Setting up LoRaServer.io on AWS

RG1xx Gateway

Version 1.0

REVISION HISTORY

Version	Date	Notes	Contributors	Approver
1.0	6/17/2019	Initial release	Jim Veneskey	Seokwoo Yoon

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 gateway features an MQTT Forwarder that is designed to work with the open-source LoRaServer.io. While it's simple to configure the forwarder in RG1xx gateway, setting up LoRaServer.io is more complicated. This document details how to install LoRaServer on AWS using an Ubuntu image and configuring it with the RG1xx. It is strongly recommended to have a secure communication channel such as SSL when using MQTT, which is also covered in this guide.

By the end of this guide, any LoRaWAN end-devices should be able to pass data to the LoRa server using port 8883 or 8886 for MQTT through SSL.

2 Prerequisites

- RG1xx gateway with the latest firmware (see the user guide on the RG1xx product page for upgrading firmware)
- AWS account - https://portal.aws.amazon.com/billing/signup#/start
- A LoRaWAN end-devices (such as the RM1xx)

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 LoRaServer. 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	
Custom TCP Rule	TCP	8080	Used for the management interface
Custom UDP Rule	UDP	1700	LoRa Traffic
Custom TCP Rule	TCP	1883	Normal MQTT (optional)
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.

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

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

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

```
INSTALLER:~$ ssh usbrocaar
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.

4 Optional: Setup DDNS Host Name

If you want to use the secure MQTT Forwarder feature, you need to 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 like: "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 2 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.

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.

The project files can be retrieved from here: LoRa Server 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 BrocaarInstall
INSTALLER:~/Downloads$ cd BrocaarInstall
INSTALLER:~/Downloads/BrocaarInstall$ wget https://github.com/brocaar/loraserver-
setup/archive/master.zip
INSTALLER:~/Downloads/BrocaarInstall$ unzip master.zip
INSTALLER:~/Downloads/BrocaarInstall$ cd loraserver-setup-master/
```

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

```
INSTALLER:~/Downloads/BrocaarInstall/loraserver-setup-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.

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

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.

The following is an example of ~/.ssh/config:

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

In the iptables portion of the configuration file, if you'd like to enable MQTT/SSL you will need to unblock the ports. The iptables portion is redundant in the case of AWS security setup, but it is enabled by default and easy to modify.

Port 8883 will be used for MQTT/SSL with signed certificates, and port 8886 will be used with MQTT/SSL with self-signed certificates. Both of these ports are optional, but are required for fully utilizing the MQTT Secure Forwarder feature in the Gateway.

To add these ports, add the following text to single_server.yml.

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 "-" lines up with the previous entries.

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


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

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*.

Note: The user loraserver_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: loraserver_app
    password: loraserver_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 you want to use MQTT SSL with signed certificates, you can install them by enabling the "Let's Encrypt" section.

Substitute the fully qualified DNS name you created on www.duckdns.org for DUCK_NAME_HERE.

Use an active email address so you can receive news of any issues updating the certs. Be sure to enable the block by changing the default *False* to *True.*

```
# lora-app-server configuration
lora_app_server:
  letsencrypt:
    domain: DUCK_NAME_HERE
    email: YOUR_EMAIL
    accept_letsencrypt_tos: True
```

Save the file.

6.3 lora-gateway-bridge.toml

Review this file and modify the usernames if you changed them in single_server.yml

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


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

6.4 lora-app-server.toml

It is recommended to update the JWT Secret in this file. It is optional.

```
vim roles/lora-app-server/templates/lora-app-server.toml
```

You can generate a new one to add to the file with the following command.

```
openssl rand -base64 32
```

6.5 lora-app-server.toml

This file controls the 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/loraserver/templates/loraserver.toml
```

For US operation, modify the *name* field as follows:

```
name="US_902_928"
```

7 Run Installer

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

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

This will take a little while, it needs to SSH into the AWS instance, and from there it will download and configure the LoRa Server project files.

