smart square

Juni 2021

White Paper

Softwarelizenz oder Public Cloud Service? Open Source oder kommerziell?

MQTT-Broker

für professionelle Anwendungen

Inhalt

Einführung

MQTT ist heute de facto ein IoT-Standard und der MQTT-Broker eine erfolgskritische Komponente in Betrieb und Entwicklung. Dementsprechend sorgfältig sollte er ausgewählt werden.

Der Einsatz von digitalen Technologien und die Vernetzung der beteiligten Komponenten waren lange vor Aufkommen des Begriffes *Industrie 4.0* ein Thema. Einige Kritiker brandmarken ihn als reinen Werbebegriff und wehren sich dagegen, von einer neuen industriellen Revolution zu sprechen. Unbestritten ist jedoch, dass Umfang, Art und Qualität der Vernetzung von Maschinen und Geräten sich immer schneller ändern.

Digitalisierung soll Nutzen durch weitere Produktionszeitverkürzung und Automatisierung sowie intelligente Datenauswertung stiften. Voraussetzung dafür ist eine viel weitergehende Kommunikation zwischen den eingesetzten Systemen und Komponenten, als das bisher der Fall ist. Jeder einzelne Sensor soll heute online sein können.

In den resultierenden, heterogenen Systemarchitekturen muss die M2M-Kommunikation daher herstellerunabhängig stattfinden. Das MQTT-Protokoll hat sich über die letzten Jahre als de facto Standard dafür herausgebildet. Es basiert auf der bewährten Idee, die gesamte Kommunikation nachrichtenbasiert über einen zentralen Vermittler, den sogenan-nten Broker, abzuwickeln. Dieser Ansatz ermöglicht sehr große Infrastrukturen mit vielen angebundenen Geräten und einem hohen Nachrichtenaufkommen unter den schwierigen Bedingungen im industriellen Umfeld und in mobilen Netzen.

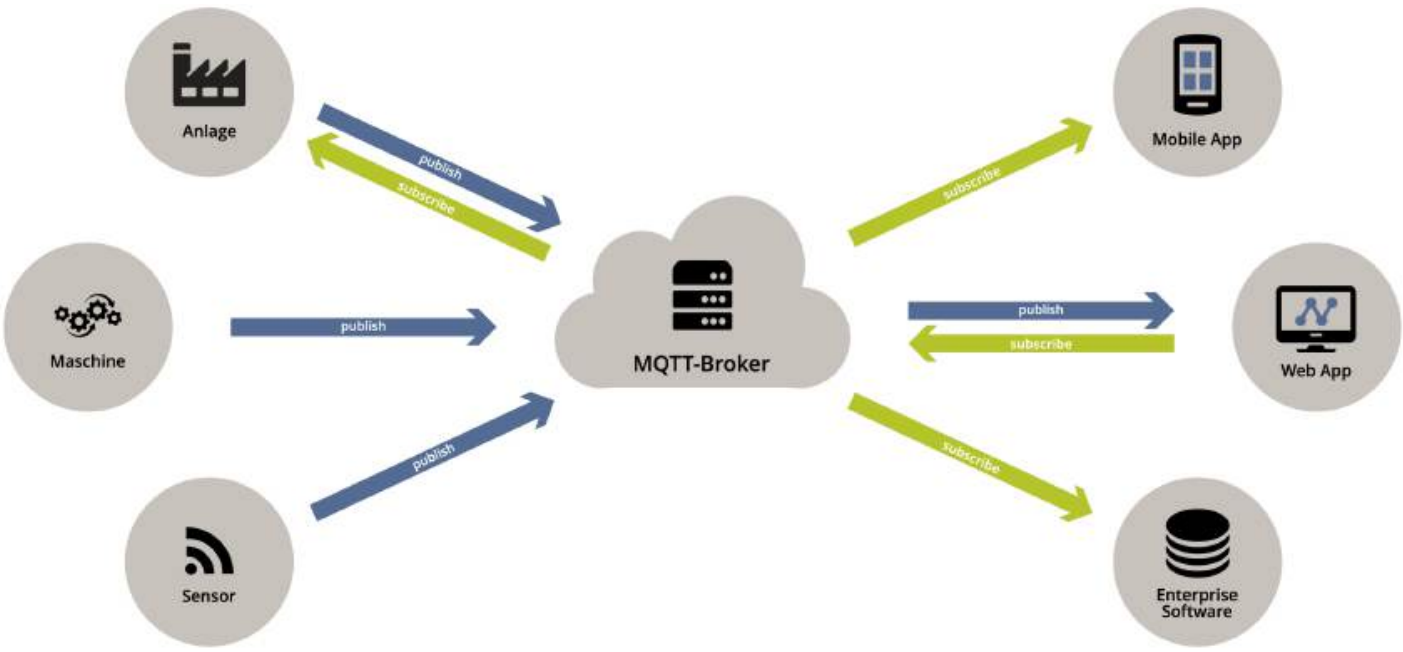
Die große Auswahl an Produkten für die zentrale Softwarekomponente *MQTT Message Broker* kann in drei Hauptströmungen unterteilt werden: Services aus der Public Cloud (also z.B. AWS oder Azure) einerseits sowie Broker zum Eigenbetrieb mit kommerzieller oder Open Source Lizenz andererseits.

Dieses White Paper vergleicht diese drei Ausprägungen an Hand jeweils eines Vertreters und erläutert die relevanten Kriterien für die Auswahl eines Brokers zum Einsatz im profes-sionellen Umfeld.

MQTT Kurzübersicht

Effiziente und zuverlässige Kommunikation zwischen Tausenden von Geräten, die über instabile Netze mit niedrigen Bandbreiten Millionen von Nachrichten austauschen.

MQTT steht für Message Queue Telemetry Transport[1] und ist ein offenes Protokoll für Datenaustausch, das seit 2013 von OASIS[2] standardisiert wird. Es implementiert eine ereignisbasierte Publish/Subscribe Architektur, in der sich alle beteiligten Geräte mit einem zentralen Message Broker verbinden. Datenerzeugende Geräte veröffentlichen dort Nachrichten in thematisch organisierten Kanälen, sogenannten Topics. An den Gerätedaten interessierte Prozesse abonnieren ihrerseits diese Kanäle. Der Broker sorgt als Vermittler für den ordnungsgemäßen Austausch der Nachrichten.



