Master-Praktikum: IoT

(IN2106, IN4224)

- IoT Core Team -

Final Presentation
30.07.2018

Topic

Presenting the design and capabilities of the reinvented IoT platform

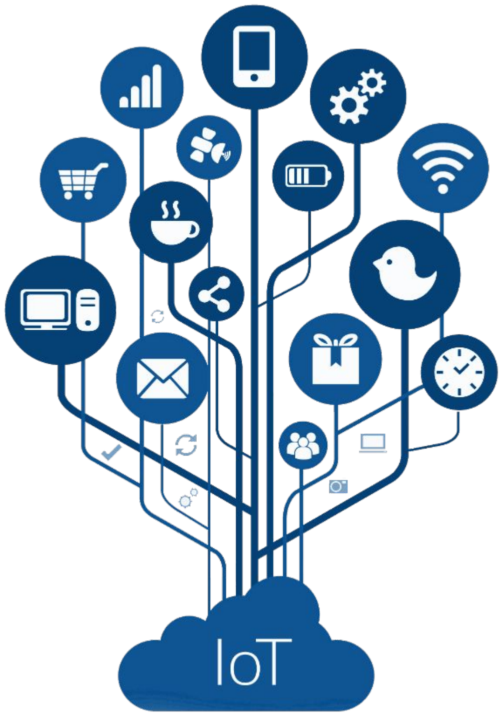
Presenters

Peeranut Chindanonda
Helge Dickel
Christoph Gebendorfer
Bahareh Hosseini
Hans Kirchner

Agenda

1. Defined Requirements
2. Architecture Design
3. Data Model
4. Live Demo
5. Security
6. Benchmarks
7. Comparison to Existing Solutions
8. Misc
9. Conclusion, Challenges & Outlook

