RAK® Documentation Center

RAK7268V2/RAK7268CV2 Supported LoRa Network Servers

AWS IoT Core for LoRaWAN

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

Set up Roles and Policies in IAM

Add an IAM Role for CUPS Server

Adding an IAM role will allow the Configuration and Update Server (CUPS) to handle the wireless gateway credentials.

This procedure needs to be done only once, but must be performed before a LoRaWAN gateway tries to connect with AWS IoT Core for LoRaWAN.

1. Go to the IAM Roles page on the IAM console.
2. Choose **Create role**.
3. On the Create Role page, choose **Another AWS account**.
4. Enter your **Account ID**, then select **Next**.
5. In the search box next to the **Filter Policies**, type ***AWSIoTWirelessGatewayCertManager***.
   - If the search results show the policy named ***AWSIoTWirelessGatewayCertManager***, select it by clicking the checkbox.
   - If the policy does not exist, create one.
     - Choose **Create Policy**, then select the **JSON** tab to open the policy editor.
     - Replace the existing template with a trust policy document.

```json
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "IoTWirelessGatewayCertManager",
"Effect": "Allow",
"Action": [
"iot:CreateKeysAndCertificate",
"iot:DescribeCertificate",
"iot:ListCertificates",
"iot:RegisterCertificate"
         ],
"Resource": "*"
          }
      ]
}
```

     - Select **Next Tags** and click **Next: Review** to open the Review Page.
     - For the **Name**, type ***AWSIoTWirelessGatewayCertManager***.

> 📝 **NOTE:**
>
> You must enter the name as ***AWSIoTWirelessGatewayCertManager*** and must not use a different name. This is for consistency with future releases.

- For the **Description**, enter a description of your choice.
- Then choose **Create policy**. You will see a confirmation message showing the policy has been created.

6. Click **Next**.

7. In **Role** name, enter ***IoTWirelessGatewayCertManagerRole***, and then choose to **Create role**.

> 📝 **NOTE:**
>
> You must not use a different name. This is for consistency with future releases.

8. In the confirmation message, choose ***IoTWirelessGatewayCertManagerRole*** to edit the new role.
9. In the **Summary**, choose the **Trust relationships** tab, and then choose **Edit trust relationship**.
10. In the **Policy Document**, change the **Principal** property to represent the IoT Wireless service:

```json
"Principal": {
"Service": "iotwireless.amazonaws.com"
},
```

- After changing the Principal property, the complete policy document should look like the following:

```json
{
"Version": "2012-10-17",
"Statement": [
    {
"Effect": "Allow",
"Principal": {
"Service": "iotwireless.amazonaws.com"
        },
"Action": "sts:AssumeRole",
"Condition": {}
    }
    ]
}
```

11. Choose **Update Trust Policy** to save your changes and exit. At this point, you have created the ***IoTWirelessGatewayCertManagerRole*** and you won't need to do this again.

> 📝 **NOTE:**
>
> The examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best meet your business and security requirements. For more information, refer to **Example Policies** and **Security Best Practices**

Add IAM Role for Destination to AWS IoT Core for LoRaWAN

Creating a Policy

Creating a policy gives the role permissions to describe the IoT endpoint and publish messages to AWS IoT.

1. Go to the IAM console .
2. Choose **Policies** from the navigation pane.
3. Choose **Create Policy**, then choose the **JSON** tab to open the policy editor. Replace the existing template with this trust policy document:

```json
{
"Version": "2012-10-17",
"Statement": [
   {
"Effect": "Allow",
"Action":
[
"iot:DescribeEndpoint",
"iot:Publish"
],
"Resource": "*"
   }
   ]
}
```

4. Click **Next: Tags** then **Next: Review** to open the Review page.
5. For **Name**, enter a name of your choice.
6. For **Description**, enter a description of your choice.
7. Choose **Create policy**. You will see a confirmation message indicating that the policy has been created.

Creating the Role

1. In the **IAM console**, choose **Roles** from the navigation pane to open the Roles page.
2. Choose **Create Role**.
3. In **Select type of trusted entity**, select **AWS account** and click **Next**.
4. In **Account ID**, enter your AWS account ID, and then choose **Next: Permissions**.
5. Search for the **IAM policy** you just created by entering the policy name in the search bar.
6. In the search results, select the checkbox corresponding to the policy.
7. Click **Next**.
8. For **Role name**, enter an appropriate name of your choice.
9. For **Description**, enter a description of your choice.
10. Choose **Create role**. You will see a confirmation message indicating that your role has been created.

Updating your Trust Policy

Update your role's trust relationship to grant AWS IoT Core for LoRaWAN permission to assume this IAM role when delivering messages from devices to your account.

1. In the IAM console, choose **Roles** from the navigation pane to open the Roles page.
2. Enter the name of the role you created earlier in the search window, and click on the role name in the search results. This opens up the **Summary** page.
3. Choose the **Trust relationships table** to navigate to the Trust relationships page.

4. Click **Edit trust relationship**. The principal AWS role in your trust policy document defaults to root and must be changed. Replace the existing policy with this:

```json
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "",
"Effect": "Allow",
"Principal": {
"Service": "iotwireless.amazonaws.com"
},
"Action": "sts:AssumeRole",
"Condition": {}
   }
   ]
}
```

5. Choose **Update Trust Policy.**

Add the Gateway to AWS IoT

Requirements

To complete setting up your gateway, you need the following:

- LoRaWAN region. For example, if the gateway is deployed in a US region, the gateway must support LoRaWAN region US915.
- Gateway LNS-protocols. Currently, the LoRa Basics Station protocol is supported.
- Gateway ID (GatewayEUI) or serial number. This is used to establish the connection between the LNS and the gateway. Consult the documentation for your gateway to locate this value.
- Add minimum software versions required, including Basics Station 2.0.5.

Add the LoRaWAN Gateway

To register the gateway with AWS IoT Core for LoRaWAN, execute these steps:

1. Go to the AWS IoT console .
2. Select **Wireless connectivity** in the navigation panel on the left.
3. Choose **Gateways**, and then click **Add Gateway**.
4. In the **Add gateway section**, fill in the **GatewayEUI** and **Frequency band (RF Region)** fields.
5. Enter a descriptive name in the **Name** – optional field. It is recommended that you use the GatewayEUI as the name.
6. Choose **Add gateway**.
7. On the **Configure your Gateway** page, find the section titled **Gateway certificate**.
8. Select **Create certificate**.
9. Once the **Certificate created and associated with your gateway** message is shown, select **Download certificates** to download the certificate (*xxxxx.cert.pem*) and private key (*xxxxxx.private.key*).
10. In the section **Provisioning credentials**, choose **Download server trust certificates** to download the **CUPS (cups.trust)** and **LNS (lns.trust)** server trust certificates.
11. Copy the CUPS and LNS endpoints and save them in a `.txt` file for use while configuring the gateway.
12. Choose **Submit** to add the gateway.

Add a LoRaWAN Device to AWS IoT

Requirements:

- Locate and note the following specifications about your endpoint device.
    - **LoRaWAN Region** - This must match the gateway LoRaWAN region. The following Frequency bands (RF regions) are supported: o EU868 o US915 o EU433
    - **MAC Version** - This must be one of the following: o V1.0.2 o v1.0.3 o v1.1
    - OTAA v1.0x and OTAA v1.1 are supported.
    - ABP v1.0x and ABP v1.1 are supported.
- Locate and note the following information from your device manufacturer:
    - For OTAA v1.0x devices: DevEUI, AppKey, AppEUI
    - For OTAA v1.1 devices: DevEUI, AppKey, NwkKey, JoinEUI
    - For ABP v1.0x devices: DevEUI, DevAddr, NwkSkey, AppSkey
    - For ABP v1.1 devices: DevEUI, DevAddr, NwkSEnckey, FNwkSIntKey, SNwkSIntKey, AppSKey

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.

1. Navigate to the AWS IoT console . In the navigation pane, choose **Wireless connectivity** then click **Profiles**.
2. In the **Device Profiles** section, there are some pre-defined profiles listed.
3. Check each of the profiles to determine if one of them will work for you. If not, select **Add device profile** and set up the parameters as needed. For US 915 as an example, the values are:
    - MacVersion 1.0.3
    - RegParamsRevision RP002-1.0.1
    - MaxEirp 10
    - MaxDutyCycle 10
    - RfRegion US915
    - SupportsJoin true
4. Click **Add device profile** once you have set a device profile that will work for you.
5. In the **Service Profiles** section, click **Add service profile** and set up the parameters as needed. As an example, the default service profile parameters are shown below. However, only the ***AddGwMetadata*** setting can be changed at this time.
    - UlRate 60
    - UlBucketSize 4096
    - DlRate 60
    - DlBucketSize 4096
    - AddGwMetadata true
    - DevStatusReqFreq 24
    - DrMax 15
    - TargetPer 5

- MinGwDiversity 1

6. 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 on Destinations, refer to the AWS LoRaWAN Developer Guide .

A destination consists of a Rule and a Role. To set up the destination, execute the following steps:

1. Navigate to the AWS IoT console . In the navigation pane, choose **Wireless connectivity**, and then **Destinations**.
2. Choose **Add Destination**.
3. For the **Destination name**, enter ***ProcessLoRa***, and then add an appropriate description under **Destination description – optional**.
4. For **Rule name**, enter ***LoRaWANRouting***. Ignore the section **Rules configuration – Optional** for now. The Rule will be set up later in the "Hello World" sample application. See Create the IoT Rule for the destination.
5. In the **Permissions** section, choose **Select an existing service role** and select the IAM role you had created earlier, from the drop-down.

> 📝 **NOTE:**
>
> The Destination name can be anything. For getting started and consistency, choose **ProcessLoRa** for the first integration with AWS IoT Core for LoRaWAN.

5. Choose **Add Destination**. You will see a message "*Destination added*", indicating the destination has been successfully added.

Register the Device

Now, register an endpoint device with AWS IoT Core for LoRaWAN as follows:

1. Go to the AWS IoT console .
2. Select **Wireless connectivity** in the navigation panel on the left.
3. Select **Devices**, then choose **Add wireless device**.
4. On the **Add device** page, select the LoRaWAN specification version in the drop-down under **Wireless device specification**.
5. Under **LoRaWAN specification and wireless device configuration**, enter the **DevEUI** and confirm it in the **Confirm DevEUI** field.
6. Enter the remaining fields as per the OTAA/ABP choice you made above.
7. Enter a name for your device in the **Wireless device name – optional field**.
8. In the **Profiles** section, under **Wireless device profile**, from the drop-down option find the device profile you have created or the one that corresponds to your device and region.

> 📝 **NOTE:**
>
> Compare your device details to ensure the device profile is correct. If there are no valid default options, you will have to create a new profile. See the Verify Profiles section.

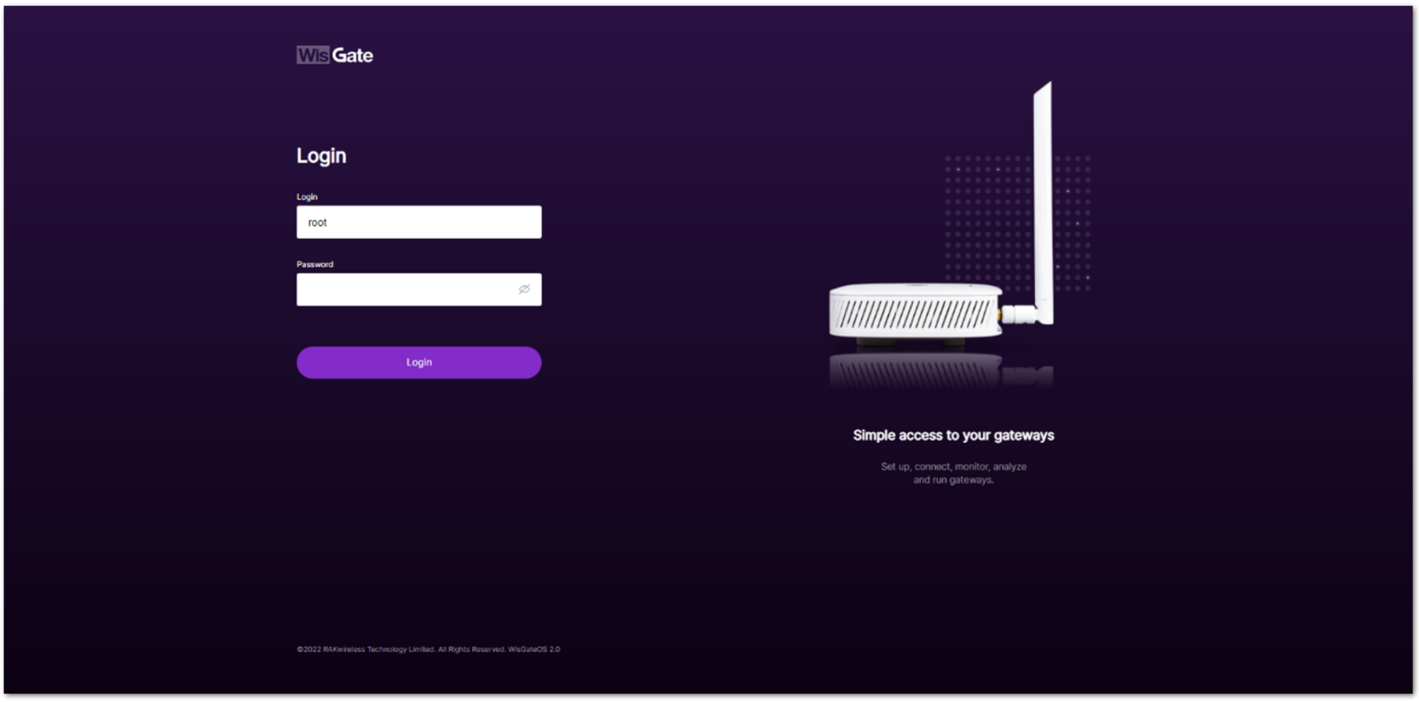
9. Click **Next** and then choose the destination you created earlier (*ProcessLoRa*) from the drop-down under **Choose destination**.
10. Choose **Add device**. You will see a message saying "*Wireless device added*", indicating that your device has been set up successfully.

Set up the Gateway

- **Set up the Gateway Hardware**
- **Set up the Gateway Software**

Configure the Gateway Device

1. Using your preferred Web browser, access the gateway. To access the gateway, see the Quick Start guide.



WisGate

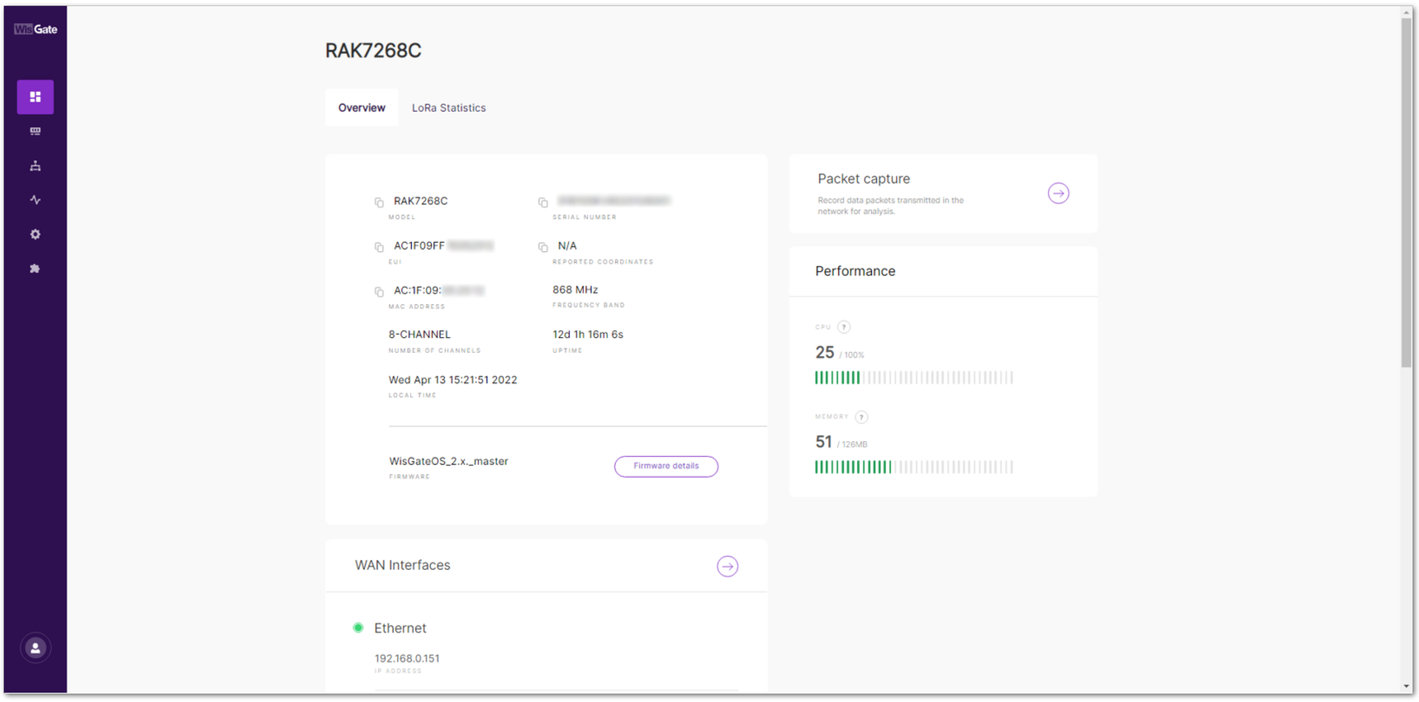
Login

Login: root

Password:

Login
Simple access to your gateways
Set up, connect, monitor, analyze and run gateways.
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 1: Web User Interface Log-in



RAK7268C

Overview | LoRa Statistics

RAK7268C — MODEL
[redacted] — SERIAL NUMBER
AC1F09FF[redacted] — EUI
N/A — REPORTED COORDINATES
AC:1F:09:[redacted] — MAC ADDRESS
868 MHz — FREQUENCY BAND
8-CHANNEL — NUMBER OF CHANNELS
12d 1h 16m 6s — UPTIME
Wed Apr 13 15:21:51 2022 — LOCAL TIME

WisGateOS_2.x._master — FIRMWARE

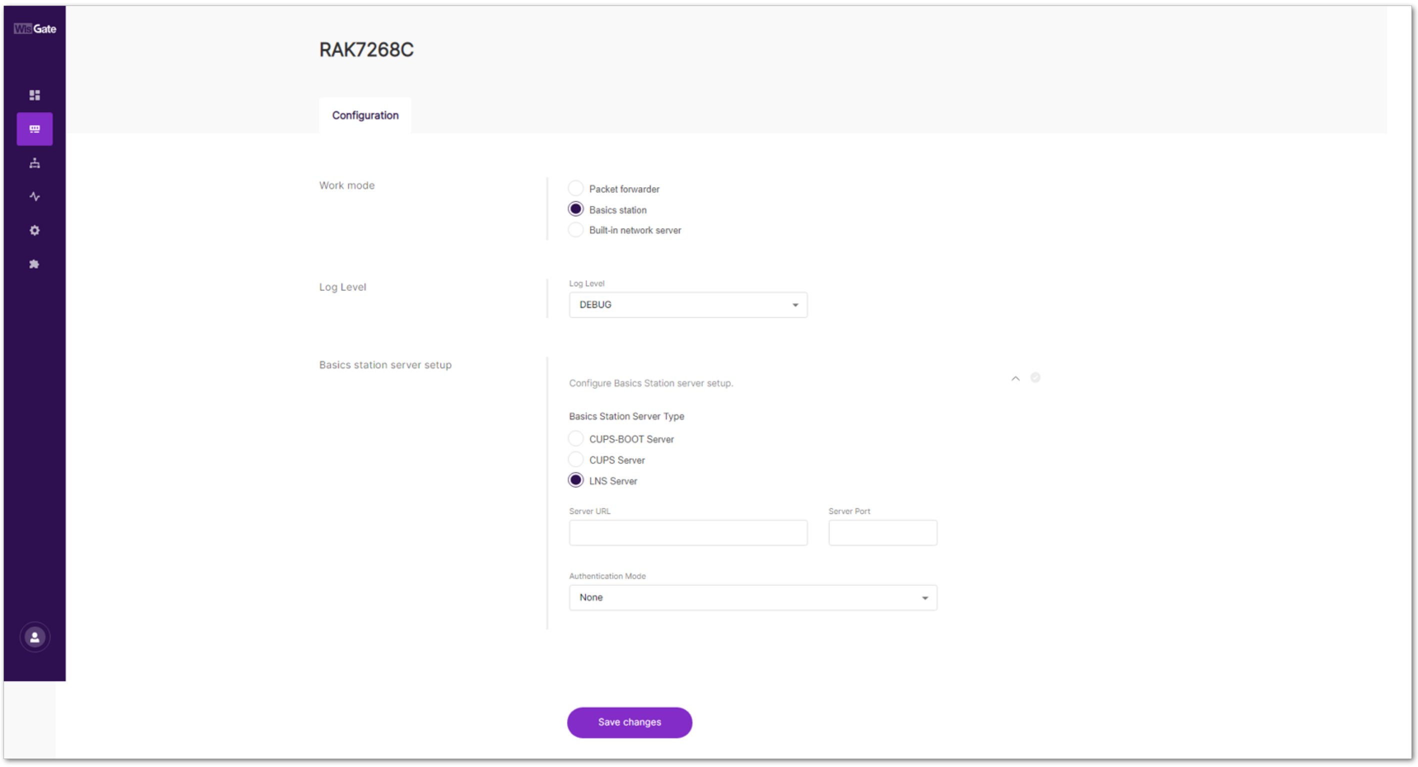
Packet capture
Record data packets transmitted in the network for analysis.

Performance
CPU 25 / 100%
MEMORY 51 / 126MB

WAN Interfaces
Ethernet
192.168.0.151 — IP ADDRESS

Figure 2: Checking the Firmware Version

2. Configure **Network Mode** to **Basics Station**.
3. Navigate to **LoRa**. For **Work mode**, select **Basics station** and click **Configure Basics Station server setup** to expand the Basics Station settings.



RAK7268C

Configuration

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Basics station server setup

Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

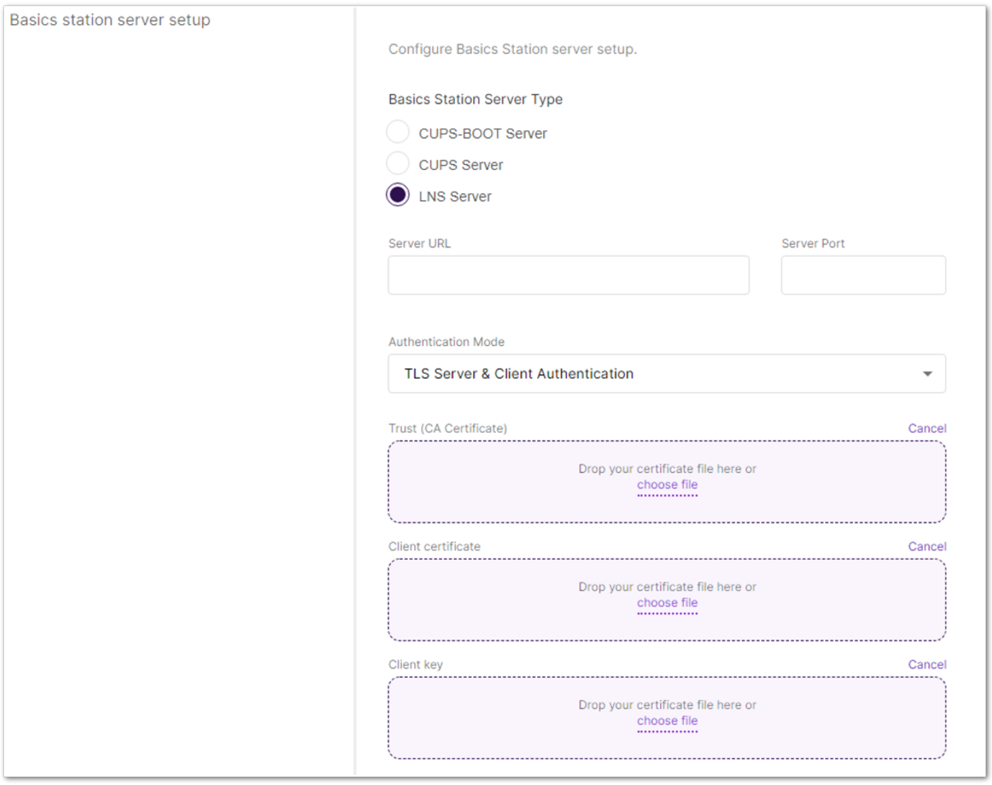
Server URL

Server Port

Authentication Mode: None

Figure 3: Basics Station work mode

4. Select **LNS Server** from Server, then choose **TLS Server and Client Authentication** from Authentication Mode.


Cancel
Drop your certificate file here or
choose file
Cancel
Drop your certificate file here or
choose file
Cancel
Drop your certificate file here or
choose file

Basics station server setup

Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

Server URL

Server Port

Authentication Mode: TLS Server & Client Authentication

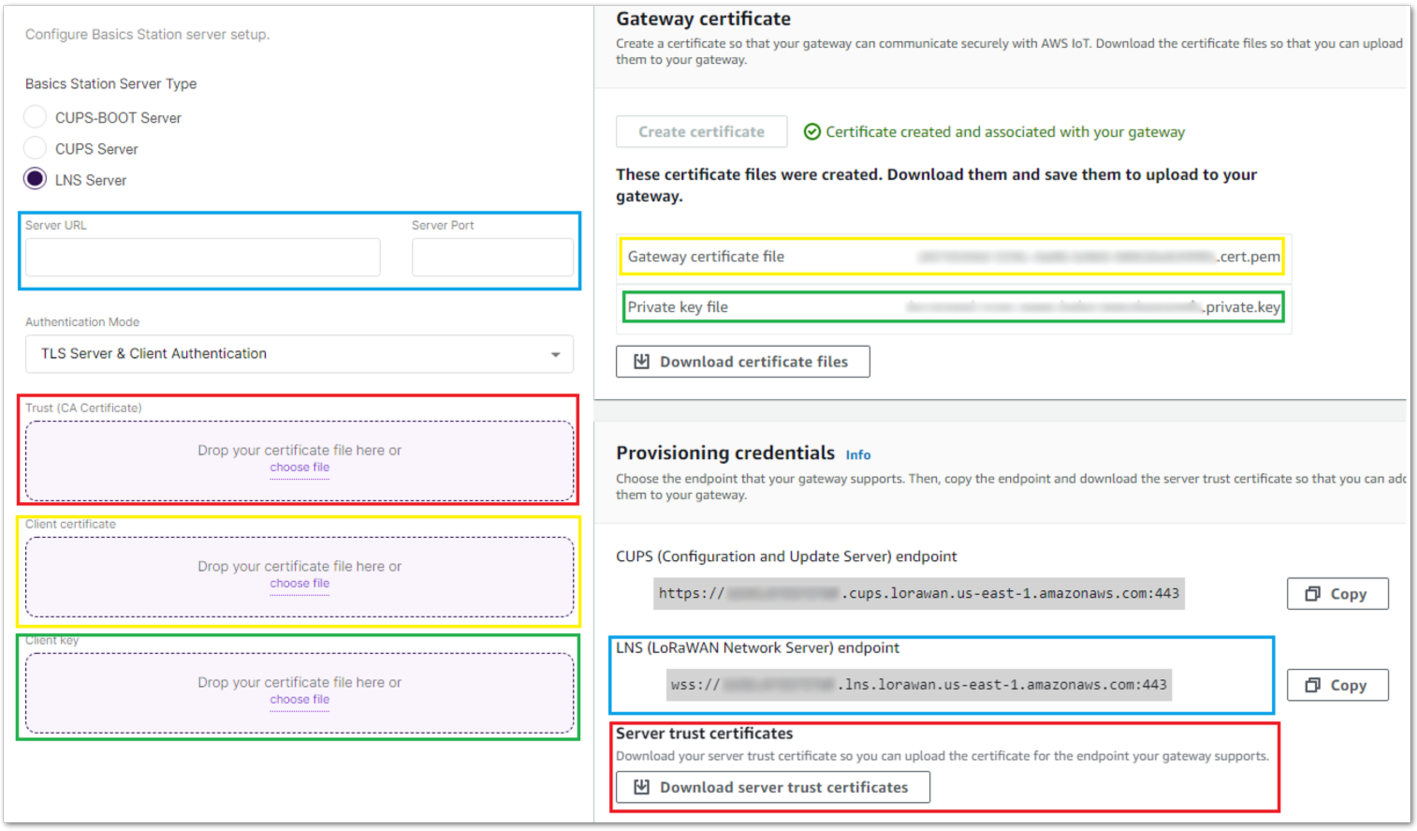
Trust (CA Certificate)

Client certificate

Client key

Figure 4: Configuring Network Mode to Basics Station

5. Configure URI, Port, and Authentication Mode.



Configure Basics Station server setup.

Basics Station Server Type

○ CUPS-BOOT Server

○ CUPS Server

● LNS Server

Server URL

Server Port

Authentication Mode

TLS Server & Client Authentication

Trust (CA Certificate)

Drop your certificate file here or choose file

Client certificate

Drop your certificate file here or choose file

Client key

Drop your certificate file here or choose file

Gateway certificate

Create a certificate so that your gateway can communicate securely with AWS IoT. Download the certificate files so that you can upload them to your gateway.

Create certificate ✓ Certificate created and associated with your gateway

These certificate files were created. Download them and save them to upload to your gateway.

Gateway certificate file	[redacted].cert.pem
Private key file	[redacted].private.key

Download certificate files

Provisioning credentials Info

Choose the endpoint that your gateway supports. Then, copy the endpoint and download the server trust certificate so that you can adc them to your gateway.

CUPS (Configuration and Update Server) endpoint

```
https://[redacted].cups.lorawan.us-east-1.amazonaws.com:443
```

Copy

LNS (LoRaWAN Network Server) endpoint

```
wss://[redacted].lns.lorawan.us-east-1.amazonaws.com:443
```

Copy

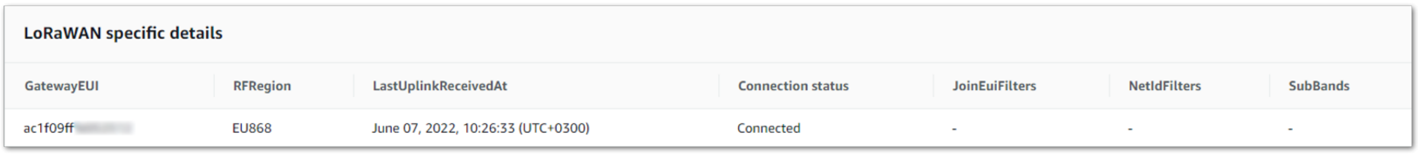
Server trust certificates

Download your server trust certificate so you can upload the certificate for the endpoint your gateway supports.

Download server trust certificates

Figure 5: Configuring URI, Port, and Authentication Mode

6. Click **Save**. Check if the gateway is online in AWS IoT console.



LoRaWAN specific details

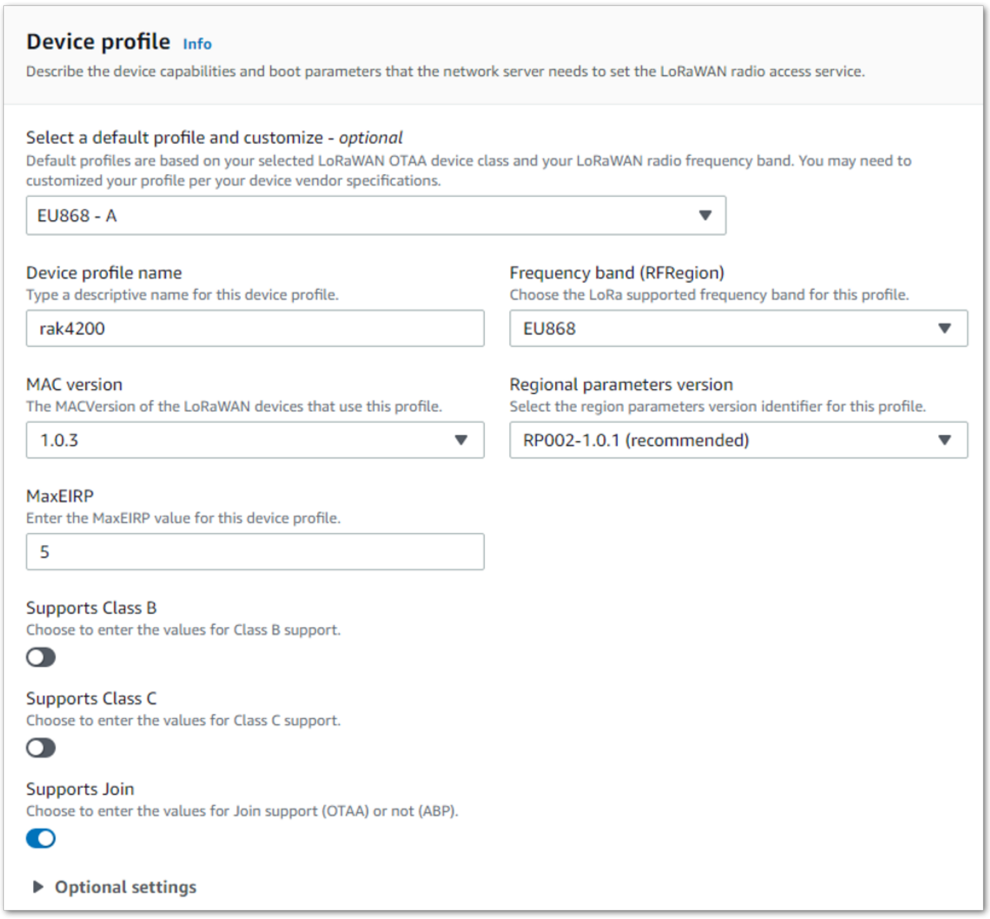
GatewayEUI	RFRegion	LastUplinkReceivedAt	Connection status	JoinEuiFilters	NetIdFilters	SubBands
ac1f09ff[redacted]	EU868	June 07, 2022, 10:26:33 (UTC+0300)	Connected	-	-	-

Figure 6: Verifying Operation

Add End Devices

This section shows an example of how to join the AWS IoT LoRaWAN server.

1. Add Device Profile.



Device profile Info

Describe the device capabilities and boot parameters that the network server needs to set the LoRaWAN radio access service.

Select a default profile and customize - *optional*
Default profiles are based on your selected LoRaWAN OTAA device class and your LoRaWAN radio frequency band. You may need to customized your profile per your device vendor specifications.
EU868 - A

Device profile name
Type a descriptive name for this device profile.
rak4200

Frequency band (RFRegion)
Choose the LoRa supported frequency band for this profile.
EU868

MAC version
The MACVersion of the LoRaWAN devices that use this profile.
1.0.3

Regional parameters version
Select the region parameters version identifier for this profile.
RP002-1.0.1 (recommended)

MaxEIRP
Enter the MaxEIRP value for this device profile.
5

Supports Class B
Choose to enter the values for Class B support.

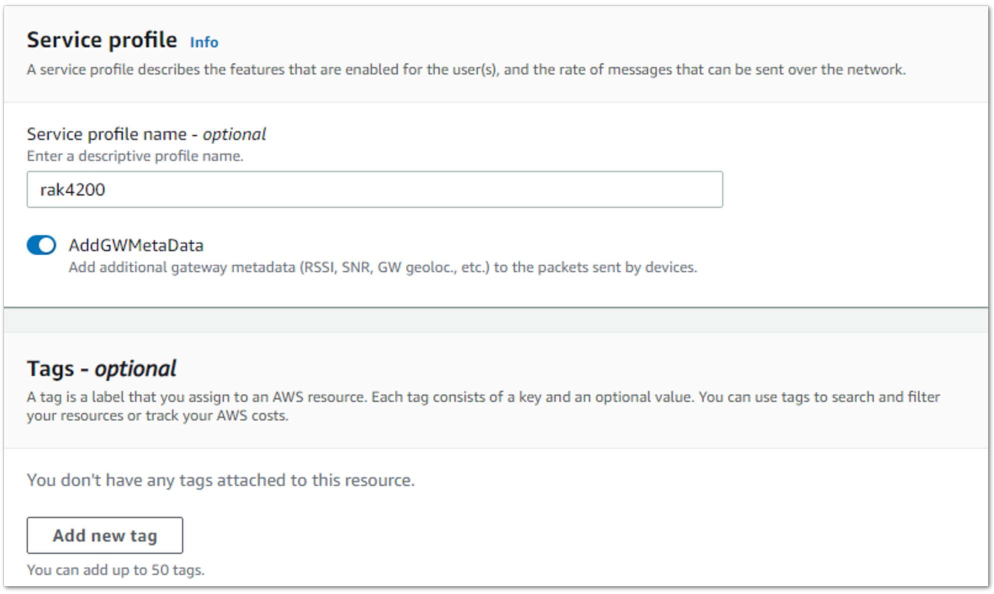
Supports Class C
Choose to enter the values for Class C support.

Supports Join
Choose to enter the values for Join support (OTAA) or not (ABP).

▶ Optional settings

Figure 7: Adding the Device Profile

2. Add Service Profile.



Service profile Info

A service profile describes the features that are enabled for the user(s), and the rate of messages that can be sent over the network.

Service profile name - *optional*
Enter a descriptive profile name.
rak4200

AddGWMetaData
Add additional gateway metadata (RSSI, SNR, GW geoloc., etc.) to the packets sent by devices.

Tags - *optional*

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

You don't have any tags attached to this resource.

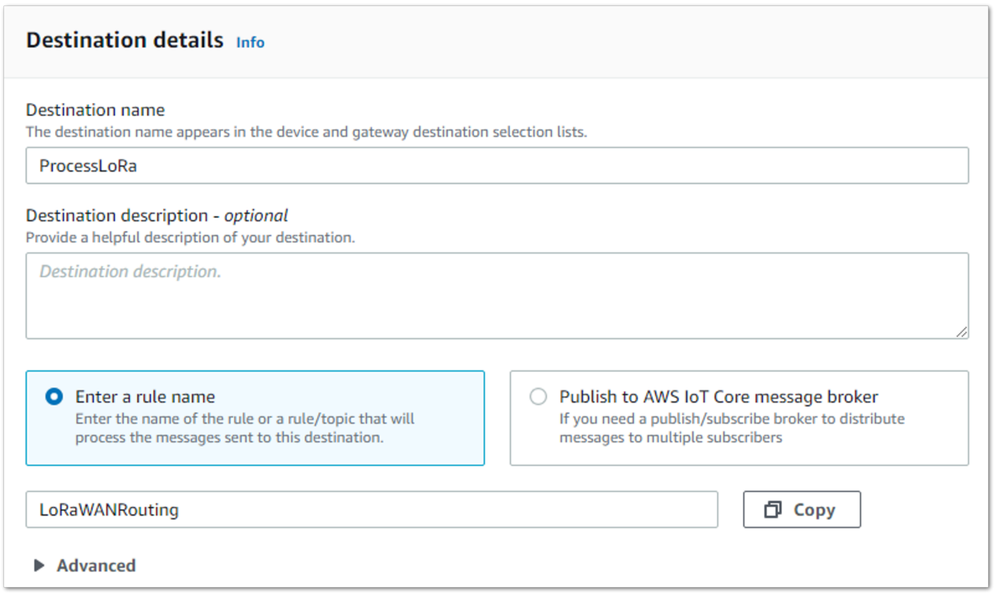
Add new tag

You can add up to 50 tags.

Figure 8: Adding the Service Profile

3. Add Destination.

Before adding the destination, follow the Add IAM role for Destination to AWS IoT Core for LoRaWAN section to configure the IAM policy and role.



Destination details Info

Destination name
The destination name appears in the device and gateway destination selection lists.
ProcessLoRa

Destination description - *optional*
Provide a helpful description of your destination.
Destination description.

● Enter a rule name
Enter the name of the rule or a rule/topic that will process the messages sent to this destination.

○ Publish to AWS IoT Core message broker
If you need a publish/subscribe broker to distribute messages to multiple subscribers

LoRaWANRouting

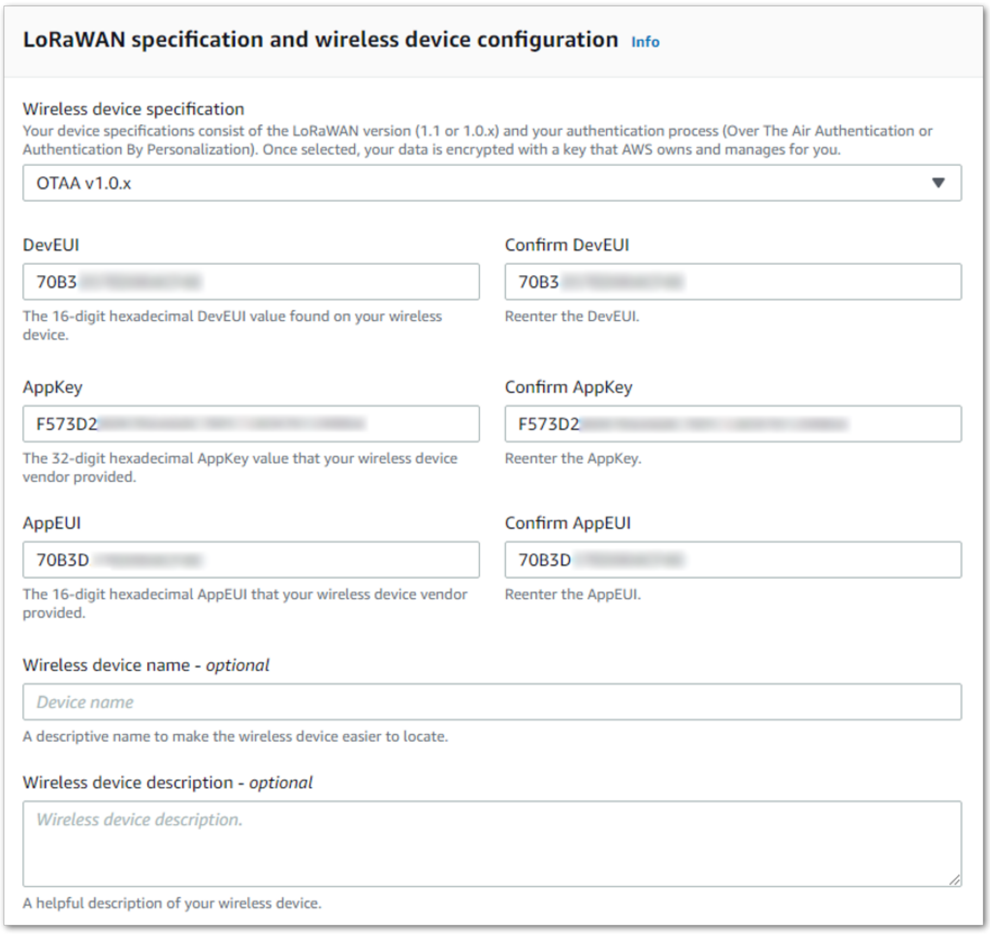
Figure 9: Adding Destination

4. Add Device.

Before adding a device to AWS IoT, retrieve the **DevEui**, **AppEui**, and **AppKey** from the end Device's console. You can use AT command `at+get_config=lora:status` to obtain the information.