Begriffe und Abkürzungen

MQTT Client; Software, die auf einem Gerät oder auch einem Server läuft und Nachrichten sendet oder empfängt.

Message; Datenpaket, das zwischen den Clients ausgetauscht wird und beliebig aufgebaute Nutzdaten enthält.

Publisher; Client, der Daten erzeugt und als Nachrichten beim Broker veröffentlicht.

Subscriber; Client, der Nachrichten vom Broker erhält und die darin enthaltenen Daten nutzt.

Broker; Server-Komponente, die Verbindungen zu Clients verwaltet und für die korrekte Übermittlung von Nachrichten zuständig ist.

QoS, Quality of Service; zwischen den Clients vereinbarter Service-Level, mit dem die Zuverlässigkeit der Nachrichtenübertragung konfiguriert wird:

- **Level 0:** max. eine Zustellung
- **Level 1:** mind. eine Zustellung
- **Level 2:** genau eine Zustellung

Topic; Thema (oder Kanal), in dem Nachrichten publiziert werden. Subscriber eines Topics empfangen alle dort veröffentlichten Messages. Kann beliebig tief hierarchisch organisiert werden, z.B.:

strasse-2/presse-1/betriebszustand

LWT, Last Will and Testament; spezielle Nachricht, die bei einem ungeordneten Verbindungsabbruch („ungraceful disconnect") publiziert wird. Damit kann gezielt auf Störungen im Netzwerk reagiert werden.

Für IoT gemacht

MQTT ist für das Versenden und Empfangen von Datenpaketen in weit verteilten, wenig zuverlässigen und ggf. auch langsamen Netzwerken entworfen worden[3]. Die miteinander vernetzten Geräte können kleine und leistungsschwache Devices bis hinunter zu einzelnen Sensoren sein, kleinere Rechner mit ARM-Prozessoren oder auch leistungsstarke Server. In solchen Konstellationen punktet MQTT mit seinen größten Stärken:

- verschiedene Servicequalitäten für Zuverlässigkeit in instabilen Netzen
- effiziente Nutzung geringer Bandbreiten
- datenagnostische Nachrichtenstruktur für beliebige Nutzdaten
- ressourcenschonende Implementierung auf kleinen Geräten
- einfache Wiederaufnahme von zeitweilig unterbrochenen Verbindungen
- unmittelbare Erkennung von Verbindungsabbrüchen

De facto Standard

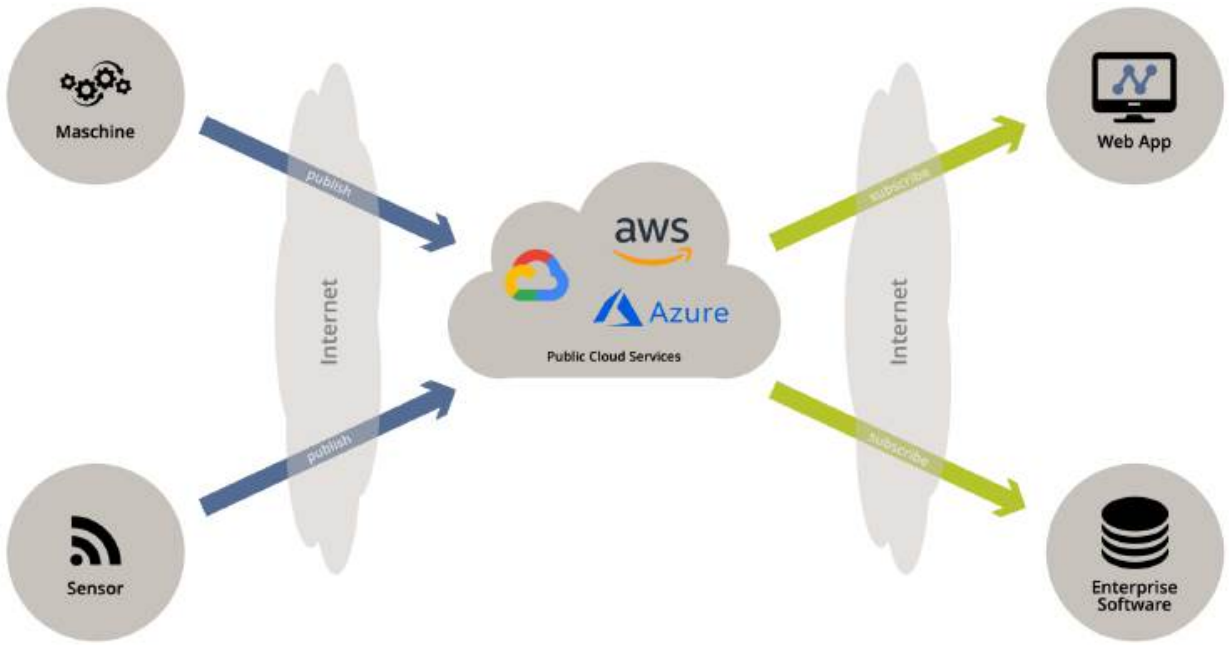
Neben MQTT gibt es weitere Protokolle, die in der Praxis für M2M-Kommunikation eingesetzt werden. So findet sich beispielsweise das XML-basierte Protokoll XMPP[4], das jedoch aufgrund eines hohen Overheads ineffizient und weniger für stark limitierte Netzwerke geeignet ist. CoAP[5] als neuere Alternative basiert auf dem gut etablierten REST-Modell[6]. Es ist sehr effizient und für ressourcenschwache Geräte mit schwacher Netzanbindung geeignet, koppelt jedoch Clients und Server stark. Dadurch ist es weniger universell als MQTT und eher für begrenzte Spezialanwendungen geeignet.

