mosquitto by cedalo

Full comparison of open-source and Pro Mosquitto features

Last updated: 9.4.2024

Feature name	Feature description	Open-source Mosquitto	Pro Mosquitto
High Availability & Clustering			
Clustering for High Availability	The High Availability feature connects multiple brokers and syncs their internal status, configuration, and data - providing fail-safety against hardware defects or network outages. Clustering for High Availability runs a cluster of Mosquitto nodes with a single leader and multiple followers, ensuring uninterrupted service even if a single node goes down. Users can choose between two cluster modes - Full Sync and Dynamic-Security Sync. Learn more.	✗	✓
Cluster Management UI	Mosquitto Management Center provides a UI for managing broker clusters. Learn more.	✗	✓
Load balancer	The Mosquitto MQTT cluster relies on a load balancer to monitor server availability and closes ports for inactive nodes. If the leader server fails, the cluster reorganizes and appoints a new leader, by which the load balancer will direct all clients to the new leader. Learn more.	✗	✓
Cluster Management REST API	The Cluster Management REST API provides access to detailed Mosquitto cluster information, e.g., which HA clusters the Mosquitto Management Center is connected to and which broker nodes belong to which cluster.	✗	✓
Dynamic-Security Cluster Mode	An HA cluster mode with all nodes available for load distribution, but these nodes only sync their cluster state and dynamic-security-related changes (e.g., adding a client). Learn more.	✗	✓
Enhanced High Availability (HA) Monitoring	The enhanced HA monitoring provides users with real-time output metrics such as voting node count, leader connectivity status, and cluster status indication for deeper insights into the HA cluster's health and resilience. Learn more.	✗	✓
Administration			
Mosquitto Management Center (MMC)	MMC provides a web-based user interface for managing Mosquitto brokers. Learn more.	✓ (Limited to the OS functionality)	✓
Maximal connected brokers	Connects to and manages multiple broker instances with the Mosquitto Management Center. Learn more.	✗ (Limited to one broker node)	✓
Standalone Web Application	Mosquitto Management Center can run independently and does not need to be on the same server as the connected Mosquitto broker instances (including authentication with a TLS client certificate).	✓ (in general)	✓ (TLS)

Broker restart	Mosquitto Management Center enables you to restart your (single) cloud broker instance.	✗	✓
Security			
Client Authentication: ID+password	Basic authentication, where clients provide a username and a password.	✓	✓
Client Authentication by Certificate	MQTT clients have to use the encrypted connection via TLS and provide a client certificate to authenticate.	✓	✓
Client Authentication by Pre-Shared Key (PSK)	The client and the broker both have a Pre-Shared Key, which is used for authentication and verification. Here, no certificate management is needed.	✓	✓
Dynamic security plug-in	This broker extension provides dynamic user management, authentication, and authorization via configuration file or a control API for run-time changes. Learn more.	✓	✓
Access Control Lists (ACLs) based on client, role, and group levels	Access Control Lists (ACLs) can be used for authorization based restrictions on configured topics.	✓	✓
Anonymous client access	Allows clients to access and connect to the broker anonymously without a username or password.	✓	✓
Login rate limit	Limits the number of login attempts for Mosquitto Management Center users to increase security.	✓	✓
Certificate Management	Management of Client Certificate -based Authentication. Learn more.	✗	✓
Custom CAs	Custom CA verifies broker authenticity when connecting Mosquitto Management Center to a broker.	✗	✓
HTTPS/TLS termination at Mosquitto Management Center	HTTPS/TLS terminates directly at Mosquitto Management Center, eliminating the need for a reverse proxy.	✓ (Manual configuration)	✓
Audit Trail	The audit trail creates an append-only log of critical activities captured within Pro Mosquitto or Mosquitto Management Center.	✗	✓
Integrations			
MQTT Bridge (In, Out, or Bidirectional Topic remapping)	An MQTT Bridge connects a broker with another MQTT broker or service. A bridge defines topics the source broker forwards (publishes) or subscribes to on the target broker. When a bridge defines topics for subscribing and publishing, it enables bi-directional communication. Learn more.	✓	✓

