Running a Self-Hosted MQTT Broker

Eclipse **Mosquitto** is the de-facto standard open-source MQTT broker, widely used with Meshtastic for private mesh deployments. This guide installs Mosquitto on a Raspberry Pi or Ubuntu/Debian server, secures it with TLS, adds authentication, and optionally bridges it to the public `mqtt.meshtastic.org` server.

Prerequisites

- A Linux host (Raspberry Pi OS, Ubuntu 22.04, Debian 12) with a static LAN IP or a public IP / DDNS hostname.
- A domain name or static IP for the TLS certificate (required for Let's Encrypt; a self-signed cert works for LAN-only deployments).
- Ports 1883 (MQTT) and/or 8883 (MQTT over TLS) open on the host firewall and your router NAT if internet-accessible.

Step 1 - Install Mosquitto

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients


# Confirm the service is running
sudo systemctl status mosquitto
```

On Mosquitto 2.0+, a fresh install only accepts connections from the local machine and anonymous access is disabled by default (`allow_anonymous false`) - it will **not** accept remote clients on 1883 until you define a listener and configure authentication. (Older 1.x versions did listen on 1883 with anonymous access, which is the source of the common misconception.) In all cases, explicitly configure a listener, authentication, and TLS before any internet exposure.

Step 2 - Minimal Configuration for Meshtastic

Mosquitto's main config is at `/etc/mosquitto/mosquitto.conf`. The `conf.d/` directory is for drop-in configs. Because a 2.0+ install binds localhost-only with no listener, you **must** define a listener bound to `0.0.0.0` (plus authentication) for other hosts - including your Meshtastic nodes - to connect. Create a Meshtastic-specific config:

```
sudo nano /etc/mosquitto/conf.d/meshtastic.conf
```

```
# /etc/mosquitto/conf.d/meshtastic.conf

# ── Plain-text listener (LAN only or behind VPN) ──────────────────
# Required: without an explicit listener, Mosquitto 2.0 binds localhost only
listener 1883 0.0.0.0
protocol mqtt

# Require authentication on all listeners
allow_anonymous false
password_file /etc/mosquitto/passwd

# ── TLS listener ──────────────────────────────────────────────────
listener 8883 0.0.0.0
protocol mqtt
tls_version tlsv1.2

# For a Let's Encrypt SERVER cert, point certfile at fullchain.pem and
# keyfile at privkey.pem. Do NOT set cafile here unless you are requiring
# CLIENT certificate auth - cafile verifies clients, not the server chain.
certfile /etc/letsencrypt/live/mqtt.example.com/fullchain.pem
keyfile /etc/letsencrypt/live/mqtt.example.com/privkey.pem

# ── Persistence ───────────────────────────────────────────────────
persistence true
persistence_location /var/lib/mosquitto/

# ── Logging ───────────────────────────────────────────────────────
```

```
log_dest file /var/log/mosquitto/mosquitto.log
log_type all
```

Step 3 - Add User Authentication

```
# Create the password file and add a user
sudo mosquitto_passwd -c /etc/mosquitto/passwd meshuser
# Enter password at prompt (choose a strong password)

# Add additional users (omit -c flag to append rather than recreate)
sudo mosquitto_passwd /etc/mosquitto/passwd seconduser
```

Step 4 - TLS Certificate (Let's Encrypt)

If your broker has a public domain name:

```
sudo apt install -y certbot
sudo certbot certonly --standalone -d mqtt.example.com --agree-tos --email admin@example.com

# Certbot stores certs in /etc/letsencrypt/live/mqtt.example.com/
# Mosquitto must be able to read the key
sudo usermod -aG ssl-cert mosquitto # Debian; or adjust file perms directly
```

For LAN-only deployments without a public domain, generate a self-signed certificate:

```
sudo mkdir -p /etc/mosquitto/tls
cd /etc/mosquitto/tls

sudo openssl req -x509 -newkey rsa:4096 -keyout server.key -out server.crt -days 3650 -nodes -
subj "/CN=mqtt.local/O=MeshAmerica/C=US"

sudo chown mosquitto:mosquitto server.key server.crt
sudo chmod 600 server.key
```

