WEBINAR

The Four Paradigm Shifts for the Connected Car of the Future

HIVEMQ & ESR LABS

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

Daniel Himmelein

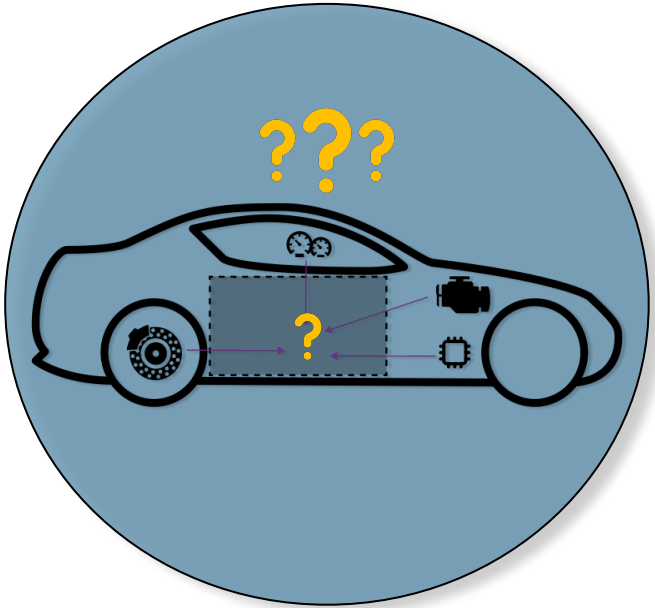
@himmele

linkedin.com/in/daniel-himmelein-a5125946/

- **Software Architect and Engineer @ ESR Labs**
- Strong background in operating systems, distributed systems and computer networks
- Works mainly on automotive series projects for German OEMs
- Creator of the Mindroid application frameworks

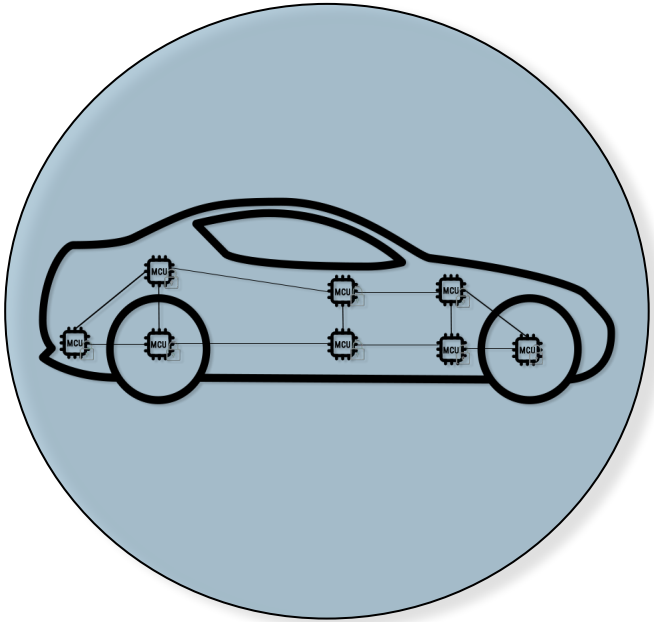
CONNECTED CAR IN THE AUTOMOTIVE INDUSTRY

Current Situation - Vehicle as a black box



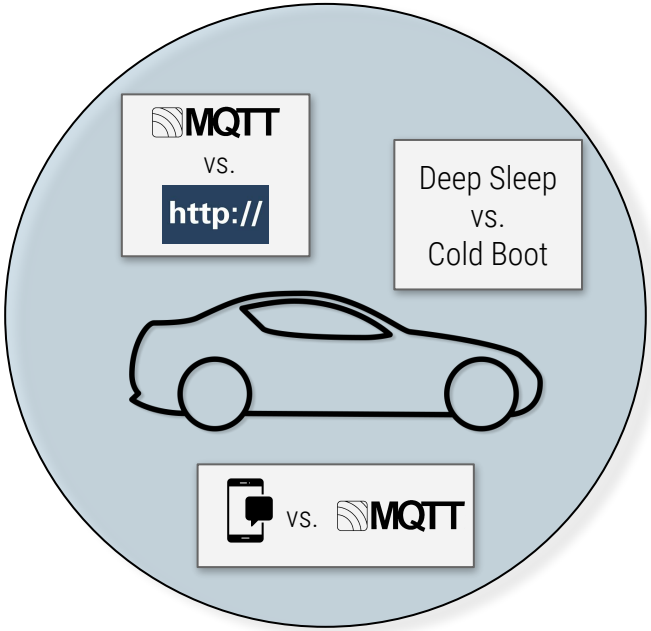
- No software platform mindset
- No or few common platforms and SDKs for application development
  - results in slow, sometimes error-prone development and testing

