WEBINAR

# HiveMQ + Kafka: The Ideal Solution for IoT MQTT Data Integration

HIVEMQ

# WELCOME

## Dominik Obermaier

*@dobermai*

*linkedin.com/in/dobermai/*

**HiveMQ CTO**

- Strong background in distributed and large scale systems architecture
- OASIS MQTT TC Member
- Author of „The Technical Foundations of IoT“
- Conference Speaker and Author
- Program committee member for German and international IoT conferences

## Magi Erber

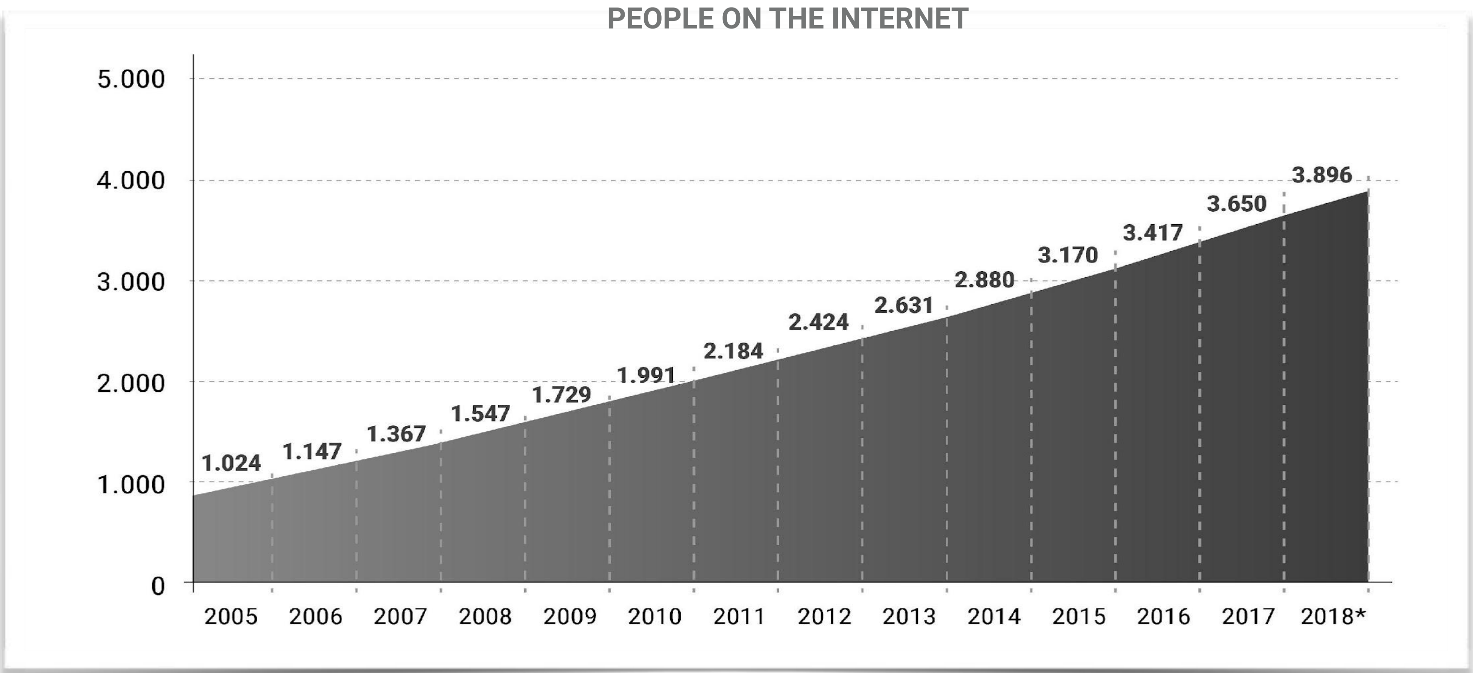
*@ErberMagi*

*linkedin.com/in/margaretha-erber/*

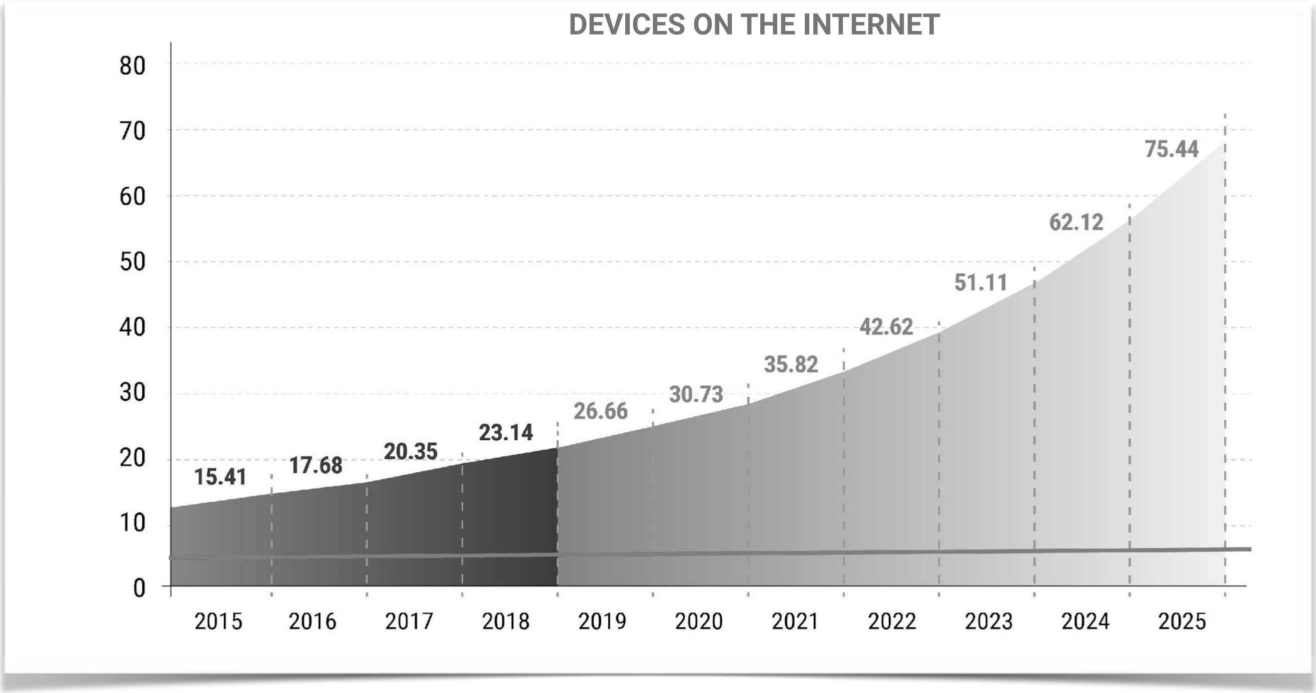
**Product Manager @HiveMQ**

- Conference Speaker
- Author
- Expert for cloud native technologies and Apache Kafka

# The Internet of Things is HUGE



# The Internet of Things is HUGE



# Millions of Devices

Customers, ARR, ...

# Technical IoT Challenges

# Challenge 1 - Scalability

- Enterprise IT infrastructure is **not suitable** for IoT

- **Massive scalability required** for millions of devices

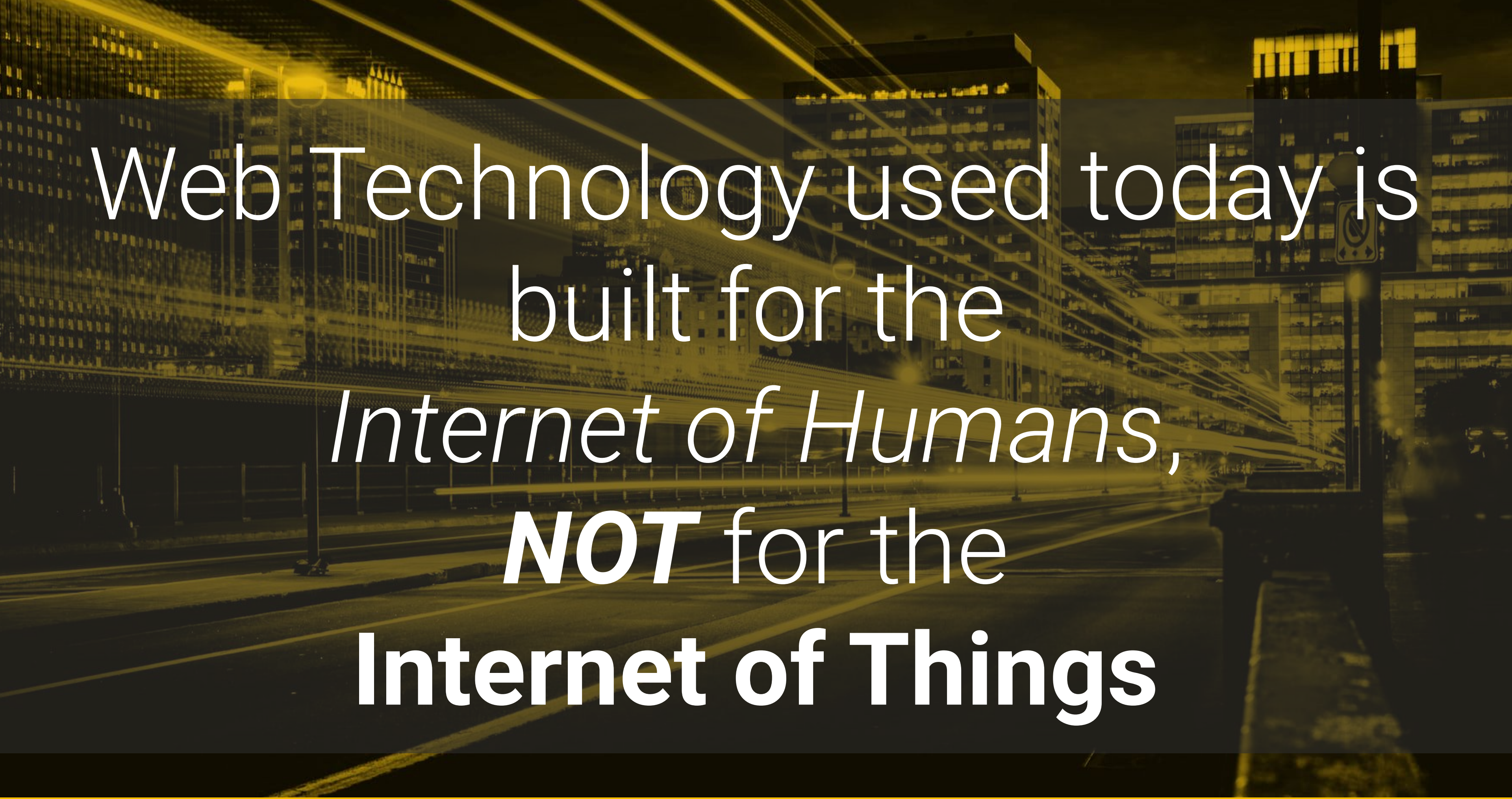
# Challenge 2 - Instant Data Delivery required