For more AT commands, refer to the RAK4200 AT Command Manual.

```
at+get_config=lora:status\r\n
OK Work Mode: LoRaWAN
Region: EU868
Send_interval: 600s
Auto send status: false.
MulticastEnable: true.
Multi_Dev_Addr: 260XXXXX
Multi_Apps_Key: F13DDFA2619B10411F02XXXXXXXXXXXX
Multi_Nwks_Key: 1D1991F5377C675879C3XXXXXXXXXXXX
Join_mode: OTAA
DevEui: 70B3XXXXXXXXXXXX
AppEui: F573D2XXXXXXXXXX
AppKey: 70B3DXXXXXXXXXXXXXXXXXXXXXXXXXXX
Class: C
Joined Network:false
IsConfirm: unconfirm
AdrEnable: true
EnableRepeaterSupport: false
RX2_CHANNEL_FREQUENCY: 869525000, RX2_CHANNEL_DR:0
RX_WINDOW_DURATION: 3000ms
RECEIVE_DELAY_1: 1000ms
RECEIVE_DELAY_2: 2000ms
JOIN_ACCEPT_DELAY_1: 5000ms
JOIN_ACCEPT_DELAY_2: 6000ms
Current Datarate: 4
Primeval Datarate: 4
ChannelsTxPower: 0
UpLinkCounter: 0
DownLinkCounter: 0
```



LoRaWAN specification and wireless device configuration

Wireless device specification
Your device specifications consist of the LoRaWAN version (1.1 or 1.0.x) and your authentication process (Over The Air Authentication or Authentication By Personalization). Once selected, your data is encrypted with a key that AWS owns and manages for you.

OTAA v1.0.x

DevEUI: 70B3[redacted]
The 16-digit hexadecimal DevEUI value found on your wireless device.

Confirm DevEUI: 70B3[redacted]
Reenter the DevEUI.

AppKey: F573D2[redacted]
The 32-digit hexadecimal AppKey value that your wireless device vendor provided.

Confirm AppKey: F573D2[redacted]
Reenter the AppKey.

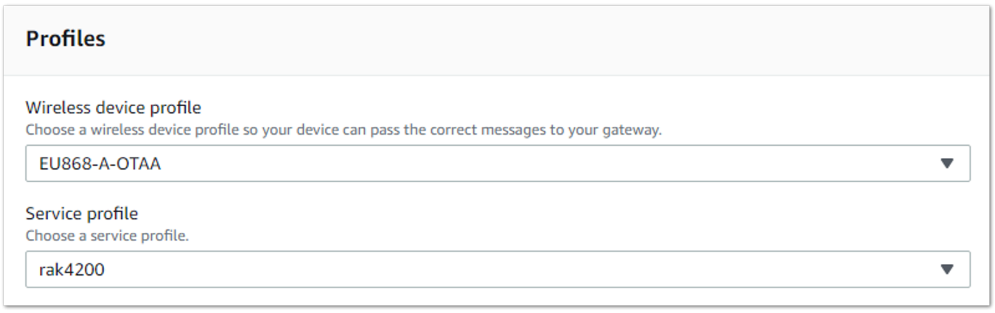
AppEUI: 70B3D[redacted]
The 16-digit hexadecimal AppEUI that your wireless device vendor provided.

Confirm AppEUI: 70B3D[redacted]
Reenter the AppEUI.

Wireless device name - *optional*
Device name
A descriptive name to make the wireless device easier to locate.

Wireless device description - *optional*
Wireless device description.
A helpful description of your wireless device.

Figure 10: LoRaWAN specifications and wireless device configuration

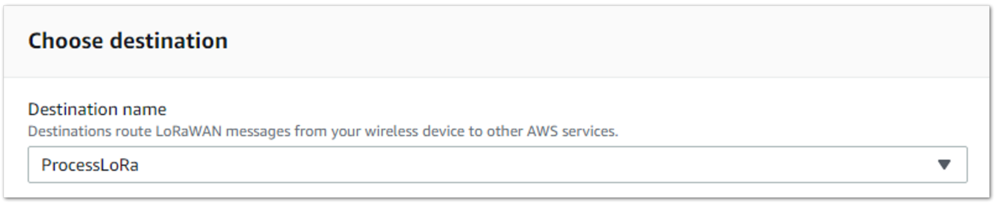


Profiles

Wireless device profile
Choose a wireless device profile so your device can pass the correct messages to your gateway.
EU868-A-OTAA

Service profile
Choose a service profile.
rak4200

Figure 11: Choosing a Wireless Device Profile



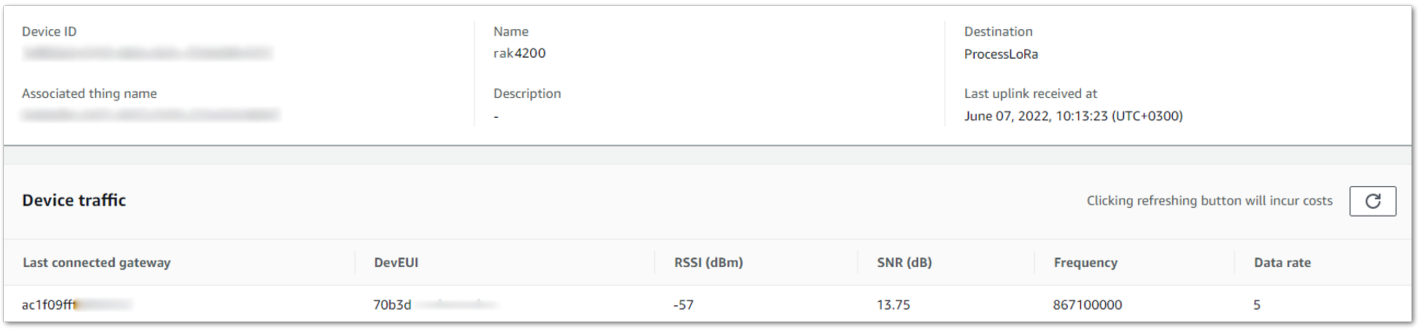
Choose destination

Destination name
Destinations route LoRaWAN messages from your wireless device to other AWS services.
ProcessLoRa

Figure 12: Choosing a Destination

5. Restart the end Device, and it should join the AWS IoT LoRaWAN server.

```
EVENT:0:STARTUP
SYSLOG:4:OTAA Join Request
SYSLOG:4:OTAA Join Success
EVENT:1:JOIN_NETWORK
SYSLOG:4:LoRa Tx :
```



Device ID	Name	Destination
[redacted]	rak4200	ProcessLoRa
Associated thing name	Description	Last uplink received at
[redacted]	-	June 07, 2022, 10:13:23 (UTC+0300)

Device traffic

Clicking refreshing button will incur costs

Last connected gateway	DevEUI	RSSI (dBm)	SNR (dB)	Frequency	Data rate
ac1f09ff[redacted]	70b3d[redacted]	-57	13.75	867100000	5

Figure 13: Joined device

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.

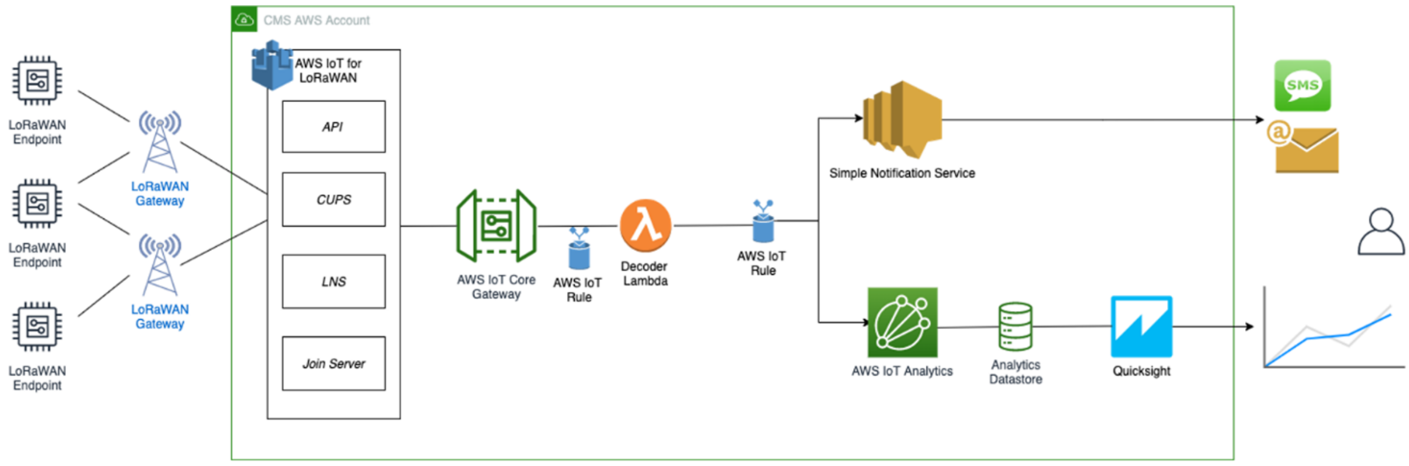


Figure 14: Sending Uplink Architecture

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. Click **Functions** in the navigation pane.
3. Click **Create function**.
4. Select **Author** from scratch.
5. Under **Basic Information**, enter the function name and choose ***Runtime Python 3.8***. from the drop-down under **Runtime**.
6. Click **Create function**.
7. 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')
                              + '/project/sensor/decoded', qos=0,
                              payload=json.dumps(data))
    return response
```

8. Once the code has been pasted, choose **Deploy** to deploy the lambda code.
9. Click the **Configuration** tab of the lambda function and click **Permissions**.
10. Change the **Lambda Role Policy** permission.
    - Under **Execution role**, click 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":[
      "*"
   ]
}
```

- After the change the code should look like this:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect":"Allow",
        "Action":[
             "iot:Publish"
        ],
         "Resource":[
          "*"
         ]
        }
    ]
}
```

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

11. Go back to the Lambda function **Code source** and create a test event that will allow you to test the functionality of the lambda function.

- Click **Test** next to **Deploy**.
- In the **Configure test event**, enter a name for the test event in the **Event name** field.
- Paste the following sample payload in the area under **Event JSON** field:

```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 **Save** to save the event.
12. In a new window, navigate to the AWS IoT console, choose **Test** on the navigation pane, and select MQTT test client.
13. In Subscription topic field type “#” (all topics) and click Subscribe to topic. The MQTT will 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. The output should look similar to this:

```json
000000000000000088/project/sensor/decoded      February 09, 2021, 14:45:29 (UTC+0800)
    {
      "devEui": "000000000000000088",
      "fPort": 1,
      "freq": "868100000",
      "timestamp": "2021-02-08T04:00:40Z",
      "temperature": -4.1
    }
```

Create the Destination Rule

In this section, 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:lora:1:01670110`.
- You should see traffic similar to that 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": "3eff83dd-9159-XXXX-XXXX-XXXXXXXXXXXX",
     "PayloadData": "AWcBEA==",
     "WirelessMetadata": {
         "LoRaWAN": {
             "DataRate": "4",
             "DevEui": "3933XXXXXXXXXXXX ",
             "FPort": 1,
             "Frequency": "867100000",
             "Gateways": [
                {
                     "GatewayEui": "ac1ff09ffXXXXXXXX",
                     "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. Select **Text Messaging** (SMS) and choose **Publish text message**.
3. Under **Message type**, select **Promotional**.
4. Enter your phone number (phone number that will receive text alerts).
5. Enter "*Test message*" for the **Message** and choose **Publish** message.
6. If the phone number you entered is valid, you will receive a text message and your phone number will be confirmed.
7. 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*".
   - Select **Create topic**.
8. Create a subscription for this topic:
   - On the newly created *text_topic* page, 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, choose **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, timest
```

6. Under **Set one or more actions**, 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 **Create role**.
13. Choose **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.

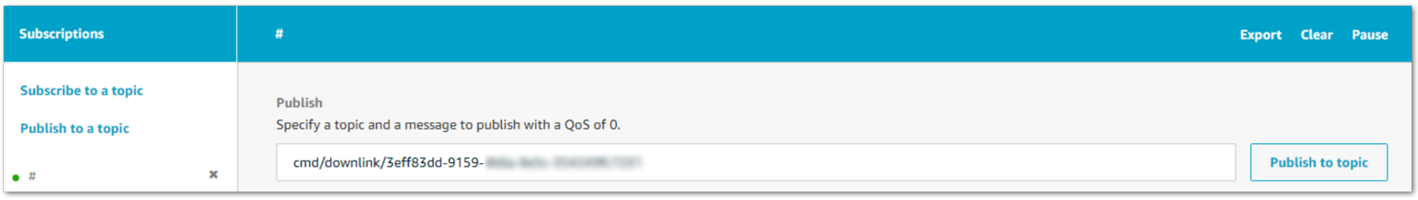
Wait for an uplink from the device. Here is the message from mobile after sending an uplink message.

```json
{
    "device_id": "3933XXXXXXXXXXXX",
    "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 the end Device.

1. Install the AWS SAM CLI .
2. Deploy SAM template to AWS .
3. Send Payload to End Device.
   - 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.
   - Specify the topic to ***cmd/downlink/{WirelessDeviceId}*** and a base64-encoded message.



Publish

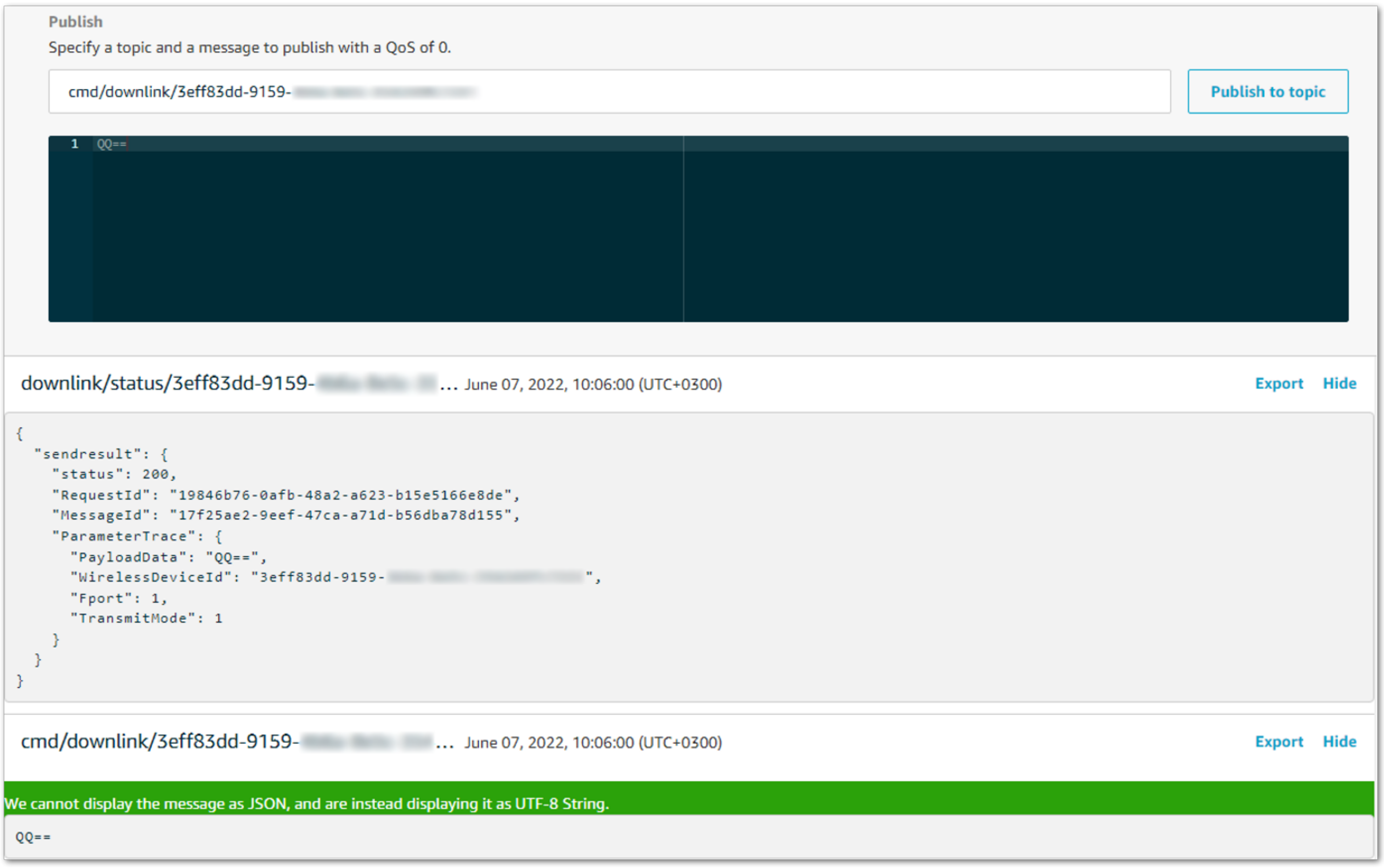
Specify a topic and a message to publish with a QoS of 0.

cmd/downlink/3eff83dd-9159-[redacted]

Figure 15: Specifying a topic

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

```json
downlink/status/3eff83dd-9159-XXXX-XXXX-XXXXXXXXXXXX  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": "3eff83dd-9159-XXXX-XXXX-XXXXXXXXXXXX",
            "Fport": 1,
            "TransmitMode": 1
        }
    }
}
```



cmd/downlink/3eff83dd-9159-[redacted]

```
QQ==
```

downlink/status/3eff83dd-9159-[redacted]… June 07, 2022, 10:06:00 (UTC+0300)

```
{
  "sendresult": {
    "status": 200,
    "RequestId": "19846b76-0afb-48a2-a623-b15e5166e8de",
    "MessageId": "17f25ae2-9eef-47ca-a71d-b56dba78d155",
    "ParameterTrace": {
      "PayloadData": "QQ==",
      "WirelessDeviceId": "3eff83dd-9159-[redacted]",
      "Fport": 1,
      "TransmitMode": 1
    }
  }
}
```

cmd/downlink/3eff83dd-9159-[redacted]… June 07, 2022, 10:06:00 (UTC+0300)

We cannot display the message as JSON, and are instead displaying it as UTF-8 String.

```
QQ==
```

Figure 16: 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*.
11. Choose **Quick Create**.
12. Once the Quick Create Finished message is displayed, choose **Add action**.
13. 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 **Datasets**.
3. Select the data set generated by the Quick Create in Create an IoT Analytics Rule
4. In the Details section, click **Edit** in 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. Click **Update.
7. Navigate to **Schedule**, and click **Edit**.
8. Under Frequency, choose **Every 1 minute**, and then click **Update**.

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

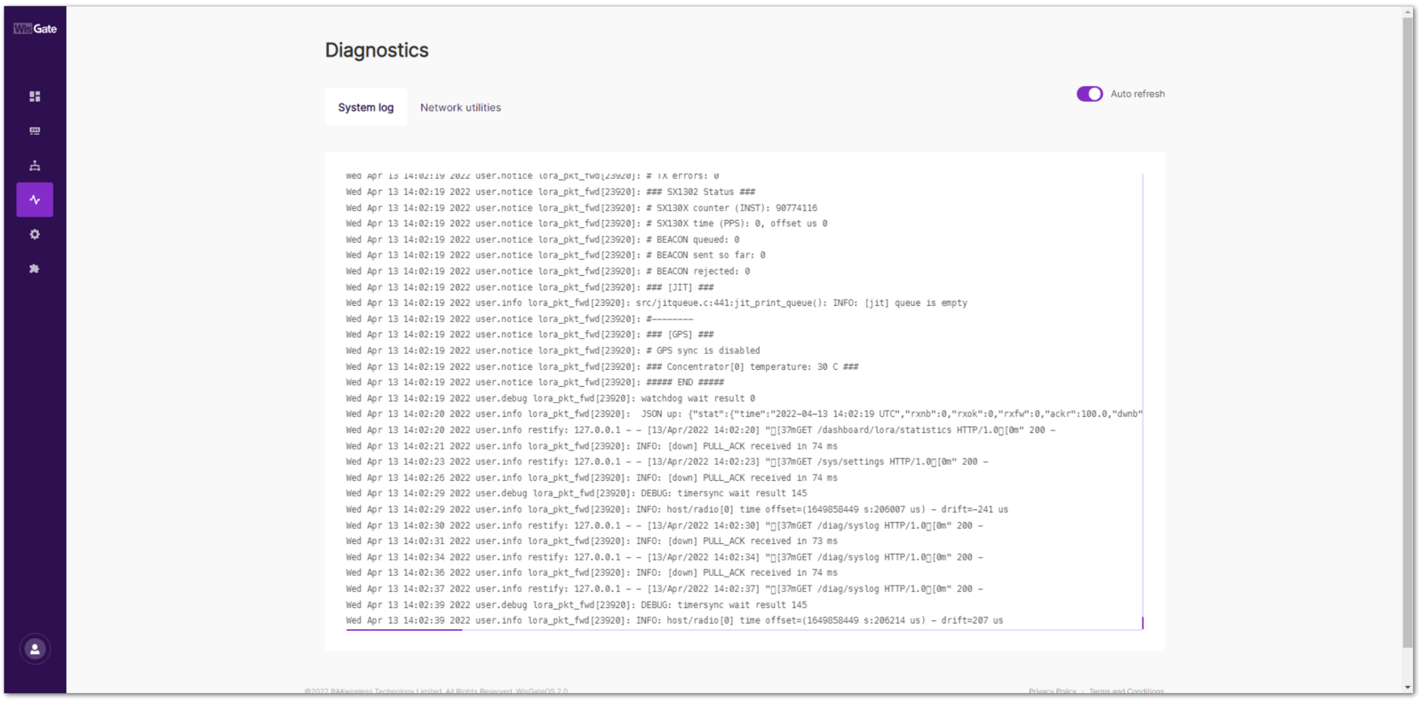
Using your device, create a condition to generate an event such as a high-temperature condition. If the temperature is above the configured threshold then you will receive a text alert on your phone. This alert will include key parameters about the alert.

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. Select time on the X-axis, Value as temp (Average), and Color as device_id to see a chart of your dataset.

Debugging

After login to the device using the web browser, the system log can be viewed from **Diagnostics** > **System Log**.


WisGate
Diagnostics
System log
Network utilities
Auto refresh
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

```
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # TX errors: 0
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: ### SX1302 Status ###
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # SX130X counter (INST): 90774116
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # SX130X time (PPS): 0, offset us 0
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # BEACON queued: 0
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # BEACON sent so far: 0
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # BEACON rejected: 0
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: ### [JIT] ###
Wed Apr 13 14:02:19 2022 user.info lora_pkt_fwd[23920]: src/jitqueue.c:441:jit_print_queue(): INFO: [jit] queue is empty
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: #--------
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: ### [GPS] ###
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: # GPS sync is disabled
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: ### Concentrator[0] temperature: 30 C ###
Wed Apr 13 14:02:19 2022 user.notice lora_pkt_fwd[23920]: ##### END #####
Wed Apr 13 14:02:19 2022 user.debug lora_pkt_fwd[23920]: watchdog wait result 0
Wed Apr 13 14:02:20 2022 user.info lora_pkt_fwd[23920]:  JSON up: {"stat":{"time":"2022-04-13 14:02:19 UTC","rxnb":0,"rxok":0,"rxfw":0,"ackr":100.0,"dwnb"
Wed Apr 13 14:02:20 2022 user.info restify: 127.0.0.1 - - [13/Apr/2022 14:02:20] "▯[37mGET /dashboard/lora/statistics HTTP/1.0▯[0m" 200 -
Wed Apr 13 14:02:21 2022 user.info lora_pkt_fwd[23920]: INFO: [down] PULL_ACK received in 74 ms
Wed Apr 13 14:02:23 2022 user.info restify: 127.0.0.1 - - [13/Apr/2022 14:02:23] "▯[37mGET /sys/settings HTTP/1.0▯[0m" 200 -
Wed Apr 13 14:02:26 2022 user.info lora_pkt_fwd[23920]: INFO: [down] PULL_ACK received in 74 ms
Wed Apr 13 14:02:29 2022 user.debug lora_pkt_fwd[23920]: DEBUG: timersync wait result 145
Wed Apr 13 14:02:29 2022 user.info lora_pkt_fwd[23920]: INFO: host/radio[0] time offset=(1649858449 s:206007 us) - drift=-241 us
Wed Apr 13 14:02:30 2022 user.info restify: 127.0.0.1 - - [13/Apr/2022 14:02:30] "▯[37mGET /diag/syslog HTTP/1.0▯[0m" 200 -
Wed Apr 13 14:02:31 2022 user.info lora_pkt_fwd[23920]: INFO: [down] PULL_ACK received in 73 ms
Wed Apr 13 14:02:34 2022 user.info restify: 127.0.0.1 - - [13/Apr/2022 14:02:34] "▯[37mGET /diag/syslog HTTP/1.0▯[0m" 200 -
Wed Apr 13 14:02:36 2022 user.info lora_pkt_fwd[23920]: INFO: [down] PULL_ACK received in 74 ms
Wed Apr 13 14:02:37 2022 user.info restify: 127.0.0.1 - - [13/Apr/2022 14:02:37] "▯[37mGET /diag/syslog HTTP/1.0▯[0m" 200 -
Wed Apr 13 14:02:39 2022 user.debug lora_pkt_fwd[23920]: DEBUG: timersync wait result 145
Wed Apr 13 14:02:39 2022 user.info lora_pkt_fwd[23920]: INFO: host/radio[0] time offset=(1649858449 s:206214 us) - drift=207 us
```

Figure 17: System Log

Troubleshooting

1. Unable to see the web login:

   - Check that your wifi is connected to **RAK7268_XXXX**.
   - Try ping **192.168.230.1**.

2. Lost password to login to the web login.

   - Hold the reset button for 10 seconds to factory reset the device

AWS IoT Core Integration

AWS IoT Core Integration is a software service that enables your LoRaWAN gateway to work with AWS IoT Core. The AWS Integration for WisGate Edge V2 tutorial will show you how to set up a LoRaWAN end-node and view its data on the AWS IoT Console. In addition, it'll show you how to send a message from AWS IoT Console to the end-node as well.

The Things Network (TTN)

WisGateOS 2 Basics Station to TTNv3

This tutorial illustrates how to configure and connect your RAK Edge Gateway V2 with WisGateOS 2 to a LoRaWAN Network Server by using the Basics Station protocol. For this example, it will be shown how to connect the gateway to TTNv3.

> 📝 **NOTE:**
>
> LoRa Basics Station is an implementation of a LoRa packet forwarder. This protocol simplifies the management of large-scale LoRaWAN Networks. More information about the Basics Station protocol can be found in the explanatory document provided by Semtech.

Registering the Gateway

1. Log in first and head on to TTNv3 website . If you already have a TTN account, you can use your The Things ID credentials to log in.

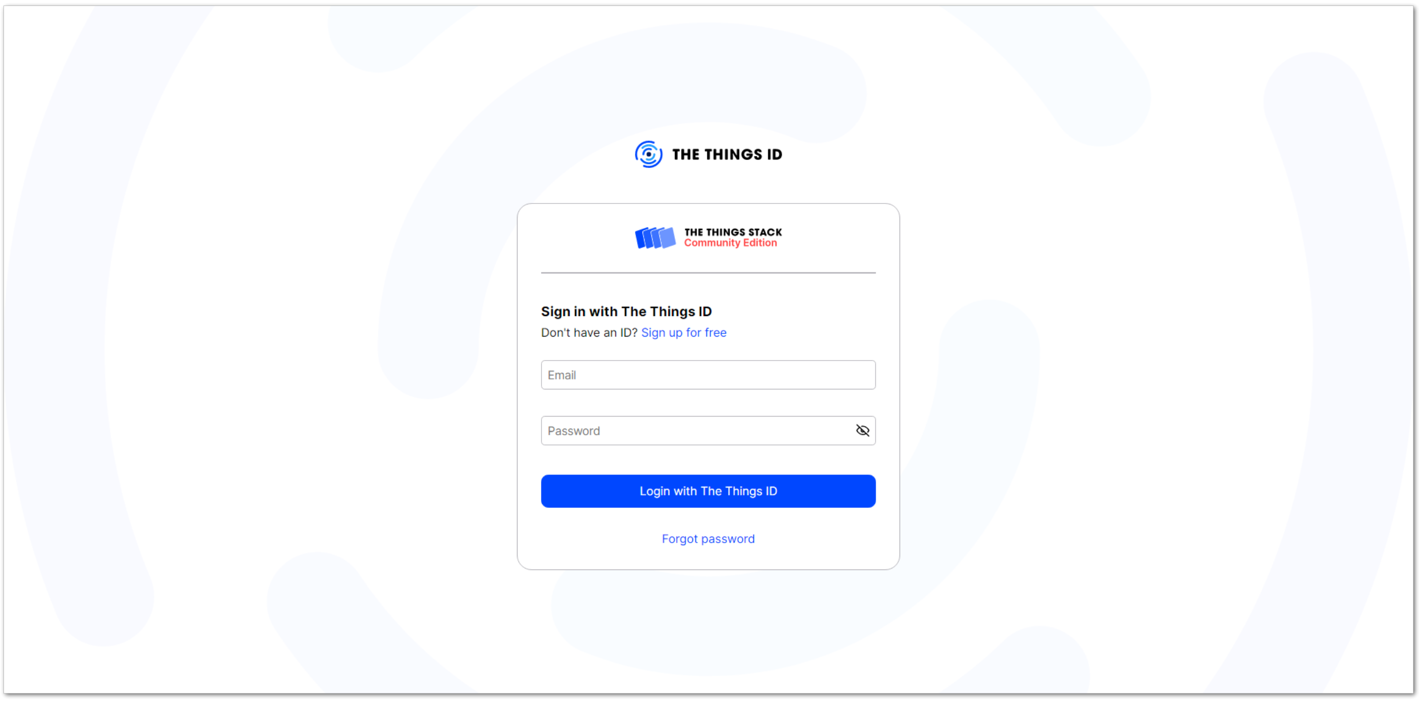
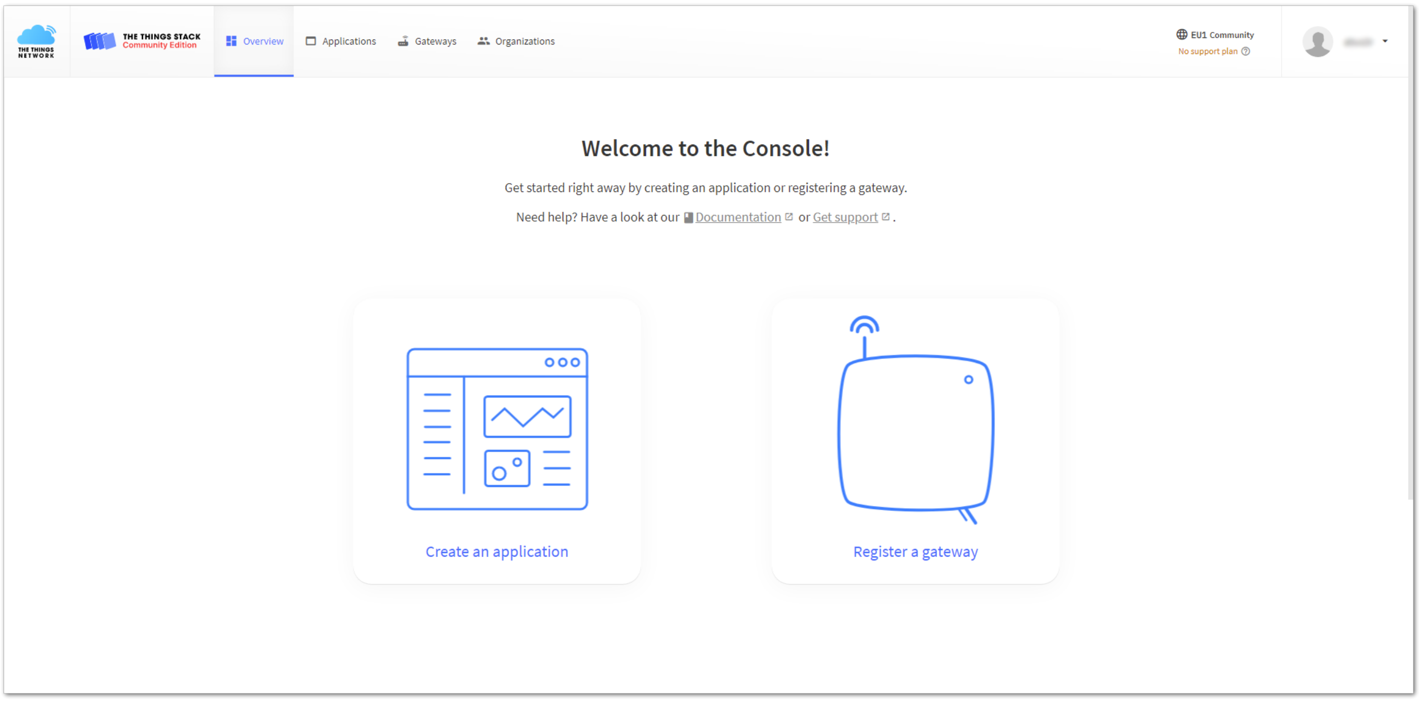


Figure 18: The Things Stack login page

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

2. 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).



Welcome to the Console!

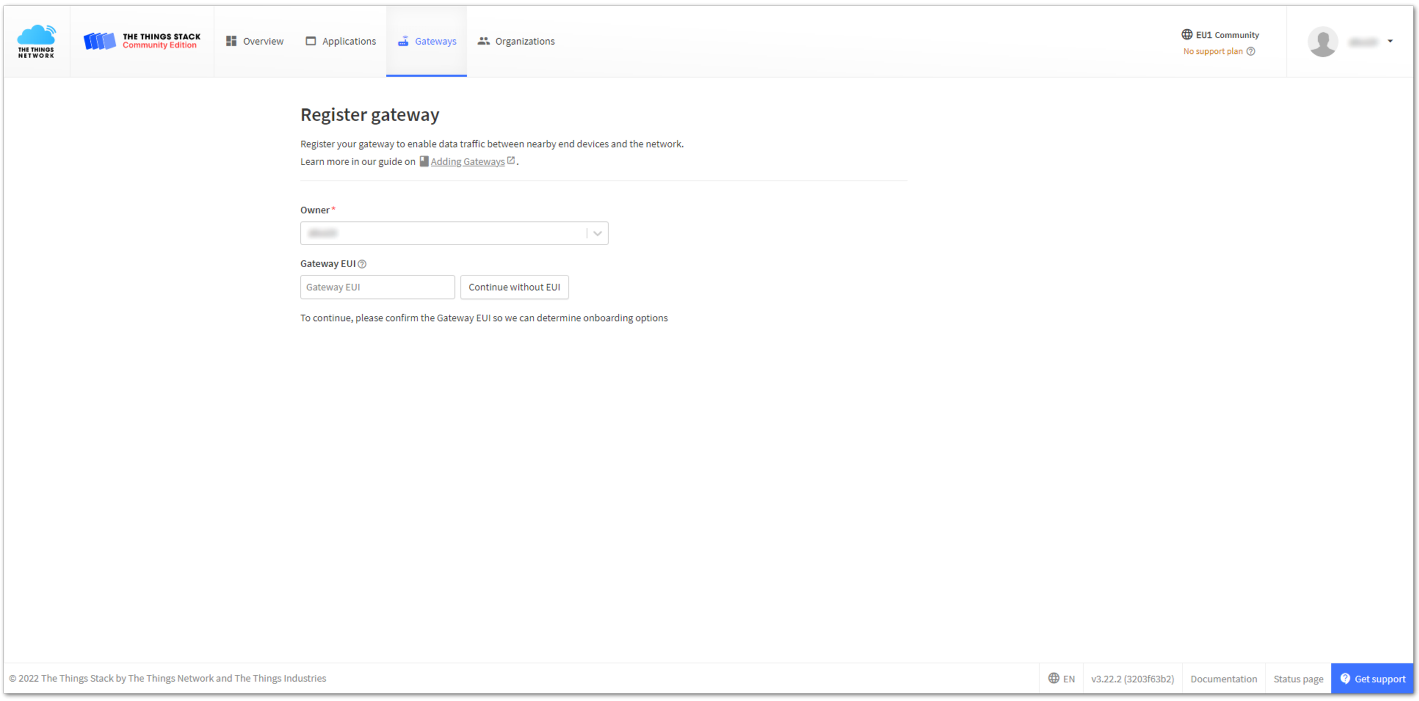
Get started right away by creating an application or registering a gateway.

Need help? Have a look at our Documentation or Get support .

Figure 19: Console Page after a successful login

3. You will be redirected to the **Register gateway** page.

4. In the **Gateway EUI** field, type the EUI of the 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 Quick Start Guide .



Register gateway

Register your gateway to enable data traffic between nearby end devices and the network.
Learn more in our guide on Adding Gateways .

Owner *
[redacted]

Gateway EUI
Gateway EUI | Continue without EUI

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

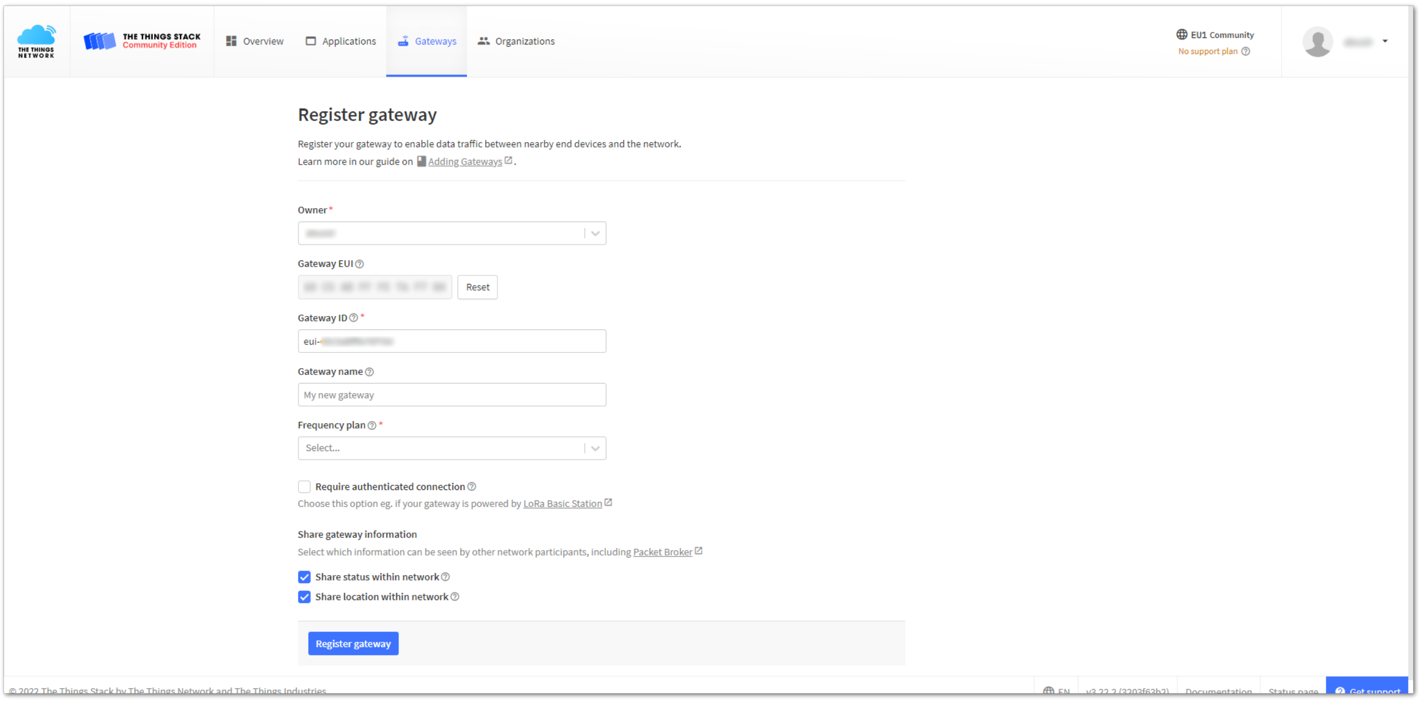
© 2022 The Things Stack by The Things Network and The Things Industries

Figure 20: Register gateway

5. After typing the EUI, click on **Confirm**. Additional fields will pop up. Fill in the following information:

- **Gateway ID** – This will be the unique ID of your gateway in the Network. An ID based on the EUI is automatically generated. You can change it if you need. Note that the ID must contain only lowercase letters, numbers, and dashes (-).
- **Gateway name** – Optionally, you can type a name for your gateway.
- **Frequency plan** - The frequency plan used by the gateway.

> 📝 **NOTE:**
>
> - The other settings are optional and can be changed to satisfy your requirements.
> - For this tutorial, we will use Europe 863-870 MHz (SF12 for RX2).



Register gateway

Register your gateway to enable data traffic between nearby end devices and the network.
Learn more in our guide on Adding Gateways.

Owner *
[redacted]

Gateway EUI
[redacted] Reset

Gateway ID *
eui-[redacted]

Gateway name
My new gateway

Frequency plan *
Select...