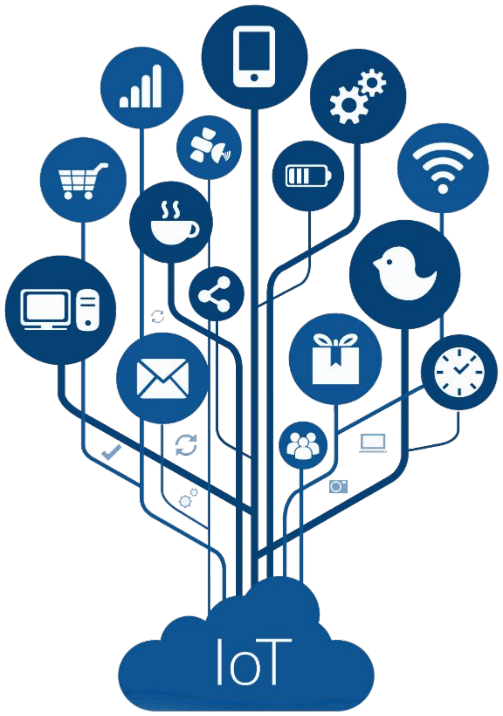
Defined Requirements



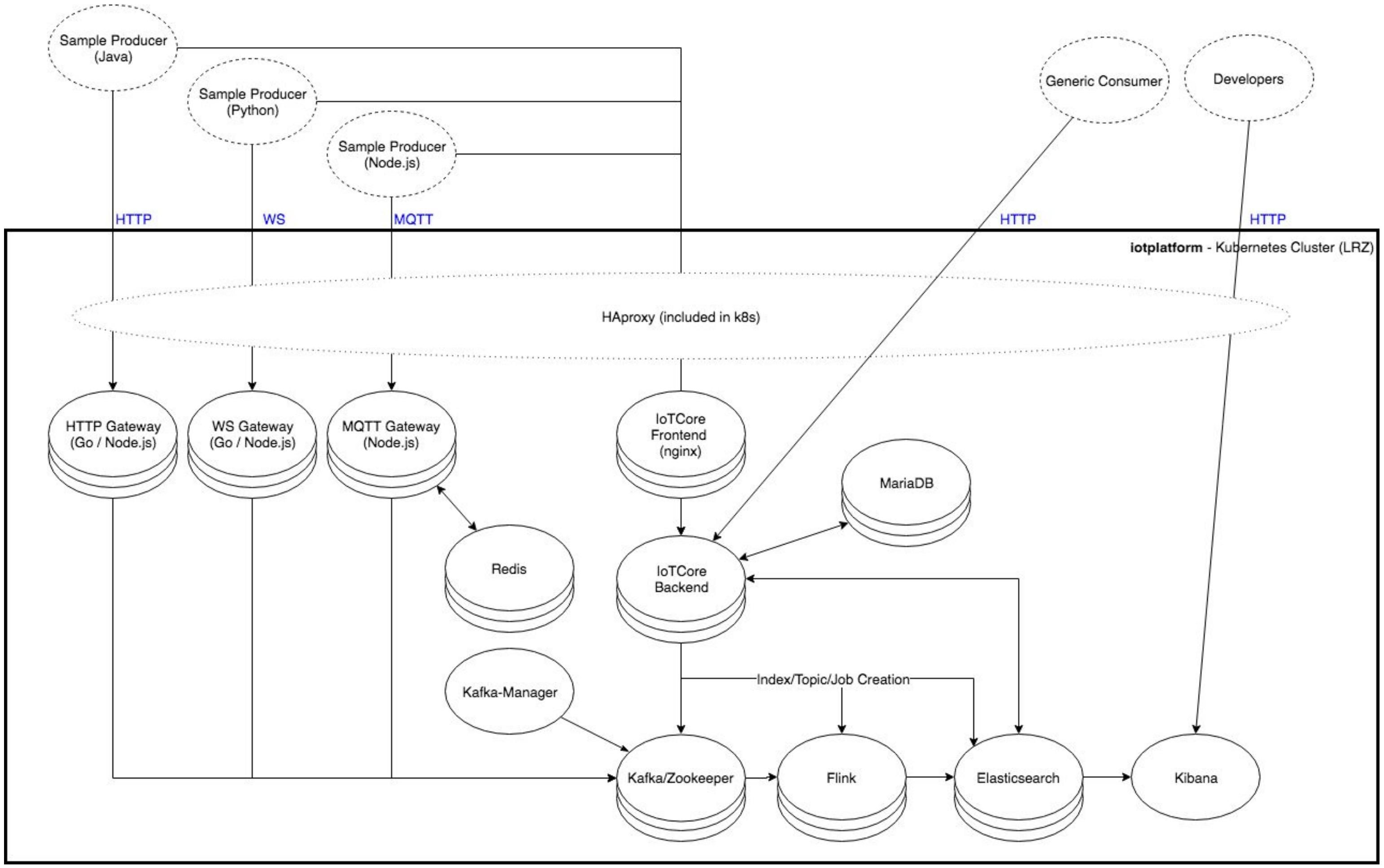
Defined Requirements

	Requirement	Previous Status
R1	Secure communication & transmission of data	n.a.
R2	Storage	Persisted inside VM
R3	Data Provisioning to Consumers	REST-API
R4	Tested Processing of large amounts of data (>100k msg/sec) backed by scalability and load balancing on all tiers - ingestion to extraction	n.a.
R5	Load throughput testing	n.a.
R6	Ensure platform-independence, platform should be deployable on commonly used OS	n.a.
R7	Ensure that platform can be deployed with beginner knowledge	~ long, detailed guide available
R8	Guide for Deployment & Usage	~ long, detailed guide available
R9	Multitenancy	n.a.
R10	Support Ingestion via MQTT	n.a.

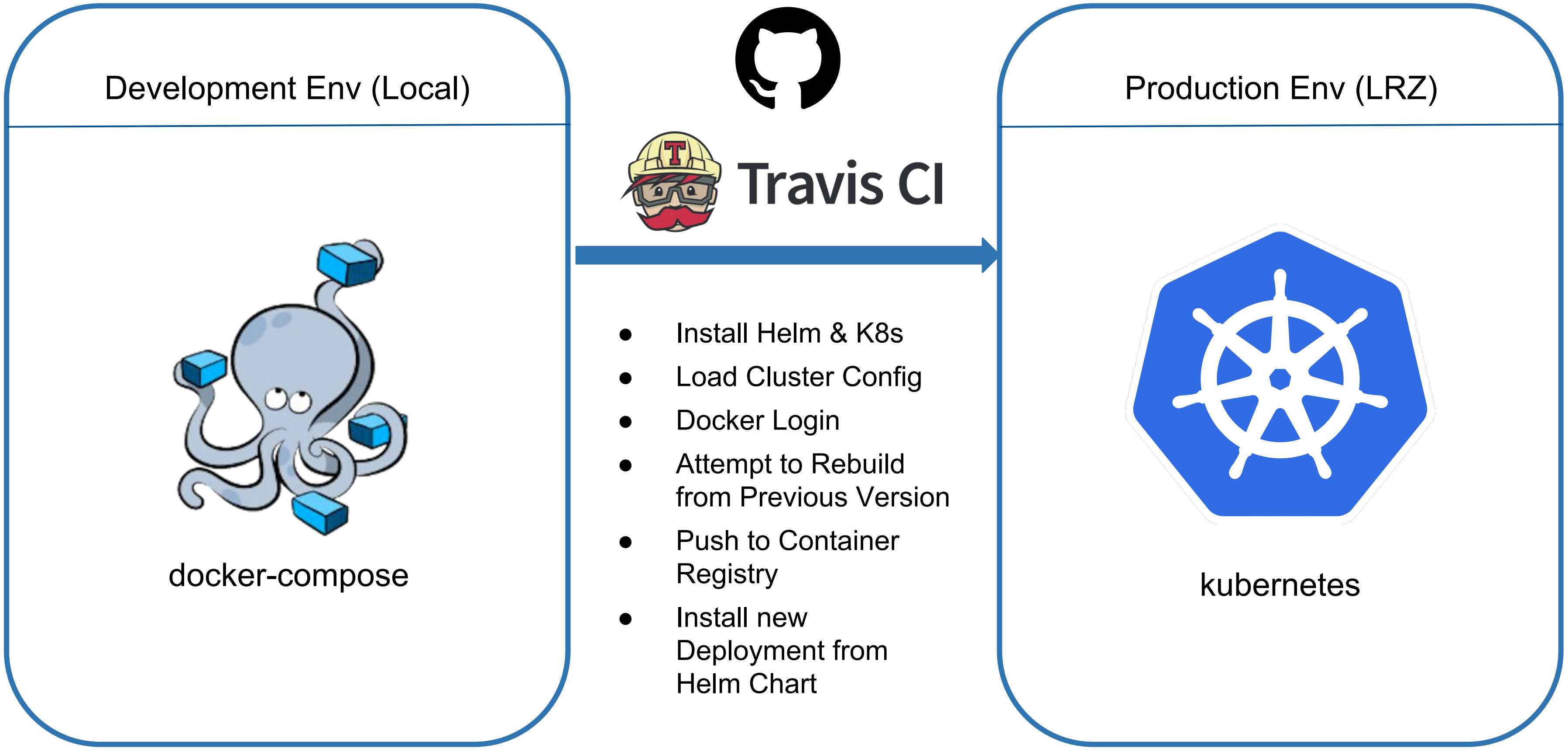
Architecture Design



Architecture Design - Pipeline

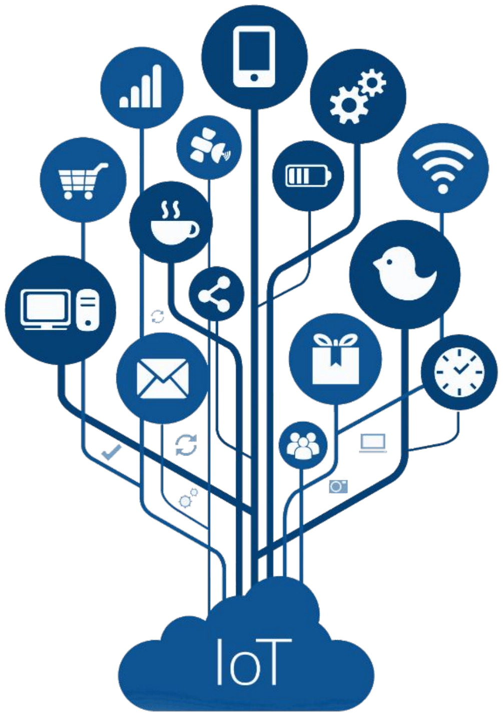


Architecture Design - Deployment



- Install Helm & K8s
- Load Cluster Config
- Docker Login
- Attempt to Rebuild from Previous Version
- Push to Container Registry
- Install new Deployment from Helm Chart

