Laird™
CONNECTIVITY

Setting Up Basics Station on ChirpStack

Sentrius RG1xx Gateway

Version 1.1

REVISION HISTORY

Version	Date	Notes	Contributors	Approver
1.0	6/17/2019	Initial release	Seokwoo Yoon	Chris Boorman
1.1	5/21/2020	Guide revised to reflect LoRaServer.io renaming and Basics Station setup with ChirpStack v3.x	Seokwoo Yoon	Chris Boorman

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

CONTENTS

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

1 Introduction

Laird Connectivity's RG1xx firmware update v93.8.5.18 (GA5.0) introduced Semtech Basics Station as a forwarder, while removing the TTN and MQTT forwarder options.

In this document, we look at using ChirpStack to demonstrate how to use Semtech Basics Station in either insecure or secure mode. We will use Ubuntu 18.04 in free tier of AWS EC2 instance for ChirpStack installation and another Linux machine to ssh into it.

While there are many new features and terms that need to be familiarized, this document does not go through them in detail. For a better understanding on the benefits of using Basics Station, please check the following page;
https://lora-developers.semtech.com/resources/tools/Basics-station/welcome-Basics-station/

Additionally, please consult the documentation published by ChirpStack that describes in detail how to install their software components;

https://www.chirpstack.io/gateway-bridge/

https://www.chirpstack.io/network-server/overview/

https://www.chirpstack.io/application-server/

This document describes setting up an example server for use testing with our RG1xx gateway. The example used is for US SubBand #2 Channel Plan (902.3, 902.5, 902.7, 902.9, 903.1, 903.3, 903.5, 903.7).
Please contact Laird Connectivity support for access to configuration files for various other regions supported by the gateway. Laird Connectivity offers different models for many different regions around the world, see our product page here for more information.

Finally, there may be features included in ChirpStack that are not covered in this document. You should also consult the ChirpStack documentation.

2 Prerequisites

- RG1xx gateway with v93.8.5.21 (or later) firmware
  (see the user guide on the RG1xx product page for upgrading firmware)
- AWS account - https://portal.aws.amazon.com/billing/signup#/start
- Linux based computer
- A LoRaWAN end-device (such as the RM1xx or RS1xx)

3 Create a Micro AWS Instance

To begin, you'll need to create a virtual server instance in Amazon EC2. This is where you'll install ChirpStack. To create the virtual server, complete the following.

1. From the AWS EC2 Console, click **Launch Instance**.
2. Select *Ubuntu Server 18.04 LTS (HVM), SSD Volume Type.*
3. Select an instance type. The default t2.micro option is free tier eligible, and works for this purpose.
4. Click **Review and Launch**.

Next, you'll need to add support for additional ports that are not opened by default. To do so, complete the following.

1. Click on **Edit Security Groups**.
2. Support for the following ports:
3. Add support for the following ports (you only need to complete Type, Protocol, and Port Range):

Type	Protocol	Port Range	Notes
SSH	TCP	22	
HTTP	TCP	80	Required to install and update Let's Encrypt certificates
HTTPS	TCP	443	

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Type	Protocol	Port Range	Notes
Custom UDP Rule	UDP	1700	LoRa Traffic
Custom TCP Rule	TCP	1883	Normal MQTT
Custom TCP Rule	TCP	1884	
Custom TCP Rule	TCP	3001	SBS Websocket
Custom TCP Rule	TCP	8083	
Custom TCP Rule	TCP	8883	Secure MQTT SSL
Custom TCP Rule	TCP	8886	Secure MQTT SSL – self signed

4. Click **Review and Launch**.
5. Click **Launch**.

Once the instance has finished spinning up, you'll need to SSH into the instance and install python2.7.

Note: The ansible playbook used in a later step, requires that the software be able to SSH (as you) into the AWS instance without being prompted for a password or anything else.