Belastbare Zahlen zu Verbreitung und Marktanteilen sind z.Zt. nicht verfügbar. In der Praxis sehen wir jedoch eine überwältigende Anzahl von (Echt-)Projekten auf der Basis von MQTT. Abdeckung durch Fachliteratur und Internetquellen sowie Unterstützung durch Soft- und Hardwareanbieter deuten ebenfalls darauf hin, dass sich MQTT inzwischen ohne Unterstützung eines zentralen Gremiums und ohne Vermarktungsdruck eines großen Unternehmens zum inoffiziellen IoT-Standard entwickelt hat.

Service oder Produktlizenz?

Die großen Public Cloud Services werden stark beworben und sind leicht zugänglich, doch ein Broker kann auch leicht selbst installiert werden. Eine grundsätzliche Entscheidung.

Drei große Anbieter dominieren zur Zeit den Public Cloud Markt. Unter den Marken Microsoft Azure, Google Cloud und Amazon AWS werden jeweils verschiedenste, eng miteinander verzahnte Produkte und Services angeboten. Die Funktionalität des MQTT-Brokers findet sich dann als ein Modul unter vielen anderen, mit denen es einfach kombiniert werden kann. Er ist somit keine separat zu betrachtende Software, sondern ein voll integrierter Service-Baustein in einem großen Ökosystem. Der Zugriff darauf erfolgt naturgemäß immer über das offene Internet, Sicherheitsthemen wie Verschlüsselung und Abschottung eigener, lokaler Netzwerke kommt somit besondere Bedeutung zu.



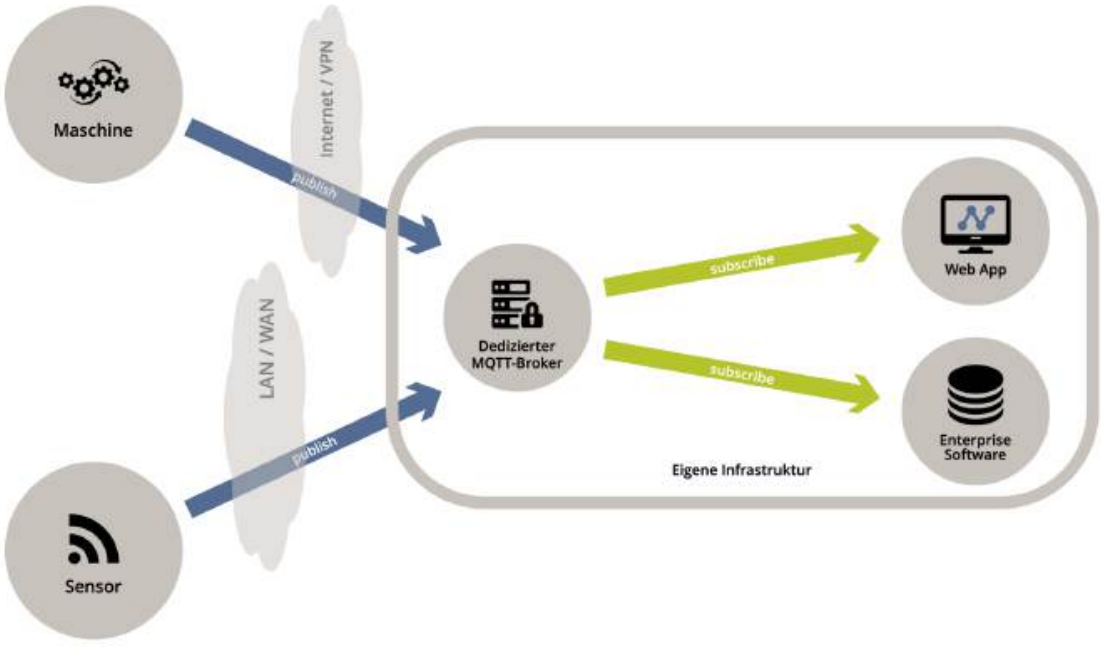
RZ, Private Cloud und andere Optionen

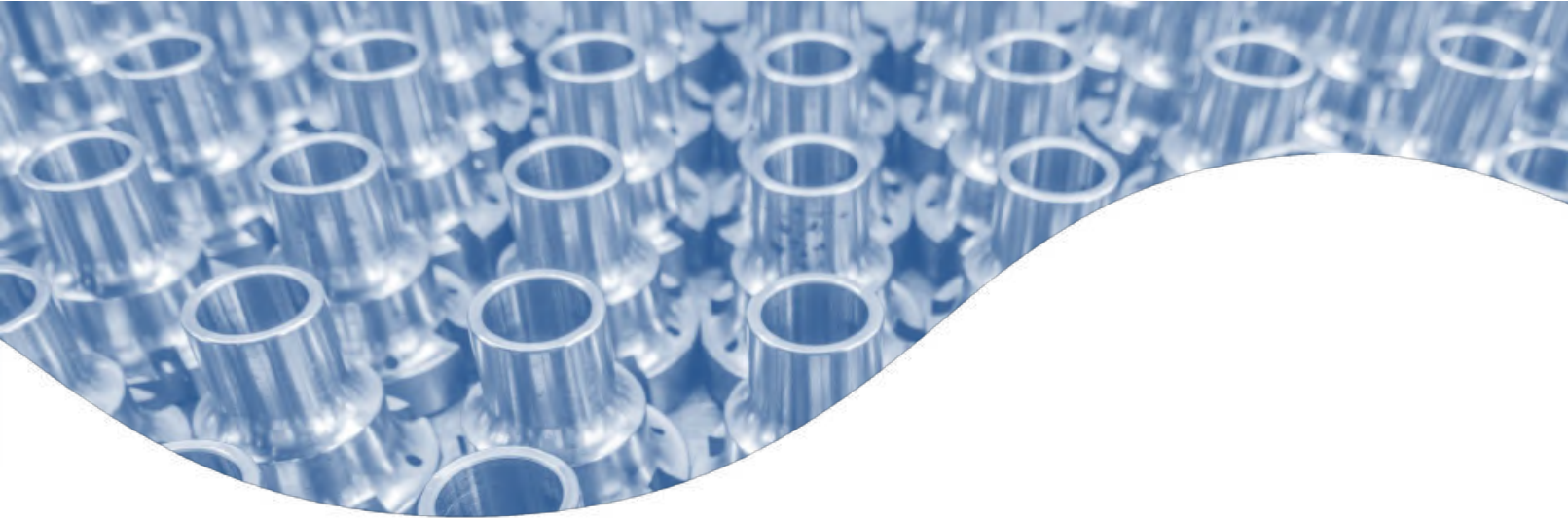
Auf Public Cloud Services zu verzichten bedeutet nicht zwingend, ein Softwarepaket im eigenen Rech-enzentrum auf „Bare Metal“ instal-lieren zu müssen. Der etablierte Virtualisierungsansatz ist in den letzten Jahren weiterentwickelt worden und hat zur Container-virtualisierung geführt. Platzhirsch ist die Software Docker[7], so dass hauptsächlich von „Docker Containern“ gesprochen wird. Mit Orchestrierungswerkzeugen wie Kubernetes lassen sich diese in hohem Maße automatisieren und für Skalierbarkeit clustern. So entsteht, entsprechende Ressourcen und Kompetenzen vorausgesetzt, eine Private Cloud hinter der eigenen Firewall und verbindet viele Vorteile der offenen Cloud mit denen des eigenen Betriebes.

