



**OST**

Ostschweizer  
Fachhochschule

# Digital Manufacturing

## 04 – Cloud Computing

Lukas Kretschmar, MSc FHO in Engineering (ICT & BEP)

15. November 2021

Industrial Engineering (OST-RJ)

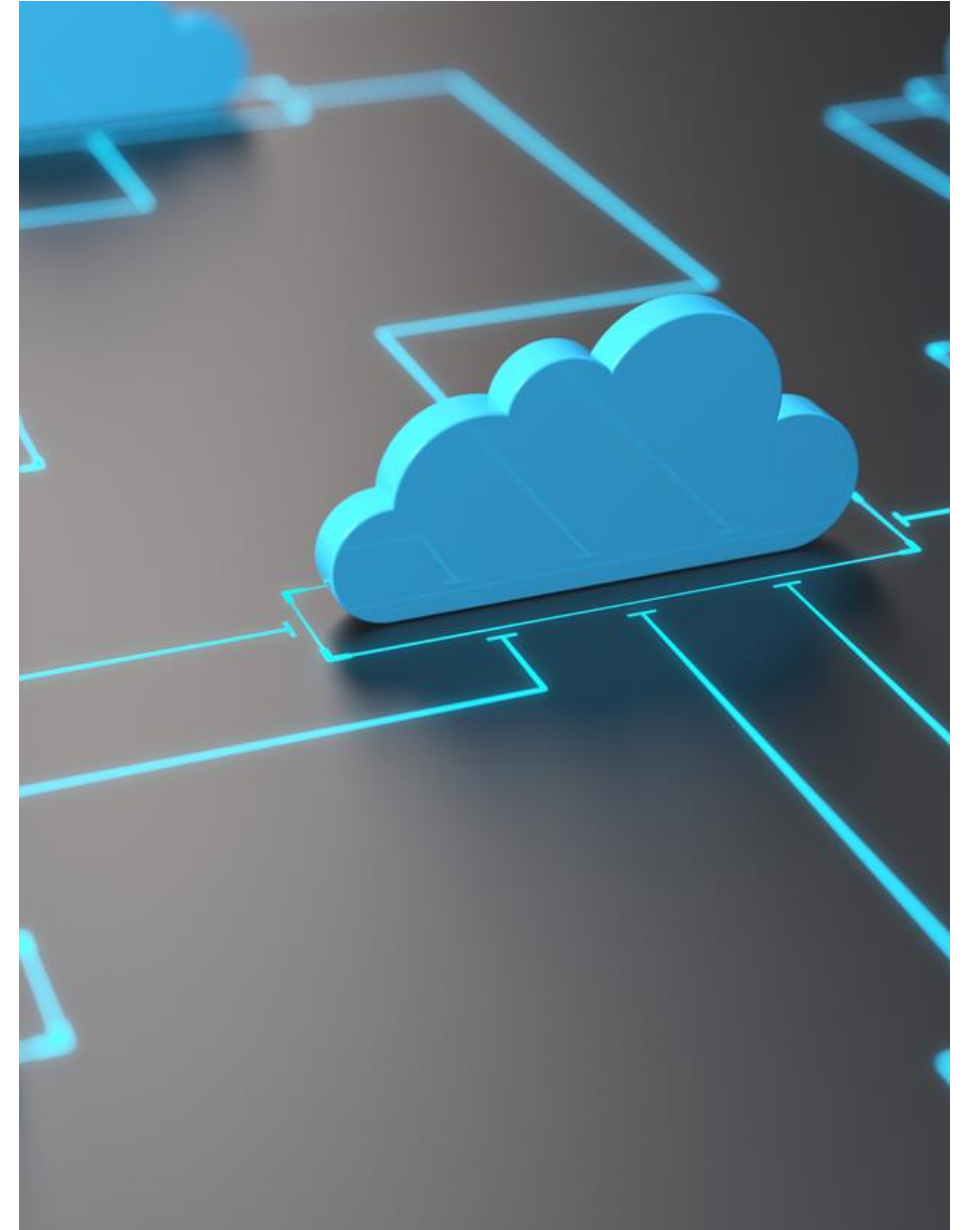
# Startup



# Digital Manufacturing

## Today's Objectives

- You can use a REST API
- You can identify the service concepts of cloud providers
- You know the basic pros and cons about the cloud
- You have a basic understanding of key security concepts
- You can identify the parts of JWT



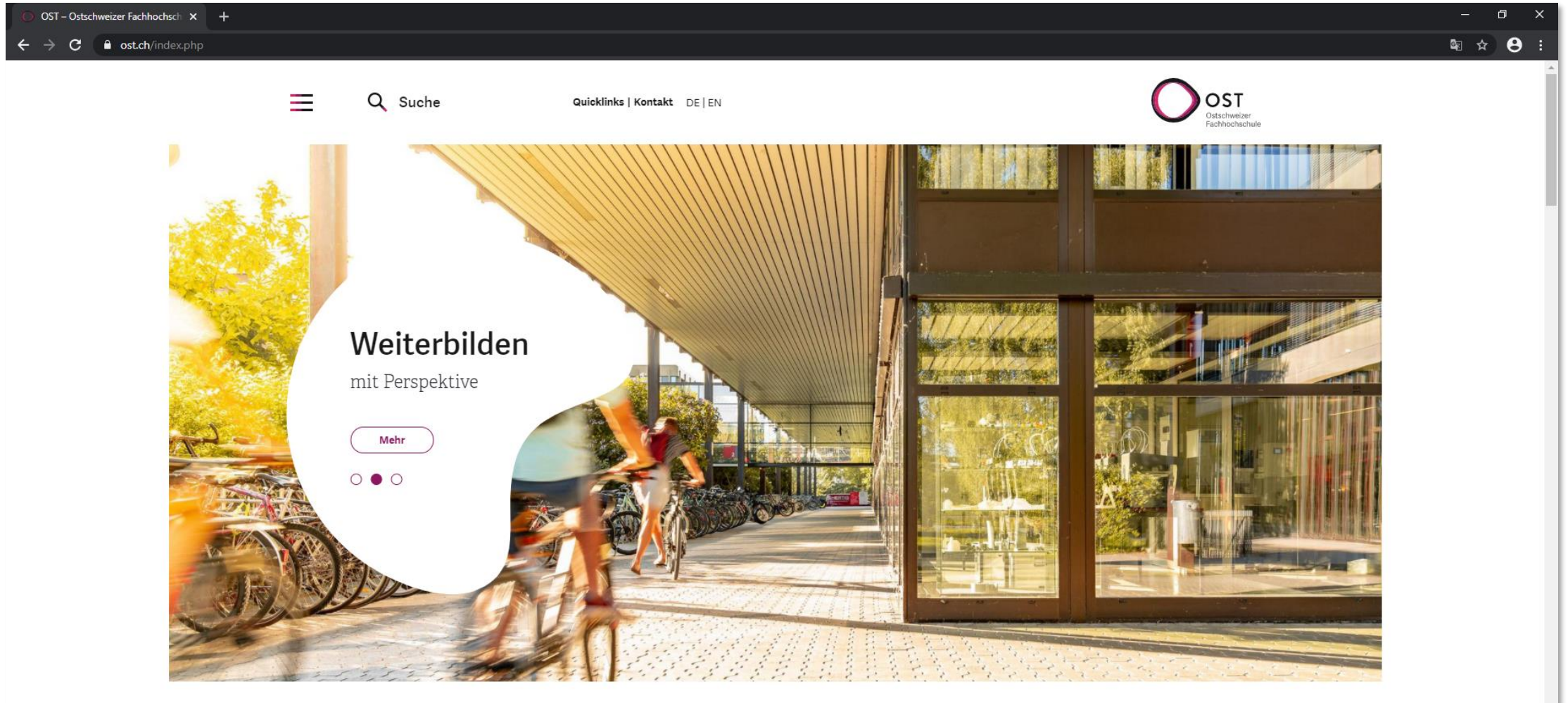
# How the Internet works

200 OK



# How the Internet works

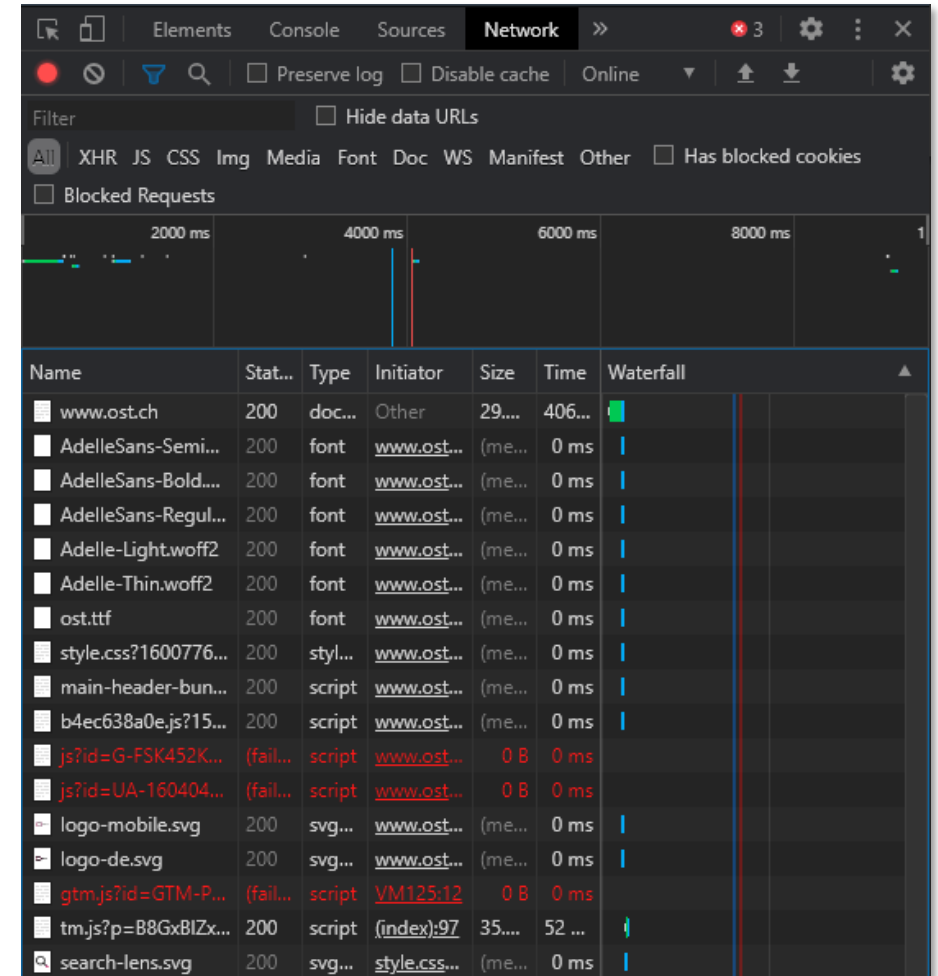
<http://www.hsr.ch> → <https://www.ost.ch>



## How the Internet works

# Hands-On: <http://www.hsr.ch> → <https://www.ost.ch>

- Let's see what happens when we open <http://www.hsr.ch>
- And land on <https://www.ost.ch>
- To see the communication, we need to get behind the scenes
  - Nowadays, every browser is shipped with some features to see the communication
  - Usually, they are called *Developer Tools* or something in that direction