- **End customers are used to *instant* user experiences** like instant messaging with WhatsApp
- **Critical systems need reliable and *instant* data transfer** like manufacturing systems
- **Bidirectional communication** required

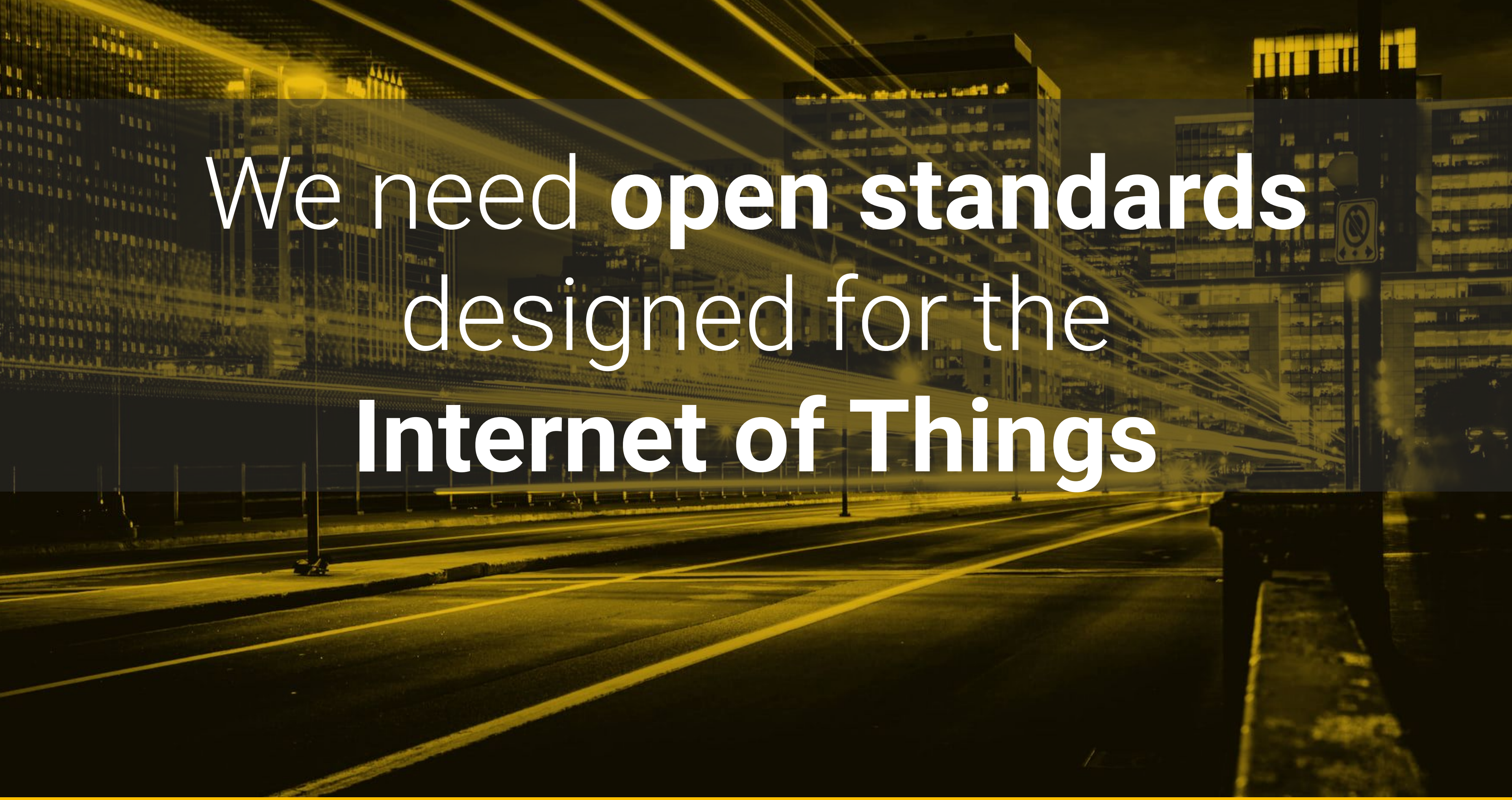
# Challenge 3 - Unreliable Networks

- **Customer experience for IoT apps and devices must be excellent even when internet connectivity is flaky**
  → Especially for moving “devices” like cars

- **Devices and apps must be easy to program and maintain, complexity should be in the cloud not on the device**
  → Cloud is easier to update than physical devices



# Web Technology used today is built for the *Internet of Humans*, ***NOT*** for the **Internet of Things**



# We need **open standards** designed for the **Internet of Things**

MQTT

# What Is MQTT?

- (I)IoT Messaging Protocol
- Created for extreme scale and instant data exchange
- Publish/Subscribe based architecture
- Easy on the device side, pushes all implementation complexity to the server
- Built for machines and constrained devices (binary, data agnostic)
- Designed for reliable communication over unreliable channels

# Benefits of MQTT

- Lightweight and efficient
- Bi-directional communications
- Scale to millions of things
- Reliable message delivery
- Support of unreliable networks
- Security Enabled

# MQTT Use Cases

Connected Car

IIoT /
Industry 4.0

Logistics

Telecommunication

IoT Messaging
Middleware

# MQTT Use Cases

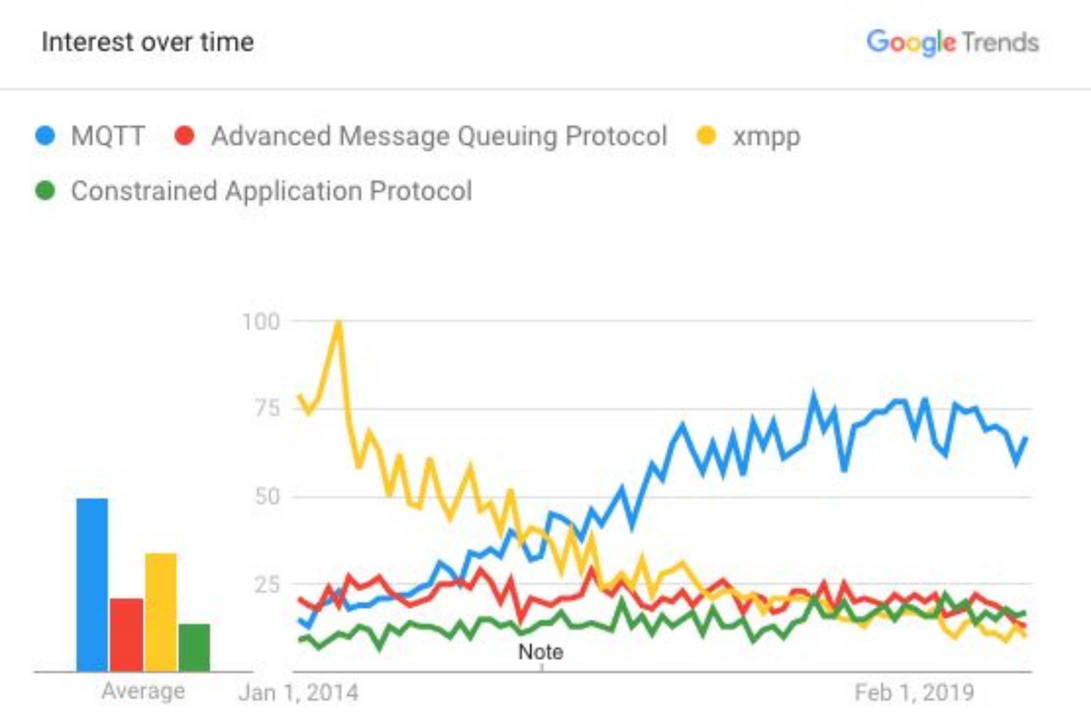
**Push Communication**

**Reliable Communication over unreliable networks**

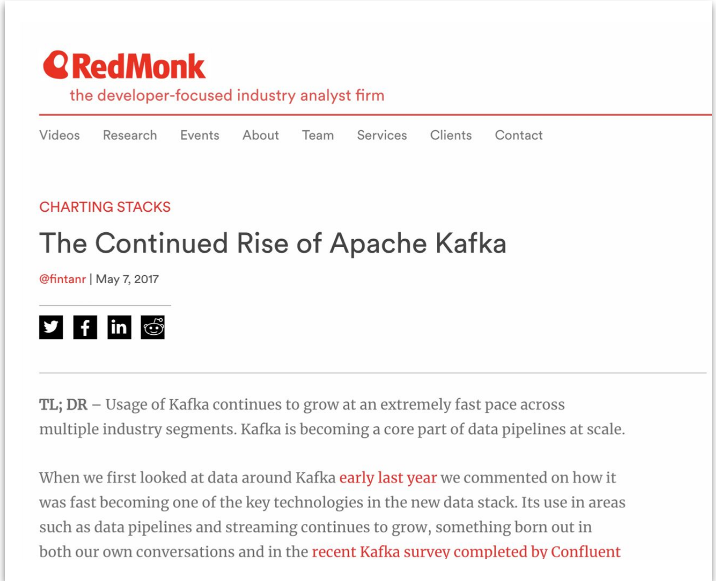
**Constrained Devices**

**Low Bandwidth and High Latency**

**Industrial Message Bus**


Interest over time
Google Trends
● MQTT
● Advanced Message Queuing Protocol
● xmpp
● Constrained Application Protocol
100
75
50
25
Note
Average
Jan 1, 2014
Feb 1, 2019

Meanwhile...