Ein weiterer Schritt ist dann der Um-zug der Container von der eigenen Hardware auf eine beliebige an-dere Cloud Plattform. Im Gegensatz zum Broker als Public Cloud Service ist die Anbieterauswahl hierfür erheblich größer. Neben den bereits genannten großen drei gibt es viele weitere, auch oft in Europa und Deutschland ansässige Provider, denen der Hardwarebetrieb überlassen werden kann. So können die Vorteile der verschiedenen Modelle flexibel kombiniert werden.

Alternativ zur Nutzung eines Services aus der Public Cloud sind auch dedizierte MQTT-Broker erhältlich. Dabei handelt es sich um klassische Softwareprodukte, die auf Basis einer Lizenzvereinbarung vom Nutzer selbst installiert und betrieben werden. Die Anbieter sind kommerzielle Unternehmen oder Open Source Communities, die den Broker als eigenständiges Produkt in ihrem Portfolio sehen und weiterentwickeln.

Welchen Ansatz man bei der Einführung des MQTT-Brokers verfolgt, ist eine kritische Entscheidung. Der Broker ist eine zentrale Komponente und maßgeblich mitverantwortlich für den Erfolg von IoT-Projekten. Dabei ist zum einen zwischen der Kontrollabgabe an einen Cloud-Anbieter und dem Aufwand für eigenen Betrieb abzuwägen. Zum anderen unterscheiden sich die Lösungen aber auch sehr stark in ihren Leistungsmerkmalen. Auf den folgenden Seiten werden diese Unterschiede an Hand von ausgewählten Beispielen herausgearbeitet. Im konkreten Entscheidungsfall kann dies als Basis dafür genutzt werden, zunächst die individuellen Ziele und Rahmenbedingungen zu formulieren.





Bewertungskriterien

Anbieter von MQTT-Brokern legen unterschiedliche Schwerpunkte und listen viele Einzelfeatures auf. Vor einem detaillierten Einzelvergleich betrachtet man aber besser die großen Fragen.

Ist die Implementierung des MQTT-Protokolls ausreichend?

Es gibt mehrere produktive Versionen des MQTT-Protokolls, die sich teilweise erheblich unterscheiden. Deshalb ist zunächst zu klären, welche Versionen von einem Broker überhaupt unterstützt werden. Das alleine genügt jedoch nicht, weil die Anbieter nicht immer alle Protokoll-Features erfüllen. Die Gegenüberstellung auf der nächsten Seite zeigt, dass einige Lösungen große Lücken aufweisen. Hier ist jeweils für den konkreten Anwendungsfall zu bewerten, wie problematisch eine solche Lücke ist.

Wie performant und skalierbar ist der Broker?

MQTT eignet sich für den Aufbau von IoT-Netzen mit extrem vielen Devices und einem sehr hohen Nachrichtenaufkommen. Gute Implementierungen unterstützen dies bereits durch effiziente Umsetzung und geringen Ressourcenverbrauch. Darüber hinaus ist für das geplante Einsatzszenario zu klären, welche Datenmengen erwartet werden. Ein einzelner Broker könnte beispielsweise bei mäßigem Nachrichtenvolumen mit einigen Tausend Clients langfristig ausreichend sein. Falls mehr Leistung benötigt wird, ist eine automatisch skalierende Lösung oder zumindest der Zusammenschluss mehrerer Broker-Instanzen zu einem Cluster notwendig.

Wie wird der Broker betrieben?

Beim Dienst aus der Public Cloud liegen die Dinge für die eigene IT-Abteilung recht einfach, denn der Anbieter trägt per Definition die Verantwortung für Betrieb, Bereitstellung, Sicherung, Sicherheit, etc. Wird eine Softwarenutzungslizenz beschafft, muss der Broker hingegen selbst installiert und betrieben werden. In diesem Fall ist es entscheidend, welche Plattformen und Deployment-

Szenarien das eigene Unternehmen für den Betrieb unterstützen können muss.

Mit welchen Kosten ist zu rechnen?

Der folgenden Gegenüberstellung ist zu entnehmen, dass die Anbieter unterschiedliche Preismodelle verfolgen. Daher ist der direkte Kostenvergleich nicht ohne Weiteres möglich, sondern setzt eine Analyse und Abschätzung des zu erwartenden Nutzungsvolumens voraus. So können Prognosen für komplexe pay-per-use Modelle erstellt und mit Lizenzgebühren verglichen werden.

Welchen Service und Support gibt es?

Neben Weiterentwicklung und ggf. Fehlerbehebung erwarten Unternehmen zumindest von kommerziellen Anbietern erweiterten Service und Support. Dies beginnt bei einer guten Dokumentation als Hilfe bei der Einführung und Integration des MQTT-Brokers. Möglich sind aber beispielsweise auch eine telefonische Unterstützung bei Problemen, Konfigurationsunterstützung oder die Vermittlung von zertifizierten Dienstleistern aus einem Partnernetzwerk.

Werden die benötigten Sicherheitsmechanismen unterstützt?