8 Bugs as of 2018/12/18

8.1 Installation Fails

This bug appears to be fixed as of January 2019. The server package of December 18, 2018 has errors which may lead to the installation failing.

```
TASK [postgresql : create role]
*********************************************************************************
*****************
fatal: [newusbrocaar]: FAILED! => {"censored": "the output has been hidden due to
the fact that 'no_log: true' was specified for this result", "changed": false}
```

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
```

Add the line *encrypted: yes* as shown below:

```
- name: create role
  postgresql_user:
    name: "Template:Item.user"
    password: "Template:Item.password"
    encrypted: yes
    role_attr_flags: LOGIN
  become_user: postgres
  no_log: true
```


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

Re-Run the installer.

8.2 Certs Directory does not exist

This error is still present as of 2019/01/28. The task “copy certificate files” fails, with the warning that the directory *etc/lora-app-server/certs* does not exist.

```
TASK [lora-app-server : copy certificate files]
*********************************************************************************
********************
failed: [usbrocaar] (item={u'dest': u'http.pem', u'src': u'fullchain.pem'}) =>
{"changed": false, "item": {"dest": "http.pem", "src": "fullchain.pem"}, "msg":
"Destination directory /etc/lora-app-server/certs does not exist"}
failed: [usbrocaar] (item={u'dest': u'http-key.pem', u'src': u'privkey.pem'}) =>
{"changed": false, "item": {"dest": "http-key.pem", "src": "privkey.pem"}, "msg":
"Destination directory /etc/lora-app-server/certs does not exist"}
```

This one is also easy to fix. SSH into the new instance and do the following:

```
sudo su
mkdir /etc/lora-app-server/certs
chown appserver:appserver /etc/lora-app-server/certs
exit
```

Re-Run the installer. Once the installation finishes, if you are unable to connect to the server, reboot the instance since some services are out of sync.

9 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. (If the above URL doesn't work try http://YOUR_HOSTNAME:8080).

The default credentials for logging in are:

- Username: admin
- Password: admin

10 Configuring The New Server

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

10.1 Creating a Network Server Configuration

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. Typically *localhost:8000* is used.
5. Ignore the *Gateway Discovery* and *TLS Certificates* tabs for now, and proceed to click **ADD NETWORK-SERVER.**

10.2Creating a Service Profile

1. In the left sidebar, click on **Service-profiles.**
2. Click **+ CREATE.**
3. Fill in the service-profile name. E.g. *service-profile-1*

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

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.

10.3Configuring Gateways

10.3.1 Creating a Gateway Profile

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 subBand #2: 8, 9, 10, 11, 12, 13, 14, 15).
5. Click **CREATE GATEWAY-PROFILE**.

10.3.2Adding a Gateway

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

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

At this point, you can configure the Gateway to point to the LoRa Server - and it should connect and start passing handshaking messages.

10.4 Adding an Application

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. Optionally, select a Payload CODEC. Select from *None* (Default), *Cayenne LPP*, or *Custom Javascript functions*.
7. Click **CREATE APPLICATION** when finished making your selections.

The custom Javascript functions are similar to the feature used in The Things Network.

10.5Adding Devices (Nodes)

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

10.5.1 Creating a Device Profile

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.**

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

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**.

10.5.2Adding a new Device

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.

11 Secure MQTT Forwarder

In order to fully utilize the potential of the Secure MQTT forwarder feature added in GA2, Mosquitto needs to be configured to use certificates.

11.1 Let's Encrypt Signed Certificates

SSH into the Ubuntu instance on AWS. Assuming that the *Let's Encrypt* certificate option was enabled, there should be certificates in the following directory.

```
/etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME
```

To add the certfiicates, edit the listeners.conf file:

```
sudo vim /etc/mosquitto/conf.d/listeners.conf
```

Add the certs as shown below. Note that MQTT and websockets listeners are shown to provide context.

```
listener 1883 0.0.0.0
protocol mqtt

listener 1884 0.0.0.0
protocol websockets
```


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

```
listener 8883
#### signed certificates
cafile   /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/chain.pem
certfile /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/cert.pem
keyfile  /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/privkey.pem
```

Save the file and restart mosquito.

