From Monolith to Microservices

Paul Puschmann

OSAD 2018, Munich

REWE digital

Our history

Details REWE GROUP

Turnover

>54 bn

Employees

>330.000

Shops

>15.000

Industries

Food Retail,

Tourism, DIY

REWE

DER Touristik

BILLA

REWE digital

PENNY.

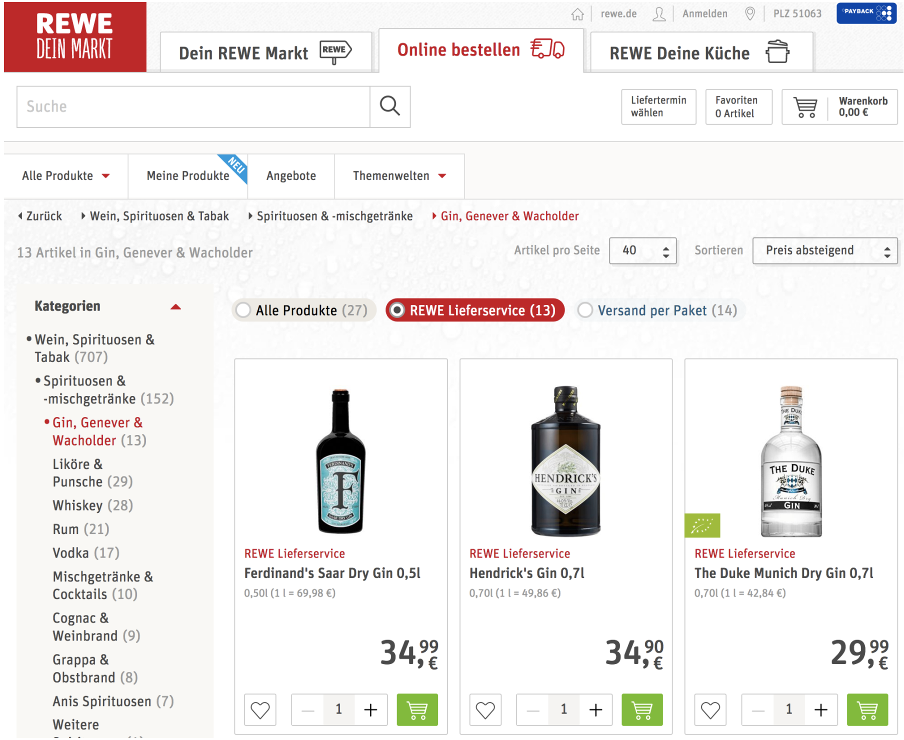
BIPA

toom

History

> 90 years

What do we actually run?



REWE DEIN MARKT

rewe.de | Anmelden | PLZ 51063 | PAYBACK

Dein REWE Markt | Online bestellen | REWE Deine Küche

Suche

Liefertermin wählen | Favoriten 0 Artikel | Warenkorb 0,00 €

Alle Produkte | Meine Produkte NEU | Angebote | Themenwelten

Zurück | Wein, Spirituosen & Tabak | Spirituosen & -mischgetränke | Gin, Genever & Wacholder

13 Artikel in Gin, Genever & Wacholder

Artikel pro Seite 40 | Sortieren Preis absteigend

Kategorien

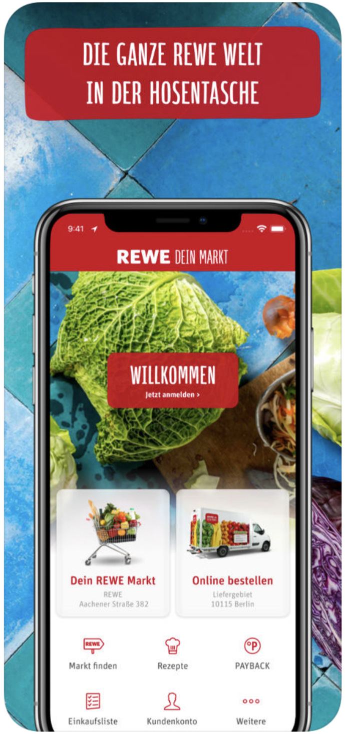
- Wein, Spirituosen & Tabak (707)
  - Spirituosen & -mischgetränke (152)
    - Gin, Genever & Wacholder (13)
    - Liköre & Punsche (29)
    - Whiskey (28)
    - Rum (21)
    - Vodka (17)
    - Mischgetränke & Cocktails (10)
    - Cognac & Weinbrand (9)
    - Grappa & Obstbrand (8)
    - Anis Spirituosen (7)
    - Weitere