□ Require authenticated connection
Choose this option eg. if your gateway is powered by LoRa Basic Station

Share gateway information
Select which information can be seen by other network participants, including Packet Broker

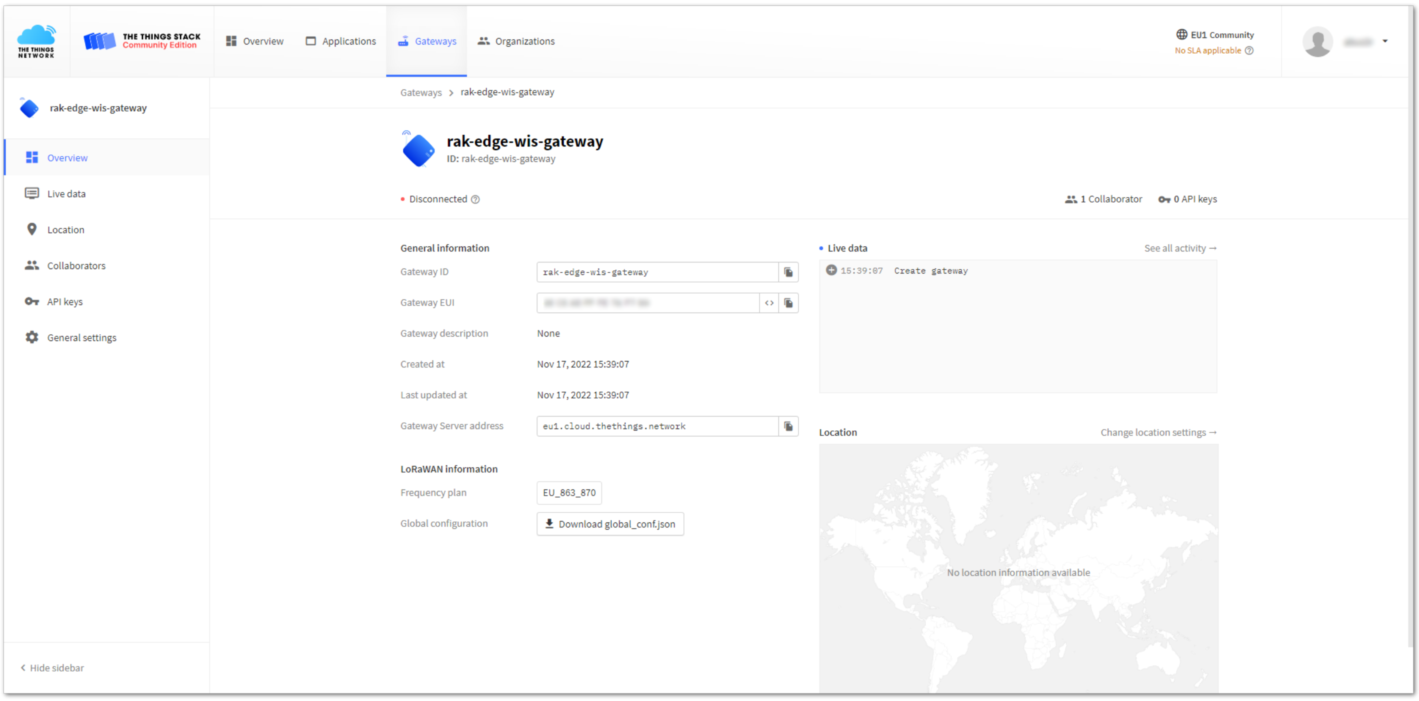
☑ Share status within network
☑ Share location within network

Register gateway

© 2022 The Things Stack by The Things Network and The Things Industries
EN
v3.22.2 (3203f63b2)
Documentation
Status page
Get support

Figure 21: Adding a gateway

6. To register your gateway, click **Register gateway**.



rak-edge-wis-gateway
ID: rak-edge-wis-gateway

Disconnected

1 Collaborator
0 API keys

General information

Gateway ID rak-edge-wis-gateway
Gateway EUI [redacted]
Gateway description None
Created at Nov 17, 2022 15:39:07
Last updated at Nov 17, 2022 15:39:07
Gateway Server address eu1.cloud.thethings.network

LoRaWAN information

Frequency plan EU_863_870
Global configuration Download global_conf.json

Live data
See all activity →

```
15:39:07  Create gateway
```

Location
Change location settings →

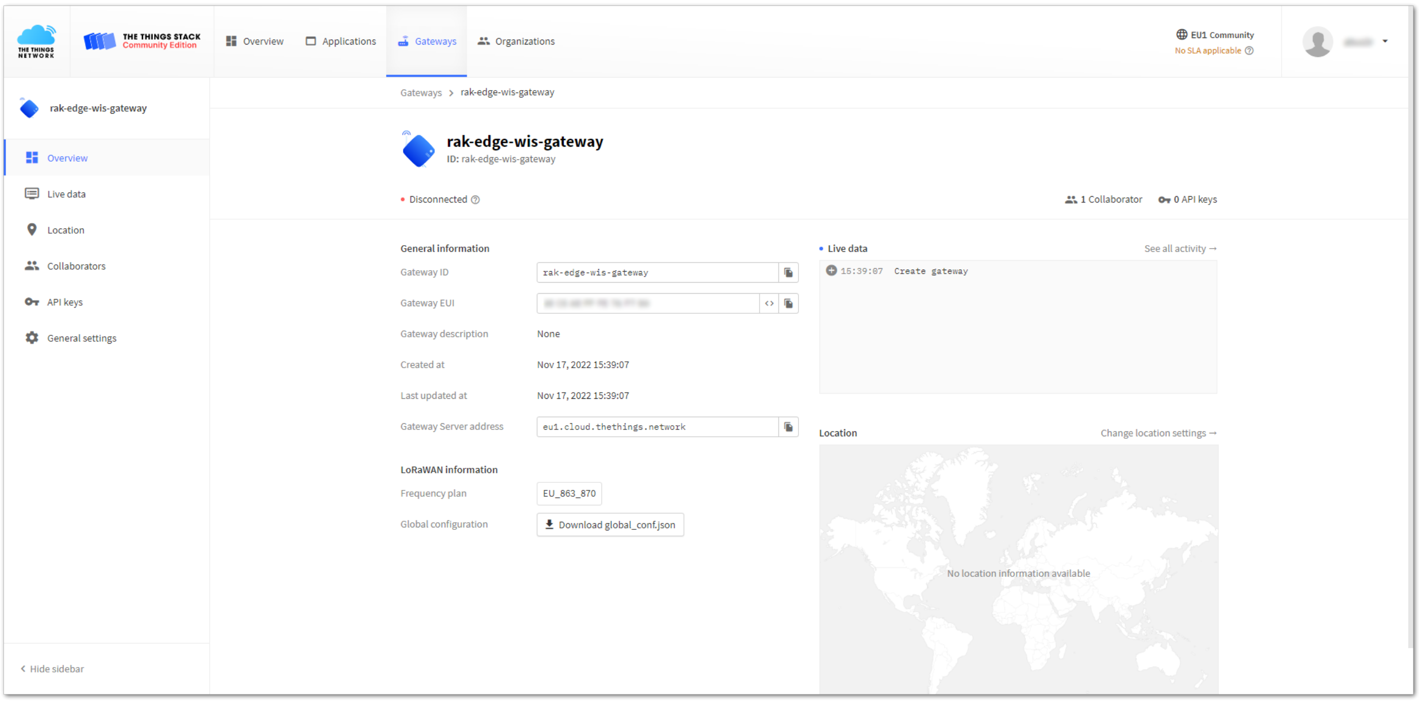
No location information available

Figure 22: Successfully added a 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**.



rak-edge-wis-gateway
ID: rak-edge-wis-gateway

• Disconnected

General information

Gateway ID	rak-edge-wis-gateway
Gateway EUI	[redacted]
Gateway description	None
Created at	Nov 17, 2022 15:39:07
Last updated at	Nov 17, 2022 15:39:07
Gateway Server address	eu1.cloud.thethings.network

LoRaWAN information

Frequency plan	EU_863_870
Global configuration	Download global_conf.json

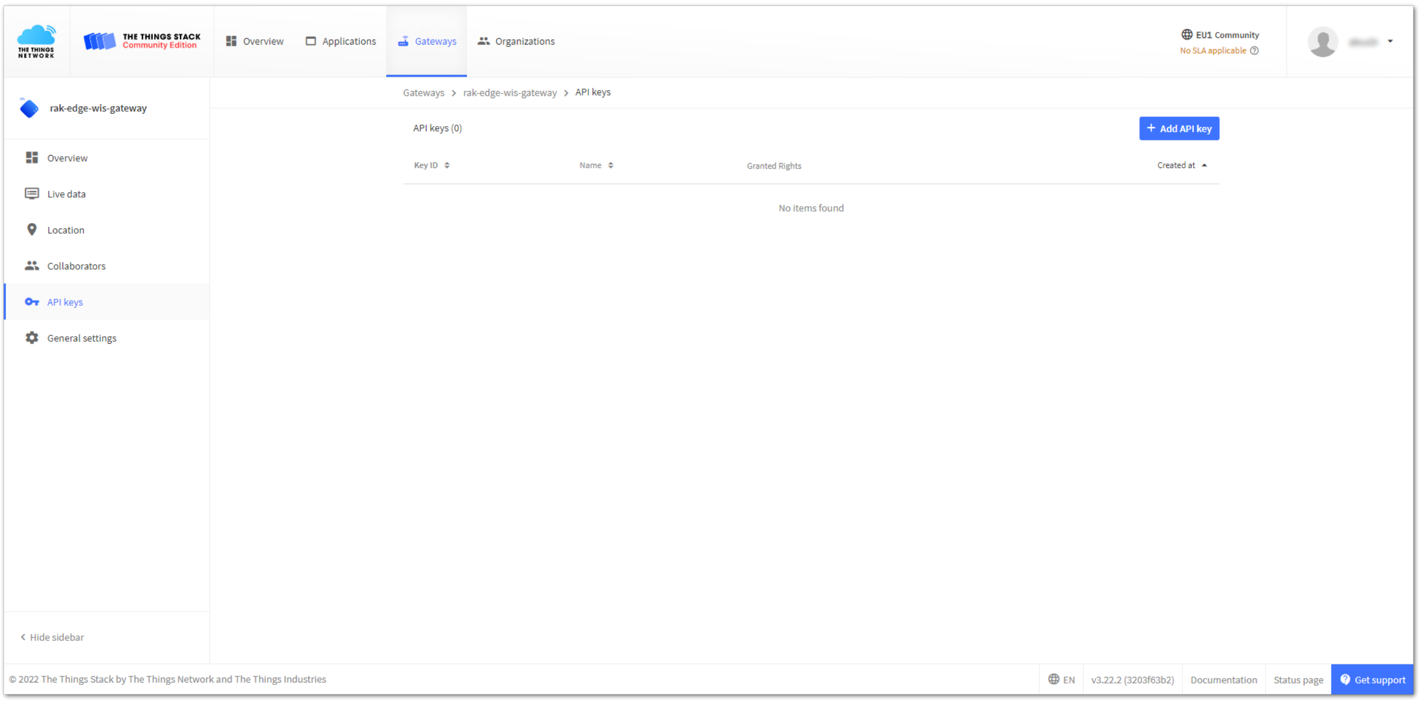
• Live data

```
15:39:07  Create gateway
```

Location

Figure 23: Overview page

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



API keys (0)

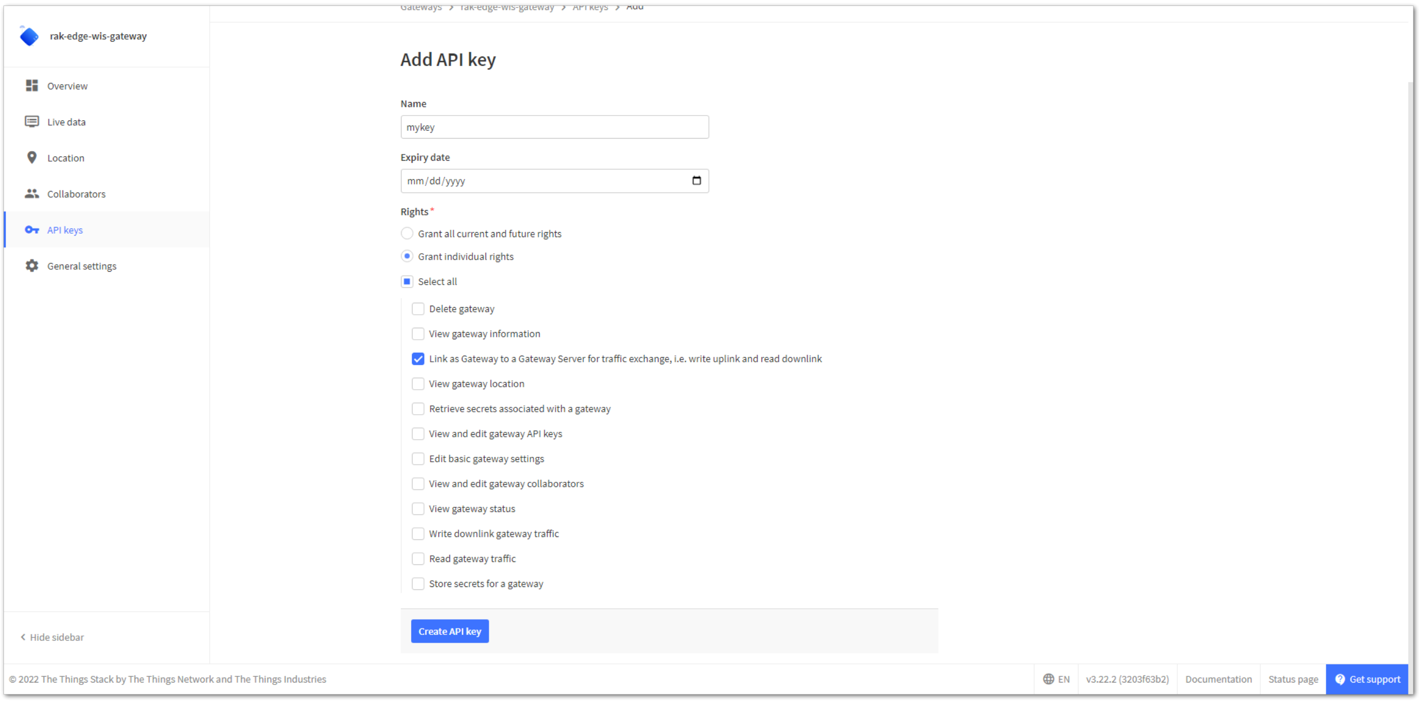
Key ID	Name	Granted Rights	Created at

No items found

© 2022 The Things Stack by The Things Network and The Things Industries v3.22.2 (3203f63b2)

Figure 24: 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**.



Gateways > rak-edge-wis-gateway > API keys > Add

rak-edge-wis-gateway

Add API key

Name: mykey

Expiry date: mm/dd/yyyy

Rights*

○ Grant all current and future rights

● Grant individual rights

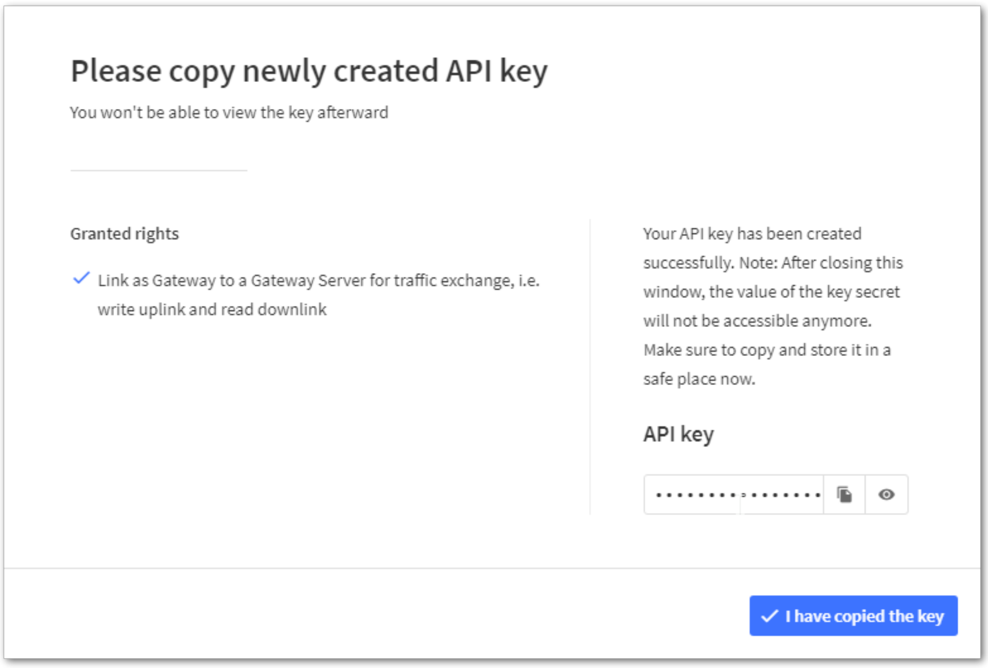
■ Select all

- □ Delete gateway
- □ View gateway information
- ☑ Link as Gateway to a Gateway Server for traffic exchange, i.e. write uplink and read downlink
- □ View gateway location
- □ Retrieve secrets associated with a gateway
- □ View and edit gateway API keys
- □ Edit basic gateway settings
- □ View and edit gateway collaborators
- □ View gateway status
- □ Write downlink gateway traffic
- □ Read gateway traffic
- □ Store secrets for a gateway

Create API key
© 2022 The Things Stack by The Things Network and The Things Industries
EN
v3.22.2 (3203f63b2)
Documentation
Status page
Get support

Figure 25: 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.



Please copy newly created API key

You won't be able to view the key afterward

Granted rights

✓ Link as Gateway to a Gateway Server for traffic exchange, i.e. write uplink and read downlink

Your API key has been created successfully. Note: After closing this window, the value of the key secret will not be accessible anymore. Make sure to copy and store it in a safe place now.

API key

••••••••••••••••••

Figure 26: Copying the generated key

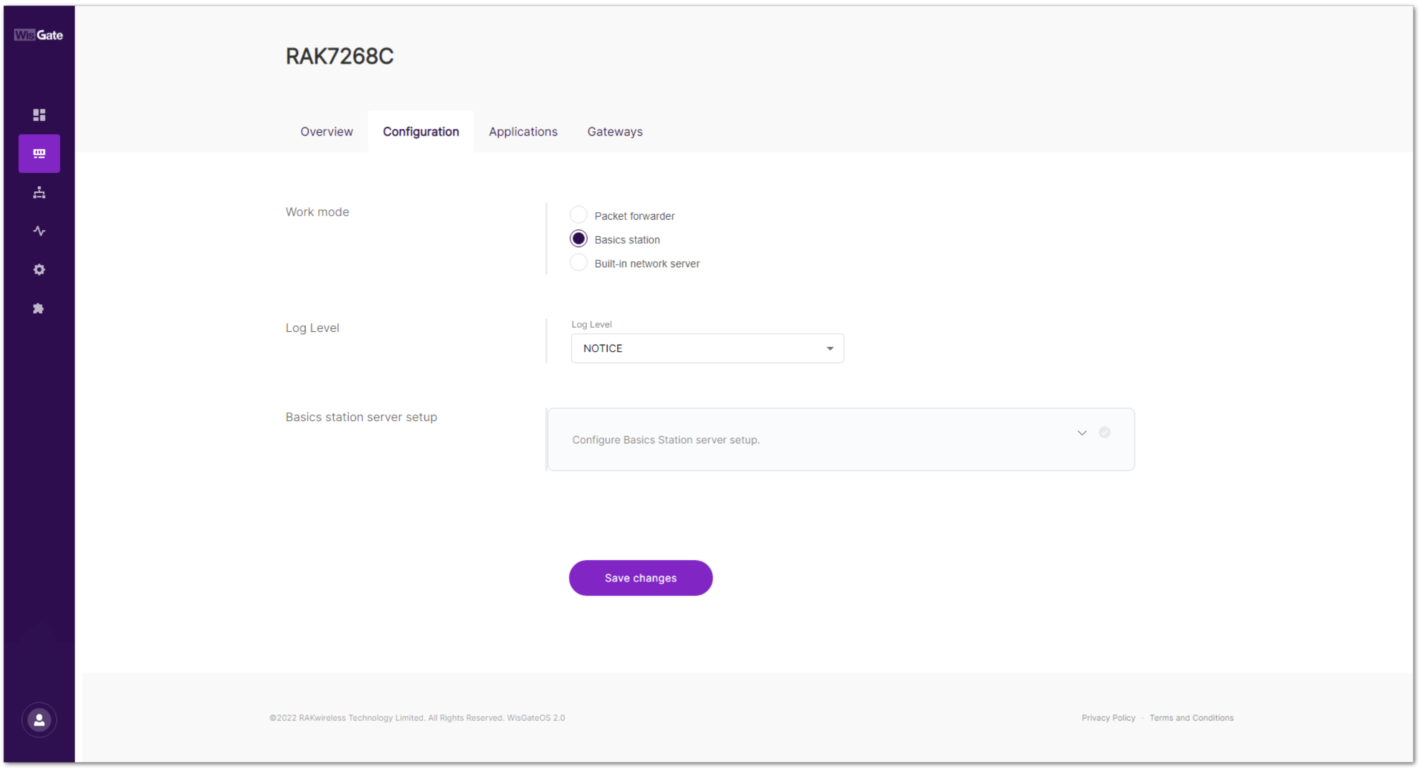
> ⚠️ **WARNING:**
>
> Copy the key and save it in a `.txt` file (or other), because you won't be able to view or copy your key after that.

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, refer to the Quick Start Guide for each gateway.

2. Navigate to **LoRa** > **Configuration** > **Work mode** and select **Basics station**.



WisGate
Overview
Configuration
Applications
Gateways

RAK7268C

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

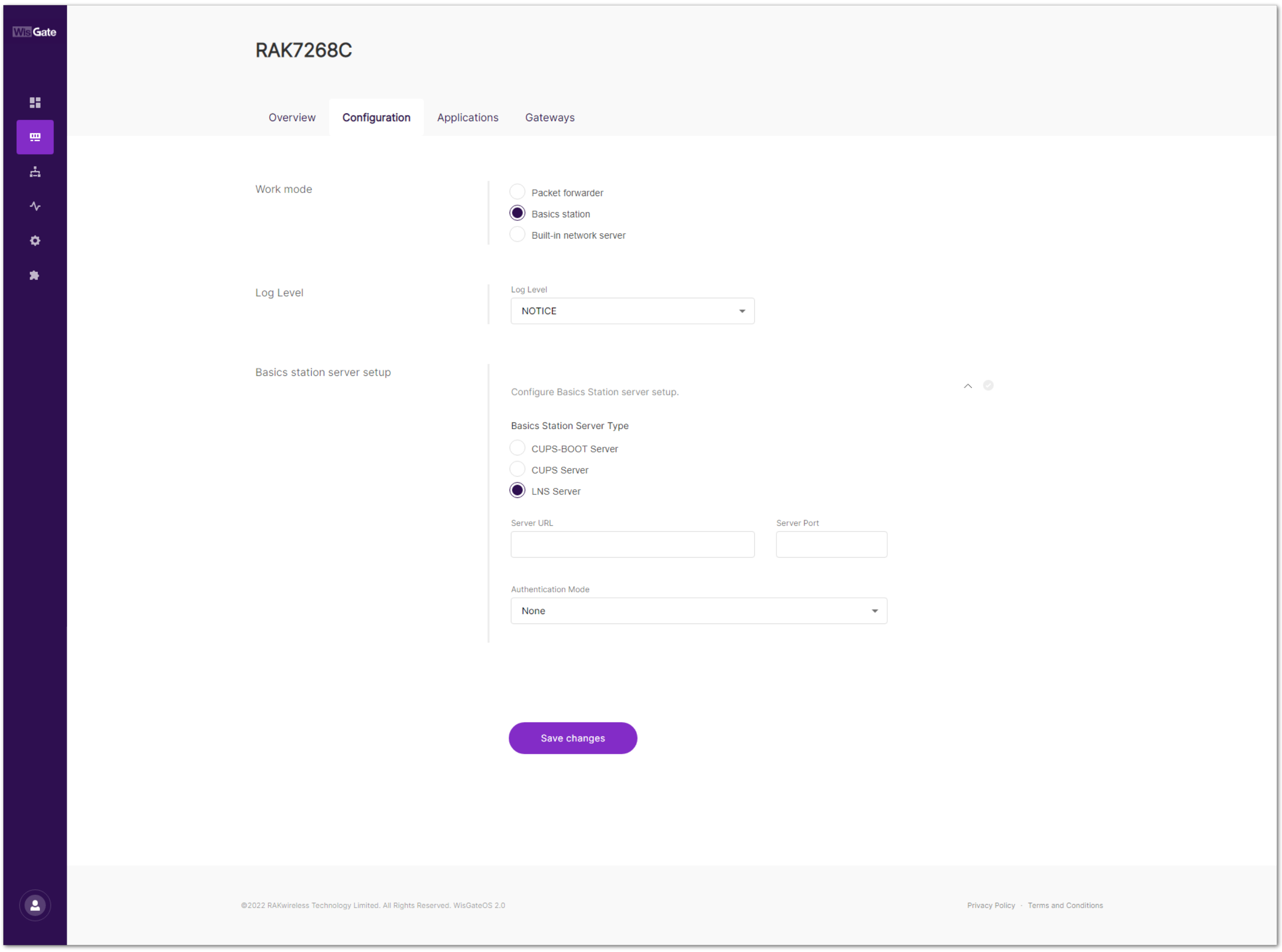
Log Level
Log Level: NOTICE

Basics station server setup
Configure Basics Station server setup.

Save changes
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

Figure 27: Changing the working mode

3. Expand the Basics Station settings by clicking **Configure Basics Station server setup.**



WisGate
Overview
Configuration
Applications
Gateways

RAK7268C

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: NOTICE

Basics station server setup
Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

Server URL:
Server Port:

Authentication Mode: None

Save changes
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

Figure 28: Expanded Basics Station settings

4. To connect the gateway to TTNv3, configure the following parameters:

- **Basics Station Server Type** – For server type, choose **LNS Server**.
- **Server URL** – This is the link to The Things Stack server. Note that, for this tutorial, the gateway is connected to the European cluster. For Europe fill in the following:

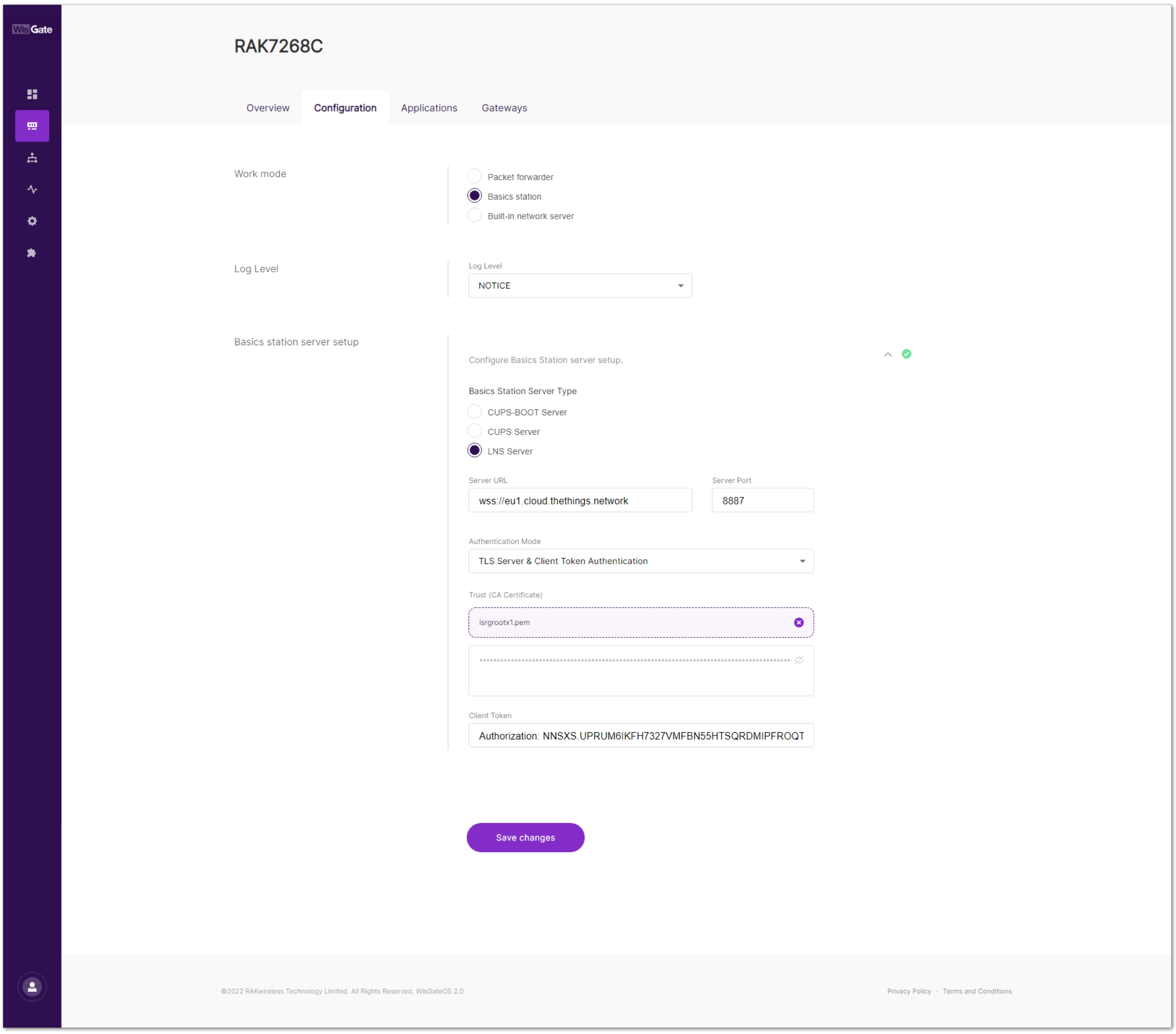
```
wss://eu1.cloud.thethings.network
```

- **Server Port** – The LNS Server uses port 8887. Type in **8887**.
- **Authentication Mode** – Choose **TLS server authentication and Client token**. When selected, the **Trust (CA Certificate)** and **Client token** fields will show up.
- **Trust (CA Certificate)** – For trust, upload the **Let’s Encrypt ISRG ROOT X1 Trust** certificate by clicking **choose file**. The file with the certificate can be downloaded directly .
- **Client Token** - This is the generated API key. The key must start with **Authorization:**.

For 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.



Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: NOTICE

Basics station server setup
Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

Server URL: wss://eu1.cloud.thethings.network
Server Port: 8887

Authentication Mode: TLS Server & Client Token Authentication

Trust (CA Certificate): isrgrootx1.pem

••

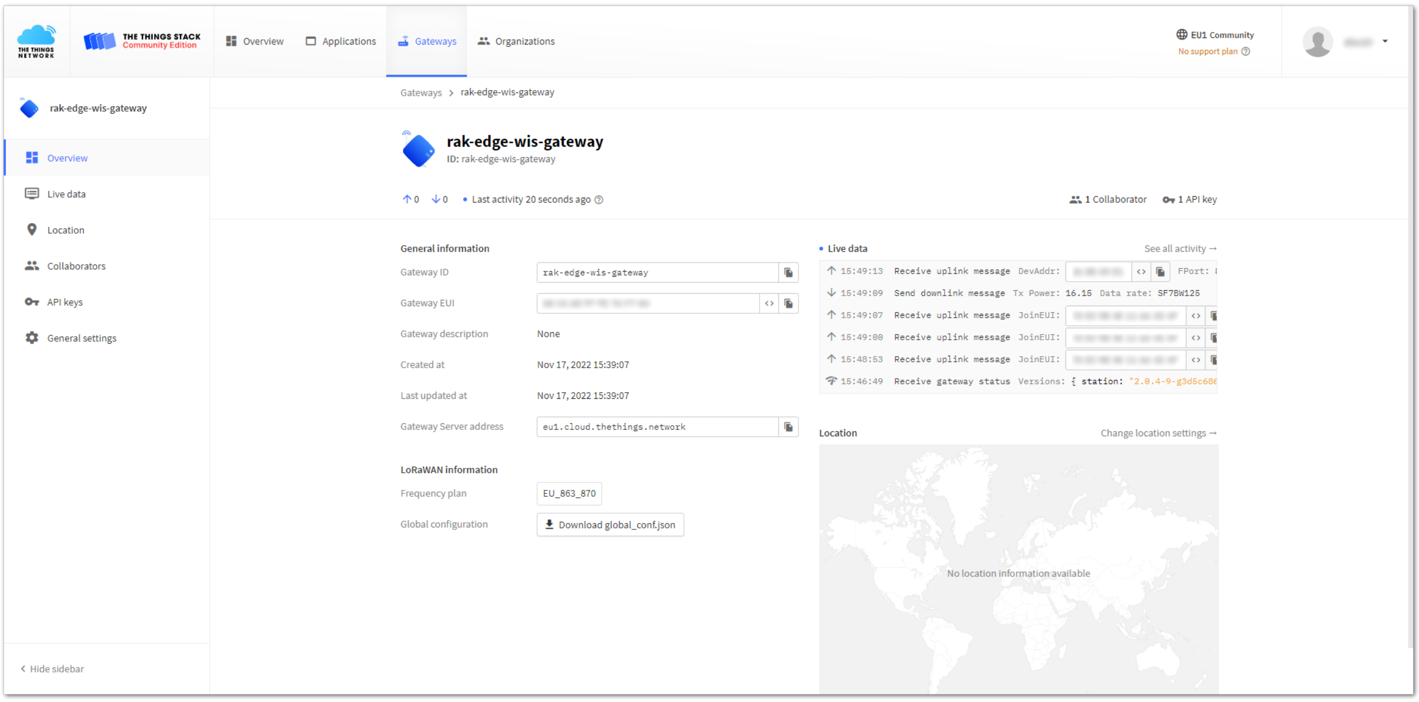
Client Token: Authorization: NNSXS.UPRUM6IKFH7327VMFBN55HTSQRDMIPFROQT

Save changes
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

Figure 29: Basics Station settings

5. To save the changes, click **Save Changes**.

Now, you can see that their gateway is connected to TTNv3 as Basics Station.



rak-edge-wis-gateway
ID: rak-edge-wis-gateway

Last activity 20 seconds ago

General information

Gateway ID	rak-edge-wis-gateway
Gateway EUI	[redacted]
Gateway description	None
Created at	Nov 17, 2022 15:39:07
Last updated at	Nov 17, 2022 15:39:07
Gateway Server address	eu1.cloud.thethings.network

LoRaWAN information

Frequency plan	EU_863_870
Global configuration	Download global_conf.json

Live data

```
↑ 15:49:13  Receive uplink message  DevAddr: [redacted]  FPort:
↓ 15:49:09  Send downlink message  Tx Power: 16.15  Data rate: SF7BW125
↑ 15:49:07  Receive uplink message  JoinEUI: [redacted]
↑ 15:49:00  Receive uplink message  JoinEUI: [redacted]
↑ 15:48:53  Receive uplink message  JoinEUI: [redacted]
  15:46:49  Receive gateway status  Versions: { station: "2.0.4-9-g3d5c68
```

Location

Figure 30: Successful connection

ChirpStack

This guide will show you how to connect the RAKWireless Commercial V2 gateway running WisGateOS 2 to a ChirpStack Network server, whether it is installed in the local or external network.

The guide is not about how to install the ChirpStack, but how to configure the gateway to send data to it.

Configuring the Edge Gateway V2 to ChirpStack

As mentioned before, the guide is for all RAK Edge V2 Series gateways running WisGateOS 2. It will be separated into two main sections based on where the ChirpStack Network server is installed:

- Local ChirpStack
- External ChirpStack

Different methods on how to connect the gateway to the server will be shown.

Local ChirpStack

In this case, the ChirpStack is installed in the local network. Three options will be considered here:

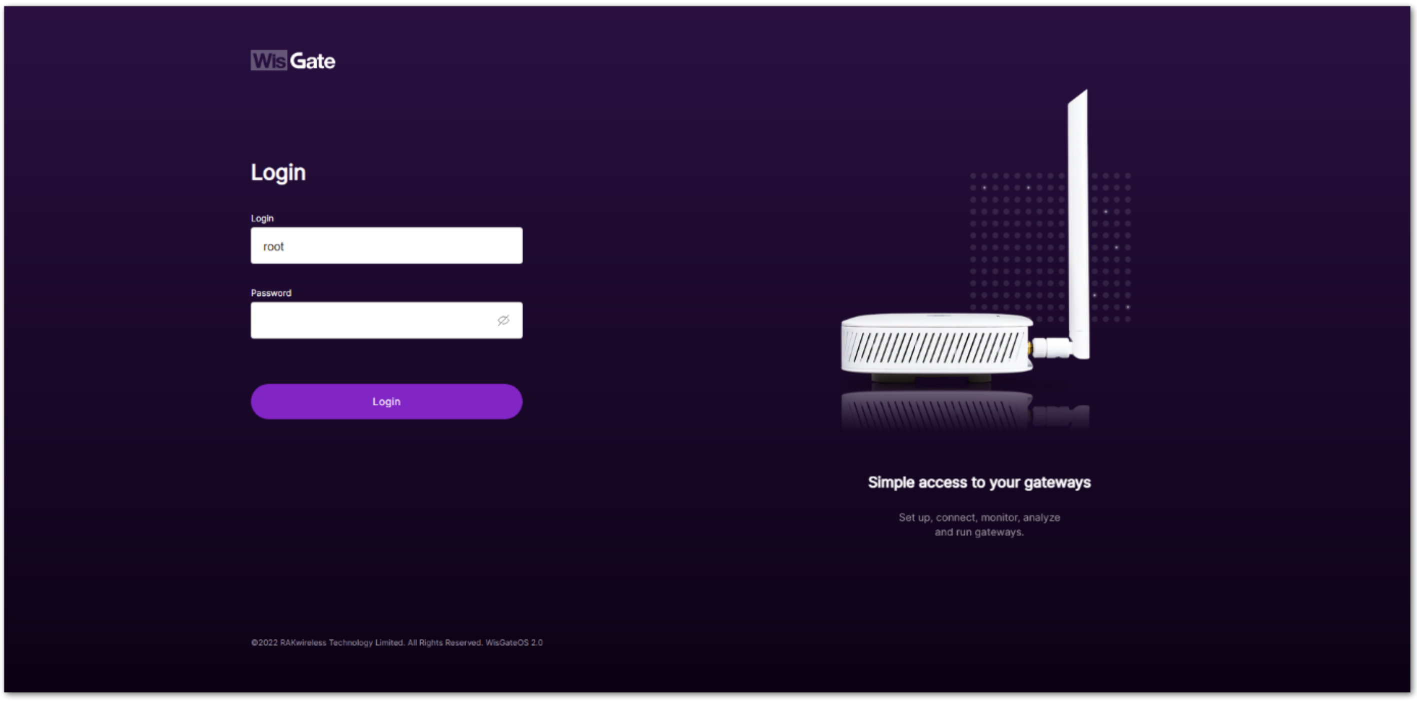
- **Connecting the Gateway via Packet Forwarder**
- **Connecting the Gateway via MQTT Bridge**
- **Connecting the Gateway via Basics Station**

Each option is explained in a separate section.

Connecting the Gateway via Packet Forwarder

In this method, you will configure the gateway's packet forwarder to send data to the ChirpStack Gateway Bridge.

1. Start by accessing the gateway. To see how to access the gateway, refer to the **Access the Gateway** section.



WisGate

Login

Login

root

Password

Simple access to your gateways

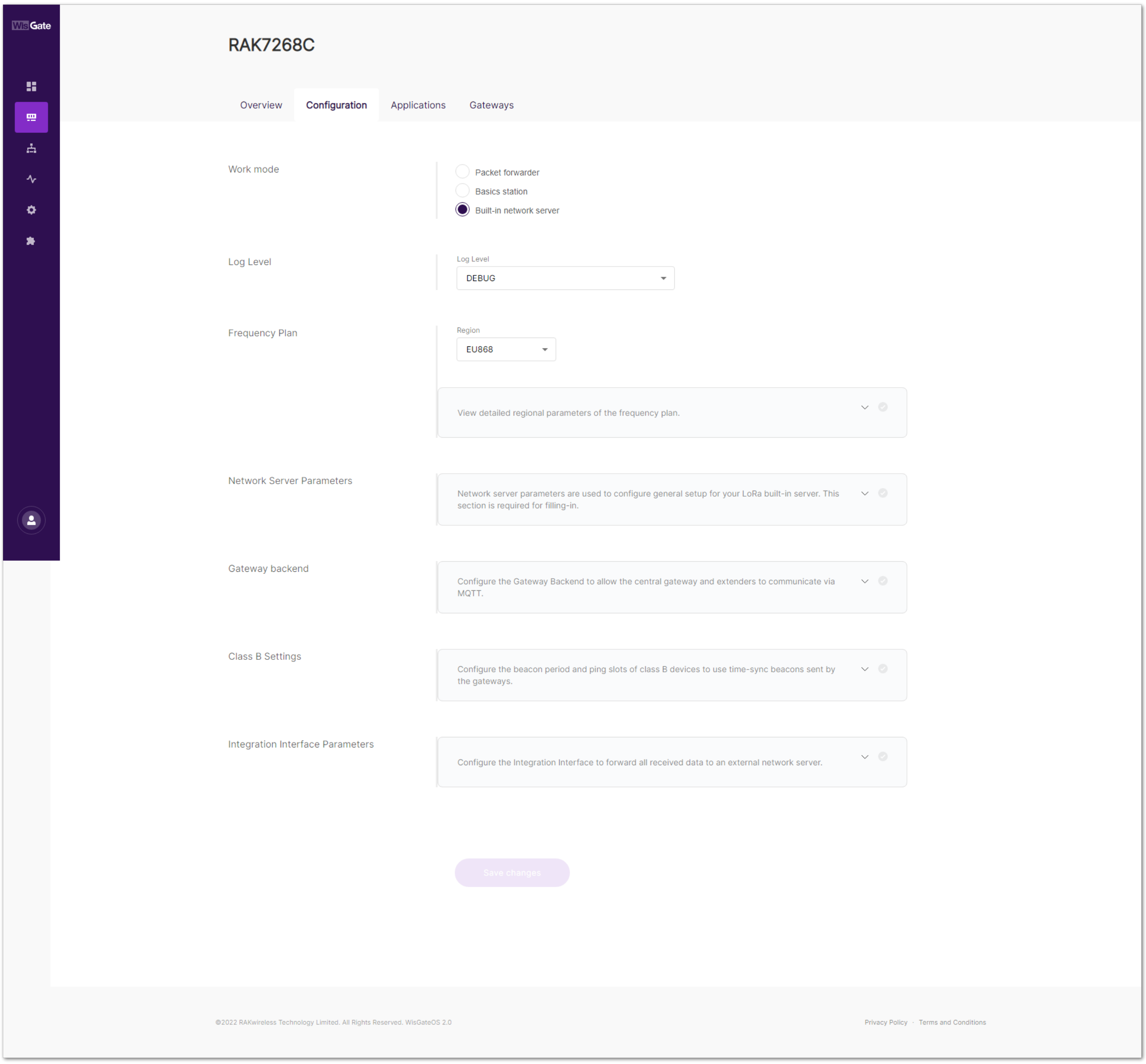
Set up, connect, monitor, analyze and run gateways.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 31: Login page

2. Login using the set credentials you have set in the **Access the Gateway** section.

3. On the left side, head to **LoRa**. By default, the gateway is configured to work as a Built-in network server.



Work mode

○ Packet forwarder
○ Basics station
● Built-in network server

Log Level

Log Level: DEBUG

Frequency Plan

Region: EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters

Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

Gateway backend

Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

Class B Settings

Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

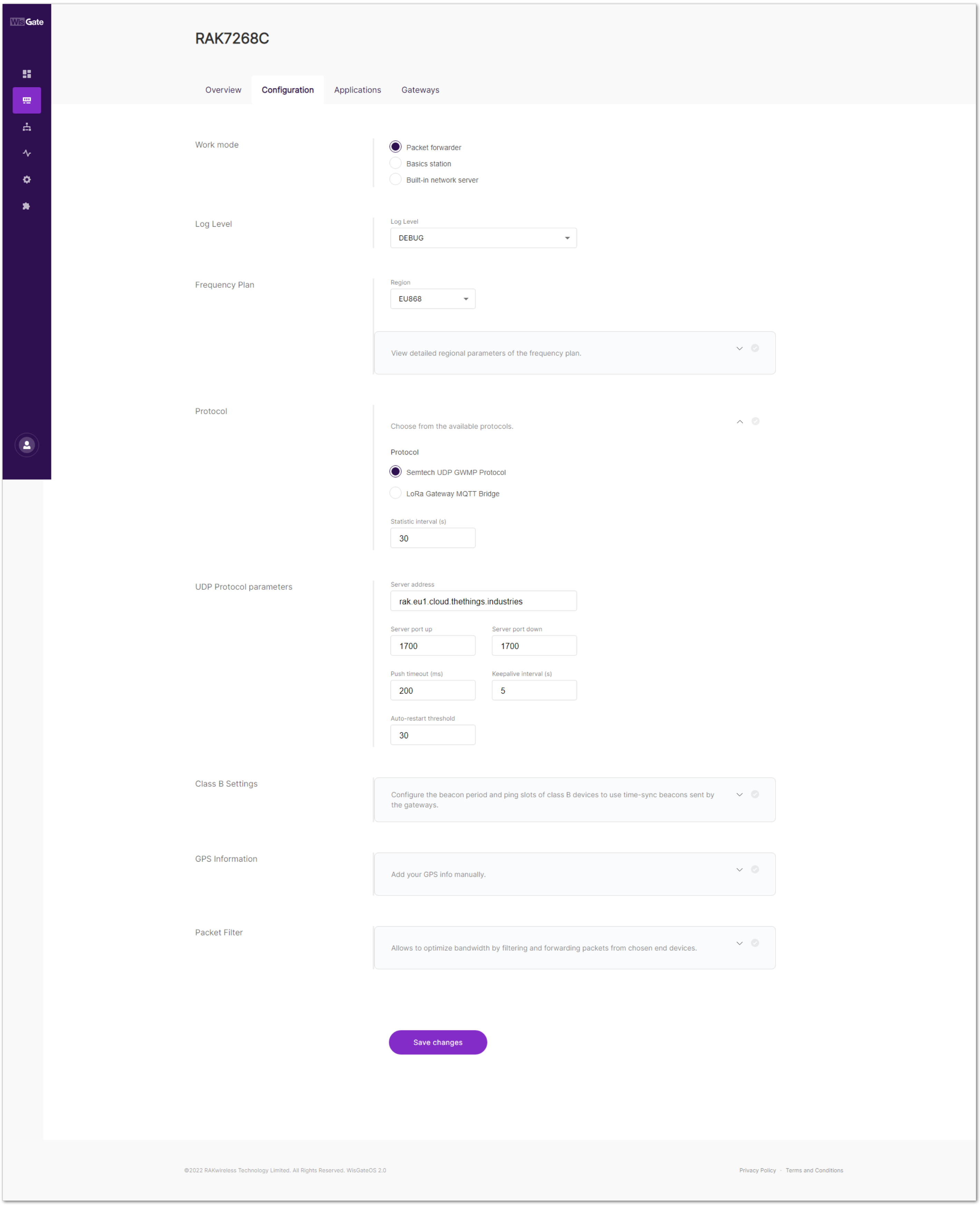
Integration Interface Parameters

Configure the Integration Interface to forward all received data to an external network server.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 32: Network server settings

4. From **Work Mode**, select **Packet forwarder**. Click **Choose from the available protocols** to expand the Packet forwarder settings.