Das Thema Sicherheit hat mehrere Aspekte, die individuell betrachtet und gewichtet werden müssen. Zeitgemäße Transportverschlüsselung aller übermittelten Daten gilt als Standard und sollte immer genutzt werden. Darüber hinaus sind jedoch auch Authentifikation und Autorisierung zu betrachten, insbesondere die Integration mit im Unternehmen vorhandenen Systemen sowie der zukünftigen Governance im fertigen IoT-Netzwerk. Auch Anwendungsfälle wie Zertifikatserneuerung oder der Ausschluss von Devices gehören dazu.

Protokollversionen

MQTT 3.1.x

MQTT 3.1.0 und 3.1.1 unterscheiden sich in nur wenigen, sehr technischen Details. Oft werden sie deshalb auch als 3.1.x zusammengefasst. Sie sind dennoch nicht ganz identisch. 3.1.1 ist die derzeit am weitesten verbreitete Version.

MQTT 4?

Die Hauptversionsnummer 4 ist im Zuge einer Angleichung der Versionsnummerierung von technischer Implementierung und Dokumentation übersprungen worden. In Datenpaketen steht die 3 für MQTT 3.1.0, die 4 für MQTT 3.1.1. Zukünftig sollen die Nummern synchron laufen und deshalb war der Sprung nötig.

MQTT 5

Version 5.0 ist erst 2018 verabschiedet worden und bringt einige große Neuerungen, die für die Zukunftssicherheit wichtig sind. Insbesondere für die leichtere Skalierbarkeit von Systemen, ein verbessertes Fehlerhandling, leichtere Erweiterbarkeit des Protokolls, einfachere Implementierung neuer Clients sowie bessere Authentifizierungs- und Autorisierungsmechanismen.

Wie gut ist die Integrierbarkeit mit vorhandenen Systemen?

Die Integrierbarkeit mit Systemen für Authentifikation und Autorisierung wie also z.B. Single Sign On Services oder Directories ist aber nur ein Punkt. Letztendlich sollen die von den Geräten übermittelten Daten verar-beitet und genutzt werden, wofür Anbin-dungen an Data Lakes, Enterprise Software Suiten, Monitoring, Webapplikationen und andere Systeme benötigt werden. Um diese realisieren zu können, bedarf es geeigneter Integrationspunkte wie Programmier- und Serviceschnittstellen. Auch in diesem Punkt unterscheiden sich die gegenübergestellten Broker deutlich und auch hier muss im Lichte der individuellen Anforderungen abgewogen werden.

Broker in diesem Vergleich

Eine Auswahl von drei bewährten MQTT-Brokern erlaubt bereits Rückschlüsse auf gänzlich unterschiedliche Philosophien von Public Cloud Services sowie kommerziellen und offenen Produkten.

Der hohe Verbreitungsgrad von MQTT kombiniert mit dem Wachstum des Themenkomplexes „Industrie 4.0, IoT und M2M“ führen auch dazu, dass viele Softwareprodukte in diesem Bereich entstehen. Alleine die entsprechende Wikipedia-Seite[8] listet zehn Broker auf, ohne Anspruch auf Vollständig-keit. Hinzu kommen dort nicht enthaltene Public Cloud IoT-Services und einige weitere kommerzielle wie freie Produkte.

Für den Echteinsatz geeignet?

Wird ein MQTT-Broker im Echtbetrieb gefahren, ist er eine kritische Komponente. Ein Ausfall legt sofort das eigene IoT-Netz lahm und führt im besten Fall lediglich zu kurzfristiger Blindheit oder Lücken in der Datenauswertung. Werden jedoch auch Geräte gesteuert oder sind Kunden spürbar von Ausfällen betroffen, entsteht direkt messbarer Schaden. Unter diesem Aspekt betrachtet wird die Anzahl der potenziell geeigneten Broker für professionelle Anwender schon geringer.

Grundsatzfragen

Die noch infrage kommenden Lösungen verfolgen unterschiedliche Philosophien und sind nur begrenzt miteinander vergleichbar. Es ist nicht möglich, einen für alle Nutzer gleich gut geeigneten „Testsieger“ zu küren. Vielmehr muss zunächst entschieden werden, welche grundsätzliche Philosophie zum eigenen Unternehmen passt: Ein möglichst schlanker Broker, eine Suite mit wachsendem Angebot aus Tools und Add-ons oder der Schritt in eine Cloud, in der Messaging nur einer von vielen kleinen Services ist?

Die drei im Folgenden gegenübergestellten Lösungen sind typische Vertreter dieser Ansätze. An ihnen wird deutlich, welche Vorzüge und Einschränkungen jeweils impliziert sind. Mit diesem Wissen kann dann eine Shortlist geeigneter MQTT-Broker für individuelle, unternehmensspezifische Anwendungsszenarien erstellt werden.

Amazon Web Services IoT Core

Amazon bietet über das Tochterunternehmen AWS eine sehr breite Palette verschiedenster Webservices an. AWS ist vor Microsoft und Google der global führende Public Cloud Services Anbieter mit stetig wachsender Kundenbasis auch in Deutschland.

AWS IoT Core ist der Service für „device connectivity and messaging" innerhalb von AWS IoT. Es steht neben weiteren Produkten z.B. für die Geräteverwaltung und Datenanalyse. Aus Sicht von AWS handelt es sich dabei also nicht um ein alleinstehendes Produkt, sondern um eine integrierte Verbindungskomponente auf Basis von MQTT.

HiveMQ - Enterprise MQTT-Platform

HiveMQ ist ein kommerzielles Produkt des deutschen Unternehmens HiveMQ GmbH (ehemals dc-square GmbH). Dort konzentriert man sich auf die Vernetzung von Geräten und IoT-Kommunikation im großen Maßstab. Seit der Gründung 2011 ist HiveMQ stark gewachsen und hat inzwischen eine breite, internationale Kundenbasis, zu der auch viele Fortune 500 Unternehmen gehören.

HiveMQ wird als das „Flagship Product" bezeichnet und als „MQTT based messaging platform" beschrieben. Das unterstreicht den deutlich erkennbaren Anspruch, über die reine Broker-Funktionalität hinaus ein Ökosystem an Tools, Erweiterungen, Adaptern u.ä. anzubieten.