Current Situation - Static Communication Paths



- Static communication paths burned into edge (in-vehicle) devices and the cloud

Current Situation - Technology Gaps

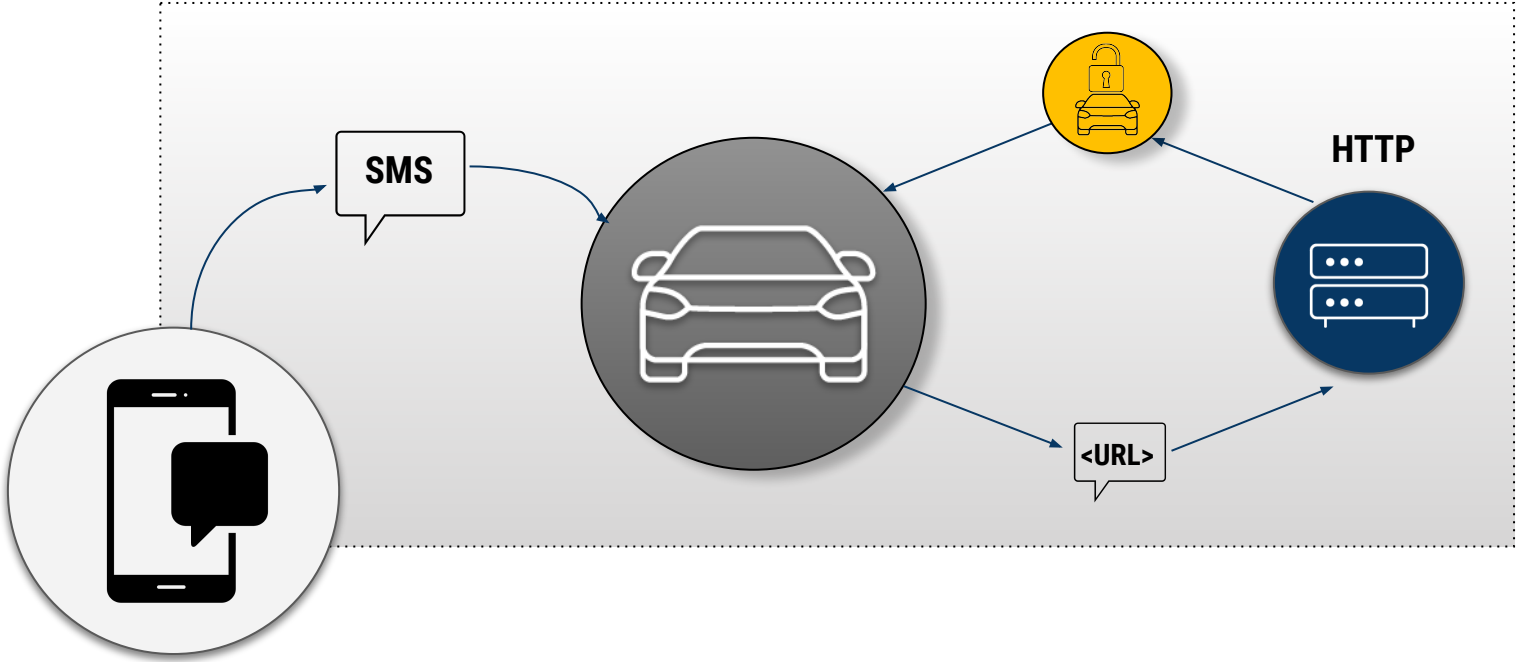


- Cold boot vs. Deep sleep
- HTTP vs. MQTT
- Wake up SMS vs. MQTT push notifications

Current Situation - Software Development

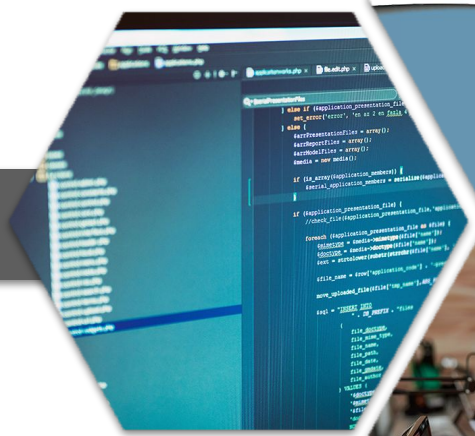
- Software development branches for vehicle models are freezed shortly after Start of Production

Current Situation - Remote Door Unlock



Four Paradigms

Publish/
Subscribe



Software Platform &
Development Kit

Software
Development Process

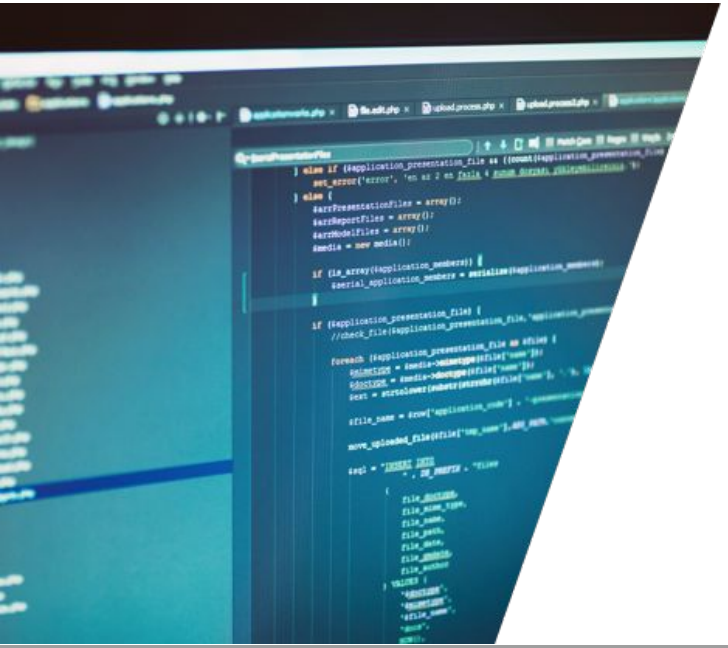
```
TOPIC: VEHICLE/LOCATION
SCHEMA: LOCATION_V1.0.0
DEFINITION:
MESSAGE LOCATION {
  UINT64 TIME = 1;
  DOUBLE LONGITUDE = 2;
  DOUBLE LATITUDE = 3;
  DOUBLE ALTITUDE = 4;
  FLOAT BEARING = 5;
  STRING PROVIDER = 6;
```