○ Alle Produkte (27) ● REWE Lieferservice (13) ○ Versand per Paket (14)

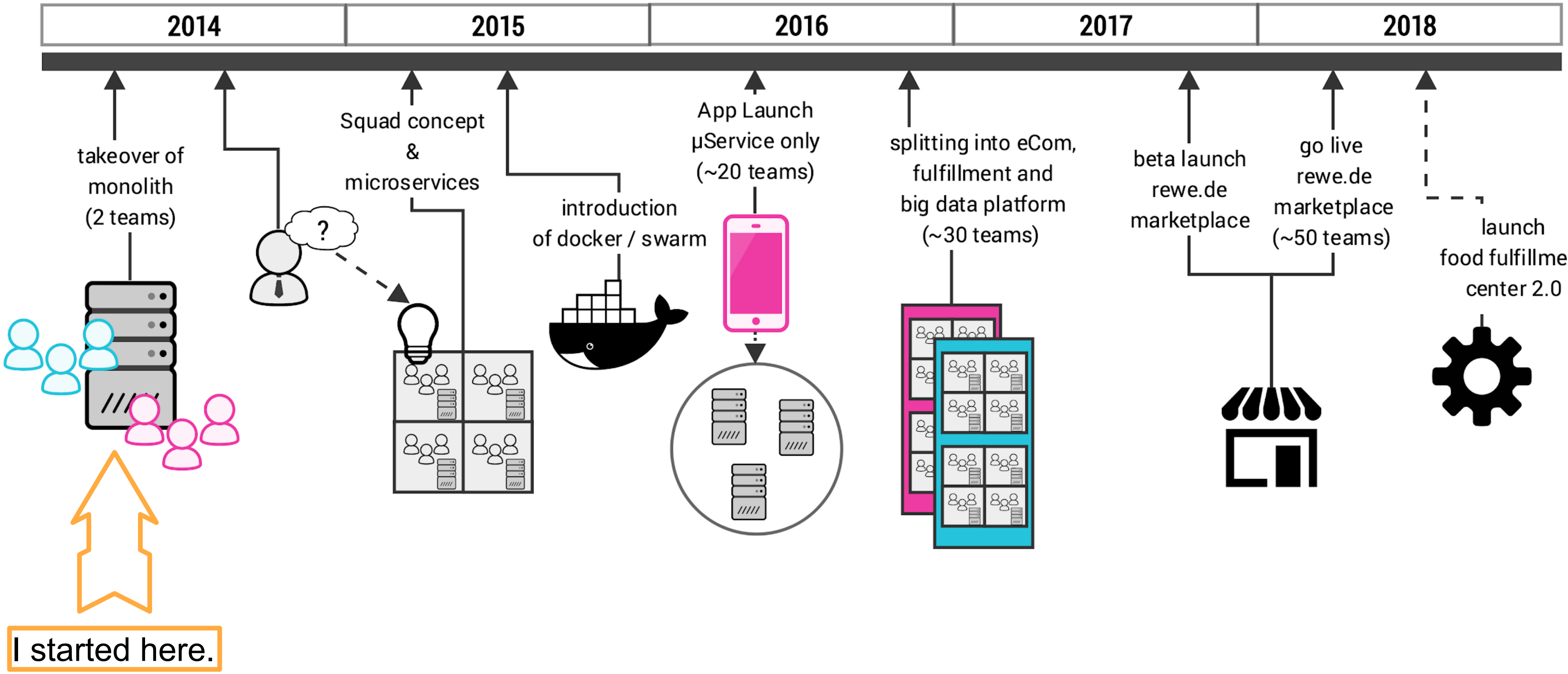
REWE Lieferservice
Ferdinand's Saar Dry Gin 0,5l
0,50l (1 l = 69,98 €)
34,99 €
— 1 +

REWE Lieferservice
Hendrick's Gin 0,7l
0,70l (1 l = 49,86 €)
34,90 €
— 1 +

REWE Lieferservice
The Duke Munich Dry Gin 0,7l
0,70l (1 l = 42,84 €)
29,99 €
— 1 +



Our history at REWE Digital



2014	2015	2016	2017	2018

takeover of monolith (2 teams)

Squad concept & microservices

introduction of docker / swarm

App Launch µService only (~20 teams)

splitting into eCom, fulfillment and big data platform (~30 teams)