RedMonk

the developer-focused industry analyst firm

Videos Research Events About Team Services Clients Contact

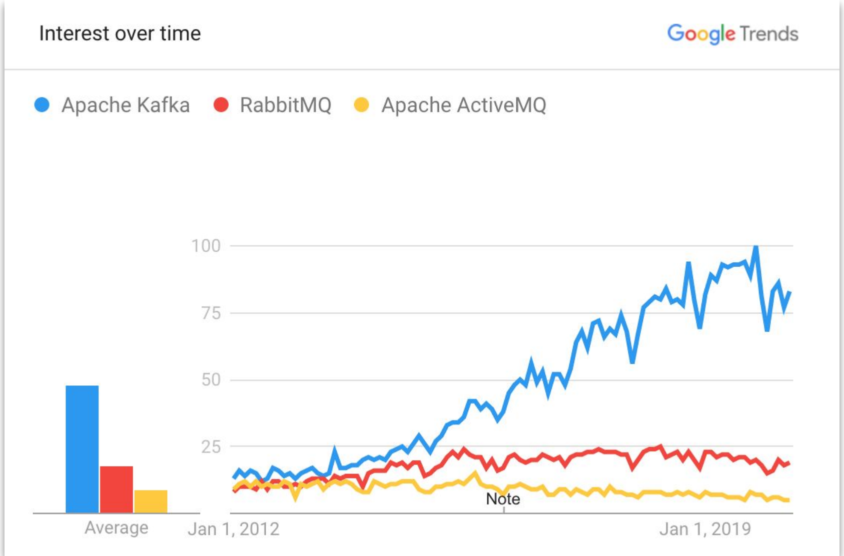
CHARTING STACKS

## The Continued Rise of Apache Kafka

@fintanr | May 7, 2017

**TL; DR** – Usage of Kafka continues to grow at an extremely fast pace across multiple industry segments. Kafka is becoming a core part of data pipelines at scale.

When we first looked at data around Kafka early last year we commented on how it was fast becoming one of the key technologies in the new data stack. Its use in areas such as data pipelines and streaming continues to grow, something born out in both our own conversations and in the recent Kafka survey completed by Confluent



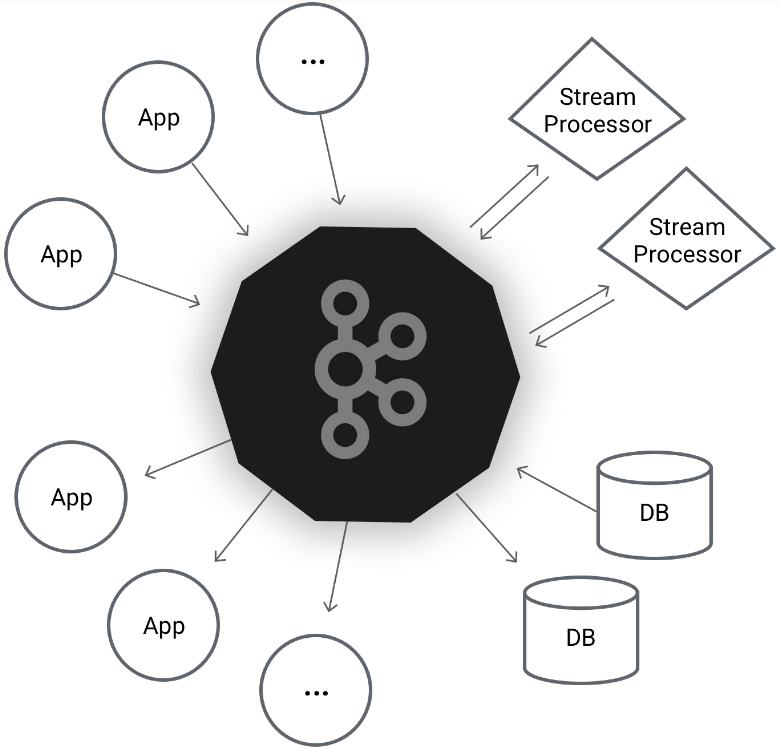
# Apache Kafka

- Distributed streaming platform
- Used by over one third of Fortune 500 companies
- Most popular Apache project on GitHub
- Central messaging and distributed stream processing application

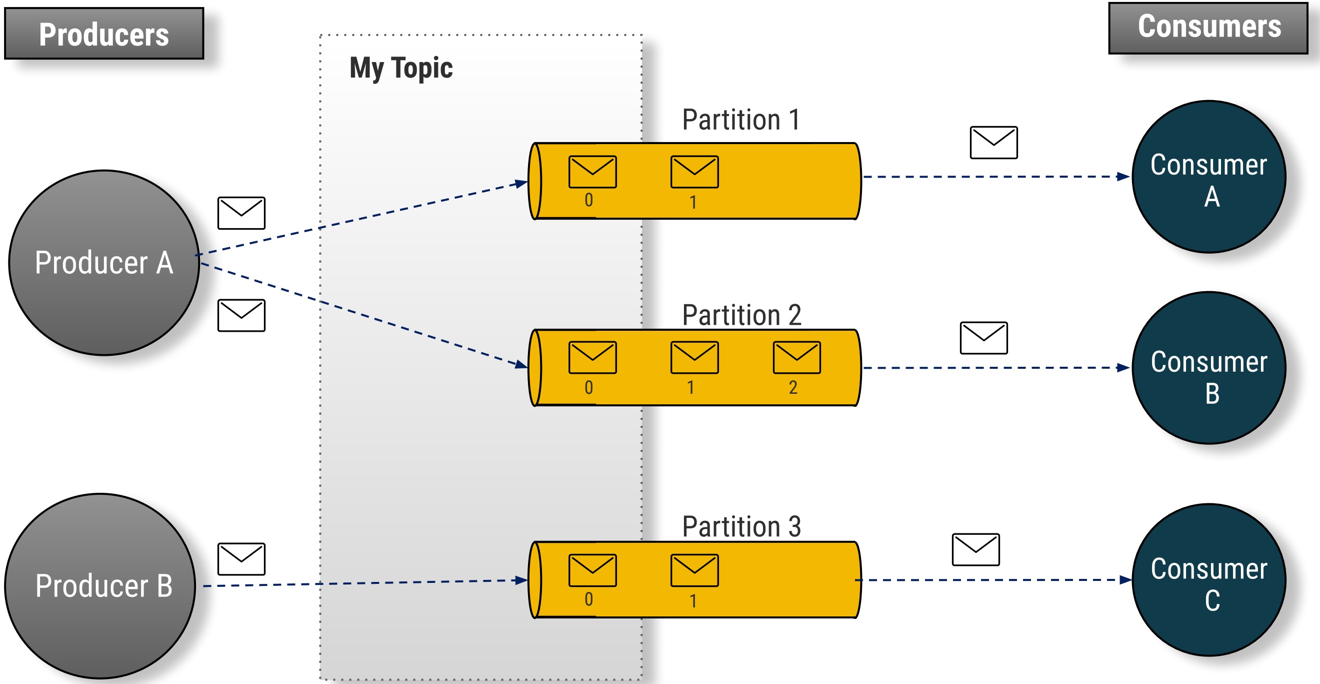
# Apache Kafka Strengths

- Optimized to stream data between systems and applications in a scalable manner
- Scale-out with multiple partitions for a topic and multiple nodes
- Perfect for inter-system communication
    - inside trusted network
    - with stable IP addresses and connections and
    - limited number of producers and consumers

# Apache Kafka



# Apache Kafka



# Apache Kafka for IoT - How does it fit in?

# IoT Reality Challenges

# Challenge 1 - Millions of Connections

☑ **IoT REALITY**

- Clients are connected over the Internet
- Load Balancers are used as first line of defense
- IP addresses of infrastructure (e.g. Kafka nodes) not exposed to the public Internet
- Load Balancers effectively act as proxy

Kafka Clients need to address Kafka brokers directly, which is not possible with L4 load balancers