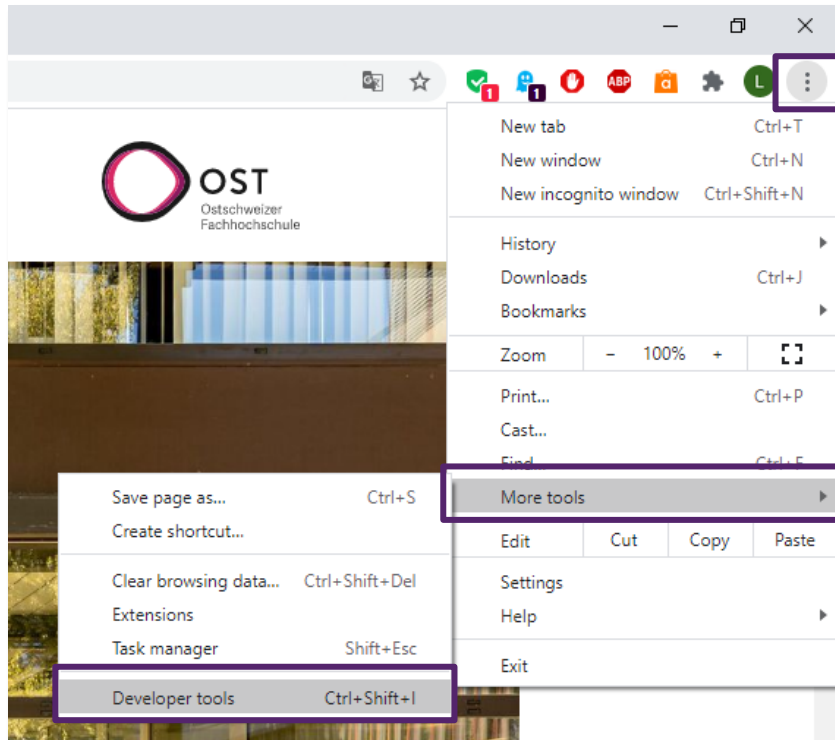


## How the Internet works

# Hands-On: <http://www.hsr.ch> → <https://www.ost.ch>

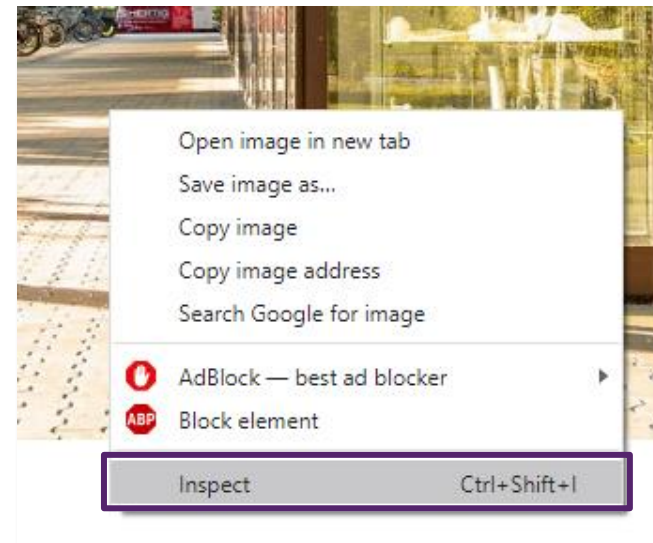
### Open them through the menu

- Usually, they use the keyboard combination of *Shift + Ctrl + I*



### Inspect a websites content

- Click on the page with the right mouse button
- Use the menu item Inspect or Inspect Element



# How the Internet works

## Hands-On: <http://www.hsr.ch> → <https://www.ost.ch>

This screenshot shows the 'Headers' tab of a browser's developer tools. The 'General' section displays the following information:

- Request URL:** `http://www.hsr.ch/`
- Request Method:** `GET`
- Status Code:** `301 Moved Permanently`
- Remote Address:** `152.96.36.83:80`
- Referrer Policy:** `no-referrer-when-downgrade`

The 'Response Headers' section shows:

- Connection:** `keep-alive`
- Content-Length:** `162`
- Content-Type:** `text/html`
- Date:** `Tue, 08 Sep 2020 06:18:05 GMT`
- Location:** `https://www.hsr.ch/`
- Server:** `nginx`

A red dashed arrow points from the 'Location' header to the right-hand screenshot.

This screenshot shows the 'Headers' tab of a browser's developer tools. The 'General' section displays the following information:

- Request URL:** `https://www.hsr.ch/`
- Request Method:** `GET`
- Status Code:** `301`
- Remote Address:** `152.96.36.83:443`
- Referrer Policy:** `no-referrer-when-downgrade`

The 'Response Headers' section shows:

- access-control-allow-origin:** `www.hsr.ch`
- content-length:** `230`
- content-type:** `text/html; charset=iso-8859-1`
- date:** `Tue, 08 Sep 2020 06:18:05 GMT`
- location:** `https://www.hsr.ch/de/`
- server:** `nginx`
- status:** `301`
- strict-transport-security:** `max-age=63072000; includeSubdomains; preload`
- x-cache-status:** `HIT`

A red dashed arrow points from the 'location' header to the right.

# How the Internet works

## Hands-On: <http://www.hsr.ch> → <https://www.ost.ch>

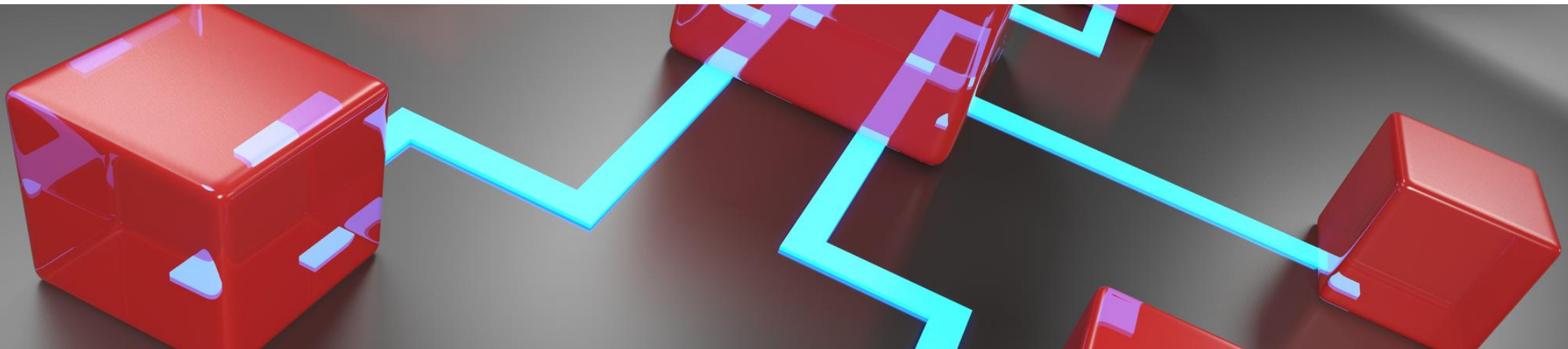
The image shows two network request details panels from a browser's developer tools. The left panel shows a request to `https://www.hsr.ch/de/` with a status code of 301 and a location header pointing to `https://www.ost.ch/index.php`. The right panel shows a request to `https://www.ost.ch/index.php` with a status code of 200 and a remote address of `146.136.105.52:443`. A red dashed arrow indicates the redirection from the first URL to the second. A red arrow points from the text 'So, the Website is hosted in Buchs' to the remote address of the second request.

Request	Request URL	Request Method	Status Code	Remote Address	Location / Response Headers
www.hsr.ch	<code>https://www.hsr.ch/de/</code>	GET	301	<code>152.96.36.83:443</code>	<code>location: https://www.ost.ch/index.php</code>
www.ost.ch	<code>https://www.ost.ch/index.php</code>	GET	200	<code>146.136.105.52:443</code>	cache-control, content-encoding, content-language, content-length, content-type, date, expires, server

So, the Website is hosted in Buchs

# API (Application Programming Interface)

## Talking to Services



# API

## What's an API?

- Don't be confused by the term programming
  - You don't program anything
  - Interface is the term to focus on
- APIs are the entry point to applications
  - Applications offer features through their APIs
  - Applications work together by using their APIs
- APIs are for applications, what GUIs (graphical user interface) are for humans
- An API allows you to
  - Access & manipulate data
  - Trigger actions
  - Hide the implementation
  - Limit the possibilities what you can do with an application
  - Enable modularization of systems
    - SOA (Service-oriented architecture)
    - Applications talking to other applications that talk to other applications

# API

## REST

- Representational State Transfer (REST)
  - Simple client-service architecture
  - Builds upon the HTTP protocol
  - Uses JSON
  - Does not keep a state
  - Endpoints can change
    - Subsequent calls do not necessarily need to be handled by the same instance
- Current state-of-the-art type of webservices

The screenshot shows the Swagger UI for the 'Data API v1'. The interface is clean and organized, with a green header bar containing the 'swagger' logo and a dropdown menu for selecting a specification, currently set to 'Data API'. Below the header, the API title 'Data API v1' is displayed, along with a link to the Swagger JSON file and a brief description: 'A simple REST API to store and load data as JSON. Contact Lukas Kretschmar MIT'. The main content area is divided into two sections: 'Database' and 'Home'. The 'Database' section lists several endpoints with their respective HTTP methods and descriptions: a GET endpoint for loading all entries, a POST endpoint for storing JSON data, a GET endpoint for loading data by ID, and two DELETE endpoints for removing containers and their entries. The 'Home' section lists two GET endpoints: one for testing API reachability and one for retrieving the license file.

[https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)

# API

## SOAP

Acronym has been dropped in v1.2



- “Simple Object Access Protocol”
  - Protocol independent
  - Service description and messages are defined in XML
    - XML generates overhead
    - But distinguishes between data and metadata
  - Service description can be exchanged (Metadata Exchange)
    - Service can be generated on the client side
    - Service description is also in XML
- More common between applications within an intranet (local network)
  - OPC-UA (OPC Unified Architecture)

<https://en.wikipedia.org/wiki/SOAP>

## OPC-UA (Open Platform Communications Unified Architecture)

- OPC Unified Architecture
  - Open standard for M2M communication
  - Cross platform
  - Using SOAP
    - Integrated Security
    - Integral information model
      - Faults are part of the model
  - Supported protocols
    - `opc.tcp://` *Binary*
    - `http://` *Plaintext*
- It's really complex
  - Everything is specified
  - System integration is easier
  - But we still rely on the implementation and usage of the vendor



- Request & Response look the same as with HTTP
- Content is in XML

```
1 <?xml version="1.0"?>
2 <soap:Envelope
3     xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
4     xmlns:m="http://www.example.org">
5     <soap:Header>
6     </soap:Header>
7     <soap:Body>
8     </soap:Body>
9 </soap:Envelope>
```

[https://en.wikipedia.org/wiki/OPC\\_Unified\\_Architecture](https://en.wikipedia.org/wiki/OPC_Unified_Architecture)