Domain Model

Paradigm Shift 1 - Software Platforms & Development Kits

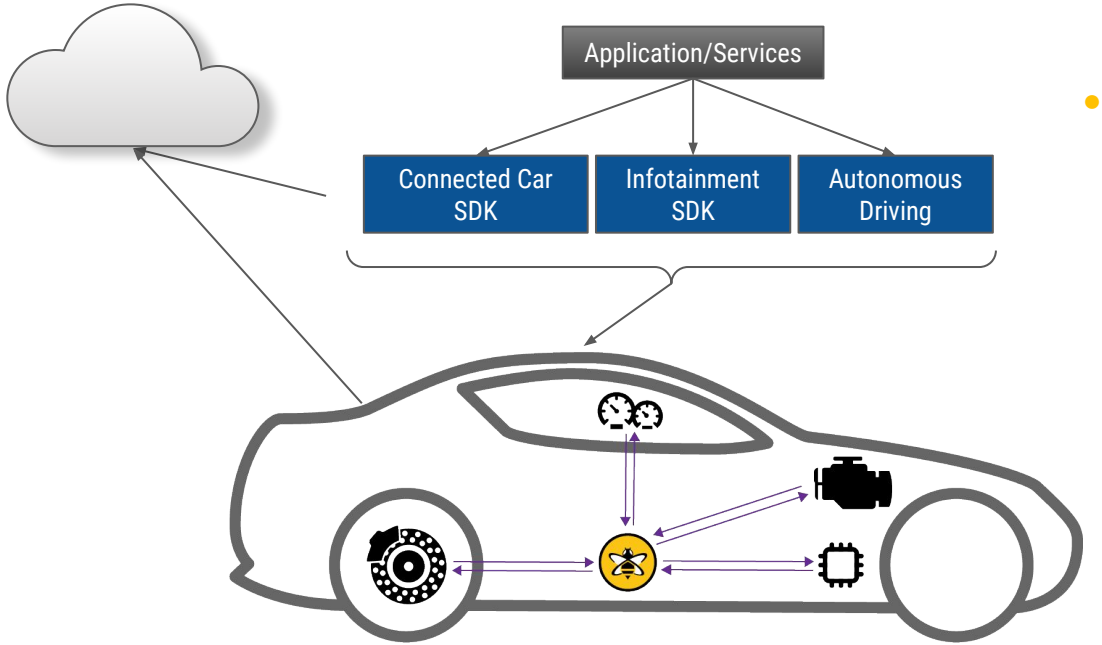
> “The key in making great and growable systems is much more to design how its modules communicate rather than what their internal properties and behaviors should be”
>
> (Alan Kay)

Paradigm Shift 1 - Software Platforms & Development Kits



- Major design goals are simplicity, modularity and security
- Application framework composes features like
  - deployment
  - logging and monitoring
  - debugging
  - testing
- Building blocks and constraints guide developers
- SDK with examples and documentation

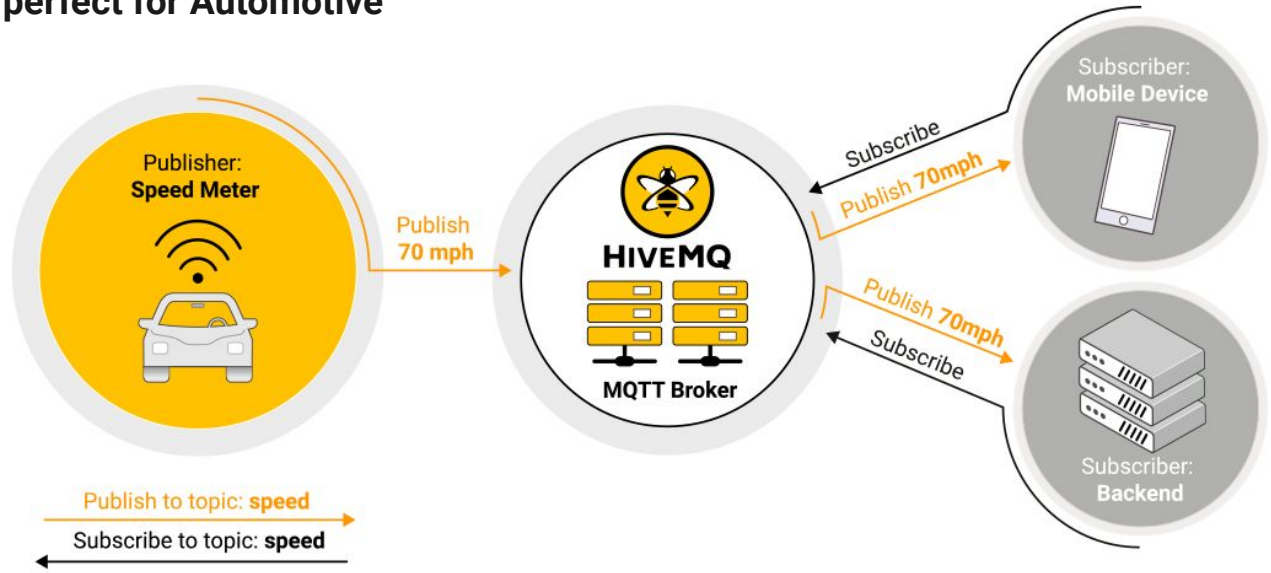
Paradigm Shift 1 - Software Platforms & Development Kits