# Challenge 2 - Scalability and Topics

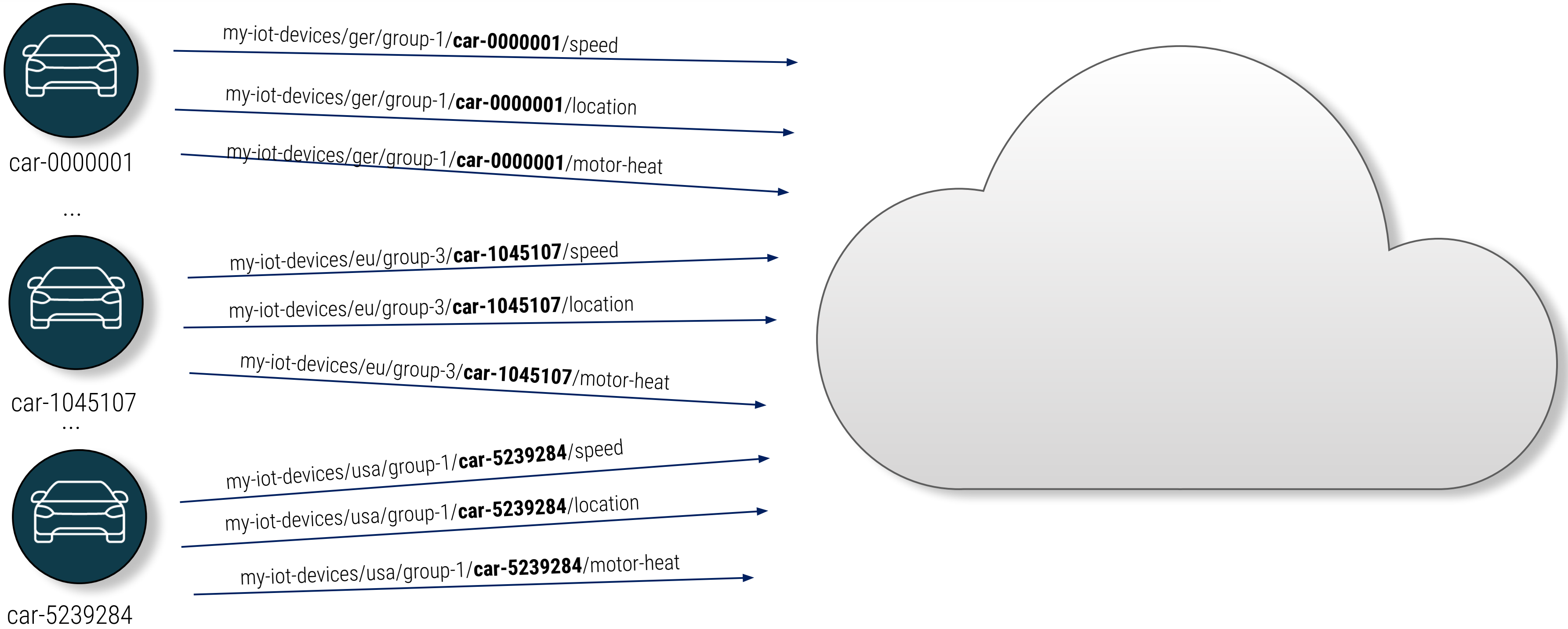
✓ **IoT REALITY**

- IoT devices typically are segmented to use individual topics
- Individual topics very often contain data like unique device identifier
- Multiple millions of topics can be used in a single IoT scenario
- Ideal for security as it's possible to restrict devices to only produce and consume for specific topics
- Topics are usually dynamic

APACHE kafka®

- Kafka is hard to scale to multiple thousands or even millions of topics

# Challenge 2 - Scalability and Topics



car-0000001

my-iot-devices/ger/group-1/**car-0000001**/speed

my-iot-devices/ger/group-1/**car-0000001**/location

my-iot-devices/ger/group-1/**car-0000001**/motor-heat

...

car-1045107

my-iot-devices/eu/group-3/**car-1045107**/speed

my-iot-devices/eu/group-3/**car-1045107**/location

my-iot-devices/eu/group-3/**car-1045107**/motor-heat

...

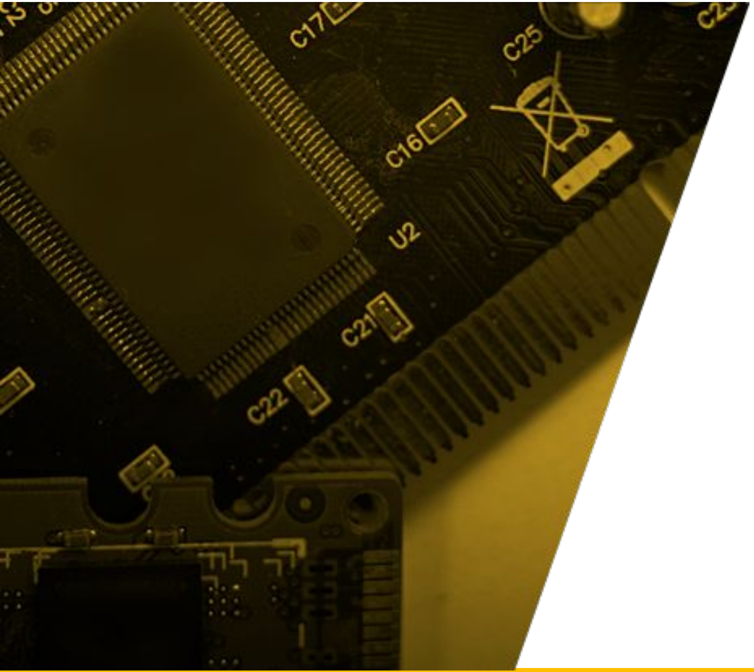
car-5239284

my-iot-devices/usa/group-1/**car-5239284**/speed

my-iot-devices/usa/group-1/**car-5239284**/location

my-iot-devices/usa/group-1/**car-5239284**/motor-heat

# Challenge 3 - Constraint Devices



## ☑ IoT REALITY

- IoT devices are typically very constrained (computing power and memory)
- Device programmer need very simple APIs AND full flexibility when it comes to library behavior
- Single IoT devices typically don't require lot of throughput
- Important to limit and understand the number of TCP connections, especially over the Internet. Very often only one TCP connection to the backend desired

APACHE kafka®

- Kafka Clients are reasonable complex by design (e.g. use multiple TCP connections)
- Libraries optimized for throughput
- APIs for Kafka libraries are simple to use but the behavior sometimes isn't configurable easily (e.g. async send() method can block)

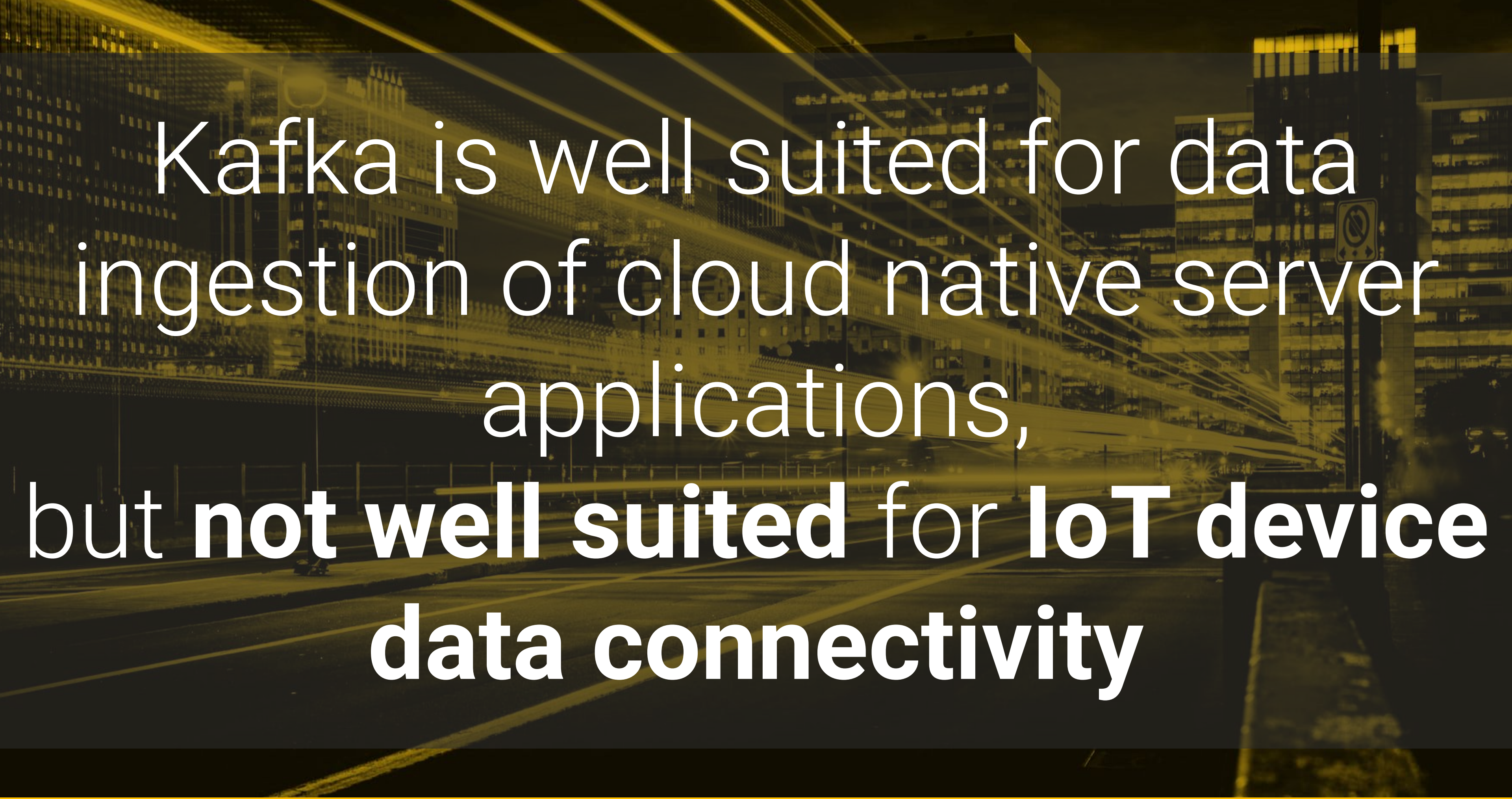
# Challenge 4 - Unreliable Network

☑ **IoT REALITY**

- Features like on/off notifications are often required
- Unreliable networks require lightweight keep-alive mechanisms for producers and consumers (half-open connections)
- Device communication over the Internet requires minimal communication overhead

APACHE kafka®

- No on/off notification mechanism
- No Keep-Alive mechanism individual TCP connections for producers
- Kafka Protocol for producers rather heavyweight over the Internet (lots of communication)
  - ○



Kafka is well suited for data ingestion of cloud native server applications,

but **not well suited** for **IoT device data connectivity**

# Challenges for Apache Kafka in IoT

- Kafka brokers need to be addressed directly by the clients
- Kafka does not support large amounts of topics
- Kafka clients are reasonably complex and resource intensive compared to client libraries for IoT protocols
- Kafka clients require a stable TCP connection for best results
- It’s unusual (and very often not even possible at all) to have tens of thousands or even millions of clients connected to a single Kafka cluster
- Kafka is missing some key IoT features