# API

## HTTP Methods

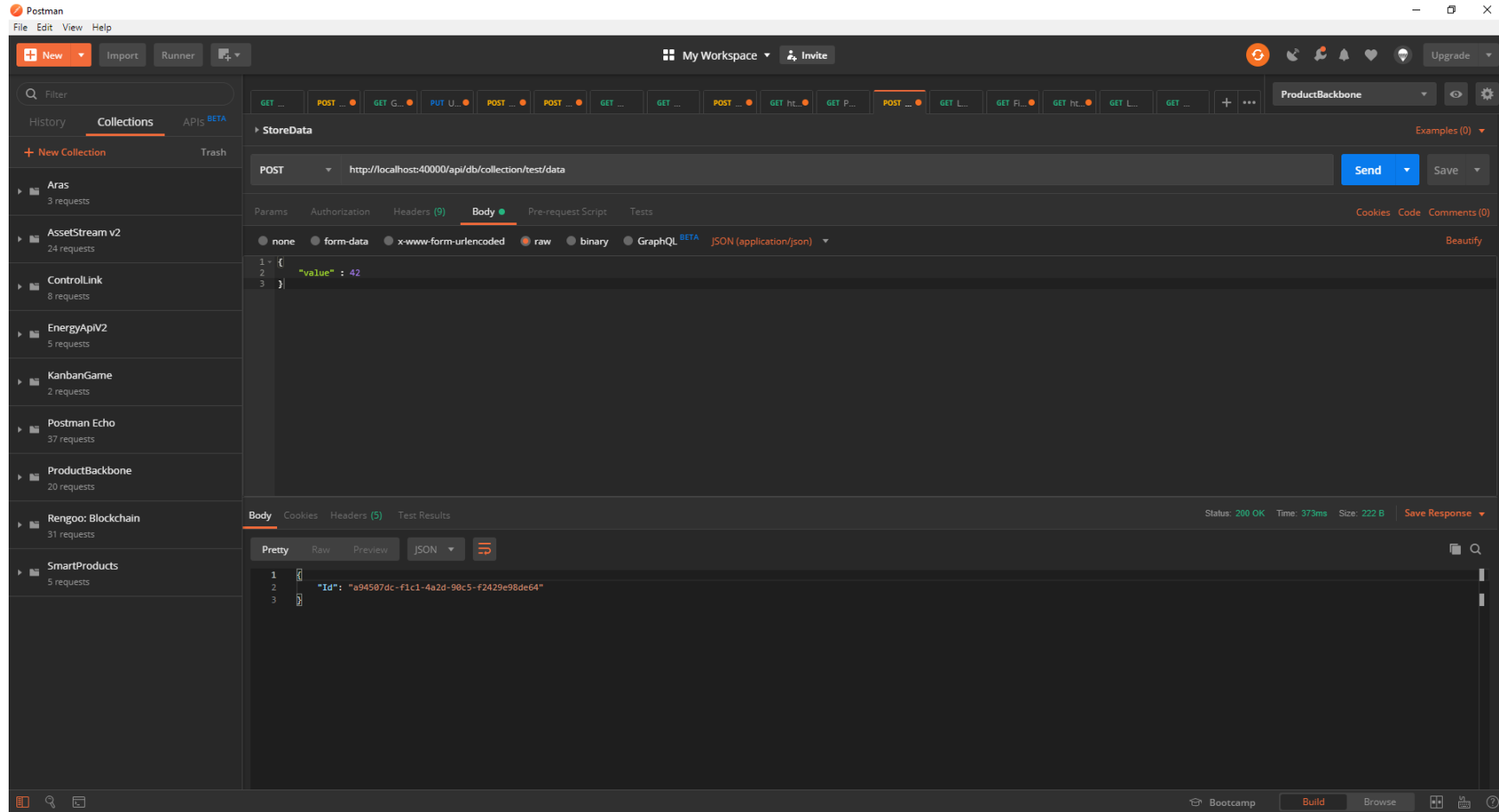
- GET
  - Requesting data from an URI
- POST
  - Store data beneath the URI
- PUT
  - Store/replace data at URI
- DELETE
  - Remove data at URI

Database	
GET	<code>/api/db/collection/{containerid}/data</code> Loads all entries of a container within the provided timestamps.
POST	<code>/api/db/collection/{containerid}/data</code> Stores JSON data into a container.
GET	<code>/api/db/collection/{containerid}/data/{dataid}</code> Loads data by its id (GUID) from a specific container.
DELETE	<code>/api/db/collection/{containerid}/data/{dataid}</code> Removed container and all its entries.

*There are some more,  
but not relevant here*

# API

# Postman



<https://www.postman.com>



# API

## Reading Data

The screenshot shows a REST client interface with the following components:

- Method:** GET
- Path:** `http://localhost:4000/api/db/collection/test/data`
- Query:** `?from=2019-10-16T14:53:16.316Z&to=2019-08-08T09:19:12.000Z`
- Status Code:** 200 OK
- Time:** 23ms
- Response (JSON):**

```
1 [{"Id": "a94507dc-f1c1-4a2d-90c5-f2429e98de64",
2   "Timestamp": "2019-10-16T14:53:16.316Z",
3   "Data": {
4     "value": 42
5   }
6 },
7 {"Id": "83ddecb2-e08d-4ab4-b07f-47936fc7235f",
8   "Timestamp": "2019-08-08T09:19:16.137Z",
9   "Data": {
10    "value": 42
11  }
12 },
13 {"Id": "6c4b7678-47a4-40d8-b88e-d9c64d80b472",
14   "Timestamp": "2019-08-08T09:19:12.523Z",
15   "Data": {
16    "value": 21
17  }
18 }
19 ]
```

# API

## Writing Data

The screenshot displays a REST client interface with the following details:

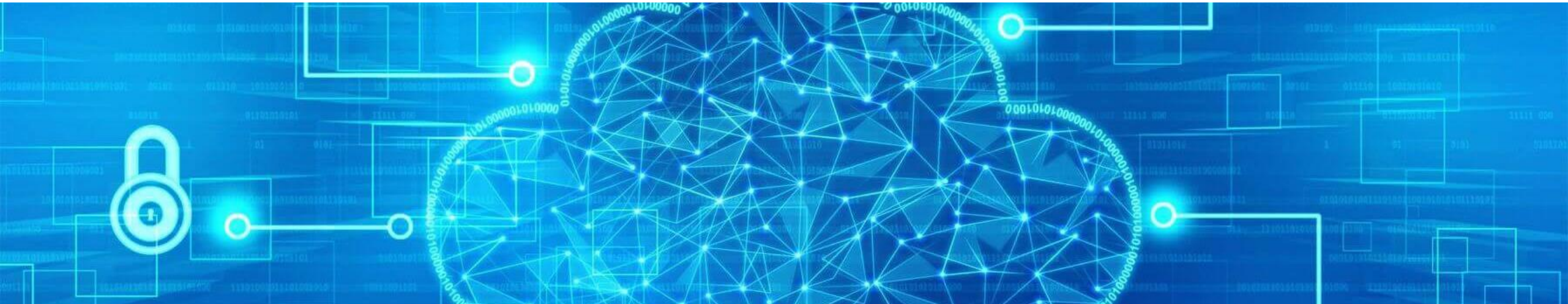
- Method:** POST
- Path:** http://localhost:40000/api/db/collection/test/data
- Content-Type:** JSON (application/json)
- Content (Request Body):**

```
{  
  "value" : 42  
}
```
- Status Code:** 200 OK
- Response (Response Body):**

```
{  
  "Id": "a94507dc-f1c1-4a2d-90c5-f2429e98de64"  
}
```