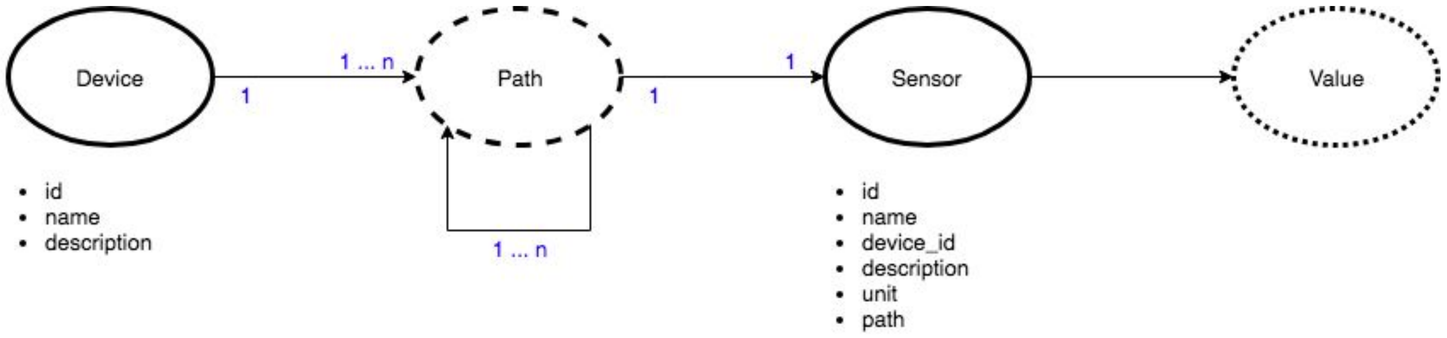
Data Model



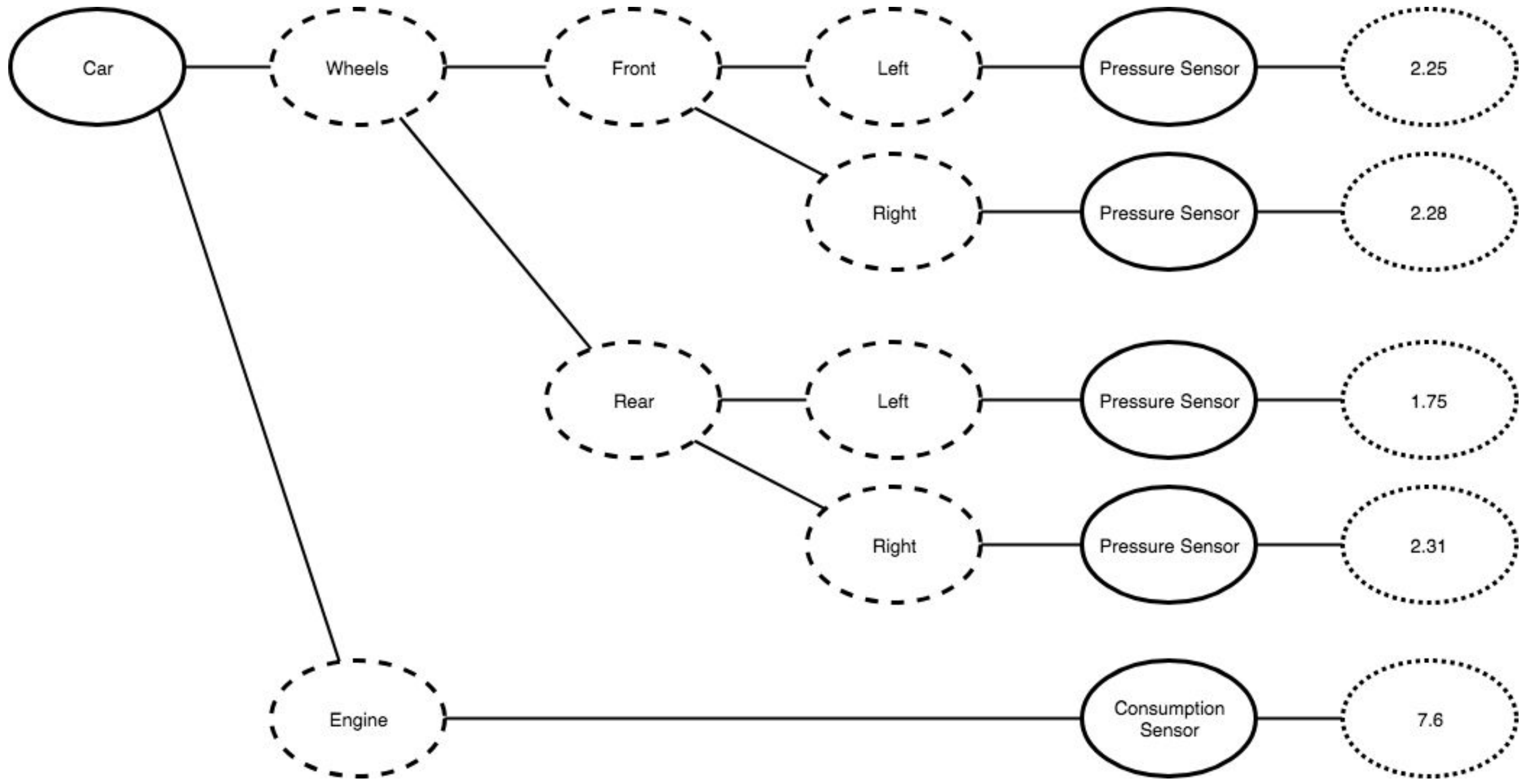
Data Model - Device-/Sensor Model

- Our Intent
  - Support of generic devices
  - Enable (optional) multi level nesting of sensors

