

MQTT auf vorhandenem LXC installieren und konfigurieren

MQTT steht für "Message Queuing Telemetry Transport" und ist ein leichtgewichtiges, offenes Kommunikationsprotokoll, das für die Übertragung von Nachrichten zwischen Geräten in einem Netzwerk entwickelt wurde. Es wurde speziell für M2M (Machine-to-Machine) und IoT (Internet of Things) Anwendungen entworfen, bei denen eine zuverlässige und effiziente Übertragung von Daten mit begrenzten Ressourcen erforderlich ist.

Hier sind einige grundlegende Konzepte und Eigenschaften von MQTT:

1. Publisher/Subscriber-Modell: MQTT basiert auf dem Publisher/Subscriber-Messaging-Modell. Es gibt einen zentralen Vermittler, der als **Broker** bezeichnet wird. Geräte, die Daten senden möchten, sind sogenannte **Publisher**, während Geräte, die Daten empfangen möchten, als **Subscriber** bezeichnet werden. Publisher senden Nachrichten an **bestimmte Themen (Topics)**, und Subscriber abonnieren diese Themen, um Nachrichten zu empfangen, die an sie gerichtet sind.

2. Topics: Topics sind hierarchische Namen oder Kanäle, die verwendet werden, um Nachrichten zu organisieren und zu filtern. Sie können nach Belieben benannt werden und ermöglichen eine flexible Kategorisierung von Nachrichten. Beispielsweise kann ein Thema wie "Sensordaten/Temperatur" verwendet werden, um alle Nachrichten zu abonnieren, die mit der Temperurmessung von Sensoren zusammenhängen.

3. Quality of Service (QoS): MQTT unterstützt verschiedene QoS-Level für die Nachrichtenübertragung. Es gibt drei Ebenen:

- **QoS 0:** "At most once" - Die Nachricht wird einmal gesendet, ohne eine Bestätigung oder Überprüfung der Zustellung. Es besteht die Möglichkeit, dass Nachrichten verloren gehen.
- **QoS 1:** "At least once" - Die Nachricht wird mindestens einmal zugestellt. Es kann jedoch zu Duplikaten kommen.
- **QoS 2:** "Exactly once" - Die Nachricht wird genau einmal zugestellt und Duplikate werden vermieden. Dieses Level erfordert die umfangreichsten Kommunikationsmechanismen.

4. Lightweight: MQTT ist darauf ausgelegt, ressourcenschonend zu sein, sowohl in Bezug auf die Netzwerkbänderbreite als auch auf die Systemressourcen der Geräte. Die Nachrichtenheader sind klein, was die Effizienz bei der Übertragung verbessert. Daher eignet sich MQTT gut für Umgebungen mit begrenzten Ressourcen, wie z.B. eingebettete Systeme oder IoT-Geräte.

5. Zuverlässigkeit: MQTT unterstützt eine zuverlässige Übertragung von Nachrichten, indem es Mechanismen wie die Zustellungsbestätigung (Acknowledgement) und Wiederholungsmechanismen bietet. Dies ermöglicht eine robuste Kommunikation in instabilen Netzwerkumgebungen.

MQTT wird häufig in IoT-Anwendungen eingesetzt, bei denen Sensoren, Aktoren und andere Geräte Daten austauschen müssen. Es bietet eine einfache und effiziente Möglichkeit, Nachrichten zwischen den Geräten zu übertragen und ermöglicht die Skalierbarkeit von IoT-Systemen.

Voraussetzungen

- LXC-Container mit Ubuntu 20.04/22.04/24.04 (getestet mit Ubuntu 24.04)
- Netzwerkzugriff auf den Container
- Root- oder anderer User mit `sudo`-Berechtigungen

Mosquitto wird von der **Eclipse Foundation** entwickelt und unter den freien Lizenzen **EPL/EDL** veröffentlicht.

🔧 Vorbereitung: Mosquitto installieren

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

(Mosquitto-Clients werden nur zum testen auf dem System benötigt.)

```
pdal@mqttmaac1tl5150:~$ sudo apt install -y mosquitto mosquitto-clients
Reading package lists... done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  lib cJSON1 lib dlt2 lib mosquitto1 lib websockets19t64
The following NEW packages will be installed:
  lib cJSON1 lib dlt2 lib mosquitto1 lib websockets19t64 mosquitto mosquitto-client
0 upgraded, 6 newly installed, 0 to remove and 2 not upgraded.
Need to get 688 kB of archives.
After this operation, 2038 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble/universe amd64 lib cJSON1 amd64 1.7
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 lib mosquitto1 amd64
Get:3 http://archive.ubuntu.com/ubuntu noble/universe amd64 lib dlt2 amd64 2.18.
Get:4 http://archive.ubuntu.com/ubuntu noble/universe amd64 lib websockets19t64
Get:5 http://archive.ubuntu.com/ubuntu noble/universe amd64 mosquitto amd64 2.0
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 mosquitto-clients a
Fetched 688 kB in 0s (2803 kB/s)
Selecting previously unselected package lib cJSON1:amd64.
(Reading database ... 17339 files and directories currently installed.)
Preparing to unpack .../0-lib cJSON1_1.7.17-1_amd64.deb ...
Unpacking lib cJSON1:amd64 (1.7.17-1) ...
Selecting previously unselected package lib mosquitto1:amd64.
Preparing to unpack .../1-lib mosquitto1_2.0.18-1build3_amd64.deb ...
```

```
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