In Linux, this is accomplished by editing ~/.ssh/config and adding a corresponding entry pointing to the key file. On the machine that is being used to launch the Ansible installer (which we'll refer to as the installer machine), the example is as follows:

```
host uschirpstack
        hostname ec2-18-191-26-226.us-east-2.compute.amazonaws.com
        user ubuntu
        IdentityFile ~/.ssh/ChirpStack.pem
```

Properly configured, you can connect to the instance by invoking SSH on the installer machine as follows:

```
INSTALLER:~$ ssh uschirpstack
Welcome to Ubuntu 16.08.x
```

Once logged in - update all of the packages to the latest versions

```
ubuntu@AWSInstance:~$ sudo apt update
ubuntu@AWSInstance:~$ sudo apt -y upgrade
```

If prompted about grub config files, choose to keep the local ones. When the updates have finished, reboot as follows:

```
ubuntu@AWSInstance:~$ sudo reboot
```

Log back in and install python:

```
ubuntu@AWSInstance:~$ sudo apt-get -y install python
```

You can now logout. If you choose to set up Duck DDNS for Let's Encrypt, stay logged in and complete the steps in the next section.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

4 Optional: Set Up DDNS Host Name

If you want to use the secure MQTT Forwarder feature, you must have a hostname that resolves to your AWS instance. *Let's Encrypt* purposely blacklists AWS host names, so it's necessary to work around that.

One suitable option is Duck DNS, a free Dynamic DNS forwarder. Navigate to www.duckdns.org and create an account. You can then create a customized hostname that ends in *.duckdns.org*.

An example name might be something such as*: usloratest.duckdns.org*. Once you have created an account and a forwarder entry, copy the token to a file for use later.

By default, the hostname you just created is pointing to your machine, or what it thinks is your machine's IP address. This is probably not what you want.

You will need to SSH into the newly created instance and create a small shell script to update duckdns.org with the correct IP address to forward to.

Log into the instance and create a directory for duckdns, then cd into the directory:

```
ubuntu@AWSInstance:~$ mkdir duckdns
ubuntu@AWSInstance:~$ cd duckdns/
ubuntu@AWSInstance:~/duckdns$
```

Next you'll enter the following command with two field customized to you:

- Use the token string saved from the duckdns.org website in place of TOKEN_GOES_HERE
- Use your custom hostname in place of SOME_NAME (only enter the subdomain – e.g if your hostname is *lora123456.duckdns.org*, enter only *lora123456* in place of SOME_NAME)

```
ubuntu@AWSInstance:~/duckdns$ echo 'echo
url="https://www.duckdns.org/update?domains=SOME_NAME&token=TOKEN_GOES_HERE&ip=" | curl -
k -o ~/duckdns/duck.log -K -' > duck.sh
```

This creates a small shell script called *duck.sh* in the duckdns directory. Change the permissions on duck.sh, then run the shell script to make sure it is working properly, as shown:

```
ubuntu@AWSInstance:~/duckdns$ chmod a+rx duck.sh
ubuntu@AWSInstance:~/duckdns$ ./duck.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100     2    0     2    0     0      2      0 --:--:-- --:--:-- --:--:--     2
ubuntu@AWSInstance:~/duckdns$ cat duck.log
OK
```

Note: The log only contains two characters: *OK*. If there is an error, the log reads *KO*.

If the log reads *OK*, next you'll create a cron job to keep the IP address up to date as follows:

```
(on AWSInstance)
crontab -e
```

If this is your first time editing cron, you will be asked to specify your preferred editor. Next, add the following line to the end of your current cron - which is all comments by default...

```
*/5 * * * * ~/duckdns/duck.sh >/dev/null 2>&1
```

Save the file and exit.