RAK7268C

Work mode
● Packet forwarder
○ Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Frequency Plan
Region: EU868
View detailed regional parameters of the frequency plan.

Protocol
Choose from the available protocols.
Protocol
● Semtech UDP GWMP Protocol
○ LoRa Gateway MQTT Bridge
Statistic interval (s): 30

UDP Protocol parameters
Server address: rak.eu1.cloud.thethings.industries
Server port up: 1700
Server port down: 1700
Push timeout (ms): 200
Keepalive interval (s): 5
Auto-restart threshold: 30

Class B Settings
Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

GPS Information
Add your GPS info manually.

Packet Filter
Allows to optimize bandwidth by filtering and forwarding packets from chosen end devices.

Save changes

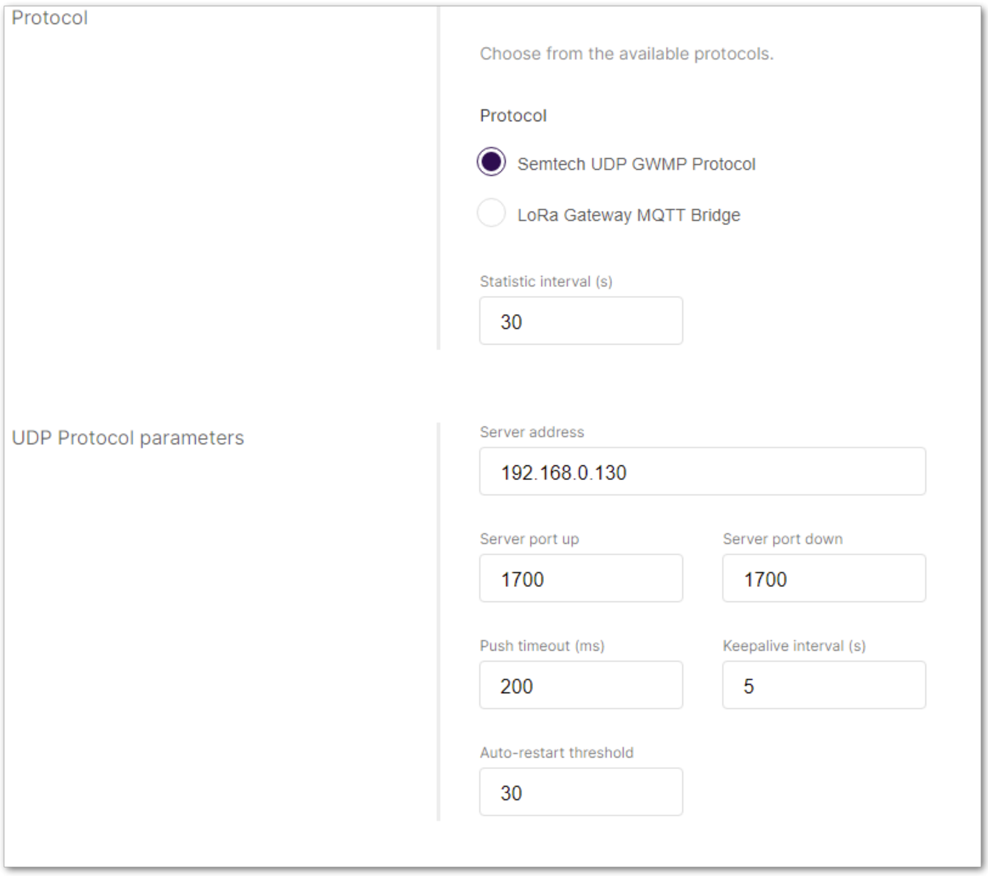
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

Figure 33: Setting Packet Forwarder Mode

4. By default, when Packet Forwarder mode is chosen, the **Semtech UDP GWMP Protocol** is selected.

5. To point the gateway to the ChirpStack network using the packet forwarder, you only need to set the **Server address** of the ChirpStack.

In this case, the ChirpStack is installed locally on an Ubuntu machine on IP `192.168.0.130` (yours will be different). The other fields are filled with default parameters and can be left by default.



Protocol

Choose from the available protocols.

Protocol

● Semtech UDP GWMP Protocol

○ LoRa Gateway MQTT Bridge

Statistic interval (s)

30

UDP Protocol parameters

Server address

192.168.0.130

Server port up

1700

Server port down

1700

Push timeout (ms)

200

Keepalive interval (s)

5

Auto-restart threshold

30

Figure 34: Configuring Packet Forwarder to ChirpStack

6. Click **Save changes** to save the changes.

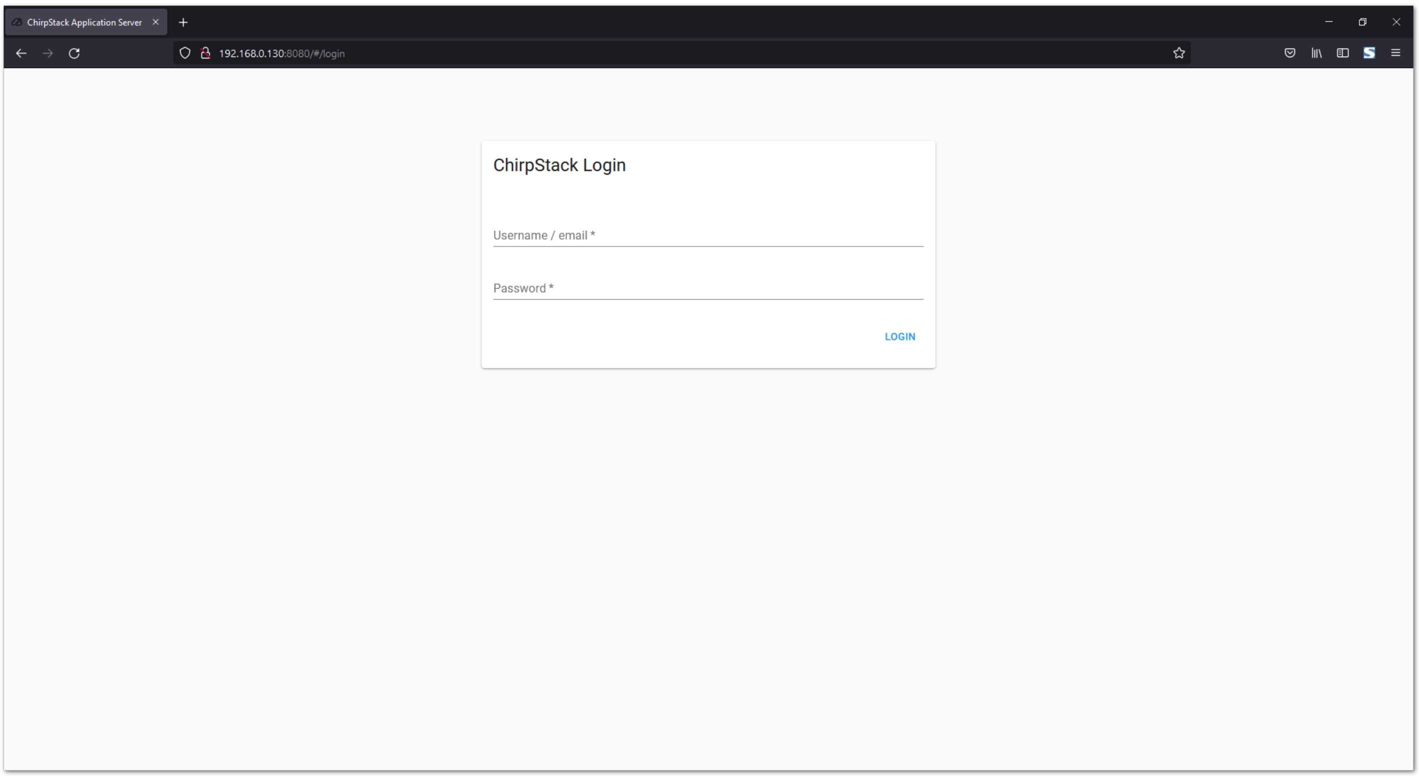
Now you need to register the gateway in ChirpStack. The steps are the same for all options.

Registering the Gateway in ChirpStack Network Server

1. To register the gateway in the ChirpStack Network server, access the ChirpStack UI. To do that, open a web browser and type the server address of the ChirpStack with port 8080.

```
<IP address of ChirpStack>:8080
```

In this case, the ChirpStack is installed on a local Ubuntu machine with IP `192.168.0.130`. The server address will be `192.168.0.130:8080`.



ChirpStack Login

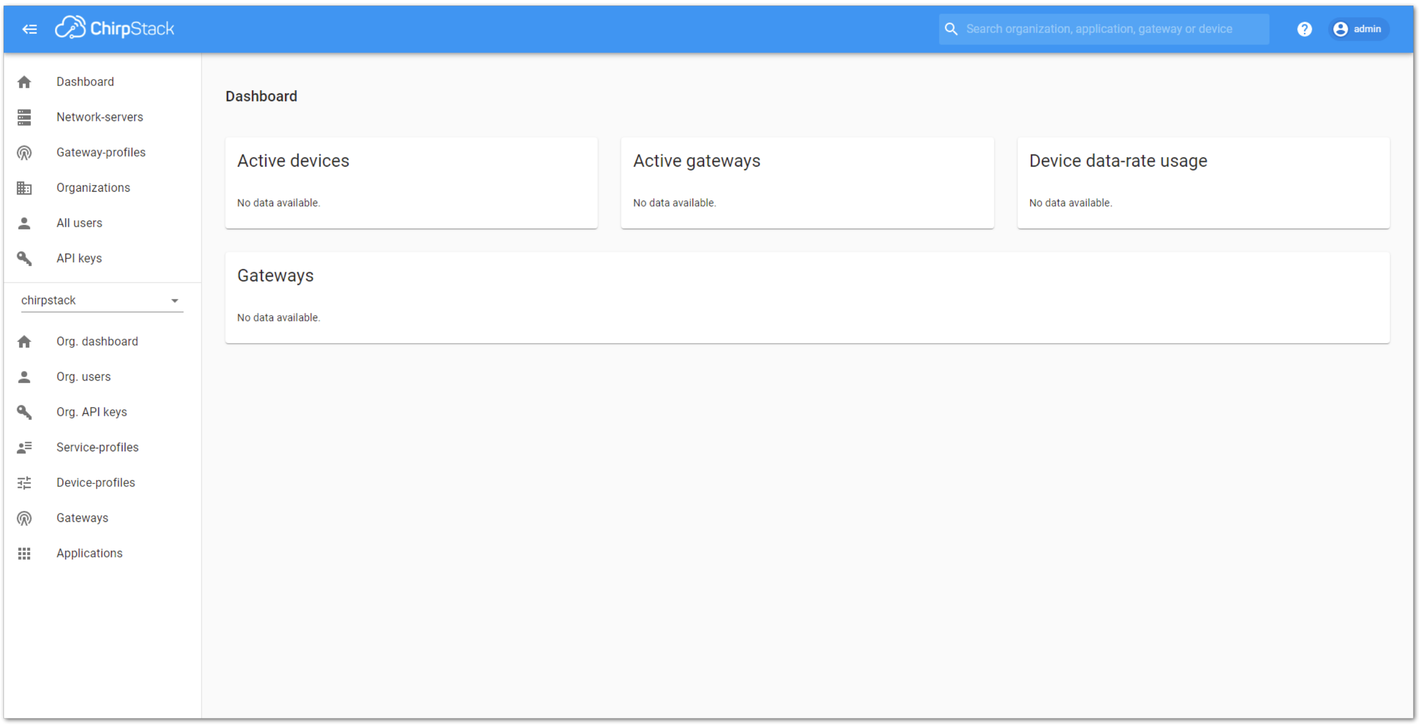
Username / email *

Password *

Figure 35: ChirpStack Login page

2. Login using the following credentials:

- Username/email: **admin**
- Password: **admin**



Dashboard

Active devices

No data available.

Active gateways

No data available.

Device data-rate usage

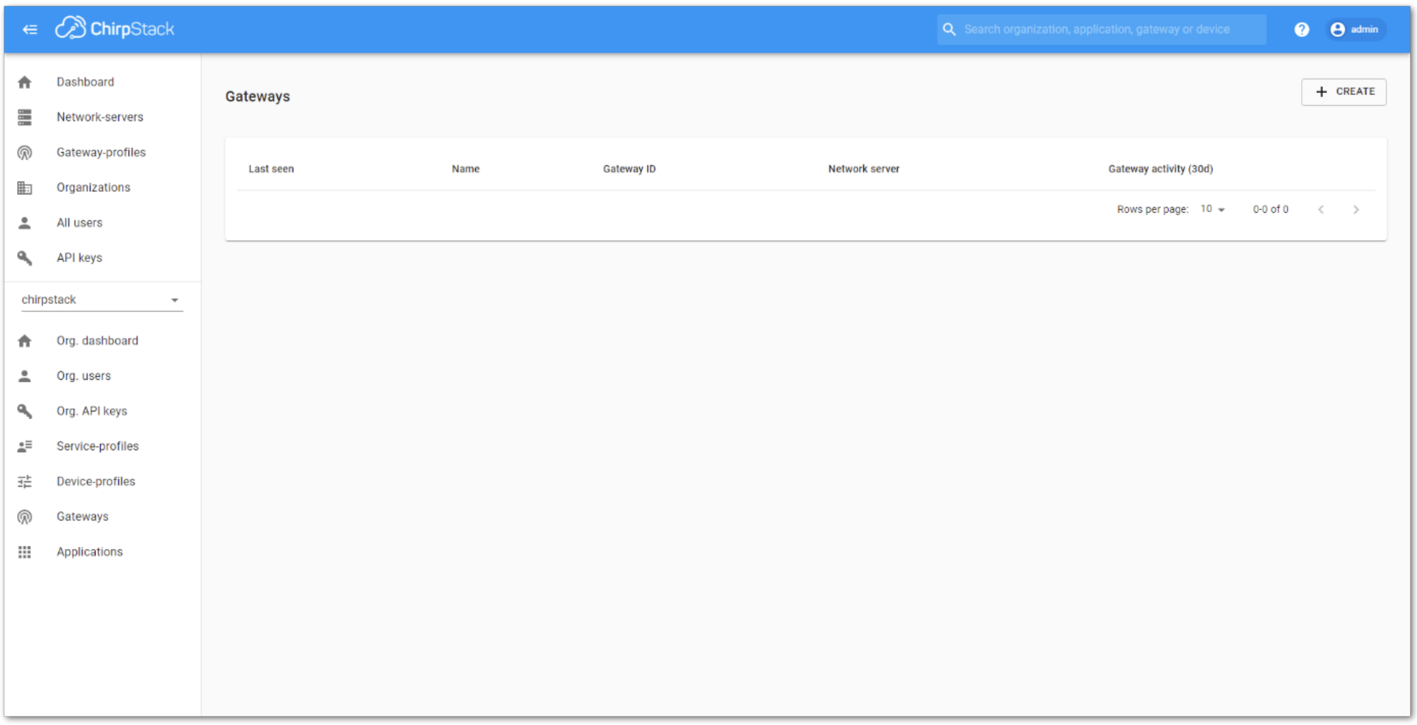
No data available.

Gateways

No data available.

Figure 36: ChirpStack dashboard

3. Head to **Gateways**, on the left pane.



Gateways

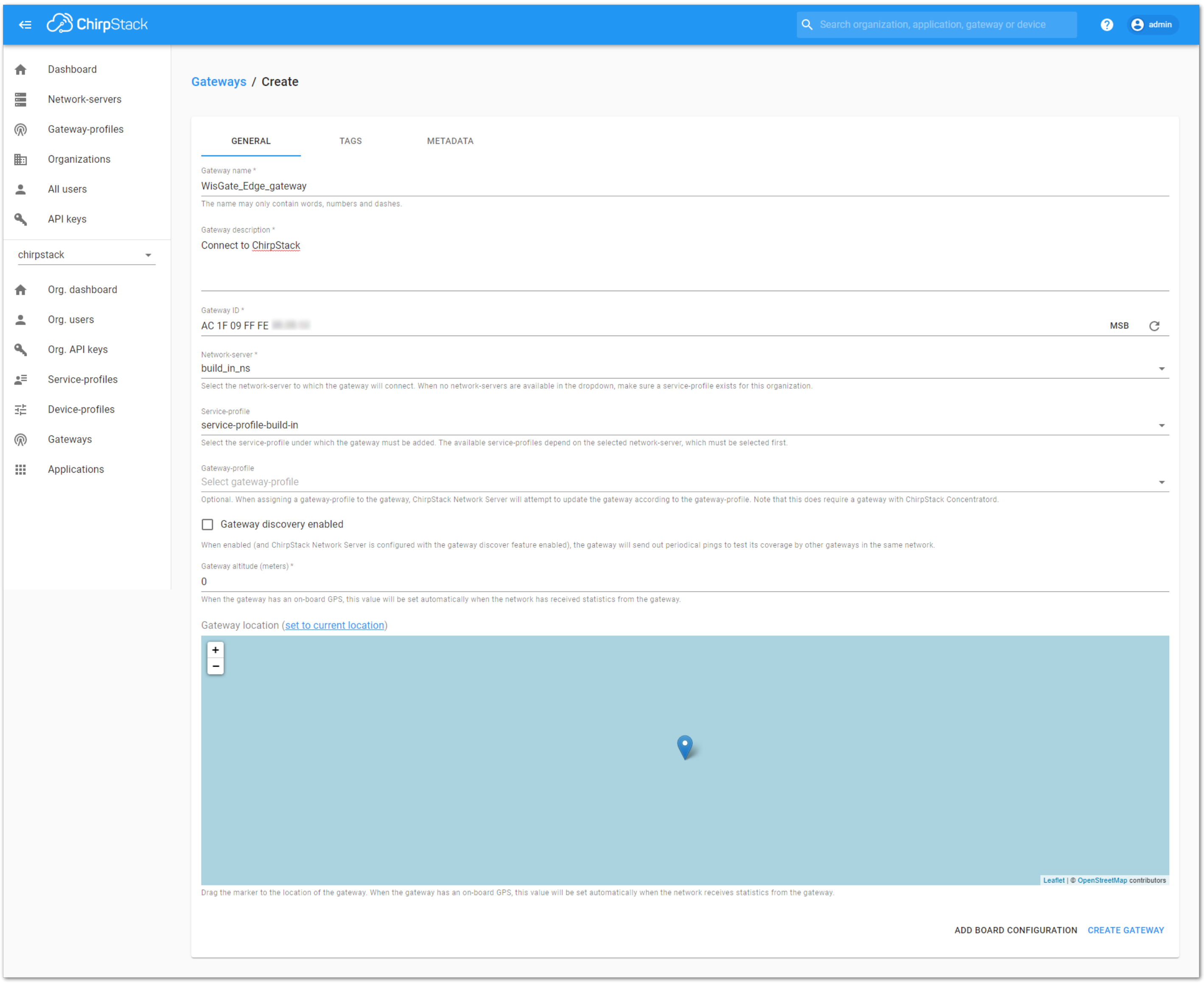
Last seen	Name	Gateway ID	Network server	Gateway activity (30d)

Rows per page: 10 0-0 of 0

Figure 37: Gateway list

4. By default, no gateways are registered. To register one, click **+ Create**.

5. In the **General menu**, you need to set the gateway parameters.



Gateways / Create

Gateway name *
WisGate_Edge_gateway
The name may only contain words, numbers and dashes.

Gateway description *
Connect to ChirpStack

Gateway ID *
AC 1F 09 FF FE [redacted] MSB

Network-server *
build_in_ns
Select the network-server to which the gateway will connect. When no network-servers are available in the dropdown, make sure a service-profile exists for this organization.

Service-profile
service-profile-build-in
Select the service-profile under which the gateway must be added. The available service-profiles depend on the selected network-server, which must be selected first.

Gateway-profile
Select gateway-profile
Optional. When assigning a gateway-profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway-profile. Note that this does require a gateway with ChirpStack Concentratord.

☐ Gateway discovery enabled
When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical pings to test its coverage by other gateways in the same network.

Gateway altitude (meters) *
0
When the gateway has an on-board GPS, this value will be set automatically when the network has received statistics from the gateway.

Gateway location (set to current location)

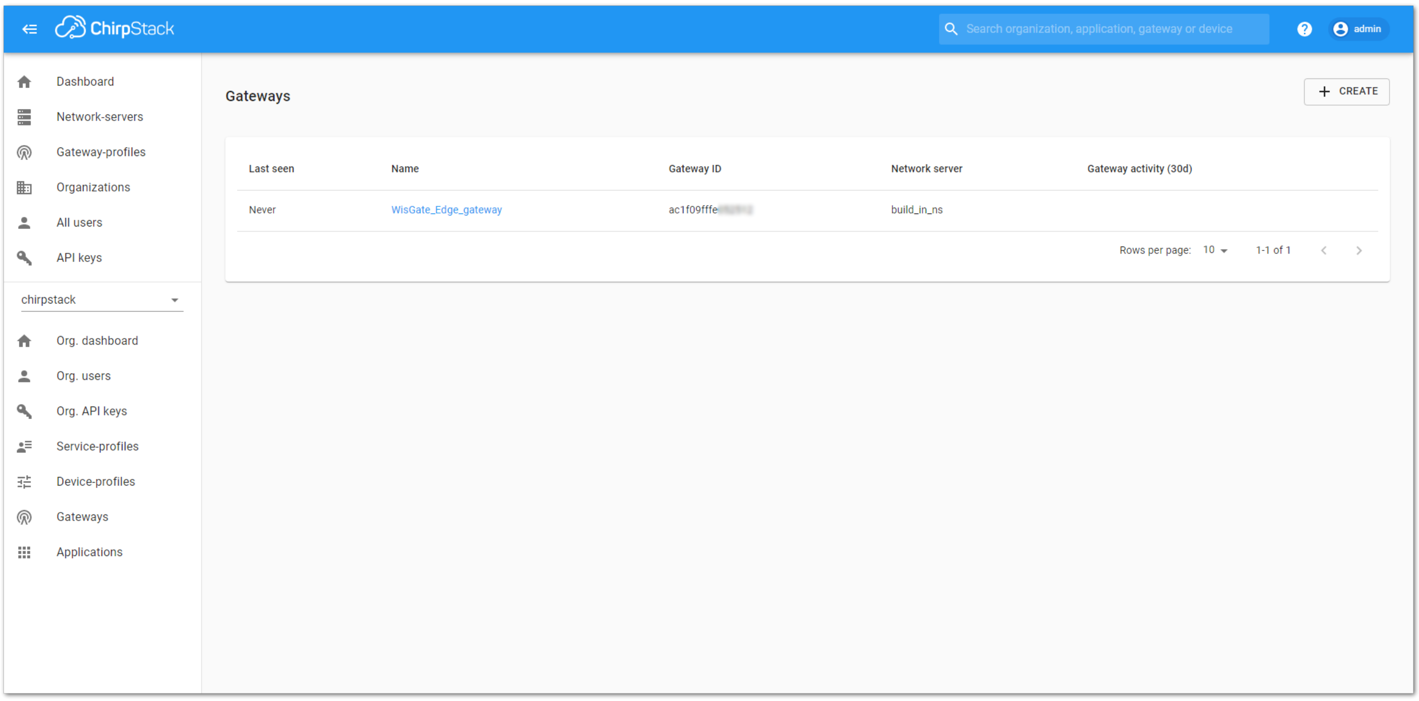
Drag the marker to the location of the gateway. When the gateway has an on-board GPS, this value will be set automatically when the network receives statistics from the gateway.

Figure 38: Register the gateway

- **Gateway name** – unique name for the gateway on the Network server. The name may only contain words, numbers, and dashes.
- **Gateway description** – a brief description of the gateway.

- **Gateway ID** – the Extended Unique Identifier (EUI) of the gateway. The EUI can be found, on the Overview page of the Dashboard menu of the web UI of the gateway.
- **Network-server** - the network server to which the gateway will connect. When no network servers are available in the dropdown, make sure a service profile exists for this organization.
- **Service profile** - the service profile under which the gateway must be added. The available service profiles depend on the selected network server, which must be selected first.
- **Gateway profile** – this field is optional. When assigning a gateway profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway profile. Note that this does require a gateway with ChirpStack Concentratord.
- **Gateway discovery enabled** - When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical pings to test its coverage by other gateways in the same network.
- **Gateway attitude** - When the gateway has an onboard GPS, this value will be set automatically when the network has received statistics from the gateway.
- **Gateway location** – you can drag the marker to the location of the gateway. When the gateway has an onboard GPS, this value will be set automatically when the network receives statistics from the gateway.

6. Once, everything is set, click **Create gateway** to register the gateway. You will see the registered gateway in the **Gateway list**.



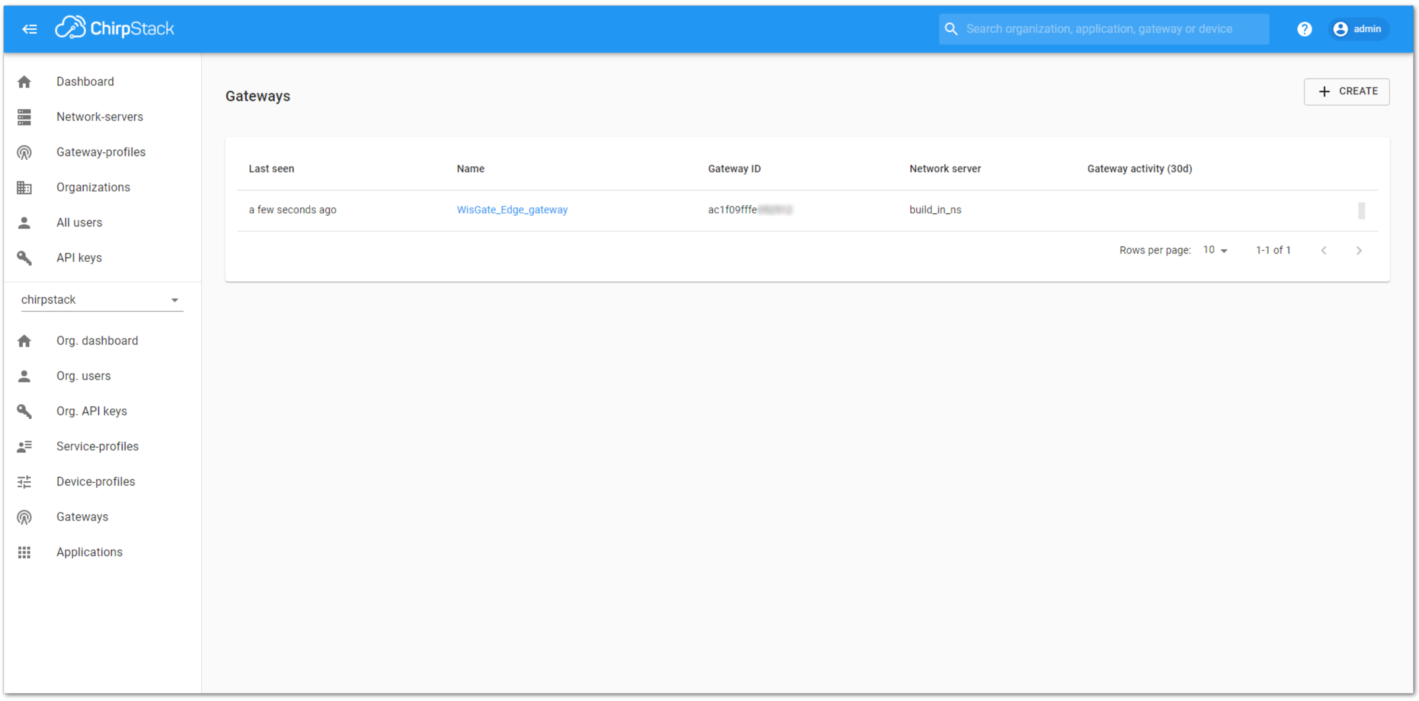
Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
Never	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10 1-1 of 1

Figure 39: Registered gateway

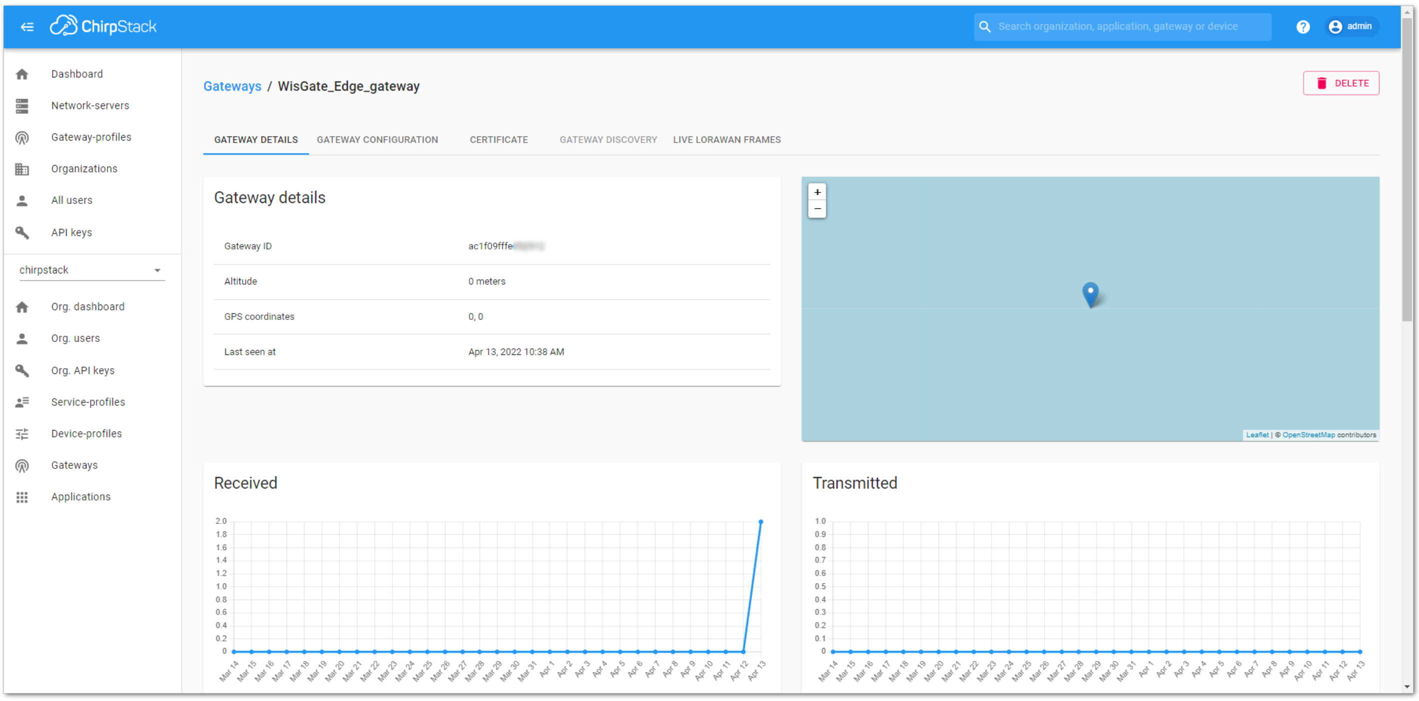
7. If everything is set correctly, the **Last seen** status will state a few seconds ago in a while.



Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Figure 40: Connect the gateway

You can click the gateway name to inspect the gateway traffic.



Gateway details

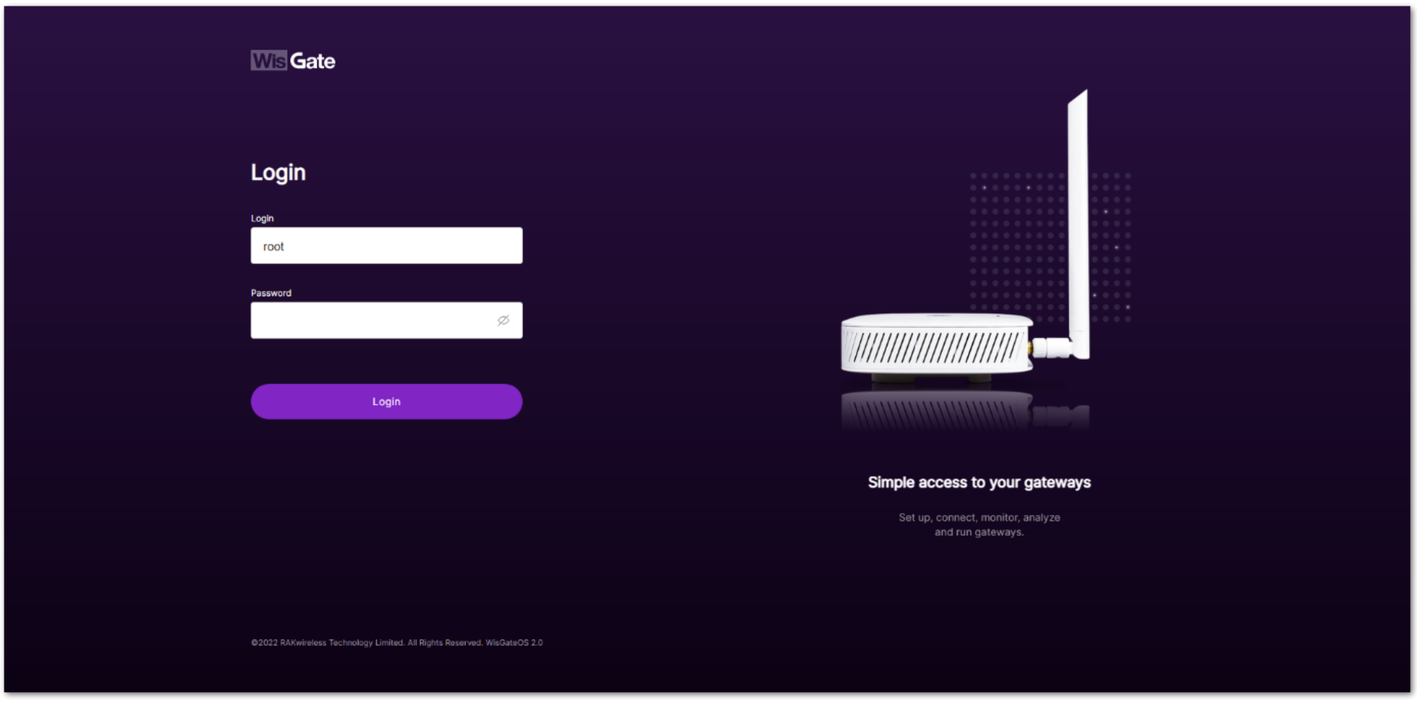
Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

Figure 41: Gateway details

Connect the Gateway via MQTT Bridge

In this method, you will configure the gateway's built-in gateway bridge to send data to the ChirpStack Broker.

1. Start by accessing the gateway. To access the gateway check the Access the gateway.



WisGate

Login

Login: root

Password:

Login

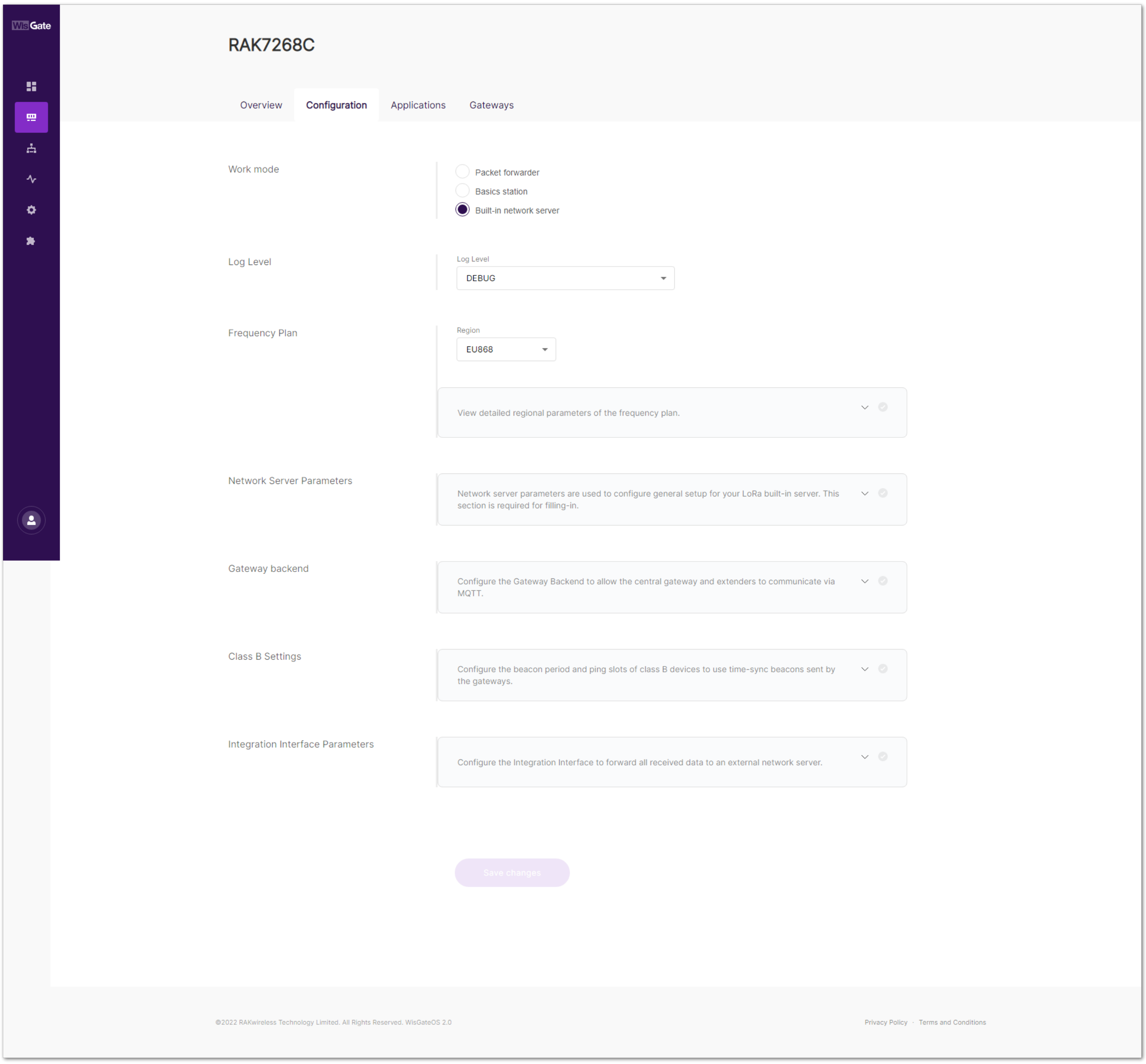
Simple access to your gateways

Set up, connect, monitor, analyze and run gateways.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 42: Login page

2. Login using the set credentials you have set in the **Access the gateway**.
3. On the left side, head to **LoRa**. By default, the gateway is configured to work as a **Built-in network server**.



RAK7268C

Overview | Configuration | Applications | Gateways

Work mode
○ Packet forwarder
○ Basics station
● Built-in network server

Log Level
Log Level: DEBUG

Frequency Plan
Region: EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters
Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

Gateway backend
Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

Class B Settings
Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

Integration Interface Parameters
Configure the Integration Interface to forward all received data to an external network server.

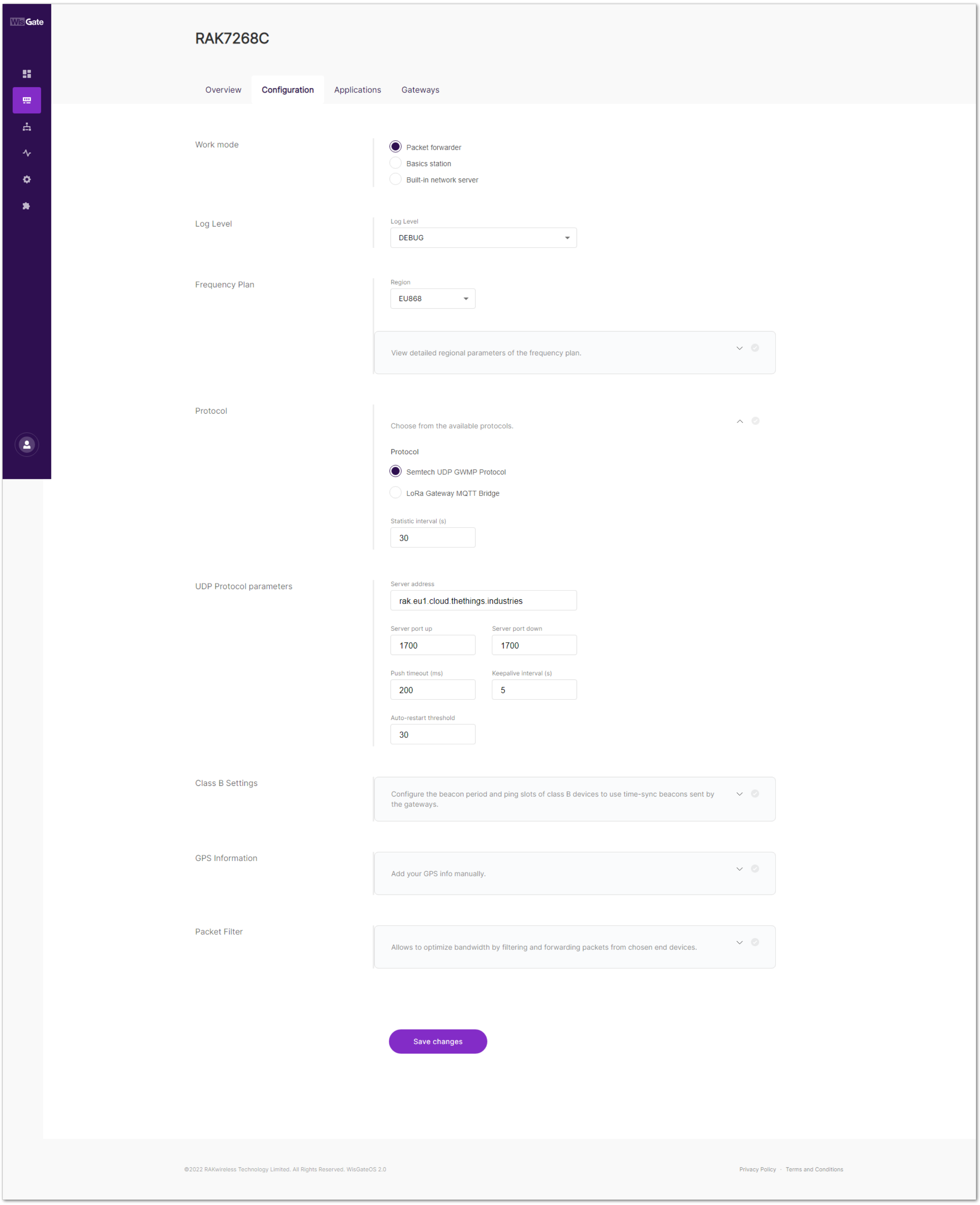
Save changes

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Privacy Policy · Terms and Conditions

Figure 43: Network server settings

4. From **Work Mode**, select **Packet forwarder**. Click **Choose from the available protocols** to expand the Packet forwarder settings.