```
pdal@mqttmaac1tl5150:~$ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mosquitto
pdal@mqttmaac1tl5150:~$ sudo systemctl start mosquitto
pdal@mqttmaac1tl5150:~$ systemctl status mosquitto.service
* mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
   Active: active (running) since Wed 2025-07-09 13:32:49 CEST; 4min 57s ago
     Docs: man:mosquitto.conf(5)
           man:mosquitto(8)
     Main PID: 6731 (mosquitto)
        Tasks: 1 (limit: 4389)
       Memory: 1.0M (peak: 1.3M)
          CPU: 262ms
        CGroup: /system.slice/mosquitto.service
                  '-6731 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
pdal@mqttmaac1tl5150:~$
```

✍ Anonyme, unverschlüsselte MQTT-Kommunikation

1. Konfigurationsdatei anpassen

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

Inhalt:

```
# Speichert MQTT-Nachrichten dauerhaft für Neustarts
persistence true

persistence_location /var/lib/mosquitto/

listener 1883
allow_anonymous true
```

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

#pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

listener 1883
allow_anonymous true
```

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

```
include_dir /etc/mosquitto/conf.d
```

Diese Konfiguration erlaubt allen Clients den unverschlüsselten Zugang ohne Authentifizierung.

2. Dienst neu starten

```
sudo systemctl restart mosquitto
```

```
pdal@mqttmaacltls150:~$ sudo systemctl restart mosquitto.service
pdal@mqttmaacltls150:~$ █
```

Überprüfung ob der Dienst **enabled** ist. (sorgt für automatisches starten des Dienstes beim booten des Containers)

```
sudo systemctl status mosquitto
```

```
pdal@mqttmaacltls150:~$ sudo systemctl status mosquitto
* mosquitto.service - Mosquitto MQTT Broker
  Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
  Active: active (running) since Wed 2025-07-09 13:46:18 CEST; 2min 10s ago
    Docs: man:mosquitto.conf(5)
          man:mosquitto(8)
   Process: 7012 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 7014 ExecStartPre=/bin/chown mosquitto:mosquitto /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 7015 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 7018 ExecStartPre=/bin/chown mosquitto:mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
 Main PID: 7020 (mosquitto)
   Tasks: 1 (limit: 4389)
  Memory: 1.0M (peak: 1.6M)
    CPU: 148ms
   CGroup: /system.slice/mosquitto.service
           `-7020 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

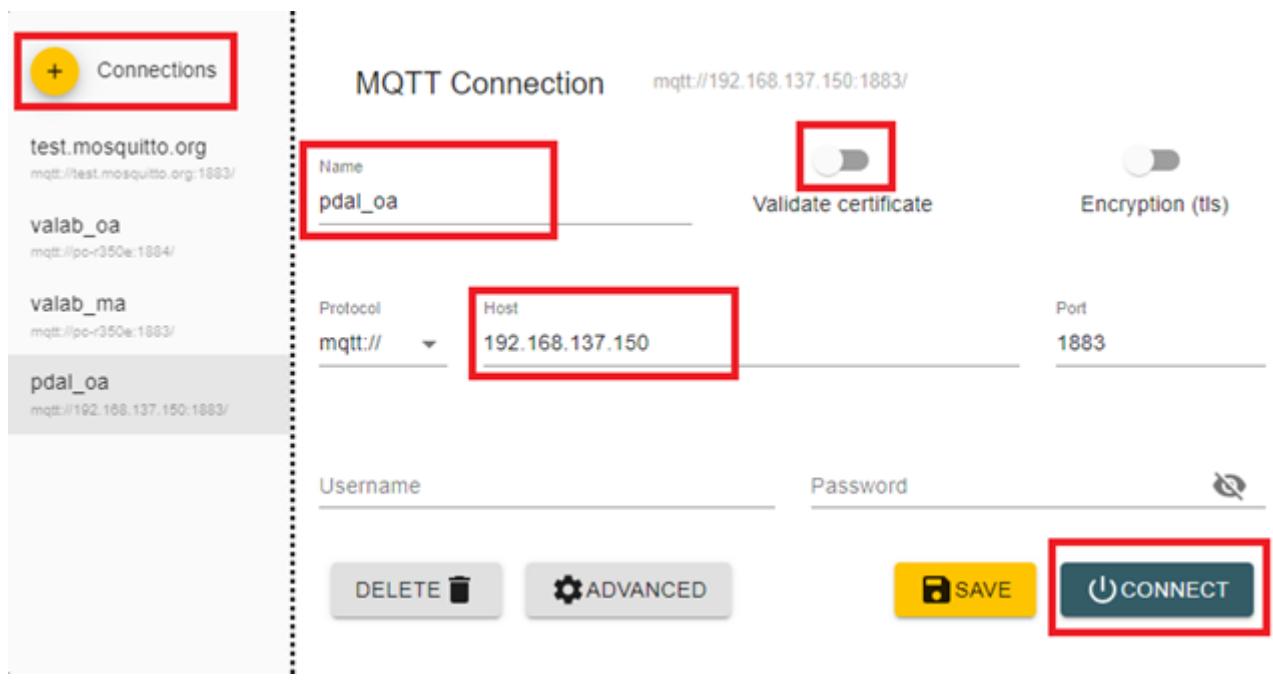
Jul 09 13:46:18 mqttmaacltls150 systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
Jul 09 13:46:18 mqttmaacltls150 systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
```