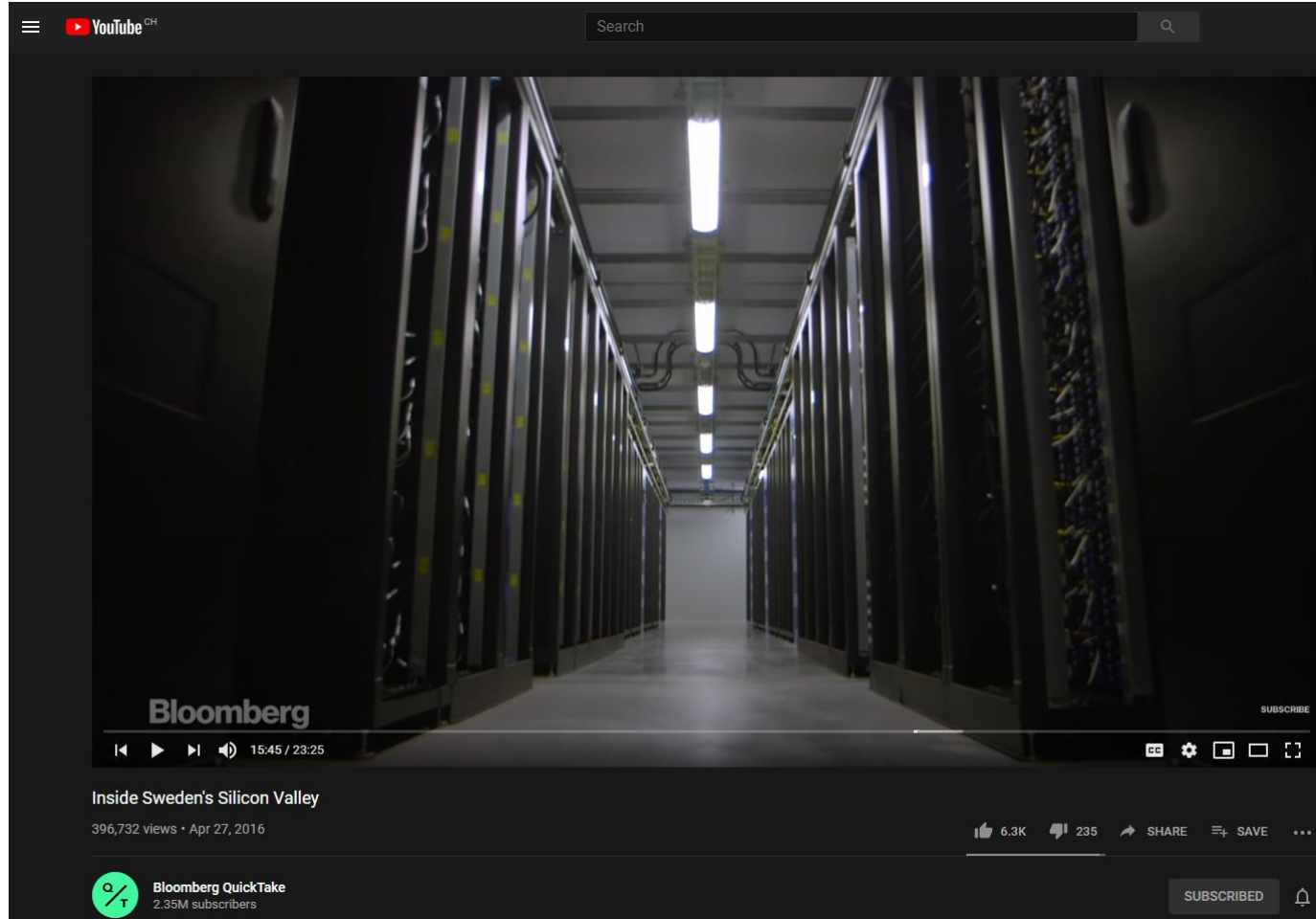
# The Cloud

It's just someone else's Computer



# The Cloud

## Hello Cloud



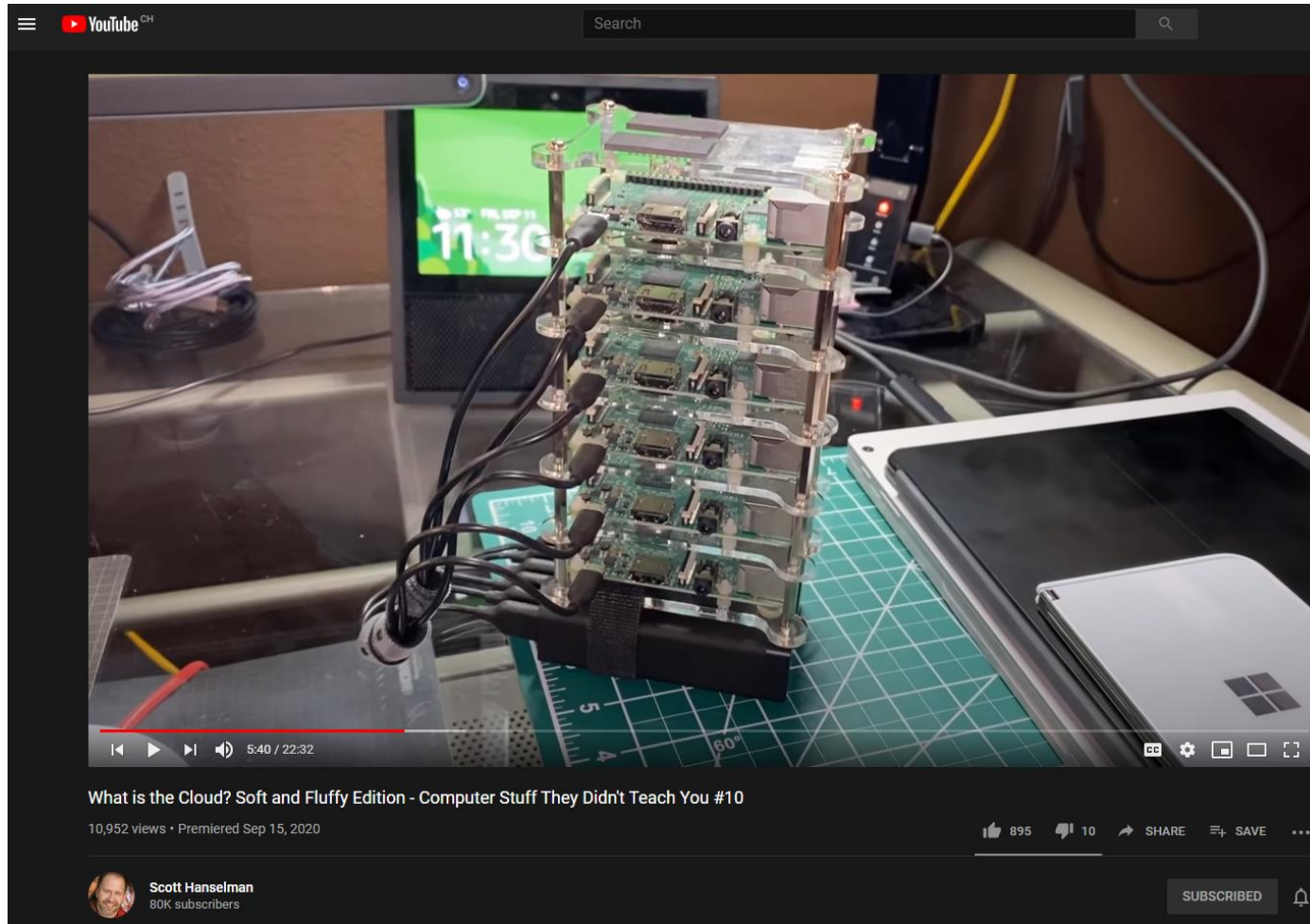
For those who are interested

<https://youtu.be/d915p1aXIkE?list=PLqq4LnWs3oIU-bP2R9uD8YXbt02JjocOk&t=858>

Bloomberg Hello World Playlist: <https://www.youtube.com/playlist?list=PLqq4LnWs3oIU-bP2R9uD8YXbt02JjocOk>

