MQTT to Kafka: Benefits, Use Case & A Quick Guide

EMQX Team Mar 26, 2024 Eco & Integration

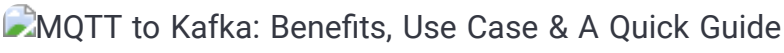


Table of Contents

How Is MQTT Used with Kafka?

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol for efficient communication between devices in constrained networks. Apache Kafka is a distributed streaming platform. It is designed to handle large-scale, real-time data streaming and processing.

Kafka and MQTT are complementary technologies that enable end-to-end integration of IoT data. By integrating Kafka and MQTT, businesses can establish a robust IoT architecture that guarantees reliable connectivity and efficient data exchange between devices and IoT platforms. At the same time, it also facilitates high throughput real-time data processing and analysis throughout the entire IoT system.

There are many IoT use cases where integrating MQTT and Kafka provides significant value, such as **Connected Cars** and Telematics, Smart City Infrastructure, **Industrial IoT** Monitoring, Logistics Management, etc. In this blog post, we will explore the seamless integration of MQTT data with Kafka for the IoT Application.

Which IoT Challenges Can Kafka and MQTT Address?

When designing an IoT platform architecture, several challenges arise that need to be addressed:

- **Connectivity and network resilience:** Critical IoT scenarios, such as Connected Cars, rely on network connectivity to transmit data to the platform. The architecture should be designed to handle intermittent connectivity, network latency, and varying network conditions.
- **Scaling:** As the number of devices increases, the architecture must be scalable to handle the growing volume of data generated by IoT devices.
- **Message Throughput:** IoT devices generate a vast amount of data in real-time, including sensor readings, location information, and so on. The platform architecture must be capable of handling high message throughput to ensure

that all data is efficiently collected, processed, and delivered to the appropriate components.

- **Data storage:** IoT devices generate a continuous stream of data, which needs to be stored and managed effectively.

The Need to Integrate MQTT with Kafka in an IoT Architecture

While Kafka excels in its role as a reliable streaming data processing platform for facilitating data sharing between enterprise systems, certain limitations make it less ideal for IoT use cases:

- **Client complexity and resource intensiveness:** Kafka clients are known for their complexity and resource requirements. This poses difficulties for smaller IoT devices with constrained resources, as running a Kafka client on such devices may be impractical or inefficient.
- **Topic scalability:** Kafka has limitations in handling a large number of topics. This can be problematic for IoV deployments with extensive topic definition, as they may not seamlessly fit into Kafka's architecture, especially in scenarios involving a significant number of devices and multiple topics in each device.
- **Unreliable connectivity:** Kafka clients require a stable IP connection, which proves challenging for IoT devices operating over unreliable mobile networks. These networks can introduce intermittent connectivity issues, disrupting the consistent communication required by Kafka.

Integrating MQTT with Kafka can help address most of the limitations of Kafka in IoT device connectivity scenarios:

- **Direct addressing:** MQTT supports load balancing, enabling IoT devices to connect to Kafka brokers indirectly through load balancers.
- **Topic scalability:** MQTT is well-suited for handling many topics, making it an ideal candidate for IoT platform deployments with extensive topic design.

- **Reliable connectivity:** MQTT is designed to operate over unreliable networks, making it a reliable messaging protocol for IoT devices and connections.
- **Lightweight client:** **MQTT clients** are designed to be lightweight, making them more suitable for resource-constrained IoV devices.

Comparison of Viable MQTT-Kafka Integration Solutions

When integrating MQTT and Kafka in an IoT platform, several viable solutions are available. Each solution offers its own advantages and considerations. Let's explore some of the popular MQTT + Kafka integration options:

EMQX Kafka Data Integration

EMQX is a popular **MQTT broker** that offers seamless integration with Kafka through its Kafka Data Integration feature. As a bridge between MQTT and Kafka, EMQX enables smooth communication between the two protocols.

This integration allows the creation of data bridges to Kafka in two roles: producer (sending messages to Kafka) and consumer (receiving messages from Kafka). EMQX allows users to establish data bridges in either of these roles. With its bi-directional data transmission capability, EMQX provides flexibility in architecture design. Additionally, it offers low latency and high throughput, ensuring efficient and reliable data-bridging operations.

Learn more about the EMQX Kafka data integration: **Stream Data into Kafka**.

Confluent MQTT Proxy

Confluent is the company behind Kafka. Its MQTT Proxy connects MQTT clients and Kafka brokers, allowing them to publish and subscribe to Kafka topics. This solution

simplifies the integration process by abstracting the complexities of direct communication with Kafka brokers.

Currently, this solution is limited to supporting MQTT version 3.1.1, and the performance of MQTT client connections may influence the throughput.