Eclipse Mosquitto - MQTT-Broker

Mosquitto ist eine Open Source Software der Eclipse Foundation, einer etablierten Dachorganisation für Open Source Projekte. Bekannt ist die Eclipse Foundation zwar für die gleichnamige Programmierumgebung, es werden jedoch viele weitere Top Level Projekte mit mehr als 350 zugeordneten Open Source Projekten vorangetrieben.

Am Top Level Projekt Eclipse IoT, zu dem Mosquitto gehört, arbeiten Unternehmen wie Bosch, Red Hat, IBM, Microsoft aber auch die Fachhochschule Dortmund. Mosquitto ist als „leightweight message broker" ausgelegt und klar auf diese Funktionalität fokussiert.

Gegenüberstellung der Broker 1/2

Produkt	AWS IoT Core
Anbieter / Hersteller	Amazon Web Services
Internet	https://aws.amazon.com/de/iot-core/
Lizenz(en)	kommerziell
Abrechnungsmodell(e)	Nutzungsabhängig nach einem komplexen Preismodell[9]: • keine Mindestgebühr oder Mindestabnahmemenge • separate Abrechnung von Verbindungen, Nachrichten, Gerätestatus- und Gerätedatenspeicher sowie Rules Engine • kostenfreies Kontingent ist enthalten
Herstellersupport	• Basis-Support immer kostenfrei enthalten • Developer-, Business- und Enterprise-Support z.B. mit abgestuften Verfügbarkeitszeiten, Reaktionszeiten, Kommunikationskanälen und Support-Umfängen kostenpflichtig buchbar • Abrechnung nach einem nochmals in sich gestaffelten, nutzungsabhängigen Preismodell mit Mindestgebühr
Dienstleistungsangebote des Anbieters / Herstellers	• Anwendungsfallbezogene Unterstützung und Beratung sind Teil kostenpflichtiger Supportpläne • "Proactive Programme" ist extra buchbar oder Teil des Enterprise Supportplans und enthält Betriebsunterstützung, Architektur und Konfiguration • AWS betreibt aktiv den Aufbau eines Partner- und Dienstleistungsnetzwerkes, so dass auch dritte (zertifizierte) Unternehmen Dienstleistungen anbieten
Plattform und Betrieb	ausschließlich AWS Public Cloud
verfügbare Management Tools	• AWS Management Console • AWS IoT-Dashboard • Amazon CloudWatch

	HiveMQ	Mosquitto
	dc-square GmbH	Eclipse Foundation
	https://www.hivemq.com	https://mosquitto.org
	• kommerziell (Editionen Professional & Enterprise) • Apache v2 (Edition Community & MQTT Client)	• Eclipse Public License v1.0 • Eclipse Distribution License v1.0 (BSD)
	kommerzielle Lizenz kostenpflichtig, Preise nur auf Anfrage	Kostenfrei, auch für kommerzielle Nutzung; EPL und EDL sind *business-friendly* und haben keinen *Copyleft*-Effekt[10].
	• Basis-Support in kostenpflichtigen Angeboten • individuelle Support-Vereinbarungen sind möglich (auch 24/7)	• nur über die Open Source Community • keine kommerziellen Anbieter bekannt
	• Consulting • Produktionsunterstützung • Entwicklung individueller Erweiterungen • Prüfung und Zertifizierung selbstentwickelter Erweiterungen • Schulungen und Workshops	keine
	Java-basierende Software zum Deployment auf: • Bare Metal Server • Virtuellen Maschinen • Docker und Docker Clustern • Private, Hybrid und Public Clouds	Binärpakete für diverse Linux Distributionen, Windows, Mac und Raspberry Pi sowie Quelltext zum Kompilieren auf anderen Zielplattformen. Deployment möglich auf: • Bare Metal Server • Virtuellen Maschinen • Docker und Docker Clustern • Private, Hybrid und Public Clouds
	• Monitoring Dashboard und Analysewerkzeuge • Extension Management • Extensions für System Monitoring Für den Produktivbetrieb wird der Einsatz spezialisierter Monitoringwerkzeuge empfohlen, für die Anbindungen vorgesehen sind.	keine

Gegenüberstellung der Broker 2/2

Produkt	AWS IoT Core
MQTT Clients und Client Libraries von diesem Anbeiter / Hersteller	• AWS IoT Device SDKs für die Client-Entwicklung werden für die Sprachen C und Node.js sowie für die Arduino-Plattform von AWS zur Verfügung gestellt • Die SDKs sind Open Source und können für eigene Entwicklungen verwendet sowie in andere Sprachen und Plattformen übertragen werden.
implementierte MQTT Protokollversionen	3.1.1, weicht jedoch in einigen Punkten von der Spezifikation ab
unterstützte Quality of Service Level	0, 1
Persistent Sessions und Message Queuing	ja Anmerkung: Wird erst seit Januar 2019 unterstützt, im Internet finden sich noch oft gegenteilige Aussagen.
Retained Messages	nein
Last Will and Testament (LWT)	ja
Skalierbarkeit	Praktisch unbegrenzt skalierende Cloud-Infrastruktur von Amazon, transparent für den Anwender.
Integration mit Drittsystemen	• direkte Anbindung an das umfangreiche AWS-Ökosystem z.B. DynamoDB, Kinesis, Lambda, SNS, SQS, CloudWatch, Elasticsearch Service • Anbindung an externe Systeme über Lambda möglich
Authentifikation	Amazon Cognito, X.509-Zertifikat, Anbindung externer Authentication Services über Lambda möglich
Autorisierung	AWS IoT-Richtlinien (für Amazon Cognito-Identitäten), Amazon IAM-Richtlinien, Anbindung externer Authorization Services über Lambda möglich
	TLS / SSL (zwingend)
IP4 / IP6 / Websockets	ja / ja / ja
offizielle Benchmarks / Showcases	• Skaliert "auf Milliarden von verschiedenen Geräten und Billionen von Nachrichten" • Showcases zeigen erfolgreich implementierte Anwendungsfälle mit verschiedensten Mengengerüsten

