



INTERNET OF THINGS CONFERENCE

Christian Götz & Dominik Obermaier | dc-square

Bau deine eigene IoT Cloud

Wer sind wir?



CHRISTIAN GÖTZ
[@goetzchr](https://twitter.com/goetzchr)



DOMINIK OBERMAIER
[@dobermai](https://github.com/dobermai)

dc-square



HiveMQ



HiveMQ Plugin Entwicklung



IoT Beratung



Workshops & Schulung

Ziel des Workshops

Architektur einer IoT Plattform

Implementieren eines Prototypen

Endergebnis



Entwicklungsumgebung

Was jeder installiert haben sollte

- Java JDK 7 ✓
- Java IDE (IntelliJ, Eclipse, Netbeans, vi, ...) ✓
- GIT ✓
- Maven ✓
- HiveMQ 2.0.2 ✓
- MySQL Datenbank ✓
- MySQL Datenbankviewer ✓
- Aktueller Webbrowser ✓

GIT Repository

clone

[https://github.com/dc-square/
build-your-own-iot-cloud-
workshop](https://github.com/dc-square/build-your-own-iot-cloud-workshop)

oder

USB-Stick

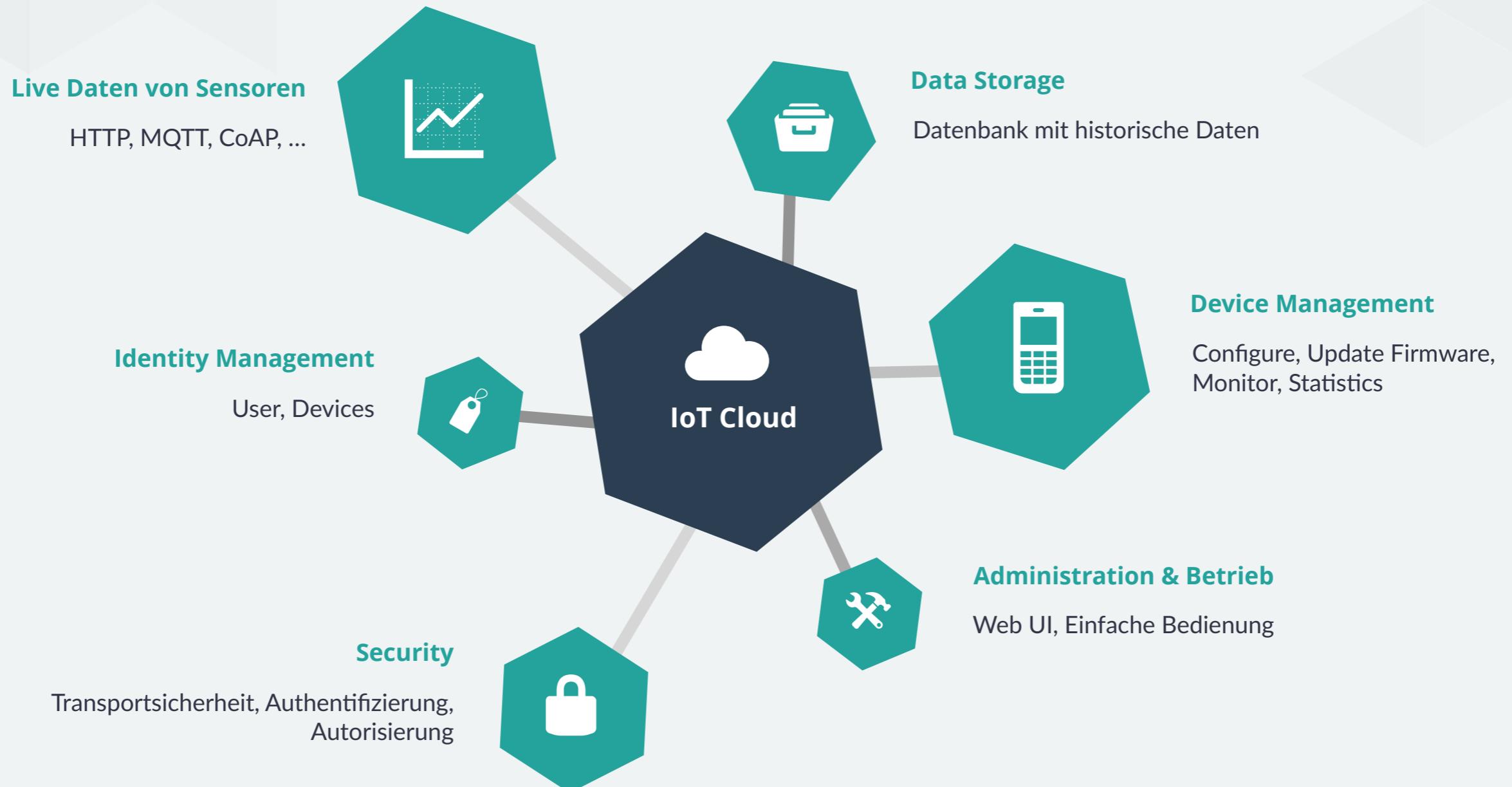
IoT Cloud Platform

Internet der Dinge

Technologie die Geräte über Kabel oder Drahtlos vernetzt

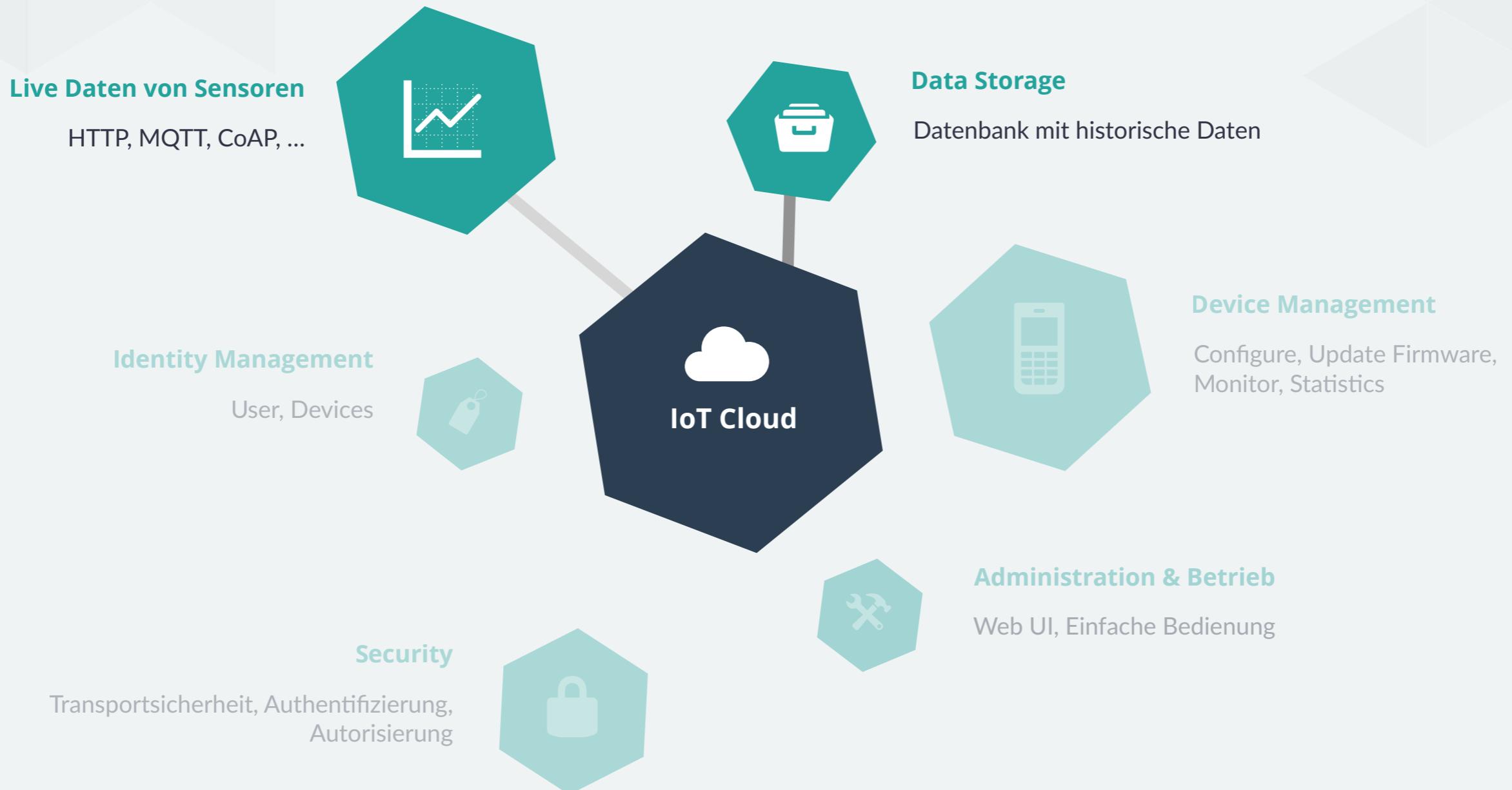
IoT Cloud Platform

Komponenten



IoT Cloud Platform

Komponenten, die wir heute benutzen



Technologien

Prototyp



Eclipse Paho,
HTTP
Geräte

HiveMQ
MQTT Broker



Dropwizard
REST API



HTML/JS
Web App



Technologien

Geräte



Eclipse Paho,
HTTP
Geräte

- Wetterstationen liefern Live Daten per Push (MQTT)
 - Simulation der Wetterstationen mit *Eclipse Paho*
- manche Geräte liefern Daten per HTTP

