Quick Install

This tutorial describes the steps needed to setup the LoRa Server project **including all requirements** on a single Ubuntu 16.04 LTS instance. Note that this version of Ubuntu is not required by LoRa Server, but is used in this tutorial as it is the latest Ubuntu LTS version. Please refer to the other install pages for more generic installation instructions.

Assumptions

Many configurations of these packages is possible. Dependent software packages could be installed on any number of remote servers, and the packages themselves could be on their own servers. However, in order to simplify the initial installation, we will assume the following deployment architecture:

- All LoRa Server components and their dependencies will be installed on a single server instance
- The LoRa Gateway Bridge (../../lora-gateway-bridge/) component will be installed on the server, but can also be installed on the gateway itself.
- SSL/TLS certificates will be needed for mosquitto and lora-app-server for these instructions, and potentially for other interfaces should the installation be varied from these instructions. While self-signed certificates are possible, it is much easier in the long run to get signed certificates. These details are left to the installer of this software.

Of course, optimizations may need to be made depending on the performance of your systems. You may opt to move the PostgreSQL database to another server. Or you may decide to put your MQTT server on a different system, or even use a different server than the one recommended in this document. These and other installation changes are beyond the scope of this document. However, you should be able to find the information here that would make these changes relatively straight-forward.

Install requirements

Software dependencies for the LoRa Server compoments:

- **MQTT broker** - A publish/subscribe protocol that allows users to publish information under topics that others can subscribe to. A popular implementation of the MQTT protocol is Mosquitto (https://mosquitto.org/).
- **Redis** - A database used to store relatively transient data.
- **PostgreSQL** - The long-term storage database used by the open source packages.

Use the package manager apt to install these dependencies on the Ubuntu 16.04 LTS server:

```
sudo apt install mosquitto mosquitto-clients redis-server redis-tools postgresql
```

Mosquitto authentication

Mosquitto, as the main conduit for messaging between the gateways and the LoRa servers and the applications receiving LoRa data, should be secured to prevent third party access to the data. To set up Mosquitto security:

```
# Create a password file for your mosquitto users, starting with a “root” user.
# The “-c” parameter creates the new password file. The command will prompt for
# a new password for the user.
sudo mosquitto_passwd -c /etc/mosquitto/pwd loraroot

# Add users for the various MQTT protocol users
sudo mosquitto_passwd /etc/mosquitto/pwd loragw
sudo mosquitto_passwd /etc/mosquitto/pwd loraserver
sudo mosquitto_passwd /etc/mosquitto/pwd loraappserver

# Secure the password file
sudo chmod 600 /etc/mosquitto/pwd
```

Note that further configuration is possible, such as limiting the topics to which the various users can have access. These settings are beyond the scope of this document.

Mosquitto configuration

Add a new local configuration file (this should survive mosquitto upgrades) called `/etc/mosquitto/conf.d/local.conf` with the following configuration:

Tell mosquitto where the password file is by adding the lines:

```
allow_anonymous false
password_file /etc/mosquitto/pwd
```

If you set up SSL/TLS certificates for your server (recommended for production environments) add lines like these pointing to the respective files. Using SSL/TLS is a good idea so that passwords cannot be read as they are sent to Mosquitto for login:

```
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/hostname.crt
keyfile /etc/mosquitto/certs/hostname.key
```

After saving this configuration, restart Mosquitto with the new settings:

```
sudo systemctl restart mosquitto
```

PostgreSQL databases and users

To enter the command line utility for PostgreSQL:

```
sudo -u postgres psql
```

Inside this prompt, execute the following queries to set up the server. It would be wise to change the usernames and passwords. Just remember to use these other values when setting up the configuration for loraserver and lora-app-server. Since these two applications both use the same table to track database upgrades, they must have separate databases.

```
-- set up the users and the passwords (note that it is important to use single quotes and a semicol
create role loraserver_as with login password 'dbpassword';
create role loraserver_ns with login password 'dbpassword';

-- create the database for the servers
create database loraserver_as with owner loraserver_as;
create database loraserver_ns with owner loraserver_ns;

-- exit psql
\q
```

Setup LoRa Server software repository

LoRa Server provides a repository that is compatible with the Ubuntu apt package system.