```
sudo systemctl restart mosquitto
```

Mosquitto is now configured to utilize the Let's Encrypt signed certificates on port 8883. To ensure the Let's Encrypt certificates are kept updated automatically via cron, update the root user's crontab:

```
sudo crontab -e
```

Add the following line (if there is already an entry for certbot, replace it with this one):

```
15 3 * * * certbot renew --noninteractive --post-hook "systemctl restart
mosquitto"
```

Configure the LoRa Gateway to use the MQTT Forwarder as follows:

1. Fill in the server address using this format: ssl://FULLY QUALIFIED HOSTNAME:8883
2. Assuming credentials are at default, use *loraserver_gw* as both the username and Password.
3. Click **Update.**

Once the LoRa status indicator on the Gateway shows that it is connected, start the LoRa client, and verify that the client can join and pass traffic.

11.2 Self Signed Certificates

Before utilizing the Self Signed Certificates, they need to be created on the AWS instance.

11.2.1 Generating the CA Keys

SSH into the Ubuntu instance on AWS. Generate the key pair as follows:

```
openssl genrsa -des3 -out ca.key 2048
```

You will be prompted for a password/passphrase.

11.2.2 Creating a CA Certificate

Create a certificate for the CA using the new CA keys as follows:

```
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
```

When prompted, the crucial setting is the common name. This must be set to be the hostname for the server, as follows:

```
Common Name (e.g. server FQDN or YOUR name) []:yourhostname.duckdns.org
```

11.2.3 Creating a Server Key Pair

Create a server key pair for use by mosquito as follows:

```
openssl genrsa -out server.key 2048
```

11.2.4 Certificate Signing Request

Create a certificate signing request using the new server key. Again, make sure the CN matches the server FQDN.

```
openssl req -new -out server.csr -key server.key
```


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

11.2.5 Signing Server Certificate

Before the server certificate can be used, it needs to be signed by the local CA authority. Enter the following command:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
server.crt -days 360
```

This command generates the following files:

```
ubuntu@AWSInstance:~/bak$ ls -l
total 24
-rw-rw-r-- 1 ubuntu ubuntu 1338 May  2 12:14 ca.crt
-rw-rw-r-- 1 ubuntu ubuntu 1751 May  2 12:14 ca.key
-rw-rw-r-- 1 ubuntu ubuntu   17 May  2 12:14 ca.srl
-rw-rw-r-- 1 ubuntu ubuntu 1220 May  2 12:14 server.crt
-rw-rw-r-- 1 ubuntu ubuntu 1013 May  2 12:14 server.csr
-rw-rw-r-- 1 ubuntu ubuntu 1679 May  2 12:14 server.key
ubuntu@AWSInstance:~/bak$
```

11.2.6 Installing the Certificates

Copy the files you need into the mosquitto certificates directory:

```
sudo cp ca.crt server.crt server.key /etc/mosquitto/certs
```

11.2.7 Configure Mosquittot to use the certs

Add the certificates by editing listeners.conf. First, open listeners.conf for editing as follows.

```
sudo vim /etc/mosquitto/conf.d/listeners.conf
```

Add the certification records as shown below. The Let's Encrypt certifications are shown here for context.

```
listener 8883
#### signed certificates
cafile   /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/chain.pem
certfile /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/cert.pem
keyfile  /etc/letsencrypt/live/FULLY QUALIFIED HOSTNAME/privkey.pem

listener 8886
#### self signed certificates
cafile   /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile  /etc/mosquitto/certs/server.key
```

Save the file and restart mosquito as follows:

```
sudo systemctl restart mosquitto
```

Mosquitto should now be configured to utilize the self-signed certificates on port 8886

