

# A competitive food retail architecture with microservices

A five year journey with microservices

Sebastian Gauder  
Software Architect eCommerce  
 @rattakresch

11.02.2019 ObjektForum Köln

**REWE** digital

A vibrant arrangement of fresh vegetables on a weathered blue wooden background. The vegetables include orange and yellow carrots, purple and orange sweet potatoes, small yellow potatoes, and several beets with their green leafy tops. A central blue rectangular overlay contains the text.

# Our history

# Details REWE GROUP

Turnover

>57 bn

Employees

>345.000

Shops

>15.000

Industries

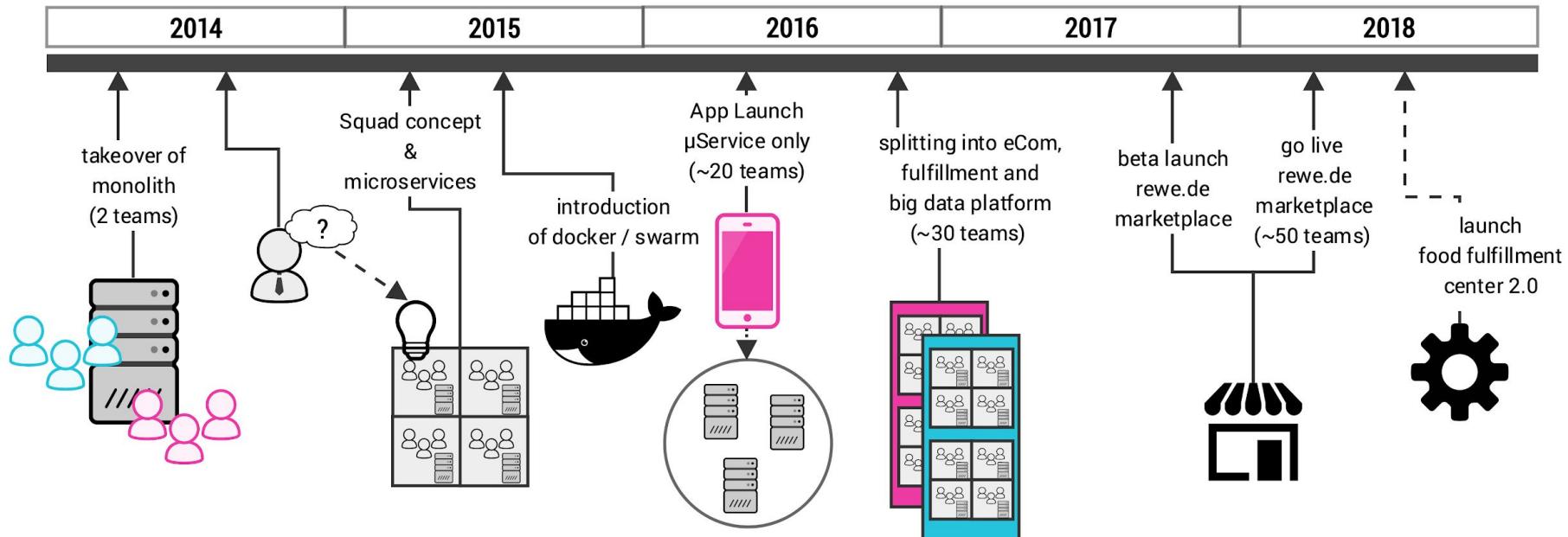
Food Retail,  
Tourism,  
DIY



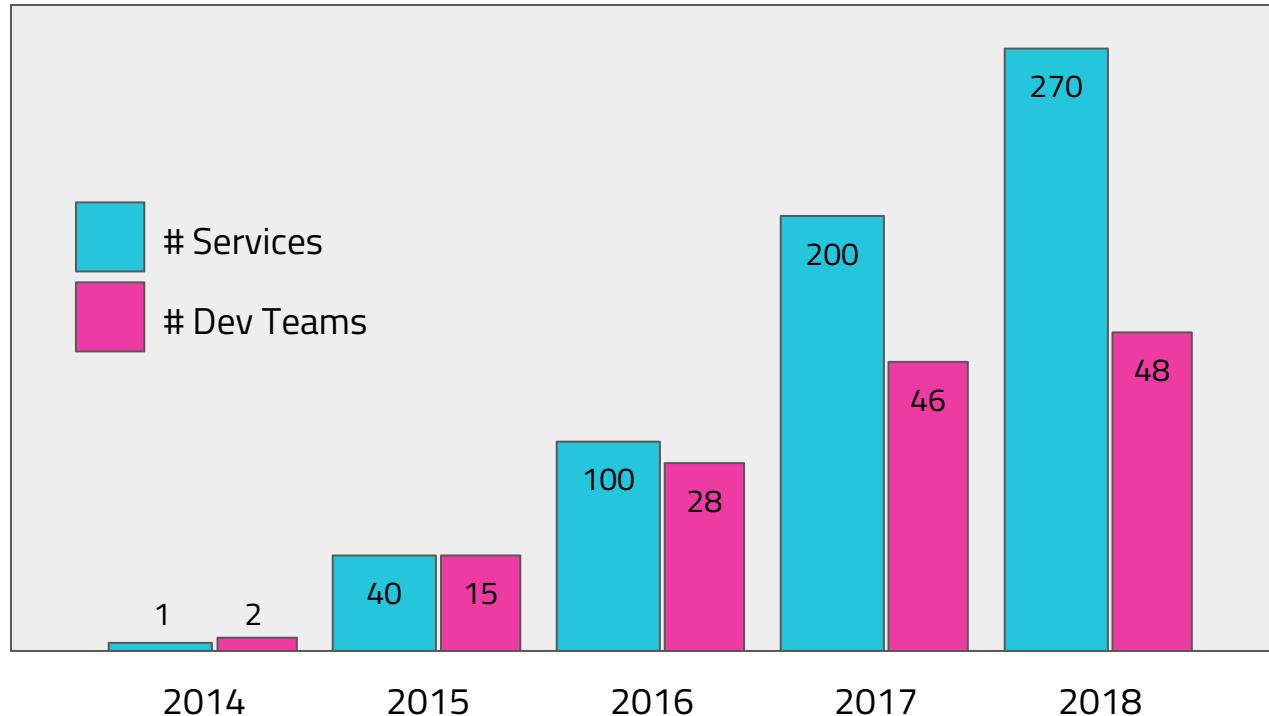
History

>90 years

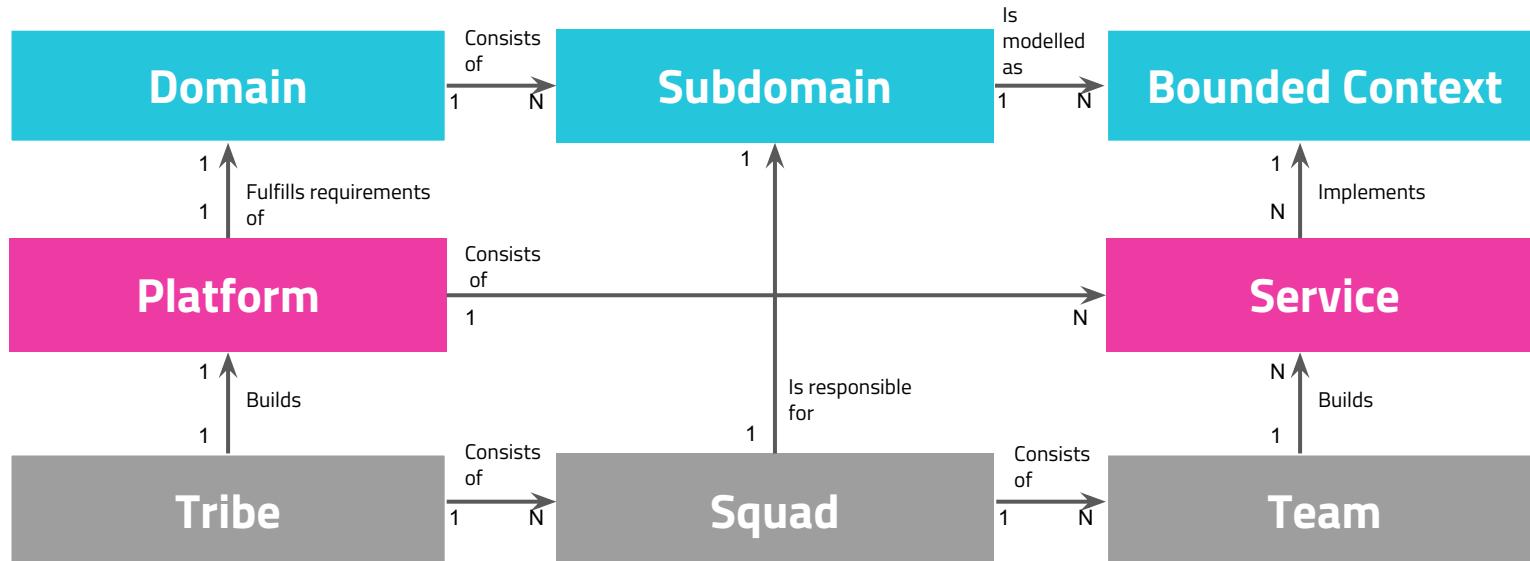
# Our history (Spoiler alert)



# Our history



# Organization and Architecture at Rewe Digital



A large, dark, textured rock or monolith is positioned on the left side of the frame, partially overlapping the text area. It has a rough, granular surface with various shades of brown, tan, and black.