3. Test auf dem MQTT-Host

Zum testen nutzen wir den MQTT-Explorer.(CC-BY-ND-4.0) Man kann ihn [MQTT-Explorer](#) hier herunterladen. MQTT Explorer ist ein grafisches Desktop-Tool zur Visualisierung, Analyse und Verwaltung von MQTT-Datenströmen. Es dient hauptsächlich dazu, eine Verbindung zu einem MQTT-Broker herzustellen und die gesendeten und empfangenen Nachrichten in einer übersichtlichen Baumstruktur darzustellen. Dabei zeigt es alle Topics, deren Hierarchie sowie die zugehörigen Nachrichteninhalte (Payloads), inklusive Informationen wie QoS-Level, Retain-Status und Zeitstempel.

Mit MQTT Explorer lassen sich Nachrichten nicht nur beobachten, sondern auch aktiv an beliebige Topics senden (Publish-Funktion). Das Tool eignet sich ideal zum Testen, Debuggen und Überwachen von IoT-Geräten, Smart-Home-Systemen oder anderen MQTT-basierten Anwendungen. Es unterstützt dabei auch Sicherheitsfunktionen wie die Verbindung über TLS, die Verwendung von Benutzernamen und Passwörtern sowie die Authentifizierung per Zertifikat.

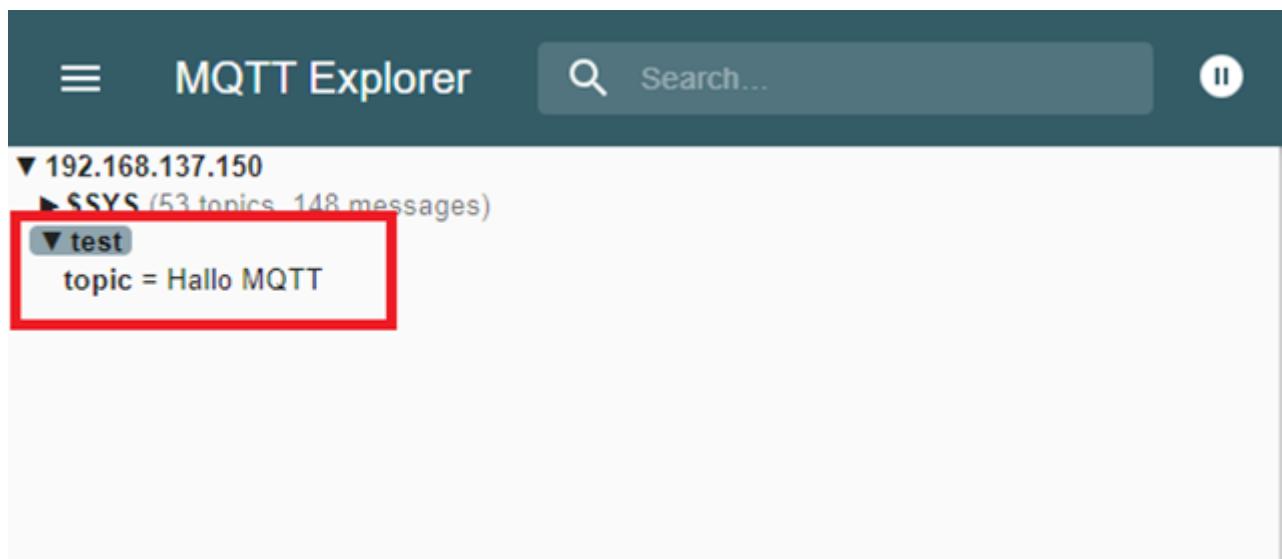
Durch seine benutzerfreundliche Oberfläche und die Echtzeit-Darstellung ist MQTT Explorer besonders hilfreich, wenn man schnell einen Überblick über den Zustand eines MQTT-Systems erhalten oder Fehlerquellen identifizieren möchte. Es ist plattformübergreifend verfügbar für Windows, macOS und Linux.