HTTP Bridge	An HTTP Bridge forwards data from a broker to a web service that provides an HTTP endpoint. Learn more.	✓	✓
Kafka Bridge	The Kafka bridge enables streaming MQTT data into the Kafka ecosystem by establishing a unidirectional connection between the broker and Kafka server. Users can specify the server, select topics to publish, and map MQTT topics to Kafka topics.	✗	✓
MongoDB Bridge	MongoDB bridge establishes a unidirectional flow of MQTT-generated data from the broker to MongoDB. Users can specify the database and collection for importing, select topics and message fields to import, map topics to specific collections, and choose the fields to be included in the collections.	✗	✓
Prometheus Exporter	Prometheus Exporter allows Prometheus to query Pro Mosquitto for metrics to be monitored directly. The metrics set encompasses the number of clients connected, the count of individual MQTT message types sent and received, and more.	✗	✓
Google Pub/Sub Bridge	The Google Pub/Sub bridge enables one-way data transfer from MQTT topics to Google Pub/Sub topics. Users can specify the Google Pub/Sub server, select MQTT topics to publish, map MQTT topics to Google Pub/Sub topics, select Google Pub/Sub topics for consuming, and map Google Pub/Sub topics to MQTT topics. Learn more.	✗	✓
Google AlloyDB Bridge	Google AlloyDB bridge establishes a one-way data transfer from the broker to Google AlloyDB databases for efficient storage and analysis of high volumes of data. Users can specify the target Google AlloyDB database and table for data import, and map MQTT topics to corresponding Google AlloyDB tables for organized data storage. Learn more.	✗	✓
MySQL Bridge	The MySQL bridge facilitates a one-way flow of MQTT-generated data from the broker to MySQL databases. Users can specify the desired database and table for data import, choose specific MQTT topics and message fields, and map MQTT topics to corresponding MySQL tables for well-organized data storage. Learn more.	✗	✓
PostgreSQL Bridge	The PostgreSQL bridge seamlessly integrates MQTT data into PostgreSQL databases unidirectionally and offers an effective approach to data management and analytics capabilities. Users can specify the target PostgreSQL database and table for data import, map MQTT topics to corresponding PostgreSQL tables for organized data storage, and more. Learn more.	✗	✓

TimescaleDB Bridge	The Timescale DB bridge incorporates MQTT data into TimescaleDB databases to ensure efficient data storage and enhanced analytics capabilities. This bridge empowers users to define the target TimescaleDB database for seamless data ingestion, establish mapping between MQTT topics and TimescaleDB tables, and configure specific fields for insertion into TimescaleDB tables based on individual requirements. Learn more.	✗	✓
MongoDB Atlas Bridge	The MongoDB Atlas Bridge extends the capabilities of the MongoDB Bridge to enable one-way data transfer from MQTT to MongoDB Atlas. Users can generate analytics by specifying the MongoDB Atlas database and collection for data import, selecting MQTT topics and message fields to import into MongoDB, mapping MQTT topics to specific collections within MongoDB, and choosing which message fields to insert into MongoDB collections. Learn more.	✗	✓
InfluxDB Metrics Exporter	The InfluxDB Metrics Exporter exports Pro Mosquitto's operation metrics, including the number of connected clients, MQTT message transmissions, and other relevant performance indicators, to InfluxDB. Users can promptly detect and respond to abnormal behavior, ensuring optimal performance from the Pro Mosquitto broker. Learn more.	✗	✓
Monitoring			
Listing of currently connected clients on a node	The broker provides a control API with information about currently and previously connected clients and properties like MQTT protocol versions.	✗	✓
Client inspection	Detailed information on each connected client, such as: - connection status - connect/disconnect time - protocol information - IP address - TLS encryption information - last will message - message queue usage - subscribed topics Learn more.	✗	✓
Client Control	Client Control allows managing connected clients via a central MQTT API, e.g., subscribe to and unsubscribe MQTT clients from topics, disconnect them, etc. Learn more.	✗	✓
Broker status	The broker status provides information about the status of the connected brokers. Learn more.	✓	✓
Topic tree with drill-down	The topic tree gives you an overview of the topic hierarchy for a given broker. It lets you inspect which topics are used, which messages are sent to those topics, and more drill-down metrics. Learn more.	✓	✓