Set up the key for this new repository:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
```

Add the repository to the repository list by creating a new file:

```
sudo echo "deb https://repos.loraserver.io/ubuntu xenial testing" | sudo tee /etc/apt/sources.list.d
```

Update the apt system:

```
sudo apt update
```

Installing LoRa Gateway Bridge

Note: when you intent to run the LoRa Gateway Bridge (../../lora-gateway-bridge/) only on the gateways itself, you can skip this step. Running LoRa Gateway Brige on your server, without setting up any firewall rules (which is not covered in this guide) allows anybody to send data to your LoRa Server network.

Install the package using apt:

```
sudo apt install lora-gateway-bridge
```

Set up your configuration (only the most important settings are addressed here) by changing the configuration file `/etc/lora-gateway-bridge/lora-gateway-bridge.toml`:

- `packet_forwarder.udp_bind` - The interface and port on which the LoRa Gateway Bridge listens for packets from your gateways. This should be `0.0.0.0:1700` to allow access on all network interfaces.
- `backend.mqtt.username` and `backend.mqtt.password` - since the MQTT server is publicly accessible (so LoRa Gateway Bridge (../../lora-gateway-bridge/) instances can send data), it is best to have a username and password for the server here. These credentials were the ones set up in the *Mosquitto authentication* step.

Installing LoRa Server

Install the package using apt:

```
sudo apt install loraserver
```

Set up your configuration (only the most important settings are addressed here) by changing the configuration file `/etc/loraserver/loraserver.toml`:

- `network_server.band.name` - The ISM band to use. E.g. for US installations, use `US_902_928`.
- `postgresql.dsn` - The URL to the postgres server. Add `username:password@` to the URL. e.g., `postgres://loraserver_ns:dbpassword@localhost/loraserver?sslmode=disable`. Be careful with this setting. It is easy to get wrong, and can produce a number of different error messages.
- `postgresql.automigrate` - Leave this set to true, as it only takes a moment to run at server startup, and ensures that database changes will always be applied with each upgrade.
- `network_server.gateway.backend.mqtt.username` and `network_server.gateway.backend.mqtt.password` - since the MQTT server is publicly accessible (so LoRa Gateway Bridge (../../lora-gateway-bridge/) instances can send data), it is best to have a username and password for the server here. These credentials were the ones set up in the *Mosquitto authentication* step.
- `network_server.gateway.stats.create_gateway_on_stats` - creates a gateway record for the gateway when stats are seen. Otherwise gateway data must be populated manually using the LoRa App Server (../../lora-app-server/) UI or via the gRPC interface to LoRa Server (../../loraserver/).
- `network_server.gateway.stats.aggregation_intervals` - defines collection time periods for statistics gathering on gateways.
- `network_server.gateway.api.jwt_secret` - a secret value used to generate the gateway tokens. This can contain any value, for example the output of the following command `openssl rand -base64 32`.

Start the LoRa Server service:

```
sudo systemctl start loraserver
```

Logging for loraserver is accessible via (add `-f` to *follow*):

```
journalctl -u loraserver
```

Note that you may see errors at this point along the lines of:

```
INFO[0001] grpc: addrConn.resetTransport failed to create client transport: connection error: desc =
```

This indicates that LoRa App Server (../../lora-app-server/) is not yet running. LoRa Server (../../loraserver/) is trying to communicate with LoRa App Server via the gRPC api. Once LoRa App Server is running, this error should stop.

Installing LoRa App Server

Install the package using apt:

```
sudo apt install lora-app-server
```

Set up your configuration (only the most important settings are addressed here) by changing the configuration file `/etc/lora-app-server/lora-app-server.toml` :

- `postgresql.dsn` - The URL to the postgres server. Add `username:password@` to the URL. e.g., `postgres://loraserver_as:dbpassword@localhost/loraappserver?sslmode=disable` . Be careful with this setting. It is easy to get wrong, and can produce a number of different error messages.
- `postgresql.automigrate` - Leave this set to true, as it only takes a moment to run at server startup, and ensures that database changes will always be applied with each upgrade.
- `application_server.integration.mqtt.username` and `application_server.integration.mqtt.password` - since the MQTT server is publicly accessible (so LoRa Gateway Bridge (../../lora-gateway-bridge/) instances can send data), it is best to have a username and password for the server here. These credentials were the ones set up in the *Mosquitto authentication* step.
- `application_server.api.bind` - The port that serves up the api server. This should be `localhost:8001` as LoRa Server (../../loraserver/) is on the same system.
- `application_server.external_api.bind` - The port that serves up the public api server used by the web UI. This is usually `0.0.0.0:8080` to enable access from all network interfaces, but can be limited to a specific interface if desired.
- `application_server.external_api.tls_cert` and `application_server.external_api.tls_key` - These settings point to the certificate and key files and support SSL on the web ui REST interface and the public gRPC interface. Since users log into the system via these interfaces from remote systems, these settings are very important. Note that default files are shipped with the software package, but they should be replaced for security.
- `application_server.external_api.jwt_serect` - a secret value used to generate the JWT returned as part of the login process, and is used again to verify the validity of the token. This can be a classic password, or it could

be a generated value such as one generated by the command `openssl rand -base64 32`.

- `general.password_hash_iterations` - The number of iterations to use to generate a password hash. The goal is to have enough iterations that generation takes a second (and so verification takes a second) making brute force login attacks painful to the attacker. The default is a good place to start, but finding a good value for the server will take some trial and error.

Start the LoRa App Server service:

```
sudo systemctl start lora-app-server
```

Logging for LoRa App Server is accessible via (add `-f` to *follow*):

```
journalctl -u lora-app-server
```

Install LoRa Gateway Bridge on the gateway

It is advised to run LoRa Gateway Bridge on each gateway itself, to enable a secure connection between your gateways and your server.

External dependencies:

- The LoRa gateway (through the packet-forwarder (https://github.com/lora-net/packet_forwarder) software) communicates with the LoRa Gateway Bridge (../../lora-gateway-bridge/) via UDP.
- LoRa Gateway Bridge (../../lora-gateway-brige/) must have access to the MQTT broker through which it will communicate with LoRa Server (../../loraserver/). This server should be secured with SSL and passwords as previously mentioned. Typical installation would put this MQTT broker on the same server as LoRa Server and LoRa App Server.

As there are many types of gateways available, please refer to the LoRa Gateway Bridge instructions for installing LoRa Gateway Bridge on the server (../../lora-gateway-bridge/install/gateway/).

Setting up an initial application

In order to receive data from a device / node, that device must be associated with an application. This can be done via the REST interface via some external application, or via the UI that comes with LoRa App Server (../../lora-app-servr/).

To access the LoRa App Server web-interface, enter the IP address of your server and the port as defined in `application_server.external_api.bind` in your browser specifying the https protocol in your browser's address bar, example: `https://localhost:8080`.

This will forward to a login screen. A default administrator account is available at installation time with the username of `admin` and a password of `admin` (it would be wise to reset the password upon first login).

In order to connect you LoRa App Server instance with the LoRa Server instance, click *Network servers* and then *Add network-server*. As LoRa Server is installed on the same host as LoRa App Server, use `localhost:8000` as network-server name.

After adding the network-server, click on *LoRa Server* to go back to the home screen. Then open the *Service profiles* tab and click *Create service-profile* to create a service-profile for the LoRa Server organization. This will also associate the organization with the network-server instance.

After creating the service-profile, go to the *Device profiles* tab and click *Create device-profile*. Here you can define the device properties and capabilities (e.g. OTAA vs ABP).

After connecting LoRa App Server with the LoRa Server network-server, creating a service- and device-profile it is time to create your first application and add a device!

Go to the *Applications* tab and click the *CREATE APPLICATION* button, and add an application for your device(s). This only requires an application name and description. Once the application is created, you can click on the title to get a list of the devices associated with the application (none at system installation). Click on the *CREATE DEVICE* button to create the node (device). The basic fields that are required are the *Device name*, *Description*, *Device EUI* and the *Device-profile*. After creating the device, you will be redirected to a page to enter the *AppKey* in case of an OTAA device or to activate the device in case of an ABP device.

Once the device and it application are created, the LoRa Server and LoRa App Server will be able to handle messaging from the device.

Conclusion

At this point you should be able to follow the logs and see no errors. Also, you can user the loraroot account on Mosquitto to watch the message flow:

```
mosquitto_sub -v -t "#" -u loraroot -P {password} -h localhost -p 1883
```

Where:

- `-v` - verbose output - includes the *topic* of the message
- `-t "#"` - any message. `"#"` is a multi-level wildcard. Other possibilities include:
  - `"gateway/#"` - any gateway messages
  - `"application/#"` - any application messages
- `-u` - The user to log into mosquitto with
- `-P` - The password for the user
- `-h` - The host to log in to
- `-p` - The mosquitto port