Publisher Testen mit Mqtt-Client:

```
mosquitto_pub -h 192.168.137.150 -t test/topic -m "Hallo MQTT"
```

```
pdal@mqttmaacltl1s150:~$ mosquitto_pub -h 192.168.137.150 -t test/topic -m "Hallo MQTT"
pdal@mqttmaacltl1s150:~$ █
```



Das Ergebnis ist im MqttExplorer sofort zu sehen.

Subscriber

Test mit Mqtt-Client:

```
mosquitto_sub -h 192.168.137.150 -t test/topic
```

Senden einer Nachricht über MqttExplorer.

```
pdal@mqttmaac1tls150:~$ mosquitto_sub -h 192.168.137.150 -t test/topic
Hello zurück
```

Für einen Langzeittest können Sie zwei weitere LXC-Container erstellt. Einen Container nutzen wir als Publisher, und den anderen nutzen wir als Subscriber. Eine genaue Anleitung hierzu findest du in dieser Dokumentation. [[0755 MqttClients]]

MQTT mit Benutzeranmeldung, unverschlüsselt

Zur besseren Zugriffskontrolle werden noch User mit Passwort eingerichtet.

1. Passworddatei erstellen

```
sudo mkdir -p /etc/mosquitto/passwords
sudo mosquitto_passwd -c /etc/mosquitto/passwords/mqtt_users pdal
```

```
pdal@mqttmaac1tls150:~$ sudo mkdir -p /etc/mosquitto/passwords
[sudo] password for pdal:
pdal@mqttmaac1tls150:~$ sudo mosquitto_passwd -c /etc/mosquitto/passwords/mqtt_users pdal
Password:
Reenter password:
pdal@mqttmaac1tls150:~$
```

Jetzt wird nach einem Passwort gefragt, welches für den User **pdal** gesetzt werden muss. Für dieses Beispiel wurde das Passwort **JadeHS20** gewählt.

Nach dem Erstellen der Datei **mqtt_users**, müssen hierfür ggf. die Berechtigungen angepasst werden.

```
ls -l /etc/mosquitto/passwords/mqtt_users
```

Die Berechtigungen müssen wie folgt gesetzt sein damit der Mosquitto Dienst einwandfrei funktioniert.

```
sudo chown root:root /etc/mosquitto/passwords/mqtt_users
sudo chmod 644 /etc/mosquitto/passwords/mqtt_users
```

```
pdal@mqttmaac1tl1s150:~$ ls -l /etc/mosquitto/passwords/mqtt_users
-rw----- 1 root root 118 Jul 10 11:57 /etc/mosquitto/passwords/mqtt_users
pdal@mqttmaac1tl1s150:~$ sudo chown root:root /etc/mosquitto/passwords/mqtt_users
sudo chmod 644 /etc/mosquitto/passwords/mqtt_users
[sudo] password for pdal:
pdal@mqttmaac1tl1s150:~$ █
```

1. **sudo chown root:root /etc/mosquitto/passwords/mqtt_users**

Der Befehl setzt den Besitzer und die Gruppe der Datei **mqtt_users** auf **root**, sodass nur der Systemadministrator (root) vollen Zugriff darauf hat.

2. **sudo chmod 644 /etc/mosquitto/passwords/mqtt_users**

Der Befehl erlaubt dem Besitzer der Datei, sie zu lesen und zu schreiben, während Gruppe und andere Benutzer die Datei nur lesen dürfen. Nachdem die Berechtigungen und Owner geändert wurden, überprüfen Sie mit dem nachfolgenden Befehl, ob diese auch wirklich angepasst wurden.

```
ls -l /etc/mosquitto/passwords/mqtt_users
```

```
pdal@mqttmaac1tl1s150:~$ ls -l /etc/mosquitto/passwords/mqtt_users
-rw-r--r-- 1 root root 118 Jul 10 11:57 /etc/mosquitto/passwords/mqtt_users
pdal@mqttmaac1tl1s150:~$ █
```

Weitere User können so erstellt werden.

```
sudo mosquitto_passwd /etc/mosquitto/passwords/mqtt_users Kai
```