At this point, if you return to Duck DNS you should see the proper AWS IP Address is now assigned to your hostname.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

5 Retrieve the Installation Files

The instructions assume the installer machine is a Linux machine, but these same instructions may work under Windows with minor modifications but are not tested in that environment. This step requires ansible to be installed. If it is not, install it. pip install ansible. Please refer to the Ansible installation guide for more details.
https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

The project files can be retrieved from here: ChirpStack Setup

Either perform a 'git clone' in the target directory, or simply download the repo as a Zip file.

```
INSTALLER:~$ cd Downloads
INSTALLER:~/Downloads$ mkdir ChirpStackInstall
INSTALLER:~/Downloads$ cd ChirpStackInstall
INSTALLER:~/Downloads/ChirpStackInstall$ wget https://github.com/brocaar/chirpstack-
ansible-playbook/archive/master.zip
INSTALLER:~/Downloads/ChirpStackInstall$ unzip master.zip
INSTALLER:~/Downloads/ChirpStackInstall$ cd chirpstack-ansible-playbook-master
```

Since ansible will be used during the setup, ensure that you have at least version 2.1 available

```
INSTALLER:~/Downloads/ChirpStackInstall/chirpstack-ansible-playbook-master $ ansible --
version
ansible 2.4.1.0
```

6 Customize Installation Files

There are several customizations you'll need to make to the installation files to make sure it all works on your Ubuntu instance. Note that you'll perform these customizations on the installer machine, prior to running the install.

6.1 Inventory

Make a working copy of the *inventory.example* file, naming it *inventory*, as shown.

```
cp inventory.example inventory
vim inventory
```

Change the default "example.com" in *inventory* to the host entry in your ~/.ssh/config that corresponds with the AWS instance.

For example, if your host entry looked like this in ~/.ssh/config:

```
host remotelora
   hostname address of AWSInstance
   user ubuntu
   IdentityFile ~/.ssh/ChirpStack.pem
```

In your *inventory* file, you'd replace "***example.com***" with "***remotelora***".

Change the user from ***root*** to ***ubuntu***.

Save the file and exit the editor.

6.2 single_server.yml

Make a working copy of the *group_vars/single_server.example.yml* file, naming it *group_vars/single_server.yml*, as shown.

```
cp group_vars/single_server.example.yml group_vars/single_server.yml
vim group_vars/single_server.yml
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

In the iptables portion of the configuration file, we will unblock 8886 to enable MQTT/SSL self-signed and port 3001 for Semtech Basics Station.

NOTE Port 1700 is included here to provide the context for where you should add ports 8888 and 8886 – do not duplicate it.

Ensure there are no blank lines between the entries, and that the section delimiter "**-**" line up with the previous entries.

```
    -
      port: 1700
      source: 0.0.0.0/0
      protocol: udp
    -
      port: 1883
      source: 0.0.0.0/0
      protocol: tcp
    -
      port: 1884
      source: 0.0.0.0/0
      protocol: tcp
    -
      port: 3001
      source: 0.0.0.0/0
      protocol: tcp
    -
      port: 8883
      source: 0.0.0.0/0
      protocol: tcp
    -
      port: 8886
      source: 0.0.0.0/0
      protocol: tcp
```

Add support for Let's Encrypt certificate generation to MQTT at installation time. To automatically configure signed certificate operation for port 8883, modify this section as follows. (Use your own server and email address)

Having these certificates will also allow you to use secure websockets with Semtech Basics Station later.

```
# mosquitto configuration (MQTT)
#
# this defines the listeners (and protocols) and the usernames that
# are allowed to connect to the MQTT broker and to which MQTT topics
# they are restricted
mosquitto:
  # the full domain by which the MQTT broker is reachable
  # e.g. subdomain.example.com
  fqdn: somename.duckdns.org
  letsencrypt:
    email: youremail@lairdconnect.com
    request: True
```

You can optionally add an MQTT user to monitor client traffic. This can be helpful for debugging.

The following example shows how to add a user named *testguy* with the password *guytest*.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Note: The user chirpstack_app is included here to provide the context for where you should add additional users – do not duplicate it.

Ensure there are no blank lines between the entries, and that the section delimiter "-" lines up with the previous entries

```
  -
    user: chirpstack_app
    password: chirpstack_app
    topics:
      - write application/+/node/+/tx
      - read application/+/node/+/rx
      - read application/+/node/+/join
      - read application/+/node/+/ack
      - read application/+/node/+/error
  -
    user: testguy
    password: guytest
    topics:
      - read application/+/node/+/+
      - read gateway/+/+
      - write application/+/node/+/+
      - write gateway/+/+
```

If using Let's Encrypt - also enter your host and email address in this section. Note - be sure to enable the block by using *True* instead of the default *False*.

