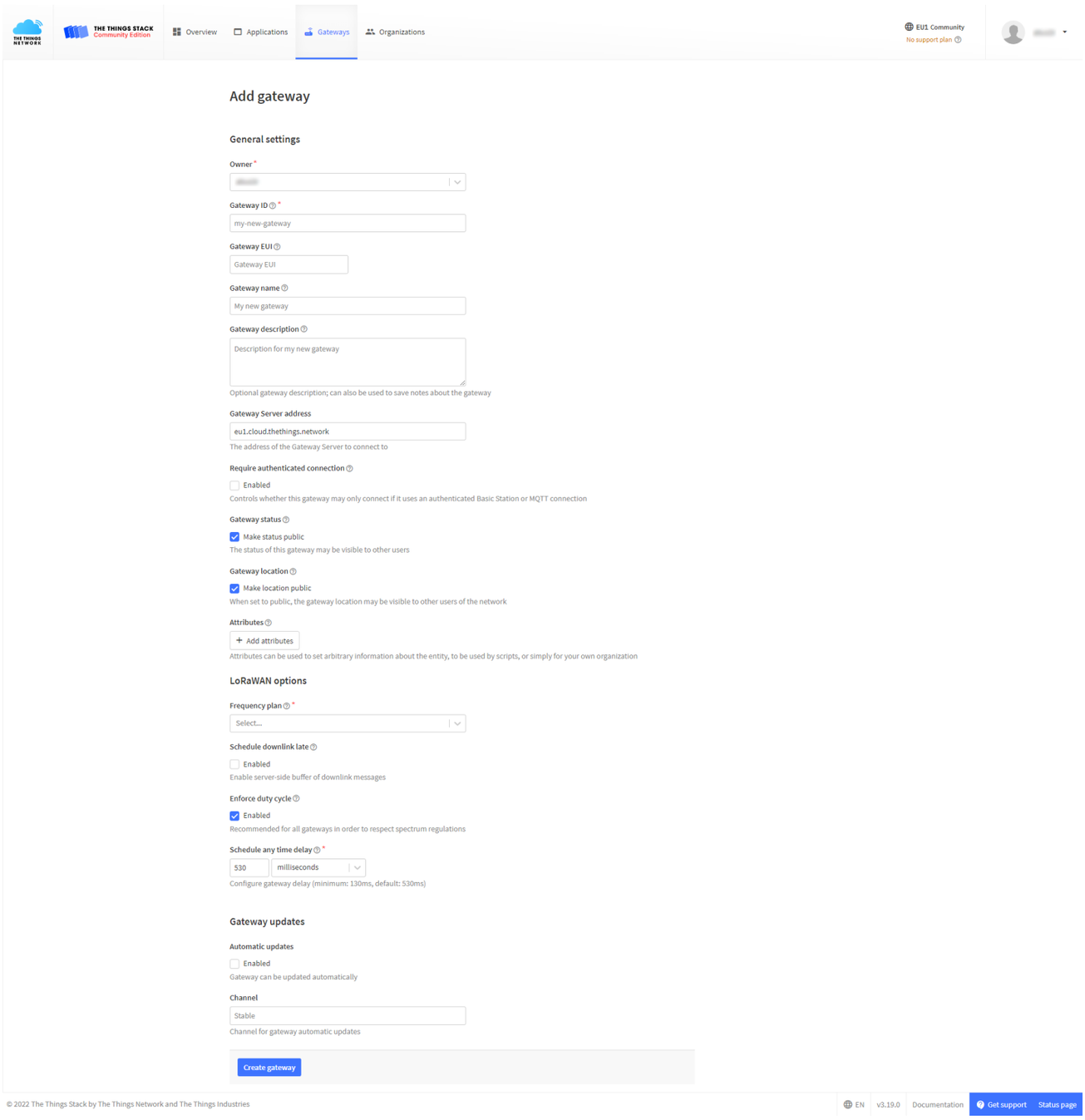

Figure 14: Adding a gateway

3. To register your gateway, scroll down and click **Create gateway**.

TTNv3 supports TLS server authentication and Client token, which requires a trust file and a key file to configure the Gateway to successfully connect it to the network.