# How to scale monoliths?

# Design Goals

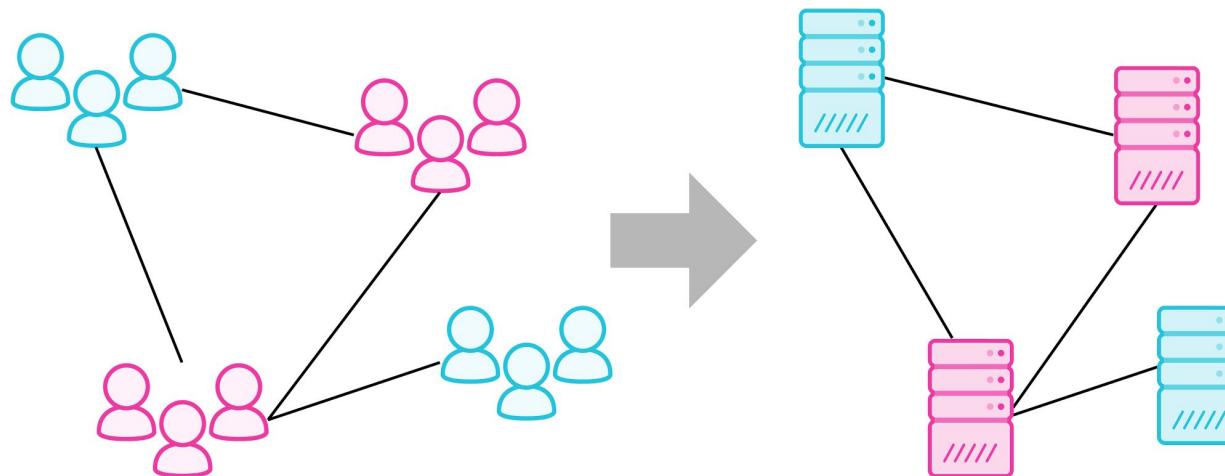
Decentralization

Vertical Boundaries

# Conway's law

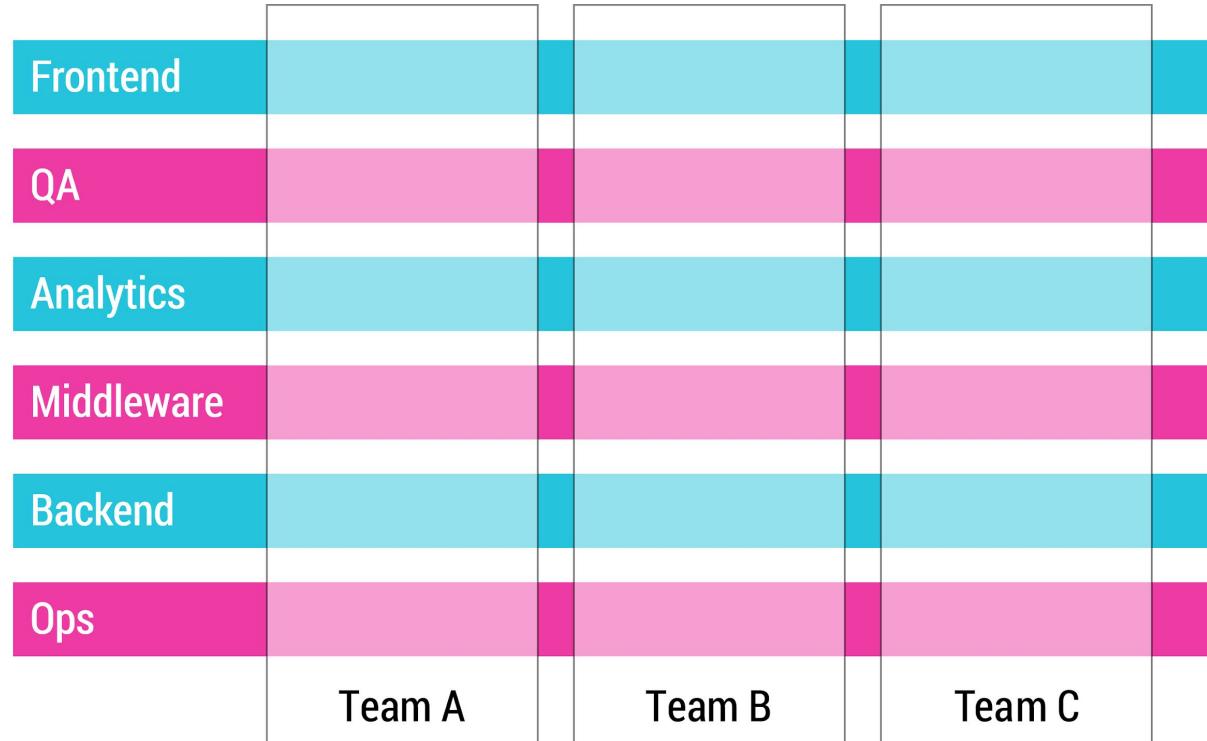
*"Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."*

Melvin Conway (1967)