# The Cloud

## Introduction to the Cloud by Scott Hanselman

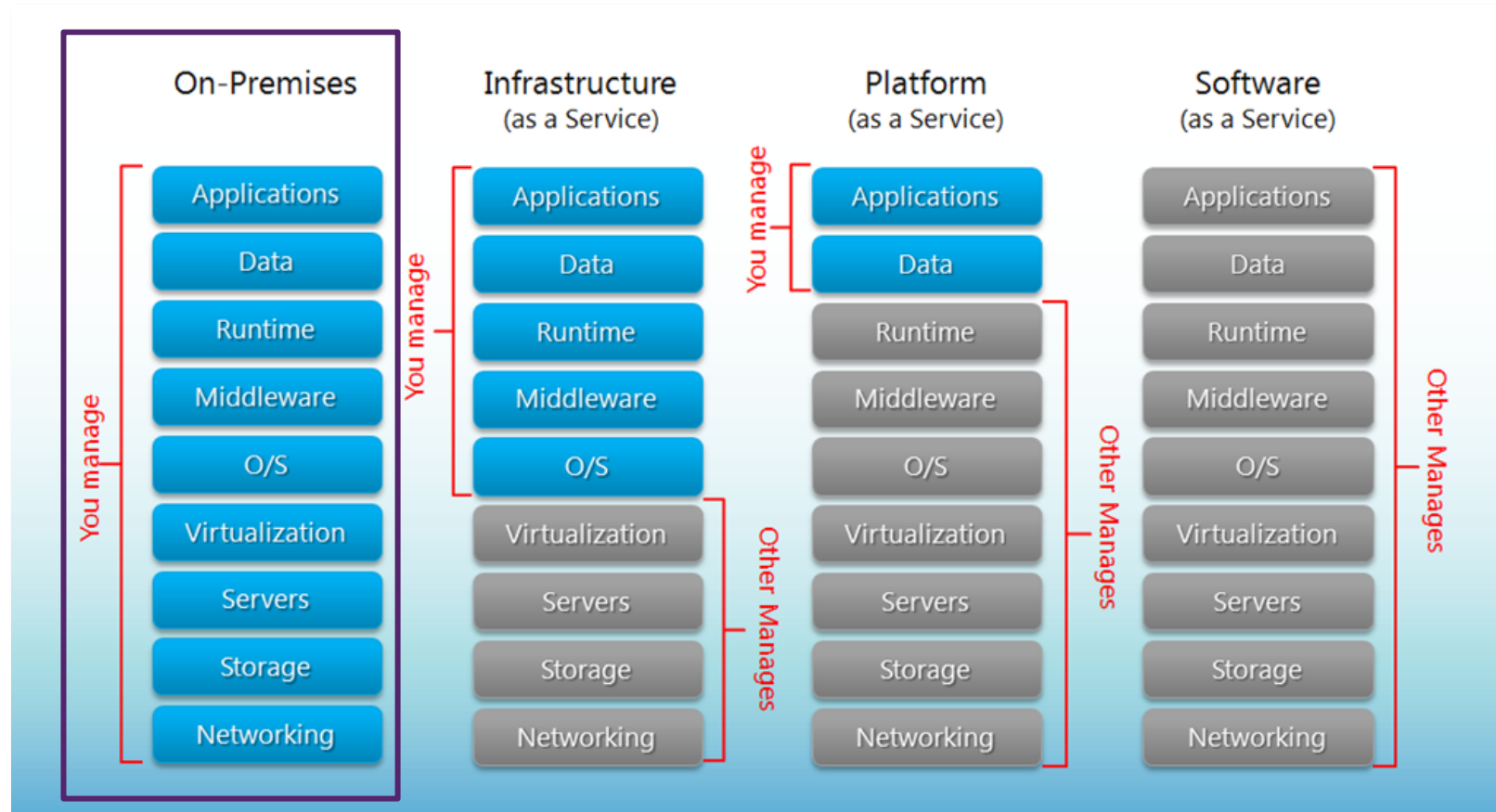


For self-study if interested

<https://www.youtube.com/watch?v=BO6jvQ88ICQ>

# The Cloud

## XaaS (Everything as a Service)



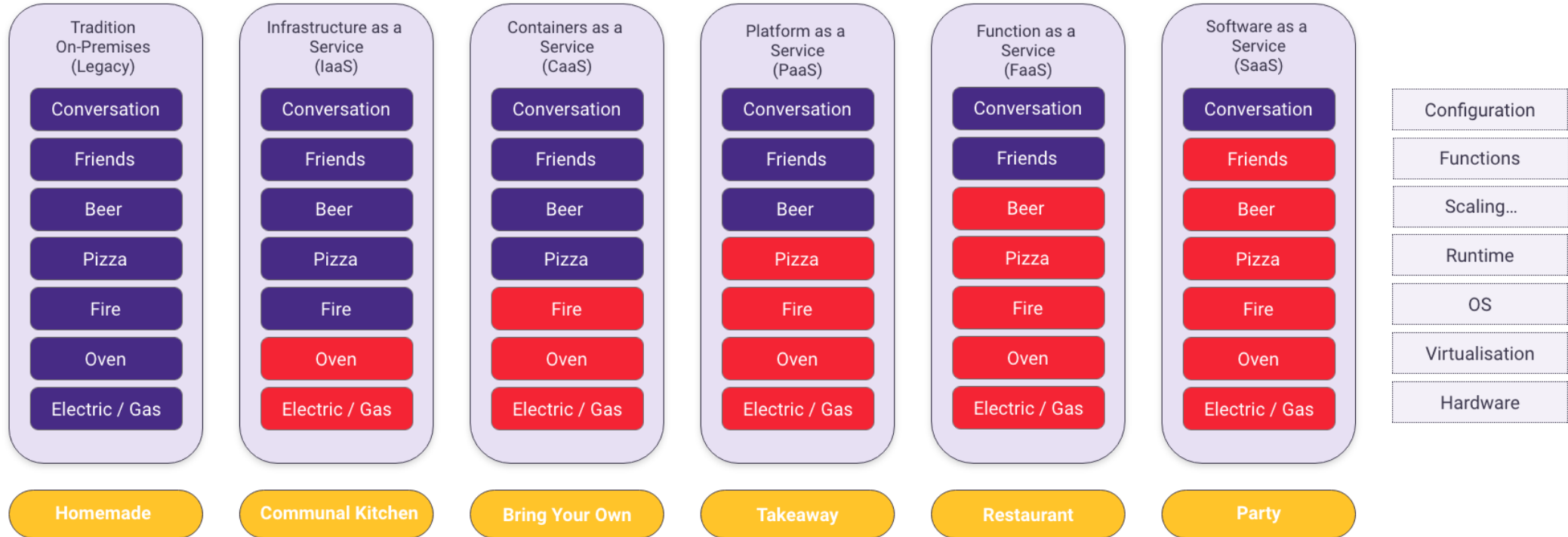
# The Cloud

## XaaS (Everything as a Service)

*Well - it also works like this*



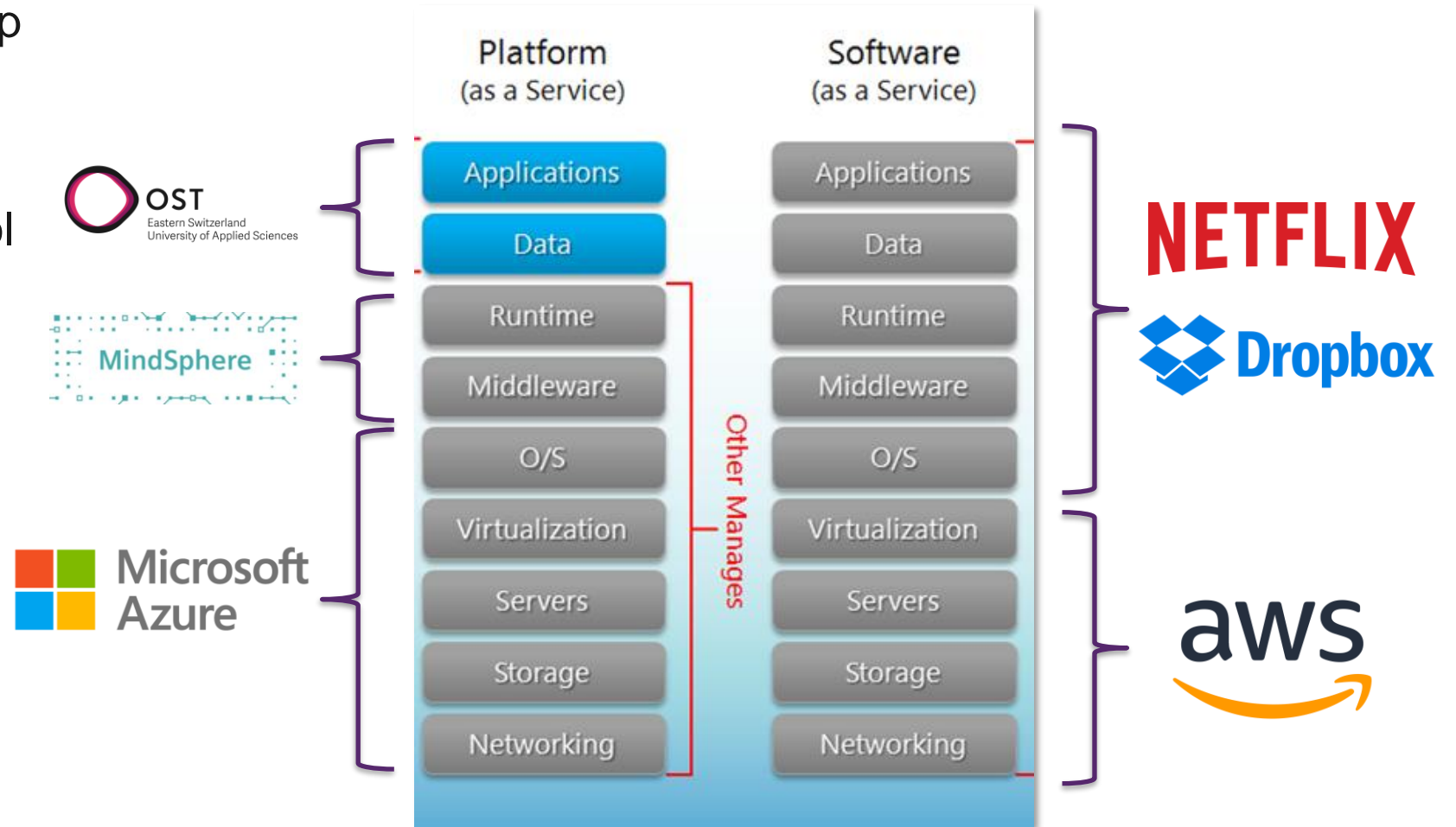
### PIZZA AS A SERVICE



# The Cloud

## XaaS (Everything as a Service)

- Services can build on top of each other
- Not every provider has every layer under control
  - May be managed by another provider



# Data Modeling for MindSphere

We need some Structure



# Data Modeling

## PlantUML

- <https://plantuml.com>
- Coding your Diagrams
  - Pro: Simple way to build your model
  - Cons: Limited flexibility on layout & design
- VS Code Extension
  - <https://marketplace.visualstudio.com/items?itemName=jebbs.plantuml>

The collage displays various PlantUML outputs:

- Code Snippet 1:**

```
@startuml
(*) --> "Initialization"

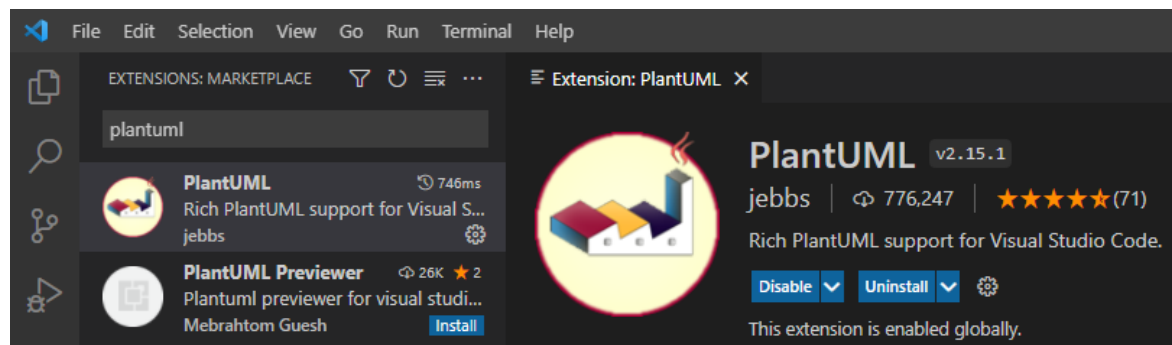
if "Some Test" then
-->[true] "Some Action"
--> "Another Action"
-right-> (*)
else
-->[false] "Something else"
-->[Ending process] (*)
endif

@enduml
```
- Sequence Diagram:** Shows a message exchange between Alice and Bob. Alice sends an "Authentication Request" to Bob, who responds with an "Authentication Response". This sequence repeats with "Another authentication Request" and "Another authentication Response".
- Activity Diagram:** Starts with an "Initialization" node, leading to a decision diamond labeled "Some Test". If true, it goes to "Some Action"; if false, it goes to "Something else". "Something else" leads to "Another Action", which then reaches a final node.
- Class Diagrams:** Shows two classes: "Dummy" with attributes "String data" and methods "void methods()", and "Flight" with attributes "flightNumber : Integer" and "departureTime : Date".
- Code Snippet 2:**

```
@startuml
class Dummy {
String data
void methods()
}

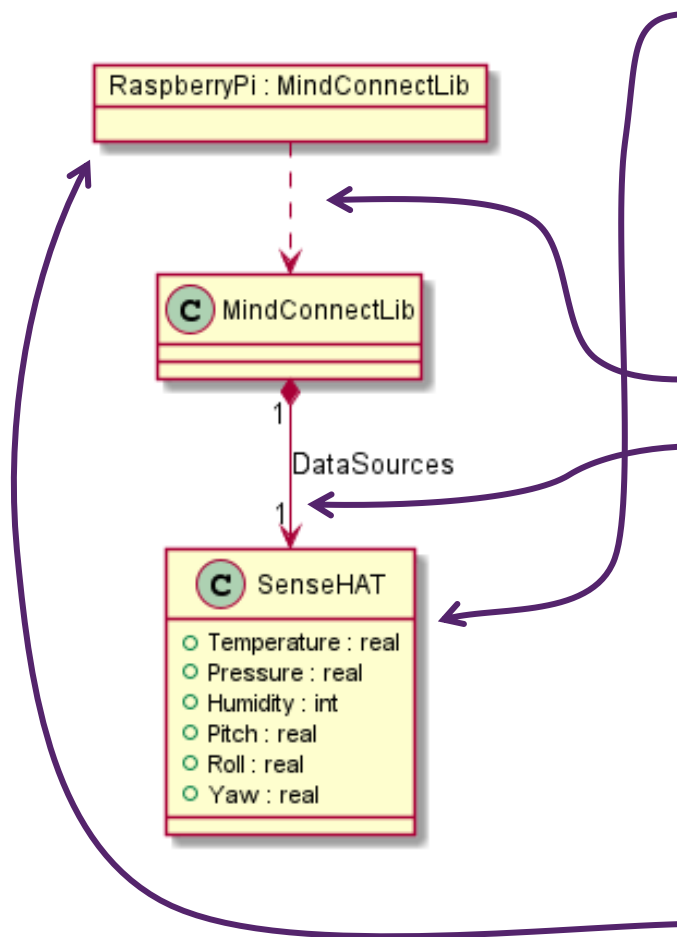
class Flight {
flightNumber : Integer
departureTime : Date
}

@enduml
```



## Class Diagrams

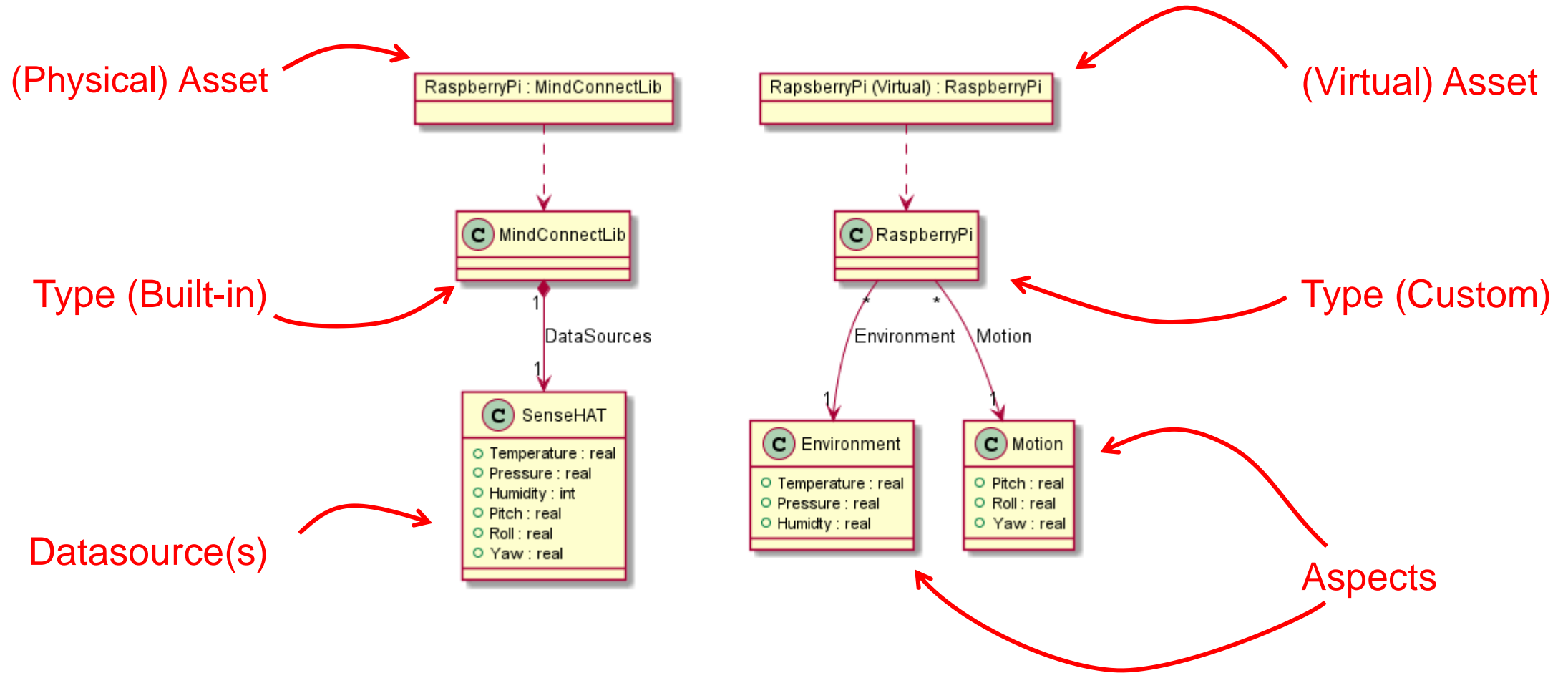
- A class is a definition of a concept (object)
  - It contains properties and operations (functions) as members
  - Every member has a visibility (+ public, - private, # protected, ~ package)
- Classes have relationships
  - Dependencies
  - Associations
- Associations can be detailed
  - Multiplicity
  - Direction
  - Aggregation/Composition
- We can also define instances of classes (realizations of objects)