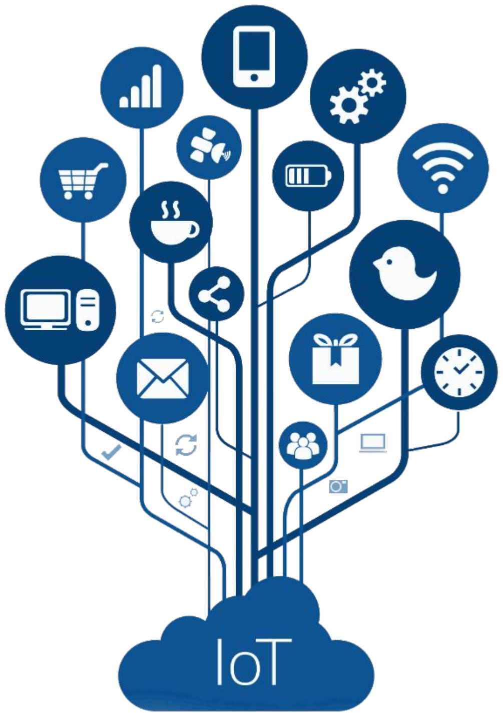
- Device
  - Entity that may contain one or many sensors and represents a single physical logical unit
- Sensor
  - Entity that belongs to a device
  - Represents a single physical measuring point, producing a time series of data



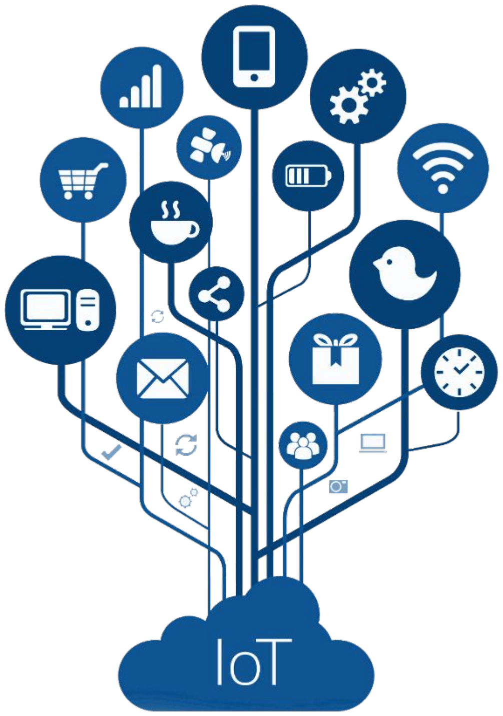
Data Model - Example



Live Demo



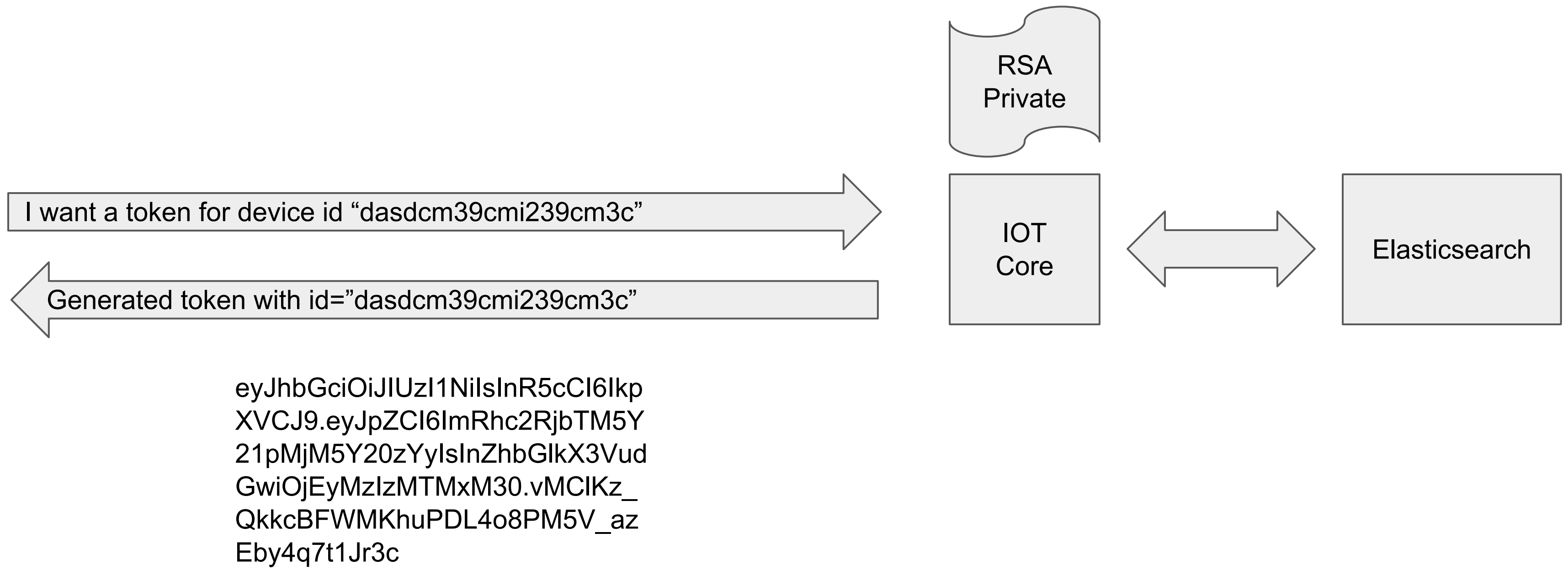
Security



Security

Security via JWT (JSON Web Token, IETF RFC7519 standard) [https://tools.ietf.org/html/rfc7519]

Retrieving JWT Tokens:

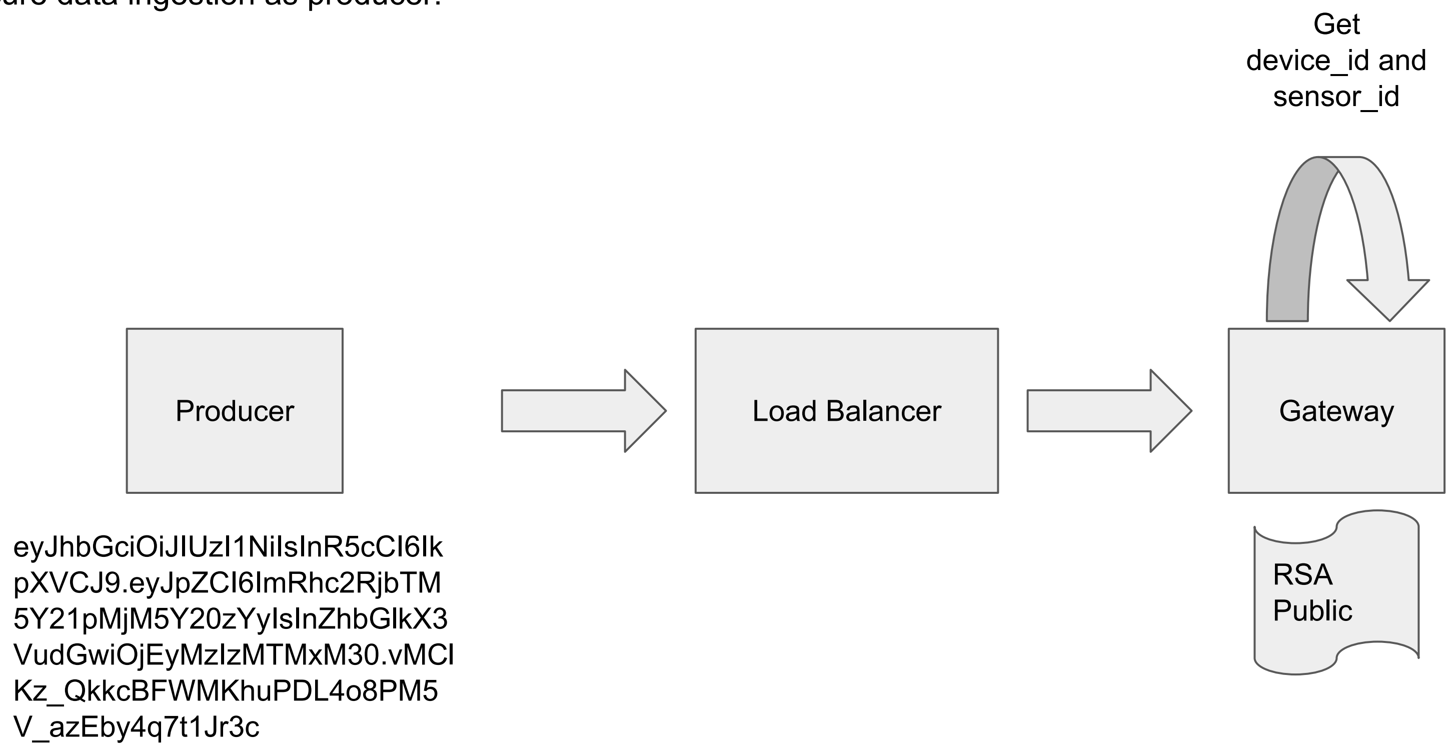

RSA Private
I want a token for device id “dasdcm39cmi239cm3c”
Generated token with id=”dasdcm39cmi239cm3c”
IOT Core
Elasticsearch

eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp
XVCJ9.eyJpZCI6ImRhc2RjbTM5Y
21pMjM5Y20zYyIsInZhbGlkX3Vud
GwiOjEyMzIzMTMxM30.vMCIKz_
QkkcBFWMKhuPDL4o8PM5V_az
Eby4q7t1Jr3c

Security

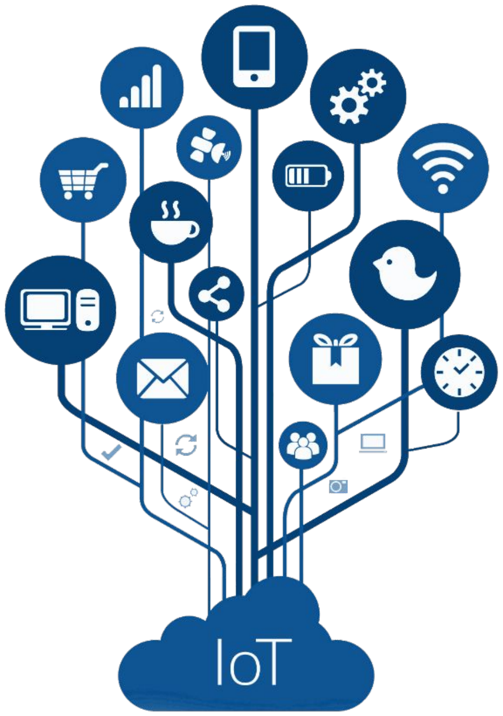
Security via JWT (JSON Web Token, IETF RFC7519 standard) [https://tools.ietf.org/html/rfc7519]

Secure data ingestion as producer:



eyJhbGciOiJIUzI1NiIsInR5cCI6Ik
pXVCJ9.eyJpZCI6ImRhc2RjbTM
5Y21pMjM5Y20zYyIsInZhbGlkX3
VudGwiOjEyMzIzMTMxM30.vMCl
Kz_QkkcBFWMKhuPDL4o8PM5
V_azEby4q7t1Jr3c