Generating the Token

1. To generate a key file, from the **Overview page** of the registered Gateway navigate to **API keys**.

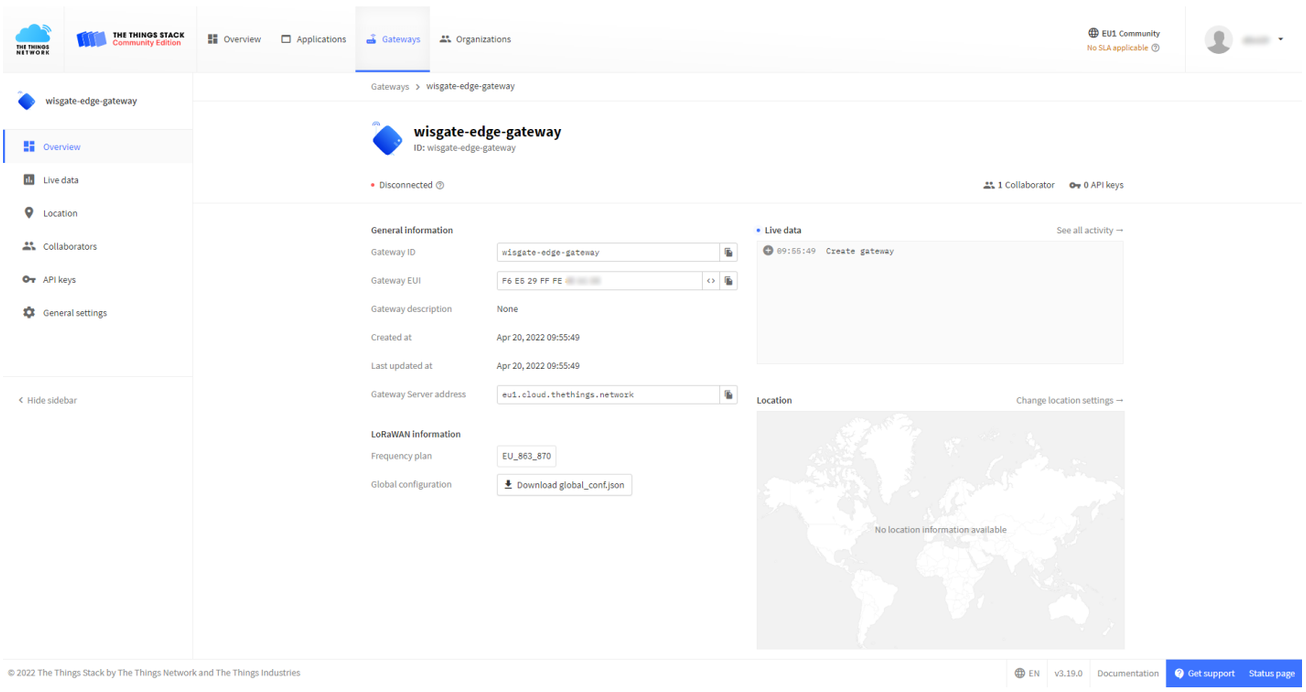

Figure 15: Overview page

2. On the **API keys page**, choose **+ Add API key**.