```
# ChirpStack Application Server configuration.
chirpstack_application_server:
  # the full domain by which ChirpStack Application Server is reachable
  # e.g. subdomain.example.com
  fqdn: somename.duckdns.org
  letsencrypt:
    email: youremail@lairdconnect.com
    request: True
```

Save the file.

6.3 chirpstack-gateway-bridge.toml

Only one backend can be active at a time, either the legacy Semtech forwarder or the new Semtech Basics Station. By default, the Semtech forwarder is selected via this brief configuration:

```
# See https://www.chirpstack.io/gateway-bridge/install/config/ for a full
# configuration example and documentation.

[integration.mqtt.auth.generic]
server="tcp://127.0.0.1:1883"
username="chirpstack_gw"
password="chirpstack_gw
```

If you want to test Semtech Basics Station, you will need to heavily modify this file.

```
vim roles/chirpstack-gateway-bridge/templates/chirpstack-gateway-bridge.toml
```

This is an example for US SubBand #2 Channel Plan (902.3, 902.5, 902.7, 902.9, 903.1, 903.3, 903.5, 903.7).
Please contact Laird Connectivity support for access to configuration files for various other regions supported by the gateway. Laird Connectivity offers different models for many different regions around the world, see our product page here for more information.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
# See https://www.chirpstack.io/gateway-bridge/install/config/ for a full
# configuration example and documentation.
# Gateway backend configuration.
[backend]

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

  [backend.Basics_station]

  # ip:port to bind the Websocket listener to.
  bind="UPDATE_ME:3001"

  # TLS certificate and key files.
  #
  # When set, the websocket listener will use TLS to secure the connections
  # between the gateways and ChirpStack Gateway Bridge (optional).
  tls_cert=""
  tls_key=""

  # TLS CA certificate.
  #
  # When configured, ChirpStack Gateway Bridge will validate that the client
  # certificate of the gateway has been signed by this CA certificate.
  ca_cert=""

 # Ping interval.
 ping_interval="1m0s"

  # Read timeout.
  #
  # This interval must be greater than the configured ping interval.
  read_timeout="1m5s"

  # Write timeout.
  write_timeout="1s"

  # Region.
  #
  # Please refer to the LoRaWAN Regional Parameters specification
  # for the complete list of common region names.
  region="US_902_928"

  # Minimal frequency (Hz).
  frequency_min=902000000

  # Maximum frequency (Hz).
  frequency_max=928000000

   #Concentrator configuration.

  # This section contains the configuration for the SX1301 concentrator chips.
  # Example:
   [[backend.Basics_station.concentrators]]

     # Multi-SF channel configuration.
     [backend.Basics_station.concentrators.multi_sf]
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
      # Frequencies (Hz).
      frequencies=[
        903900000,
        904100000,
        904300000,
        904500000,
        904700000,
        904900000,
        905100000,
        905300000,
      ]

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

     # Frequency (Hz).
     frequency=904600000

     # Bandwidth (Hz).
     bandwidth=500000

     # Spreading factor.
     spreading_factor=7

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

     # Frequency (Hz).
     #frequency=868800000
[integration.mqtt.auth.generic]
server="tcp://127.0.0.1:1883"
username="chirpstack_gw"
password="chirpstack_gw"
```

Note: You must replace UPDATE_ME with the instances correct IP address later.

```
# ip:port to bind the Websocket listener to.
 bind="UPDATE_ME:3001"
```

Note the default values for the certs above:

```
tls_cert=""
tls_key=""
ca_cert=""
```

We'll come back to these later for enabling WSS or Secure Web Sockets.