Technologien

HiveMQ



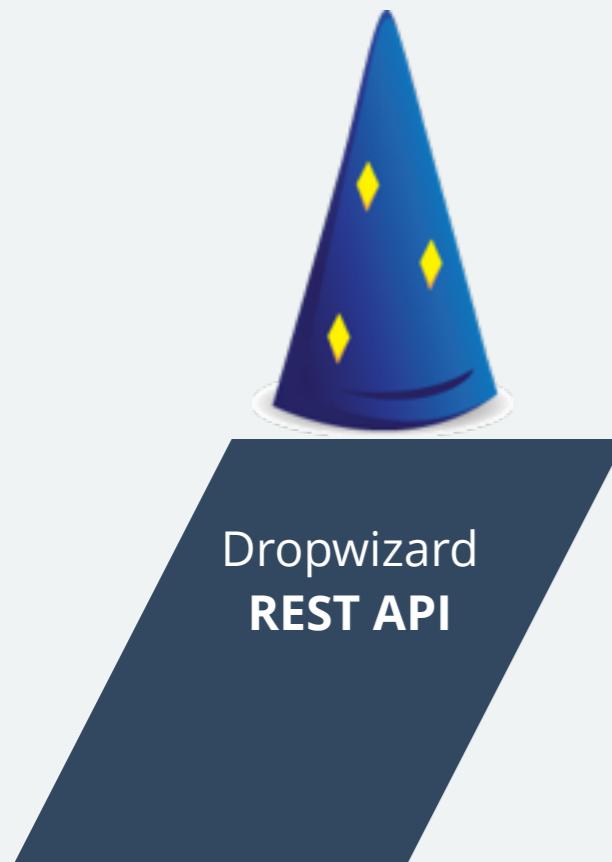
HiveMQ

HiveMQ
MQTT Broker

- Austausch von Daten zwischen Geräten
- Persistieren der Daten in die Datenbank

Technologien

Dropwizard



- Historische Daten zur Verfügung stellen
- eingehende POST Requests an HiveMQ weiterleiten

Technologien

Webapp



HTML/JS
Web App

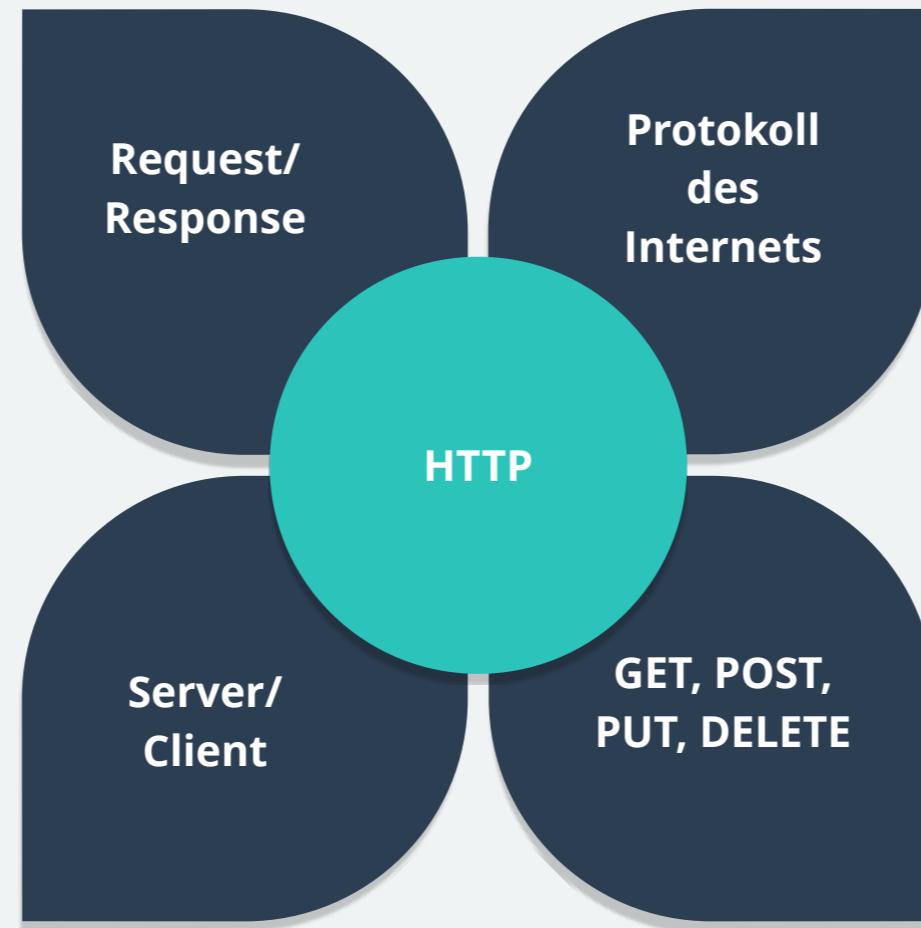
- Anzeige von Live Daten über MQTT
- Anzeige von historischen Daten über REST API

Teil 1: Geräte

MQTT, HTTP

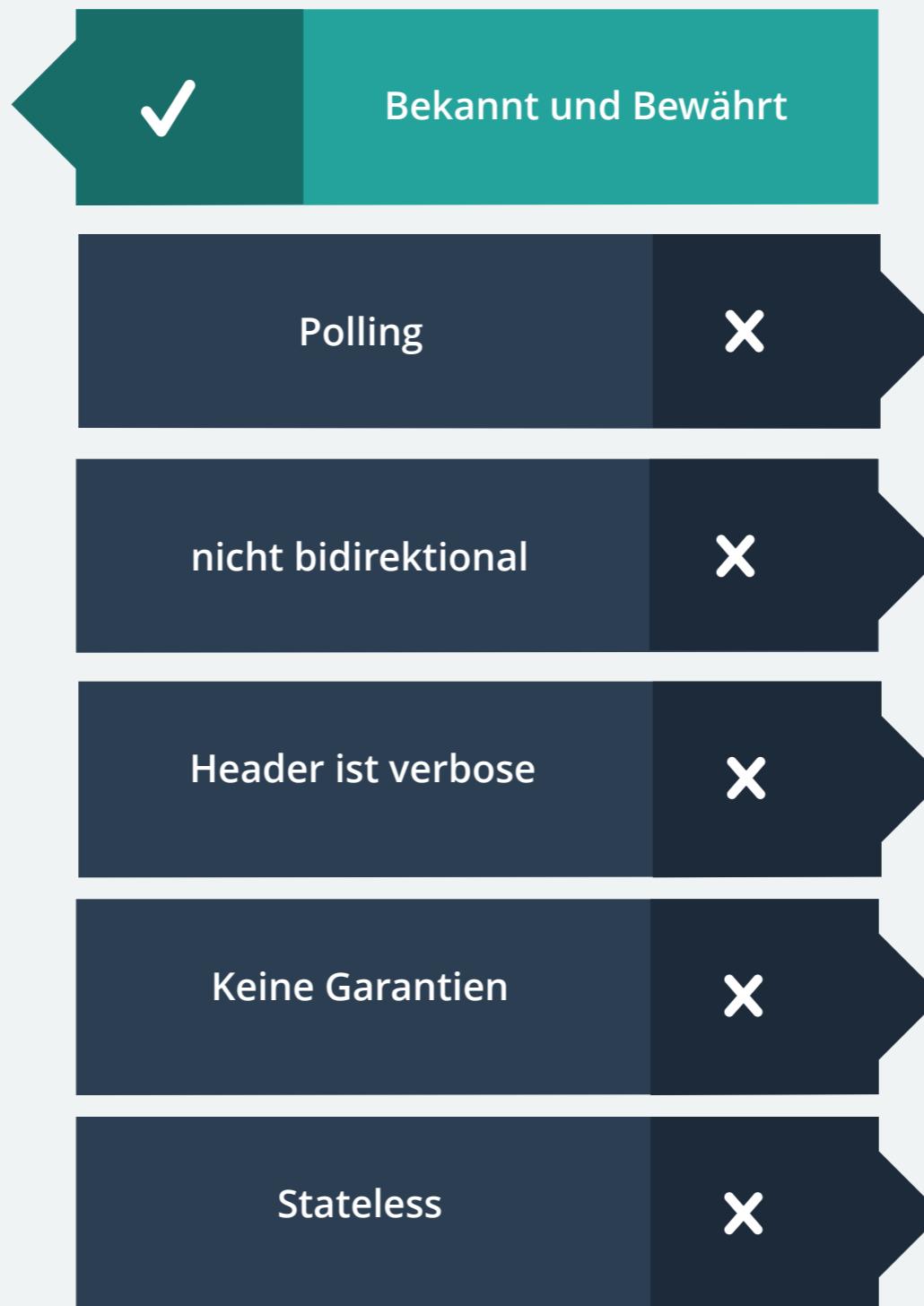
HTTP

Fakten



HTTP

Vorteile/Nachteile



MQTT

Kurzübersicht

Einfach

Connect, Publish, Subscribe, Unsubscribe,
Disconnect

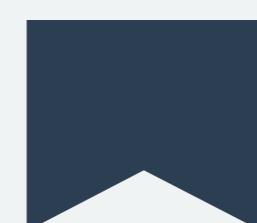
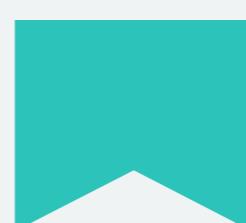