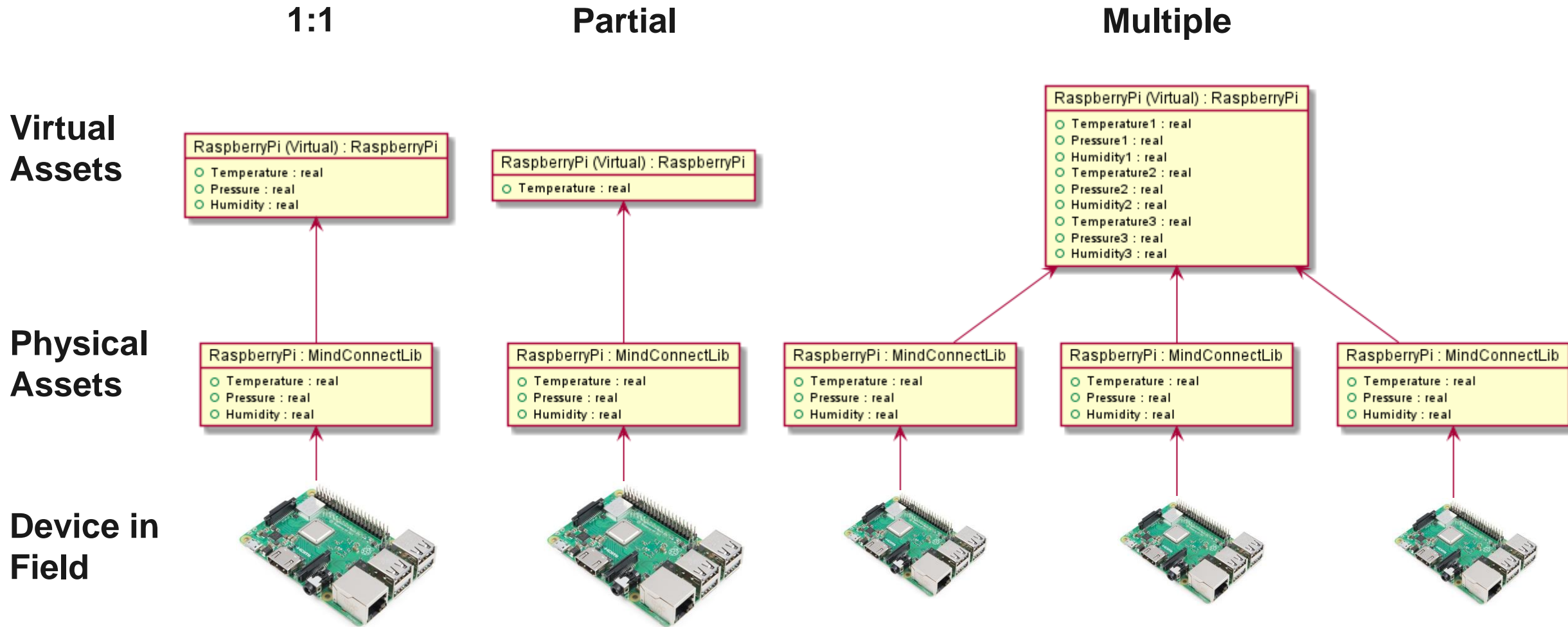
# Modeling for MindSphere

- In MindSphere, everything is an Asset
  - We distinguish between two different concepts of Assets
    - Physical Asset
      - The model of a real device
      - Can receive data
    - Virtual Asset
      - View on your data
      - Can map on Physical Assets
        - Partially
        - 1:1
        - Multiple
  - Every Asset builds upon a Type
    - For the Physical Asset, we rely on a built-in Type (**MindConnectLib**)
    - For the Virtual Asset, we define our own Type
  - Our custom Type builds upon Aspects
    - An Aspect is a group of variables
    - A variable is mapped on a data point
- See exercise for a hands-on tutorial

## Modeling your Assets



## Why a Physical and Virtual Asset?



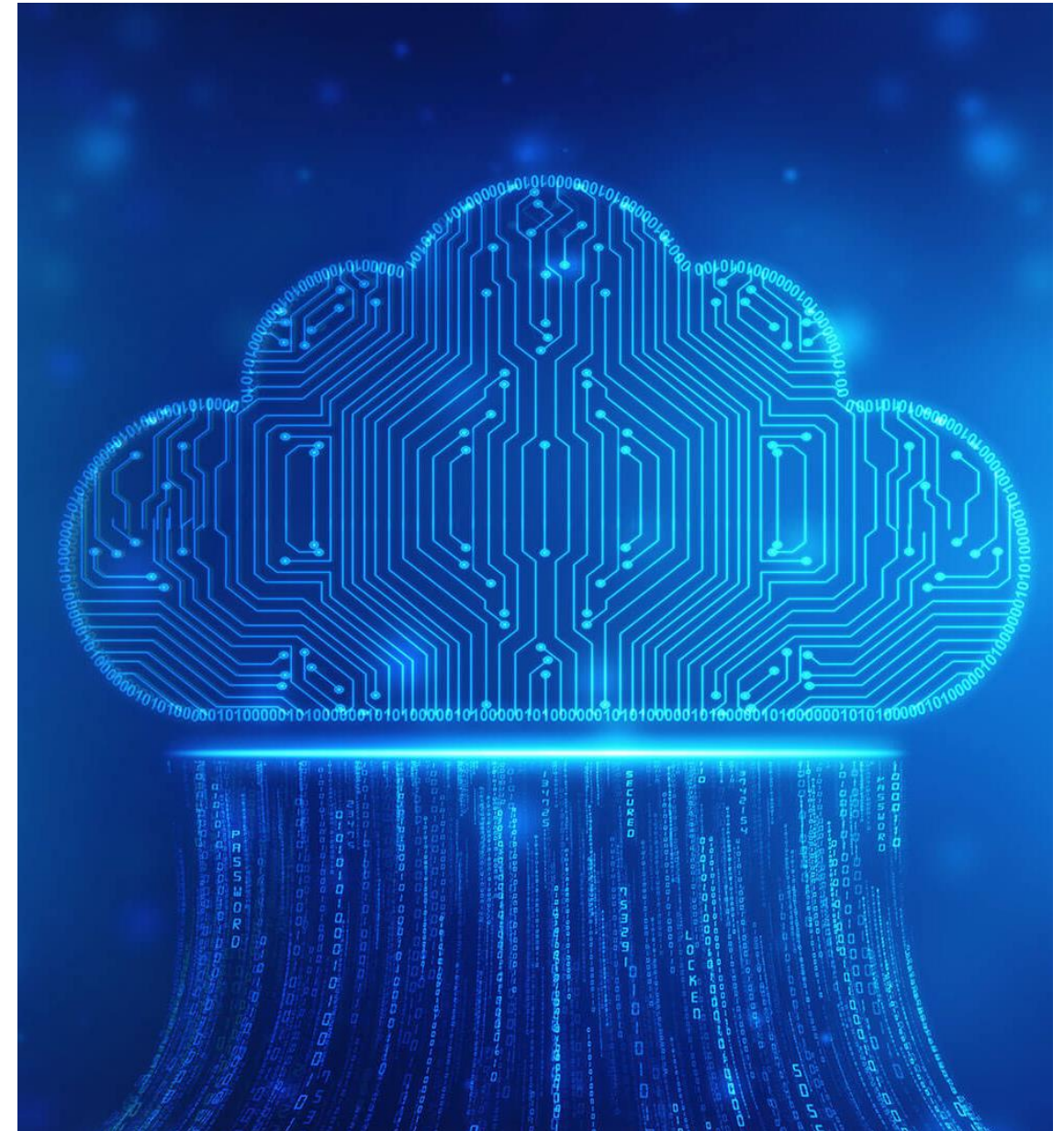
# Working with MindSphere

## Exercise



# Cloud Computing

## Lecture



# The Cloud

## Pros

### Major

- **On-demand elasticity**
  - You can get more resources when you need them (e.g., CPU, RAM, Storage, Servers/VMs)
  - You can free resources when you no longer need them
- **Security is no longer your concern** (to some extent)
  - You still need some expertise
  - But not to the same degree as with on-premises solutions

### Minor

- Pay as you go
  - Only pay for what you use
- Usage of shared infrastructure
  - Reduced costs
- Increased focus on the application layer
  - Tools and features are your main concern
- Don't worry about the hardware
  - You don't own the hardware

# The Cloud

## Cons

### Major

- **You depend on connectivity**
  - If you are disconnected from the internet, you are disconnected from your data and applications
  - Always make sure that
    - **You can operate with just (limited) offline capabilities**
    - You do not lose data
      - Delayed upload is not a problem
- **You don't know where your data is stored**
  - Some data should not be stored in the cloud
  - You don't know which route your data takes to the data center

### Minor

- You lose direct control over the underlying hardware
  - But if this is an issue, moving to the cloud is probably not a good idea
- Hard to diagnose performance issues, due to limited visibility and virtualization
  - You depend on tech support of the provider
- You depend on the features of your provider
  - Evaluate providers before you commit
  - Switching providers is possible, but will take some effort

# Security Fundamentals

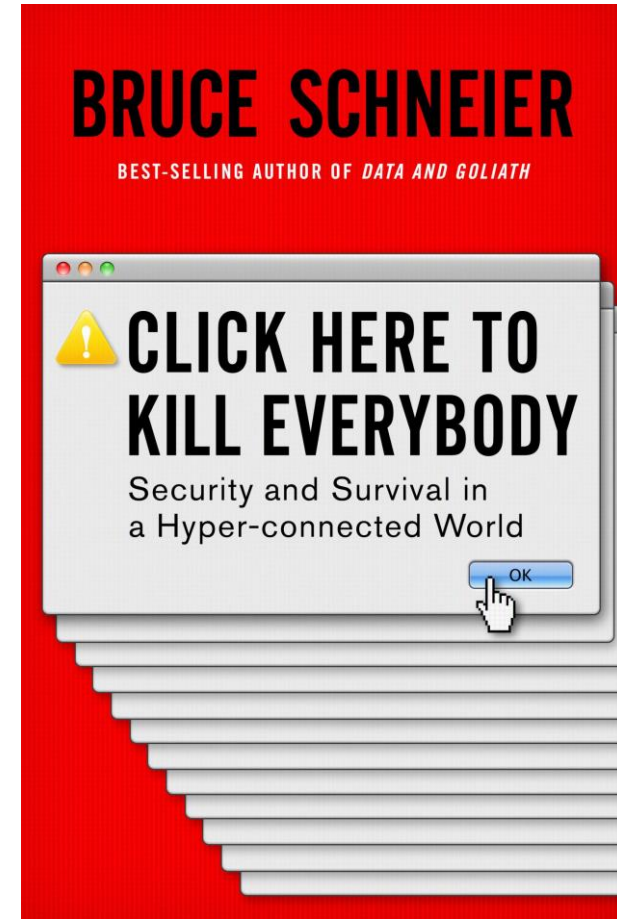
## Protecting your Stuff



# Security Fundamentals

## Why Security?

- You use it every day
  - Encrypted hard drive
  - Browsing with https://
  - Messengers (WhatsApp, iMessenger, Threema, etc)
- It is an important topic
- It will be more important
  - You can't avoid it
- "Security by Obscurity"
  - Using complexity to avoid security measures
  - Using secrecy as security method



<https://www.schneier.com/books/click-here/>

## Data Representation

- Binary

- 2 Symbols = 0 and 1
- 01001000 01100101 01101100 01101100 01101111 00100000  
01010111 01101111 01110010 01101100 01100100

UTF-8

- Hexadecimal

- 16 Symbols = 0 – 9 and A - F
- 48 65 6C 6C 6F 20 57 6F 72 6C 64

“Hello World”

- Base64

- 64 Symbols = A – Z, a – z, 0 – 9, + and /
  - “=” symbol at the end to fill the group of 4 symbols
- SGVsbG8gV29ybGQ=

– and \_ for Base64URL encoding

Have fun decoding this one...