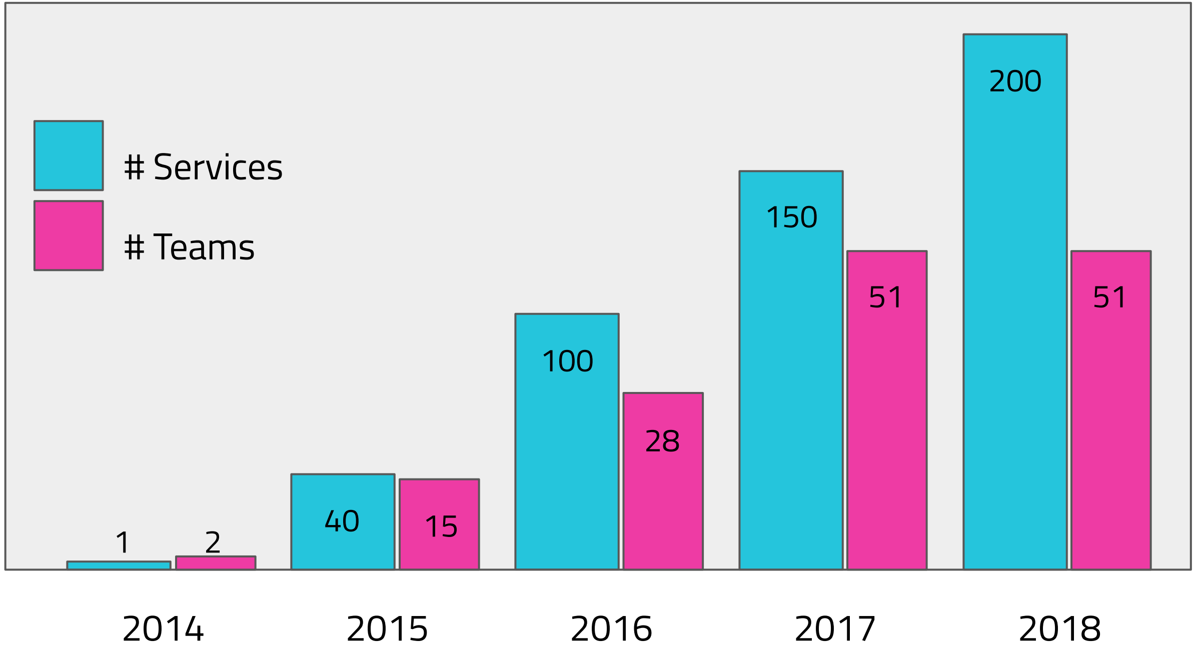
beta launch rewe.de marketplace

go live rewe.de marketplace (~50 teams)

launch food fulfillment center 2.0

Our history at REWE Digital

This is an approximation...



Main Goals

- Have a good platform / software architecture
- Scale the application
- Enable fast delivery of features, accelerate the business

What did the Ops-people do?

- Take care of our “Managed-Hosting”
- *Re-automate* an already existing PROD-environment with Ansible
- Keep everything running
- Support our developers
- Do Pager-Duty

Status Quo of the Monolith

2014 / beginning of year 2015

- Integration of new features — *difficult*
- Deployments every two weeks — *slow*
- Deployments took eventually 1h — *slow*
- “Everything” in the monolith (plus databases) — *had dependency constraints*

Wishes

Beginning of year 2015

Wishes of the stakeholders:

- Features, features, features
- Application must not break

Developers

- We want to code, test, deploy

Ops

- Really!?

The Plan

Beginning of 2015

- New features aren’t built into the monolith anymore,

  but as a separate applications

- We have strong guidelines regarding

  - API

  - Monitoring interfaces

  - Logging

The Plan

Continued...

- While building new functionality in to micro-services, existing features were extracted from the monolith (scoop out) to allow faster, independent development of features
- This *should* remove all BL from the monolith *soon*

Containers? Yes, but no.

How should we manage all those new applications?

The pressure of having a perfectly working runtime-environment soon was quite high.

Pragmatic decision: ***poor-mans micro-services***

- @devs: please package your app in a .deb-package
- we do the rest via Ansible and HAProxy

Bring this to life, then move on

Our solution Containers

Containers? Yes, but how?

How should we manage all those containers?

Should we use the early versions of Kubernetes or Mesosphere Marathon?

No.

We wanted to have an environment

we were able to *understand*, *automate* and *manage*.

So we created a custom Docker-environment with Docker, Consul & Swarm.

Our solution consists of...

- Debian machines (VMs & Metal)
- Docker-CE
- Docker Swarm (not swarm-mode)
- Consul
- Consul-Template
- Dnsmasq
- Nginx
- Deployment with Ansible
- Secrets managed by “Ops"

  - sorry, no “Hashicorp Vault”, yet

Reinventing the wheel?