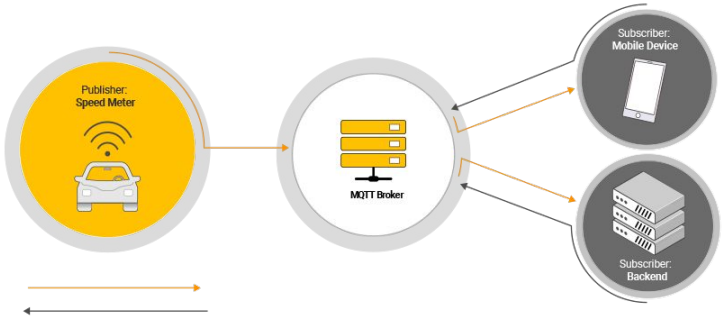
- **Vehicle as a software platform and companion SDKs for**
  - embedded body domain
  - infotainment domain
  - connected car domain
  - autonomous driving domain
  - cloud domain

Paradigm Shift 2 - Publish / Subscribe

MQTT is perfect for Automotive



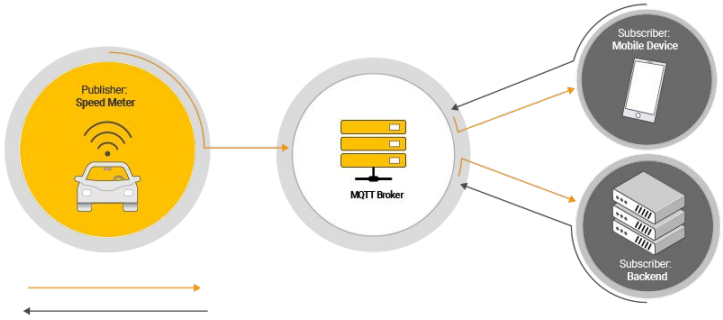
Paradigm Shift 2 - Publish / Subscribe



MQTT

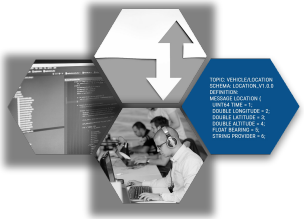
- Simple, clean and robust messaging paradigm
  - Topic-based publish/subscribe fits best for most IoT use-cases
- Build loosely coupled, flexible, scalable and easy to test (async) software systems
  - Dependency avoidance, updateability, monitorability, access control.

Paradigm Shift 2 - Publish / Subscribe



- Communication supports one-to-many, many-to-one, and many-to-many paradigms
- Data streams (data flow thinking)
- Service-oriented architectures complect distributed systems and add dependencies
- RPC for session-based communication, e.g. control shared resources like telephony module

Paradigm Shift 3 - Domain Model



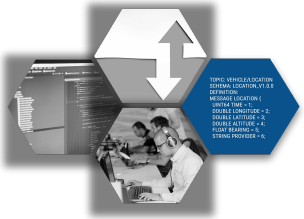
Example: Location

Topic: vehicle/location
Schema: location_v1.0.0
Definition:

```
message Location {
  uint64 time = 1;
  double longitude = 2;
  double latitude = 3;
  double altitude = 4;
  float bearing = 5;
  string provider = 6;
```

> In software engineering, a domain model is a conceptual model of the domain that incorporates both behaviour and data (Wikipedia)