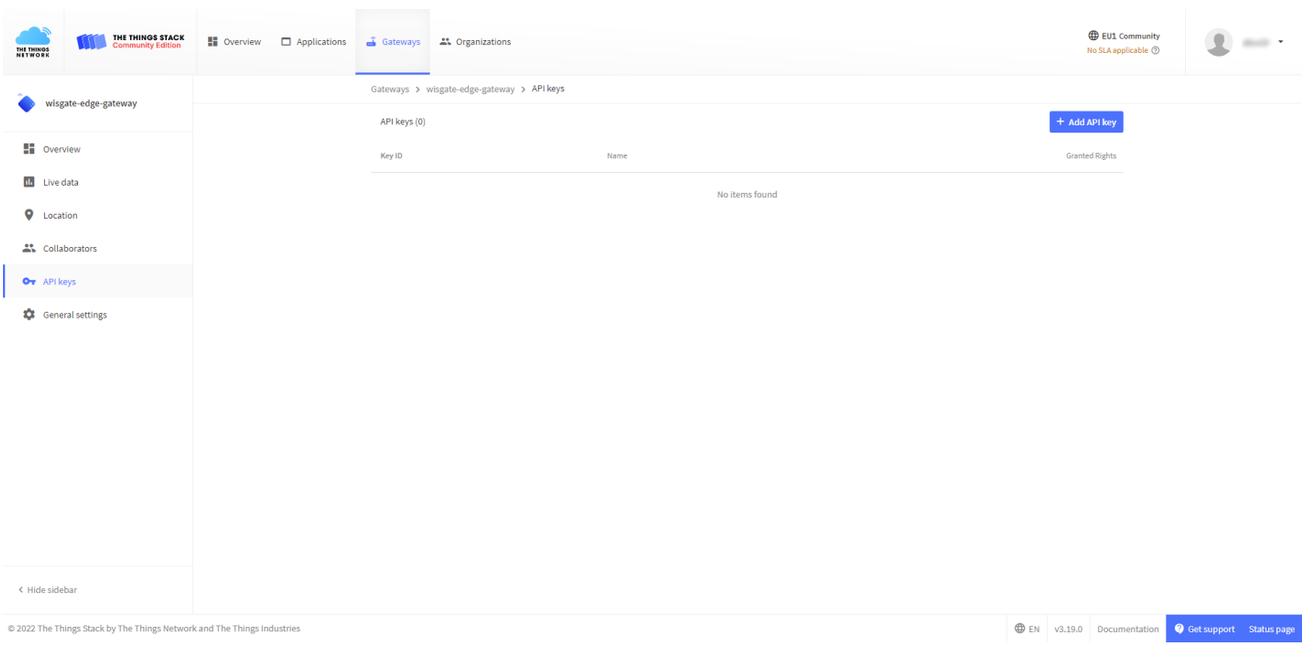

Figure 16: API key page

3. In the **Name field** type the name of your key (for example - mykey). Choose **Grant individual rights** and select **Link as Gateway to a Gateway for traffic exchange, i.e. read uplink and write downlink**.