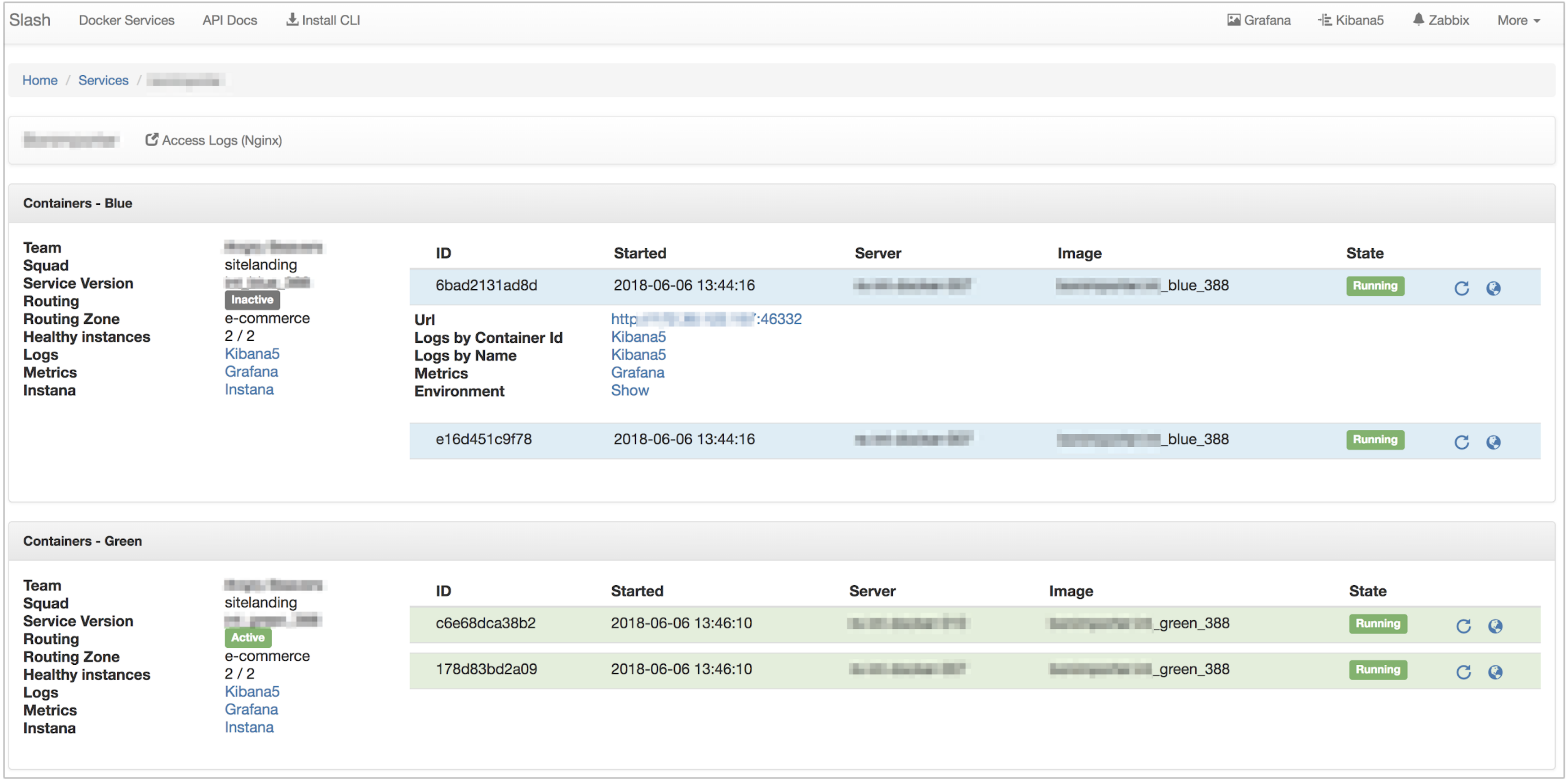
We say: No.

Because we created a solution *we* were capable to run and fits *our* needs.

“Best Practises” don’t work for everybody.