Messaging Protokoll

Austausch von Nachrichten, teilweise Queuing

basiert auf TCP

Port 1883

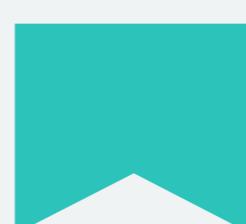


Published/Subscribe

Clients abonnieren auf Topics auf denen andere
Clients Nachrichten veröffentlichen

Minimaler Overhead

teilweise nur 2 Byte



Ausgelegt für unstabile Netze

individuelle Garantien möglich für jede Nachricht

MQTT History

Timeline



1999

Entwickelt
IBM/Arccom



2013

Eclipse IoT (M2M)



2015



2011

MQTT released
royalty free



The Organization for the Advancement of Structured Information Standards

2014

OASIS Standard

MQTT - Topics

Feature

- 1 > einfacher Topic
`iotcloud/device1`
- 2 > einfacher Topic
`iotcloud/device1/temp`
- 3 > Single Wildcard
`iotcloud/+temp`
- 4 > Multi-level Wildcard
`iotcloud/#`

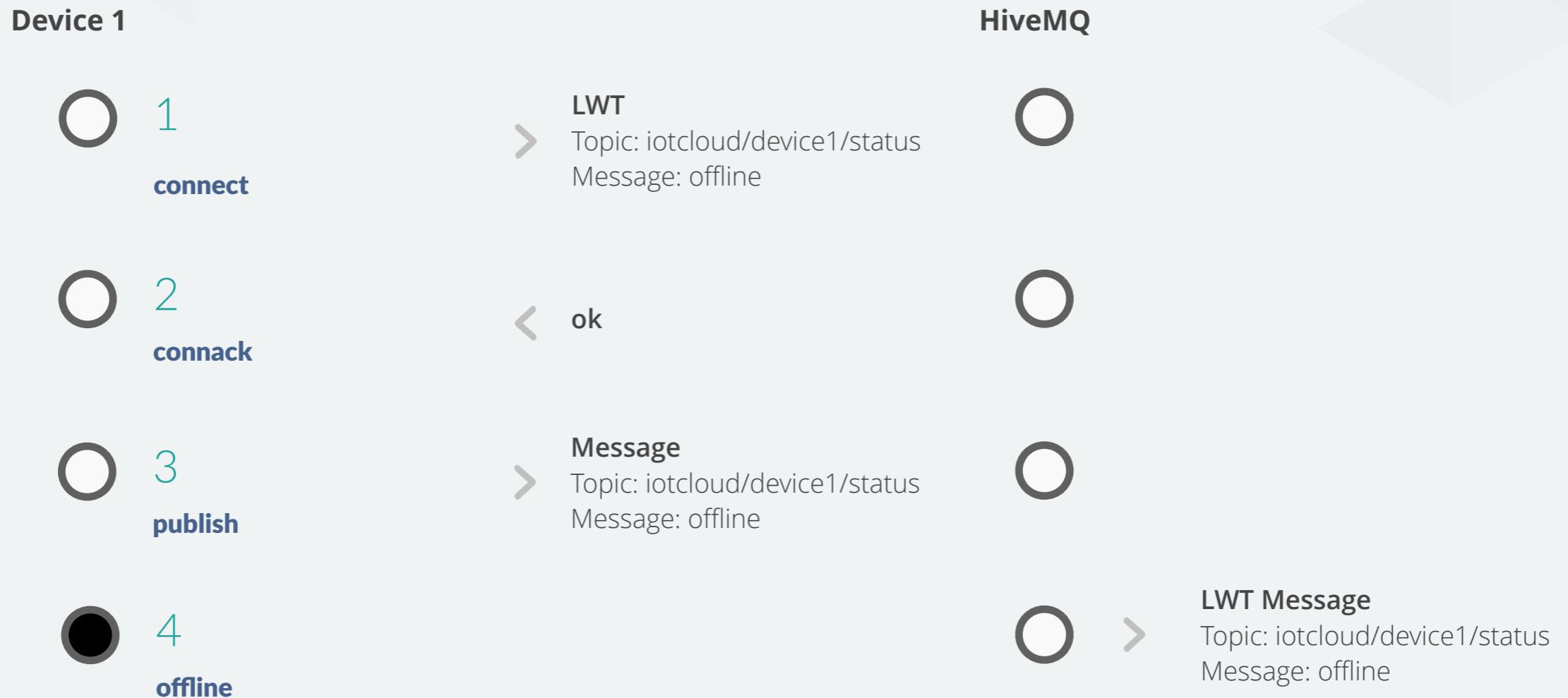
MQTT - Quality of Service

Feature



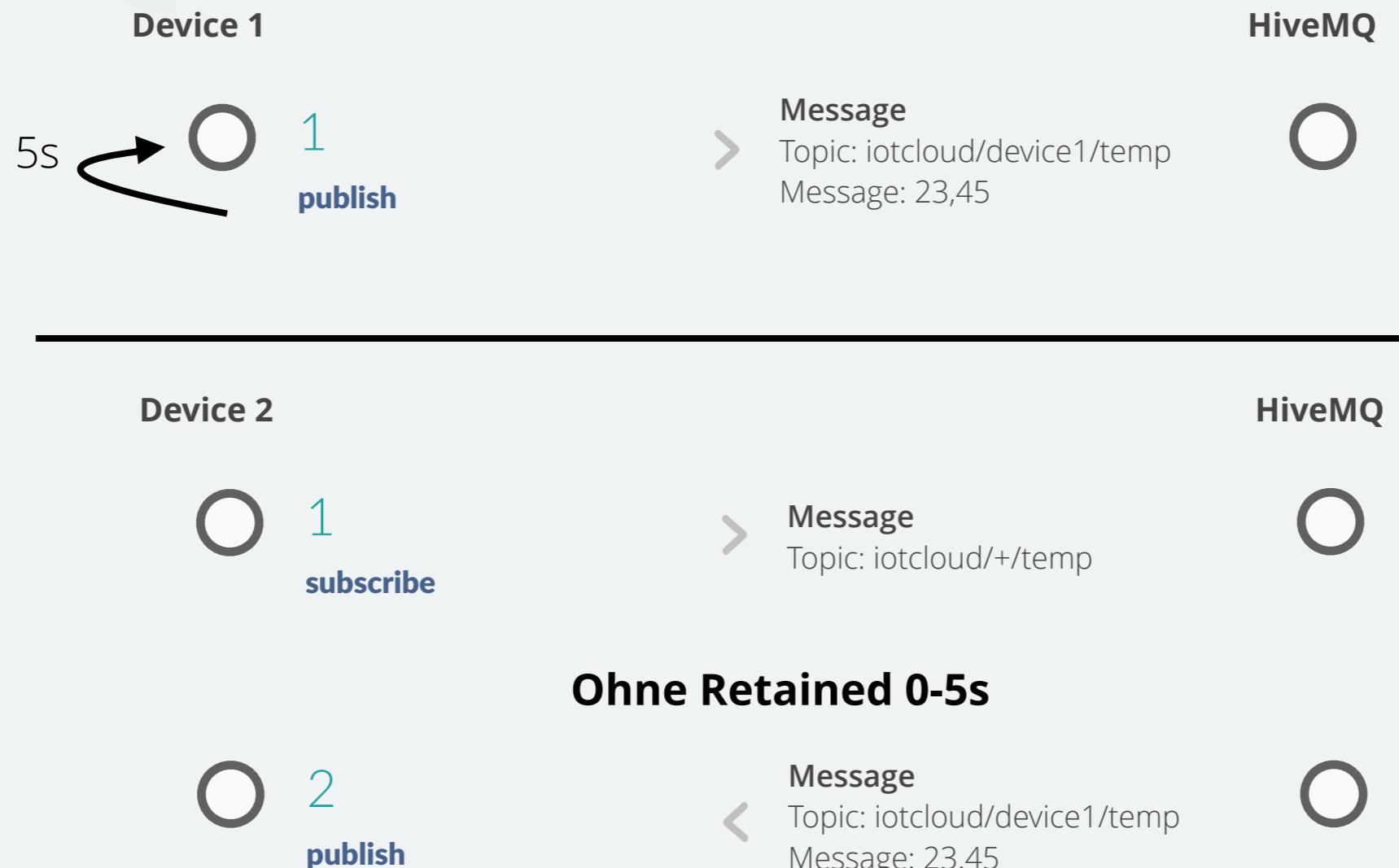
MQTT - LWT

Feature



MQTT - Retained Msg

Feature



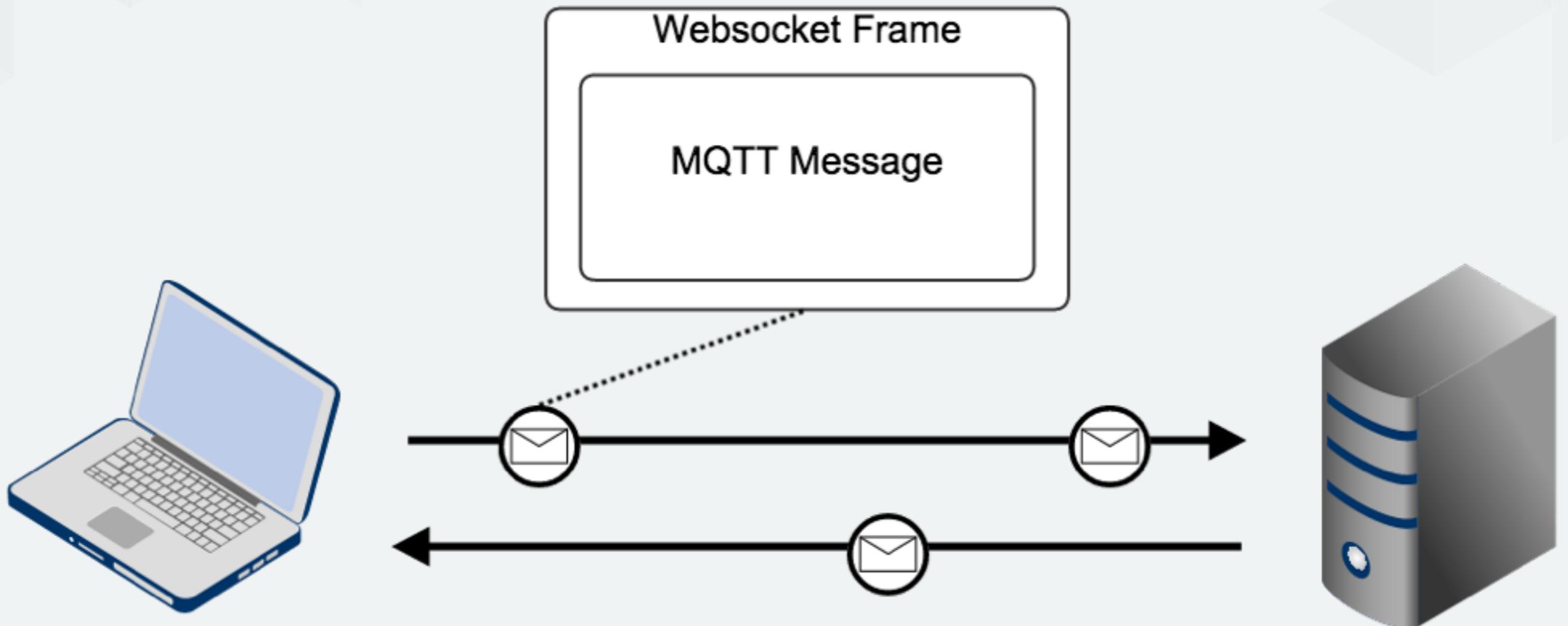
Ohne Retained 0-5s

< Message
Topic: `iotcloud/device1/temp`
Message: `23.45`

Mit Retained sofort!