# How to use the best of both worlds?

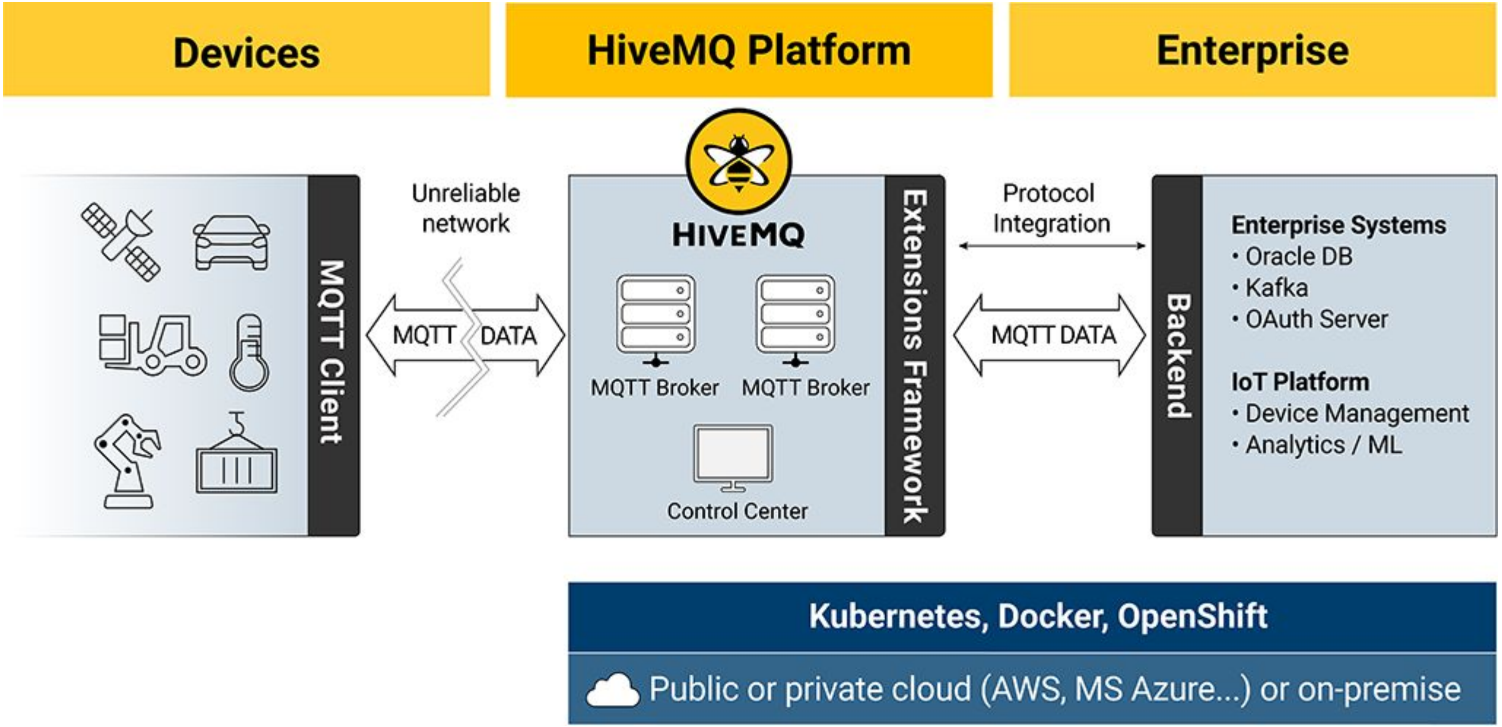
# A Love Story Made in Heaven

MQTT + APACHE kafka® = ❤️

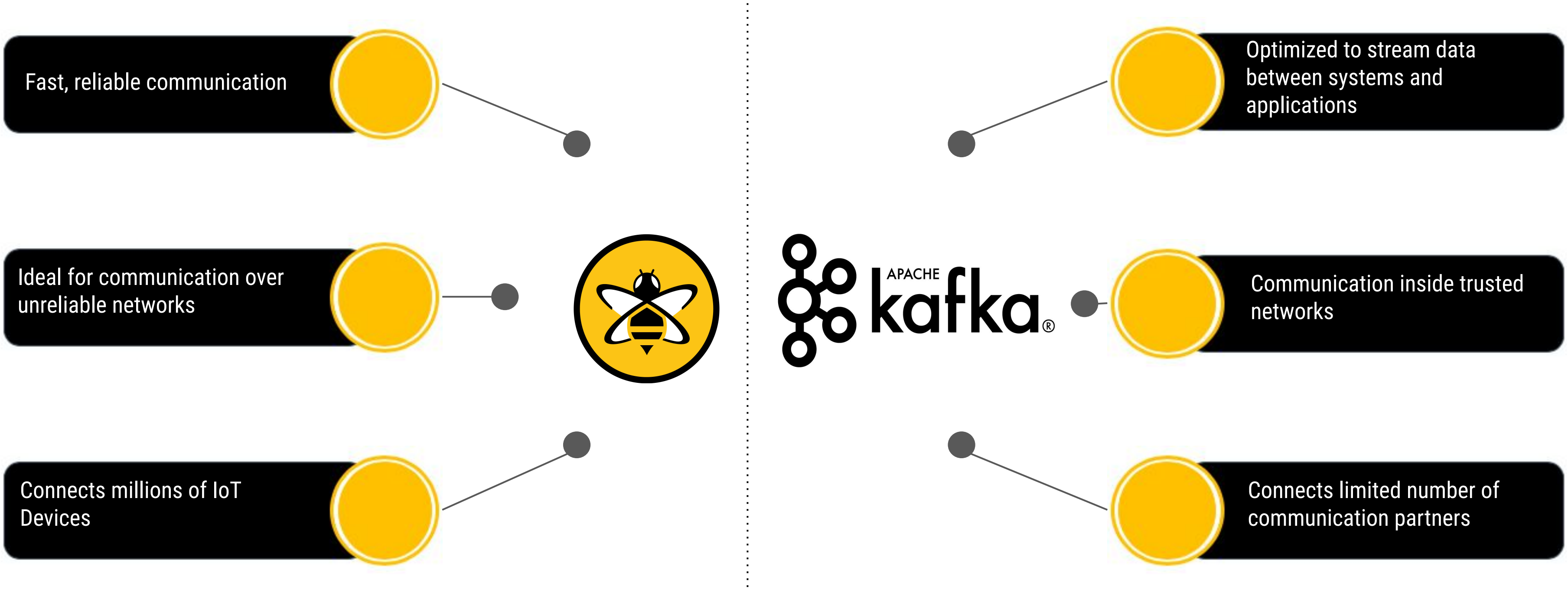
# HiveMQ - Enterprise MQTT Broker

- Connectivity and Messaging Platform
- Based on standard IoT protocol (MQTT)
- Scales to more than 10 million always-on devices
- Allow multi-cloud and Enterprise software integration

# HiveMQ - Enterprise MQTT Broker



# HiveMQ and Kafka



Fast, reliable communication

Ideal for communication over unreliable networks

Connects millions of IoT Devices

Optimized to stream data between systems and applications

Communication inside trusted networks

Connects limited number of communication partners

# HIVEMQ ENTERPRISE EXTENSION FOR KAFKA

# HiveMQ Enterprise Extension for Kafka

HIVEMQ
EXTENSION
FOR KAFKA

- Native implementation of Kafka protocol
- End to-end persistent messaging guarantees
- Bi-directional communication
- High Scalability and resilience
- Support of Local Schema Registry (Avro, JSON)
- Support of Confluent Schema Registry (Avro)
- Stream to multiple Kafka instances

# Support of Different Kafka Distributions

- Apache Kafka (Open Source)
- Confluent Platform
- Confluent Cloud
- AWS MSK
- Multiple Kafka deployments simultaneously

# High Scalability and Resilience

Scales elastically with the MQTT broker

No message loss: Queues messages, if Kafka cluster is temporarily not available

Extreme throughput. Can write hundreds of thousands of MQTT messages per second to Kafka

# Bidirectional Communication

# Bidirectional Communication

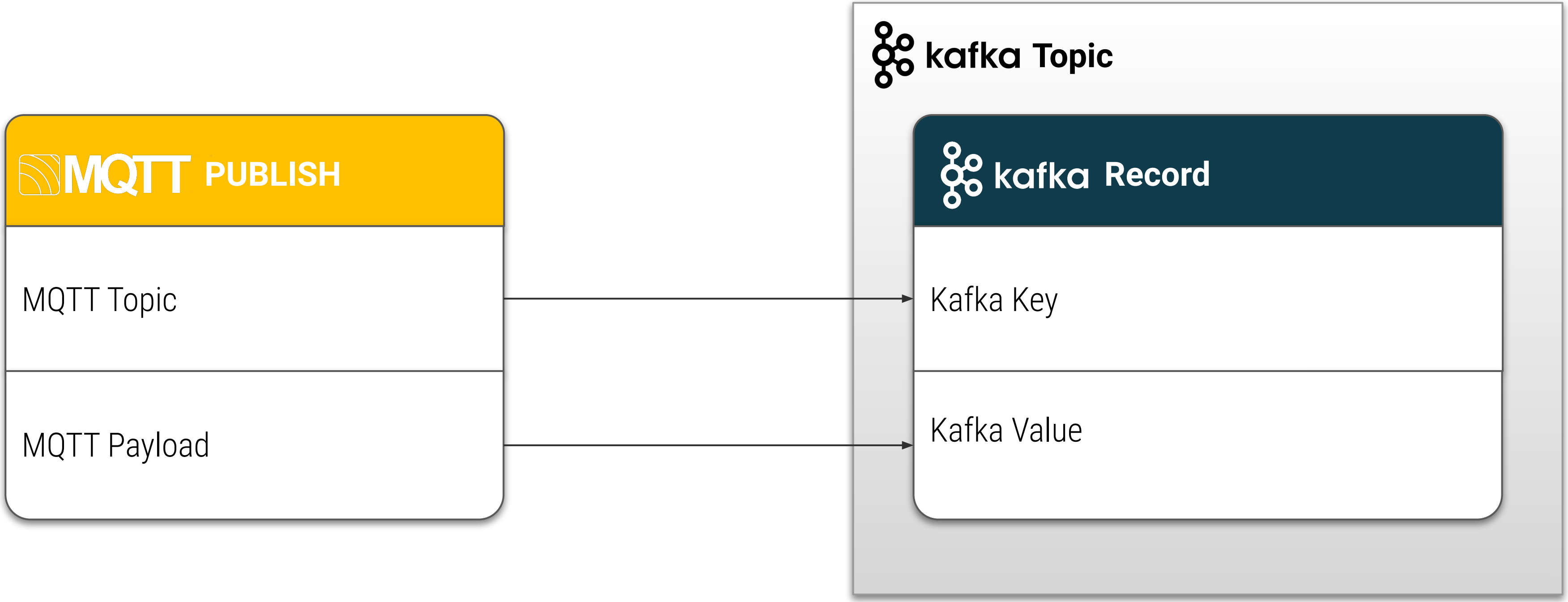
MQTT → APACHE kafka®

- **MQTT to Kafka Topic Mapping**
  - One to One Topic Mapping
  - Aggregating MQTT Topics to Fireho[illegible] Topic (using topic filters)