User management			
Authentication	Secured access to Mosquitto Management Center by using username/password authentication. Learn more.	✓	✓
Unlimited number of users	The number of MMC users.	✗	✓
User roles	Configure Mosquitto Management Center users and assign roles to grant or deny them access to specific features and functionalities.	✗	✓
Role-Based Access Control (RBAC)	Role-Based Access Control (RBAC) defines broker groups that are visible for certain users, roles, or user groups. Learn more.	✗	✓
Single Sign-On (SSO)	Integrates the Mosquitto Management Center User Management with your SAML-based SSO provider. Learn more.	✗	✓
REST APIs			
Dynamic Security	This feature enables the management of MQTT clients, groups, and ACL access rights. Learn more.	✗	✓
Topic Tree	The Topic Tree REST API provides information about the topic tree for every broker connected to Mosquitto Management Center.	✗	✓
Mosquitto Management Center User management	The User Management REST API allows for managing MMC users, groups, and their roles. Learn more.	✗	✓
Cluster Management	The Cluster Management REST API allows you to get detailed information about Mosquitto clusters, e.g., which clusters Mosquitto Management Center is connected to and which Mosquitto nodes belong to which cluster.	✗	✓
Connections	The Connections REST API helps manage connections from Mosquitto brokers to the Mosquitto Management Center (e.g., creating, updating, or deleting them). Learn more.	✗	✓
Application Tokens	The Application Tokens REST API enables the management of application tokens with role-based access & expiration dates. Learn more.	✗	✓
Monitoring	The Monitoring REST API provides monitoring information about the broker instances connected to Mosquitto Management Center and the different metrics for each broker instance. Learn more.	✗	✓
Access control by application tokens	Each Rest API can be accessed using username, password or API token for authentication. Learn more about application tokens and REST APIs.	✗	✓

MQTT protocol versions			
MQTT V3.1.1	An older protocol version that is still widely used and a well-supported standard of MQTT. Learn more.		
MQTT V5	The current MQTT standard provides additional features like session expiry, topic alias, reason codes, shared subscriptions, and more. Learn more.		
Supported protocols			
MQTT over TLS (MQTTS)	Transport Layer Security (TLS) encrypted connection to protect the data sent and received by the broker. Learn more.		
WebSockets (WS)	MQTT connection over WebSockets is a standard protocol that establishes persistent connections based on HTTP. Websockets can help create connections through restricted networks and firewalls. Learn more.		
WebSockets over TLS (WSS)	TLS encrypted MQTT over WebSocket connections.		
Sparkplug	Sparkplug is a communication protocol, providing a standardized way for IoT clients to exchange real-time data efficiently and securely.		
MQTT basics			
Quality of Service (QoS) Levels	MQTT QoS is a level of service that serves as a consensus between a publisher and a broker, as well as a broker and a subscriber. Regarding the latter, it guarantees to dispatch an MQTT message successfully. Learn more.		
Last Will messages	The message that gets published on a specific broker topic if the client unexpectedly loses its connection to the broker. Learn more.		
Retained messages	The broker keeps the last message received on a given topic. Clients will immediately receive these once they subscribe to that topic.		
Persistent connections	The broker will maintain a persistent session with the client, even if the client disconnects.		
Mount points	Mount points are topic prefixes that can create isolated topic trees per listener.		
Supported platforms			
Docker (Linux, Windows, macOS, Raspberry Pi)	—		

RPM (RedHat, CentOS, Rocky Linux)	—	✗	✓
Kubernetes (Linux, Windows, macOS)	Integrate Pro Mosquitto with Kubernetes container orchestration to effortlessly deploy, manage, and scale Mosquitto instances within Kubernetes clusters.	✗	✓
OpenShift	OpenShift enables users to deploy and manage Pro Mosquitto within OpenShift containerized environments, ensuring the secure and scalable functionality of applications and services.	✗	✓
Additional features			
Persistent Queueing	Persistent queueing in Pro Mosquitto stores messages to disk, thus increasing the broker's storage capacity. It further improves outgoing message retention per client and MQTT bridge and enhances reliable delivery even in environments with unstable connectivity. Learn more.	✗	✓
Support of LMDB persistence	The LMDB (Lightning Memory Mapped Database) provides Mosquitto with an even faster ability to save client sessions, retain message information to disk, and recover them in case of a restart.	✗	✓
Processing streams	The number of processing streams that can be defined (e.g., selective redressing, persistence, and replay on separate topics). Learn more.	✗	✓
White labeling of management center UI	Users can customize the logo and colors used in Mosquitto Management Center. Learn more.	✗	✓