# Cryptographic Hash Functions

- Definition
  - $\text{Hash} : \Sigma \rightarrow \Sigma^*, x \mapsto x^*$
  - $\Sigma$  denotes the set of every possible input
  - $\Sigma^*$  denotes the set of every possible output
  - $\text{Hash}(x)$  is  $O(n)$ ,  $n = |x|$
- Creates a fixed length output  $x^*$  representing the input
  - e.g., SHA-256 creates a 256-bit output ( $2^{256} = 64$  HEX symbols)
  - Input length  $|x|$  does not matter
- Collision Resistance
  - $\forall x \in \Sigma, \forall y \in \Sigma, x \neq y : \text{Hash}(x) \neq \text{Hash}(y)$ 
    - Two different inputs cannot have the same output
    - Theoretical possible, but not in practice
- Hiding
  - $\text{Hash}(x) = x^* : \nexists \text{Hash}^{-1}(x^*) = x$ 
    - It's not possible to get the input from the output
    - Except by trying every possible input

# Security Fundamentals

## Hash Examples

"Smart Factory" = dcb6fe8fd753123cef283a601dbe6ffe3fd1d69b819d21c346dfc4e1ed9bcb01

"SmartFactory" = 84d0681e3e1757a3194156f806be354c0d2496b235a0accb2d1c75cf605b7a6b

*Changing one character has an effect on the whole result*

- Used for storing your passwords
  - This way, your password is not stored in plain text
    - And therefore, cannot be retrieved
  - A salt (random data) is added to prevent pre-generating hashes (rainbow tables)
    - [https://en.wikipedia.org/wiki/Rainbow\\_table](https://en.wikipedia.org/wiki/Rainbow_table)
- Checking your password is a comparison of two hashes
  - If equal, the given password must be the same as the one that is known by the system

# Symmetric Cryptography

- Using the same *key* for encryption and decryption
  - $\text{msg} \oplus \text{key} = \text{msg}_{\text{enc}}$
  - $\text{msg}_{\text{enc}} \oplus \text{key} = \text{msg}$
- Algorithms
  - AES (Advanced Encryption Standard)
  - DES (Data Encryption Standard)
  - IDEA (International Data Encryption Algorithm)
- Advantages
  - It's fast
  - It's simple
- Disadvantages
  - If someone gets the key, he can decrypt the data
  - Does not scale in a network where every connection needs a unique key
- Examples
  - Encrypt and decrypt local data

[https://en.wikipedia.org/wiki/Symmetric-key\\_algorithm](https://en.wikipedia.org/wiki/Symmetric-key_algorithm)

# Asymmetric Cryptography

- We have two keys
- A private key  $key_{priv}$ 
  - Only known by you (thus private)
- A public key  $key_{pub}$ 
  - Known by everybody (publicly available)
- Algorithms
  - RSA
- Advantages
  - It scales well in networks
- Disadvantages
  - Slow (compared to symmetric encryption)
  - Can potentially be solved since one key is known
    - Relays on features of large prime numbers
    - It's just hard to crack, but not impossible

[https://en.wikipedia.org/wiki/Public-key\\_cryptography](https://en.wikipedia.org/wiki/Public-key_cryptography)

## Two Ways of using Asymmetric Cryptography

### Encryption

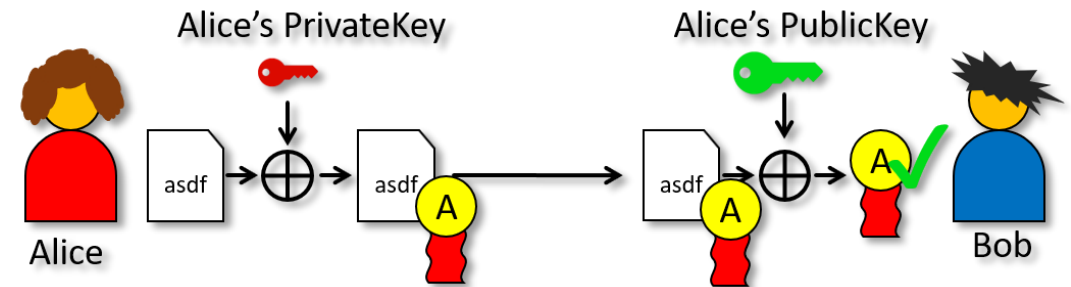
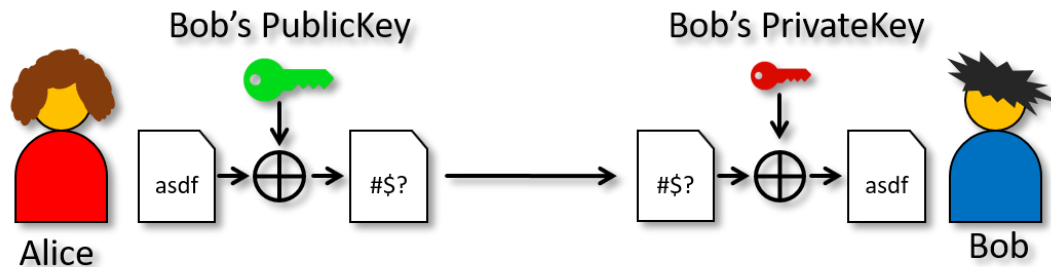
- $msg \oplus key_{pub} = msg_{enc}$
- $msg_{enc} \oplus key_{priv} = msg$
- Only the one with the corresponding private key  $key_{priv}$  (e.g., the recipient) can decrypt  $msg_{enc}$



### Signing

- $msg \oplus key_{priv} = sig_{msg}$
- $sig_{msg} \oplus key_{pub} = msg$
- Everyone can verify the signature  $sig_{msg}$  was created by the owner of the private key  $key_{priv}$

Digital Signatures







# Security Fundamentals

## JWT (JSON Web Token)

- Common way to grant access to an API
- Uses JSON to structure claims
  - Claims are your granted rights
- Token is signed to ensure its validity
  - A token could be changed
  - But you can detect the changes when you regenerate the signature

<b>Header</b>	<pre>{   "alg" : "RS256",   "typ" : "JWT" }</pre>	Metadata of the token e.g., Algorithm used
<b>Payload</b>	<pre>{   "ten" : "hsr",   "iat" : 1570549729,   "exp" : 1571154529 }</pre>	Data of the token e.g., who you are, what you are allowed to do, when the token was generated, how long the token is valid, etc.
<b>Signature</b>	<pre>Hash(   secret,   base64urlEncoding(header) + '.' +   base64urlEncoding(payload) )</pre>	Signature of the token To ensure the token is valid and no information was changed

Token = base64urlEncoding(header) + '.'  
+ base64urlEncoding(payload) + '.'  
+ base64urlEncoding(signature)

[https://en.wikipedia.org/wiki/JSON\\_Web\\_Token](https://en.wikipedia.org/wiki/JSON_Web_Token)



## What's behind the MindSphere Token?

- Signature

```
RE76sDFKsSDQiZD2lzHxYL7pXwYD3NfeKsAAJ_0-  
ua_90JuYAUXfLIIW0w4I09GZ9upGY9DiE6aW5FjUiRVHJre36IKsTwyVq9o7  
UO2HBNyTLMP8aHuW14unm96sRmEx_sAGUNT2GDiegWDN1WNDVqqt-  
CNpxef8O9v8pc7by7RH3DzLwphqpTwHXqEnFruqTOwbdD38SEr5o90rCd9  
HCprBfVroFiLzXX6RJ_L1r6nejtMLC5vQnfH0qNYK9qFRFq7OSi9WkZb-  
AWZBvD_FfhVvCe0WJ7EBw36kzgNeWxEpyZrDGi_CyfMKyxUtMEu-  
IPN8eWaW_OaoBG0_sPsCUQ
```

- We can't do much with that since it's a hash
- This is used within MindSphere to check if the header and payload are correct
  - If true, sent data is accepted
  - If not, the data is simply rejected
    - You are not interested in knowing what has changed, just that something has changed