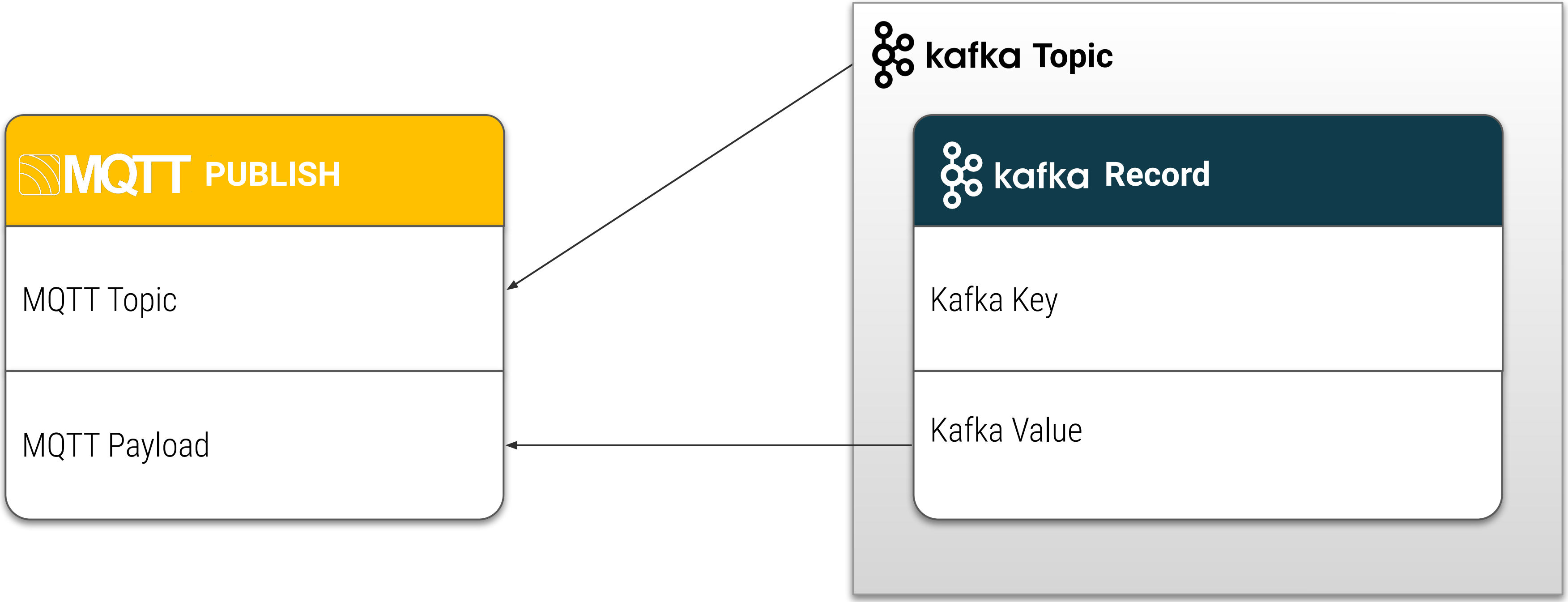
APACHE kafka® → MQTT

- **Kafka to MQTT Topic Mapping**
  - One to One Topic Mapping
  - One (M[illegible]) to [illegible] Topic Mapping
  - Message transfo[illegible]tion (using parts of Kafka message)

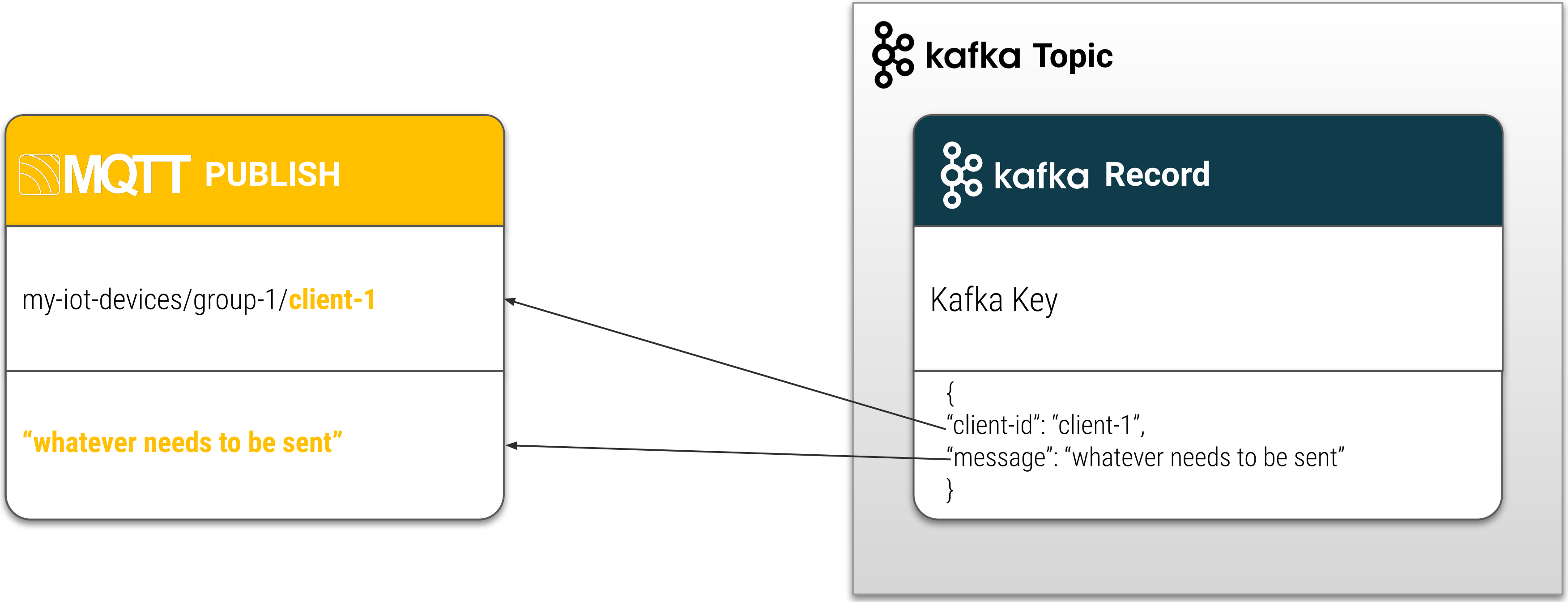
# Bidirectional Communication



# Bidirectional Communication



# Message Transformation



MQTT PUBLISH

my-iot-devices/group-1/**client-1**

**“whatever needs to be sent”**

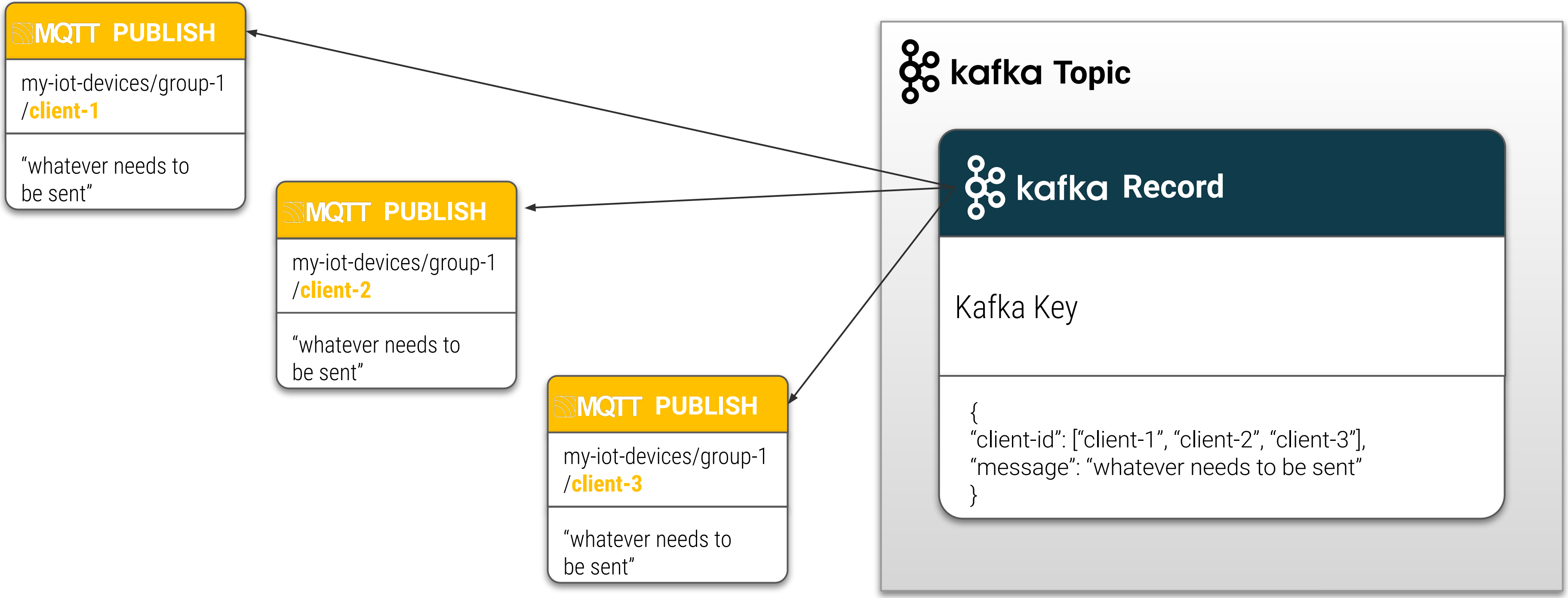
kafka Topic

kafka Record

Kafka Key

```
{
“client-id”: “client-1”,
“message”: “whatever needs to be sent”
}
```