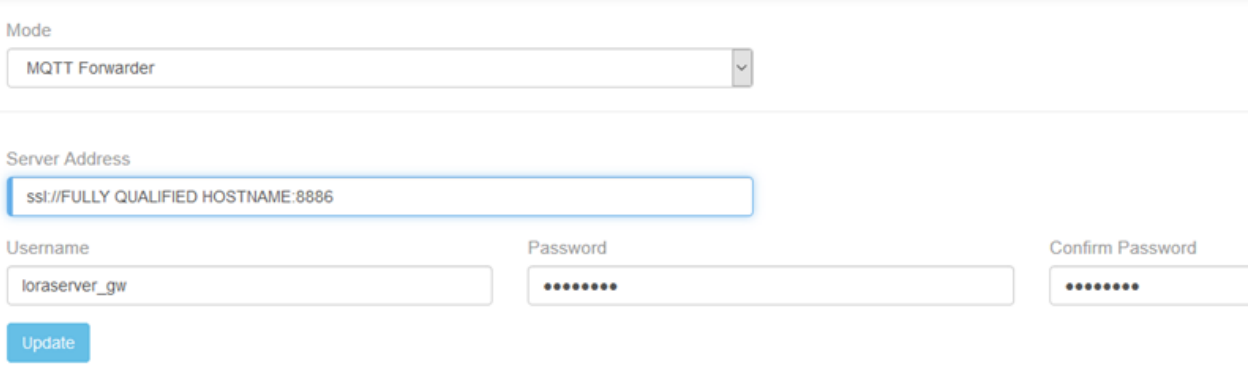
11.3 Configure the RG1xx to use the MQTT Forwarder

For this step, you'll configure the RG1xx to use the newly configured MQTT forwarder and update it with the certificates required to authenticate.

To do so, you'll need to make updates to the RG1xx configuration. For more information on configuring the RG1xx, see the RG1xx user guide on the RG1xx product page.

To configure the RG1xx, complete the following steps.

1. In the RG1xx configuration dashboard, navigate to **LoRa** > **Forwarder** and select the MQTT forwarder.
2. Fill in the server address as shown below using this format ssl://FULLY QUALIFIED HOSTNAME:8886
3. Assuming the LoRa server credentials are at default, use "loraserver_gw" as both the username and password.
4. Click **Update.**



Note: At this time, the gateway should be reporting an error that it can't verify the server certificate, since it is self-signed.

Open the RG1xx configuration dashboard. Open the logging window at the bottom of the page, and click **Auto Update Logs.** The gateway shows an error similar to the following:

```
RG1xx29354F   lora   user.notice   May 2 12:41:14   time="2018-05-02T12:41:32Z"
level=error msg="could not setup mqtt backend, retry in 2 seconds: Network Error :
x509: certificate signed by unknown authority"
RG1xx29354F   lora   user.notice   May 2 12:41:14   time="2018-05-02T12:41:32Z"
level=info msg="backend: connecting to mqtt broker"
server="ssl://usbrocaar.duckdns.org:8886"
RG1xx29354F   lora   user.notice   May 2 12:41:14   time="2018-05-02T12:41:30Z"
level=error msg="could not setup mqtt backend, retry in 2 seconds: Network Error :
x509: certificate signed by unknown authority"
RG1xx29354F   lora   user.notice   May 2 12:41:14   time="2018-05-02T12:41:30Z"
level=info msg="backend: connecting to mqtt broker"
server="ssl://usbrocaar.duckdns.org:8886"
```

Using scp, copy the ca.crt file to the machine being used to configure the Gateway. It is recommended to name it distinctively. Execute this command on the installer machine:

```
scp AWSInstance:ca.crt newserver_ca.crt
```

You now have a local copy of the CA certificate you created. You'll need to upload this to the gateway.

In the RG1xx dashboard, navigate to **LoRa > Advanced** and then upload the newly downloaded CA certificate.

The MQTT Forwarder needs to be restarted to use the newly uploaded certificate. To restart it, complete the following:

1. On the Gateway, Navigate to **LoRa > Forwarder.**
2. Click **Update** to restart the MQTT Secure Forwarder

At this point, the Gateway should successfully connect to the LoRa Server.

Start the LoRa client and verify that the client can join and pass traffic.