# Dashboard with Edge2Web

## Exercise



# Project

## Self-Study



# Introduction to the Project

Show me the Data



# Project

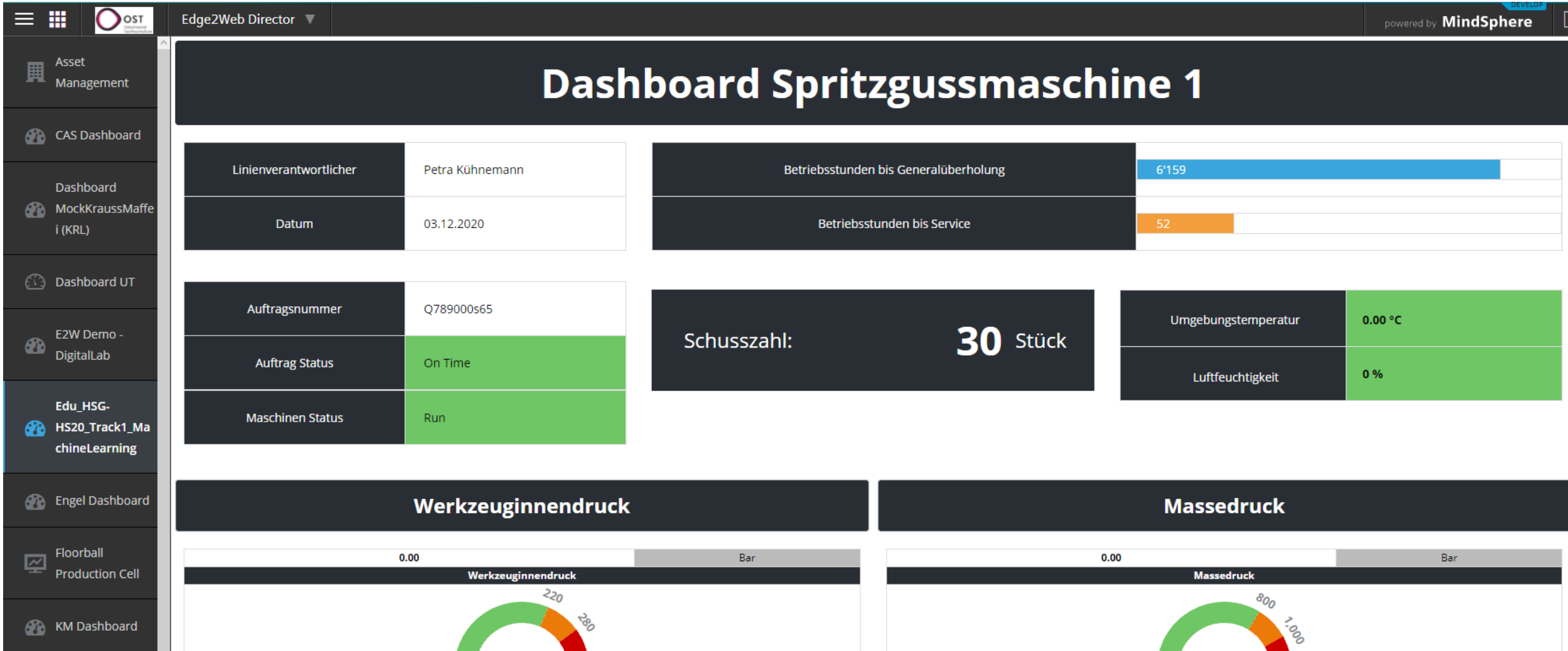
## Task 2: Development of a Dashboard

### Task Formulation

- Same groups as you had in task 1
  - 1. Visualize data on your device
  - 2. Model your assets in MindSphere
  - 3. Establish communication to your Asset and check if data is received
  - 4. Develop a minimal dashboard to visualize the data sent from your device
  - 5. Extend the dashboard with further data visualizations from a second device
- ➔ Supporting Documents
- Node-RED Dashboard Manual
  - Tutorial (Exercises)
  - Persona

### Expected Deliverables

- Dashboard with 6 tiles showing at least 12 data points
  - Documentation of your project
    - Thoughts and implications of the persona for the UI (1 page)
    - Paper prototype/design concept (4 pages)
    - Finding/conclusion (0.5+ page)
    - Reflection on the task (0.5+ page)
- ➔ *The submission of the results has to be done on Moodle until 14.12.2021 (PDF-Format)*
- Presentation (approximately 5 minutes) of the dashboard on 14.12.2021
- ➔ *The submission of the presentation has to be done on Moodle until 14.12.2021 (PDF-Format)*



# Project

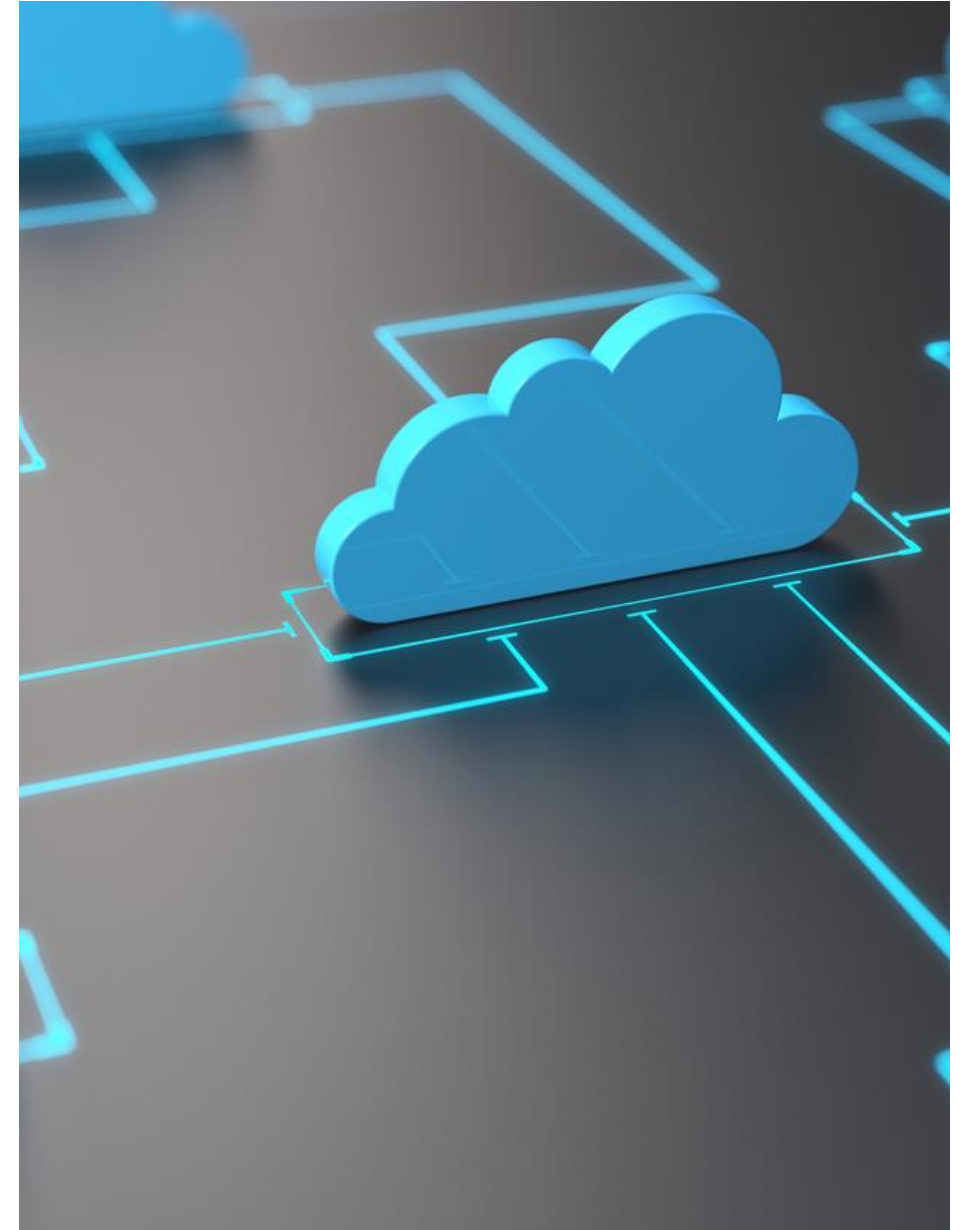
## Exercise



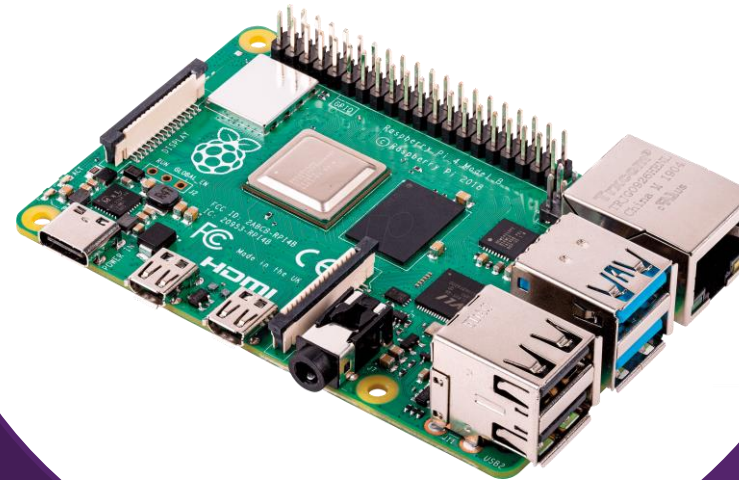
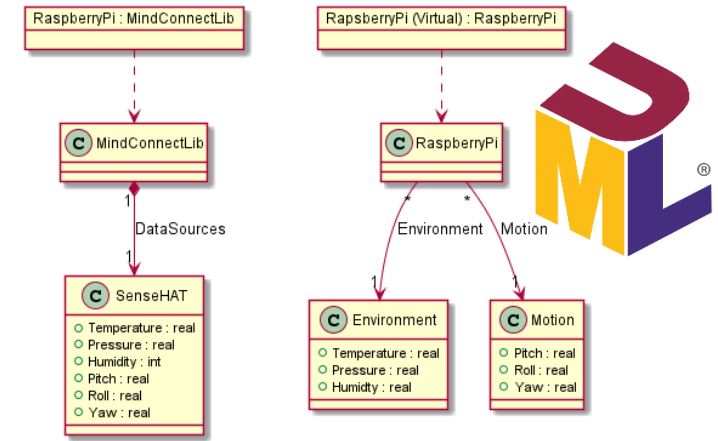
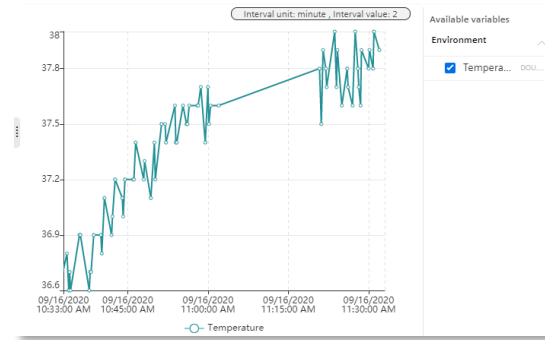
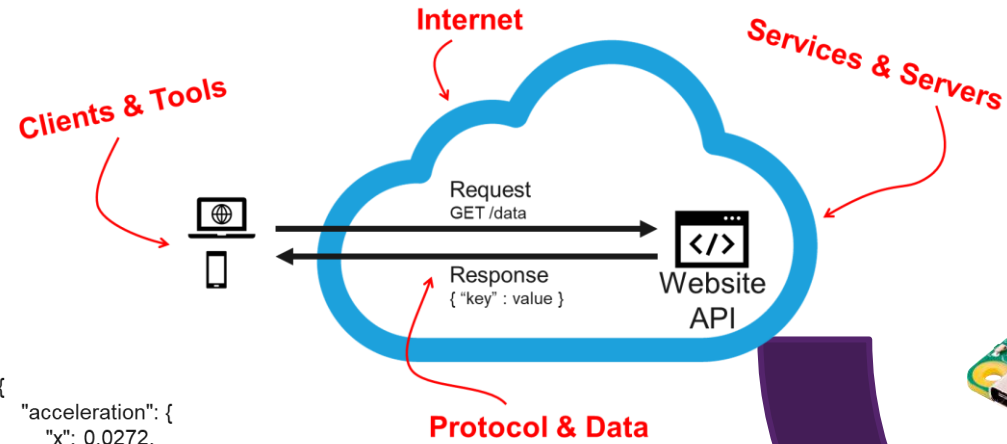
# Digital Manufacturing

## Today's Objectives

- ✓ You can use a REST API
- ✓ You can identify the service concepts of cloud providers
- ✓ You know the basic pros and cons about the cloud
- ✓ You have a basic understanding of key security concepts
- ✓ You can identify the parts of JWT



## Closing Remarks



```
{
  "acceleration": {
    "x": 0.0272,
    "y": 0.0238,
    "z": 1.0052
  },
  "gyroscope": {
    "x": 0.002,
    "y": 0,
    "z": -0.0015
  },
  "orientation": {
    "roll": 1.4011,
    "pitch": 358.4654,
    "yaw": 249.031
  },
  "compass": 249
}
```