# How to organize teams to build a scalable system?

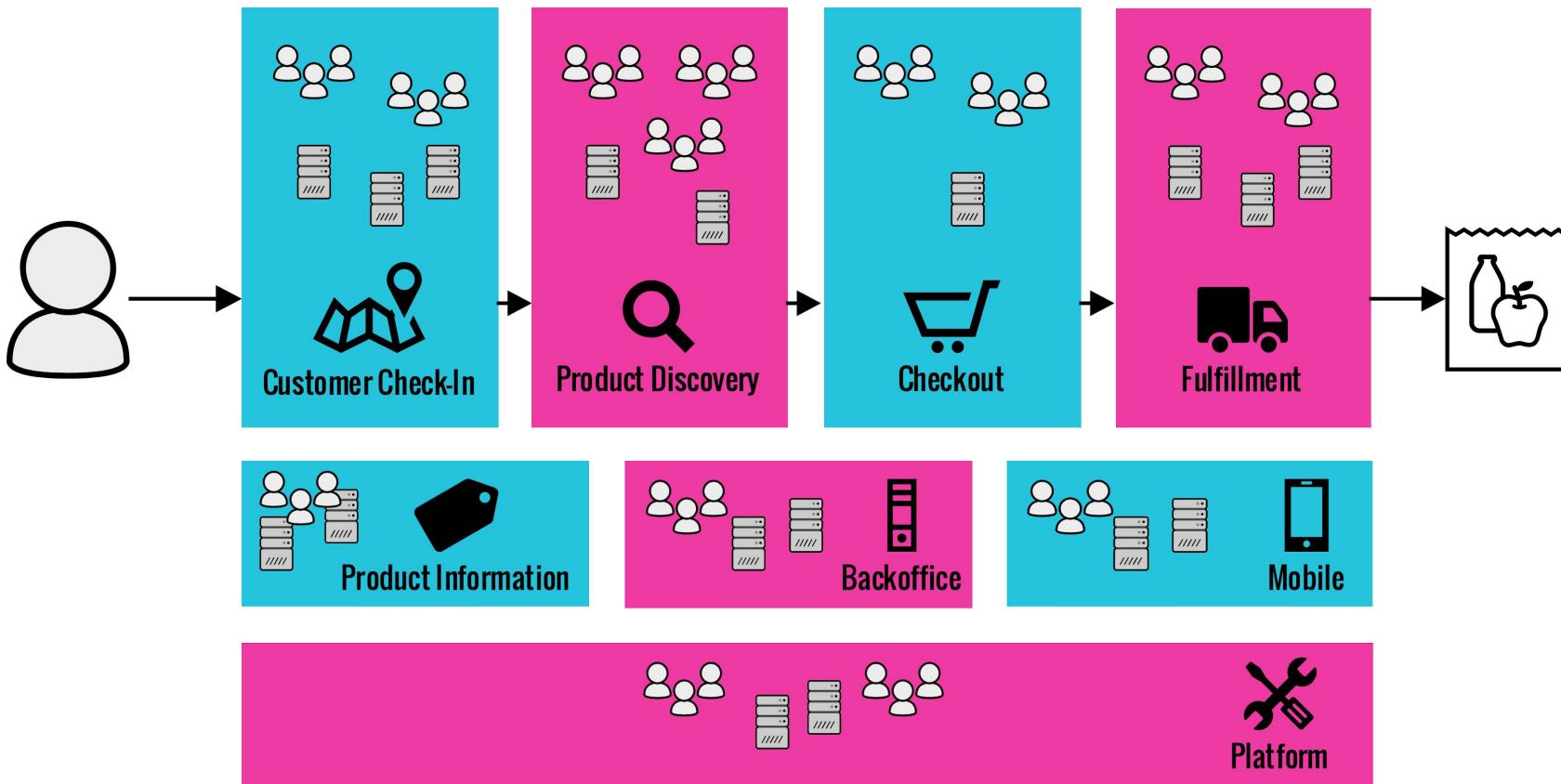
Functional and vertical teams



An aerial photograph of a salt pan complex, showing numerous rectangular pools of water and thick, white salt deposits along the edges. A small figure of a person is visible in one of the pools. A large, semi-transparent blue rectangular box covers the middle portion of the image. Inside this box, the text "How to determine boundaries?" is written in a bold, white, sans-serif font.

How to **determine boundaries?**

# Customer journey defines subdomains





# How to guide developers?

# Guarding rails for developers

## Design Goals

- Vertical boundaries
- Decentralization

## Architectural principles

- Collection of 9 basic 'laws'
- Autonomy, Automation and Communication

## Guides

- Practical manual for common tasks (RFC 2119)
- e.g. Eventing, REST, Authentication

**MUST**  
**SHOULD**  
**COULD**



autonomy

# Autonomy principles



## Deploy independently

Ensure that the services can and are deployed by the teams themselves.



## Isolate failure

Make the services as resilient as possible.



## Hide implementation Details

Different verticals must not share state. Verticals hide their implementation details.