WisGate

RAK7268C

Overview | Configuration | Applications | Gateways

Work mode

- ● Packet forwarder
- ○ Basics station
- ○ Built-in network server

Log Level

Log Level: DEBUG

Frequency Plan

Region: EU868

View detailed regional parameters of the frequency plan.

Protocol

Choose from the available protocols.

Protocol

- ● Semtech UDP GWMP Protocol
- ○ LoRa Gateway MQTT Bridge

Statistic interval (s): 30

UDP Protocol parameters

Server address: rak.eu1.cloud.thethings.industries

Server port up: 1700

Server port down: 1700

Push timeout (ms): 200

Keepalive interval (s): 5

Auto-restart threshold: 30

Class B Settings

Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

GPS Information

Add your GPS info manually.

Packet Filter

Allows to optimize bandwidth by filtering and forwarding packets from chosen end devices.

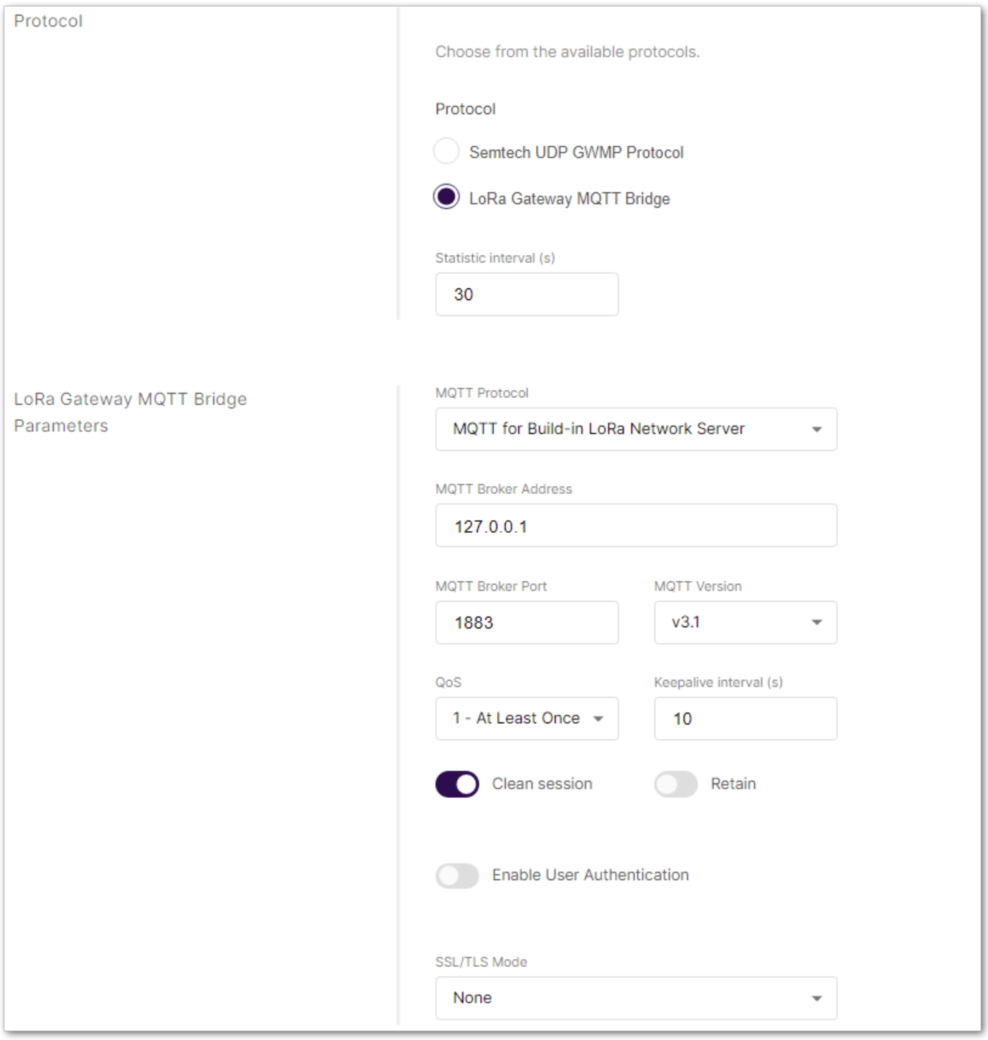
Save changes

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Privacy Policy · Terms and Conditions

Figure 44: Setting Packet Forwarder Mode

5. By default, when Packet Forwarder mode is chosen, the **Semtech UDP GWMP Protocol** is selected. To use the built-in gateway bridge, from the **Protocol** select **LoRa Gateway MQTT Bridge**.



Protocol

Choose from the available protocols.

Protocol

○ Semtech UDP GWMP Protocol

● LoRa Gateway MQTT Bridge

Statistic interval (s): 30

LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for Build-in LoRa Network Server

MQTT Broker Address: 127.0.0.1

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session: on

Retain: off

Enable User Authentication: off

SSL/TLS Mode: None

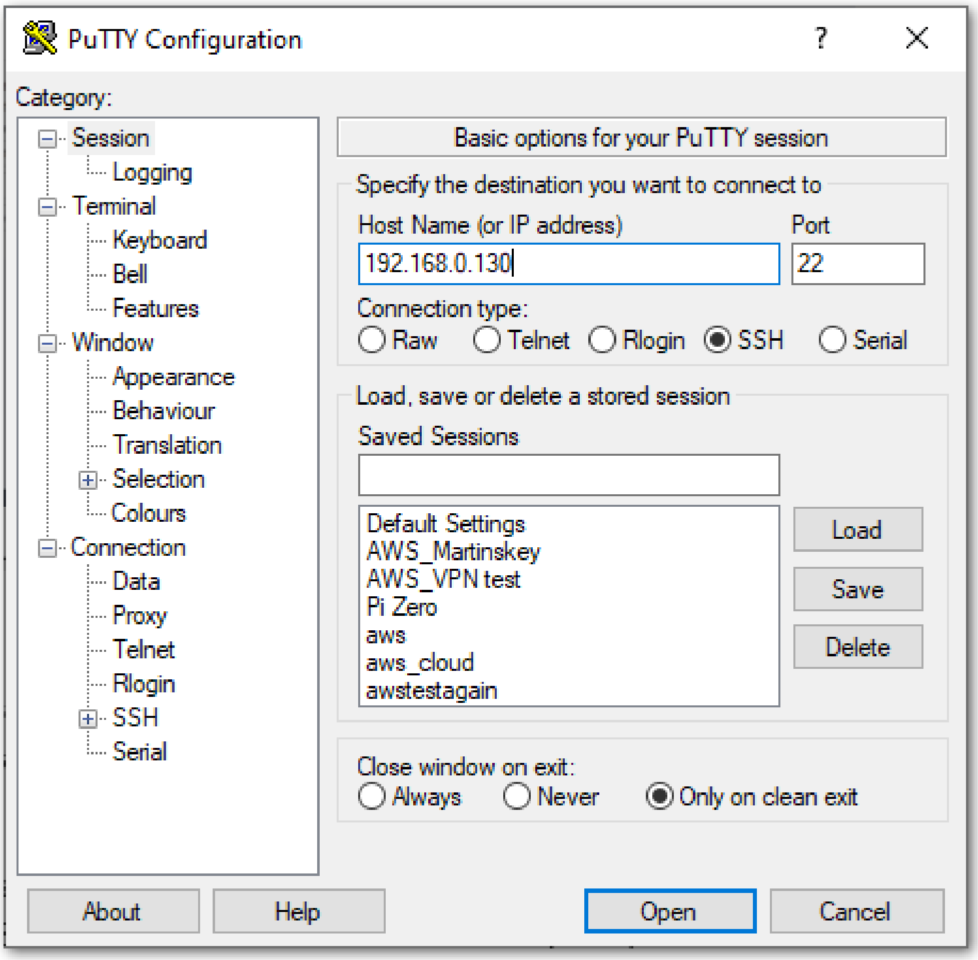
Figure 45: LoRa Gateway MQTT Bridge

The latest ChirpStack version supports both **MQTT for ChirpStack 3.x (JSON)** and **MQTT for ChirpStack 3.x (PROTOBUF).**

> 📝 **NOTE**
>
> If you want to use JSON protocol, you need to change the payload marshaler in the gateway bridge .toml file to “JSON”.

To configure the payload marshaler you need to access the configuration file of the gateway bridge. You can access it with the configuration file of the gateway bridge. You can also access it with an SSH connection. In this case, you will use the SSH client PuTTY to access the configuration files.

6. Access the ChirpStack via PuTTY. You need to type the address of the ChirpStack server in the **Host name** field and click **Open**. In this case on IP 192.168.0.130.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.130

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

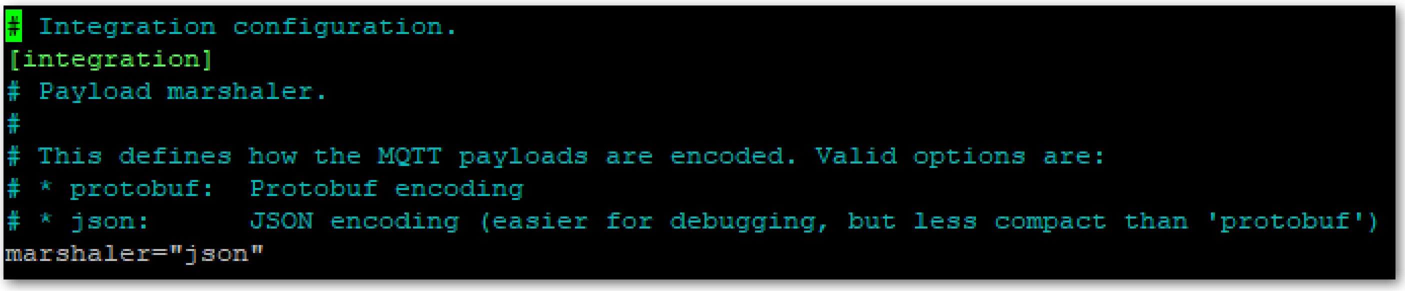
Saved Sessions

Default Settings
AWS_Martinskey
AWS_VPN test
Pi Zero
aws
aws_cloud
awstestagain

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 46: PuTTY client

7. In the `/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml`, find the **Integration** section and change the marshaler to JSON.



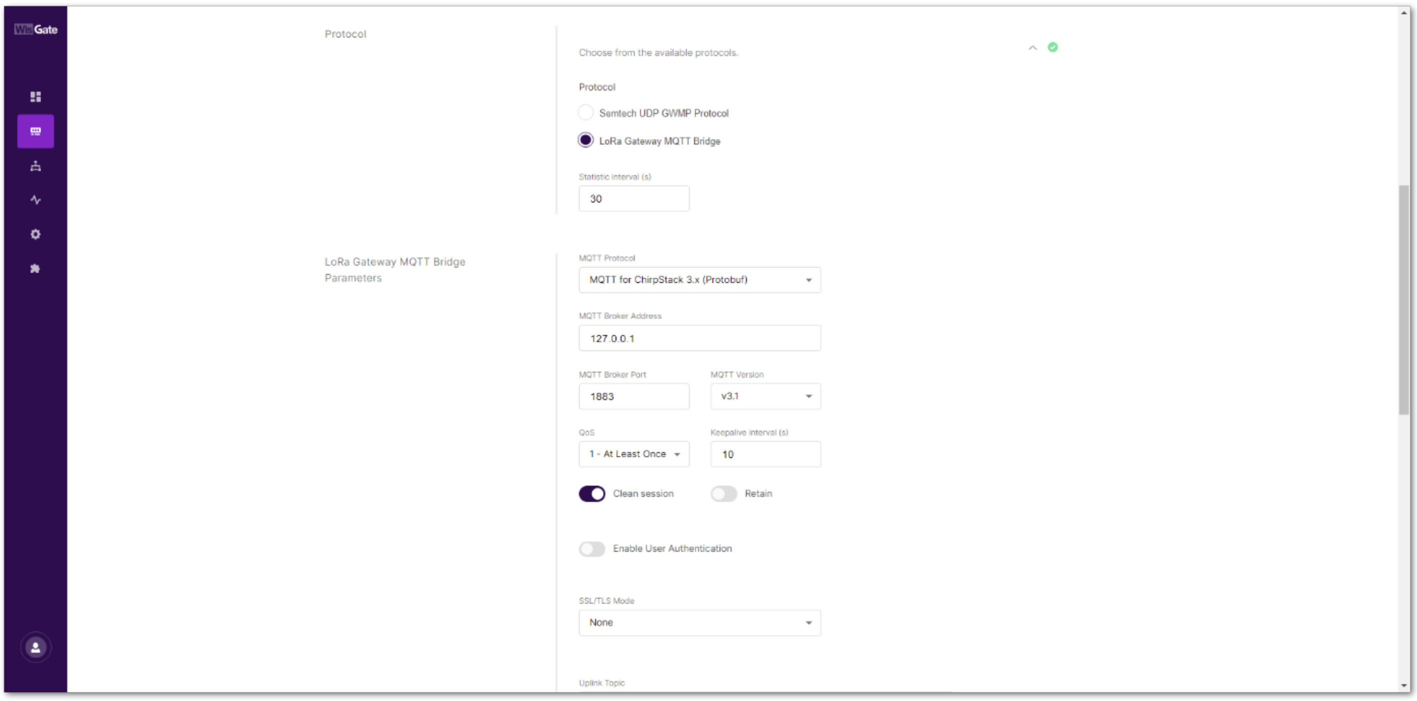
```
# Integration configuration.
[integration]
# Payload marshaler.
#
# This defines how the MQTT payloads are encoded. Valid options are:
# * protobuf:  Protobuf encoding
# * json:      JSON encoding (easier for debugging, but less compact than 'protobuf')
marshaler="json"
```

Figure 47: Payload Marshaler

8. Save and exit the file.

9. However, if you are using an earlier version of ChirpStack (V2), you will need to select **MQTT for ChirpStack 2. x**. The option **MQTT for Embedded RAK Network Server** is for a mesh network, where one gateway plays the role of a network server.

For this example, you will choose **MQTT for ChirpStack 3. x (PROTOBUF).**



Protocol

Choose from the available protocols.

Protocol

○ Semtech UDP GWMP Protocol

● LoRa Gateway MQTT Bridge

Statistic interval (s): 30

LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for ChirpStack 3.x (Protobuf)

MQTT Broker Address: 127.0.0.1

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session (on)

Retain (off)

Enable User Authentication (off)

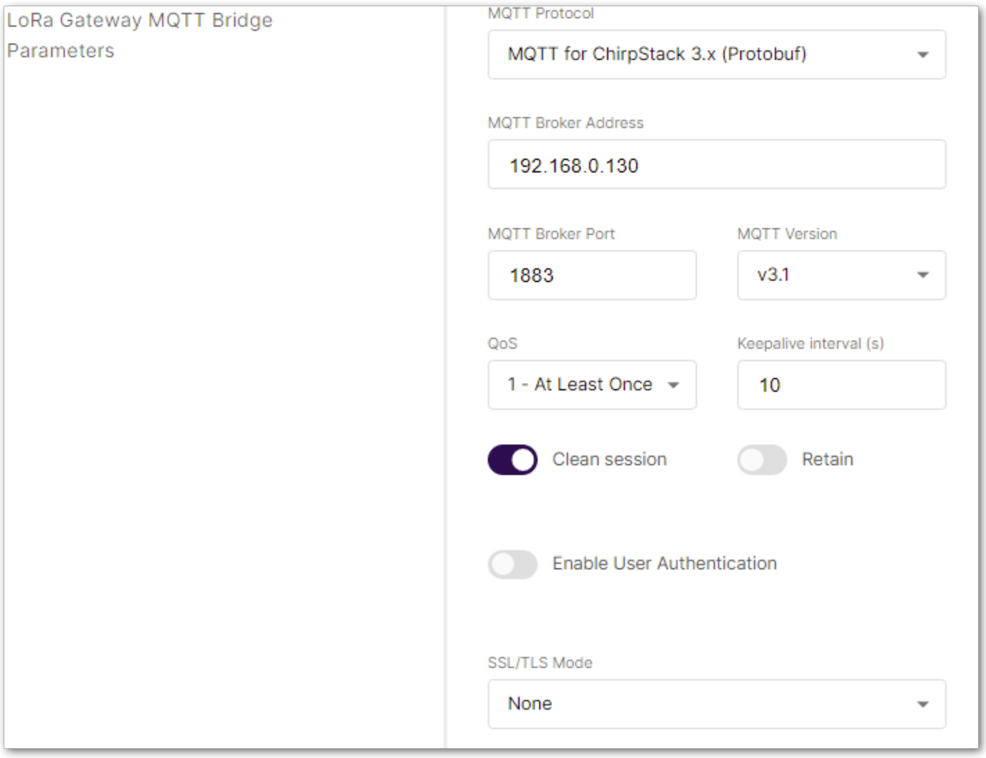
SSL/TLS Mode: None

Uplink Topic

Figure 48: MQTT for Chirpstack Protocol

10. By default, the built-in gateway bridge is pointed to the local Broker (127.0.0.1). To point the gateway to the ChirpStack network, you need to set the ChirpStack Broker address in the **MQTT Broker Address** field.

In this case, the ChirpStack is installed locally on an Ubuntu machine on IP 192.168.0.130 (yours will be different). The default port that the MQTT Broker uses is 1883.



LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for ChirpStack 3.x (Protobuf)

MQTT Broker Address: 192.168.0.130

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session (on)

Retain (off)

Enable User Authentication (off)

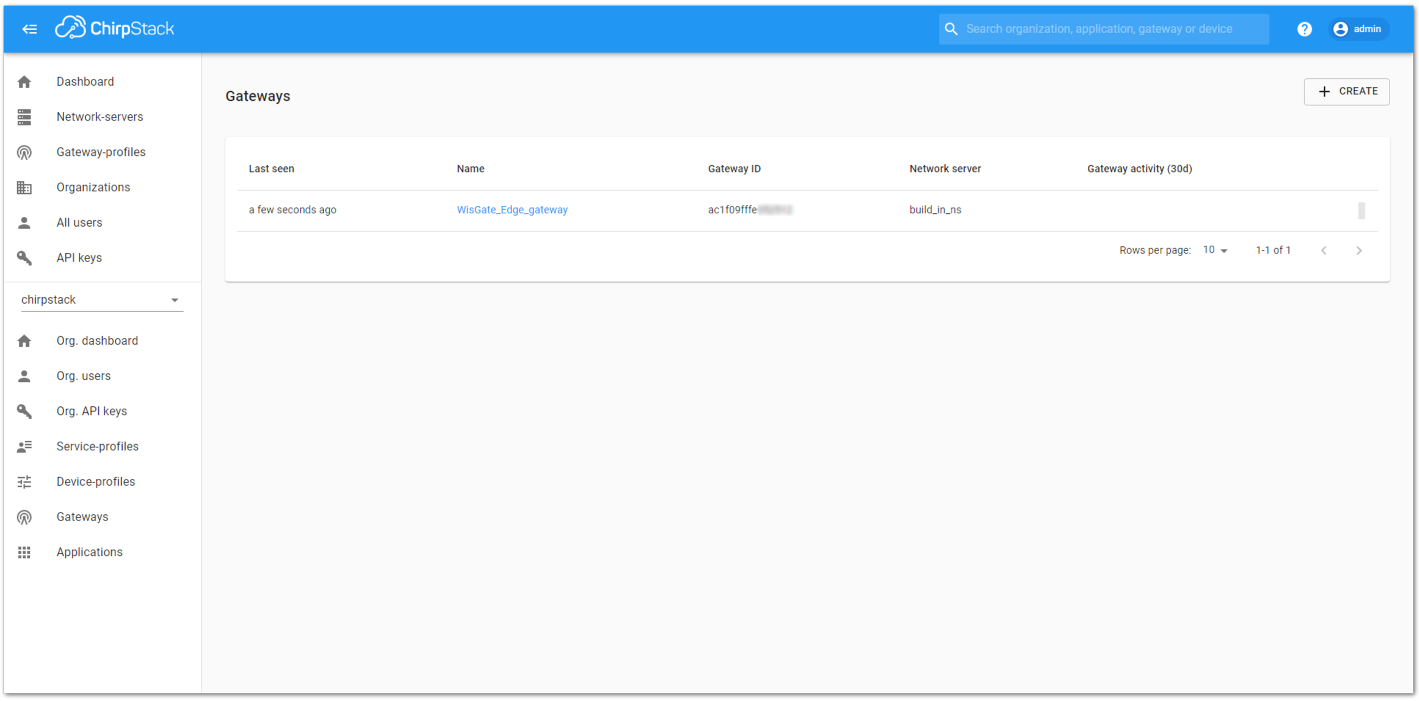
SSL/TLS Mode: None

Figure 49: Configuring Packet Forwarder to ChirpStack

11. Click **Save changes** to save the changes.

12. Now you need to register the gateway in the ChirpStack Network server. When Packet Forwarder mode is chosen, the **Semtech UDP GWMP Protocol** is selected by default. To register the gateway in ChirpStack, see **Registering gateway in ChirpStack Network server** section.

13. If everything is set correctly, the **Last seen** status will state **a few seconds ago**.



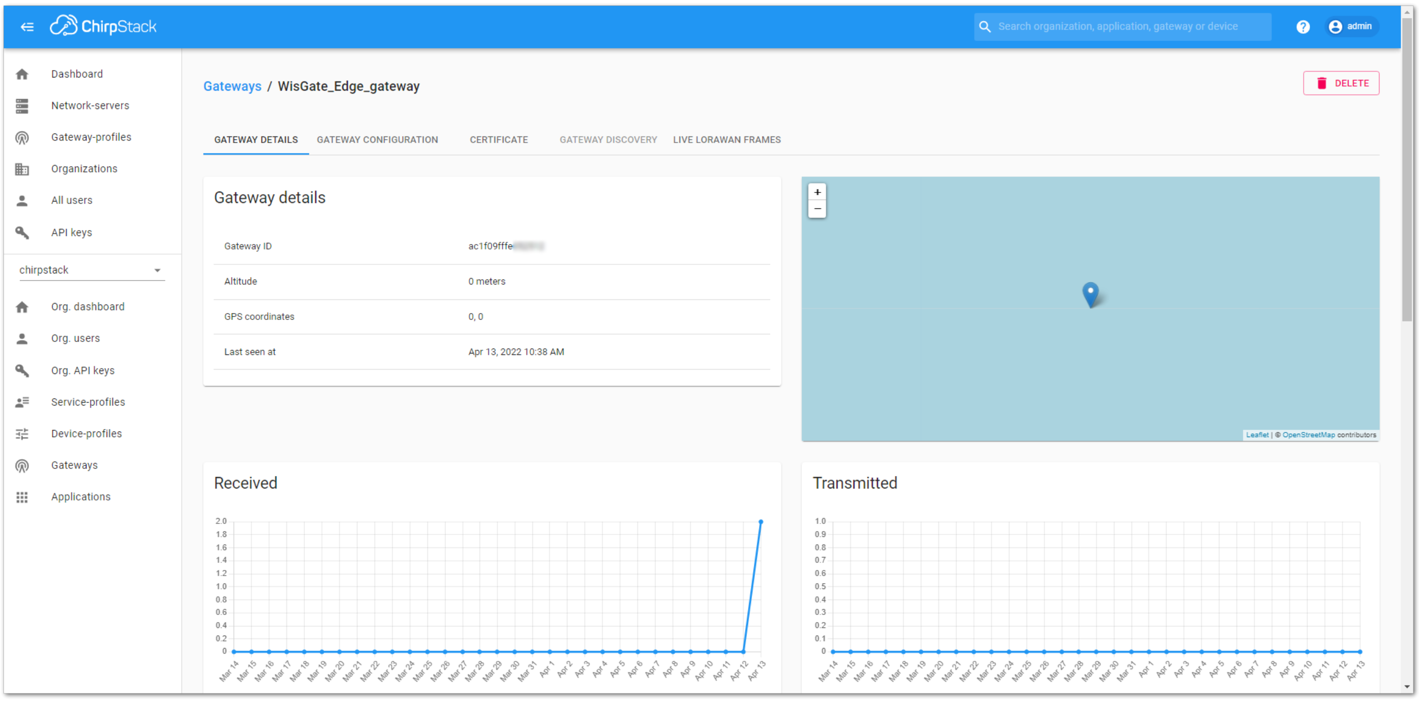
Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10 1-1 of 1

Figure 50: Registered gateway

You can click the gateway's name to inspect the gateway traffic.



Gateways / WisGate_Edge_gateway

Gateway details

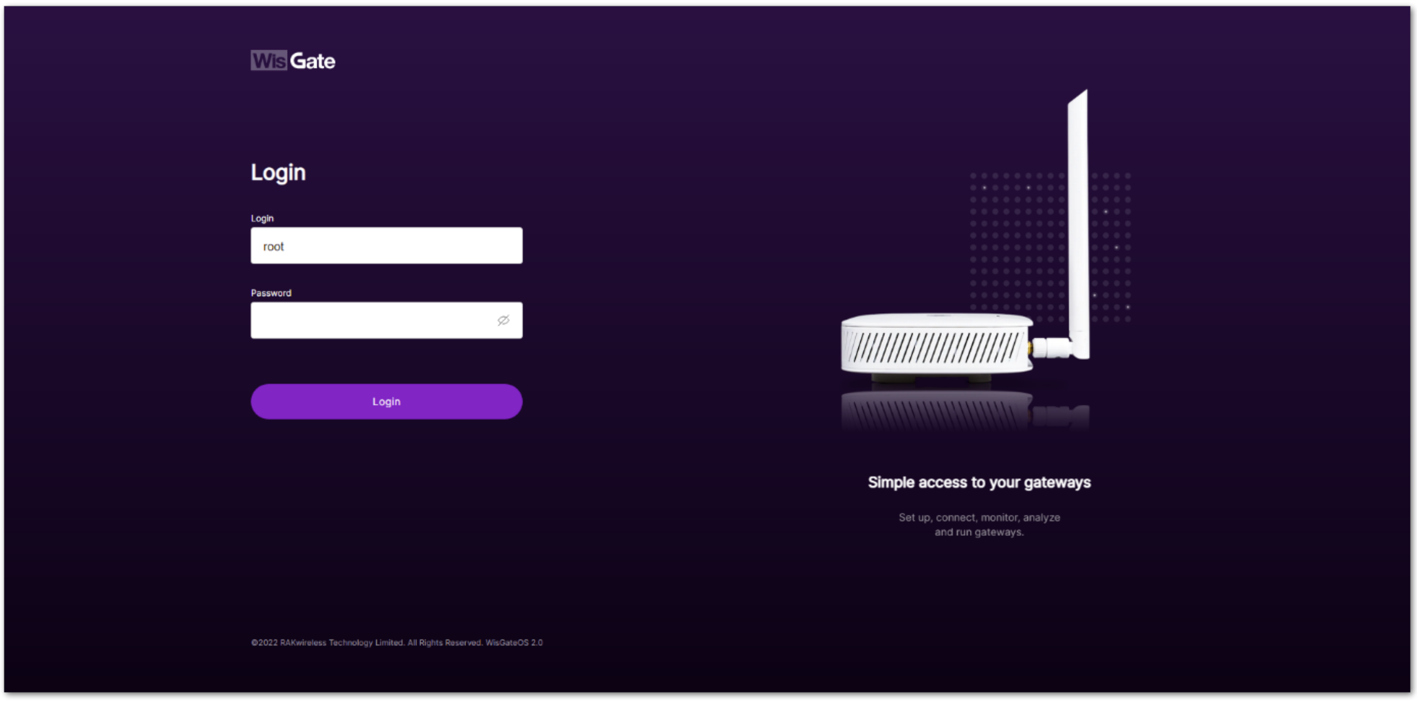
Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

Figure 51: Gateway details

Now your gateway is connected to the ChirpStack Network server.

Connecting the Gateway via Basics Station

1. In this method, you will connect the gateway to the **ChirpStack** via **Basics Station**. The **LoRa Basics™ Station** is an implementation of a LoRa packet forwarder.



WisGate

Login

Login

root

Password

Login

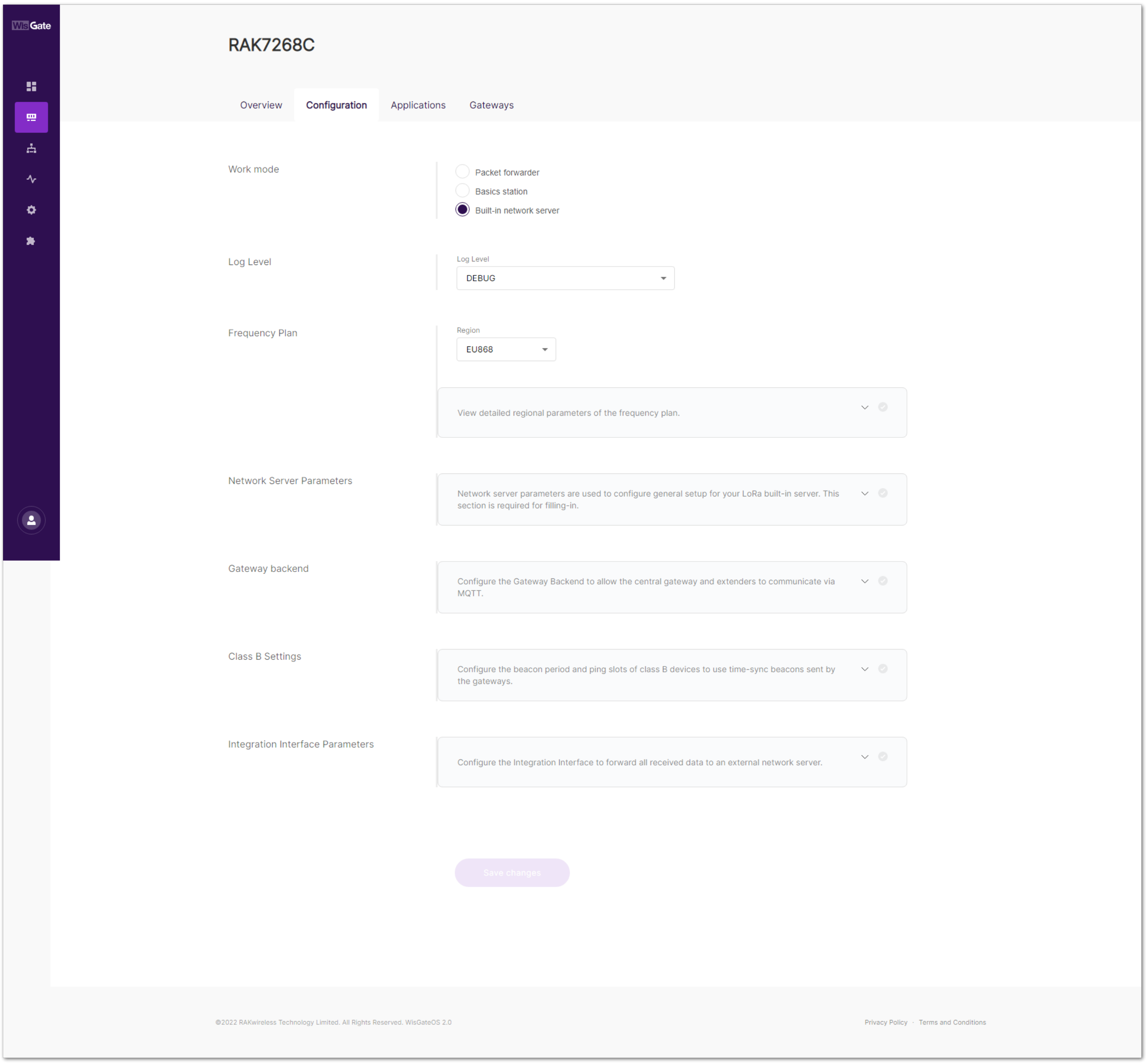
Simple access to your gateways

Set up, connect, monitor, analyze and run gateways.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 52: Login page

2. Login using the set credentials you have set in the **Access the gateway**.
3. n the left side, head to LoRa. By default, the gateway is configured to work as a Built-in network server.



WisGate

RAK7268C

Overview | Configuration | Applications | Gateways

Work mode

○ Packet forwarder
○ Basics station
● Built-in network server

Log Level

Log Level: DEBUG

Frequency Plan

Region: EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters

Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

Gateway backend

Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

Class B Settings

Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

Integration Interface Parameters

Configure the Integration Interface to forward all received data to an external network server.

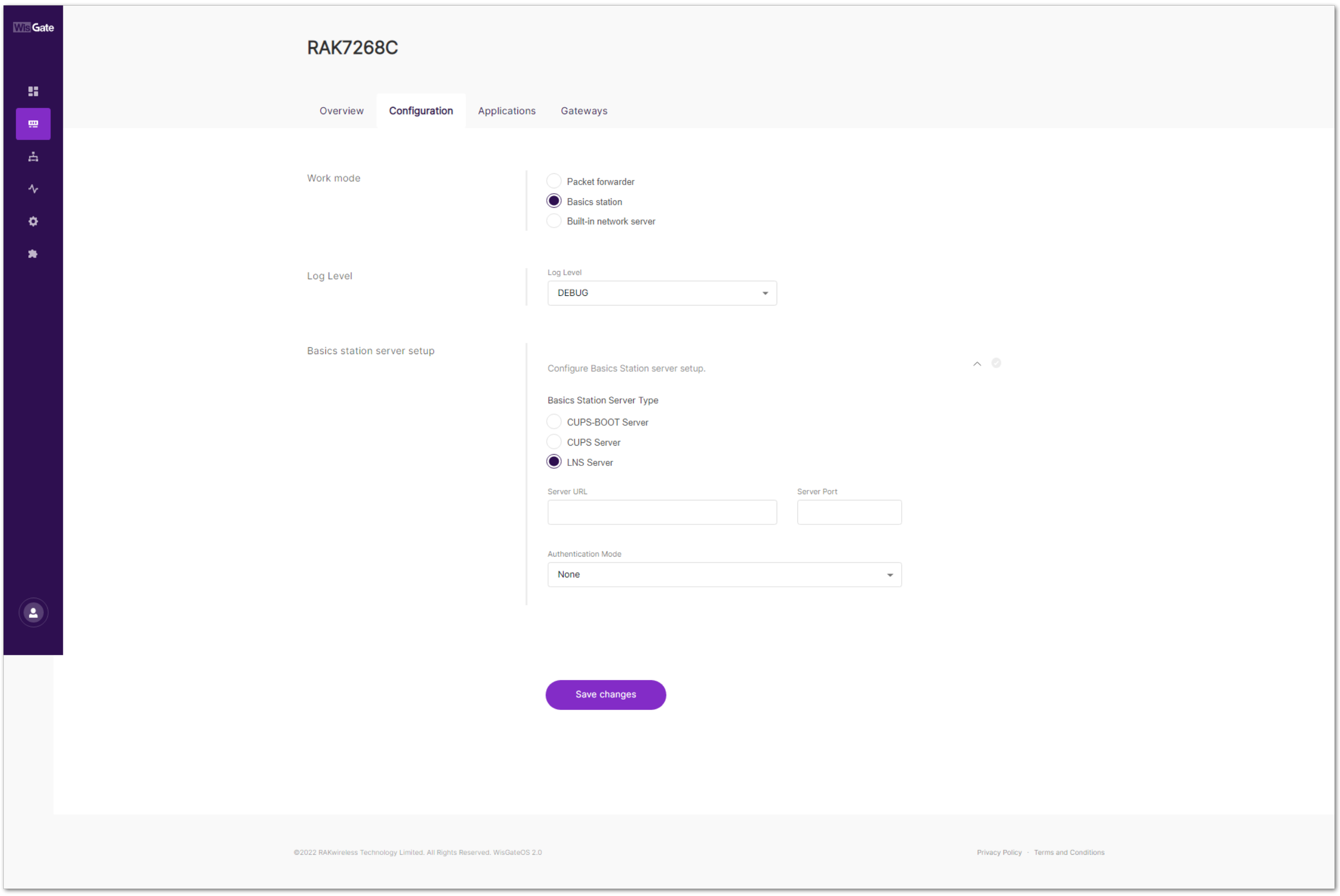
Save changes

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Privacy Policy · Terms and Conditions

Figure 53: Network Server Settings

4. For **Work Mode**, select **Basics station** and click **Configure Basics Station** server setup to expand the Basics Station settings.



RAK7268C

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Basics station server setup
Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

Server URL

Server Port

Authentication Mode: None

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

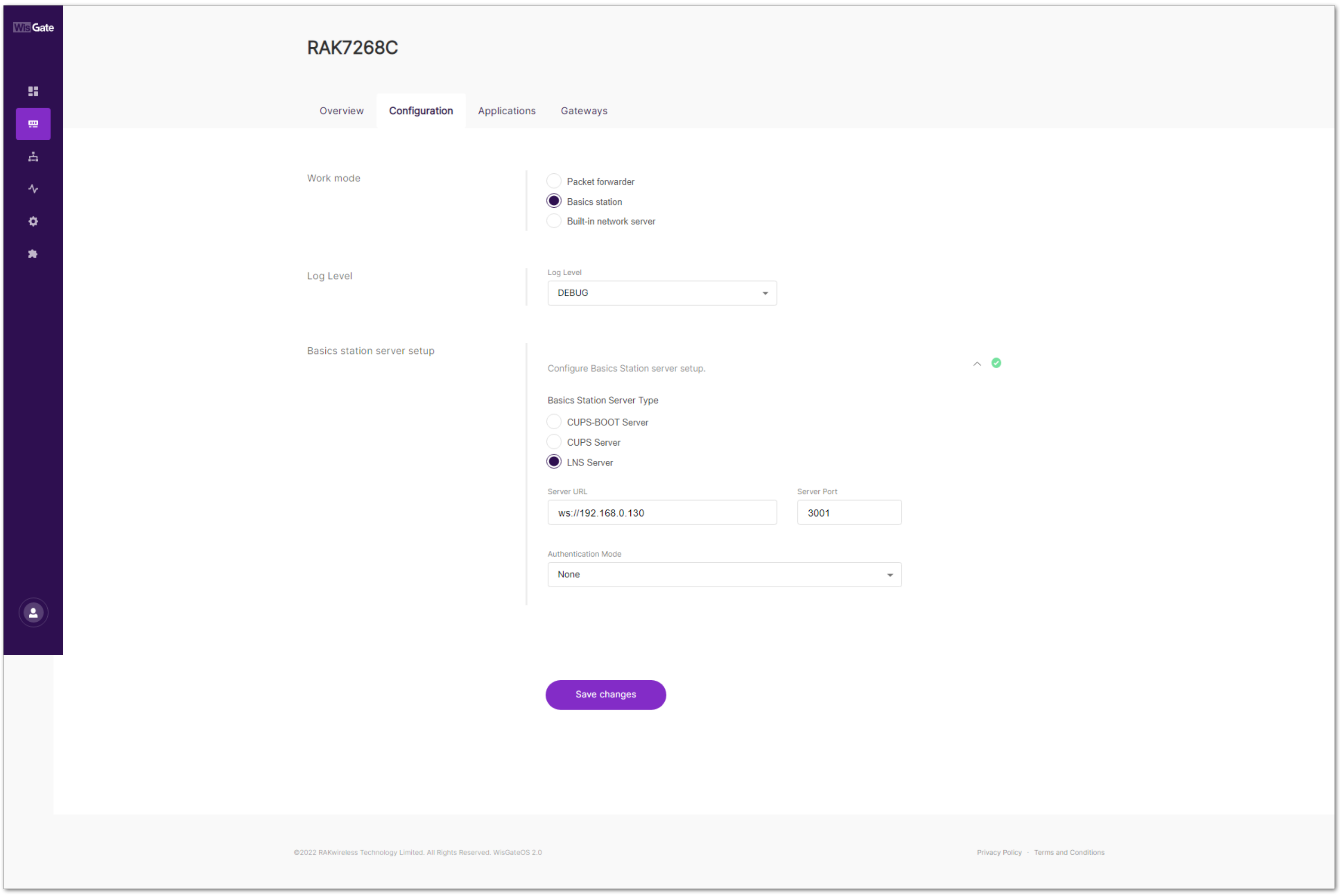
Figure 54: Setting Basics Station mode

Here you need to point the gateway to the ChirpStack Network server:

- **Server** – For server, choose LNS Server.
- **URI** – the address of the ChirpStack server. In this case, the ChirpStack is installed locally on an Ubuntu machine on IP 192.168.0.130 (yours will be different). The URI will be `ws://192.168.0.130` .

> 📝 **NOTE**
>
> The URL starts with ws:// in case a plain text connection is used. Using the wss:// scheme will trigger a TLS connection based on the `tc.{cert,key,trust}` credentials set.

- **Port** – the port to which the Websocket listens. The port is 3001.
- **Authentication Mode** – authentication for the ChirpStack server. For this case, you will use no authentication.



RAK7268C

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Basics station server setup
Configure Basics Station server setup.

Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server

Server URL: ws://192.168.0.130
Server Port: 3001

Authentication Mode: None

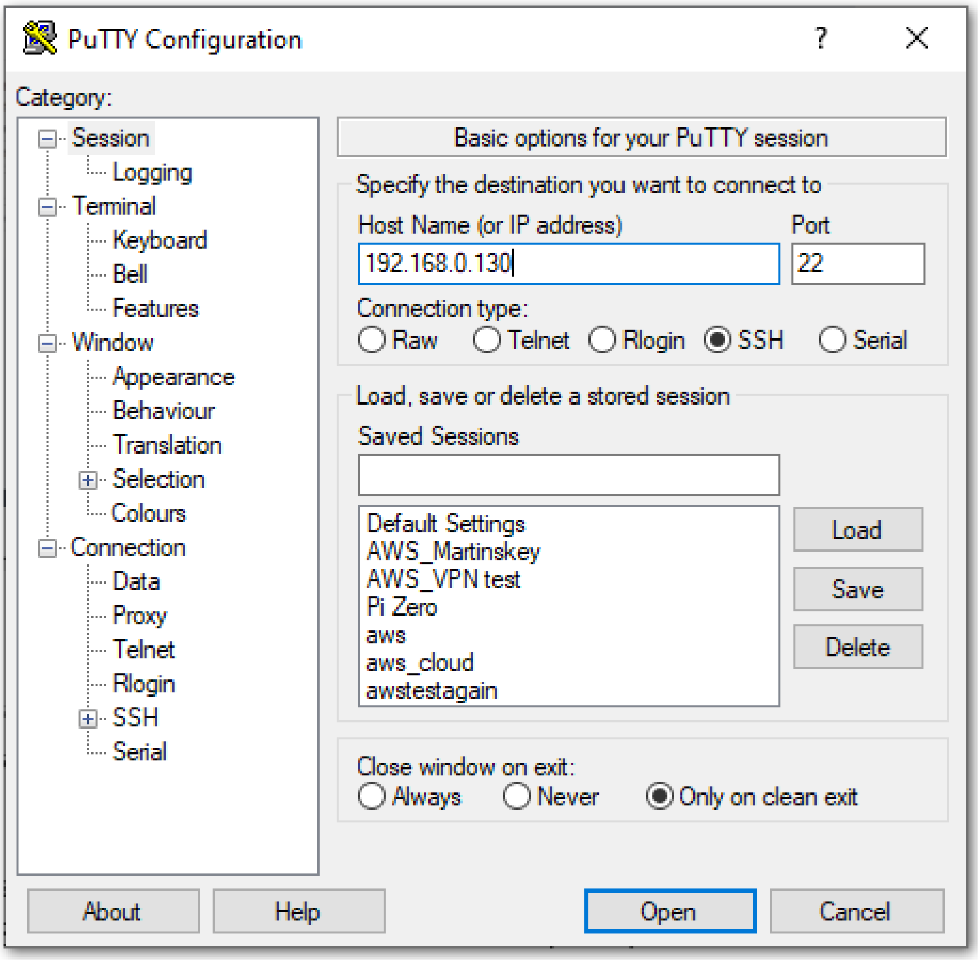
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 55: Configuring Basics Station to ChirpStack

5. Click **Save changes** to save the changes.

Now your gateway is configured to work as Basics Station, and it is pointed to the ChirpStack gateway bridge. The default installation of the ChirpStack setups the backend configuration of the ChirpStack gateway bridge to `semtech_udp` .