# Message Multicasting



MQTT PUBLISH

my-iot-devices/group-1/**client-1**

“whatever needs to be sent”

MQTT PUBLISH

my-iot-devices/group-1/**client-2**

“whatever needs to be sent”

MQTT PUBLISH

my-iot-devices/group-1/**client-3**

“whatever needs to be sent”

kafka Topic

kafka Record

Kafka Key

```
{
“client-id”: [“client-1”, “client-2”, “client-3”],
“message”: “whatever needs to be sent”
}
```

# Support of Schema Registries

## Apache Kafka

- Kafka does not care about message formats
- No verification of message correctness
- Systems are evolving and message formats will change

⇩

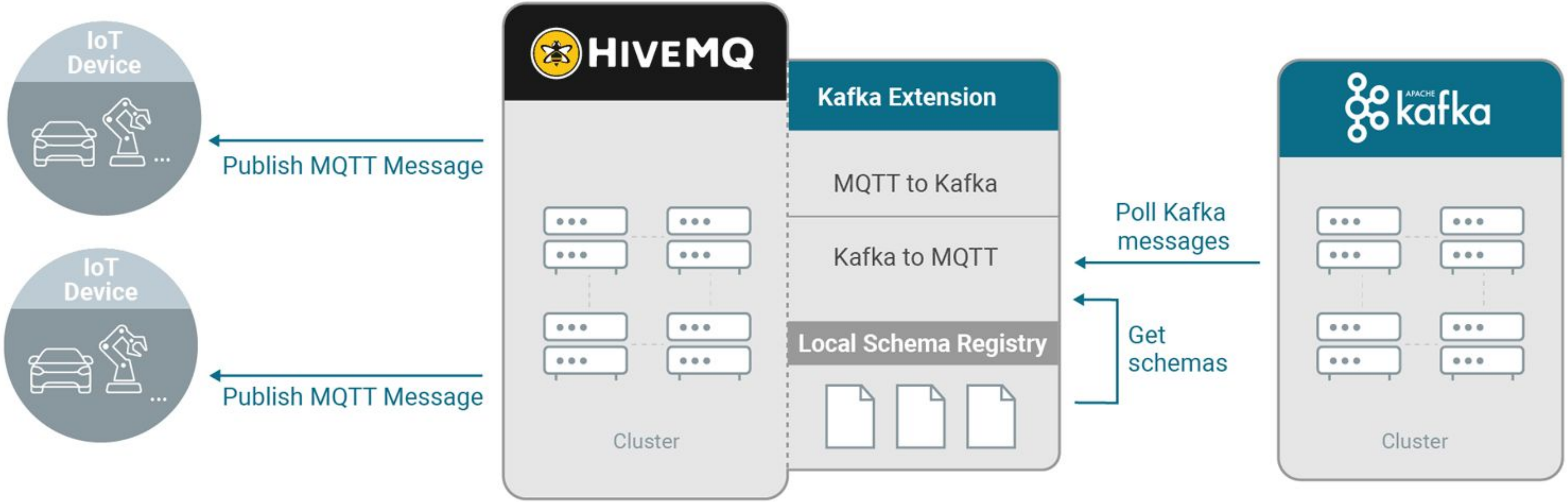
**Schema Registries**

- Guarantee the correct functionality of messaging
- Message validation
- Schema evolution

HIVEMQ ENTERPRISE EXTENSION FOR KAFKA

- Support of Local Schema Registry
  - JSON and Avro messages
  - Message validation

# Local Schema Registry



# Support of Schema Registries

### Apache Kafka

- Kafka does not care about message formats
- No verification of message correctness
- Systems are evolving and message formats will change

⇩

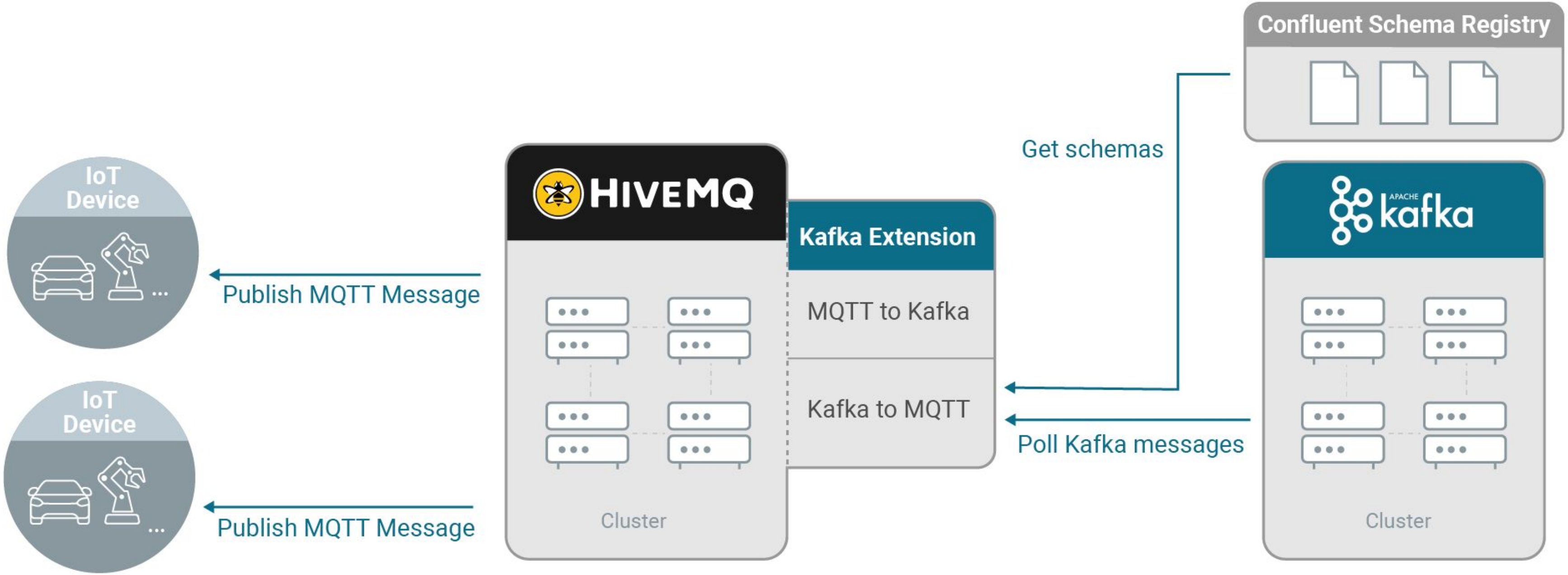
### Schema Registries

- Guarantee the correct functionality of messaging
- Message validation
- Schema evolution

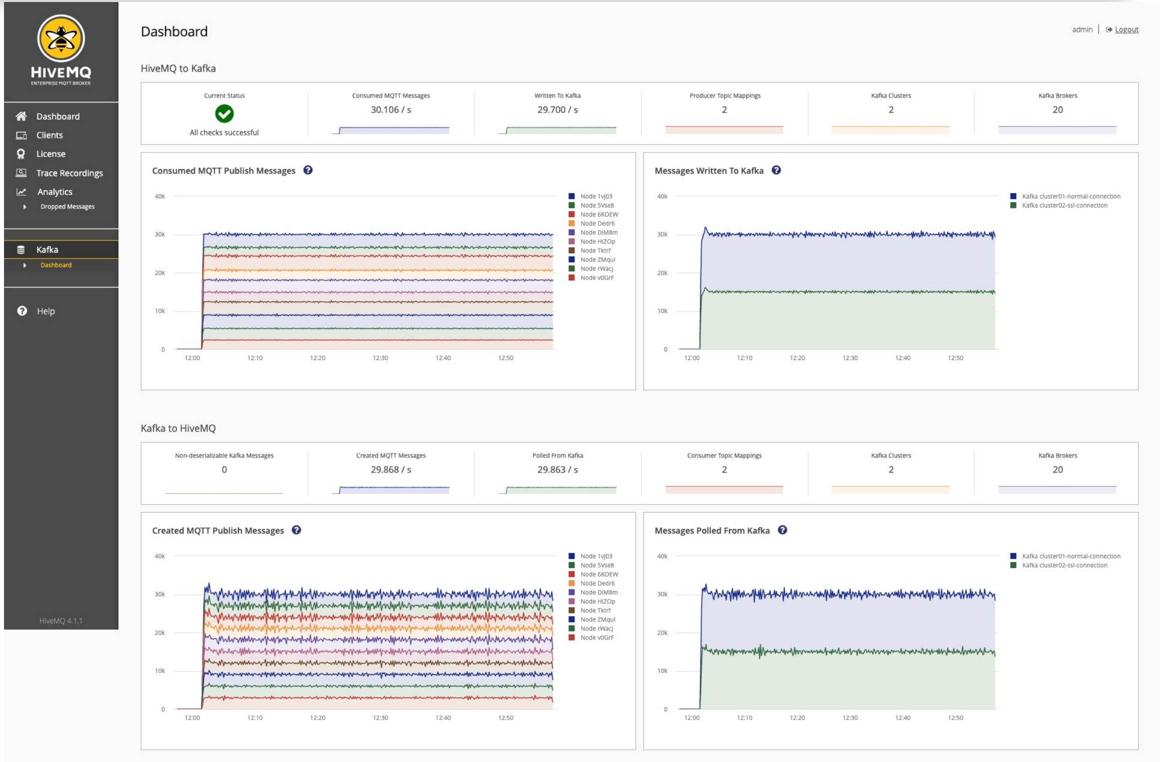
HIVEMQ ENTERPRISE EXTENSION FOR KAFKA

- Support of Local Schema Registry
    - JSON and Avro messages
    - Message validation
- Support of Confluent Schema Registry
    - Avro messages
    - Message validation
    - Schema evolution