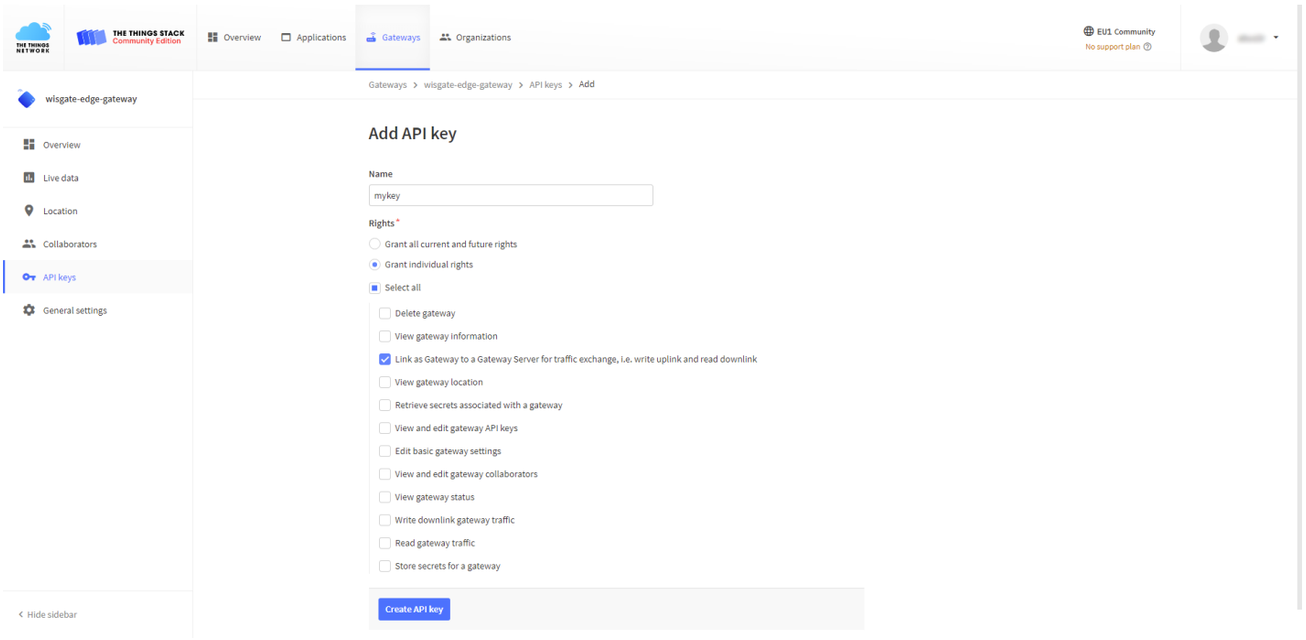

Figure 17: Generating an API key

4. To generate the key, choose **Create API key**. The following window will pop up, telling you to copy the key you just generated.

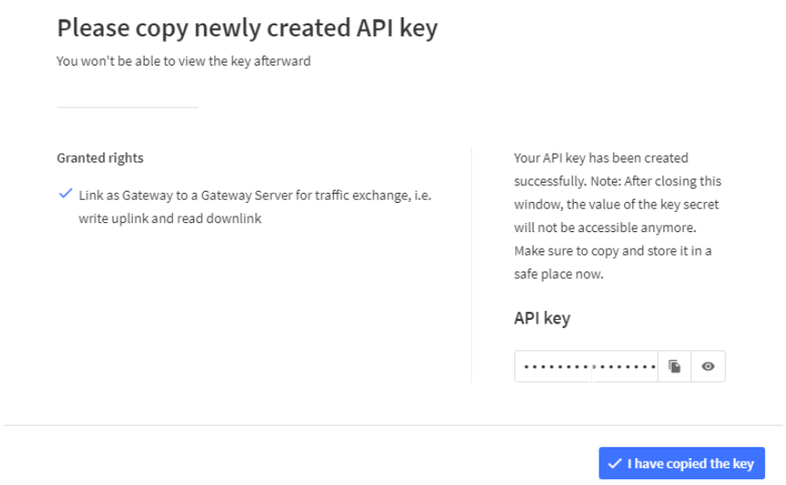

Figure 18: Copying the generated key

> ⚠ **WARNING**
>
> Copy the key and save it in a .txt file (or other), because you won't be able to view or copy your key after that.

5. Click **I have copied the key** to proceed.

Configuring the Gateway

1. To configure the gateway access it via the Web UI. To learn how to do that check out the device's Quickstart Guide mentioned before.