Paradigm Shift 3 - Domain Model



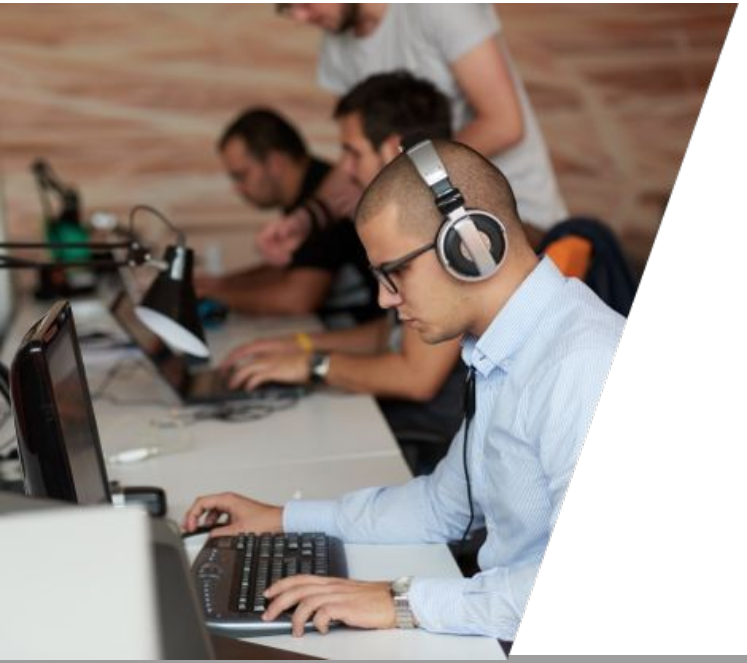
Example: Location

Topic: vehicle/location
Schema: location_v1.0.0
Definition:

```
message Location {
  uint64 time = 1;
  double longitude = 2;
  double latitude = 3;
  double altitude = 4;
  float bearing = 5;
  string provider = 6;
```

- Build unified, versioned domain model for all vehicle domains including cloud
- Use Interface Definition Language (IDL) which is available for many prog. languages, e.g. ProtoBuf
- Applications are build using application framework, the publish/subscribe mechanism and the domain model

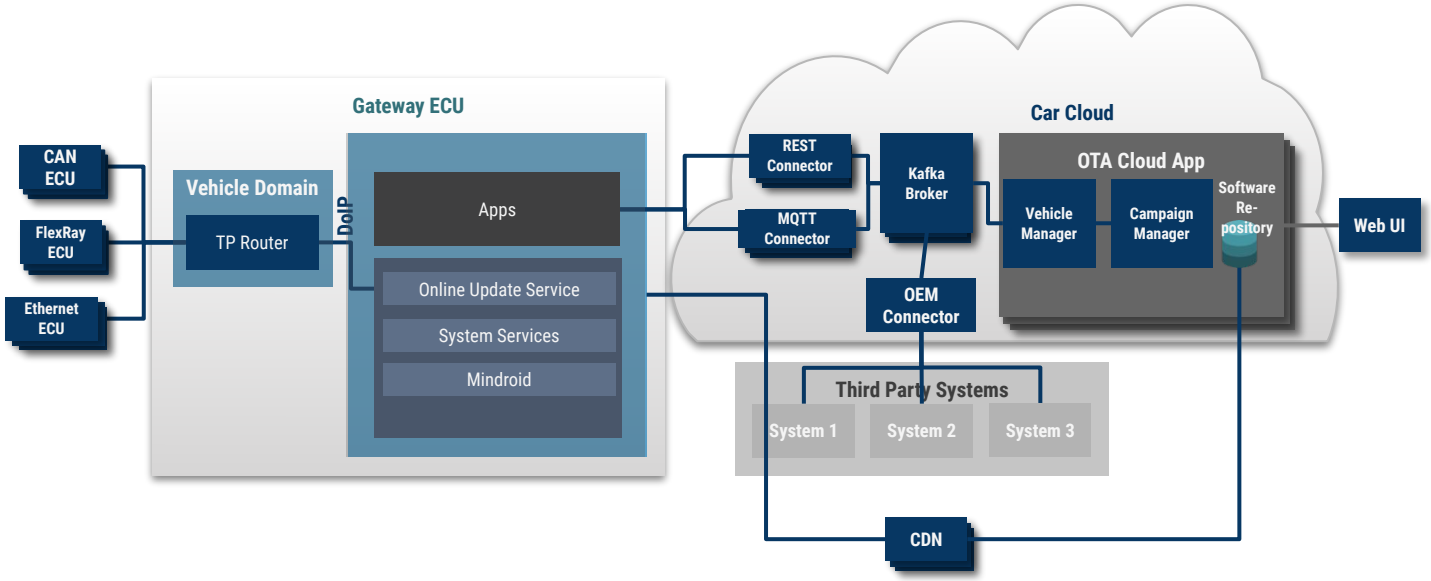
Paradigm Shift 4 - Software Development Process



- App development based on application framework and versioned SDKs
- Short software development and test cycles
- Test automation to test and support apps on various platform versions (old vehicle models)
- Small teams
- Focus on quality and time-to-market

How to Bring Paradigms to Life

Over-The-Air update full stack architecture



IoT Cloud build for simplicity and scalability:

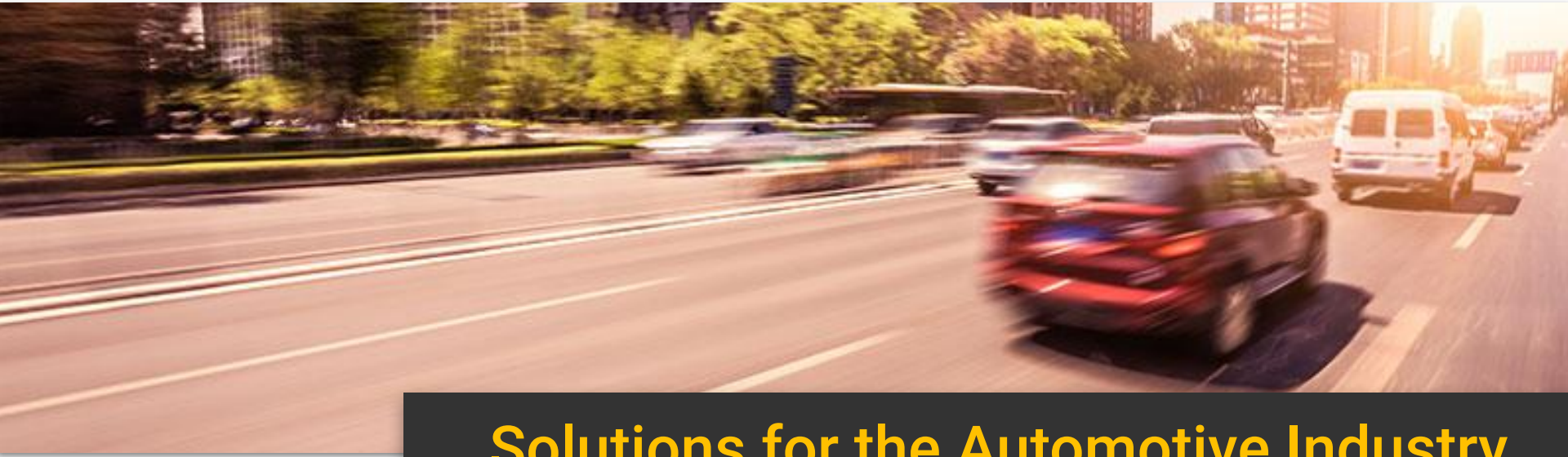
- Containerized (Docker)
- Kafka Message Broker
- MQTT Broker Cluster
- Microservices
- Cloud agnostic (AWS, Azure, etc.)
- SDK

Best practices

- Infrastructure as code
- Automated deployments
- Automated integration tests

SW platform mindset and OTA updates

1. **Software Platform and Development Kits**
   Application development based on SDKs and API levels ensures compatibility with various platform versions
2. **Publish/Subscribe Messaging**
   Topic-based publish/subscribe messaging allows for building loosely coupled applications and services
3. **Domain Model**
   The unified, versioned domain model builds the foundation for application and service interaction as well as for testing
4. **Software Development Process**
   Small teams build apps that are developed, tested and deployed continuously