	HiveMQ	Mosquitto
	• Java Client	• Open Soruces Kommandozeilen-Clients • Open Source C-Bibliothek ermöglicht Entwicklung eigener Clients
	5.0, 3.1.1, 3.1.0	5.0, 3.1.1, 3.1.0
	0, 1, 2	0, 1, 2
	ja	ja
	ja	ja
	ja	ja
	Verteilte Cluster-Architektur ohne Single Point of Failure sowie Hardware- und Software-Loadbalancing sind möglich. Müssen vom Anwender in Abhängigkeit vom Deployment-Szenario geplant und konfiguriert werden.	Bridges zu anderen MQTT Servern (Mosquitto oder auch andere) ermöglichen den Aufbau von Clustern. Müssen vom Anwender in Abhängigkeit vom Deployment-Szenario geplant und konfiguriert werden.
	• API für Integration von Drittsystemen • Extension Framework für eigene Implementierungen • im Marketplace sind fertige Extensions z.B. für Kafka sowie SQL und NoSQL Datenbanken verfügbar • Besonderheit: Über den Marketplace können selbstentwickelte Extensions auch verbreitet/ vermarktet werden.	• API für Integration von Drittsystemen • Open Source Projekt kann beliebig erweitert werden
	Client ID, Benutzername & Passwort, X.509-Zertifikat, weitere Authentication Plugins können vorgefertigt bezogen oder selbst entwickelt werden	Benutzername & Passwort, X.509-Zertifikat, weitere Authentication Plugins können nach individuellen Anforderungen selbst entwickelt werden
	Authorization Plugins können vorgefertigt bezogen oder nach individuellen Anforderungen selbst entwickelt werden.	Access Control List Datei in der Broker-Konfiguration
	TLS / SSL (optional)	TLS / SSL (optional)
	ja / ja / ja	ja / ja / ja
	• 10 Millionen Clients, die 3 Milliarden Nachrichten pro Stunde austauschen (auf AWS Infrastruktur ausgeführt). • Das Benchmark Paper kann nach Registrierung heruntergeladen werden.	• keine offiziellen Benchmarks und Showcases verfügbar • Berichtet wird von hoher Stabilität und niedrigem Ressourcenverbrauch bei 100.000 Clients und "mittlerem" Nachrichtenvolumen.

Schlussfolgerungen

Die Gegenüberstellung der drei Broker zeigt deutlich, dass nicht jeder Ansatz gleichermaßen für alle Anwendungsfälle und jedes einsetzende Unternehmen geeignet ist.

Fundamentale Unterschiede

Den Message Broker als Cloud Service zu beziehen unterscheidet sich demnach in vielen grundsätzlichen Aspekten vom Eigenbetrieb einer entsprechenden Software, sei es im eigenen Rechenzentrum oder in einer Cloud. Und entscheidet man sich für die Softwarelizenz, impliziert ein Open Source Modell ganz andere Möglichkeiten und Herausforderungen, als ein kommerzielles Angebot.

Detaillierte Einzelvergleiche von Produkten fördern natürlich noch viel mehr Facetten und Ausprägungen zutage, als die vorliegende Gegenüberstellung. Die Unter-schiede zwischen beispielsweise *AWS IoT Core* und *Azure IoT Hub* sind jedoch viel geringer als etwa zwischen *AWS IoT Core* und *HiveMQ* oder *Mosquitto*. Daher sollte zunächst geklärt werden, welcher Ansatz überhaupt geeignet ist. Danach können infrage kommende Produkte verglichen werden.

Anforderungsprofil

Aus dem vorliegenden Vergleich lassen sich zentrale Fragen zur Erstellung eines individuellen Anforderungsprofils ableiten. Auf der nebenstehenden Seite finden Sie die wesentlichen Themen kurz zusammengefasst. Nutzen Sie diese Liste, um damit den eigenen Bedarf und auch die eigenen Möglichkeiten (z.B. im Betrieb einer ggf. unternehmenskritischen Software) zu hinterfragen und ein individuelles Anforderungsprofil zu erstellen.

Mit diesem Profil werden Sie relativ schnell entscheiden können, welche Lösungsansätze für Ihre Anwendung infrage kommen. Damit wiederum schränken Sie Ihre Auswahl an Produkten stark ein. Ein Startpunkt für die weitere Arbeit ist die Liste von MQTT-Brokern auf Seite 19.

Interoperabilität und Protokollimplementierung

AWS IoT Core und Azure IoT Hub legen sichtbar weniger Wert auf die vollständige Implementierung des jeweils aktuellsten MQTT-Protokolls. Im geschlossenen Ökosystem eines Anbieters ist dies tatsächlich auch weniger relevant, da das Zusammenwirken der eigenen Komponenten ausschlaggebend ist. Aus Anwendersicht muss jedoch geklärt werden, welche MQTT-Features (QoS-Level 2 ist nur ein Beispiel) zwingend benötigt werden. Eine akkurate Protokollimplementierung ist zudem zwingend zum Anschluss von Clients, die nicht durch anbieterspezifische Clients ersetzbar sind.

Performance und Skalierbarkeit

Public Cloud Service sind praktisch unbegrenzt und skalieren mühelos. Anwender sollten jedoch zum einen analysieren, welche Mengengerüste tatsächlich abgebildet werden müssen. Zum anderen sind auch die dedizierten Broker nicht zu unterschätzen, zumal wenn die Software Clustering unterstützt und auf gut skalierbarer Cloud-Infrastruktur betrieben werden kann. HiveMQ demonstriert das sehr gut mit seinem Benchmark in der AWS Cloud. Mosquitto zu clustern ist eher schwierig, andere Open Source Lösungen sind speziell an dieser Stelle aber weiter.

Betrieb und Sicherheit

Einen MQTT-Broker selbst zu betreiben, setzt entsprechende Ressourcen (Kapazität und Kompetenz) voraus. Das gilt nicht nur für die Installation im eigenen Rechenzentrum, sondern auch beim Deployment z.B. als Docker Container in einer Cloud Infrastruktur. Sicherheitsfragen kommt dabei im IoT-Umfeld ganz besondere Bedeutung zu. Anwender sollten ehrlich hinterfragen, wie gut sie dafür aufgestellt sind und im Zweifelsfall den Betrieb an einen Cloud Service Anbieter auslagern. Auch das Sicherheitsthema ist dort gut aufgehoben.