2. Navigate to **LoRa Network** > **Network Settings** > **Mode** drop-down menu > choose **Basics Station**.

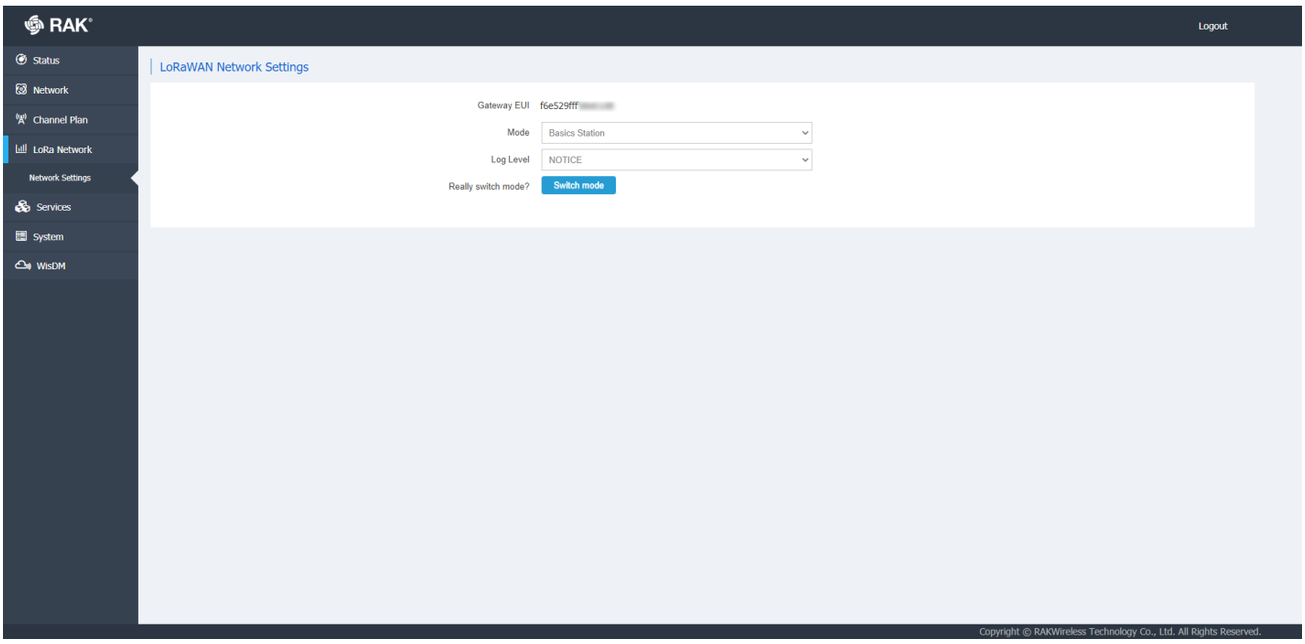

Figure 19: Changing the working mode

3. Select **Switch mode** to apply the change. After that, the **Basics Station Configuration** pane settings will show up. To connect the Gateway to TTNv3, the following parameters must be configured:

- **Server** – For server choose **LNS Server**.
- **URI** – This is the link to The Things Stack server. Note that, for this tutorial, we are connecting the gateway to the European cluster. For Europe fill in the following: wss://eu1.cloud.thethings.network
- **Port** – The LNS Server uses port 8887. Type in **8887**.
- **Authentication Mode** – Choose **TLS server authentication and Client token**. When selected, the trust and the token field will show up.
- **trust** – For trust we will use the **Let's Encrypt ISRG ROOT X1 Trust** certificate. The file with the certificate can be found here .
- **token** - This is the generated **API key**. The key must start with **Authorization:**. Example:

```
Authorization: YOUR_API_KEY
```

> 📝 **NOTE**
>
> Replace **YOUR_API_KEY** with the key generated previously. Have in mind that there should be a "space" between **Authorization:** and **YOUR_API_KEY**, as shown in the example.