MQTT over Websockets

Feature



MQTT Broker

Kurzübersicht

Herzstück von MQTT



Versenden der Nachrichten an Clients



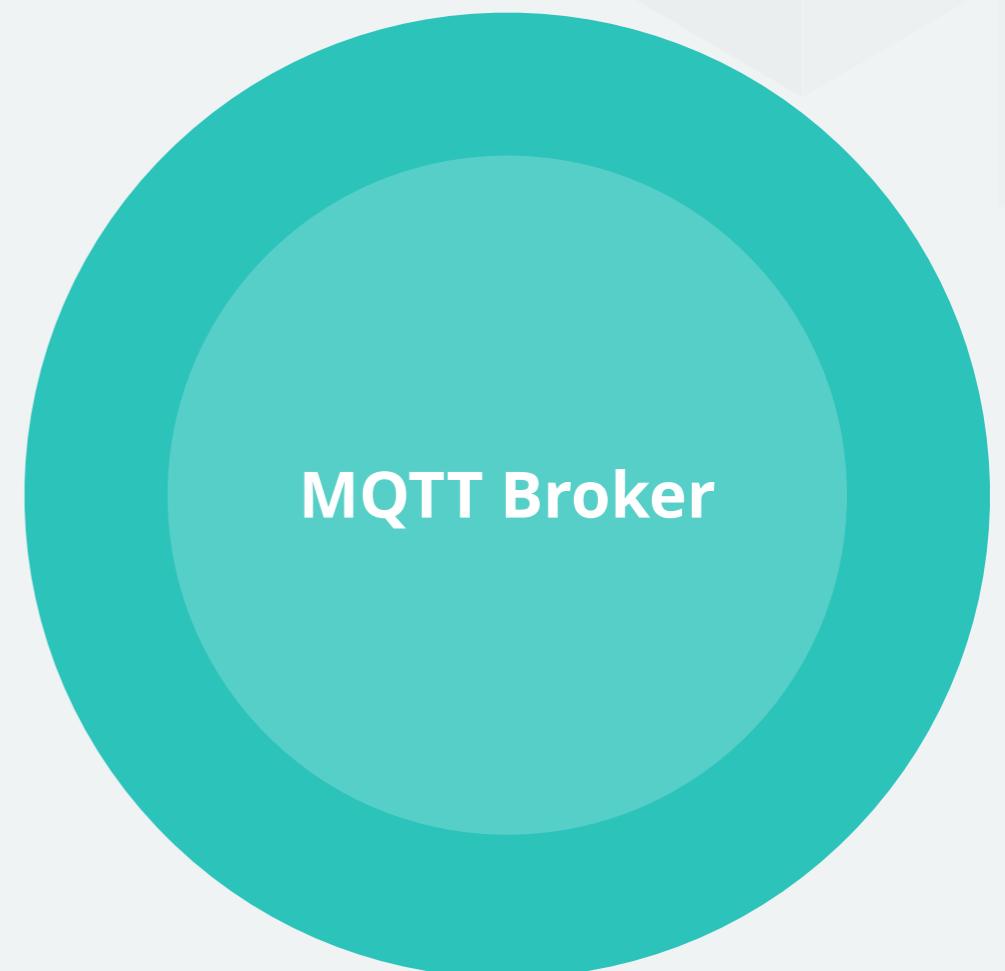
Benutzt nur Topics



Topics sind dynamisch

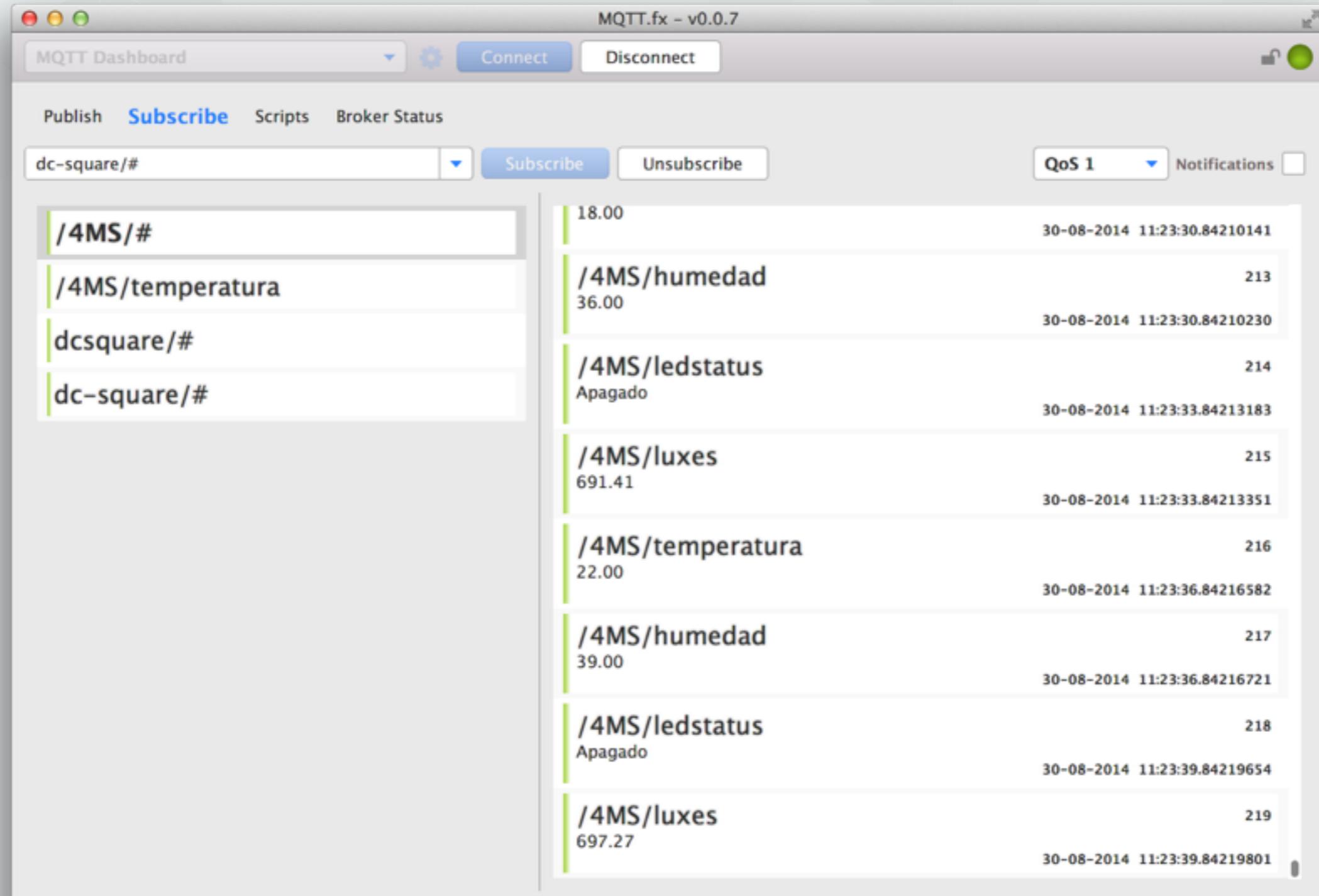


Benutzerdefinierte Funktionen



MQTT GUI Tools

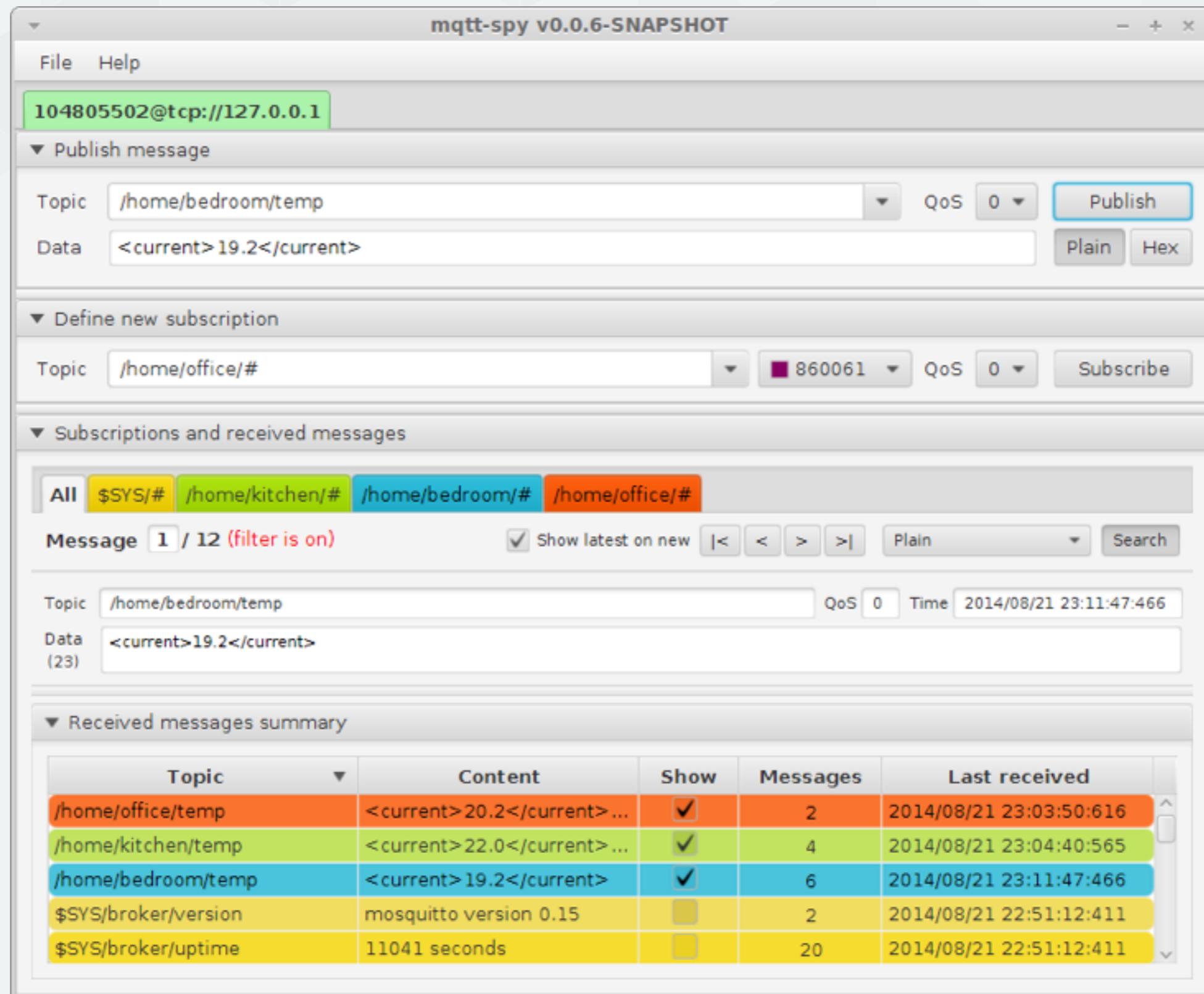
MQTT.fx 0.0.7



<http://www.jensd.de/wordpress/?p=1423>

MQTT GUI Tools

MQTT spy 0.0.6



<https://code.google.com/p/mqtt-spy/>

MQTT CMD Tools

mosquitto_pub/_sub

Mosquitto Clients

Publish/Subscribe

```
$ mosquitto_pub -h broker.mqttdashboard.com  
-t iotcloud/test -m "test" -q 1
```

http://mosquitto.org/man/mosquitto_pub-1.html

```
$ mosquitto_sub -h broker.mqttdashboard.com  
-t iotcloud/test
```

http://mosquitto.org/man/mosquitto_sub-1.html

MQTT Libraries

Eclipse Paho



Open Source

"Reference Implementation"