# Confluent Schema Registry



# Observability of Communication



Dashboard

HiveMQ to Kafka

Current Status: All checks successful

Consumed MQTT Messages: 30.106 / s

Written To Kafka: 29.700 / s

Producer Topic Mappings: 2

Kafka Clusters: 2

Kafka Brokers: 20

Consumed MQTT Publish Messages

Messages Written To Kafka

40k
30k
20k
10k
0
12:00
12:10
12:20
12:30
12:40
12:50
Kafka cluster01-normal-connection
Kafka cluster02-ssl-connection

Kafka to HiveMQ

Non-deserializable Kafka Messages: 0

Created MQTT Messages: 29.868 / s

Polled From Kafka: 29.863 / s

Consumer Topic Mappings: 2

Kafka Clusters: 2

Kafka Brokers: 20

Created MQTT Publish Messages

Messages Polled From Kafka

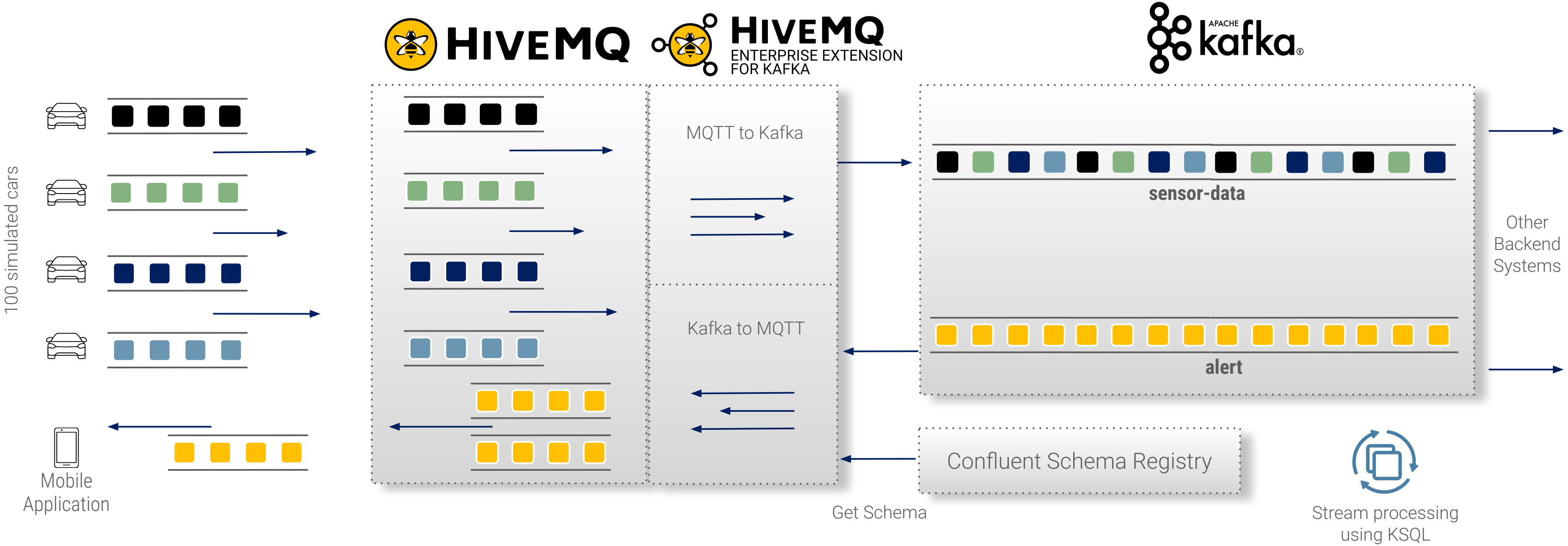
40k
30k
20k
10k
0
12:00
12:10
12:20
12:30
12:40
12:50
Kafka cluster01-normal-connection
Kafka cluster02-ssl-connection

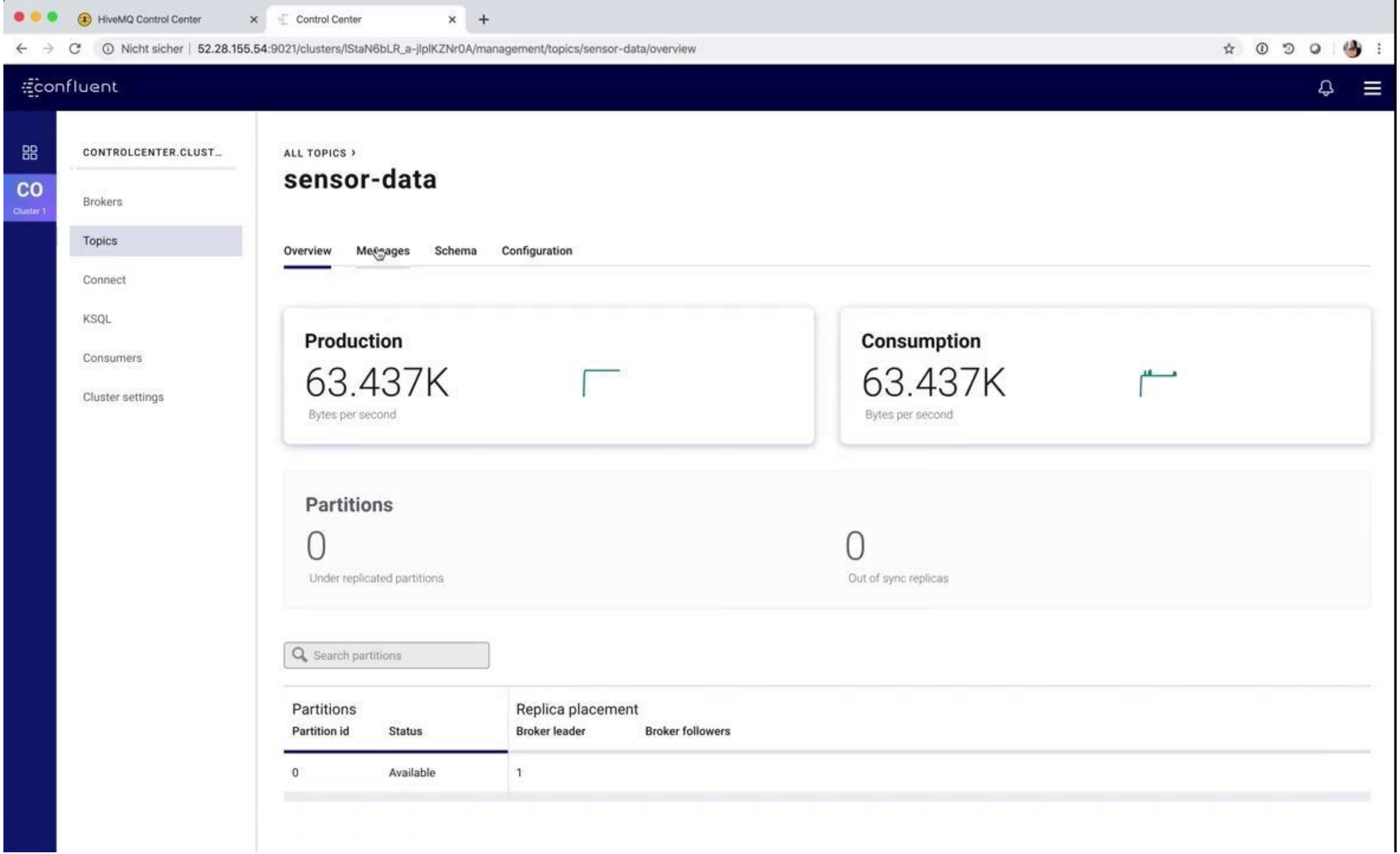
# HIVEMQ ENTERPRISE EXTENSION FOR KAFKA

- **Connect millions of IoT devices**, **over unreliable networks**, to seamlessly send data to one or multiple Kafka clusters
- **Route IoT device data to multiple Kafka clusters** allowing for a single IoT platform to support different types of devices and forward device data to different back-end applications
- **Poll information from one or multiple Kafka clusters** and distribute this information to millions of IoT devices
- Support of all **MQTT 3.1.1 and MQTT 5 features**
- **Map millions of MQTT topics** to a limited number of Kafka topics using MQTT wildcards
- **Enable end to-end persistent messaging guarantees** from device to Kafka so no messages are lost
- **Monitor MQTT messages** written to Kafka using the HiveMQ Control Center

# Demo

# HiveMQ Enterprise Extension for Kafka - Demo





sensor-data

Overview | Messages | Schema | Configuration

**Production**
63.437K
Bytes per second

**Consumption**
63.437K
Bytes per second

**Partitions**
0
Under replicated partitions
0
Out of sync replicas

Search partitions

| Partitions | | Replica placement | |
|---|---|---|---|
| Partition id | Status | Broker leader | Broker followers |
| 0 | Available | 1 | |

# Resources

Get Started with MQTT

MQTT  Essentials Series

MQTT

MQTT at OASIS

HIVEMQ

Evaluate HiveMQ Broker

HIVEMQ
CLOUD

Try HiveMQ Cloud

# ANY QUESTIONS?

Reach out to community.hivemq.com

HIVEMQ

# THANK YOU

HIVEMQ