6. To configure the backend of the ChirpStack gateway bridge, you need to access the configuration file of the bridge. To access it, you will need an SSH terminal. In this case, you will use a PuTTY client.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.130

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions

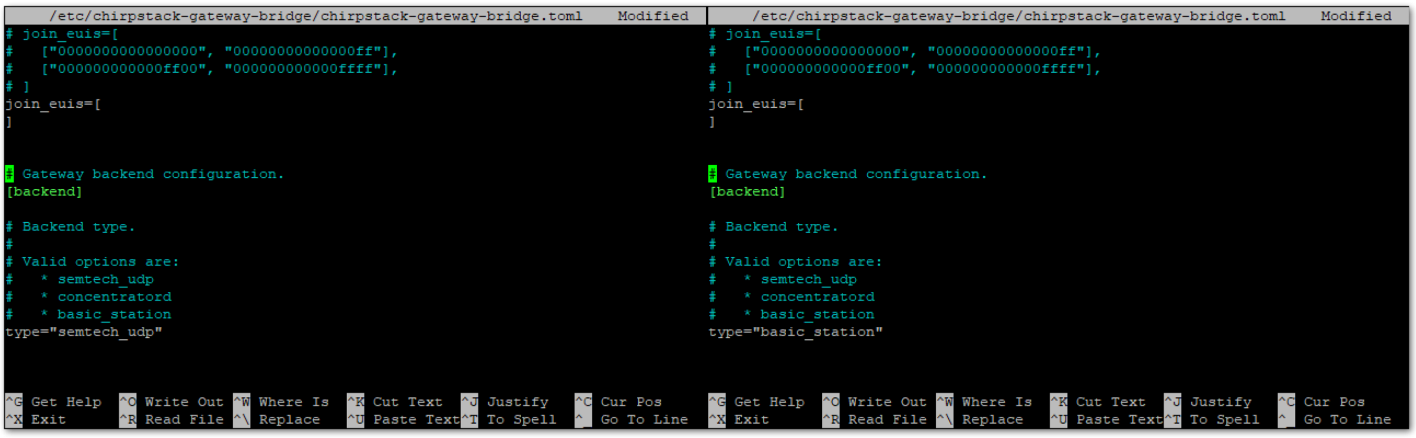
Default Settings
AWS_Martinskey
AWS_VPN test
Pi Zero
aws
aws_cloud
awstestagain

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 56: PuTTY client

Copy the configuration file's text in the ChirpStack Gateway bridge webpage and place it in `/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml`.

7. In the file, find the **Gateway backend configuration** paragraph and replace the type with `basic_station`.

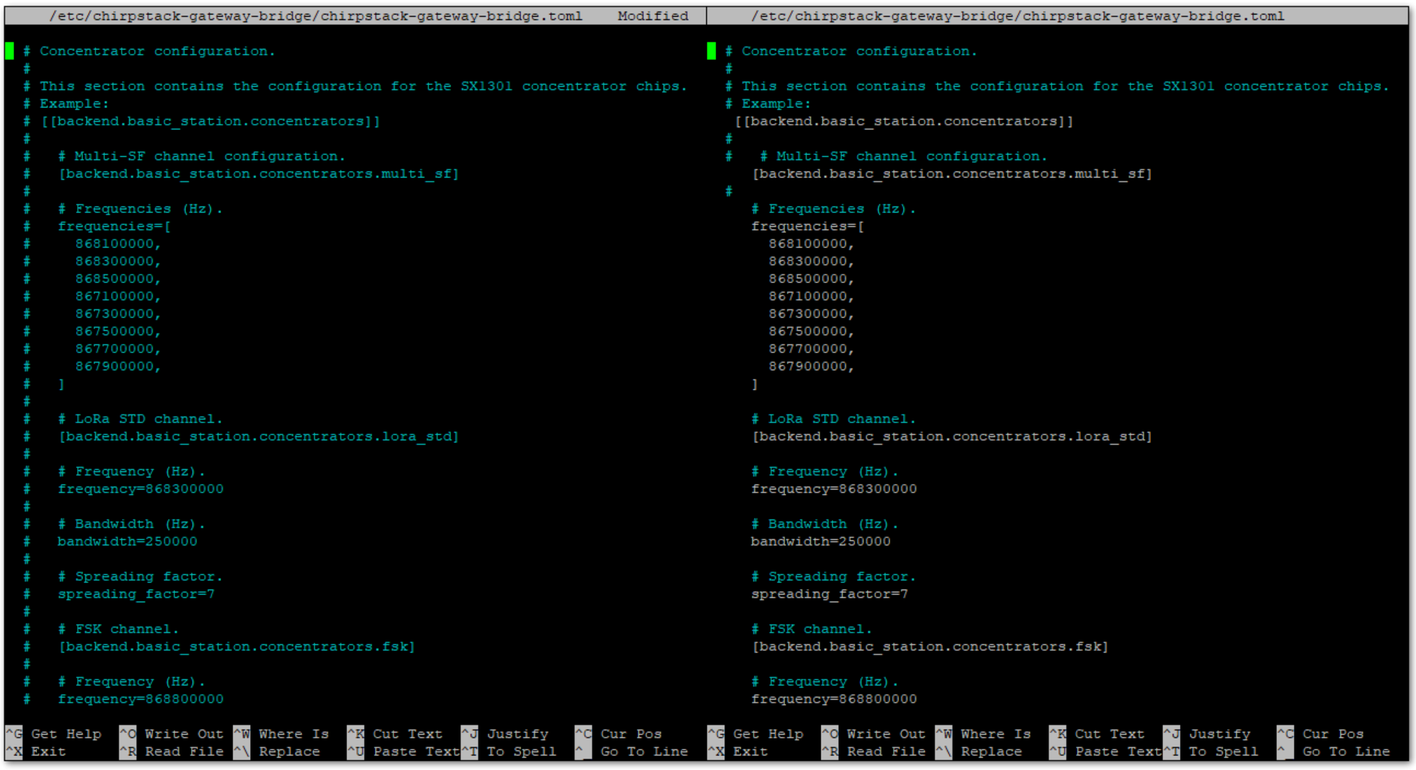


```
     /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml     Modified
# join_euis=[
#   ["0000000000000000", "00000000000000ff"],
#   ["000000000000ff00", "000000000000ffff"],
# ]
join_euis=[
]


# Gateway backend configuration.
[backend]

# Backend type.
#
# Valid options are:
#   * semtech_udp
#   * concentratord
#   * basic_station
type="semtech_udp"



^G Get Help   ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit       ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

```
     /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml     Modified
# join_euis=[
#   ["0000000000000000", "00000000000000ff"],
#   ["000000000000ff00", "000000000000ffff"],
# ]
join_euis=[
]


# Gateway backend configuration.
[backend]

# Backend type.
#
# Valid options are:
#   * semtech_udp
#   * concentratord
#   * basic_station
type="basic_station"



^G Get Help   ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit       ^R Read File ^\ Replace   ^U Paste Text^T To Spell  ^_ Go To Line
```

Figure 57: Configure gateway bridge type

8. Now scroll down until you find the **Concentrator configuration** paragraph and uncomment the following text as shown below. Uncommenting the text, enables the configuration for the SX1301 concentrator chips.



```
  /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml     Modified

# Concentrator configuration.
#
# This section contains the configuration for the SX1301 concentrator chips.
# Example:
# [[backend.basic_station.concentrators]]
#
#   # Multi-SF channel configuration.
#   [backend.basic_station.concentrators.multi_sf]
#
#   # Frequencies (Hz).
#   frequencies=[
#     868100000,
#     868300000,
#     868500000,
#     867100000,
#     867300000,
#     867500000,
#     867700000,
#     867900000,
#   ]
#
#   # LoRa STD channel.
#   [backend.basic_station.concentrators.lora_std]
#
#   # Frequency (Hz).
#   frequency=868300000
#
#   # Bandwidth (Hz).
#   bandwidth=250000
#
#   # Spreading factor.
#   spreading_factor=7
#
#   # FSK channel.
#   [backend.basic_station.concentrators.fsk]
#
#   # Frequency (Hz).
#   frequency=868800000

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text^T To Spell    ^_ Go To Line
```

```
  /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

# Concentrator configuration.
#
# This section contains the configuration for the SX1301 concentrator chips.
# Example:
 [[backend.basic_station.concentrators]]
#
#   # Multi-SF channel configuration.
   [backend.basic_station.concentrators.multi_sf]
#
   # Frequencies (Hz).
   frequencies=[
     868100000,
     868300000,
     868500000,
     867100000,
     867300000,
     867500000,
     867700000,
     867900000,
   ]

   # LoRa STD channel.
   [backend.basic_station.concentrators.lora_std]

   # Frequency (Hz).
   frequency=868300000

   # Bandwidth (Hz).
   bandwidth=250000

   # Spreading factor.
   spreading_factor=7

   # FSK channel.
   [backend.basic_station.concentrators.fsk]

   # Frequency (Hz).
   frequency=868800000

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text^T To Spell    ^_ Go To Line
```

Figure 58: Configure gateway bridge backend

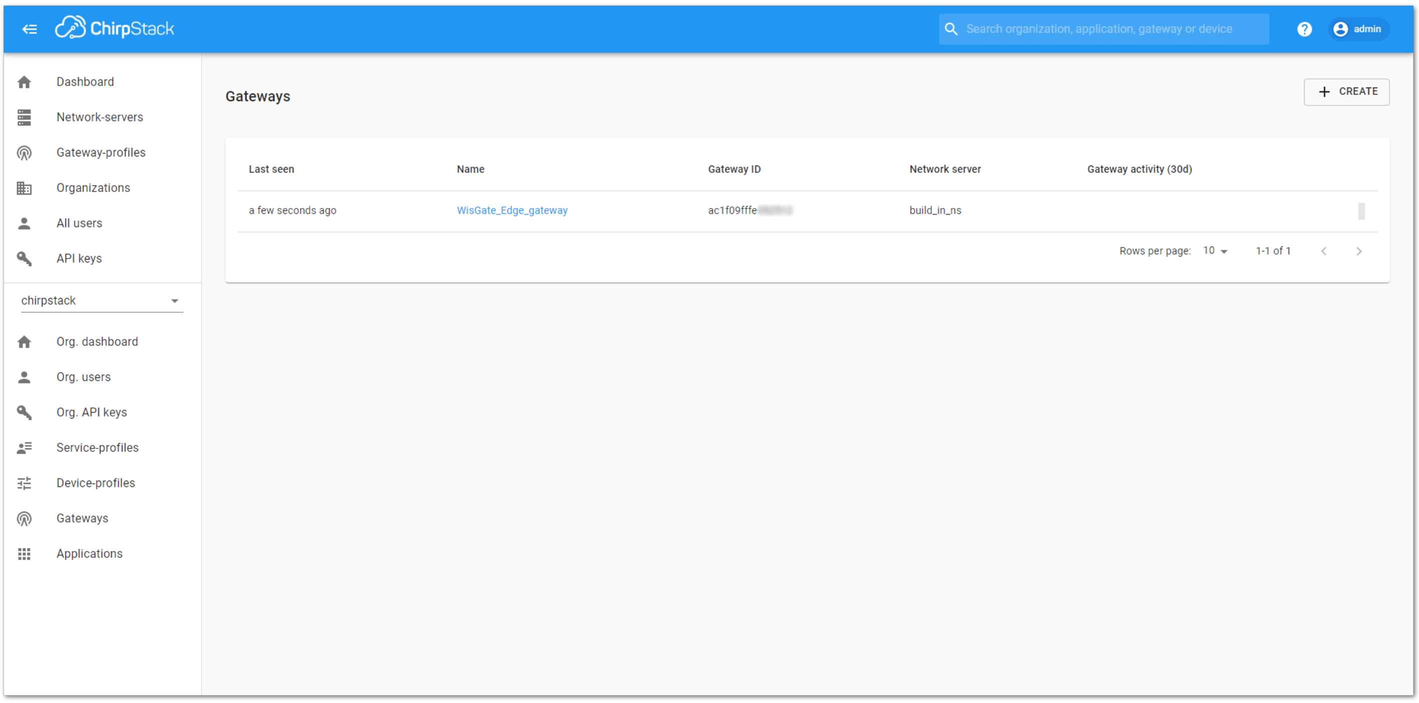
9. Save and exit the `.toml` file and restart the gateway bridge service to apply the changes by restarting the gateway bridge service with the following command:

```
sudo systemctl restart chirpstack-gateway-bridge.service
```

Now the ChirpStack backend configuration is set to basics station.

10. Then, you need to register the gateway in the ChirpStack Network server. To register the gateway in ChirpStack, see **Registering gateway in ChirpStack Network server** section.

If everything is set correctly, the **Last seen** status will state a few seconds ago. You can click the gateway name to inspect the gateway traffic.

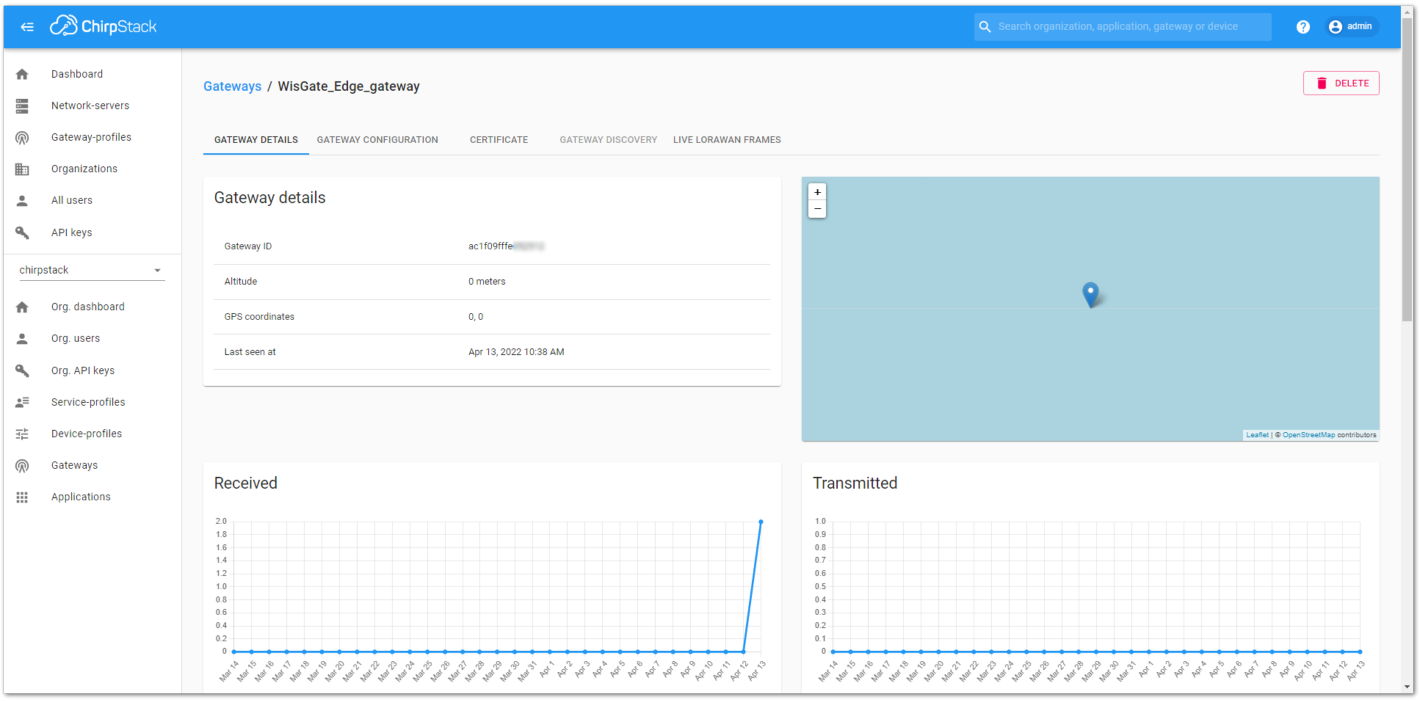


Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10 ▾ 1-1 of 1

Figure 59: Registered gateway



Gateways / WisGate_Edge_gateway

Gateway details

Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

Figure 60: Gateway details

Now your gateway is connected to the ChirpStack Network server.

External ChirpStack

In this case, the ChirpStack is installed on an external network. In the following example, the ChirpStack Network server is installed on the AWS cloud. A guide on how to install it on AWS can be found in the Knowledge Hub section.

To access the ChirpStack web UI, you need to enable **TCP port 8080** and to make the gateway to communicate with the Network server you need to enable the following ports in the inbound rules of the instance:

- The Semtech Packet Forwarder needs UDP port 1700.
- MQTT Bridge (unsecured) needs TCP port 1883.
- MQTT Bridge (secured) needs TCP port 8883.
- Basics Station needs TCP port 3001.

> 📝 NOTE
>
> A guide on how to open the above ports can be found in the guide on how to install ChirpStack on AWS.

Three options will be considered here:

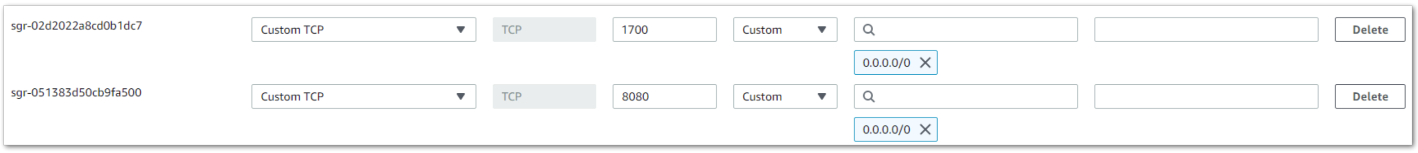
- **Connecting the Gateway via Packet Forwarder**
- **Connecting the Gateway via MQTT Bridge**
- **Connecting the Gateway via Basics Station**

Each option is explained in its own separate section.

Connecting the Gateway via Packet Forwarder

In this method, you will configure the gateway's packet forwarder to send data to the ChirpStack Gateway Bridge.

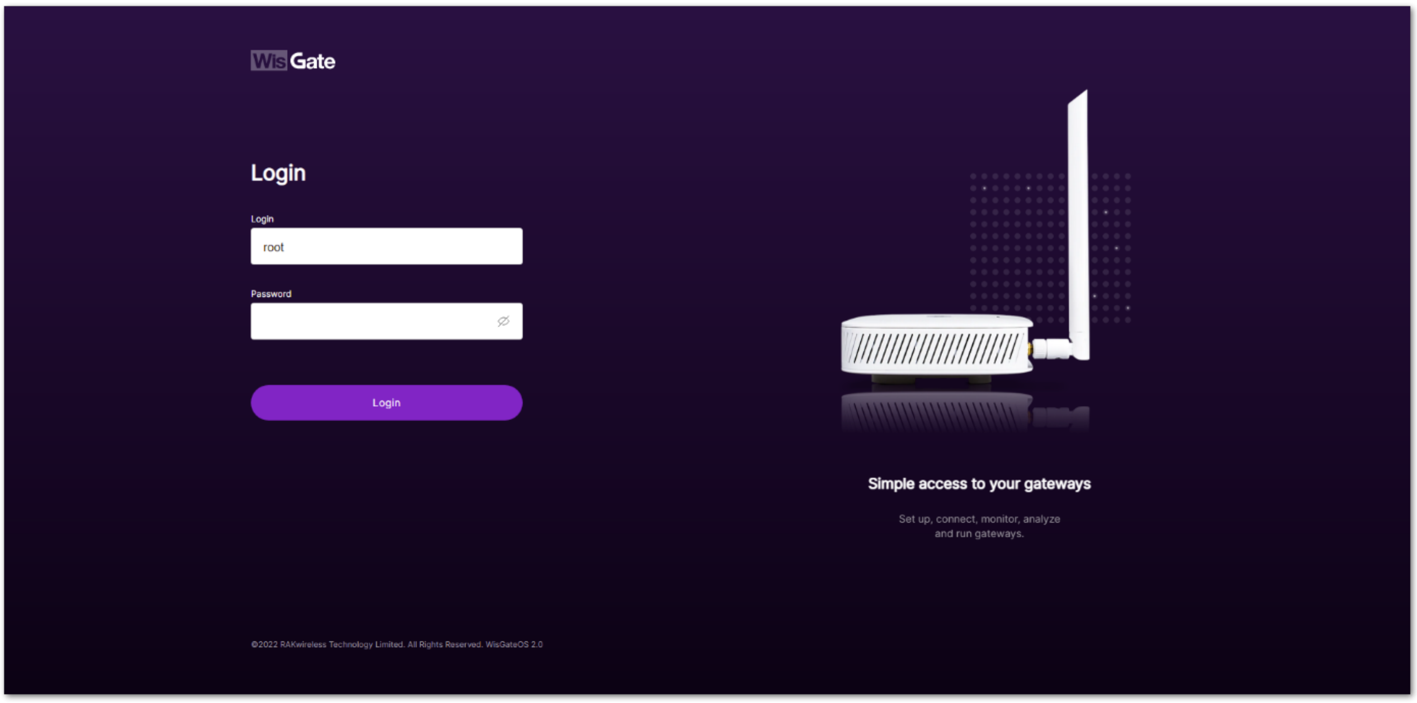
> 📝 NOTE
>
> When connecting the gateway to the ChirpStack, you will need to open ports 1700 and 8080 to enable the communication between the gateway and the server and be able to access the ChirpStack.



sgr-02d2022a8cd0b1dc7	Custom TCP ▼	TCP	1700	Custom ▼	🔍 0.0.0.0/0 ✕		Delete
sgr-051383d50cb9fa500	Custom TCP ▼	TCP	8080	Custom ▼	🔍 0.0.0.0/0 ✕		Delete

Figure 61: Opened 1700 UDP port

1. Start by accessing the gateway.



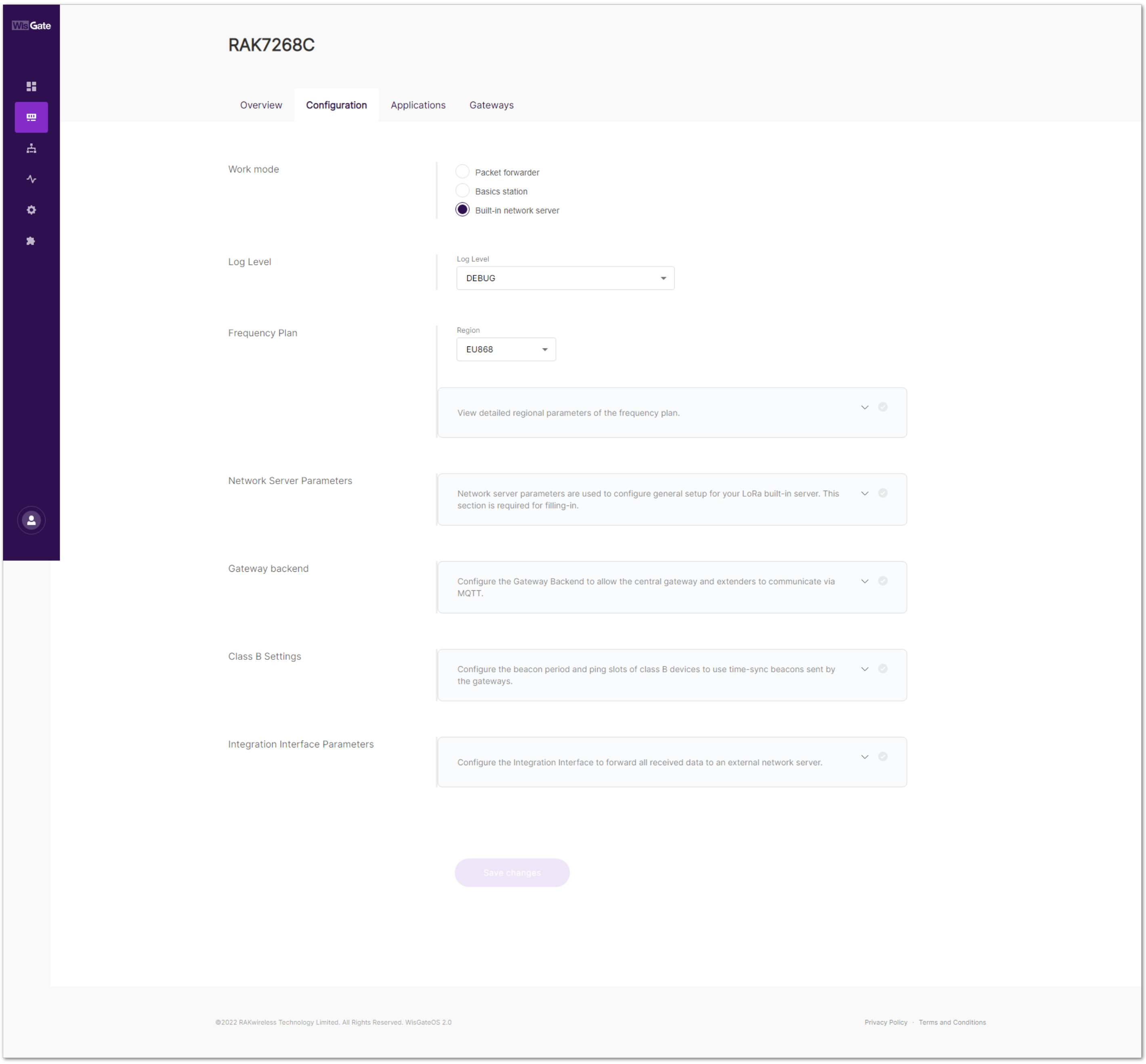
WisGate
Login
Simple access to your gateways
Set up, connect, monitor, analyze and run gateways.
Login
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Login: root

Password:

Figure 62: Login page

2. Login using the set credentials you have set in the **Access the gateway**.
3. On the left side, head to **LoRa**. By default, the gateway is configured to work as a **Built-in network server**.



WisGate
Overview
Configuration
Applications
Gateways
Save changes
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

RAK7268C

Work mode
- ○ Packet forwarder
- ○ Basics station
- ● Built-in network server

Log Level: DEBUG

Frequency Plan: Region EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters: Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

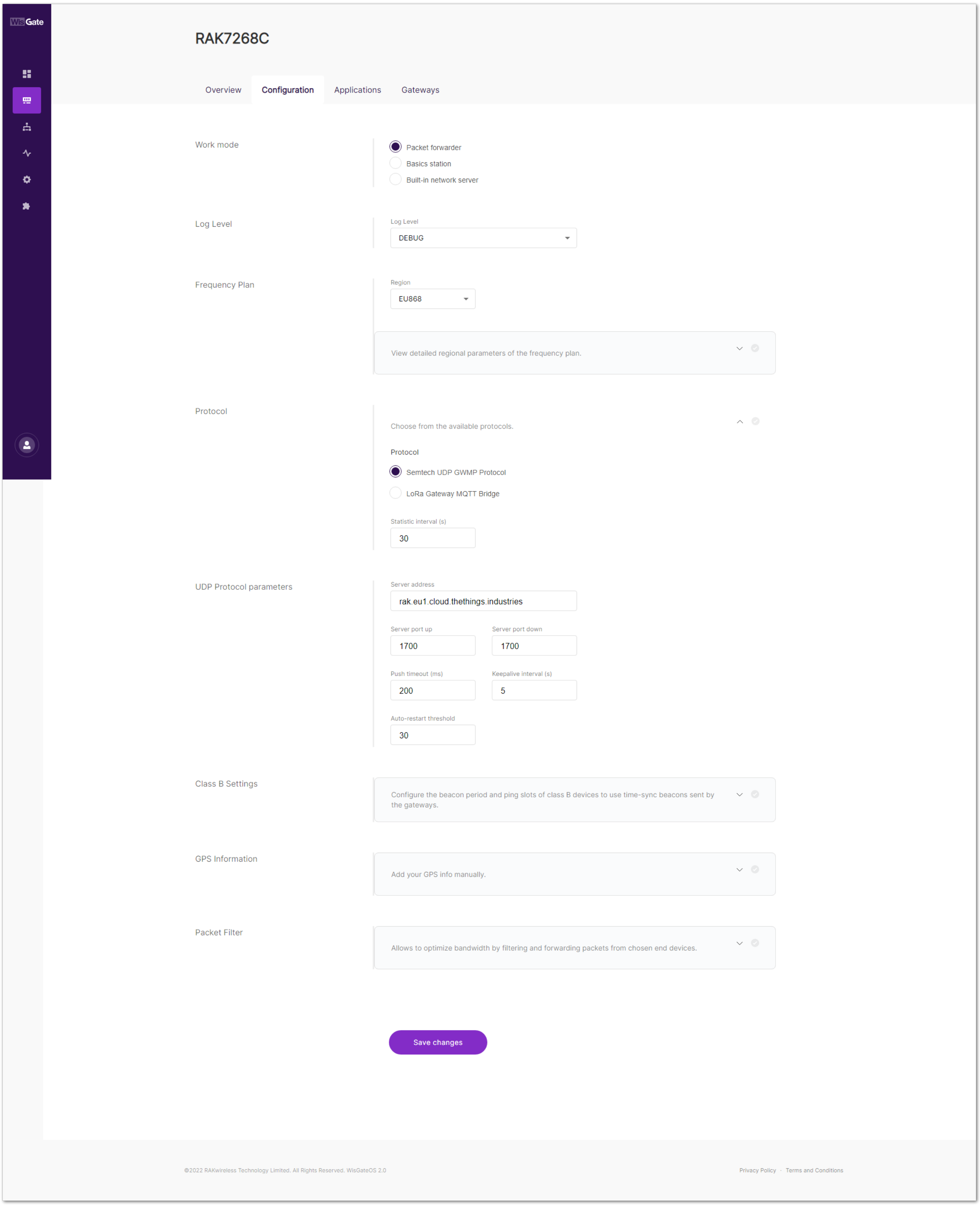
Gateway backend: Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

Class B Settings: Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

Integration Interface Parameters: Configure the Integration Interface to forward all received data to an external network server.

Figure 63: Network server settings

4. From **Work Mode**, select **Packet forwarder**. Click **Choose from the available protocols** to expand the Packet forwarder settings.



RAK7268C

Work mode
- ● Packet forwarder
- ○ Basics station
- ○ Built-in network server

Log Level
- Log Level: DEBUG

Frequency Plan
- Region: EU868
- View detailed regional parameters of the frequency plan.

Protocol
- Choose from the available protocols.
- Protocol
  - ● Semtech UDP GWMP Protocol
  - ○ LoRa Gateway MQTT Bridge
- Statistic interval (s): 30

UDP Protocol parameters
- Server address: rak.eu1.cloud.thethings.industries
- Server port up: 1700
- Server port down: 1700
- Push timeout (ms): 200
- Keepalive interval (s): 5
- Auto-restart threshold: 30

Class B Settings
- Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

GPS Information
- Add your GPS info manually.

Packet Filter
- Allows to optimize bandwidth by filtering and forwarding packets from chosen end devices.

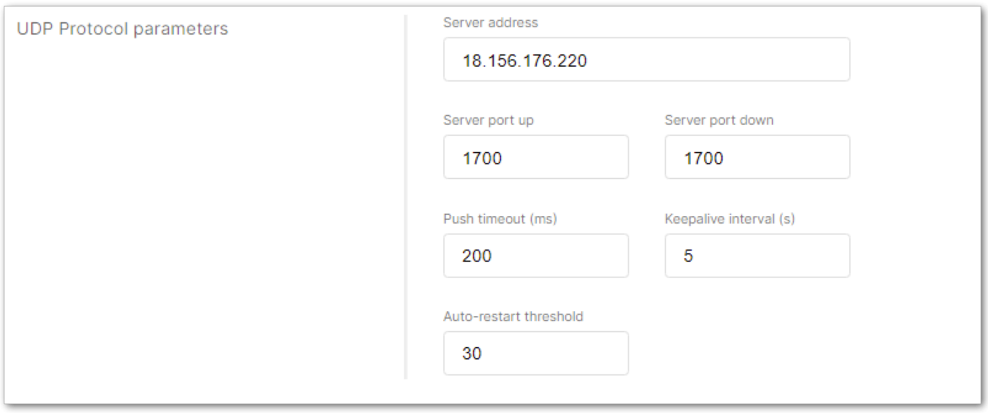
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 64: Setting packet forwarder settings

When Packet Forwarder mode is chosen, the **Semtech UDP GWMP Protocol** is selected by default.

To point the gateway to the ChirpStack network using the packet forwarder, you only need to set the When Packet Forwarder mode. The **Semtech UDP GWMP Protocol** is selected by default of the ChirpStack.

In this case, the ChirpStack is installed on the AWS cloud instance with public IP **18.156.176.220** (yours will be different). The default ports that the packet forwarder is using are 1700.



UDP Protocol parameters

Server address: 18.156.176.220

Server port up: 1700

Server port down: 1700

Push timeout (ms): 200

Keepalive interval (s): 5

Auto-restart threshold: 30

Figure 65: Configure packet forwarder to ChirpStack

5. Click **Save changes** to save the changes.

Now you need to register the gateway in the ChirpStack Network server.

Registering the Gateway in ChirpStack

The steps for registering the gateway in ChirpStack are the same for all options.

1. To register the gateway in the ChirpStack Network server, access the ChirpStack UI. To do that, open a web browser and type the server address of the ChirpStack with port 8080.