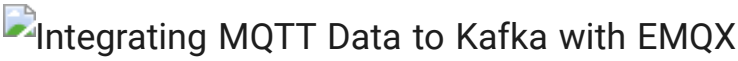
Custom Development with Open-Source MQTT Broker and Kafka

With the use of an **open-source MQTT Broker**, users have the flexibility to develop their own bridge service that connects MQTT and Kafka. This bridge service can be built using an MQTT client to subscribe to data from the MQTT Broker and utilize the Kafka producer API to publish the data into Kafka.

This solution requires development and maintenance efforts, as well as significant work to ensure reliability and scalability.

Integrating MQTT Data to Kafka with EMQX

EMQX is a highly scalable MQTT broker that offers extensive features and capabilities for IoT platforms. EMQX's data integration capability allows for easy and efficient streaming of MQTT data into or from Apache Kafka.


Integrating MQTT Data to Kafka with EMQX

EMQX provides massive-scale device connectivity. Together with the high-throughput, durable data processing capability from Kafka, this provides a perfect data infrastructure for IoT.

MQTT to Kafka features provided by EMQX include:

- **Bidirectional connection:** EMQX supports batching MQTT messages from devices towards Kafka, also fetching Kafka messages from the backend system to publish to connect IoT clients.

- **Flexible MQTT-to-Kafka topic mapping:** For example, one-to-one, one-to-many, many-to-many, including **MQTT topic** filters (wildcards).
- **EMQX Kafka producer** supporting synchronous/asynchronous write mode, making it flexible when prioritizing latency vs. throughput.
- **Realtime metrics**, such as the total number of messages, the number of succeeded/failed deliveries, messaging rate, etc., integrated with SQL IoT rules to extract, filter, enrich, and transform data before pushing the messages to Kafka or devices.

Example Use Case: Leveraging MQTT and Kafka for Connected Cars and IoV

The architecture of MQTT + Kafka offers benefits for various IoT platforms across different industries, and the domain of connected cars and the Internet of Vehicles (IoV) is a particularly compelling use case.

 The architecture of MQTT + Kafka

Here are the primary use cases for this architecture:

- **Telematics and vehicle data analytics:** MQTT + Kafka architecture allows for collecting, streaming, and analysis of large-scale real-time vehicle data, such as sensor readings, GPS location, fuel consumption, and driver behavior. This data can be utilized for vehicle performance monitoring, predictive maintenance, fleet management, and improving overall operational efficiency.
- **Intelligent traffic management:** By integrating MQTT and Kafka, it becomes possible to capture and process data from various traffic sources, including connected vehicles, traffic sensors, and infrastructure. This enables the development of intelligent traffic management systems, including real-time traffic monitoring, congestion detection, route optimization, and smart traffic signal control.

- **Remote diagnostics:** MQTT + Kafka architecture facilitates the high throughput data transmission of connected cars. It can be leveraged for remote diagnostics and troubleshooting, allowing for proactive maintenance and efficient issue resolution.
- **Energy efficiency and environmental impact:** MQTT + Kafka architecture enables the integration of connected cars with smart grid systems and energy management platforms with Bi-direction data transmission. This use case involves real-time energy consumption monitoring, demand response mechanisms, and electric vehicle charging optimization.
- **Predictive maintenance:** MQTT + Kafka architecture enables continuous monitoring of vehicle health and performance data. This use case involves high throughput real-time telemetry data collection, anomaly detection, and predictive maintenance algorithms. Car owners can proactively identify potential issues and schedule maintenance tasks.

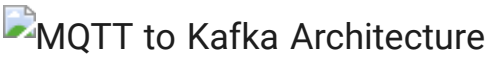
A 3-Minute Guide on Building Connected Vehicle Streaming Data Pipelines with MQTT and Kafka

In this section, we will simulate vehicle devices and their dynamic Telematics data, connect them to an **MQTT Broker**, and then send the data to Apache Kafka. We have selected **EMQX** as the MQTT Broker because it comes with a built-in Kafka data integration that simplifies the process.

Prerequisites

- Git
- Docker Engine: v20.10+
- Docker Compose: v2.20+

How It Works



MQTT to Kafka Architecture

This is a simple and effective architecture that avoids complex components. It utilizes the following 3 key components:

Component Name	Version	Description
MQTTX CLI	1.9.3+	A command line tool to generate simulated vehicle and test data.
EMQX Enterprise	5.0.4+	MQTT broker used for message exchange between vehicles and the Kafka system.
Kafka	2.8.0+	Apache Kafka serves as a distributed streaming platform for ingesting, storing, and processing vehicle data.