Jetzt wird nach einem Passwort gefragt, welches für den User **Kai** gesetzt werden muss. Für dieses Beispiel wurde das Passwort **1234** gewählt.

```
pdal@mqttmaaclts150:~$ sudo mosquitto_passwd /etc/mosquitto/passwords/mqtt_users Kai
[sudo] password for pdal:
Password:
Reenter password:
pdal@mqttmaaclts150:~$
```

Es ist möglich, dass an dieser Stelle eine Warnung erscheint, weil die Berechtigung für die Datei `mqtt_users` auf **644** gesetzt wurde. Diese Warnung besagt: **Warnung: Die Datei `/etc/mosquitto/passwords/mqtt_users` ist für alle Benutzer lesbar. Zukünftige Versionen werden das Laden dieser Datei verweigern.** Hier kann man mit Capabilities(Fähigkeiten) arbeiten, damit der "Dienst_user" `mosquitto` dennoch Leserechte für die Datei `mqtt_users` erhält, obwohl die Berechtigung auf `root:root` und **600** gesetzt ist.

🔗 Nützliche Links zu Linux Capabilities

- [Linux Capabilities – man7.org \(offizielle Doku\)](#)
- [Einführung in Linux Capabilities – linuxconfig.org](#)
- [setcap und getcap erklärt – commandmasters.com](#)
- [Capabilities vs Root – insecure.ws](#)
- [Linux Capabilities verständlich erklärt – baeldung.com](#)

2. Konfiguration aktualisieren

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

```
#pid_file /run/mosquitto/mosquitto.pid
# Speichert MQTT-Nachrichten dauerhaft für Neustarts
persistence true

# Speicherort für Persistenzdaten wie mosquitto.db
persistence_location /var/lib/mosquitto/

# Startet Mosquitto auf Port 1883 für eingehende Verbindungen
listener 1883

# Verbietet Verbindungen ohne Benutzernamen/Passwort
allow_anonymous false
```

```
# Speichert MQTT-Nachrichten dauerhaft für Neustarts
persistence true

# Speicherort für Persistenzdaten wie mosquitto.db
persistence_location /var/lib/mosquitto/

# Startet Mosquitto auf Port 1883 für eingehende Verbindungen
listener 1883

# Verbietet Verbindungen ohne Benutzernamen/Passwort
allow_anonymous false
```

```
# Pfad zur Passwortdatei für die Benutzer-Authentifizierung
password_file /etc/mosquitto/passwords/mqtt_users
```

```
#pid_file /run/mosquitto/mosquitto.pid
# Speichert MQTT-Nachrichten dauerhaft für Neustarts
persistence true

# Speicherort für Persistenzdaten wie mosquitto.db
persistence_location /var/lib/mosquitto/

# Startet Mosquitto auf Port 1883 für eingehende Verbindungen
listener 1883

# Verbietet Verbindungen ohne Benutzernamen/Passwort
allow_anonymous false

# Pfad zur Passwortdatei für die Benutzer-Authentifizierung
password_file /etc/mosquitto/passwords/mqtt_users
```

Nun ist es nicht mehr möglich sich als anonymer User anzumelden.

An dieser Stelle (`listener 1883`) könnten Sie auch den Standard-Port für den MQTT-Broker ändern; z. B. wenn Sie mehrere MQTT-Broker benötigen.