Benchmarks



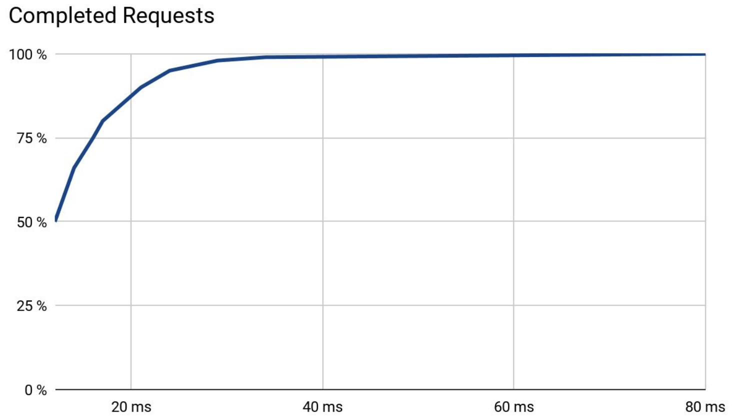
Benchmarks - HTTP - w/ Auth

Performance on Hetzner, cx41: 4 vCPUs, 16Gb of RAM

Go

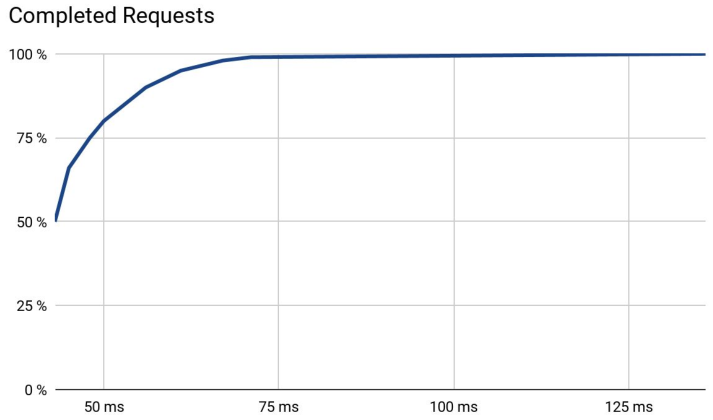
Parameters:
- 100 devices
- 200000 msg
- Max-Throughput

Statistics:
- Mean: 13.236 ms
- Max: 80 ms
- Req/s: 7554.87



Parameters:
- 100 devices
- 200000 msg
- Max-Throughput

Statistics:
- Mean: 44.942 ms
- Max: 136 ms
- Req/s: 2205.08



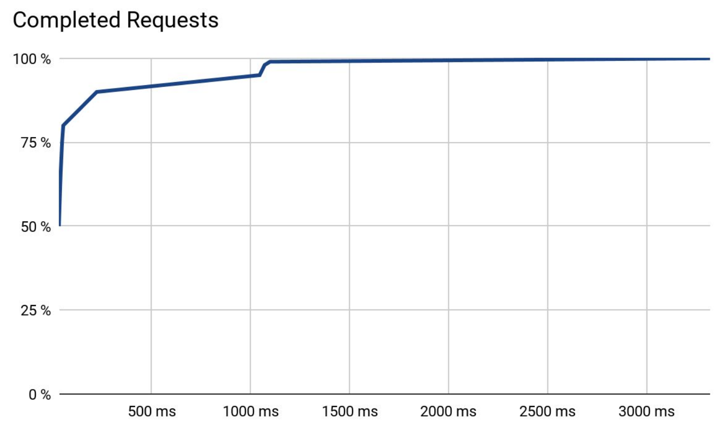
Benchmarks - HTTP - w/ Auth

Performance on Hetzner, cx41: 4 vCPUs, 16Gb of RAM

Go

Parameters:
- 1000 devices
- 100000 msg
- Max-Throughput

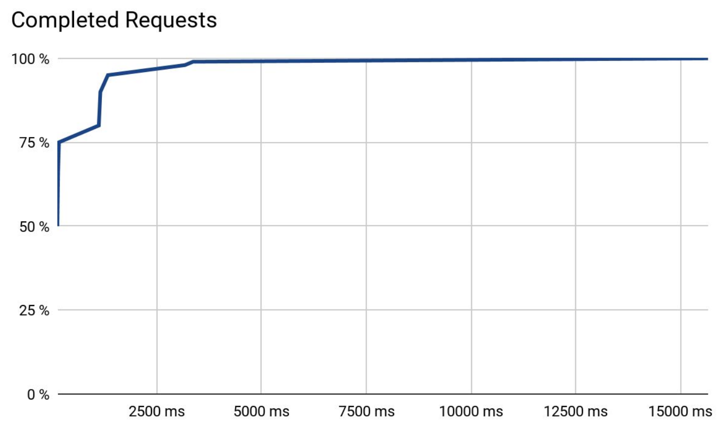
Statistics:
- Mean: 131.057 ms
- Max: 3321 ms
- Req/s: 7630.28



node JS

Parameters:
- 1000 devices
- 100000 msg
- Max-Throughput

Statistics:
- Mean: 452.459 ms
- Max: 15662 ms
- Req/s: 2210.14



Benchmarks - WebSockets - w/ Auth

Performance on Hetzner, cx41: 4 vCPUs, 16Gb of RAM

Go

Parameters:

- 1000 devices
- 1000 msg/device
- Connections pre-established

Statistics:

- Mean: 0.542 ms
- Max: 923 ms
- Req/s: 800319.53

node JS

Parameters:

- 1000 devices
- 1000 msg/device
- Connections pre-established

Statistics:

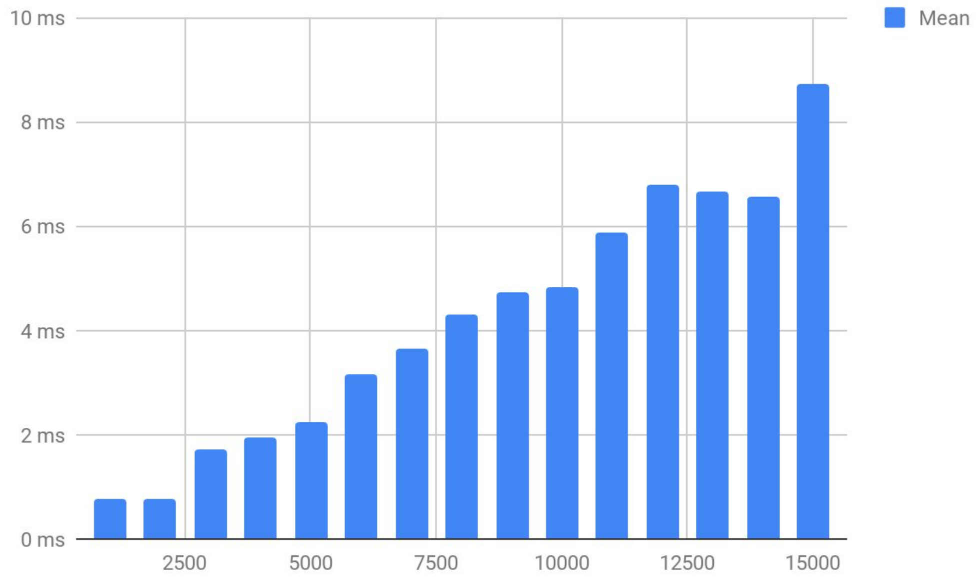
- Mean: 0.549 ms
- Max: 726 ms
- Req/s: 783266.05