In addition to the basic components, EMQX provides comprehensive observability capabilities. You can use the following components to monitor EMQX metrics and load when the system is running:

Component Name	Version	Description
EMQX Exporter	0.1	Prometheus exporter for EMQX
Prometheus	v2.44.0	Open-source systems monitoring and alerting toolkit.
Grafana	9.5.1+	Visualization platform utilized to display and analyze the collected data.

Now that you have understood the basic architecture of this project, let's get the vehicle started!

Clone the Project Locally

Clone the **emqx/mqtt-to-kafka** repository locally, and initialize the submodule to enable the EMQX Exporter (optional):

```
git clone https://github.com/emqx/mqtt-to-kafka
cd mqtt-to-kafka

# Optional
git submodule init
git submodule update
```

The codebase consists of 3 parts:

- The `emqx` folder contains EMQX-Kafka integration configurations to create rules and data bridges when launching EMQX automatically.
- The `emqx-exporter`, `prometheus` and `grafana-provisioning` folders include observability configurations for EMQX.
- The `docker-compose.yml` orchestrates multiple components to launch the project with one click.

Start MQTTX CLI, EMQX, and Kafka

Please make sure you have installed the **Docker**, and then run Docker Compose in the background to start the demo:

```
docker-compose up -d
```

Now, 10 Tesla vehicles simulated by MQTTX CLI will connect to EMQX and report their status to the topic `mqttx/simulate/tesla/{clientid}` at a frequency of once

per second.

In fact, EMQX will create a rule to ingest messages from Tesla. You can also modify this rule later to add custom processing using EMQX's **built-in SQL functions**:

```
SELECT
  payload
FROM
  "mqttx/simulate/#"
```

EMQX also creates a data bridge to produce vehicle data to Kafka with the following key configurations:

- Publish messages to the `my-vehicles` topic in Kafka
- Use each vehicle's client ID as the message key
- Use the message publish time as the message timestamp


Kafka Config

Subscribe to Vehicle Data From EMQX

> *This step has no special meaning for the demo, just to check if the MQTTX CLI and EMQX are working.*

Docker Compose has included a subscriber to print all vehicle data. You can view the data with this command:

```
$ docker logs -f mqttx
[8/4/2023] [8:56:41 AM] › topic: mqttx/simulate/tesla/mqttx_063105a2
payload: {"car_id":"WLHK53W2GSL511787","display_name":"Roslyn's Tesla",
```

To subscribe and receive the data with any MQTT client:

```
mqttx sub -t mqttx/simulate/tesla/+
```

Subscribe to Vehicle Data From Kafka

Assuming everything is functioning properly, EMQX is streaming data from the vehicle into the `my-vehicles` topic of Kafka in real-time. You can consume data from Kafka with the following command:

```
docker exec -it kafka \
  kafka-console-consumer.sh \
  --topic my-vehicles \
  --from-beginning \
  --bootstrap-server localhost:9092
```

You will receive JSON data similar to this:

```
{"vin":"EDF226K7LZTZ51222","speed":39,"odometer":68234,"soc":87,"
```

The data is inspired by **TeslaMate**, a powerful self-hosted Tesla data logger, and you can check the MQTTX CLI **script** to see how the data is generated.

View EMQX Metrics (Optional)

If you have enabled EMQX Exporter in step 1, it will faithfully collect all EMQX metrics including client connections, message rate, rule executions, etc. It provides valuable insights into the system.

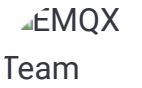
To view EMQX metrics in the Grafana dashboard, open `http://localhost:3000` in your browser, log in with username `admin` and password `public` .

Conclusion

By leveraging EMQX as an MQTT broker and utilizing EMQX Data Integration to stream data to Kafka, we have created an end-to-end solution for accumulating and processing streaming data. Next, you can directly integrate applications into Kafka to consume vehicle data and decouple them. You can also leverage Kafka Streams to perform real-time stream processing on automotive data, conduct statistical analysis and anomaly detection. The results can be output to other systems via Kafka Connect.

The MQTT + Kafka architecture is well-suited for use cases that require real-time data collecting, scalability, reliability, and integration capabilities in IoT. It enables a seamless flow of data, efficient communication, and innovative use cases such as applications and services for the connected vehicle ecosystem.

Article By



EMQX Team

The EMQX Team specializes in developing MQTT messaging infrastructure and solutions, including the most scalable open-source MQTT broker on the market – EMQX. The team also coordinates excellence on the enterprise MQTT platform at scale – EMQX Enterprise.

Related Posts

Sep 13, 2023 Jaylin

RabbitMQ vs Kafka: 5 Key…

RabbitMQ is an open-source message-…

Jul 10, 2023 EMQX Team

MQTT Performance…

In this post, we provide the…

May 21, 2024 EMQX Team

MQTT to PostgreSQL:…

In this blog post, we will demonstrate how…