How to Install and Secure the Mosquitto MQTT Messaging Broker on Debian 10

Introduction

MQTT is a machine-to-machine messaging protocol, designed to provide lightweight publish/subscribe communication to "Internet of Things" devices. It is commonly used for geo-tracking fleets of vehicles, home automation, environmental sensor networks, and utility-scale data collection.

Mosquitto is a popular MQTT server (or *broker*, in MQTT parlance) that has great community support and is easy to install and configure.

In this tutorial, we'll install Mosquitto and set up our broker to use SSL to secure our password-protected MQTT communications.

Prerequisites

Before starting this tutorial, you will need:

- A Debian 10 server with a non-root, sudo-enabled user and basic firewall set up, as detailed in this Debian 10 server setup tutorial.
- A domain name pointed at your server, as documented in our DigitalOcean DNS product documentation. This tutorial will use `mqtt.example.com` throughout.
- An auto-renewable Let's Encrypt SSL certificate for use with your domain and Mosquitto, generated using the Certbot tool. You can learn how to set this up in How To Use Certbot Standalone Mode to Retrieve Let's Encrypt SSL Certificates on Debian 10
  . You can add `systemctl restart mosquitto` as a `renew_hook` in Step 4. Be sure to use the same domain configured in the previous prerequisite step.

Step 1 — Installing Mosquitto

Debian 10 has a fairly recent version of Mosquitto in its default software repository, so we can install it from there.

First, log in using your non-root user and update the package lists using `apt update`:

```
sudo apt update
```

Now, install Mosquitto using `apt install`:

```
sudo apt install mosquitto mosquitto-clients
```

By default, Debian will start the Mosquitto service after install. Let's test the default configuration. We'll use one of the Mosquitto clients we just installed to subscribe to a topic on our broker.

Topics are labels that you publish messages to and subscribe to. They are arranged as a hierarchy, so you could have `sensors/outside/temp` and `sensors/outside/humidity`, for example. How you arrange topics is up to you and your needs. Throughout this tutorial we will use a simple test topic to test our configuration changes.

Log in to your server a second time, so you have two terminals side-by-side. In the new terminal, use `mosquitto_sub` to subscribe to the test topic:

```
mosquitto_sub -h localhost -t test
```

`-h` is used to specify the hostname of the MQTT server, and `-t` is the topic name. You'll see no output after hitting `ENTER` because `mosquitto_sub` is waiting for messages to arrive. Switch back to your other terminal and publish a message:

```
mosquitto_pub -h localhost -t test -m "hello world"
```

The options for `mosquitto_pub` are the same as `mosquitto_sub`, though this time we use the additional `-m` option to specify our message. Hit `ENTER`, and you should see **hello world** pop up in the other terminal. You've sent your first MQTT message!

Enter `CTRL+C` in the second terminal to exit out of `mosquitto_sub`, but keep the connection to the server open. We'll use it again for another test in Step 5.

Next, we'll secure our installation using password-based authentication.

Step 2 — Configuring MQTT Passwords

Let's configure Mosquitto to use passwords. Mosquitto includes a utility to generate a special password file called `mosquitto_passwd`. This command will prompt you to enter a password for the specified username, and place the results in `/etc/mosquitto/passwd`.

- `sudo mosquitto_passwd -c /etc/mosquitto/passwd sammy`

Now we'll open up a new configuration file for Mosquitto and tell it to use this password file to require logins for all connections:

- `sudo nano /etc/mosquitto/conf.d/default.conf`

This should open an empty file. Paste in the following:

/etc/mosquitto/conf.d/default.conf

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Be sure to leave a trailing newline at the end of the file.

`allow_anonymous false` will disable all non-authenticated connections, and the `password_file` line tells Mosquitto where to look for user and password information. Save and exit the file.

Now we need to restart Mosquitto and test our changes.

- `sudo systemctl restart mosquitto`

Try to publish a message without a password:

- `mosquitto_pub -h localhost -t "test" -m "hello world"`

The message should be rejected:

```
Output
Connection Refused: not authorised.
Error: The connection was refused.
```

Before we try again with the password, switch to your second terminal window again, and subscribe to the 'test' topic, using the username and password this time:

```
mosquitto_sub -h localhost -t test -u "sammy" -P "password"
```

It should connect and sit, waiting for messages. You can leave this terminal open and connected for the rest of the tutorial, as we'll periodically send it test messages.

Now publish a message with your other terminal, again using the username and password:

```
mosquitto_pub -h localhost -t "test" -m "hello world" -u "sammy" -P
"password"
```

The message should go through as in Step 1. We've successfully added password protection to Mosquitto.