Many languages: Java, Javascript,
Lua, C, C++, Go, Python

Active Community

JS Library uses MQTT over
Websockets

MQTT Libraries

FuseSource MQTT Client



MQTT-Client

Open Source

3 API Styles

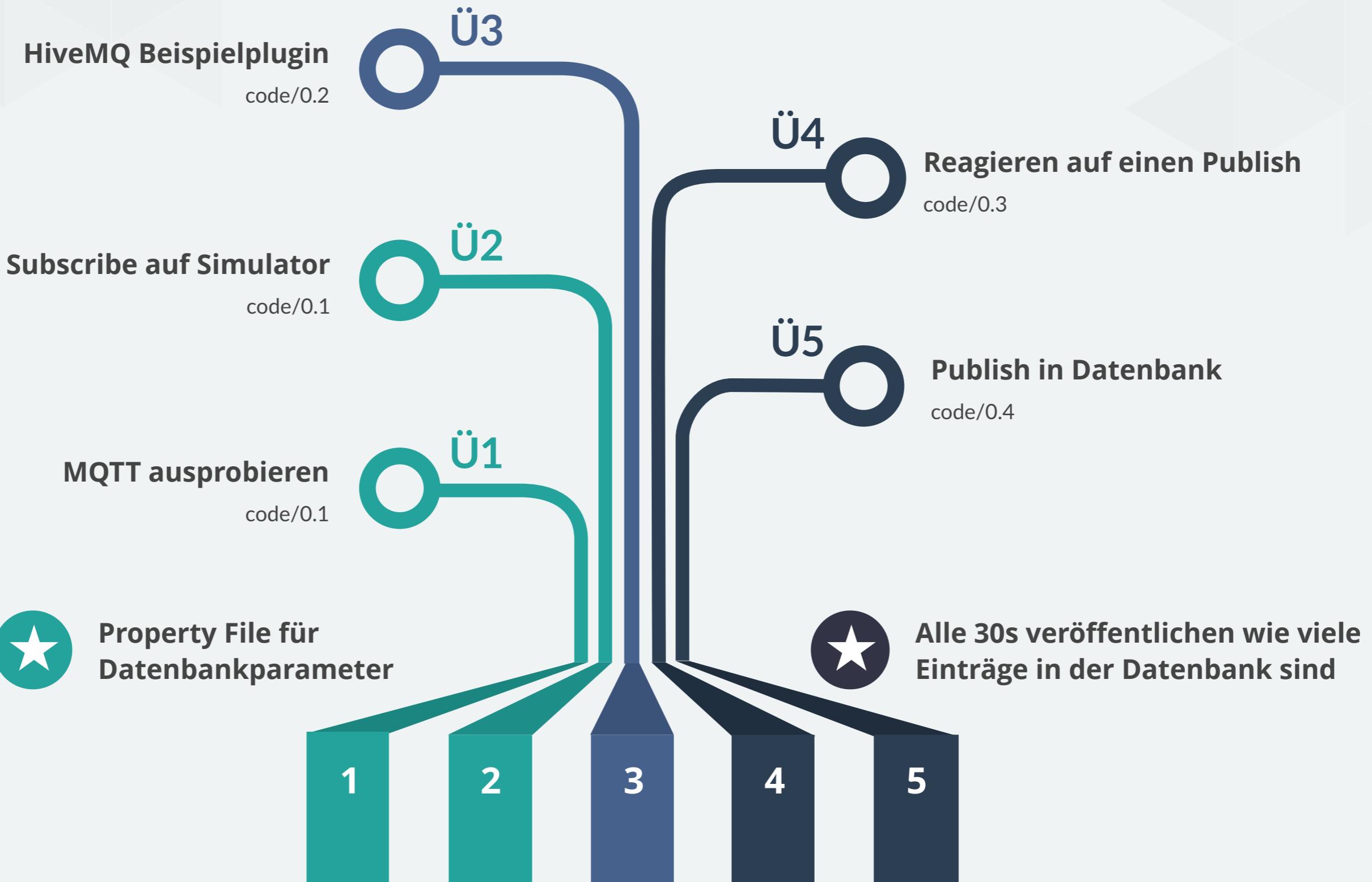
Automatic Reconnect

Maven Central

Less active Community

MQTT + HiveMQ

Hands-on



Übung 1

MQTT ausprobieren



Übung 2

Simulator benutzen und Daten von Gerät 2 empfangen



Teil 2: MQTT Broker **HiveMQ**

MQTT Broker

Hochskalierbarer MQTT Broker

> 100.000 Verbindungen

**Offenes Plugin System**

Einhängen von benutzerdefinierter Logik, Integration in bestehende Systeme

Cloud readyAmazon Web Service, Microsoft Azure,
Own Datacenter**Implementiert den MQTT Standard**

Gleichzeitiges Verbinden von MQTT Clients der Version 3.1.1 und 3.1 möglich

SicherheitTLS Verschlüsselung, X.509 Zertifikate,
Benutzerdefinierte und feingranulare
Authentifizierung und Autorisierung**Einfache Benutzung und Betrieb**\$SYS Topics, Ausführliche Dokumentation,
Monitoring über JMX und Graphite

Plugin System

Hands-on

Don't block in a Plugin. Never.

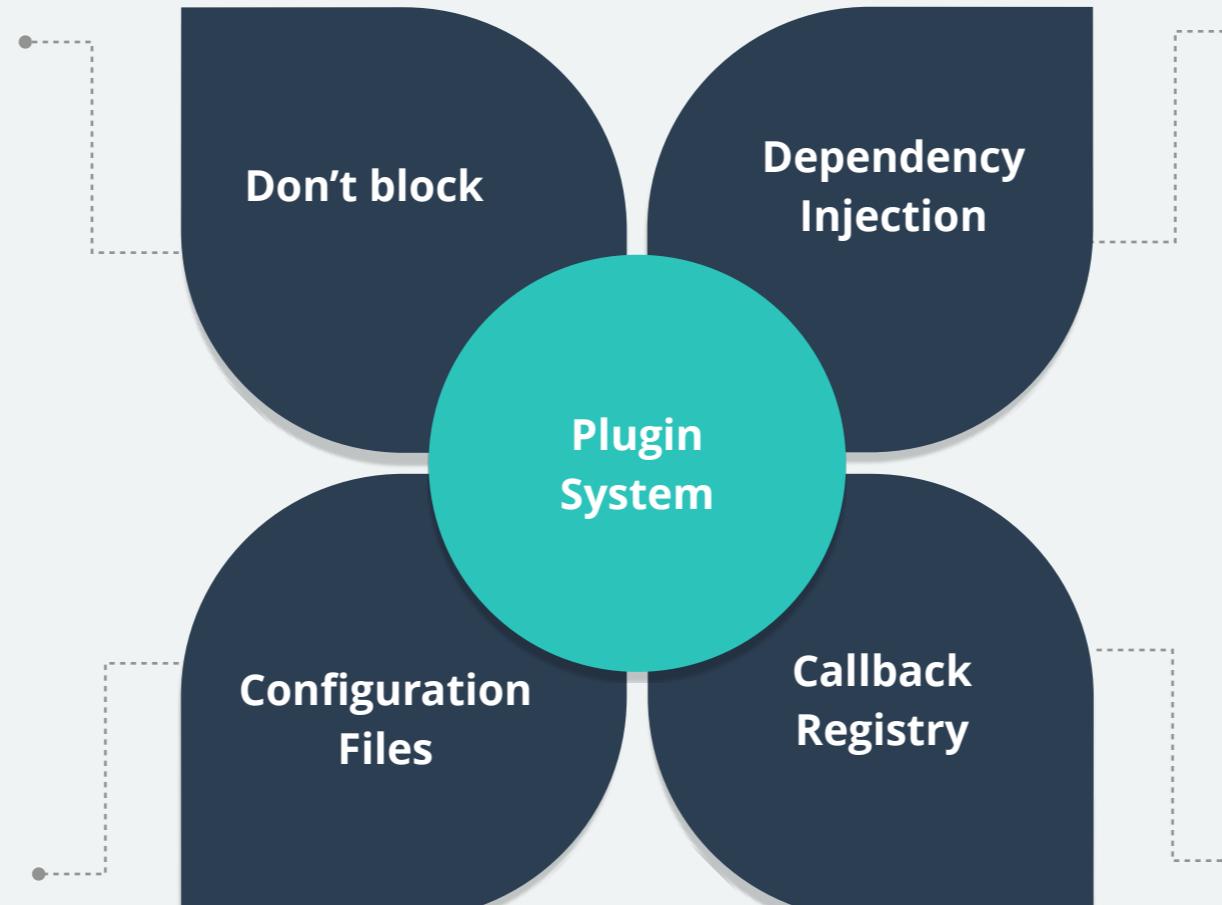
Andere Threads benutzen

Inject everything!

Steigert die Testbarkeit!

Apache Configuration Support

Einfach Config Files erstellen und
automatisch nachladen



Callback bekannt machen

Callback muss hier registriert
werden, ansonsten wird er nicht
ausgeführt

Plugin Callbacks

Einhängen von benutzerdefinierter Logik



Plugin Entwicklung

Tools für den Entwickler

- HiveMQ SPI
 - Vorraussetzung um entwickeln zu können
- Maven Plugin
 - Plugin mit HiveMQ starten
 - Debuggen im Server/Client Modus
- Assembly Plugin
 - Erstellung eines auslieferbaren Archiv
- JavaDoc