6.4 chirpstack-application-server.toml

This step is purely optional. To generate JWT, open the file below.

```
vim roles/chirpstack-application-server/templates/chirpstack-application-server.toml
```

And, you can generate a new one to add to the file with the following command.

```
openssl rand -base64 32
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

6.5 chirpstack-network-server.toml

This file controls the regulatory domain the server is configured for. By default, it is configured for *EU*. You can configure this setting for the US. Open the editor with this file as follows:

```
vim roles/chirpstack-network-server/templates/chirpstack-network-server.toml
```

For US operation on subBand #2, you must make the following changes:

```
name="US_902_928"

 [network_server.network_settings]
 # Enable only a given sub-set of channels
 #
 # Use this when ony a sub-set of the by default enabled channels are being
 # used. For example when only using the first 8 channels of the US band.
 #
 # Example:
 # enabled_uplink_channels=[0, 1, 2, 3, 4, 5, 6, 7]
 # note, the 65 in the below example configures the server to enable the 500kHz channel
corresponding to subBand #2
 # the above example (w/out 65 would enable ALL of the 500kHz channels - which is not
compatible with the Gateway)
 enabled_uplink_channels=[8, 9, 10, 11, 12, 13, 14, 15, 65]

 # Extra channels to use for ISM bands that implement the CFList
 #
 # Use this for LoRaWAN regions where it is possible to extend the by default
 # available channels with additional channels (e.g. the EU band).
 # Note: the min_dr and max_dr are currently informative, but will be enforced
 # in one of the next versions of ChirpStack!
 #
 # Example:
 # [[network_server.network_settings.extra_channels]]
 # frequency=867100000
 # min_dr=0
 # max_dr=5

 # [[network_server.network_settings.extra_channels]]
 # frequency=867300000
 # min_dr=0
 # max_dr=5

 # [[network_server.network_settings.extra_channels]]
 # frequency=867500000
 # min_dr=0
 # max_dr=5

 # [[network_server.network_settings.extra_channels]]
 # frequency=867700000
 # min_dr=0
 # max_dr=5

 # [[network_server.network_settings.extra_channels]]
 # frequency=867900000
 # min_dr=0
 # max_dr=5
```

After these files are configured, run the ansible installer from your installer machine with the following command:

```
ansible-playbook -i inventory full_deploy.yml
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

This takes a little while, it needs to SSH into the AWS instance, and from there it downloads and configures the ChirpStack project files.

Ideally, no errors are displayed.

Once the installation finishes, if you are unable to connect to the server, reboot the instance since some services may be out of sync.

7 KNOWN ISSUE

The following is a known bug as of November 22, 2019:

The current server package as of November 22, 2019 has an error which will lead to the installation failing with:

Note: This is an older **pre-chirpstack** error message - but the latest server install still has this same issue.

```
TASK [postgresql : create role]
*********************************************************************************
************************
An exception occurred during task execution. To see the full traceback, use -vvv. The
error was: HINT:  Remove UNENCRYPTED to store the password in encrypted form instead.
fatal: [deleteme]: FAILED! => {"changed": false, "module_stderr": "Traceback (most recent
call last):\n  File \"/tmp/ansible_1Zz4Dk/ansible_module_postgresql_user.py\", line 855,
in <module>\n    main()\n  File
\"/tmp/ansible_1Zz4Dk/ansible_module_postgresql_user.py\", line 816, in main\n
role_attr_flags, encrypted, expires, conn_limit)\n  File
\"/tmp/ansible_1Zz4Dk/ansible_module_postgresql_user.py\", line 270, in user_add\n
cursor.execute(query,
query_password_data)\n  File \"/usr/lib/python2.7/dist-packages/psycopg2/extras.py\",
line 144, in execute\n    return super(DictCursor, self).execute(query,
vars)\npsycopg2.NotSupportedError: UNENCRYPTED PASSWORD is no longer supported\nLINE 1:
CREATE USER \"loraserver_as\" WITH UNENCRYPTED PASSWORD 'loras...\n
^\nHINT:  Remove UNENCRYPTED to store the password in encrypted form instead.\n\n",
"module_stdout": "", "msg": "MODULE FAILURE", "rc": 1}
```