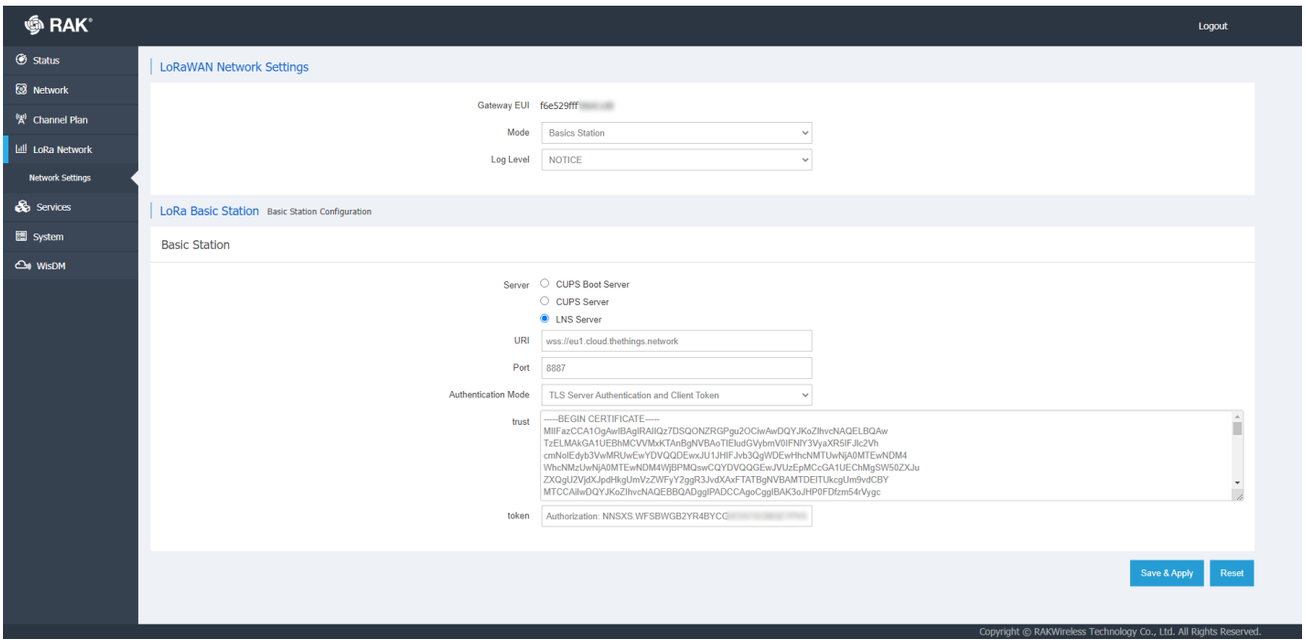

Figure 20: LoRa Basics Station settings

4. To save the changes click **Save & Apply**.

You can now see that your gateway is connected to TTNv3 as Basics Station:

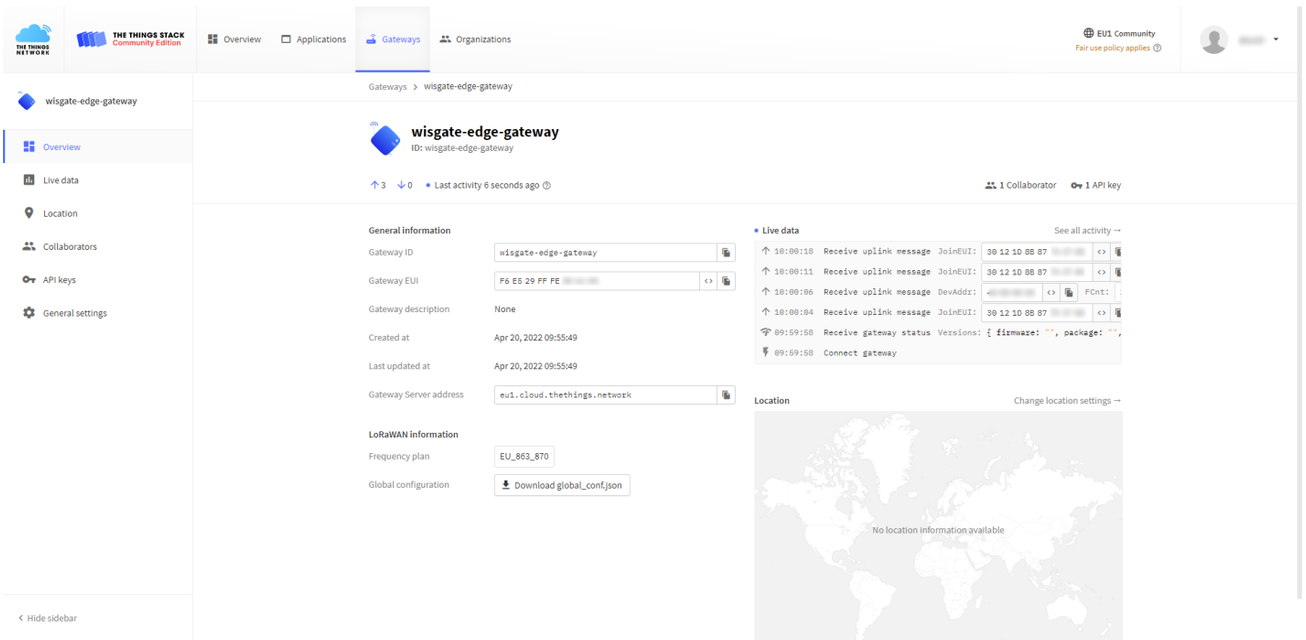

Figure 21: Successful connection

Last Updated: 7/29/2022, 10:17:19 PM