Technologien

HiveMQ



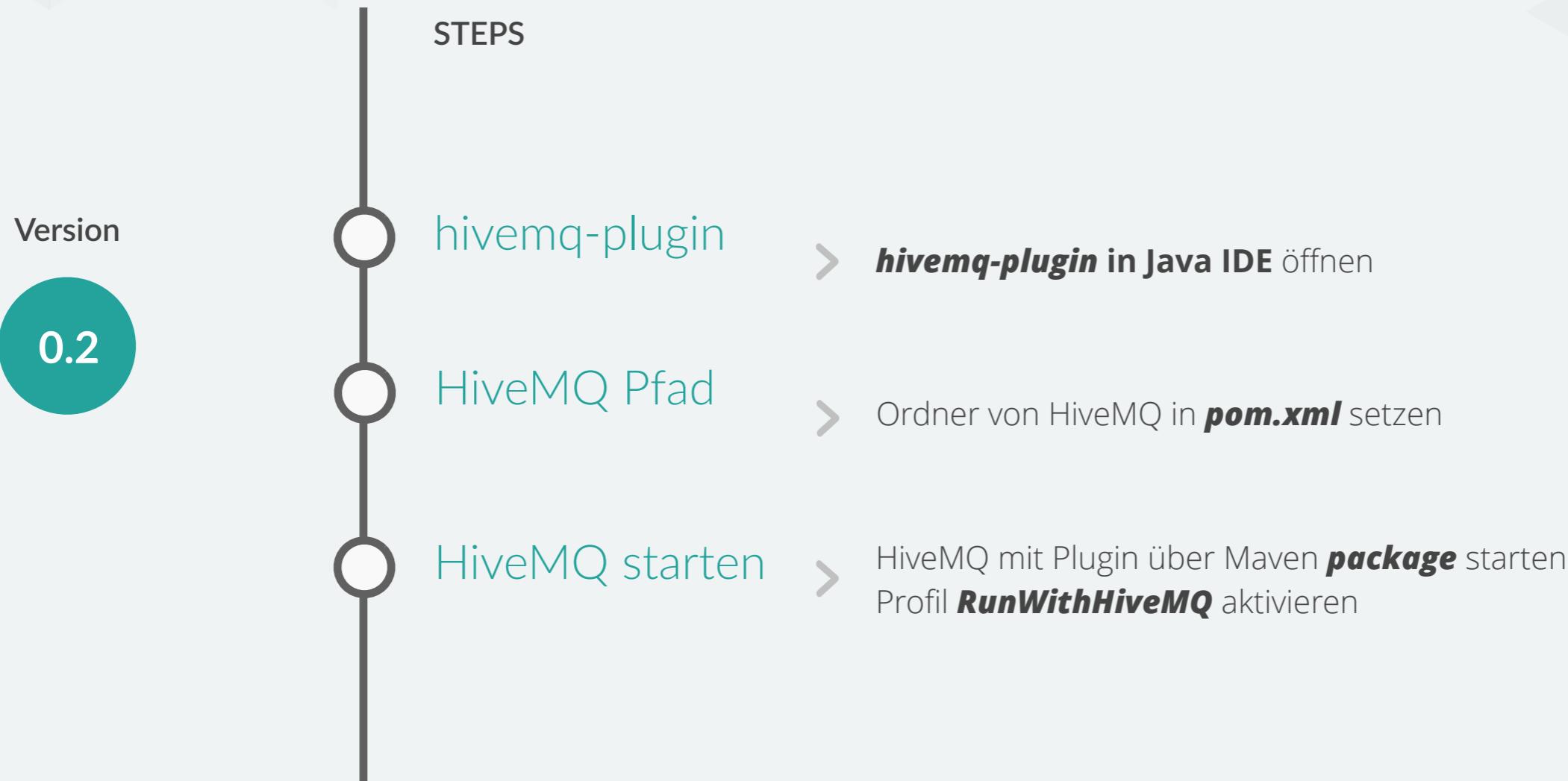
HiveMQ

HiveMQ
MQTT Broker

- Austausch von Daten zwischen Geräten
- Persistieren der Daten in die Datenbank

Übung 3

Plugin mit HiveMQ starten



Übung 4

Benutzerdefinierte Logik für Publish



Übung 5

Schreiben in die Datenbank



Verbesserungen

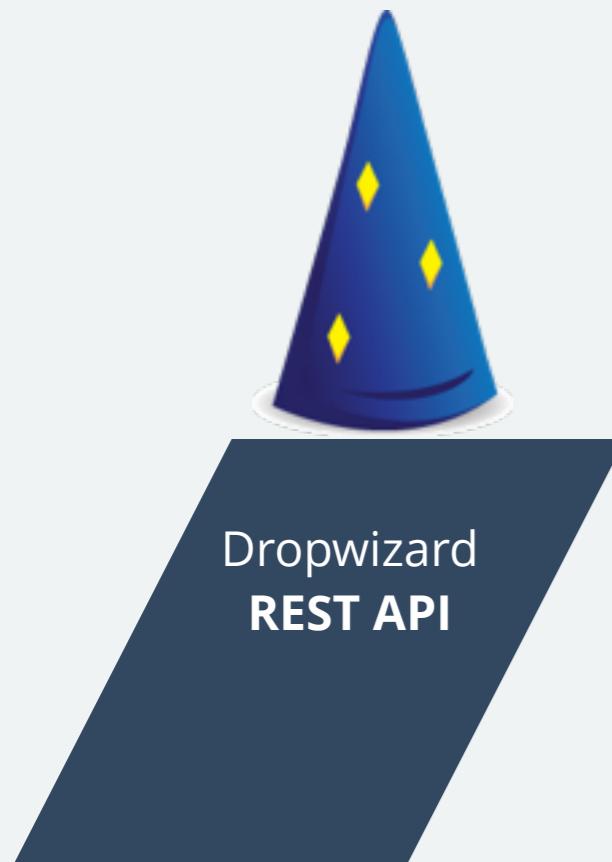
HiveMQ Plugin

- Connection Pool für Datenbankverbindung
- JDBC in extra Thread
- ORM-Framework einsetzen
- Authentifizierung und Autorisierung der Clients
- Transportverschlüsselung über TLS

Teil 3: REST API Dropwizard

Technologien

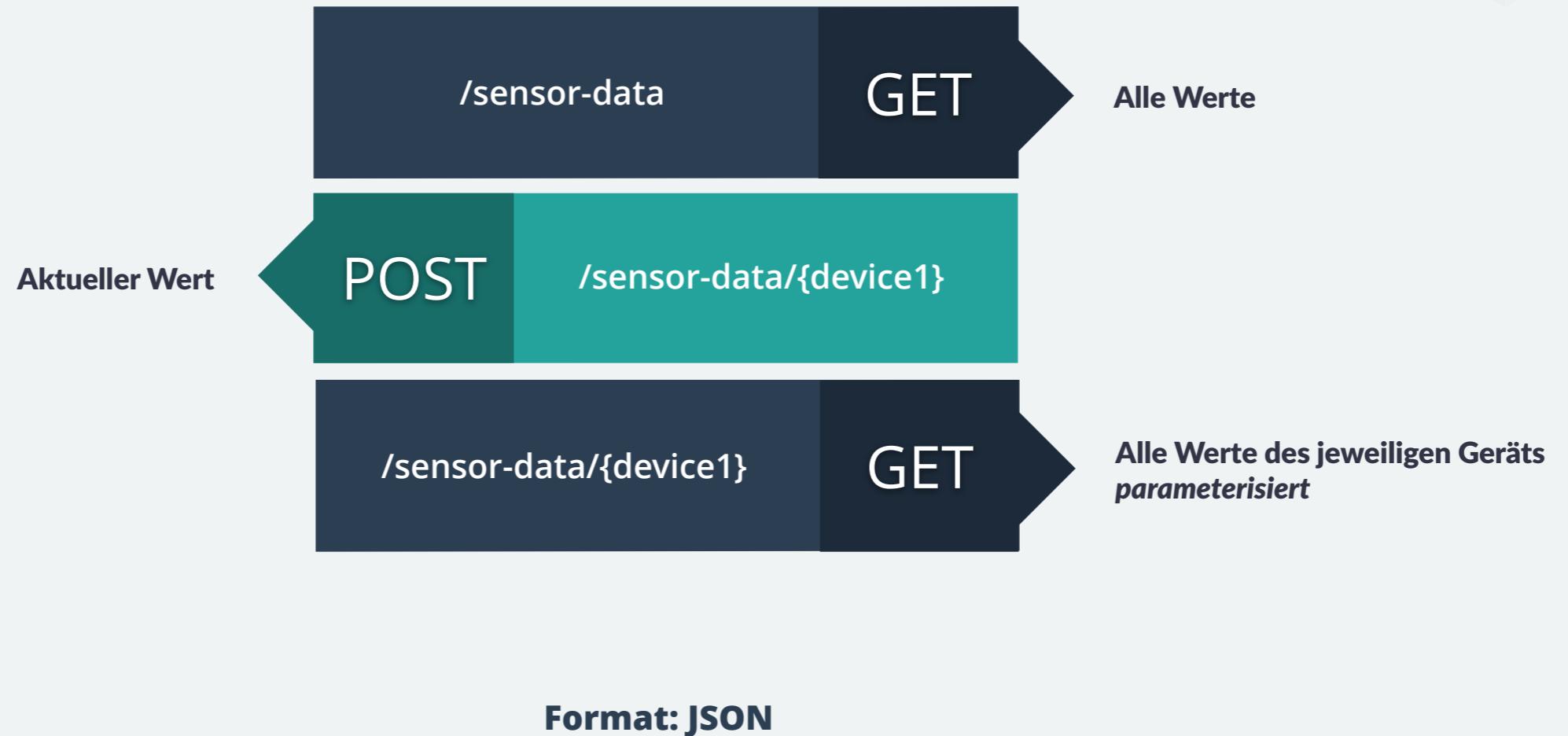
Dropwizard



- Historische Daten zur Verfügung stellen
- eingehende POST Requests an HiveMQ weiterleiten

REST Ressourcen

IoTCloud



Dropwizard

Kurzübersicht



“Produktiv einsatzbereite, hochskalierbare REST API”