```
<IP address of ChirpStack>:8080
```

2. In this case, the ChirpStack is installed on the AWS cloud with the public IP address `18.156.176.220.`

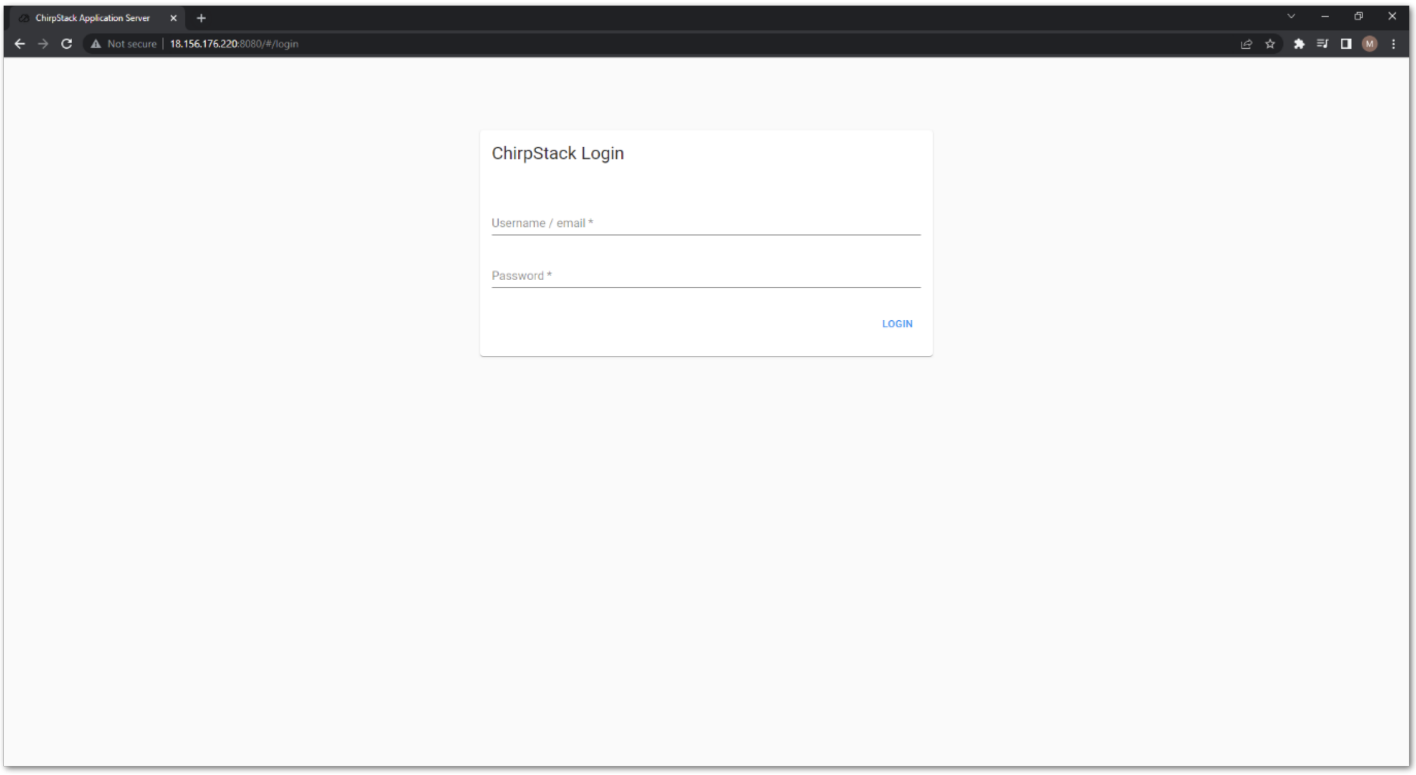

ChirpStack Application Server
Not secure | 18.156.176.220:8080/#/login
ChirpStack Login
Username / email *
Password *
LOGIN

Figure 66: ChirpStack login page

3. Login using the following credentials:

- Username/email: **admin**
- Password: **admin**

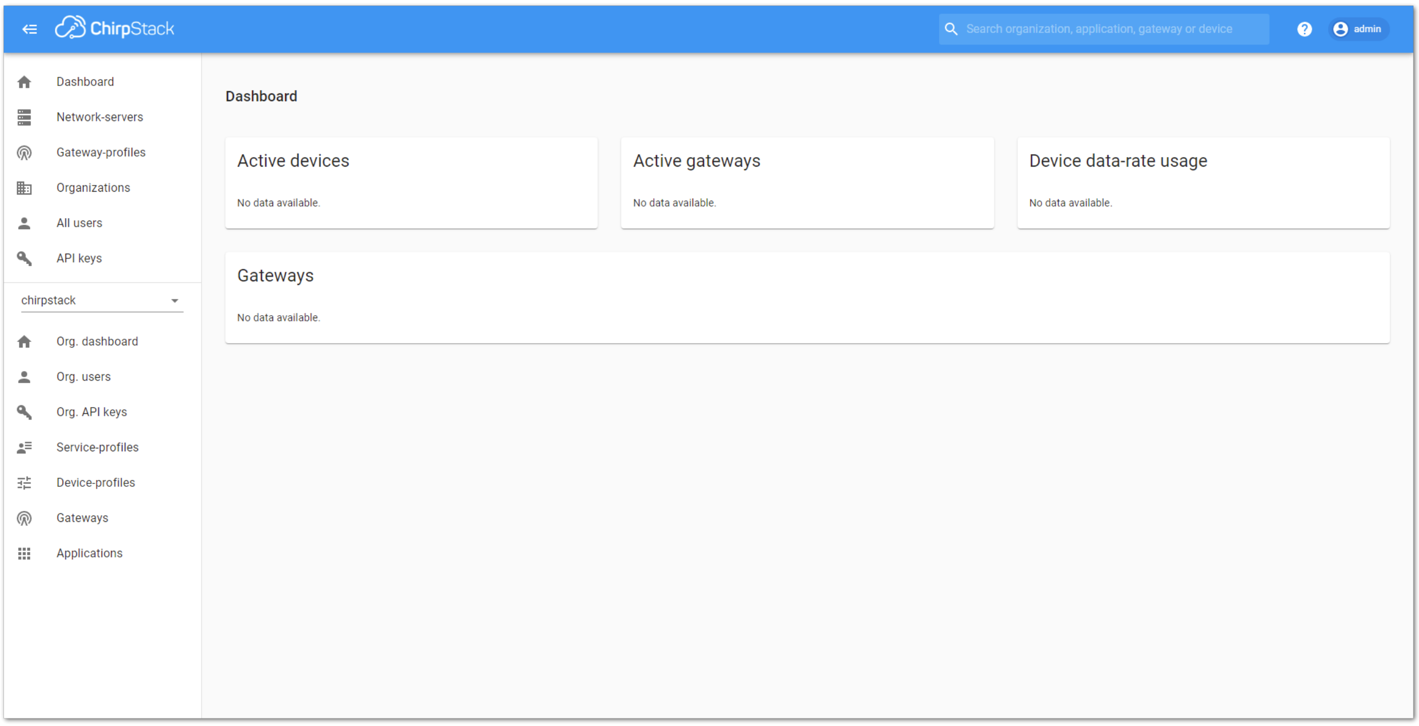


Figure 67: ChirpStack dashboard

4. On the left pane, head to **Gateways**.

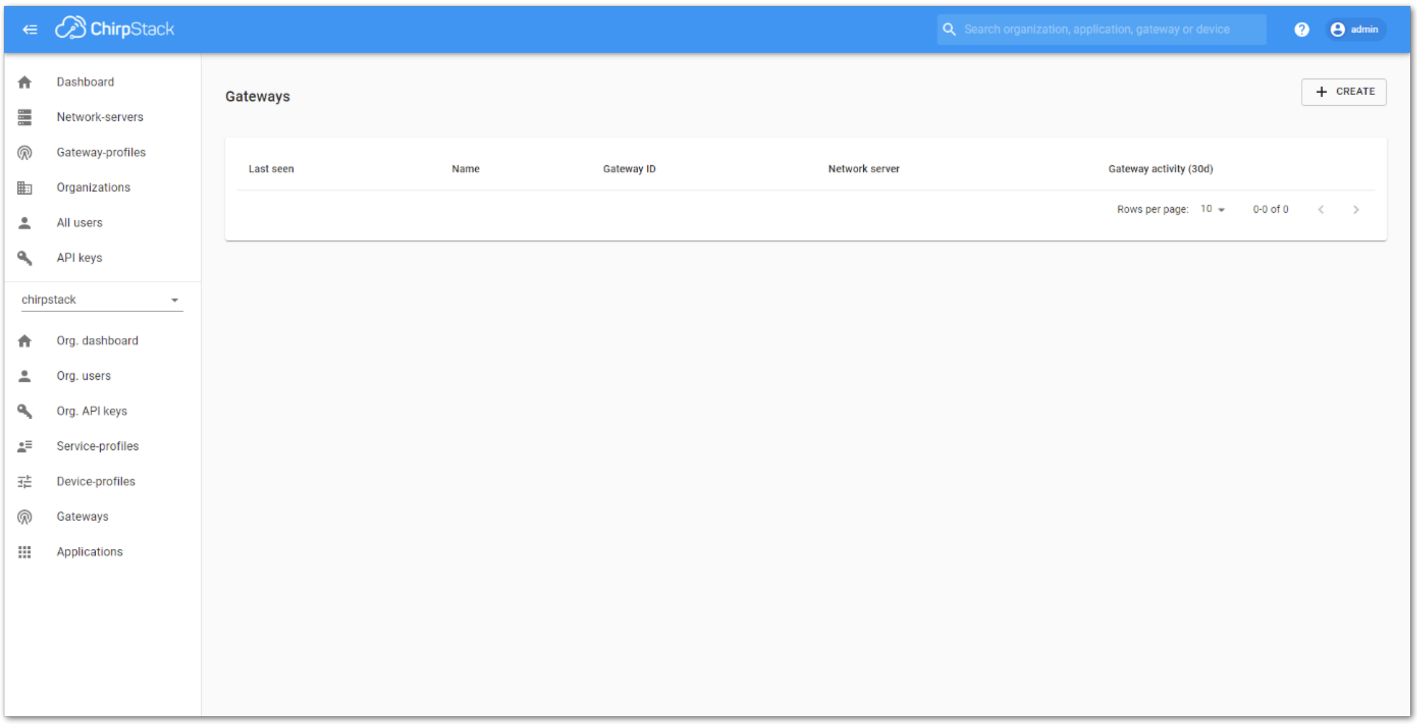
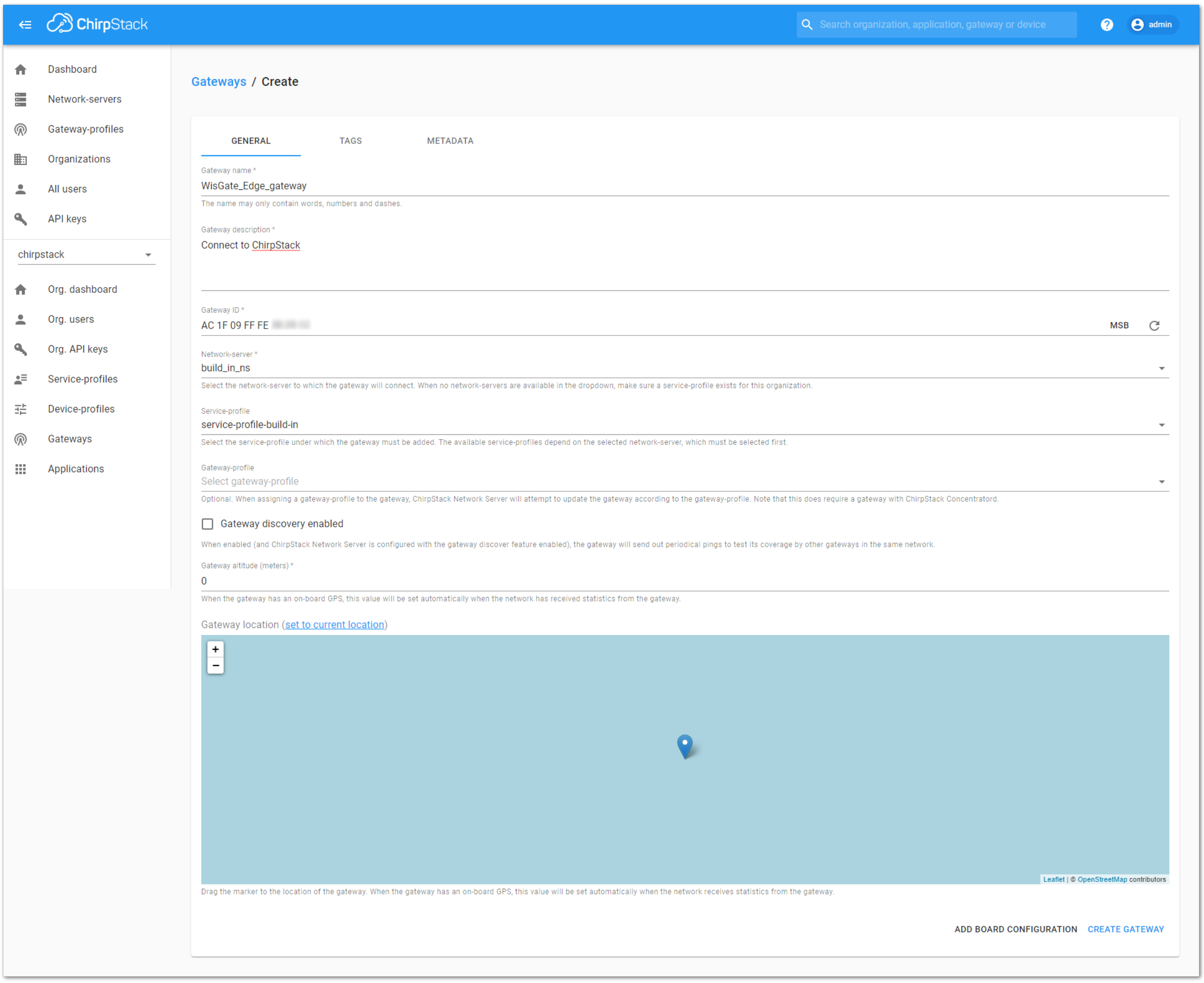


Figure 68: Gateway list

5. By default, no gateways are registered. To register one, click **+ Create.**

6. In the **General** menu, you need to set the gateway parameters.



Gateway name *
WisGate_Edge_gateway
The name may only contain words, numbers and dashes.

Gateway description *
Connect to ChirpStack

Gateway ID *
AC 1F 09 FF FE [redacted] MSB

Network-server *
build_in_ns
Select the network-server to which the gateway will connect. When no network-servers are available in the dropdown, make sure a service-profile exists for this organization.

Service-profile
service-profile-build-in
Select the service-profile under which the gateway must be added. The available service-profiles depend on the selected network-server, which must be selected first.

Gateway-profile
Select gateway-profile
Optional. When assigning a gateway-profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway-profile. Note that this does require a gateway with ChirpStack Concentratord.

☐ Gateway discovery enabled
When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical pings to test its coverage by other gateways in the same network.

Gateway altitude (meters) *
0
When the gateway has an on-board GPS, this value will be set automatically when the network has received statistics from the gateway.

Gateway location (set to current location)

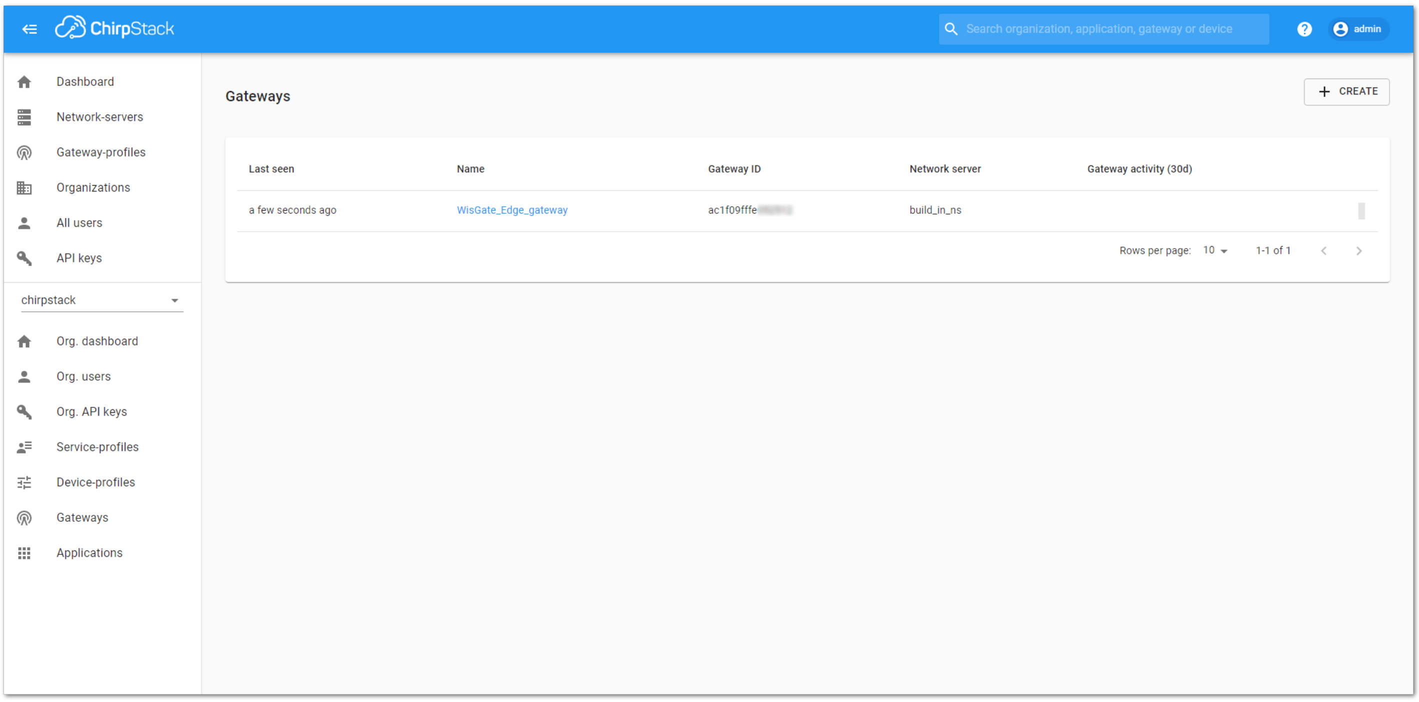
Drag the marker to the location of the gateway. When the gateway has an on-board GPS, this value will be set automatically when the network receives statistics from the gateway.

Figure 69: Register the gateway

- **Gateway name** – unique name for the gateway on the Network server. The name may only contain words, numbers, and dashes.
- **Gateway description** – a brief description of the gateway.
- **Gateway ID** – the Extended Unique Identifier (EUI) of the gateway. The EUI is in the Overview menu of the Dashboard page of the web UI of the gateway.
- **Network server** - the network server to which the gateway will connect. When no network servers are available in the dropdown, make sure a service profile exists for this organization.
- **Service-profile** - the service profile under which the gateway must be added. The available service profiles depend on the selected network server, which must be selected first.
- **Gateway profile** – this field is optional. When assigning a gateway profile to the gateway, ChirpStack Network Server will attempt to update the gateway according to the gateway profile. Note that this does require a gateway with ChirpStack Concentrator.
- **Gateway discovery enabled** - When enabled (and ChirpStack Network Server is configured with the gateway discover feature enabled), the gateway will send out periodical pings to test its coverage by other gateways in the same network.
- **Gateway attitude** - When the gateway has an onboard, this value will be set automatically when the network has received statistics from the gateway.
- **Gateway location** – you can drag the marker to the location of the gateway. When the gateway has an onboard GPS, this value will be set automatically when the network receives statistics from the gateway.

7. Once everything is set, click **Create gateway** to register the gateway. You will see the registered gateway in the Gateway list.

If everything is set correctly, the Last seen status in the ChirpStack will state **a few seconds ago.**



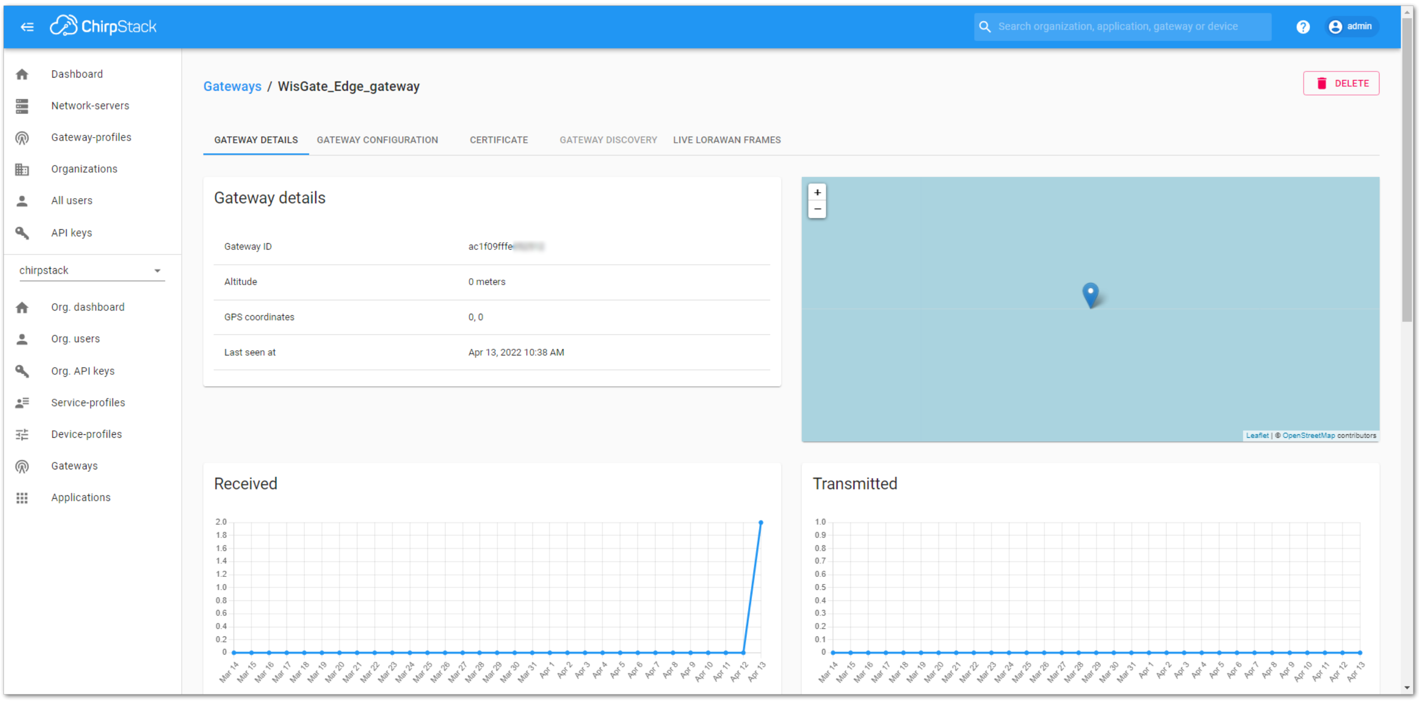
Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10 1-1 of 1

Figure 70: Registered gateway

8. You can click the gateway name to inspect the gateway traffic.



Gateways / WisGate_Edge_gateway

Gateway details

Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

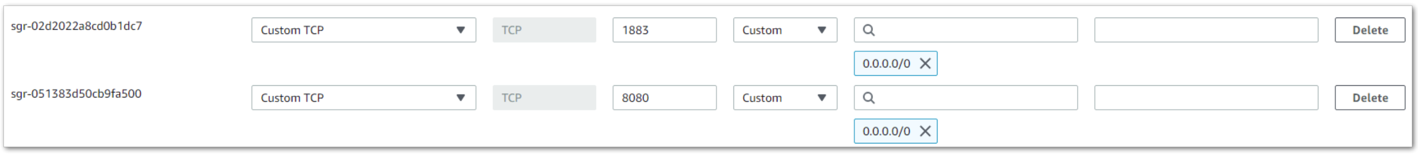
Figure 71: Gateway details

Now your gateway is connected to the ChirpStack Network server.

Connecting the Gateway via MQTT Bridge

In this method, you will configure the gateway's built-in bridge to publish the data to the ChirpStack MQTT broker.

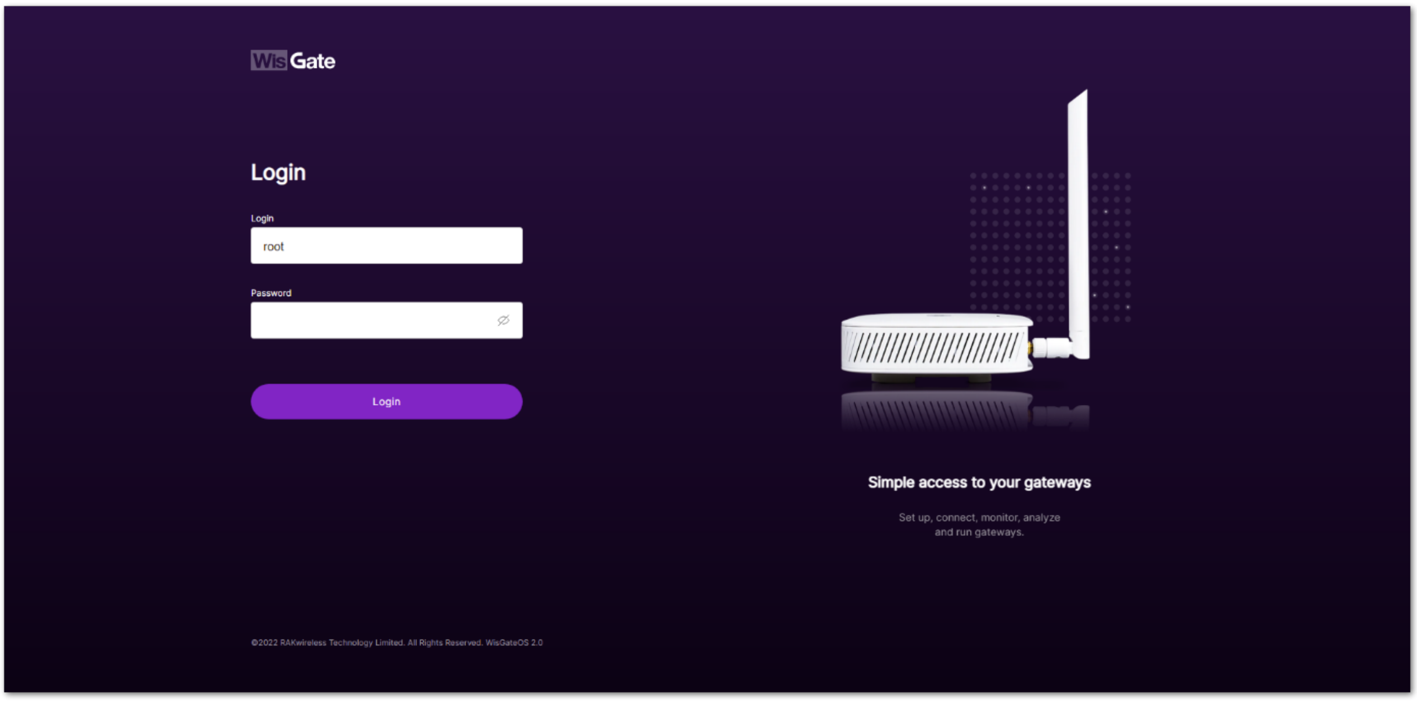
> 📝 **NOTE**
>
> When connecting the gateway to the ChirpStack, you will need to open ports 1883 and 8080 to enable the communication between the gateway and the server and be able to access the ChirpStack.



sgr-02d2022a8cd0b1dc7	Custom TCP	TCP	1883	Custom	0.0.0.0/0		Delete
sgr-051383d50cb9fa500	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete

Figure 72: Login page

1. Start by accessing the gateway.



Login: root

Password:

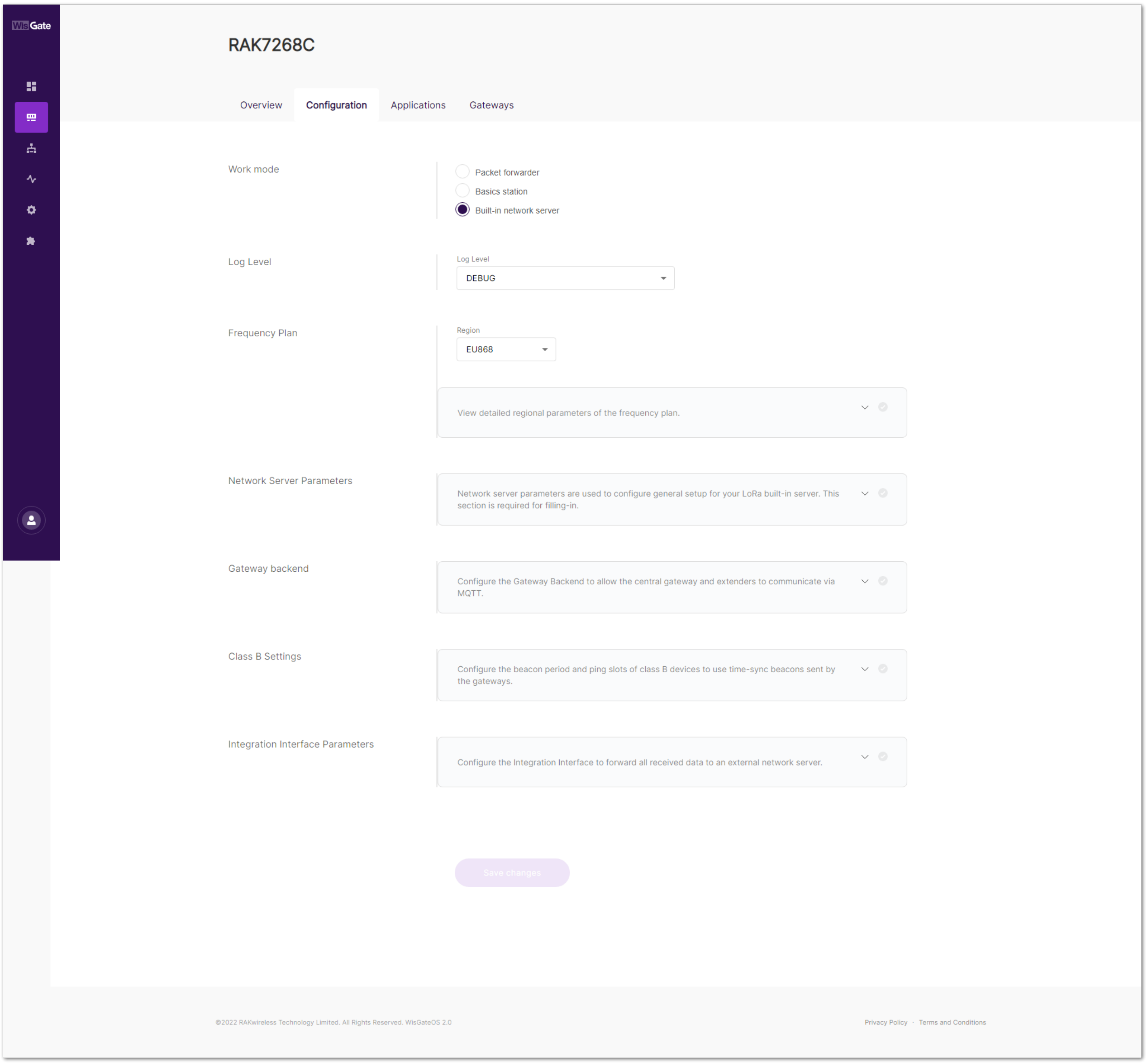
Simple access to your gateways
Set up, connect, monitor, analyze and run gateways.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 73: Login page

2. Login using the set credentials you have set in the **Access the gateway**.

3. On the left side, head to **LoRa**. By default, the gateway is configured to work as **Built-in network server**.



Work mode
○ Packet forwarder
○ Basics station
● Built-in network server

Log Level
Log Level: DEBUG

Frequency Plan
Region: EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters
Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

Gateway backend
Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

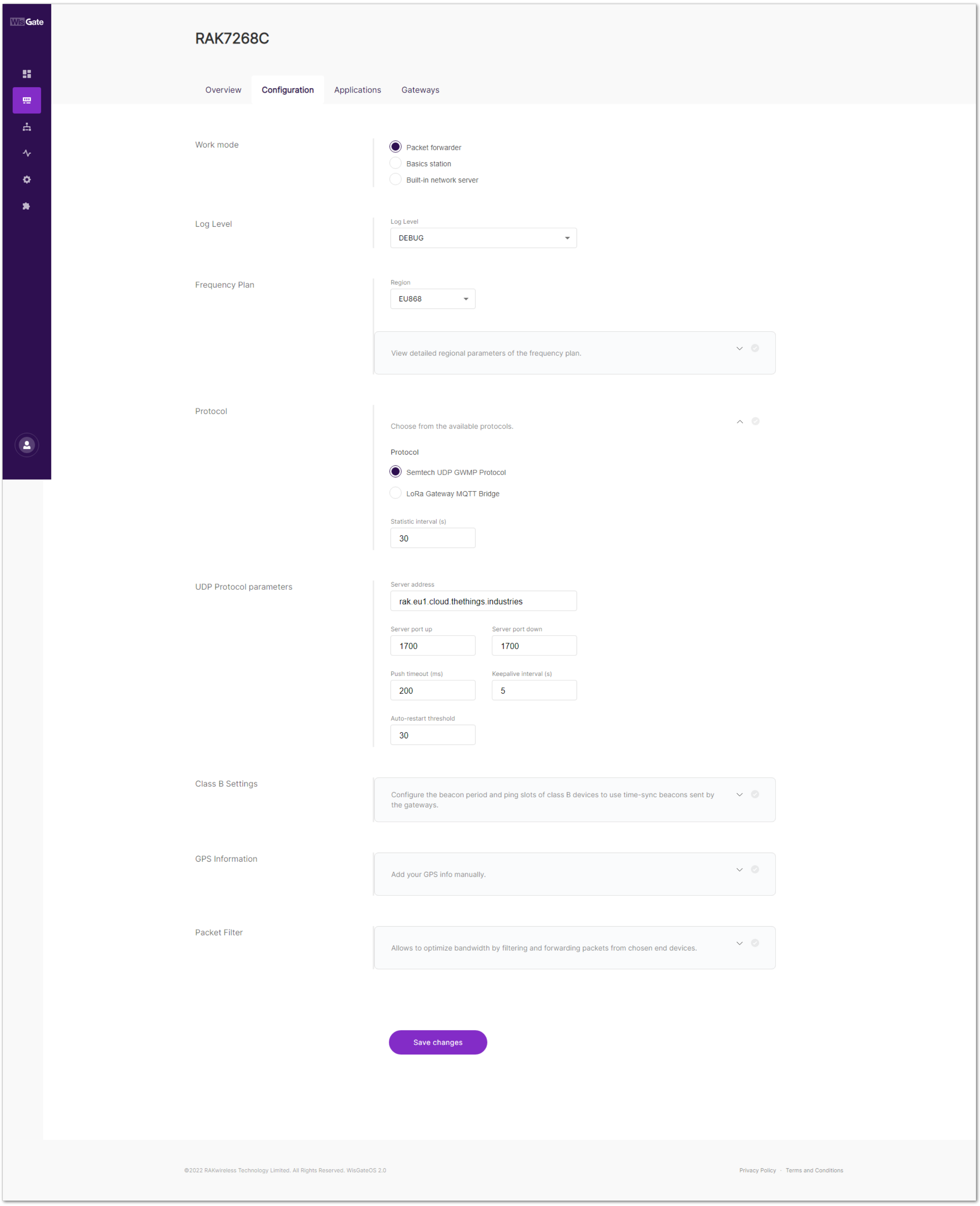
Class B Settings
Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

Integration Interface Parameters
Configure the Integration Interface to forward all received data to an external network server.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 74: Network server settings

4. From **Work Mode**, select **Packet forwarder**. Click **Choose from the available protocols** to expand the Packet forwarder settings.



RAK7268C

Work mode
● Packet forwarder
○ Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Frequency Plan
Region: EU868

View detailed regional parameters of the frequency plan.

Protocol
Choose from the available protocols.
Protocol
● Semtech UDP GWMP Protocol
○ LoRa Gateway MQTT Bridge
Statistic interval (s): 30

UDP Protocol parameters
Server address: rak.eu1.cloud.thethings.industries
Server port up: 1700
Server port down: 1700
Push timeout (ms): 200
Keepalive interval (s): 5
Auto-restart threshold: 30

Class B Settings
Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

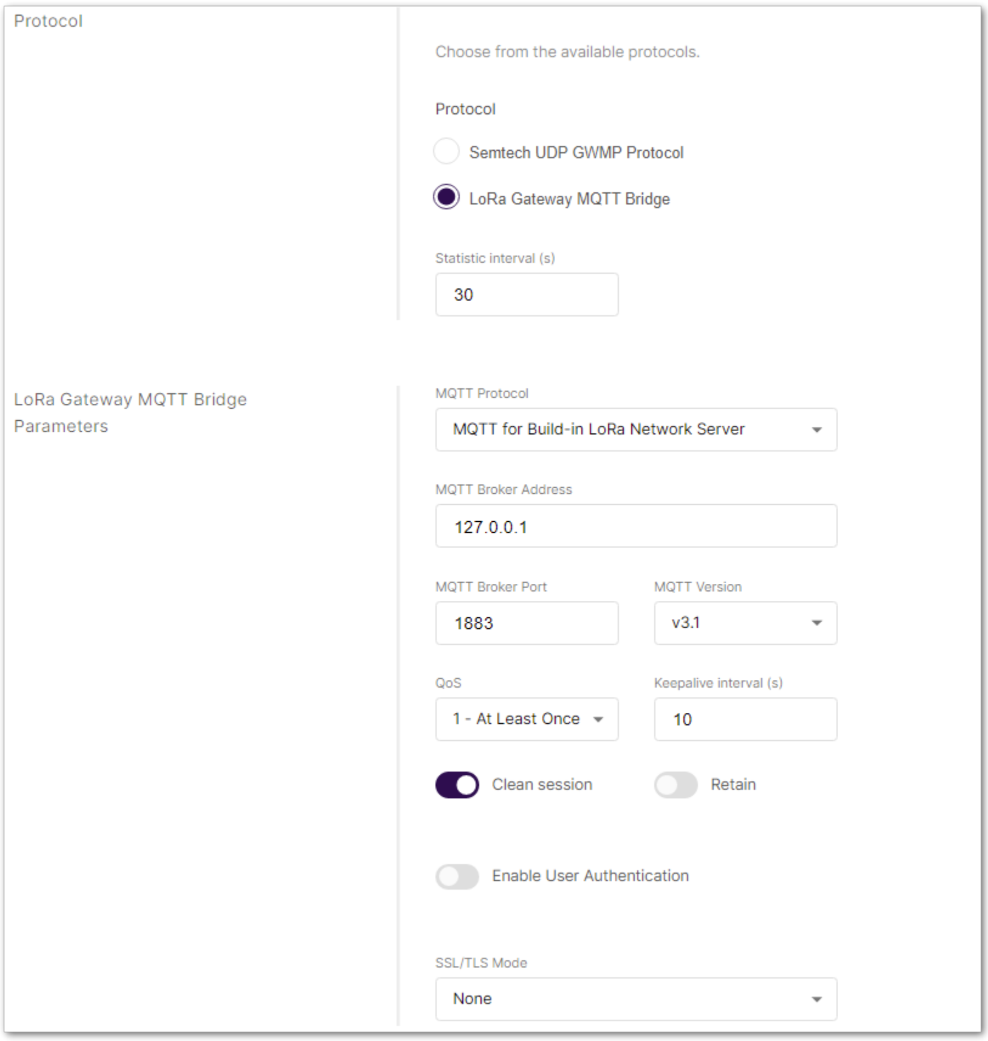
GPS Information
Add your GPS info manually.

Packet Filter
Allows to optimize bandwidth by filtering and forwarding packets from chosen end devices.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 75: Setting packet forwarder mode

5. By default, when Packet Forwarder mode is chosen, the **Semtech UDP GWMP Protocol** is selected. To use the built-in gateway bridge, from the **Protocol** select **LoRa Gateway MQTT Bridge**.



Protocol

Choose from the available protocols.

Protocol

○ Semtech UDP GWMP Protocol

● LoRa Gateway MQTT Bridge

Statistic interval (s): 30

LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for Build-in LoRa Network Server

MQTT Broker Address: 127.0.0.1

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session: on

Retain: off

Enable User Authentication: off

SSL/TLS Mode: None

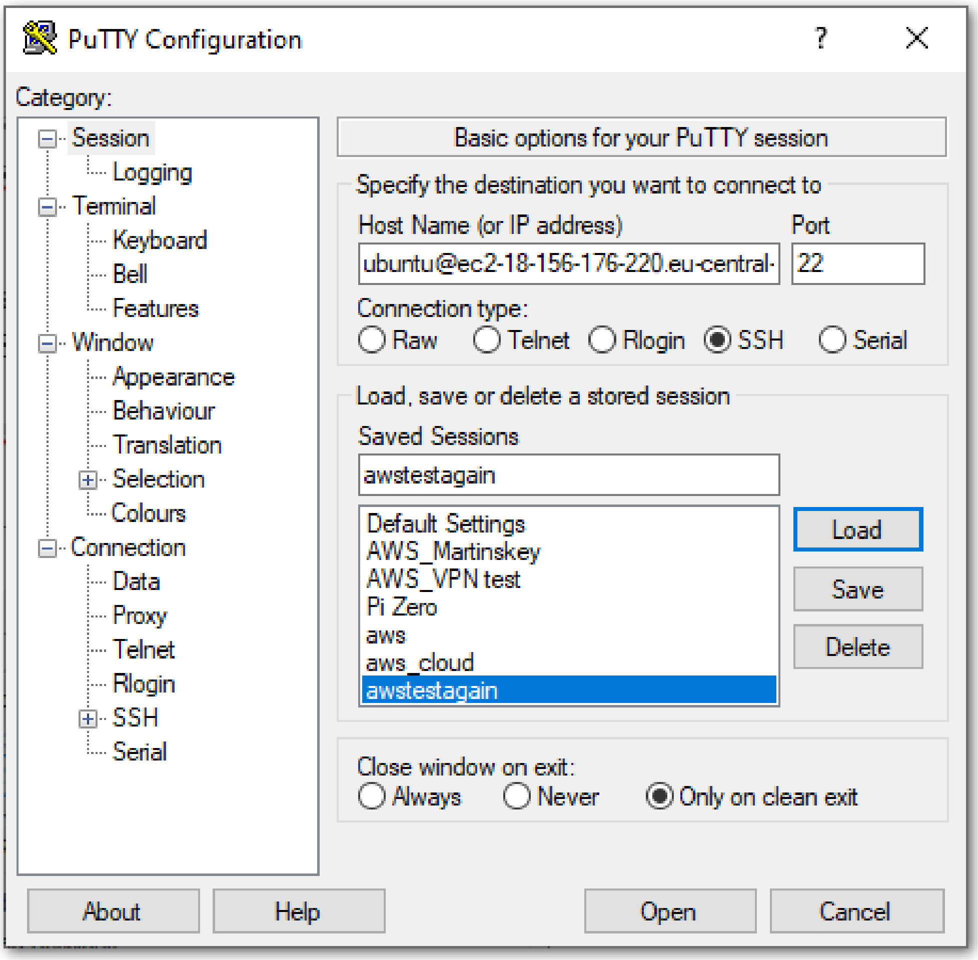
Figure 76: LoRa Gateway MQTT bridge

The latest ChirpStack version supports both **MQTT for ChirpStack 3.x (JSON)** and **MQTT for ChirpStack 3.x (PROTOBUF).**

📝 **NOTE**

If you want to use JSON protocol, you need to change the payload marshaler in the gateway bridge `.toml` file to json. By default, the marshaler is protobuf.

6. To configure the payload marshaler, you need to access the configuration file of the gateway bridge. You can access it with an SSH connection. In this case, you will use the SSH client PuTTY to access the configuration files. How to do this is explained in the Knowledge Hub section.



Host Name (or IP address): ubuntu@ec2-18-156-176-220.eu-central-
Port: 22

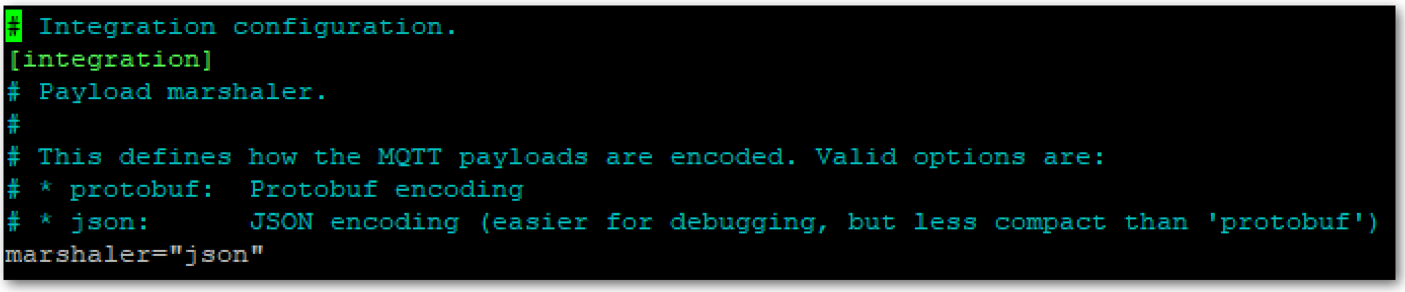
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Saved Sessions: awstestagain

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 77: PuTTY client

7. In the `/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml` find the Integration section and change the marshaler to `json` .

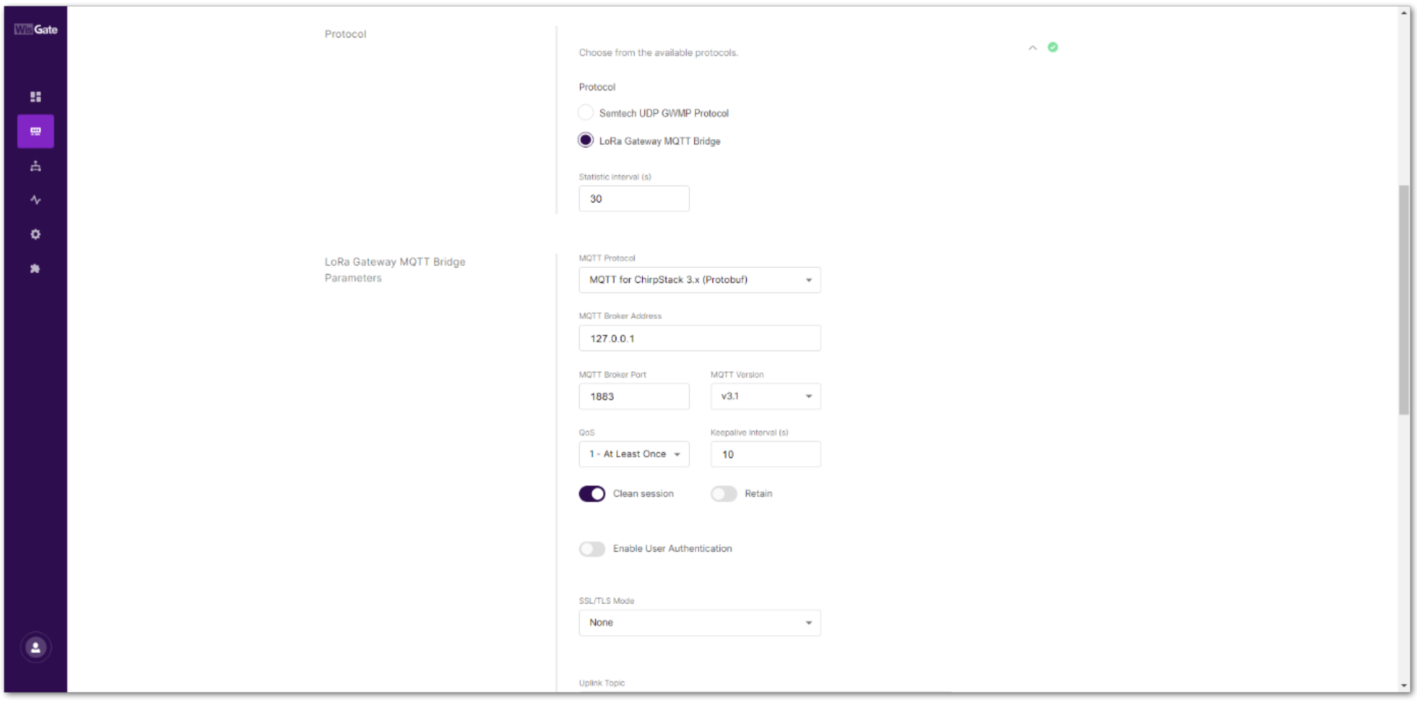