```
# Update meshtastic.conf to use these paths:
# certfile /etc/mosquitto/tls/server.crt
# keyfile /etc/mosquitto/tls/server.key
```

Meshtastic's MQTT module exposes only a *TLS Enabled* toggle - there is **no** in-app/firmware option named "TLS Verify Cert" to turn certificate verification off. When TLS is enabled the node validates the server certificate, so a self-signed certificate generally requires the node to trust the issuing CA. The simplest reliable path is to use a publicly-trusted certificate from Let's Encrypt rather than a self-signed cert. Self-signed CA-import support varies by firmware version, so verify current behavior on your build.

Step 5 - Open Firewall Ports

```
# Using ufw
sudo ufw allow 1883/tcp comment "MQTT plain"
sudo ufw allow 8883/tcp comment "MQTT TLS"
sudo ufw reload
sudo ufw status
```

Step 6 - Reload and Test

```
sudo systemctl reload mosquitto
sudo systemctl status mosquitto

# Test from local machine (no TLS)
mosquitto_sub -h localhost -p 1883 -u meshuser -P yourpassword -t "msh/#" -v &
mosquitto_pub -h localhost -p 1883 -u meshuser -P yourpassword -t "msh/test" -m "hello"

# Test TLS from a remote machine (using cert bundle)
mosquitto_sub -h mqtt.example.com -p 8883 --cafile /etc/ssl/certs/ca-certificates.crt -u
meshuser -P yourpassword -t "msh/#" -v
```

Step 7 - Bridging to mqtt.meshtastic.org (Optional)

To merge your private broker's traffic with the public network, configure a bridge. Note that the public `mqtt.meshtastic.org` broker is **not anonymous** - it requires the shared credentials `meshdev` / `large4cats` - and it now **restricts broad topic subscriptions** (you can no longer subscribe to the whole `#` tree). Review Meshtastic's public-broker policy before bridging, and prefer narrow, channel-specific topics rather than a wide `msh/REGION/#` firehose.

```
sudo nano /etc/mosquitto/conf.d/bridge_meshtastic.conf
```

```
# /etc/mosquitto/conf.d/bridge_meshtastic.conf

connection meshtastic_public
address mqtt.meshtastic.org:1883

# Bridge a narrow, channel-specific topic (the public broker now blocks
# broad subscriptions like msh/US/#). Adjust to your region/channel.
topic msh/US/2/e/LongFast/# both 0

# Bridge credentials - the public broker is NOT anonymous; these shared
# credentials are required or the bridge connection is refused.
remote_username meshdev
remote_password large4cats

# Reconnect after 30 seconds if connection drops
restart_timeout 30
keepalive_interval 60

# Use a persistent queue so messages don't drop if bridge disconnects
cleansession false
```

```
sudo systemctl reload mosquitto
journalctl -u mosquitto -f | grep bridge
```

Access Control Lists (ACLs)

Restrict which users can publish or subscribe to which topics:

```
sudo nano /etc/mosquitto/conf.d/meshtastic.conf
# Add inside the config file:
acl_file /etc/mosquitto/acls
```

```
sudo nano /etc/mosquitto/acls
```

```
# /etc/mosquitto/acls
# Format: topic [read|write|readwrite] <topic_pattern>
# Rules placed BEFORE the first "user" line apply to anonymous clients.
# Deny rules are evaluated before grants, so never put a blanket
# "topic deny #" inside a user block - it would override that user's
# own grants and lock them out.

# meshuser can read and write all msh topics
user meshuser
topic readwrite msh/#

# readonly user can only subscribe, not publish
user readonly
topic read msh/#
```

To block anonymous clients entirely, the cleanest approach is simply `allow_anonymous false` (set above). If you instead want an explicit ACL deny for anonymous access, place `topic deny #` **before** any `user` line - not inside a user block.

```
sudo systemctl reload mosquitto
```

Keeping Certificates Renewed

```
# Certbot auto-renewal cron is installed at /etc/cron.d/certbot
# After renewal, Mosquitto must reload to pick up new certs:
sudo tee /etc/letsencrypt/renewal-hooks/deploy/mosquitto.sh <<'EOF'
#!/bin/bash
systemctl reload mosquitto
EOF
sudo chmod +x /etc/letsencrypt/renewal-hooks/deploy/mosquitto.sh
```

Revision #3
Created 2026-05-03 05:36:31 UTC by Mesh America Admin
Updated 2026-06-10 05:28:02 UTC by Mesh America Admin