Only downsides so far:

- Docker swarm isn’t good at “deploy spread”
- We’ve no orchestration-service that ensures our containers are running fine and in the right number of instances



Home / Services / [redacted]

[redacted] Access Logs (Nginx)

Containers - Blue

Team	[redacted]
Squad	sitelanding
Service Version	[redacted]
Routing	Inactive
Routing Zone	e-commerce
Healthy instances	2 / 2
Logs	Kibana5
Metrics	Grafana
Instana	Instana

ID	Started	Server	Image	State
6bad2131ad8d	2018-06-06 13:44:16	[redacted]	[redacted]_blue_388	Running
e16d451c9f78	2018-06-06 13:44:16	[redacted]	[redacted]_blue_388	Running

Url	http[redacted]:46332
Logs by Container Id	Kibana5
Logs by Name	Kibana5
Metrics	Grafana
Environment	Show

Containers - Green

Team	[redacted]
Squad	sitelanding
Service Version	[redacted]
Routing	Active
Routing Zone	e-commerce
Healthy instances	2 / 2
Logs	Kibana5
Metrics	Grafana
Instana	Instana

ID	Started	Server	Image	State
c6e68dca38b2	2018-06-06 13:46:10	[redacted]	[redacted]_green_388	Running
178d83bd2a09	2018-06-06 13:46:10	[redacted]	[redacted]_green_388	Running

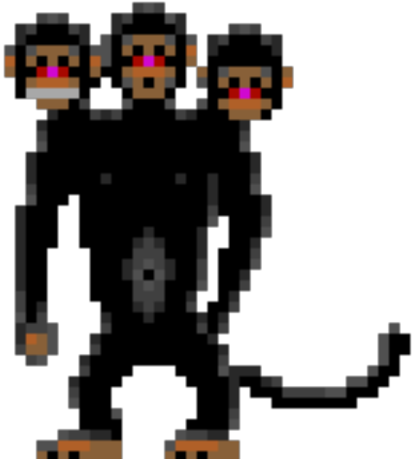
All is fine now?

What about

- Monitoring —> Check.
- Responsibility —> ? … depends
- **THE MONOLITH?**

The Monolith...

... is still in production.

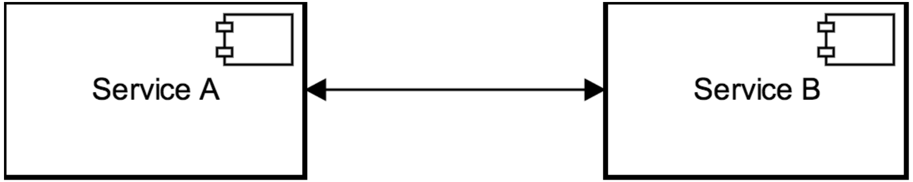


Scaling **Services**

Scale at Servicelevel

Our 45 teams are developing and running more than 150 services

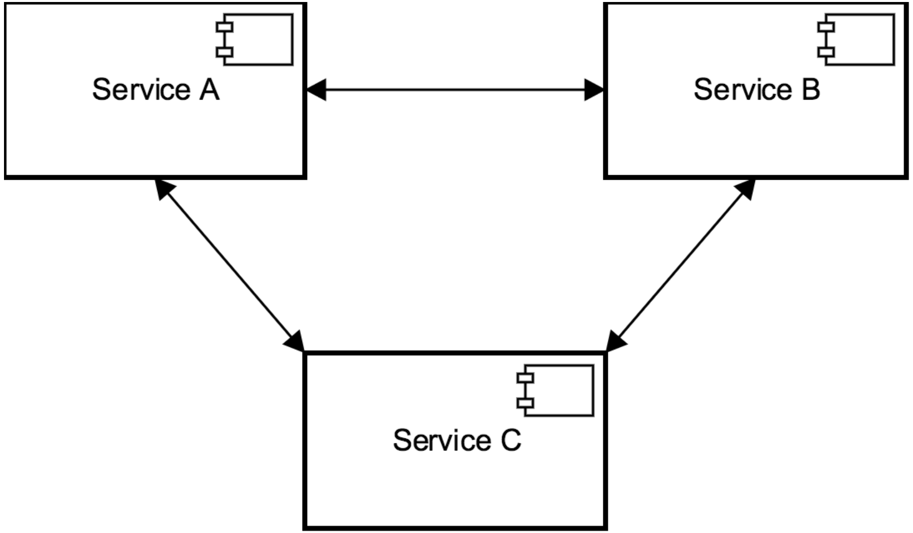
Imagine if all of them talk to each other:



Scale at Servicelevel

Our 45 teams are developing and running more than 150 services

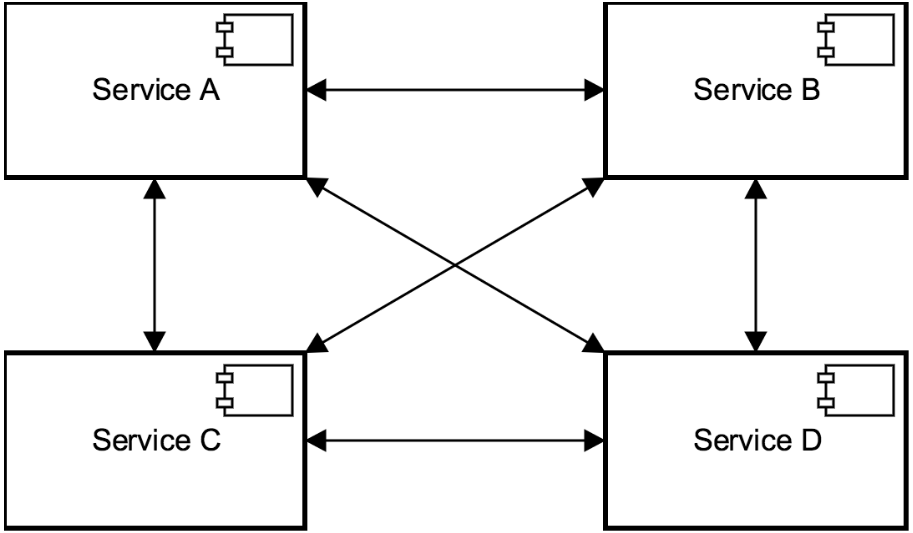
Imagine if all of them talk to each other:



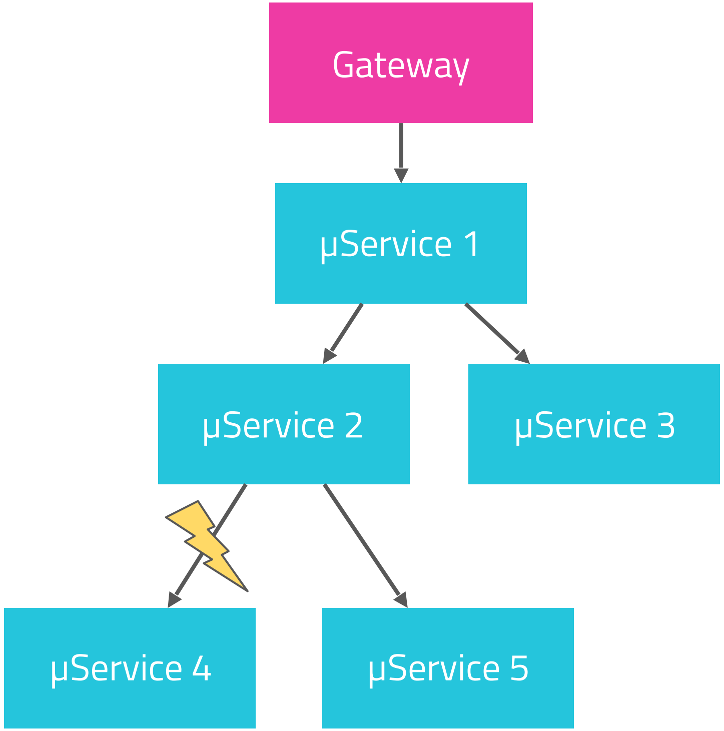
Scale at Servicelevel

Our 45 teams are developing and running more than 150 services

Imagine if all of them talk to each other:



Challenges in HTTP/REST-only architectures



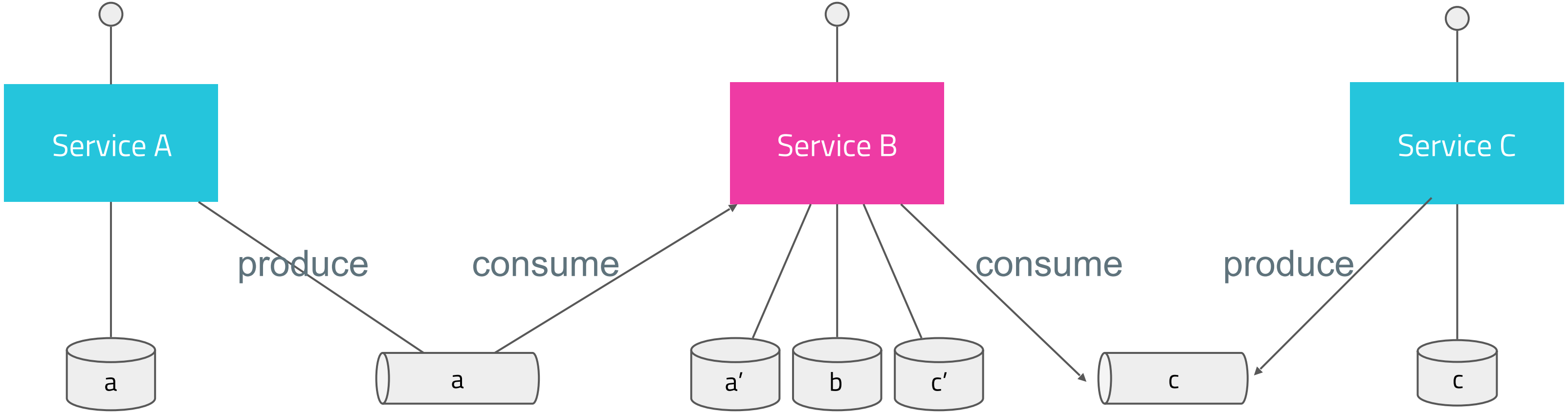
- API-Guidelines
- Timeouts
- Fallbacks
- Circuit Breakers
- **Eventing**