This issue is due to the fact that PostgreSQL no longer permits unencrypted passwords, which is what ANSIBLE defaults to. To fix this issue, make edits to the following file on the installer machine:

```
vim roles/postgresql/tasks/create_db.yml
```

Modify the following entry by adding "encrypted: yes". Note the double curly brackets breaks wiki formatting.

```
- name: create role
  postgresql_user:
    name: "{{ item.user }}"
    password: "{{ item.password }}"
    encrypted: yes
    role_attr_flags: LOGIN
  become_user: postgres
  no_log: true
```

Re-Run the installer.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

8 QUICK TEST

The unconfigured server is now running and accessible. Test the server installation by opening a web browser and navigating to https://YOUR_HOSTNAME.

The default credentials for logging in are:

- Username: admin
- Password: admin

9 CONFIGURING THE NEW SERVER

The server features a navigation sidebar - click on the icon to the left of the logo to reveal/hide it.

9.1 Creating a Network Server Configuration

To create a network server configuration, follow these steps:

1. Click on **Network Servers** in the top of the left sidebar
2. Click **+ Add**.
3. Fill in a free form name.
4. Fill in the address. Usually *localhost:8000* is used.
5. Ignore the *Gateway Discovery* and *TLS Certificates* tabs for now, click **ADD NETWORK-SERVER.**

9.2 Creating a Service Profile

To create a service profile, follow these steps:

1. Click on **Service-profiles** in the left sidebar**.**
2. Click **+ CREATE.**
3. Fill in the service-profile name. E.g. *service-profile-1*
4. Under *Network-server*, click the selector and select the newly created Network Server configuration.
5. Optionally, click **Add gateway meta-data** if you would like that data to be added to each packet sent to the application server.
6. Click **CREATE SERVICE-PROFILE** when finished making your selections.

Everything remaining in the Service Profile is optional.

9.3 Configuring Gateways

9.3.1 Creating a Gateway Profile

To create a gateway profile, follow these steps:

1. Click on **Gateway-profiles** in the left sidebar.
2. Click **+ CREATE.**
3. Fill in a profile name.
4. Fill in the active channels (example for sub-band #2: 8, 9, 10, 11, 12, 13, 14, 15).
5. Under "Network-server", click the selector and select the newly created Network Server configuration.
6. Click **CREATE GATEWAY-PROFILE**.

9.3.2 Adding a Gateway

To add a gateway, follow these steps:

1. Click on **Gateways** on the left sidebar.
2. Click on **+ CREATE**.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

3. Fill in a gateway name, using only letters, numbers and dashes.
4. Enter an optional Gateway description.
5. Fill in the Gateway EUI.
6. Select the previously configured Network Server.
7. Select the previously configured Gateway Profile

The rest of the settings are optional. Click **CREATE GATEWAY** when finished making your selections.

9.3.3 Using Legacy Semtech Forwarder

If you configured the server for the legacy Semtech forwarder. (HERE) You can now configure your Gateway to point to the ChirpStack Server - and it should connect and start passing handshaking messages.

Select the Semtech UDP Forwarder on the gateway and fill in *yourhostname.duckdns.org* as the server address and leave the port settings at the default of 1700.

The Gateway UI should show "Gateway Connected true"

If it does not, verify your gateway configuration file:

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

It should show the legacy semtech forwarder is being used:

(excerpt from top of configuration file)

```
# Valid options are:
#   * semtech_udp
#   * Basics_station
type="semtech_udp"
```

If you modify this file, be sure to restart the bridge:

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

Jump to Adding an Application.