Benchmarks - WebSockets - Golang, w/ Auth

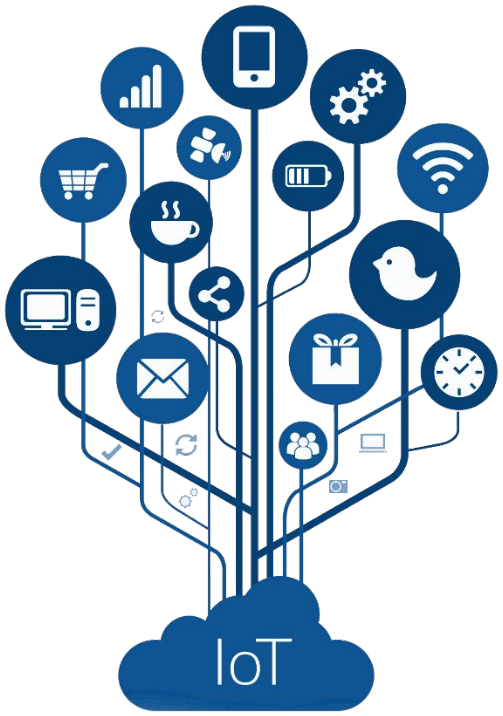
Performance on Hetzner, cx41: 4 vCPUs, 16Gb of RAM

Go

Parameters:
- n devices
- 1000 msg/device
- Connections pre-established



Comparison to Existing Solutions



Comparison to Existing Solutions - KAA

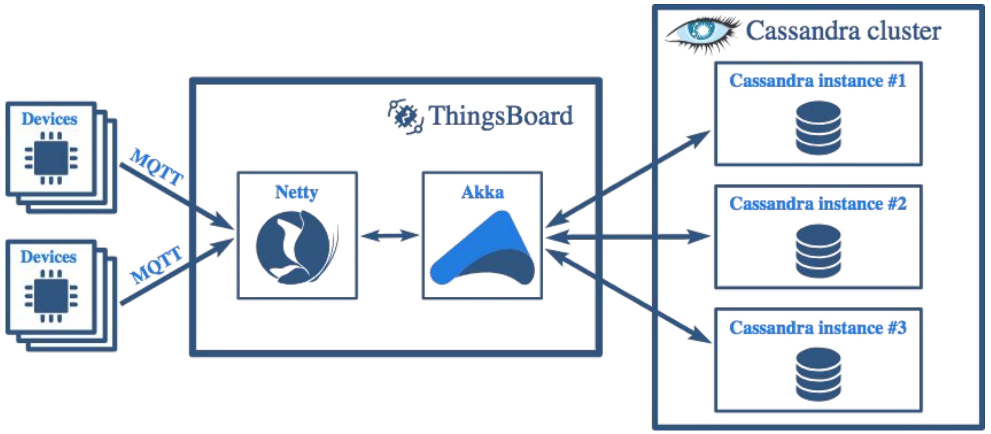
FEATURE	AVOCADO ARCHIPELAGO	BANANA BEACH (Early Access Only)
Architecture	Monolithic	Microservices
Connectivity protocol	Proprietary	Open, standards-based
Gateway connectivity model	One connection per device	Single, multiplexed connection
Communication security	RSA+AES	(D)TLS
Device credential management	No	Yes
Device metadata	Structured	Structured or unstructured
Data collection	Single data type, structured only	Unlimited types, isolated flows, structured or unstructured
Configuration management	Structured only	Structured or unstructured
Data processing and analytics	3-rd party integrations	Built-in or 3-rd party integrations
Data visualization	3-rd party integrations	Built-in customizable dashboards or 3-rd party integrations
Device notifications	Yes	No, superseded by commands
Command execution	No	Yes
Over-the-air updates	No	Yes
Technology stack	Mainly Java	Polylingual
Scalability, elasticity, self-healing	Manual	Automated container orchestration
Server configuration	Non-portable, stored in DB	Portable declarative blueprint

[https://www.kaaproject.org/whats-new/]

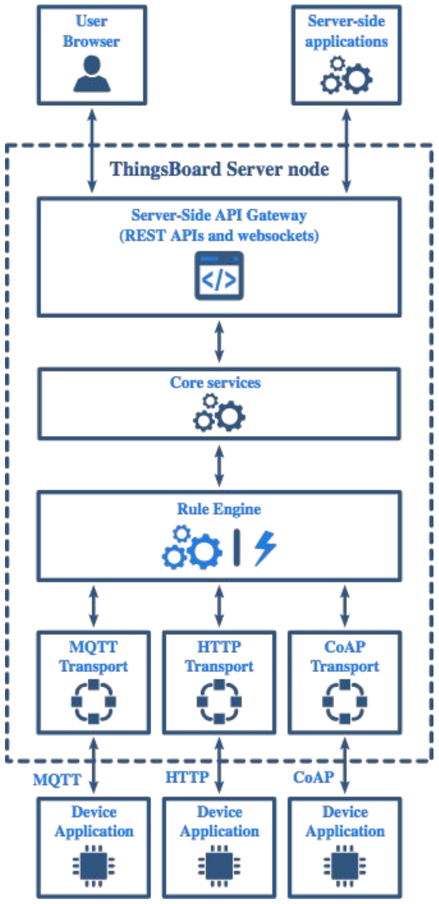
Comparison to Existing Solutions - ThingsBoard

Core Services:

- Device and credentials
- Rule chains and rule nodes
- Tenants and customers
- Widgets and dashboards
- Alarms and events



[https://thingsboard.io/docs/reference/performance/]