## Encapsulate Data Storage

For any data resource exactly one service is responsible. If possible, the data supply should proceed asynchronously in the background

# Challenges

## Autonomous Teams

- challenge for product owners as they have to rethink their features
  - they might have to reduce the scope
  - split the original feature into smaller ones for each team / bounded context

## Data-integration pattern

- Change your service behavior depending on the data you have to display
- Enable others to deliver functionality later

## Evolving Implementations

- Start with a fitting but simple solution and get better by every team implementing it

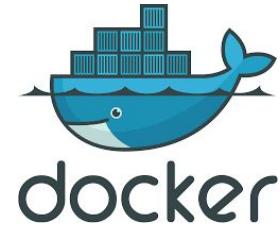
# Autonomy opens your tech stacks

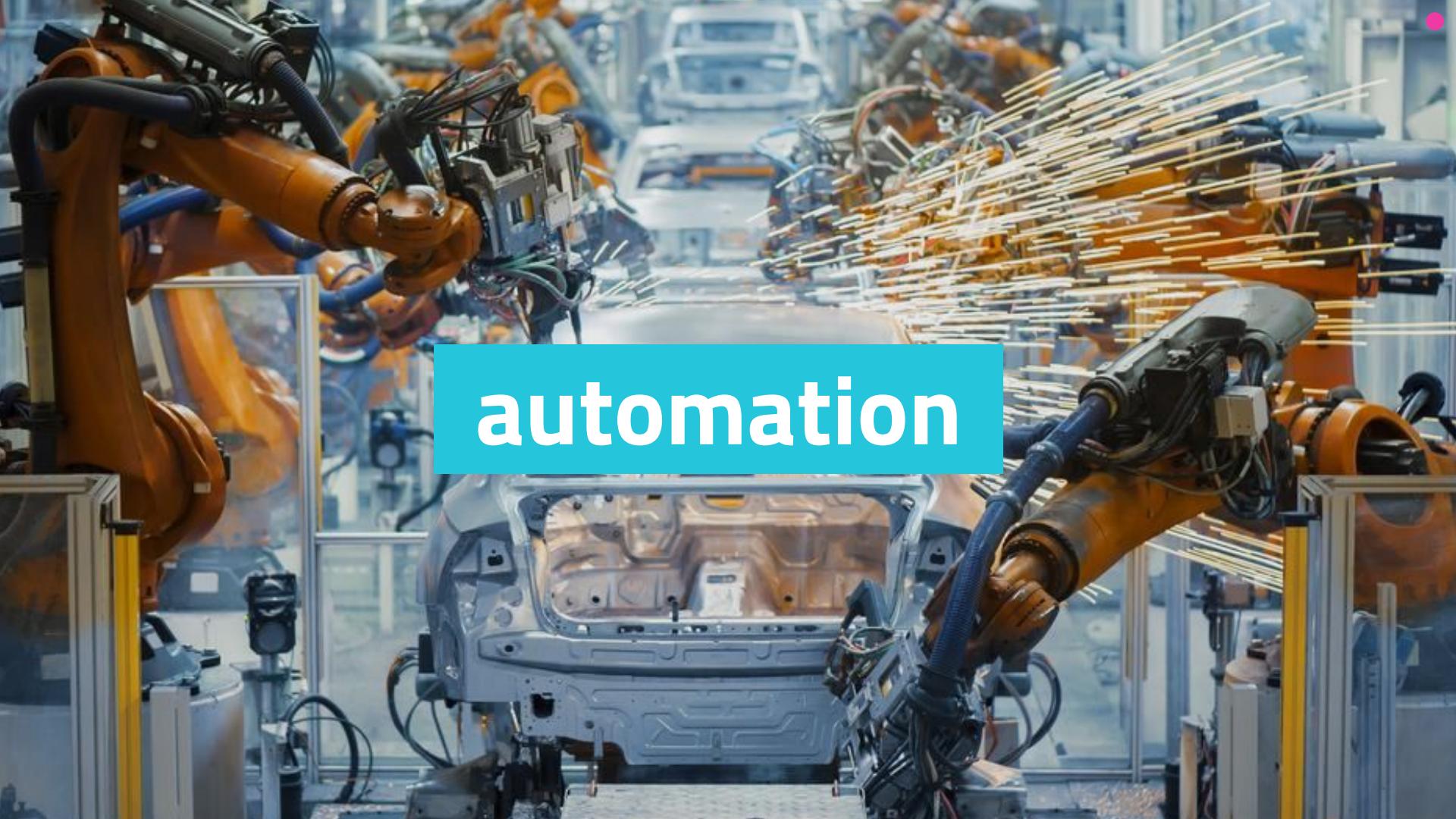


Java™



kafka





automation

# Automation principles

## ↔ Be Scalable

 Services are scaled horizontally

## ⚙ Embrace a Culture of Automation

Test, deployment and operations are completely automated

## 🔍 Be Highly Observable

 Use of semantic monitoring to see if the whole platform works correctly.