9.3.4 Using Semtech Basics Station Forwarder – insecure websockets mode

The new Semtech Basics Station forwarder can be run in either insecure websockets mode, or in secure websockets mode.

The insecure websockets mode is the simplest to set up for a quick test.

Regardless of which mode you use, you MUST update the IP address in the configuration file to reflect the servers current address.

```
ubuntu@ip-172-31-26-104:~$ ifconfig eth0 | grep "inet "
        inet 172.31.26.104  netmask 255.255.240.0  broadcast 172.31.31.255
```

In this case, you would now edit the configuration file and replace "UPDATE_ME" with "172.31.26.104"

If you modify this file, be sure to restart the bridge:

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

You can now use this URL in the Gateway's TC Server configuration field: (assuming serverhostname is your custom duckdns host entry)

```
ws://serverhostname.duckdns.org:3001
```

When the Gateway UI shows "Gateway Connected" is "true" - you can then continue configuring the server via its GUI.

Jump to Adding an Application.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

9.3.5 Using Semtech Basics Station Forwarder – secure websockets mode

Using Semtech Basics Station with Secure Websockets is more complicated than the other method.

This assumes you have updated the IP address in the configuration file, if not, see the previous section.

To enable secure websockets, you will need a set of certificates, and it's recommended to use the set installed by the Let's Encrypt certbot for this document.

By default, those certificates are placed in a directory inaccessible to the chirpstack bridge daemon.

Simply manually copying the certs to a location where the bridge user can read them would work - until they expire, in which case you'd have to do it again.

The Let's Encrypt certbot by default stores the certificates (fullchain.pem, privkey.pem) in

```
/etc/letsencrypt/live/YOURHOSTNAME/
```

We will place a copy of them in

```
/etc/chirpstack-gateway-bridge/certs
```

To get started, edit the bridge configuration file:

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

and modify

```
# When set, the websocket listener will use TLS to secure the connections
 # between the gateways and ChirpStack Gateway Bridge (optional).
 tls_cert=""
 tls_key=""
```

from the default above to

```
# When set, the websocket listener will use TLS to secure the connections
 # between the gateways and ChirpStack Gateway Bridge (optional).
 tls_cert="/etc/chirpstack-gateway-bridge/certs/fullchain.pem"
 tls_key="/etc/chirpstack-gateway-bridge/certs/privkey.pem"
```

Save the file.

The Let's Encrypt certbot can run shell scripts before or after updating the certificates. We're going to create a shell script that will be ran AFTER the certificates are updated.

This only needs to be done once, and in the future, whenever the certs are updated, a copy will be automatically made available to the bridge user.

Create the new shell script:

```
sudo vim /etc/letsencrypt/renewal-hooks/post/update_chirp_certs
```

and add the following to it: (replacing YOURHOSTNAME with the correct fully qualified hostname)

```
#!/bin/sh
# Manually copy the latest certs from
# /etc/letsencrypt/live/YOURHOSTNAME/
# to /etc/chirpstack-gateway-bridge/certs

# Where we get the certs from
source_dir=/etc/letsencrypt/live/YOURHOSTNAME
# Destination directory
daemon_cert_root=/etc/chirpstack-gateway-bridge/certs

if [ ! -d "$daemon_cert_root" ]; then
  mkdir -p $daemon_cert_root
```


Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

```
  chown gatewaybridge:gatewaybridge $daemon_cert_root
fi

# Make sure the certificate and private key files are
# never world readable, even just for an instant while
# we're copying them into daemon_cert_root.
umask 077

cp "$source_dir/fullchain.pem" "$daemon_cert_root"
cp "$source_dir/privkey.pem" "$daemon_cert_root"

# Apply the proper file ownership and permissions for
# the daemon to read its certificate and key.
chown gatewaybridge:gatewaybridge $daemon_cert_root/*
chmod 400 $daemon_cert_root/*

# restart the chirpstack bridge service
service chirpstack-gateway-bridge restart >/dev/null
```

Be sure to set the shell script as executable:

```
sudo chmod +x /etc/letsencrypt/renewal-hooks/post/update_chirp_certs
```

Before moving on, you can test the post hook handler by simulating a certificate update

```
sudo certbot renew --dry-run
```

The very last line of output should be:

```
Running post-hook command: /etc/letsencrypt/renewal-hooks/post/update_chirp_certs
```

If this is missing, double check you made the update_chirp_certs file executable.