Ausgerichtet auf den einfachen Betrieb

Modulbasiert mit vielen mitgelieferten Modulen

Technologien

Dropwizard

jetty://

HTTP

 Jersey

REST

Jackson

JSON

metrics



powered by



Getting Started

Dropwizard

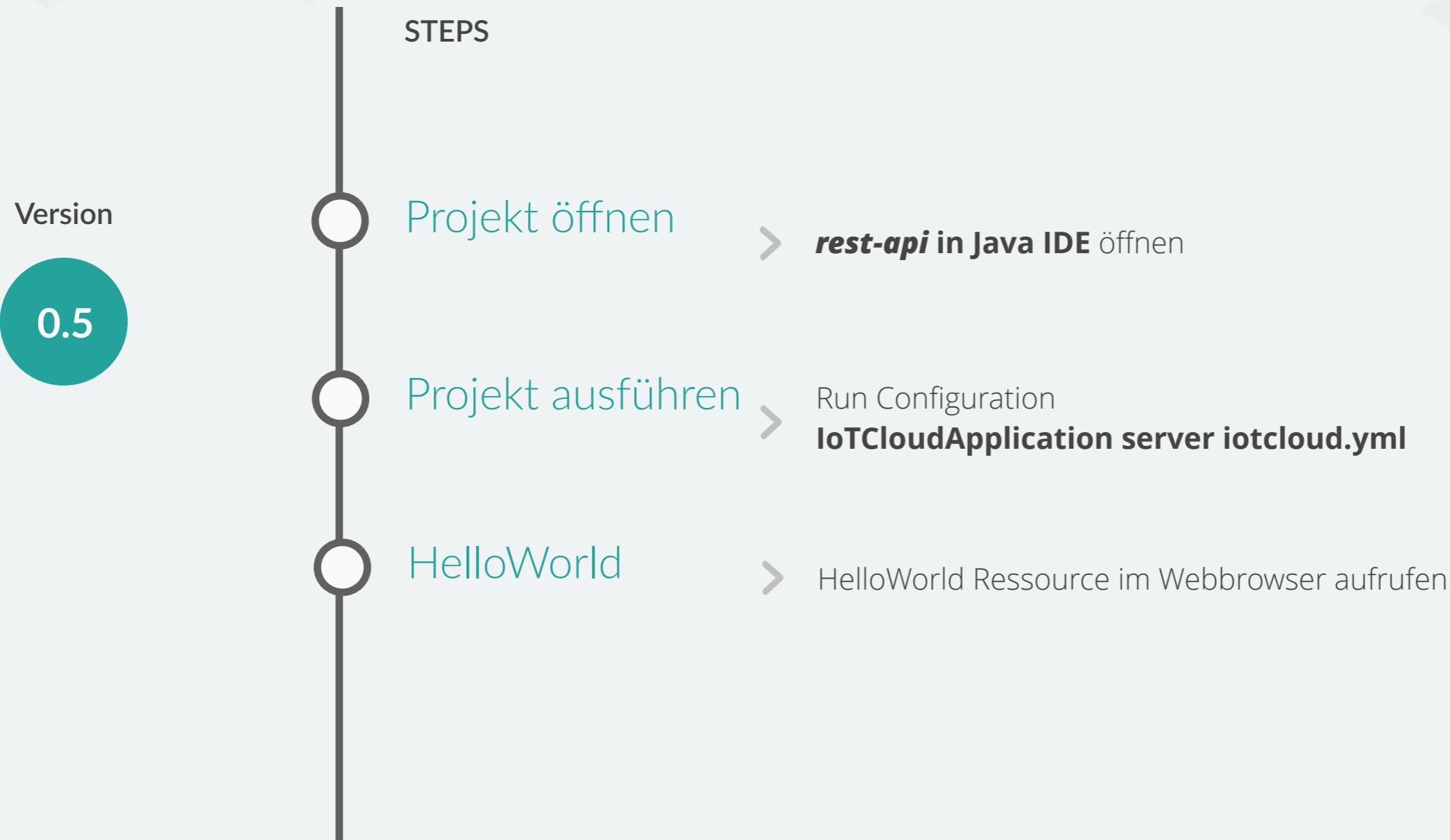
Configuration
yaml
+ class

Ressourcen
classes

Starten
server config.yml

Übung 6

Erstes Dropwizard Projekt



Übung 7

GET /sensor-data



Übung 8

JDBI statt JDBC



Verbesserungen

HiveMQ Plugin

- Caching einbauen
- Datenbankverbindung mit ConnectionPool
- ORM Framework einsetzen (JDBI, Hibernate)
- Authentifizierung, Autorisierung
- Mehrere Ressourcen
- Transportverschlüsselung mit TLS

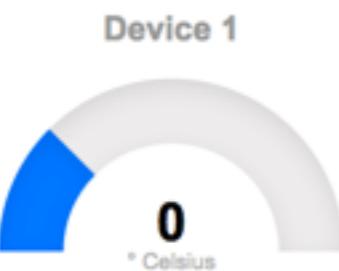
Teil4: WebUI

Erster Test für die IoT Cloud

WebUI

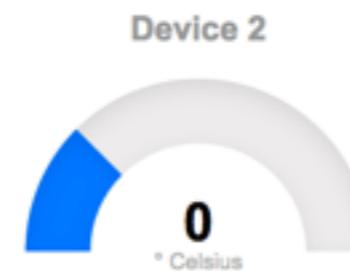
HiveMQ Plugin

Temperature 1



Show 10 entries

Temperature 2



Live Daten über MQTT

Search:

Id	Timestamp	Device ID	Value
1	31.8.2014 04:22	device1	0 °C
2	31.8.2014 04:22	device2	0 °C
3	31.8.2014 04:22	device1	0 °C
4	31.8.2014 04:22		3 °C
5	31.8.2014 04:22	device1	3 °C
6	31.8.2014 04:22	device2	6 °C
7	31.8.2014 04:22	device1	4 °C
8	31.8.2014 04:22	device2	9 °C
9	31.8.2014 04:22	device1	4 °C

Historische Daten über REST

Übung 9

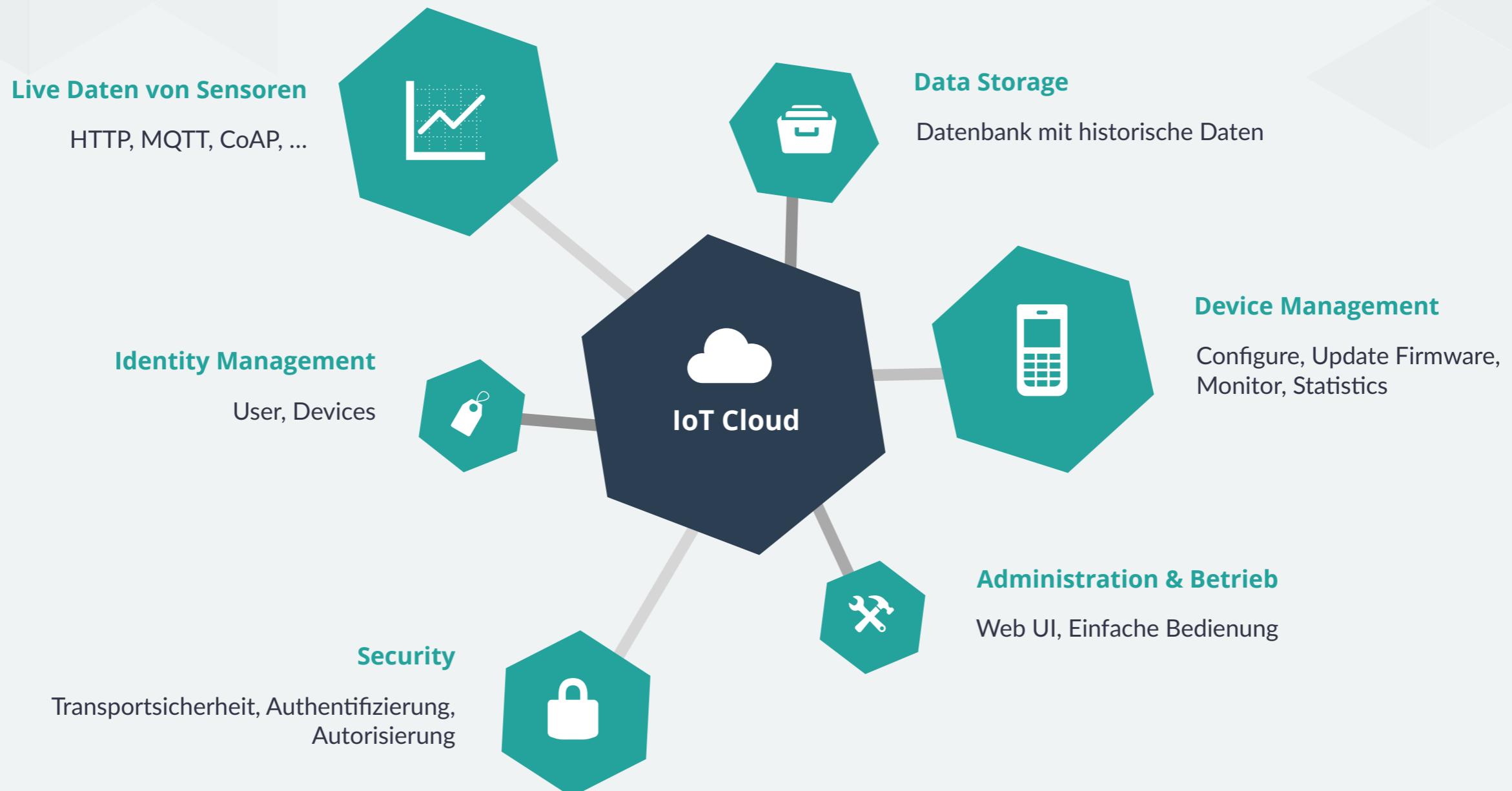
WebUI Ausliefern



Zusammenfassung

IoT Cloud Platform

Komponenten



Ausblick



Welche Herausforderungen sind noch zu lösen?

Danke!

IoT Con HiveMQ Special



<http://www.hivemq.com/iot-con-special-2014/>