```
# Integration configuration.
[integration]
# Payload marshaler.
#
# This defines how the MQTT payloads are encoded. Valid options are:
# * protobuf:  Protobuf encoding
# * json:      JSON encoding (easier for debugging, but less compact than 'protobuf')
marshaler="json"
```

Figure 78: Payload Masher

8. Save and exit the file.

However, if you are using an earlier version of ChirpStack (V2), you will need to select **MQTT for ChirpStack 2.x.** The option **MQTT for Embedded RAK Network Server** is for a mesh network, where one gateway plays the role of a network server. For this example, you will choose **MQTT for ChirpStack 3.x (PROTOBUF)**.



Protocol

Choose from the available protocols.

Protocol

○ Semtech UDP GWMP Protocol

● LoRa Gateway MQTT Bridge

Statistic interval (s): 30

LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for ChirpStack 3.x (Protobuf)

MQTT Broker Address: 127.0.0.1

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session: on

Retain: off

Enable User Authentication: off

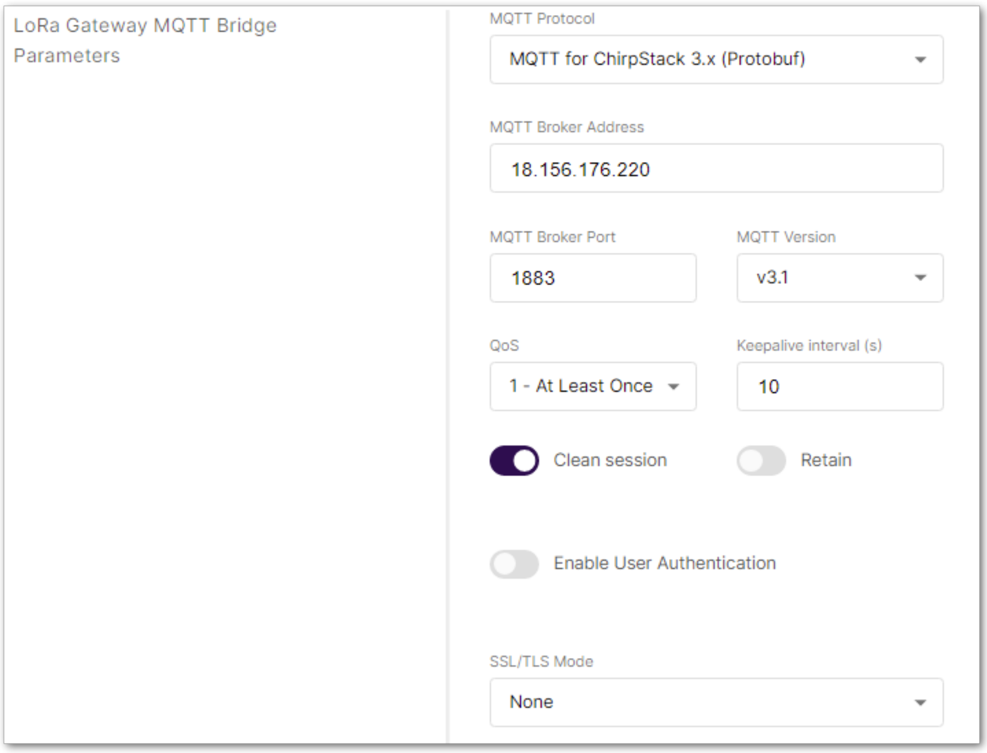
SSL/TLS Mode: None

Uplink Topic

Figure 79: MQTT for ChirpStack protocol

By default, the built-in gateway bridge is pointed to the local Broker (127.0.0.1). To point the gateway to the ChirpStack network, you need to set the ChirpStack Broker address in the **MQTT Broker Address** field.

In this case, the ChirpStack is installed on an AWS cloud instance with public IP `18.156.176.220` (yours will be different). The default port that the MQTT Broker uses is 1883.



LoRa Gateway MQTT Bridge Parameters

MQTT Protocol: MQTT for ChirpStack 3.x (Protobuf)

MQTT Broker Address: 18.156.176.220

MQTT Broker Port: 1883

MQTT Version: v3.1

QoS: 1 - At Least Once

Keepalive interval (s): 10

Clean session: on

Retain: off

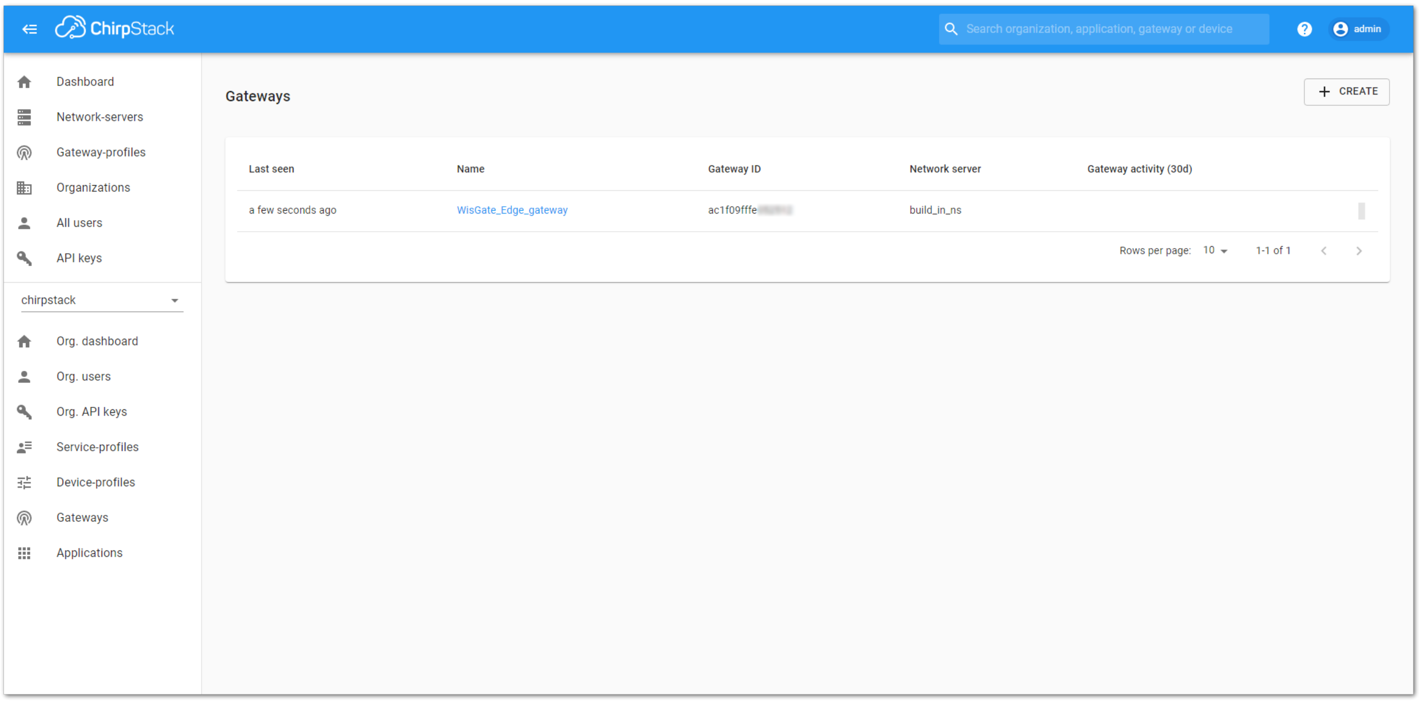
Enable User Authentication: off

SSL/TLS Mode: None

Figure 80: Configuring packet forwarder to ChirpStack

9. Click **Save changes** to save the changes.

If everything is set correctly, the **Last seen status** will state a few seconds ago. You can click the gateway name to inspect the gateway traffic.

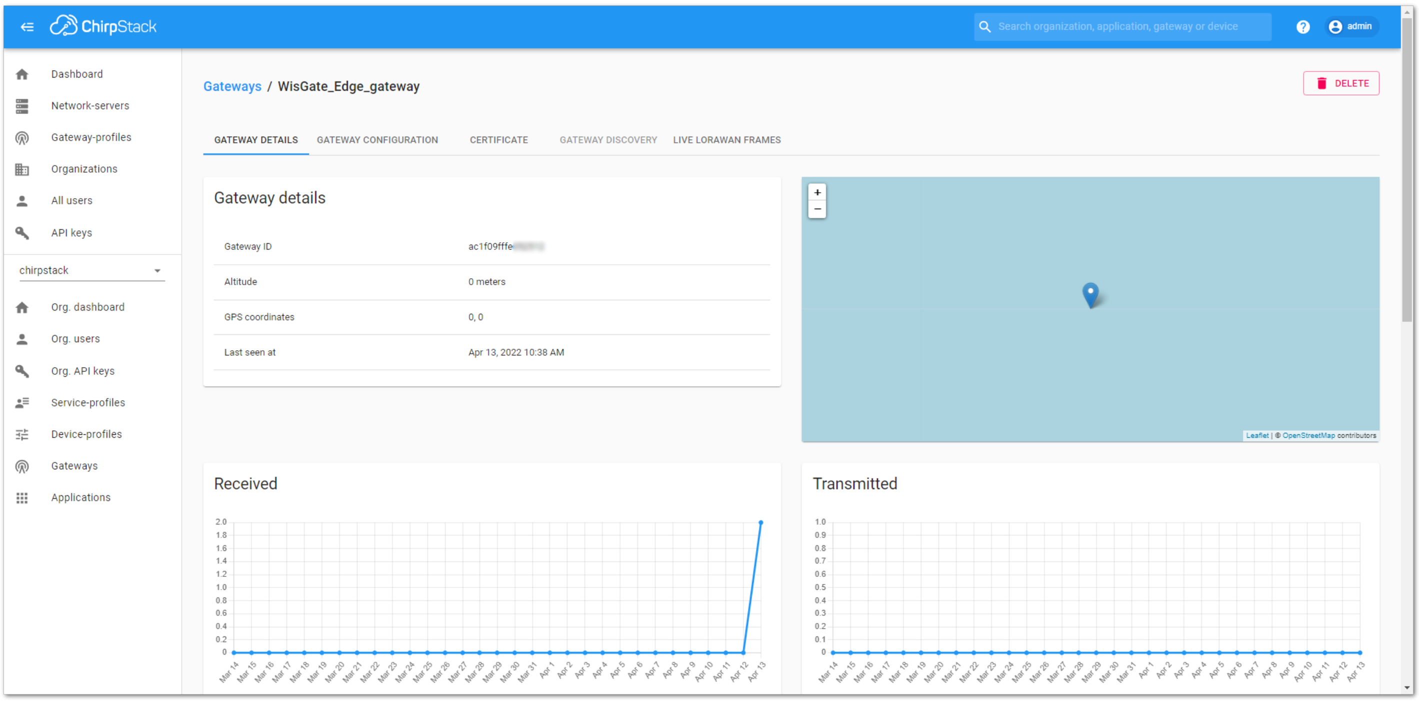


Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10 1-1 of 1

Figure 81: Registered gateway



Gateways / WisGate_Edge_gateway

Gateway details

Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

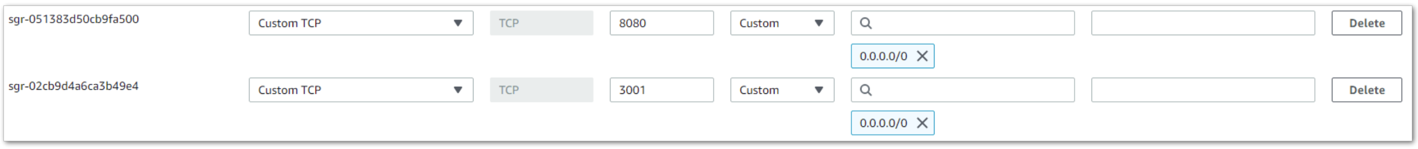
Figure 82: Gateway details

Now your gateway is connected to the ChirpStack Network server.

Connecting the Gateway via Basics Station

In this method, you will connect the gateway to the ChirpStack via Basics Station. The LoRa Basics™ Station is an implementation of a LoRa packet forwarder.

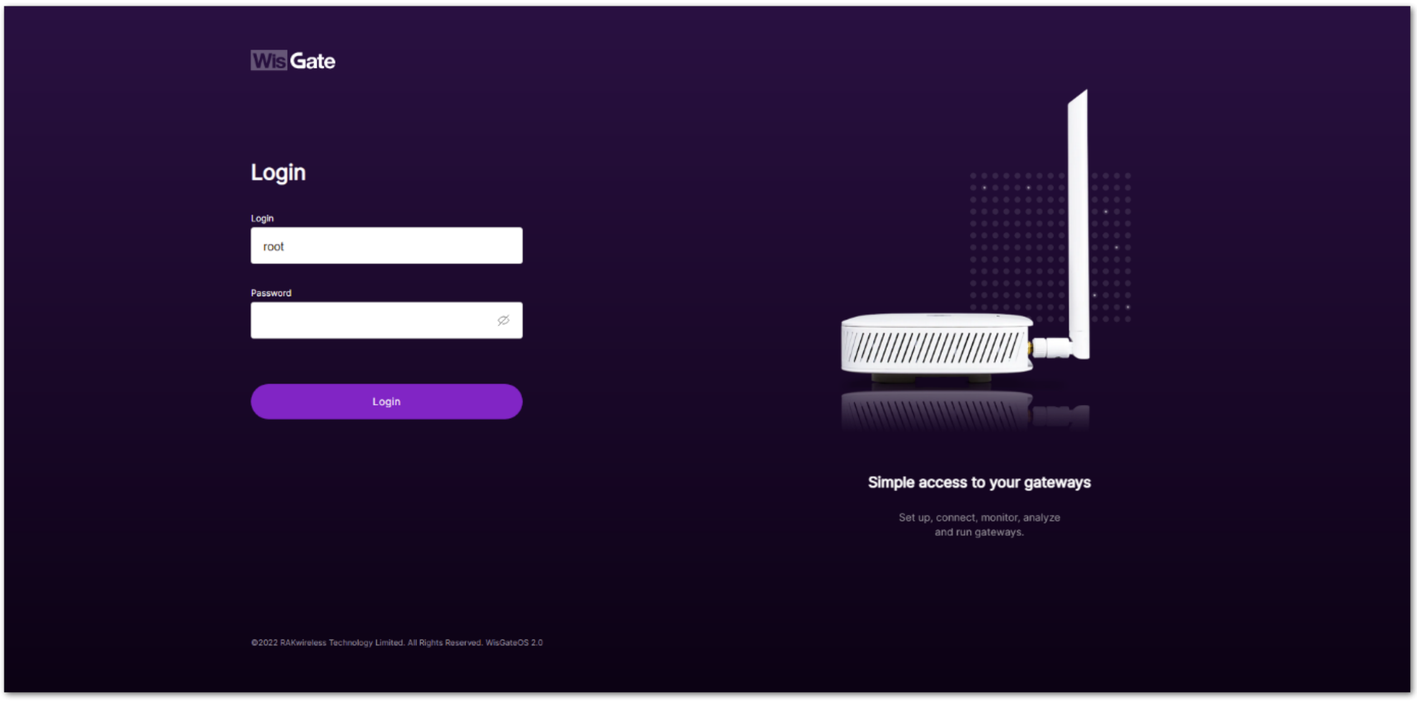
> 📝 **NOTE**
>
> When connecting the gateway to the ChirpStack, you will need to open TCP ports3001 and 8080 to enable the communication between the gateway and the server and be able to access the ChirpStack.



sgr-051383d50cb9fa500	Custom TCP	TCP	8080	Custom	0.0.0.0/0	
sgr-02cb9d4a6ca3b49e4	Custom TCP	TCP	3001	Custom	0.0.0.0/0	

Figure 83: Enable 8080 and 3001 port

1. Start by accessing the gateway.



Login: root

Password:

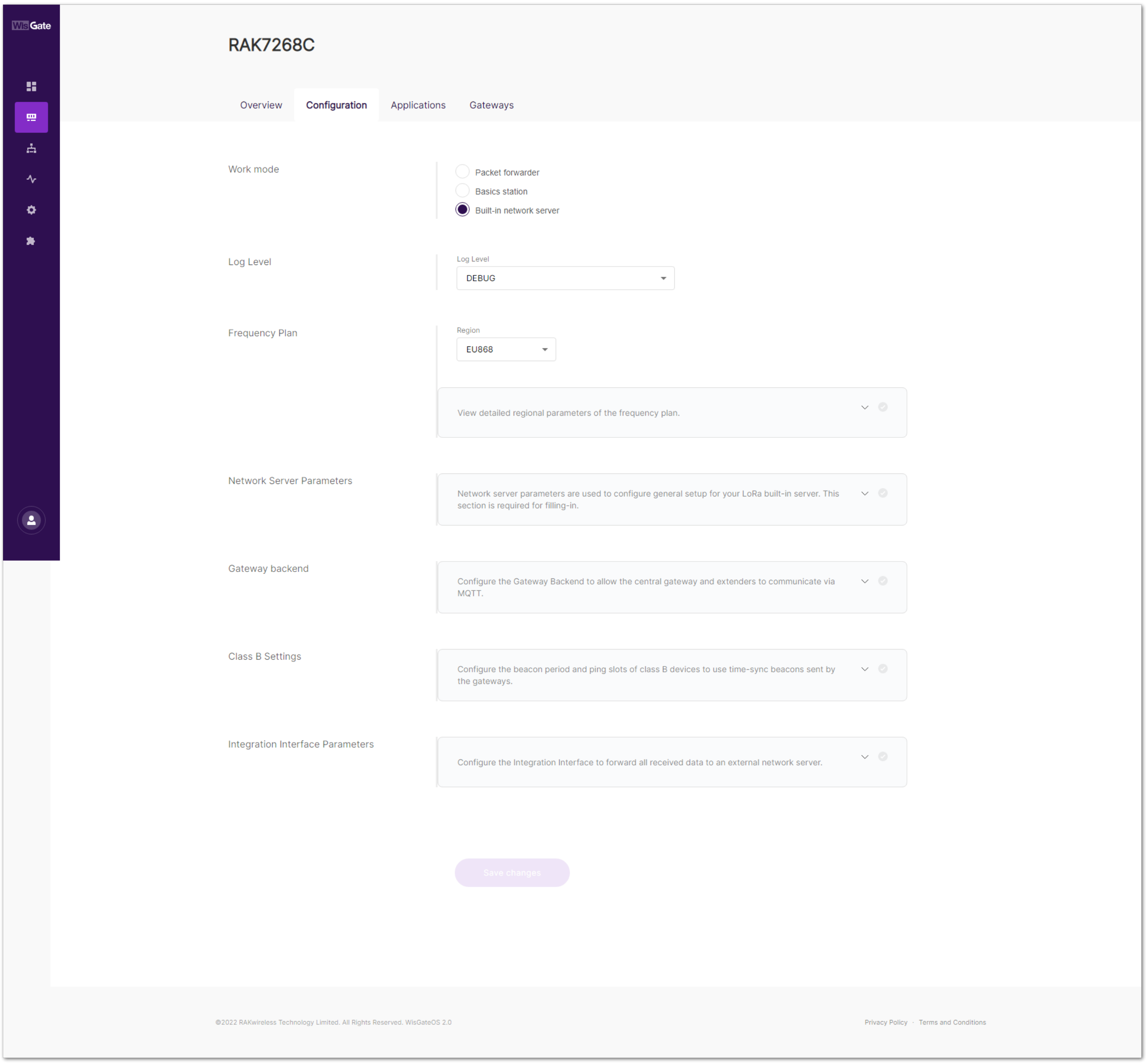
Simple access to your gateways

Set up, connect, monitor, analyze and run gateways.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Figure 84: Login page

2. Login using the set credentials you have set in the **Access the gateway**.
3. On the left side, head to **LoRa**. By default, the gateway is configured to work as a **Built-in network server**.



RAK7268C

Overview | Configuration | Applications | Gateways

Work mode

○ Packet forwarder
○ Basics station
● Built-in network server

Log Level

Log Level: DEBUG

Frequency Plan

Region: EU868

View detailed regional parameters of the frequency plan.

Network Server Parameters

Network server parameters are used to configure general setup for your LoRa built-in server. This section is required for filling-in.

Gateway backend

Configure the Gateway Backend to allow the central gateway and extenders to communicate via MQTT.

Class B Settings

Configure the beacon period and ping slots of class B devices to use time-sync beacons sent by the gateways.

Integration Interface Parameters

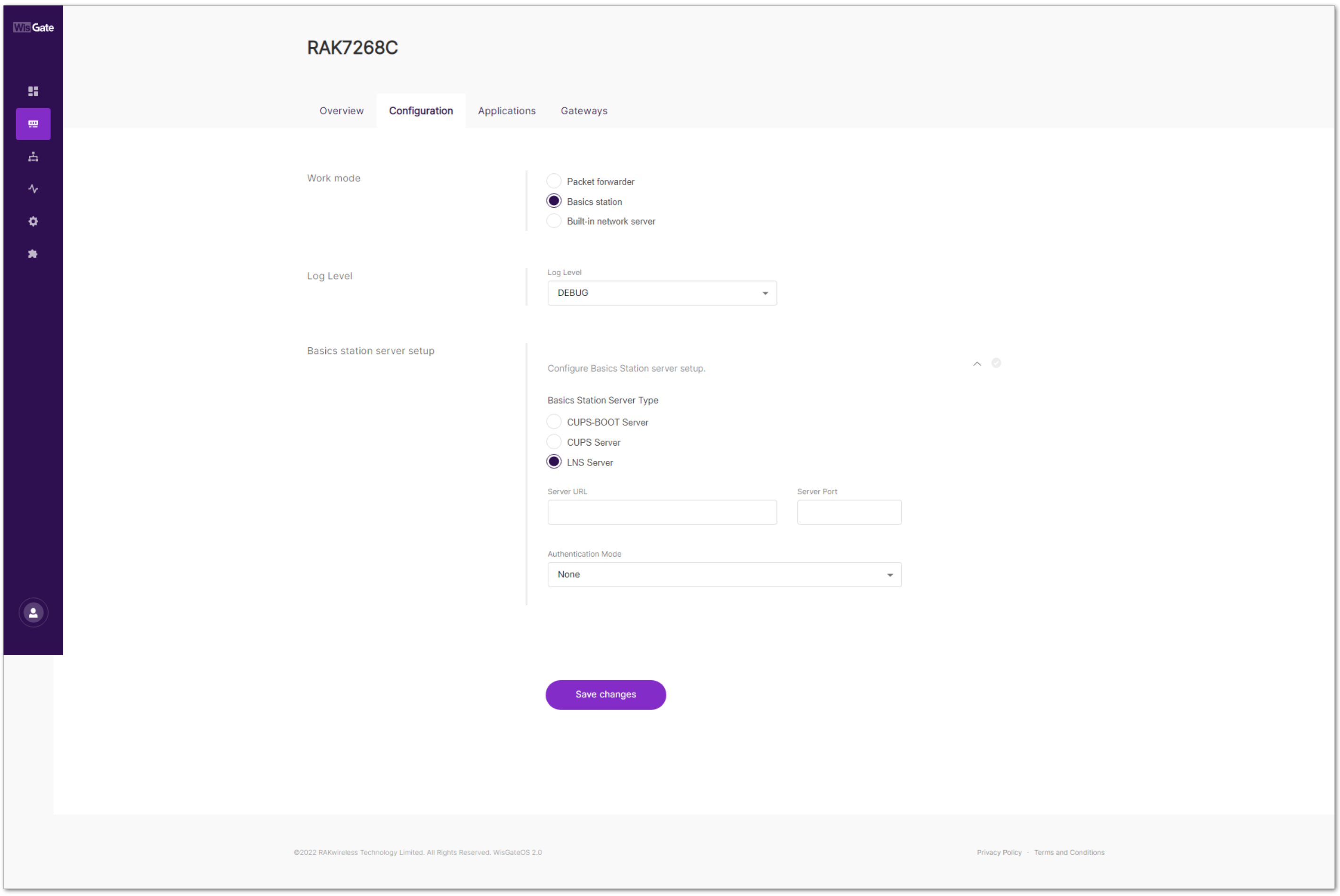
Configure the Integration Interface to forward all received data to an external network server.

©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0

Privacy Policy · Terms and Conditions

Figure 85: Network server settings

4. For **Work Mode**, select **Basics station** and click **Configure Basics Station** server setup to expand the Basics Station settings.


WisGate
RAK7268C
Overview
Configuration
Applications
Gateways

Work mode
○ Packet forwarder
● Basics station
○ Built-in network server

Log Level
Log Level: DEBUG

Basics station server setup
Configure Basics Station server setup.
Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server
Server URL
Server Port
Authentication Mode: None

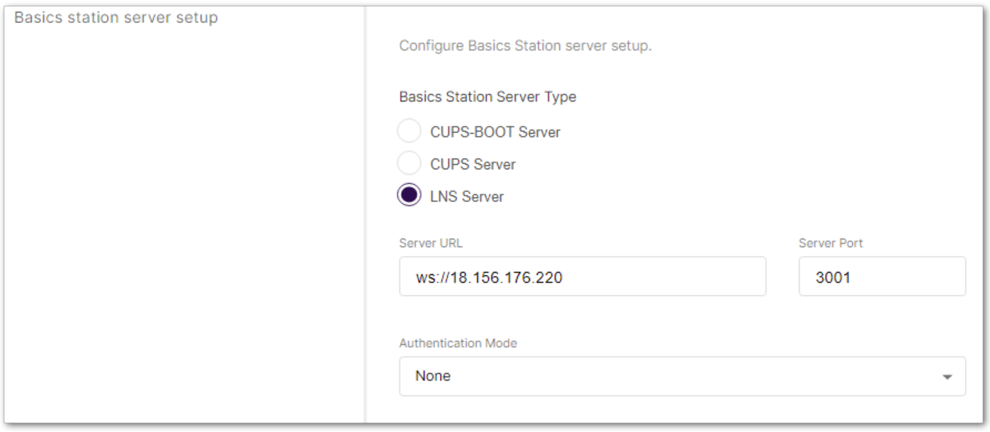
Save changes
©2022 RAKwireless Technology Limited. All Rights Reserved. WisGateOS 2.0
Privacy Policy · Terms and Conditions

Figure 86: Setting Basics Station mode

- **Server** – For server, choose LNS Server.
- **URI** – the address of the ChirpStack server. In this case, the ChirpStack is installed on an AWS cloud instance with public `IP 18.156.176.220` (yours will be different). The URI will be: `ws://18.156.176.220` .

> 📝 **NOTE**
>
> The URL starts with `ws://` in case a plain text connection is used. Using the `wss://` scheme will trigger a TLS connection based on the ``tc.{cert,key,trust} `credentials set.

- **Port** – the port to which the Websocket listens. The port is 3001.
- **Authentication Mode** – Authentication for the ChirpStack server. For this case, you will use no authentication.



Basics station server setup
Configure Basics Station server setup.
Basics Station Server Type
○ CUPS-BOOT Server
○ CUPS Server
● LNS Server
Server URL: ws://18.156.176.220
Server Port: 3001
Authentication Mode: None

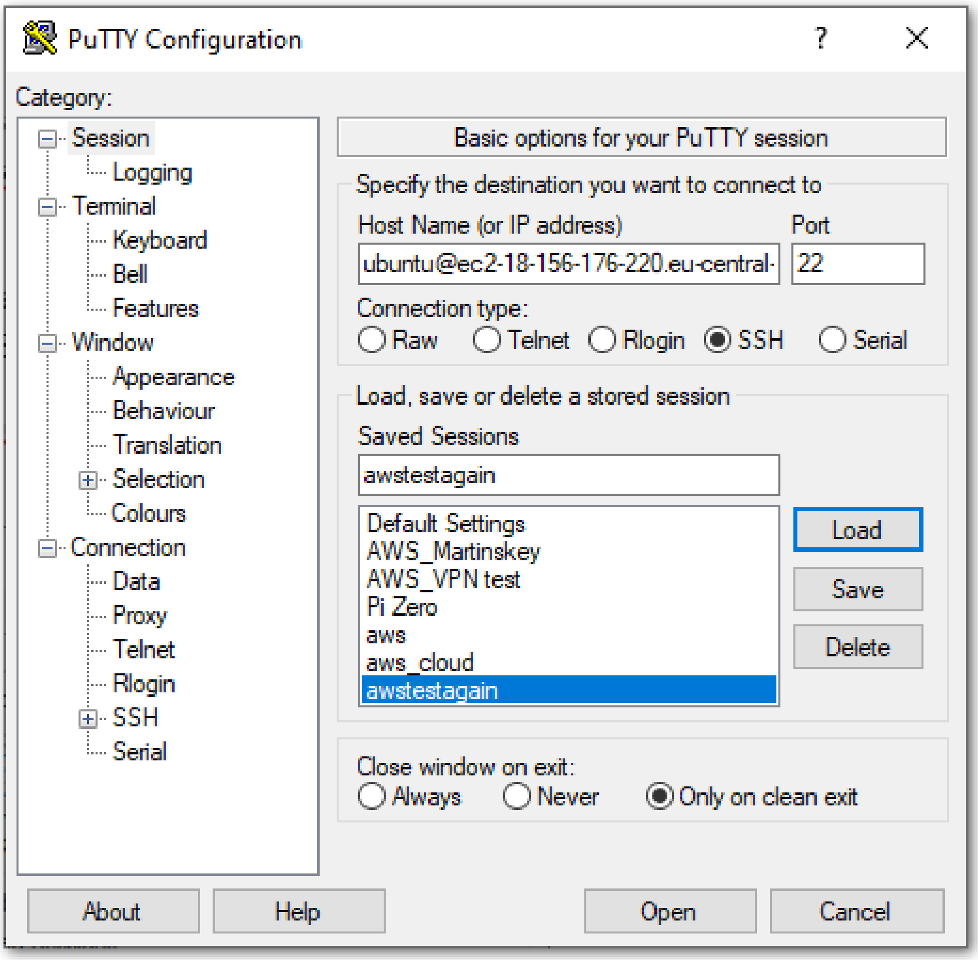
Figure 87: Configuring Basics Station to ChirpStack

5. Click **Save changes** to save the changes.

Now your gateway is configured to work as Basics Station and it is pointed to the ChirpStack gateway bridge. The default installation of the ChirpStack setups backend of the ChirpStack gateway bridge as `semtech_udp` .

To configure the backend of the ChirpStack gateway bridge, you need to access the configuration file of the bridge. To access it, you will need an SSH terminal. In this case, you will use the PuTTy client.

To access the ChirpStack configuration files, you need to access the instance. How to do this is explained in the Knowledge Hub section.



PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ubuntu@ec2-18-156-176-220.eu-central-

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: awstestagain

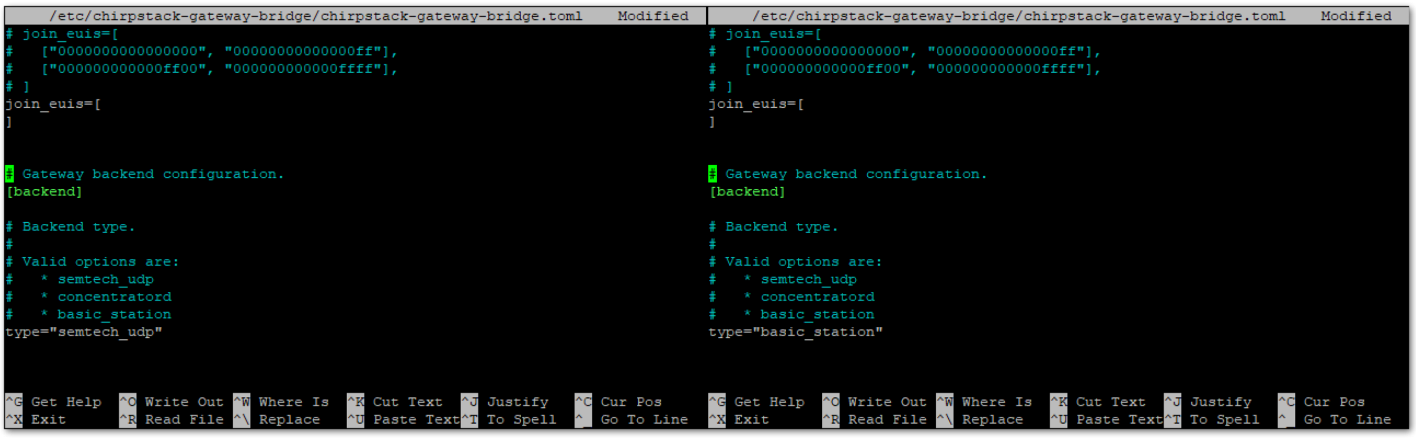
- Default Settings
- AWS_Martinskey
- AWS_VPN test
- Pi Zero
- aws
- aws_cloud
- awstestagain

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 88: PuTTY client

6. In the ChirpStack Gateway bridge webpage, copy the configuration file text and place it in `/etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml`.

7. In the file, find the gateway backend configuration paragraph and replace the type with `basic_station`.

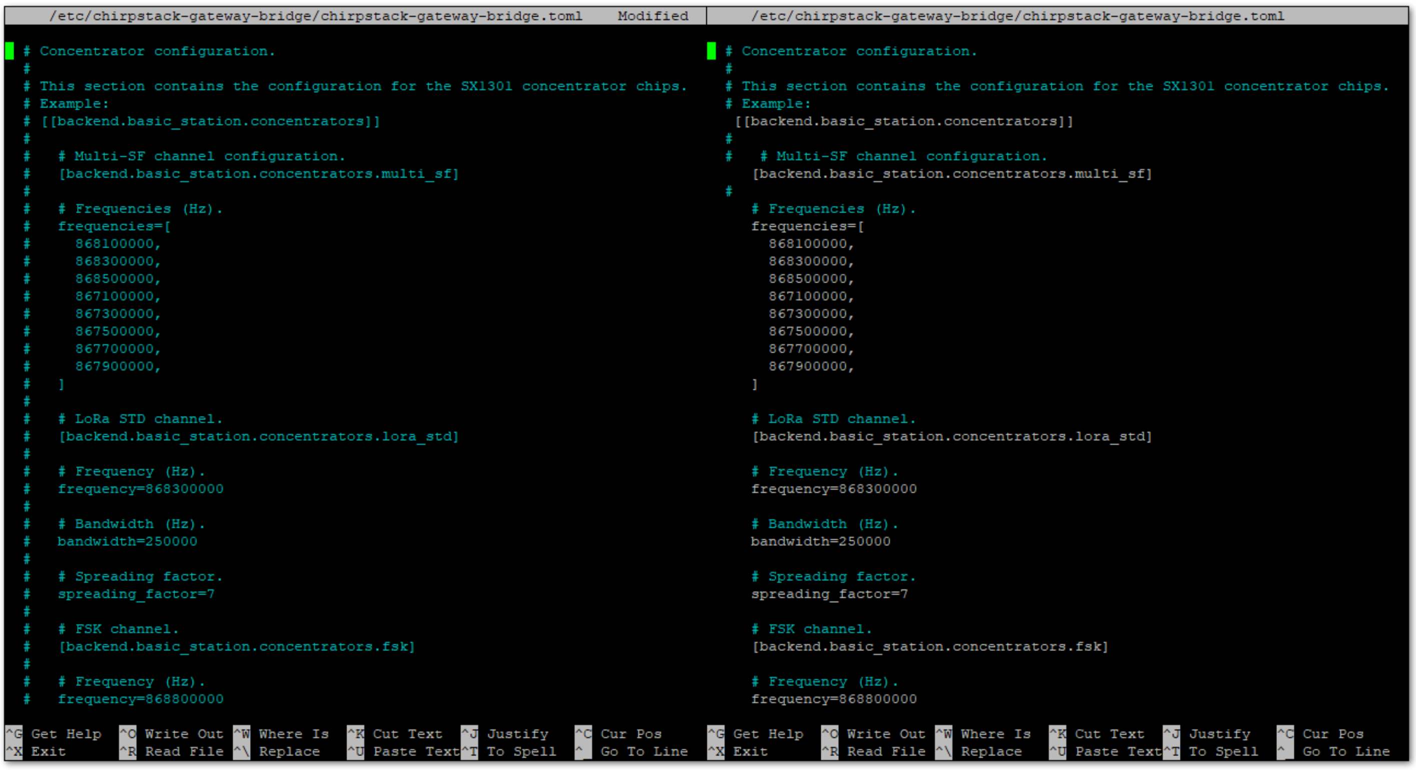


```
    /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml     Modified
# join_euis=[
#   ["0000000000000000", "00000000000000ff"],
#   ["000000000000ff00", "000000000000ffff"],
# ]
join_euis=[
]


# Gateway backend configuration.
[backend]

# Backend type.
#
# Valid options are:
#   * semtech_udp
#   * concentratord
#   * basic_station
type="semtech_udp"



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text^T To Spell    ^_ Go To Line
```

```
    /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml     Modified
# join_euis=[
#   ["0000000000000000", "00000000000000ff"],
#   ["000000000000ff00", "000000000000ffff"],
# ]
join_euis=[
]


# Gateway backend configuration.
[backend]

# Backend type.
#
# Valid options are:
#   * semtech_udp
#   * concentratord
#   * basic_station
type="basic_station"



^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text^T To Spell    ^_ Go To Line
```

Figure 89: Configure gateway bridge type

8. Now scroll down until you find the **Concentrator configuration** paragraph and uncomment the following text as shown below.



```
  /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml    Modified

# Concentrator configuration.
#
# This section contains the configuration for the SX1301 concentrator chips.
# Example:
# [[backend.basic_station.concentrators]]
#
#   # Multi-SF channel configuration.
#   [backend.basic_station.concentrators.multi_sf]
#
#   # Frequencies (Hz).
#   frequencies=[
#     868100000,
#     868300000,
#     868500000,
#     867100000,
#     867300000,
#     867500000,
#     867700000,
#     867900000,
#   ]
#
#   # LoRa STD channel.
#   [backend.basic_station.concentrators.lora_std]
#
#   # Frequency (Hz).
#   frequency=868300000
#
#   # Bandwidth (Hz).
#   bandwidth=250000
#
#   # Spreading factor.
#   spreading_factor=7
#
#   # FSK channel.
#   [backend.basic_station.concentrators.fsk]
#
#   # Frequency (Hz).
#   frequency=868800000
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell ^_ Go To Line
```

```
  /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml

# Concentrator configuration.
#
# This section contains the configuration for the SX1301 concentrator chips.
# Example:
 [[backend.basic_station.concentrators]]
#
#   # Multi-SF channel configuration.
   [backend.basic_station.concentrators.multi_sf]
#
   # Frequencies (Hz).
   frequencies=[
     868100000,
     868300000,
     868500000,
     867100000,
     867300000,
     867500000,
     867700000,
     867900000,
   ]

   # LoRa STD channel.
   [backend.basic_station.concentrators.lora_std]

   # Frequency (Hz).
   frequency=868300000

   # Bandwidth (Hz).
   bandwidth=250000

   # Spreading factor.
   spreading_factor=7

   # FSK channel.
   [backend.basic_station.concentrators.fsk]

   # Frequency (Hz).
   frequency=868800000
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text^T To Spell ^_ Go To Line
```

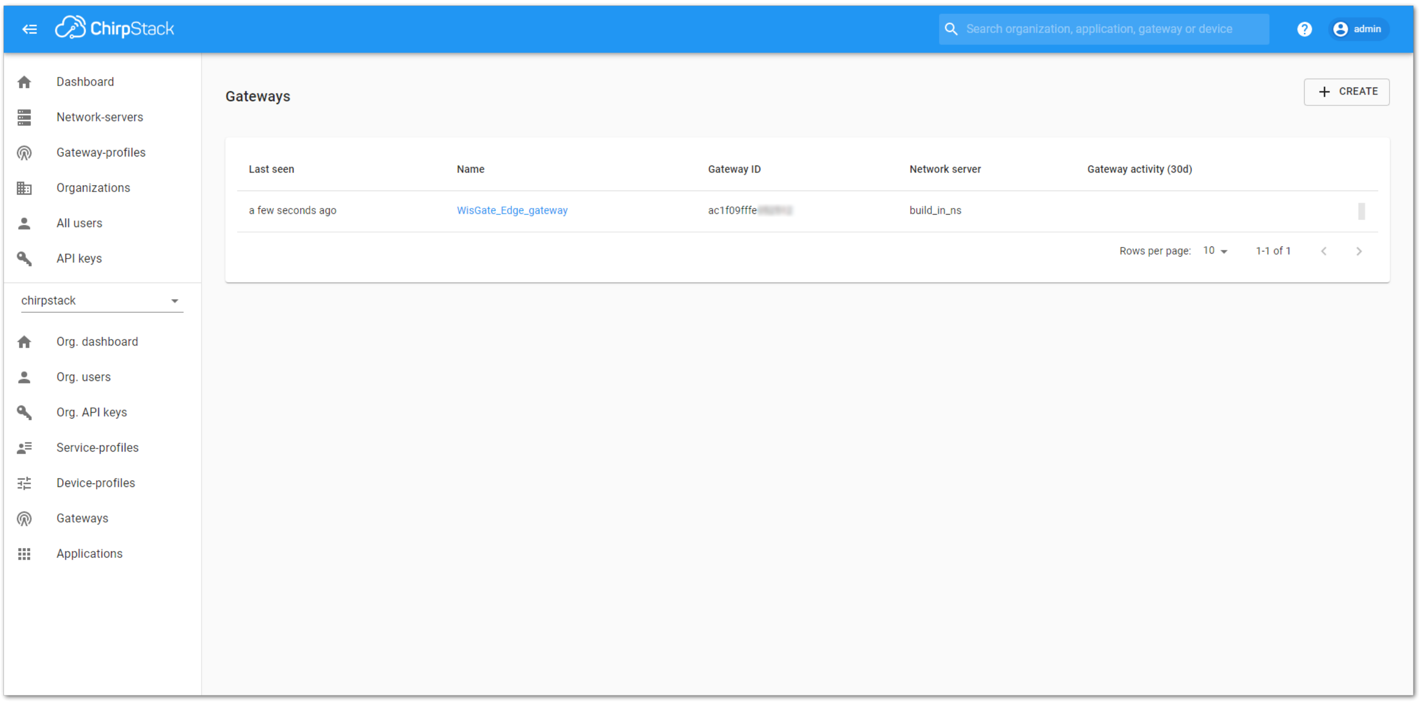
Figure 90: Configuring gateway bridge backend

9. Save and exit the `.toml` file and restart the gateway bridge service to apply the changes by restarting the gateway bridge service with the following command:

```
sudo systemctl restart chirpstack-gateway-bridge.service
```

Now the ChirpStack backend configuration is set to Basics station.

If everything is set correctly, the Last seen status will state a few seconds ago. You can click the gateway name to inspect the gateway traffic.

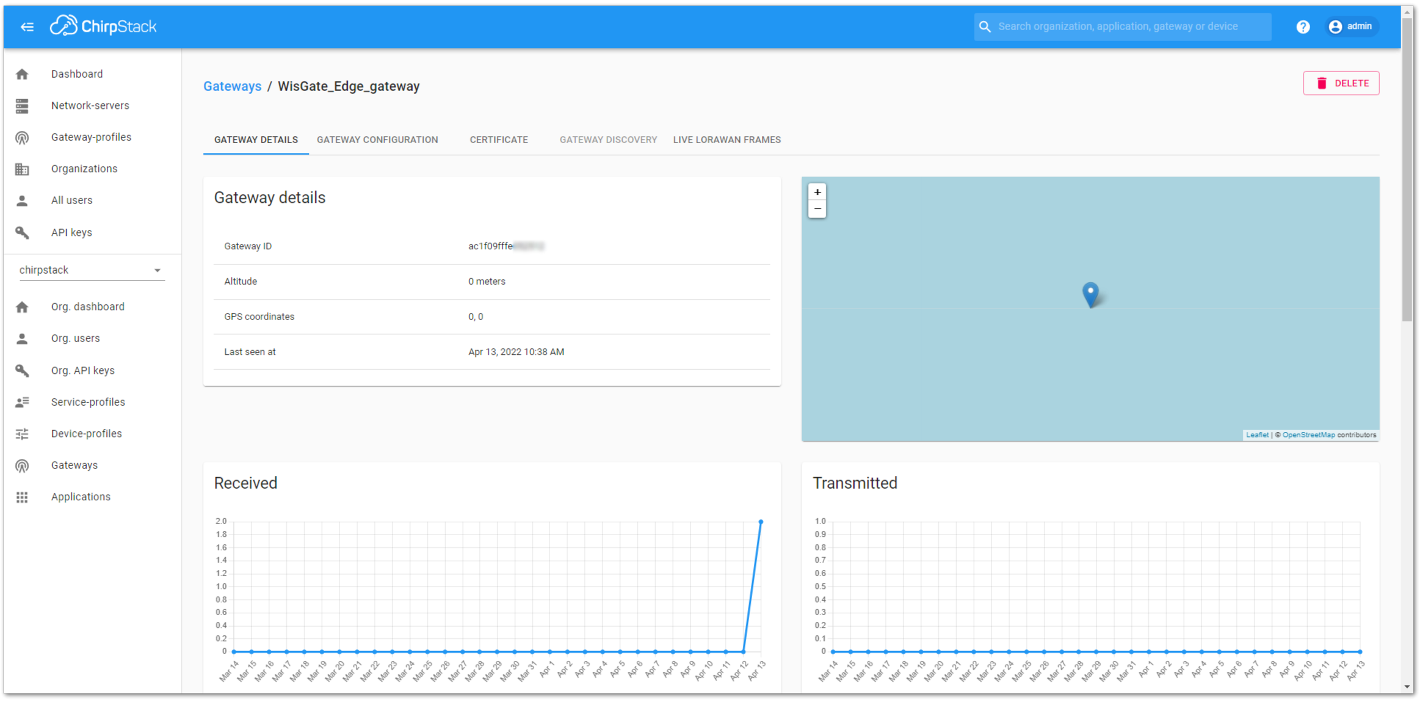


Gateways

Last seen	Name	Gateway ID	Network server	Gateway activity (30d)
a few seconds ago	WisGate_Edge_gateway	ac1f09fffe[redacted]	build_in_ns	

Rows per page: 10   1-1 of 1

Figure 91: Registered gateway



Gateways / WisGate_Edge_gateway

Gateway details

Gateway ID	ac1f09fffe[redacted]
Altitude	0 meters
GPS coordinates	0, 0
Last seen at	Apr 13, 2022 10:38 AM

Figure 92: Gateway details

Now your gateway is connected to the ChirpStack Network server.

ThingPark by Actility

In this section, you will learn how to add RAK7268V2/RAK7268CV2 WisGate Edge Lite 2 in **ThingPark**.

ThingPark is Actility's platform, in which you can register your LoRaWAN gateway and end devices. ThingPark offers a user-friendly dashboard, in which you can monitor different information about the gateway/end device like status, radio traffic, statistics, and more. Together with HTTPS integration, you can send the data received from the end nodes to an Application server for post-processing and visualizing.

For the complete step-by-step tutorial, refer to the How to Add RAK WisGate Edge Gateway V2 in ThingPark - Actility ThingPark Guide .

Last Updated: 4/11/2023, 6:43:05 AM