Now - verify the certs are where they belong and they are owned by the gatewaybridge user:

```
ubuntu@ip-172-31-16-6:~$ sudo ls -l /etc/chirpstack-gateway-bridge/certs/
total 8
-r-------- 1 gatewaybridge gatewaybridge 3578 Jan 30 19:11 fullchain.pem
-r-------- 1 gatewaybridge gatewaybridge 1708 Jan 30 19:11 privkey.pem
```

Finally, verify that the bridge service is running:

```
ubuntu@ip-172-31-16-6:~$ ps axuw | grep chirpstack-gateway-bridge
gateway+  4370  0.0  0.9 113992  9604 ?        Ssl  19:11   0:00 /usr/bin/chirpstack-
gateway-bridge
ubuntu    4411  0.0  0.1  14856  1144 pts/0    S+   19:14   0:00 grep --color=auto
chirpstack-gateway-bridge
```

/usr/bin/chirpstack-gateway-bridge indicates that it is running fine.

Restart the bridge.

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

You can now use this URL in the Gateway's TC Server configuration field: (assuming serverhostname is your custom duckdns host entry)

Note: We are using ***wss*** and NOT ***ws***.

```
wss://serverhostname.duckdns.org:3001
```

Finally, you need to upload the proper CA certificate to the Gateway so it can verify the certs on the server.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

Assuming you are using Let's Encrypt Certs - retrieve the CA certificate from [HERE] and upload it to the gateway via the "Upload Basics Station LNS Server Certificate File" file dialogue.

When the Gateway UI shows "Gateway Connected" is "true" - you can now continue configuring the server via its GUI.

9.4 Adding an Application

To add an application, follow these steps:

1. Click on **Applications** in the left sidebar.
2. Click on **+ CREATE**.
3. Fill in an application name, using only letters, numbers and dashes.
4. Enter an optional application description.
5. Using the **Service-profile** selector, select a service profile to use.
6. Click **CREATE APPLICATION** when finished making your selections.

9.5 Adding Devices (Nodes)

Before adding a device, you need to create a profile to define its behavior

9.5.1 Creating a Device Profile

To create a device profile, follow these steps:

1. Click on **Device-profiles** in the left sidebar.
2. Click on **+ CREATE**.
3. Fill in a name for this device profile, example **OTAA Profile.**
4. Select a Network-server using the selector.
5. Optionally, Specify the LoRaWAN MAC Version. Laird uses 1.0.0.
6. Optionally, specify the LoRaWAN Regional Parameters revision.
7. Optionally, specify the Max EIRP supported by the client.
8. Click **Join (OTAA / ABP)**.
9. Determine whether this profile is for OTAA or ABP by selecting or deselecting **Supports Join.**
10. Likewise, determine if this profile will support Class B or C and configure accordingly.

Save the new profile by clicking **CREATE DEVICE-PROFILE**.

9.5.2 Adding a New Device

To add a new device, follow these steps:

1. Click on **Applications** in the left sidebar.
2. Click on an Application name to create a device within the application.
3. Click on **+ CREATE.**
4. Fill in a device name, using only letters, numbers and dashes.
5. Enter an optional application description.
6. Enter the Device EUI.
7. Select a Device-profile.
8. Optionally, check the "Disable frame-counter validation" box (Not recommended).
9. Click **SUBMIT** to continue.
10. Now, fill in a hexadecimal string for use as the corresponding clients Application Key.
11. Click **SUBMIT**.

Note For LoRaWAN MAC 1.0.0 clients such as the RM191, all that is needed is the application key. At this point, try joining the client to the server.

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610