REWE digital

What is **Eventing?**

What is the goal of Eventing?

- Enable services to provide themselves with data asynchronously before it is needed in a request
  — Having data is better than needing data.
- „Kind of database replication“
- More performance & stability

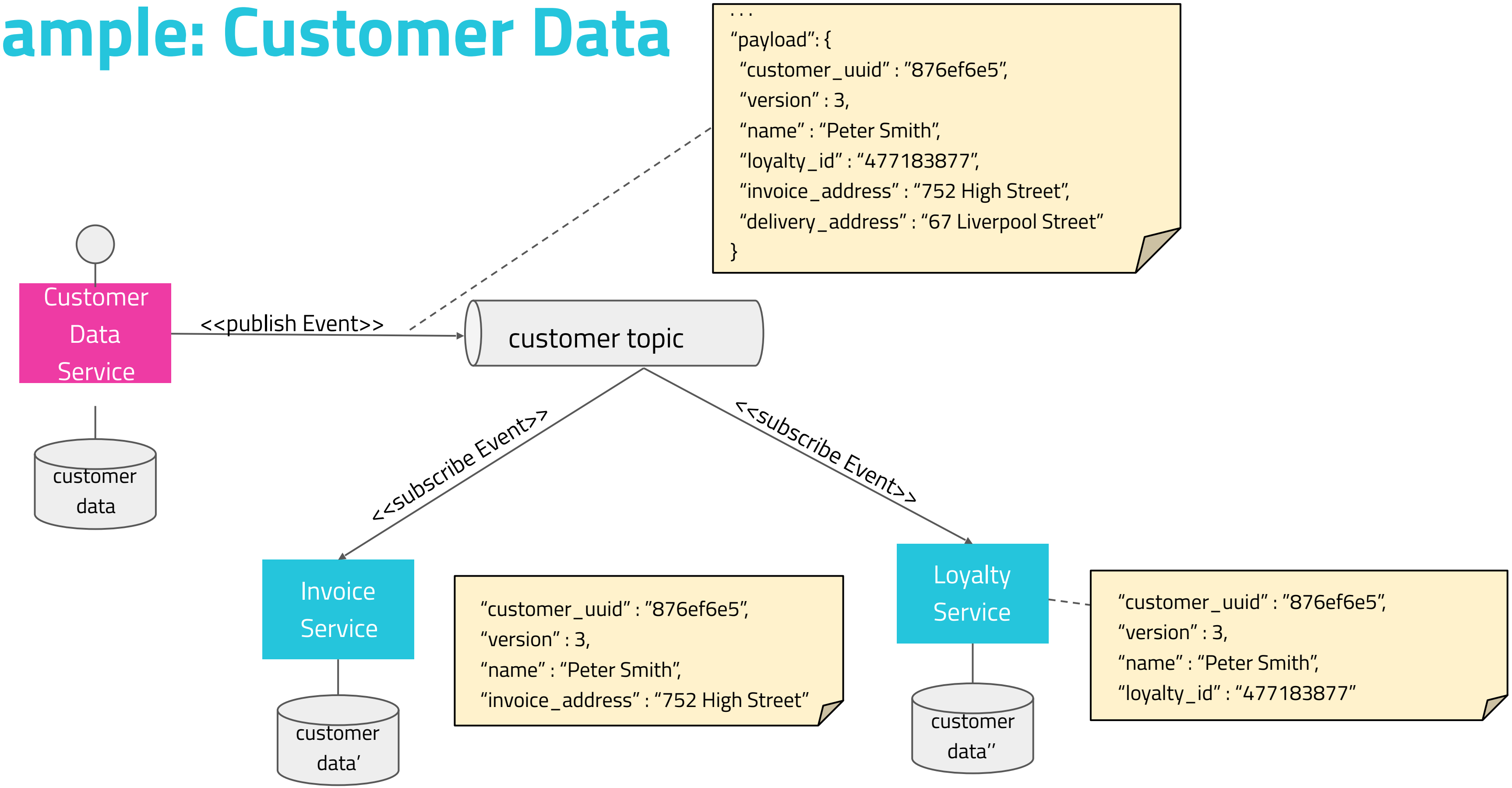


Technical Event

- ID: Unique identifier
- Key: Which entity is affected?
- Version: Which version of this entity is this?
- Time: When did the event occur?
- Type: What kind of action happened?
- Payload: What are the details?
    - Entire entity - not deltas!

```
{
“id” : “4ea55fbb7c887”,
“key” : “7ebc8eeb1f2f45”,
“version” : 1,
“time” : "2018-02-22T17:05:55Z",
“type” : “customer-registered”,
“payload” : {
  “id” : “7ebc8eeb1f2f45”,
  “version” : 1,
  “first_name” : “Paul”,
  “last_name” : “Puschmann”,
  “e-mail” : “bofh(at)rewe-digital.com”
}
}
```

Example: Customer Data



```
. . .
“payload”: {