3. Dienst neu starten und Status abfragen

```
sudo systemctl restart mosquitto
sudo systemctl status mosquitto
```

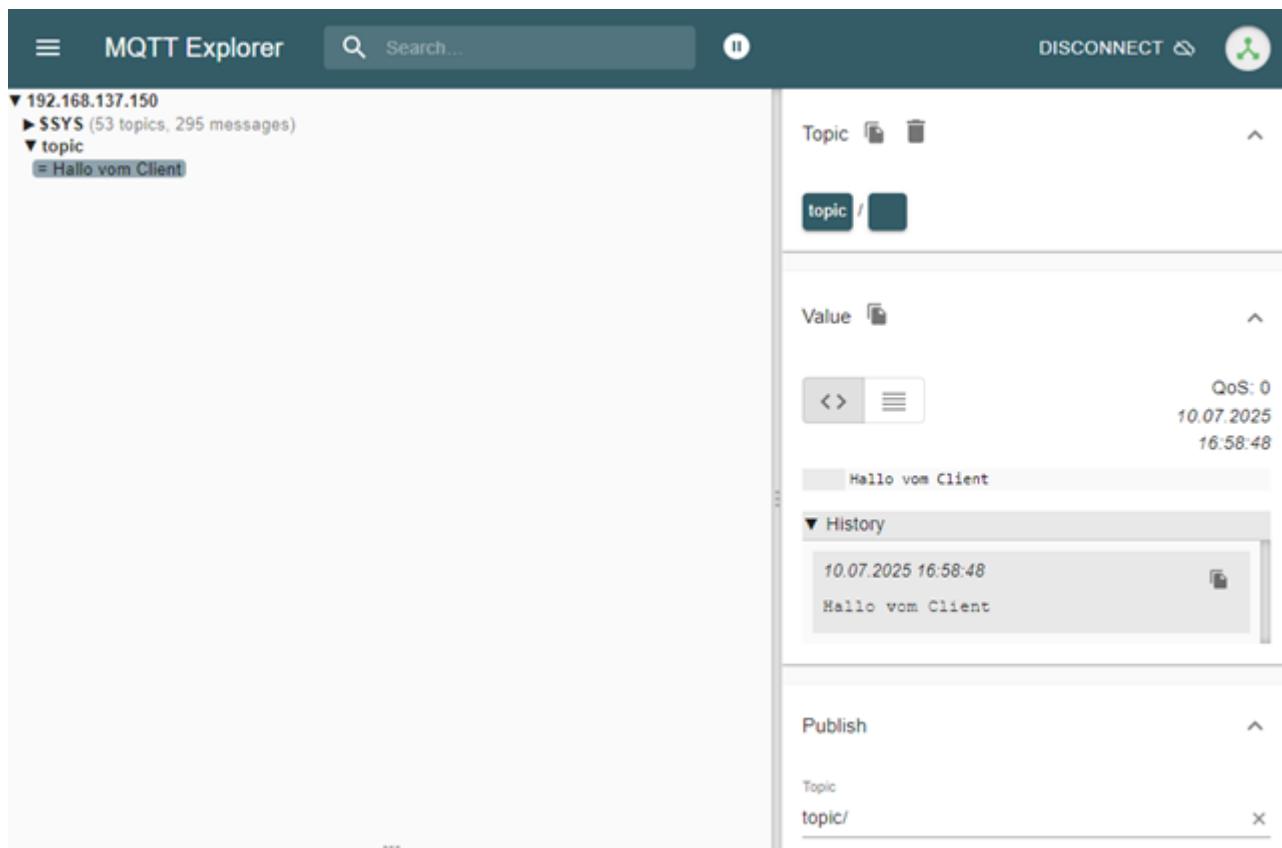
```
pdal@mqttmaacltls150:~$ sudo systemctl restart mosquitto
sudo systemctl status mosquitto
* mosquitto.service - Mosquitto MQTT Broker
  Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
  Active: active (running) since Thu 2025-07-10 10:27:14 CEST; 50ms ago
    Docs: man:mosquitto.conf(5)
          man:mosquitto(8)
   Process: 673 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 674 ExecStartPre=/bin/chown mosquitto:mosquitto /var/log/mosquitto (code=exited, status=0/SUCCESS)
   Process: 677 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, status=0/SUCCESS)
   Process: 679 ExecStartPre=/bin/chown mosquitto:mosquitto /run/mosquitto (code=exited, status=0/SUCCESS)
 Main PID: 681 (mosquitto)
   Tasks: 1 (limit: 4389)
  Memory: 1.0M (peak: 1.3M)
    CPU: 49ms
   CGroup: /system.slice/mosquitto.service
           `-- 681 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Jul 10 10:27:14 mqttmaacltls150 systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
Jul 10 10:27:14 mqttmaacltls150 systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
```

Ab jetzt wird kein anonymer User von MQTT akzeptiert. Testen Sie das mit dem "MQTT-Explorer"; versuchen Sie sich zunächst ohne User und Passwort die Verbindung aufzubauen. Nutzen Sie anschließend den User "Kai".

4. Test

```
mosquitto_pub -h 192.168.137.150 -p 1883 -t topic/ -u pdal -P JadeHS20 -m "Hallo
vom Client"
```



Sie haben den Zugriff auf den MQTT Dienst geschützt.

- Die Option **-h** stehen für den **Host** (MQTT-Broker),
- die Option **-p** steht für den **Port**,
- die Option **-t** steht für den **Topic**,
- die Option **u** steht für den **User**,
- die Option **-P** steht für das **Passwort** des Users,
- die Option **-m** steht für die **Message** (die Nachricht die wir senden wollen).

Passen Sie den Befehl `mosquitto_sub` entsprechend an und senden Sie eine nachricht mit dem "MQTT-Explorer".

MQTT mit ACLs, benutzerabhängige Topics und Sessions

Warum der Einsatz mit Access Control Lists, benutzerabhängigen Topics und Sessions.

Der Einsatz von MQTT mit **Access Control Lists** (ACLs), **benutzerabhängigen Topics** und **Sessions** erhöht die Sicherheit, Kontrolle und Zuverlässigkeit der Kommunikation in MQTT-basierten Systemen weiter.