# communication



# Communication principles



## Follow REST Principles

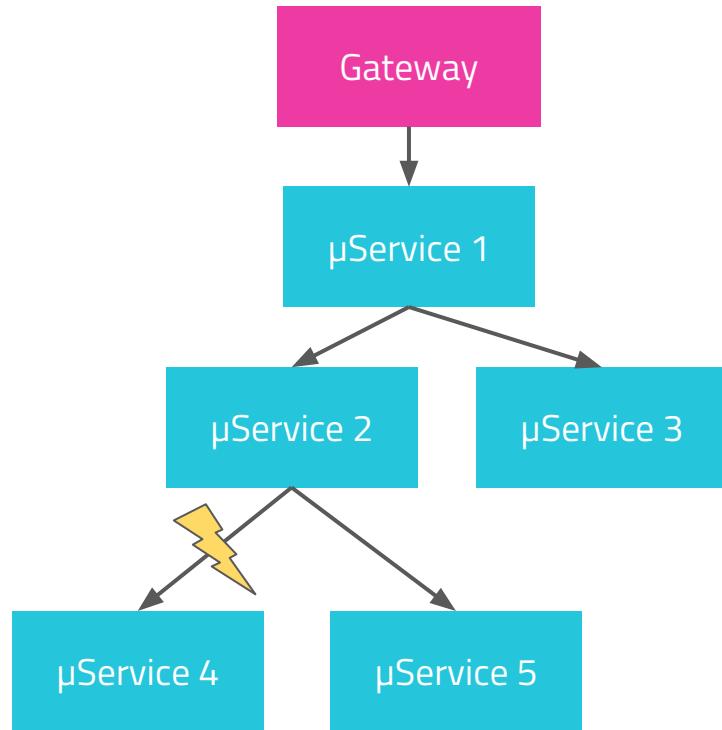
The API of a service follows the RESTful paradigm (REST maturity level 2)



## Standardize Service communication

Inter-service communication is standardized and if possible asynchronous

# Problems in HTTP/REST-only architectures



Things that help:

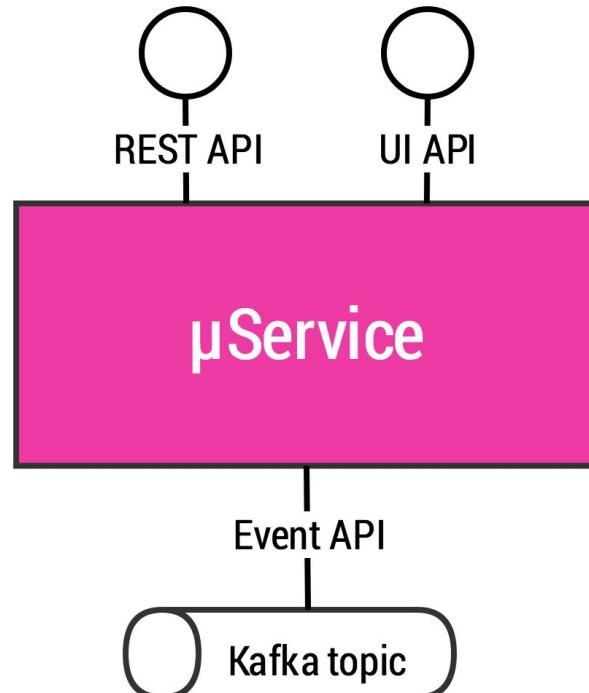
- Timeouts
- Fallbacks
- Circuit Breakers
- **Eventing**

# Communication in microservice worlds

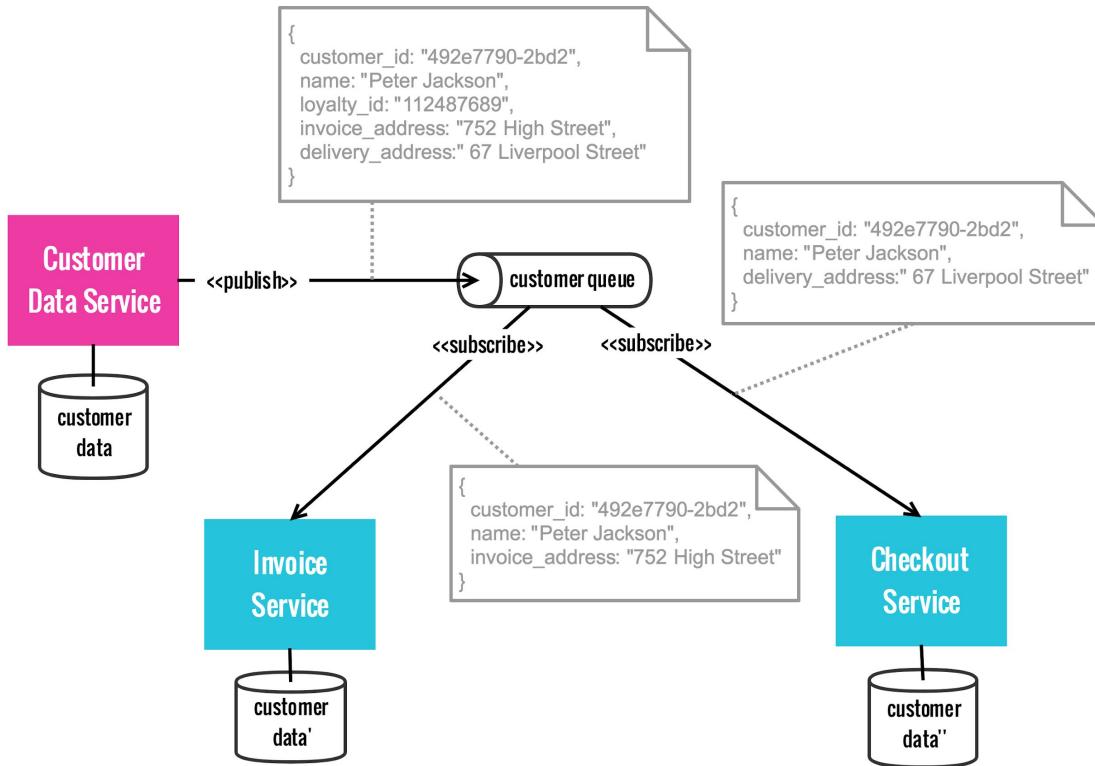
Having data is better than fetching data