Solutions for the Automotive Industry

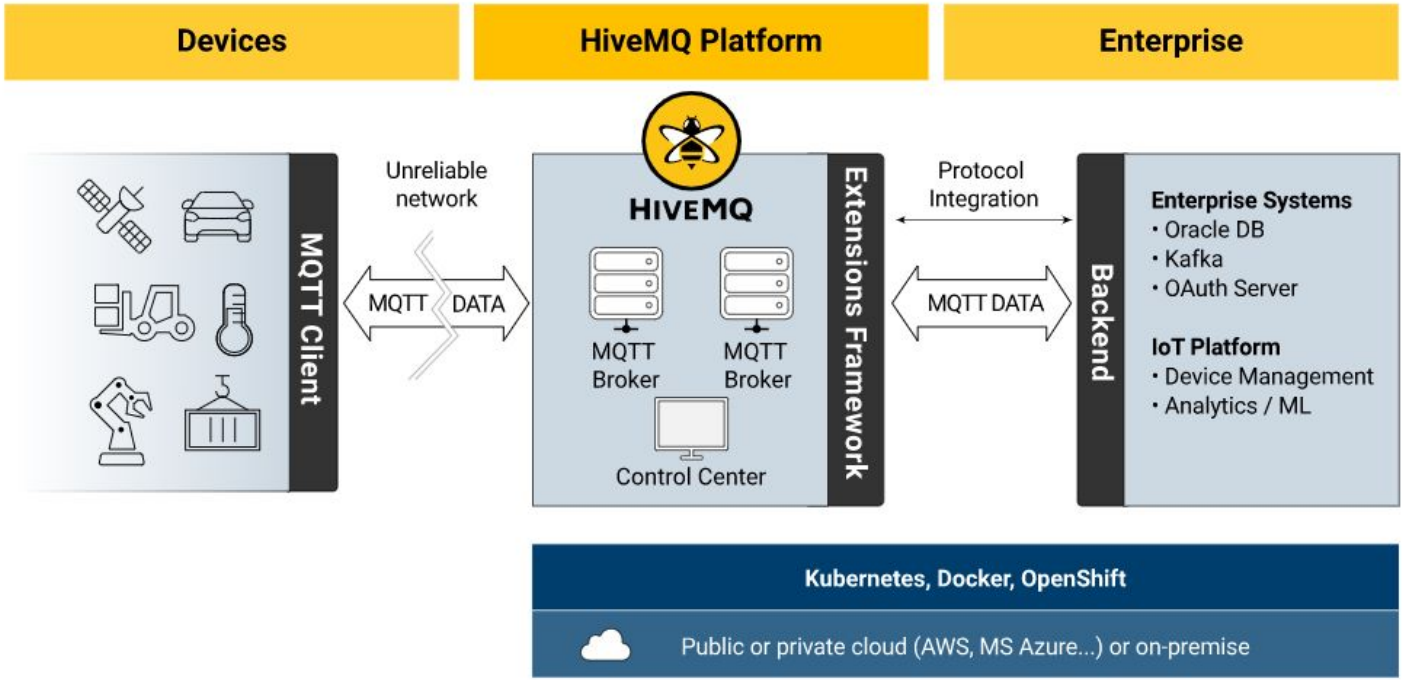
HiveMQ MQTT Broker

HIVEMQ

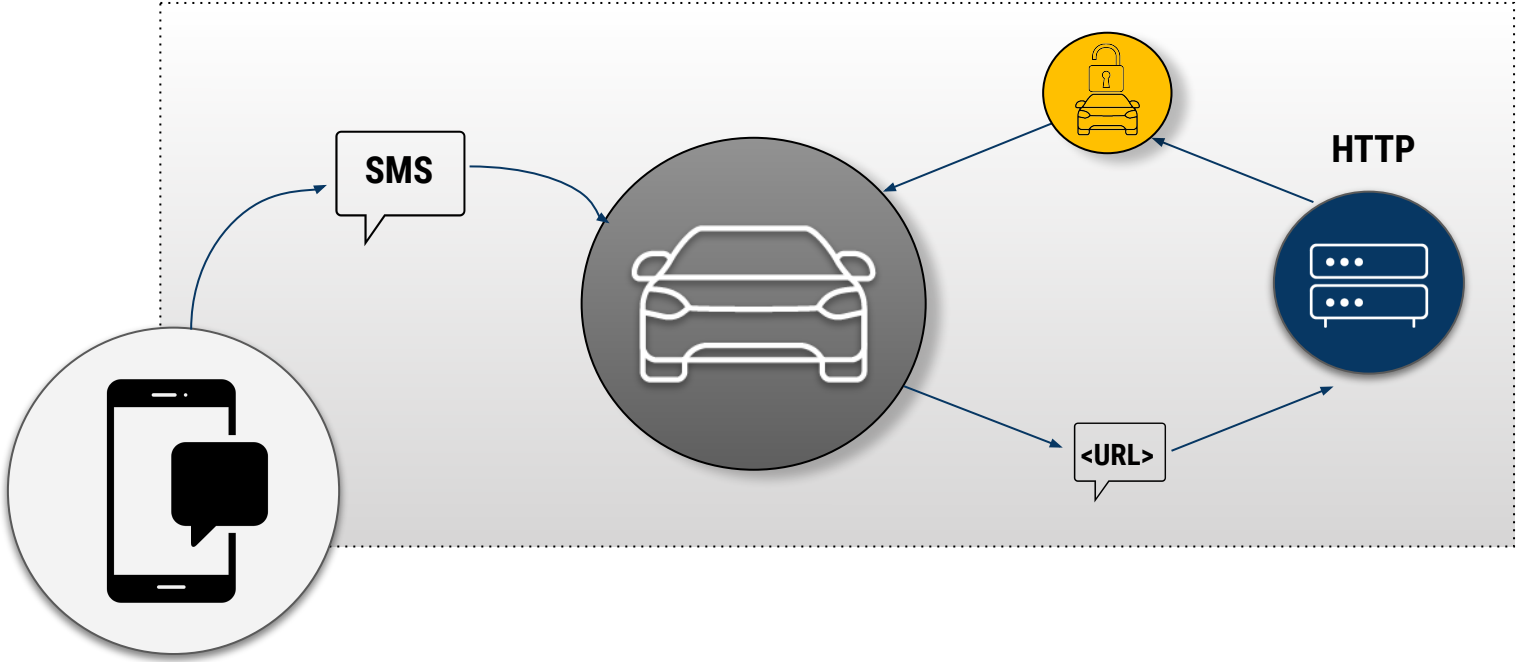
Automotive Customers

- Connected car platforms
- Car sharing services
- Connected car services

Enterprise MQTT Platform



Current Situation - Remote Door Unlock



Fast and Reliable Response Connected Car

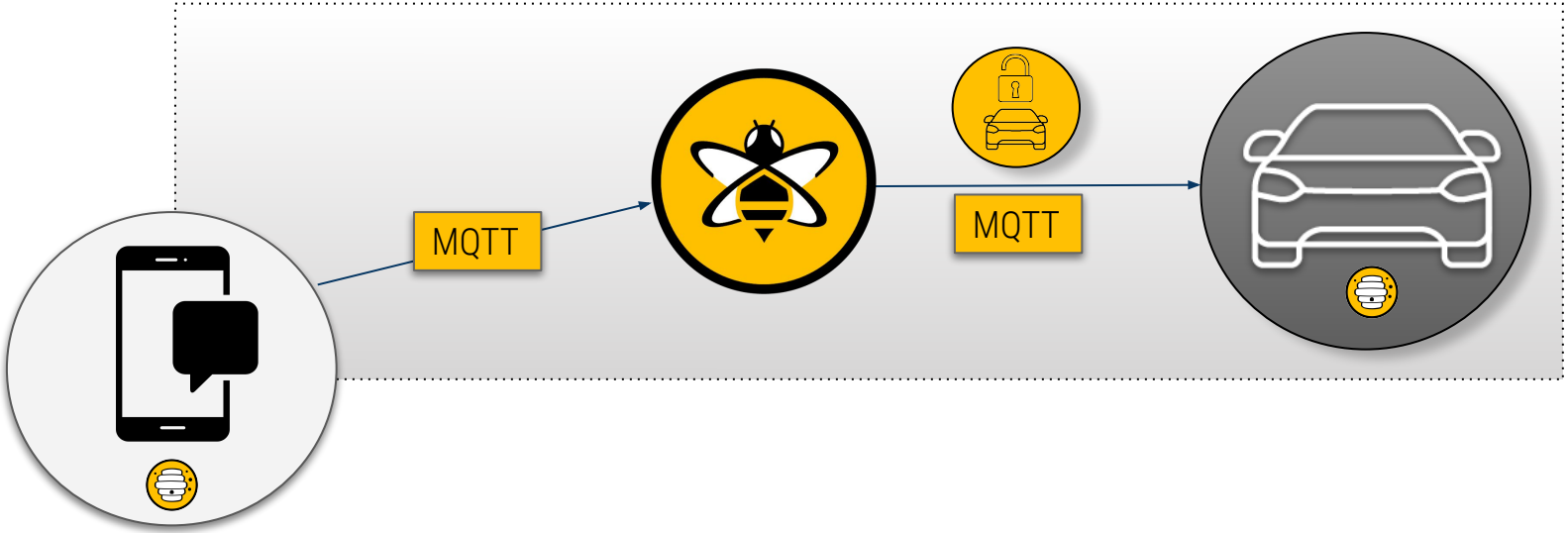
Mobile app using SMS/HTTP took up to 30 secs to unlock door.

HiveMQ solution

- Always-on connectivity for all devices
- HiveMQ runs on expandable Kubernetes cluster
- MQTT is designed for network low latency and Push communication
- HiveMQ implements all quality of service levels to guarantee delivery

Result Sub-second response time to unlock car

Remote Door Unlock with MQTT



HiveMQ for Connected Cars

- Persistent Always-on Client Connections
- Instant communication with millions of cars and less cost for bandwidth
- Always available, elastically scalable and no data loss
- Open API enables Custom Integration for Enterprise requirement
- Observability and Insights for Operations

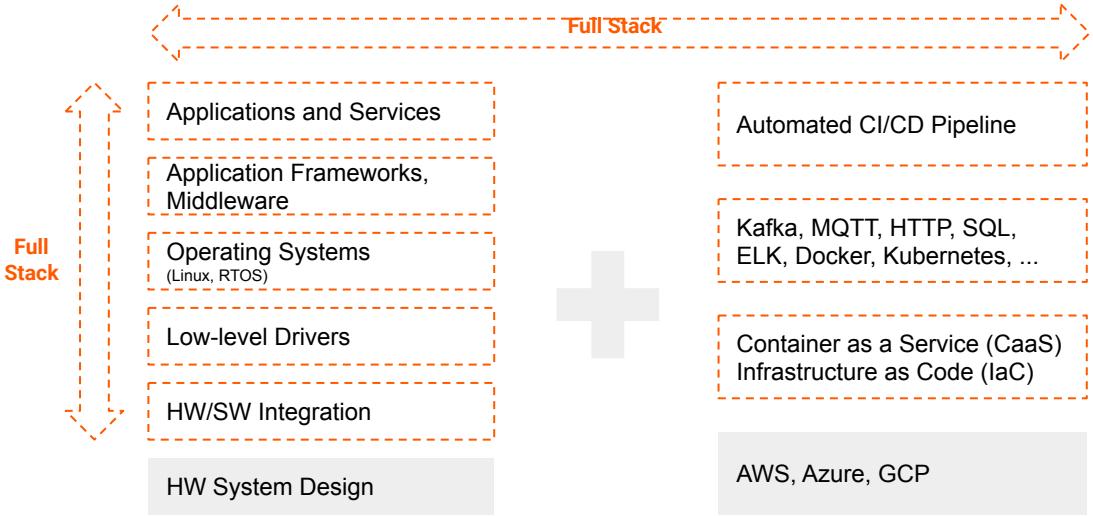
ESR Labs

Embedded Software

Edge (in-vehicle) side

Backends

Cloud side



Capability

Design, architecture, and ***development across entire SW stack*** incl. CI/CD process, HW/SW integration, app frameworks and applications.

Assets

End-to-end OTA platform allowing OEMs to deliver and install software OTA

Principles

Collaboration rooted in design-thinking and simplicity– yet precision and accuracy in functional safety (ASIL, SPICE) and E2E automotive security/privacy

As a **Software Tier 1** we develop, test and integrate all software components and build them **“ready-to-use”**

HIVEMQ

Conclusion

- Build a distributed system platform including SDKs & no black boxes
- Build extensible, loosely coupled publish/subscribe messaging systems, for both in-vehicle and vehicle-to-cloud communication
- Close technology gaps: it’s all there, you have to combine the pieces in the right way
- Enable software updates by development processes
- Think in customer experience

Conclusion

> “Simplicity is prerequisite for reliability”
>
> (Edsger W. Dijkstra)

ANY QUESTIONS?

Reach out to community.hivemq.com

THANK YOU