ACLs ermöglichen eine feingranulare Zugriffskontrolle, indem sie genau festlegen, welcher Benutzer welche Topics lesen oder schreiben darf. Dadurch wird verhindert, dass unauthorisierte Clients auf sensible Daten zugreifen oder andere Geräte stören.

Benutzerabhängige Topics sorgen dafür, dass jeder Client nur mit seinem eigenen Datenbereich interagiert. Das erhöht die Datensicherheit und Trennung zwischen Benutzern oder Geräten – ein entscheidender Faktor in Multi-User- oder IoT-Umgebungen.

Persistente Sessions stellen sicher, dass ein Client keine Nachrichten verliert, auch wenn er kurzzeitig vom Broker getrennt ist. Der Broker speichert Nachrichten und liefert sie nach, sobald der Client wieder verbunden ist – wichtig für Zuverlässigkeit und Datenkonsistenz.

Insgesamt ermöglichen diese Funktionen eine sichere, skalierbare und stabile MQTT-Architektur, besonders in produktiven oder sicherheitskritischen Anwendungen.

1. ACL-Datei anlegen

```
nano /etc/mosquitto/acl
```



Beispiel:

```
# Benutzer: pdal
user pdal

# Lese- und Schreibrechte auf pdal und Untertopics
topic readwrite pdal
topic readwrite pdal/#

# Leserecht auf Kai/inbox
topic read Kai/inbox

# Benutzer: Kai
user Kai

# Schreibrecht auf Kai/inbox
topic write Kai/inbox

# Leserechte auf Kai und Untertopics
topic read Kai
topic read Kai/#
```

```
# Benutzer: pdal
user pdal

# Lese- und Schreibrechte auf pdal und Unterthemen
topic readwrite pdal
topic readwrite pdal/#

# Leserecht auf Kai/inbox
topic read Kai/inbox

# Benutzer: Kai
user Kai

# Schreibrecht auf Kai/inbox
topic write Kai/inbox

# Leserechte auf Kai und Unterthemen
topic read Kai
topic read Kai/#
```

Welche ACL-Berechtigungen gibt es in MQTT

Bei MQTT in Kombination mit einem Broker wie Mosquitto gibt es folgende Berechtigungen, die über ACLs gesteuert werden:

MQTT-Berechtigungen (ACLs)

1. **read**

Erlaubt das Abonnieren (Subscribe) von Topics.

```
topic read sensor/temperatur
```

 **Bedeutet:** Der Client darf Nachrichten vom Topic `sensor/temperatur` empfangen, aber nicht senden.

2. **write**

Erlaubt das Veröffentlichen (Publish) von Nachrichten auf einem Topic.

```
topic write sensor/temperatur
```

 **Bedeutet:** Der Client darf auf `sensor/temperatur` Nachrichten senden, aber nicht abonnieren.

3. **readwrite** (Standard)

Erlaubt sowohl Lesen (Subscribe) als auch Schreiben (Publish) auf dem Topic.

```
topic readwrite sensor/temperatur
```

☞ **Bedeutet:** Der Client darf empfangen und senden.

4. Wildcards für Topics in ACLs

Man kann MQTT-typische Wildcards verwenden:

+ für eine Ebene

**##* für mehrere Ebenen

Beispiel:

```
topic read sensors/+status
topic write users/+/data/#
```

5. ACLs mit Benutzern kombinieren

```
user Kai
topic readwrite user/Kai/#
```

☞ **Bedeutet:** Nur Benutzer Kai darf Topics im Pfad **user/Kai/...** nutzen.

☞ **Zusammenfassung**

| Berechtigung | Beschreibung |
|--------------|--------------------------------------|
| read | Nur abonnieren |
| write | Nur veröffentlichen |
| readwrite | Beides: abonnieren + veröffentlichen |

ACLs erlauben so eine fein abgestufte Zugriffskontrolle auf Topics – ein wichtiger Bestandteil jeder sicheren MQTT-Architektur.