## Asynchronous > Synchronous

- Have as much needed data locally as possible -> resilience at request time
- Duplication of data is accepted and wanted
- Problem of eventual consistency



# Eventing with Apache Kafka



- **Eventing != Messaging**
  - Publish events that already **happened**
  - One owning service per queue/topic
  - Autonomy at request time
- **Complete entities - not deltas**
  - Transactional completeness
  - Allows log compaction
- **Re-writing and Re-reading**
  - needed in case of additional data
  - useful in case of data loss

# Lessons learned with microservice APIs

## Eventing is an API as well

- Events have to behave like APIs and avoid breaking changes.

## Writing APIs is hard

- Teams tend to write APIs for special clients (e.g. mobile apps)
- New clients often don't match and require API changes

## Breaking changes require a strict procedure

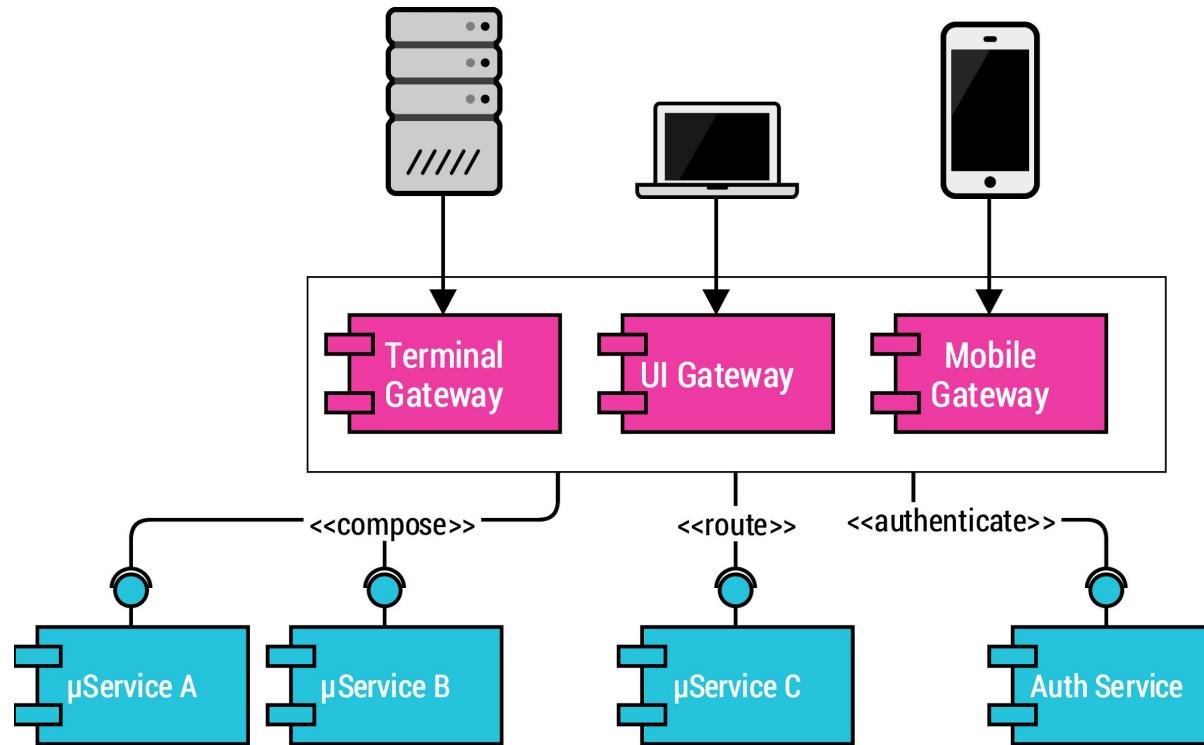
- Could be solved by new endpoints, new topics



How to access?

# Usage of API gateways as entry points

How to access Microservices?