Integration und Erweiterbarkeit

Hier repräsentiert Mosquitto die Grundidee von Open Source Produkten, dass ein gutes API und offener Quelltext für Softwareentwickler die beste Ausgangslage darstellen. Der kommerzielle Produktanbieter HiveMQ setzt auf ein Extension-Modell und AWS verweist auf die eigenen Services, mit denen ggf. auch individuelle Logik implementiert werden kann. Aus Anwendersicht ist zu klären, mit welchen anderen Systemen die eigene IoT-Lösung integriert werden soll, wie gesammelte Daten weiterzuverarbeiten und welche technischen Anforderungen umzusetzen sind.

Selbsthilfe und Herstellerunterstützung

Hinsichtlich der Anwenderdokumentation, Support sowie Verfügbarkeit von Services durch Drittanbieter stellen sich Anbieter mit starker kommerzieller Komponente erheblich breiter und vollständiger auf, als das reine Open Source Projekt Mosquitto. Dort muss das anwendende Unternehmen sich mit spartanischer Kurzdokumentation, Quelltext und dem Internet selbst zu helfen wissen. Dafür bedarf es der entsprechenden Ressourcen im Unternehmen und der dazu passenden eigenen Unternehmensphilosophie.

Management Summary

Industrie 4.0, das Internet of Things und M2M-Kommunikation sind für vielen Branchen unausweichliche, mittelfristig erfolgsentscheidende Themen. Produkte, Dienstleistungen und Geschäftsmodelle in diesem Bereich sind schon heute vielfältig. Es ist zu erwarten, dass durch steigende Möglichkeiten und den Trend zu mehr Individualisierung die Palette noch größer wird.

Diese Welt wird aus heterogenen, sehr weitläufigen IoT-Netzwerken bestehen und auf absehbare Zeit noch häufig über instabile und leistungsschwache Netze kommunizieren ganz unterschiedliche IT-Systeme verbinden. MQTT hat sich als verbindender Standard dafür etabliert und der MQTT-Broker als zentraler Nachrichtenvermittler ist eine kritische Komponente im System.

Am Markt ist eine Vielzahl von Message Brokern verfügbar, die MQTT implementieren und diese Rolle übernehmen können. In einer ersten Annäherung sollte eine grundsätzliche Wahl zwischen Public Cloud Services und dedizierten Softwareprodukten getroffen werden. Letztere unterscheiden sich nochmals stark, wenn sie statt von einem kommerziellen Anbieter von einer Open Source Community entwickelt werden. Die Auswahl muss an unternehmens- und anwendungsfallspezifischen Anforderungen in den folgenden Bereichen festgemacht werden:

- Interoperabilität und Protokollimplementierung
- Performance und Skalierbarkeit
- Betrieb und Sicherheit
- Integration und Erweiterbarkeit
- Selbsthilfe und Herstellerunterstützung

Für alle Anforderungsschwerpunkte gibt es sehr gut geeignete Produkte, die untereinander wiederum stark differenzieren. Keines der analysierten Produkte ist von vornherein grundsätzlich ungeeignet oder den anderen in allen Aspekten überlegen.

Links und Referenzen

- **[1]** *https://mqtt.org*
- **[2]** *https://www.oasis-open.org*
- **[3]** *https://de.wikipedia.org/wiki/*
- **[4]** *https://xmpp.org*
- **[5]** *https://coap.technology*
- **[6]** *https://de.wikipedia.org/wiki/Representational_State_Transfer*
- **[7]** *https://www.docker.com*
- **[8]** *https://en.wikipedia.org/wiki/Comparison_of_MQTT_implementations*
- **[9]** *https://aws.amazon.com/de/iot-core/pricing*
- **[10]** *https://en.wikipedia.org/wiki/Eclipse_Public_License*

Weitere MQTT-Broker

Public Cloud

- Google Cloud IoT Core
  (Public Cloud Service, kommerziell)
- Microsoft Azure IoT Hub
  (Public Cloud Service, kommerziell)
- HiveMQ Cloud
  (Public Cloud Service, kommerziell)
- ThingScale IoT message broker
  (Public Cloud Service, kommerziell)

Open Source

- VerneMQ
  (Softwarelizenz, Open Source)
- EMQX
  (Softwarelizenz, Dual License OS und komm.)
- HiveMQ Community
  (Softwarelizenz, Open Source)
- Apache ActiveMQ Artemis
  (Softwarelizenz, Open Source)
- RabbitMQ
  (Softwarelizenz, Open Source)

Kommerziell

- Bevywise MQTT Broker
  (Softwarelizenz, kommerziell)
- JoramMQ
  (Softwarelizenz, kommerziell)
- SwiftMQ
  (Softwarelizenz, kommerziell)
- IBM Message Sight & Websphere MQ Telemetry
  (Softwarelizenz, kommerziell)
- flespi
  (Softwarelizenz, kommerziell mit Gratisangebot)
- Litmus Automation Loop
  (Softwarelizenz, kommerziell)

Gute Ideen brauchen gute Software.

Smartsquare ist ein auf Individualentwicklung spezialisiertes Softwareunternehmen. Unsere Kunden haben besondere Ideen und Anforderungen, die man mit Programmen, Web-Applikationen oder Apps von der Stange nicht erfüllen kann. Gemeinsam entwerfen wir ideale, maßgeschneiderte Lösungen und Smartsquare entwickelt die benötigte Software. Ein Schwerpunkt liegt dabei auf IoT-Plattformen zur Vernetzung von Maschinen und Anlagen. Wir ermöglichen Teleservice aus der Cloud und eröffnen auch kleinen und mittelständischen Unternehmen in Nischenmärkten neue, digitale Geschäftsmodelle.

Smartsquare GmbH
Otto-Brenner-Str. 247
33604 Bielefeld

www.smartsquare.de

+49 521 4481 8690