2. Konfiguration erweitern

Wir wechseln in dieser Konfiguration den Port, um diesen verschärften Sicherheits-Level zu kennzeichnen oder um später Port 1883 für einen anderen Zweck freizuhalten.

```
nano /etc/mosquitto/mosquitto.conf
```

```
# Speichert MQTT-Nachrichten dauerhaft f r Neustarts
persistence true

# Speicherort f r Persistenzdaten wie mosquitto.db
persistence_location /var/lib/mosquitto/

# Startet Mosquitto auf Port 1884 f r eingehende Verbindungen
listener 1884

# Verbietet Verbindungen ohne Benutzernamen/Passwort
allow_anonymous false

# Pfad zur Passwortdatei f r die Benutzer-Authentifizierung
password_file /etc/mosquitto/passwords/mqtt_users
```

```
# Aktiviert das Speichern des Nachrichtenstatus (z. B. retained messages)
persistence true

# Pfad, wo persistente Daten gespeichert werden
persistence_location /var/lib/mosquitto/

# Broker h rt auf Port 1884 (Standard ist 1883, hier bewusst abweichend)
listener 1884

# Deaktiviert anonyme Verbindungen – Benutzername & Passwort sind erforderlich
allow_anonymous false

# Pfad zur Passwortdatei mit g ltigen Benutzern
password_file /etc/mosquitto/passwords/mqtt_users

# Pfad zur ACL-Datei, die Zugriff auf Topics regelt
acl_file /etc/mosquitto/acl

# Speichert persistente Daten alle 1800 Sekunden (30 Minuten)
autosave_interval 1800

# Speichert Daten sofort, wenn sich etwas ndert (nicht nur zeitgesteuert)
autosave_on_changes true
```

```

# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

#pid_file /run/mosquitto/mosquitto.pid
# Speichert MQTT-Nachrichten dauerhaft f  r Neustarts
persistence true

# Speicherort f  r Persistenzdaten wie mosquitto.db
persistence_location /var/lib/mosquitto/

# Startet Mosquitto auf Port 1884 f  r eingehende Verbindungen
listener 1884

# Verbietet Verbindungen ohne Benutzernamen/Passwort
allow_anonymous false

# Pfad zur Passwortdatei f  r die Benutzer-Authentifizierung
password_file /etc/mosquitto/passwords/mqtt_users

# Pfad zur ACL-Datei, die Zugriff auf Topics regelt
acl_file /etc/mosquitto/acl

# Speichert persistente Daten alle 1800 Sekunden (30 Minuten)
autosave_interval 1800

# Speichert Daten sofort, wenn sich etwas  ndert (nicht nur zeitgesteuert)
autosave_on_changes true

```

3. Dienst neu starten

```
sudo systemctl restart mosquitto
```

4. Test

Erlaubt:

```
mosquitto_pub -h 192.168.137.150 -p 1884 -t Kai/logs -m "Log-Eintrag" -u Kai -P
<passwort>
```

Verboten (z. B. Bob auf Kai/#):

```
mosquitto_pub -h 192.168.137.150 -p 1884 -t Kai/logs -m "Unzul  ssig" -u bob -P
<passwort>
```

Das Feld <passwort> durch das gesetzte Passwort ersetzen.

Nun ist das MQTT System noch sicherer. **Aber** alle Topics m ssen in der ACL-Liste gepflegt werden.

Es gibt hierzu ein weiteres Dokument, dass erklärt wie ein MQTT-Broker mittels Zertifikaten abgesichert werden kann.

Im **PDAL** kann man zugunsten der Einfachheit auf passwortgestützte Userverwaltung und die **ACL's** verzichten. In öffentlich zugänglichen Systemen sollte man Broker auf keinen Fall **ungeschützt** betreiben.

Quellen

- CommandMasters. „Understanding ‚setcap‘ Command (with Examples)“. Zugegriffen 10. Juli 2025. <https://commandmasters.com/commands/setcap-linux/>.
 - Destuynder (:kang), Guillaume. „Getcap, Setcap and File Capabilities“. kang's things & stuff, 17. Dezember 2013. <https://www.insecure.ws/2013/12/17/getcap-setcap.html>.
 - Docile, Egidio. „Introduction to Linux Capabilities“. LinuxConfig (blog), 1. November 2023. <https://linuxconfig.org/introduction-to-linux-capabilities>.
 - Inc, EMQ Technologies. „MQTT Guide 2025: Beginner to Advanced“. www.emqx.com. Zugegriffen 9. Juli 2025. <https://www.emqx.com/en/mqtt-guide>.
 - Kerrisk, Michael. The Linux Programming Interface: A Linux Und UNIX System Programming Handbook. Ninth printing. San Francisco, CA: No Starch Press, 2018.
 - „Linux Capabilities: Setting and Modifying Permissions | Baeldung on Linux“, 26. Oktober 2023. <https://www.baeldung.com/linux/set-modify-capability-permissions>.
 - Nordquist, Thomas. „MQTT Explorer“. MQTT Explorer. Zugegriffen 8. Juli 2025. <http://mqtt-explorer.com/>.
 - „paho-mqtt: MQTT version 3.1.1 client class“. MacOS:: MacOS X, Microsoft:: Windows, POSIX, Python. Zugegriffen 9. Juli 2025. <http://eclipse.org/paho>.
-

Lizenz

Dieses Werk ist lizenziert unter der **Creative Commons - Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz**.

[Zum Lizenztext auf der Creative Commons Webseite](#)