# Dynamic frontend composition

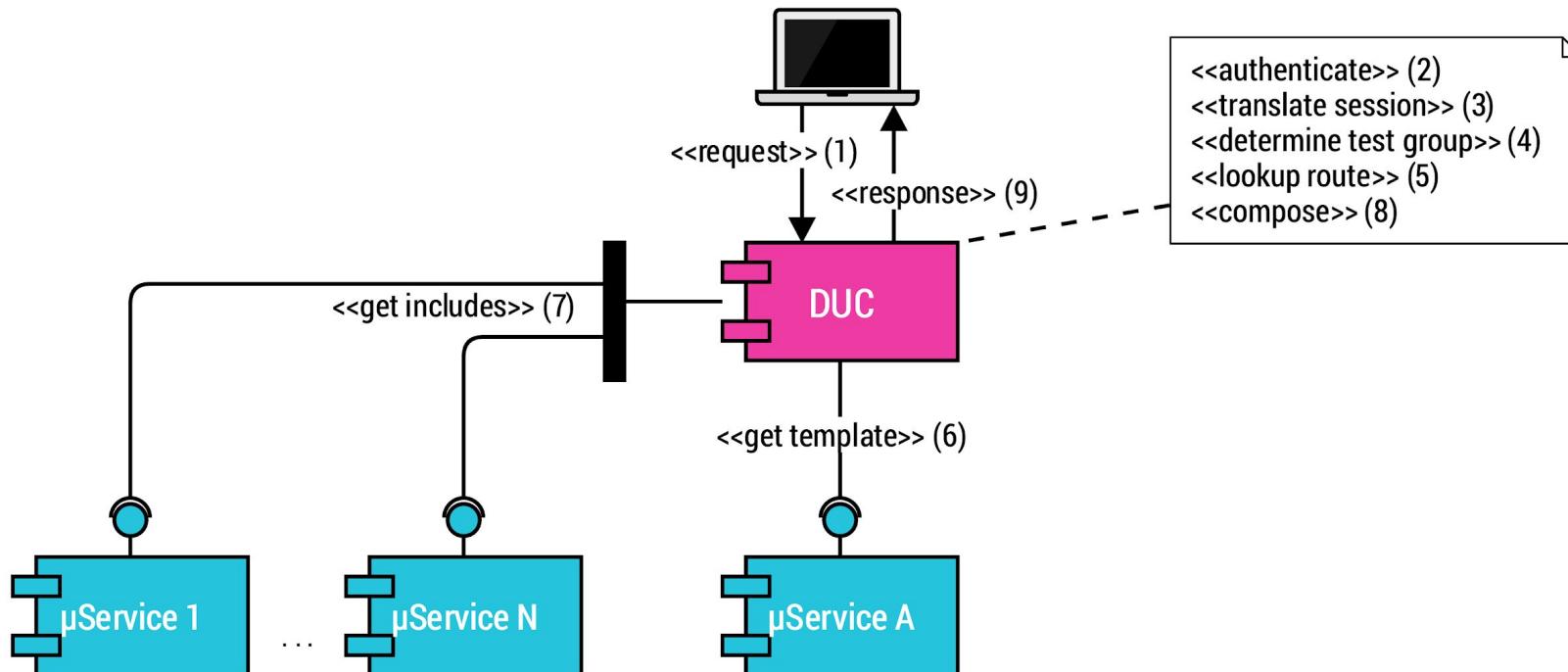
How UIs in microservice architectures should be composed

The screenshot shows the REWE website's header and a product detail page, divided into several functional teams:

- Header Team**: Contains the REWE logo, navigation links (Dein REWE Markt, Online bestellen), and user account information (Anmelden, PLZ 50674).
- Search Team**: A search bar with placeholder "Suche" and a magnifying glass icon.
- Basket Team**: Includes a shopping cart icon, a "Liefertermin wählen" button, a "Favoriten" section, and a "Warenkorb" summary showing 0 Artikel and 0,00 €.
- Navigation Team**: A horizontal menu with "Alle Produkte", "Meine Produkte" (marked as NEW), "Angebote", and "Themenwelten". Below it is a breadcrumb trail: Obst & Gemüse > Obst > Bananen.
- Product Detail Team**: Shows a product image of "REWE Beste Wahl Bananen" (Art.-Nr. 1028378) at 0,34 € per piece (1 Stück ca. 200 g (1 kg = 1,69 €)). It includes a quantity selector (1) and a "In den Warenkorb" button. A note states: "Kl.: M; Ursprungsland und Klasse können je nach Angebot abweichen. Bananen werden grün geerntet, denn sie reifen nach. Die Büschel werden gewaschen, in "Hände" zerteilt und verpackt. Sie werden auf besonderen Bananenschiffen in die Verbrauch..." and a link "Mehr Infos anzeigen".

# Dynamic UI composition with DUC

Where does my UI come from?



# Git Project: rewe-digital/integration-patterns



<https://git.io/vA2MY>



**Thank you**  
Questions ?

**REWE** digital

# A competitive food retail architecture with microservices

A five year journey with microservices

Sebastian Gauder  
Software Architect eCommerce  
 @rattakresch

11.02.2019 ObjektForum Köln

**REWE** digital