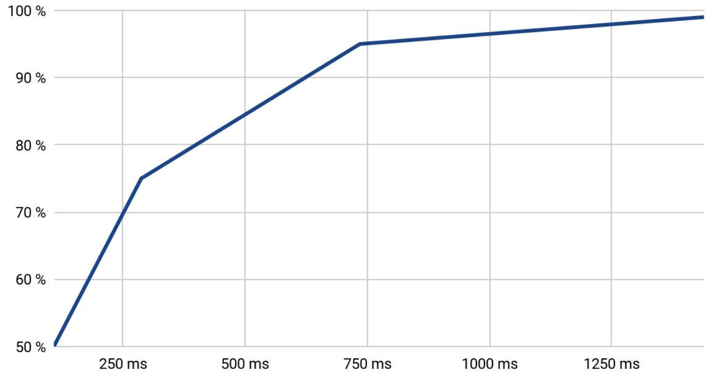
[https://thingsboard.io/docs/reference/architecture/]

Comparison to Existing Solutions - ThingsBoard

AWS, c4.2xlarge: 4 vCPUs, 7.5Gb of RAM
- MQTT
- 10000 devices
- 1 msg/sec/device
- total load: 10000 msg/sec

Statistics:
- Mean: 217 ms
- Max: 10887 ms
- Req/s: 6818.182

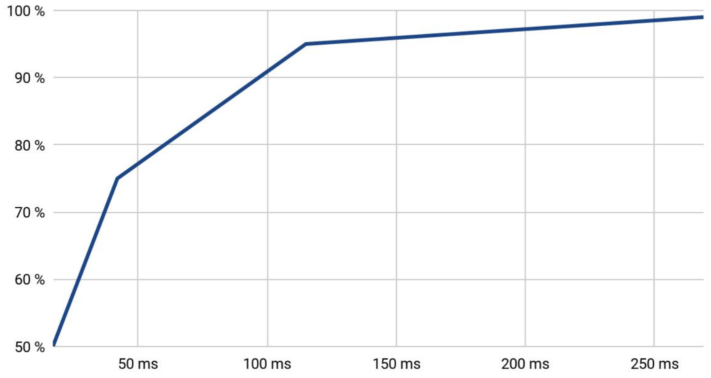
Completed Requests



AWS, c4.2xlarge: 8 vCPUs, 15Gb of RAM
- MQTT
- 10000 devices
- 1 msg/sec/device
- total load: 10000 msg/sec

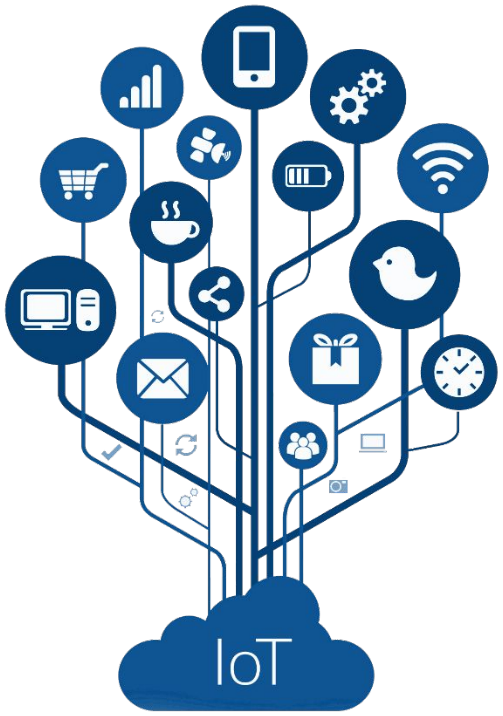
Statistics:
- Mean: 38 ms
- Max: 3270 ms
- Req/s: 8823.529

Completed Requests



[https://thingsboard.io/docs/reference/performance/]

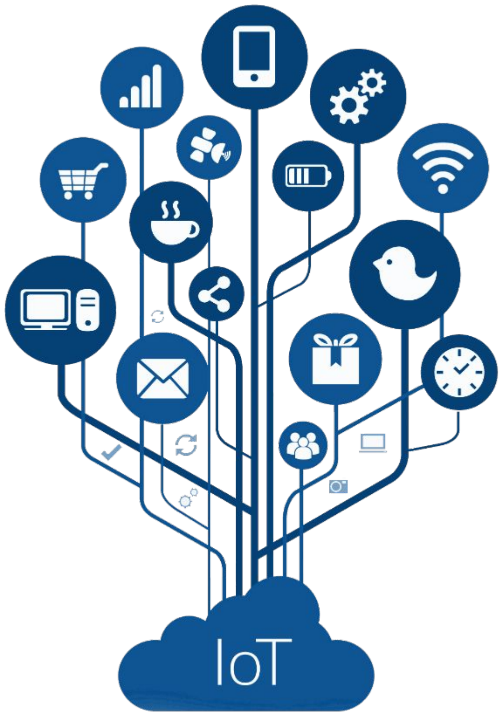
Misc



Misc

adsf

Conclusion, Challenges & Outlook



Conclusion

	Requirement	Current Status
R1	Secure communication & transmission of data	✔ Messaging & device administration secured via JWT
R2	Storage	✔ Persisted in ES container within cluster
R3	Data provisioning to consumers	✔ Secure REST-API to ES
R4	Processing large amounts of data, scalability in all tiers - ingestion to extraction	✔ Large amounts of data ~ Autoscaling capabilities via K8s
R5	Load throughput testing	✔ Gateway capabilities tested + language comparisons
R6	Ensure platform-independence	✔ Guaranteed thanks to dockerized application design
R7	Ensure accessibility with beginner knowledge	✔ As easy as “docker-compose up”
R8	Guide for deployment & usage	✔ Detailed Github Readme + docs
R9	Multitenancy	✔ Private tokens for devices
R10	Support ingestion via MQTT	✔ MQTT, HTTP & WS gateways available

Challenges

- Team members had to evolve an understanding of the considerable technology stack
- Ambitious scope
- Opinion: previous architecture design not suited for the requirements given to us
  - Almost inevitable to redesign and reimplement, adding considerable workload
- Cooperation with HAL team
  - Difficult since they have to rely on running architecture, which is hard if it is being reworked
  - Integration now possible
- JWT Authorization - Node.js library converts HTTP headers to lower case
  - Either: loop over raw headers (sacking performance), change header (abusing the standard)

Outlook

- Activate true persistence, surviving rolling deployments (only Flink missing)
- Finish up on autoscaling (Kafka missing)
- Improve frontend UX
- Security testing
- Actuator expansion (e.g. connected to Flink)
- Provide more default Flink jobs for analytics
- MQTT performance testing