  “customer_uuid” : ”876ef6e5”,
  “version” : 3,
  “name” : “Peter Smith”,
  “loyalty_id” : “477183877”,
  “invoice_address” : “752 High Street”,
  “delivery_address” : “67 Liverpool Street”
}
```

```
“customer_uuid” : ”876ef6e5”,
“version” : 3,
“name” : “Peter Smith”,
“invoice_address” : “752 High Street”
```

```
“customer_uuid” : ”876ef6e5”,
“version” : 3,
“name” : “Peter Smith”,
“loyalty_id” : “477183877”
```

We chose **Apache Kafka**

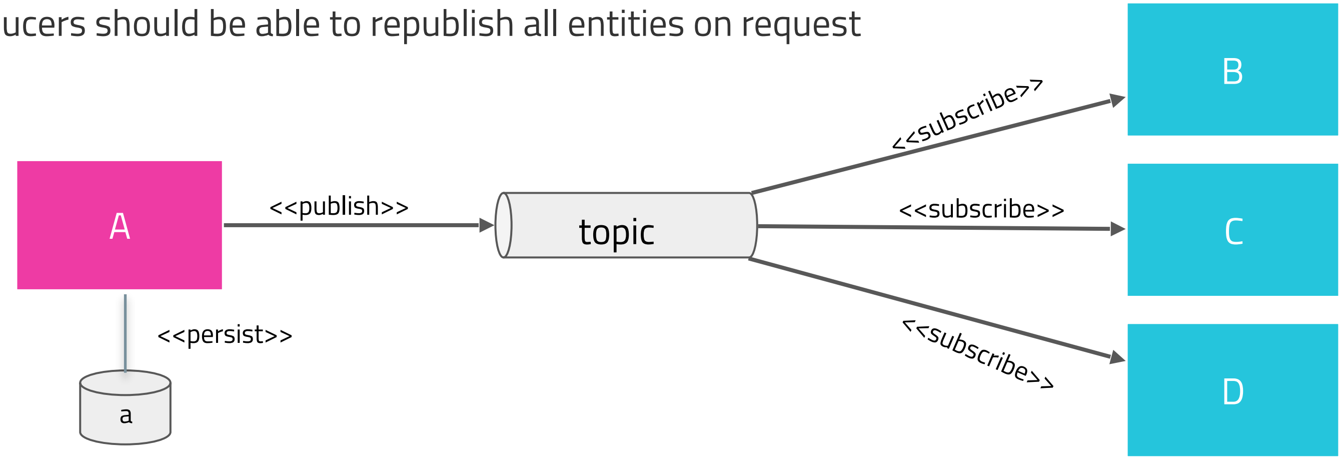
Apache Kafka

“Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.” (https://kafka.apache.org/)

- Open-source stream processing platform written in Scala and Java
- High-throughput, low-latency platform for real-time data streams
- Originally developed at Linkedin, open sourced in 2011
- Offers 4 APIs: **Producer, Consumer, Stream, Connect**
- We use Apache Kafka in a pub-sub manner. This means most of our services use the Producer and Consumer APIs

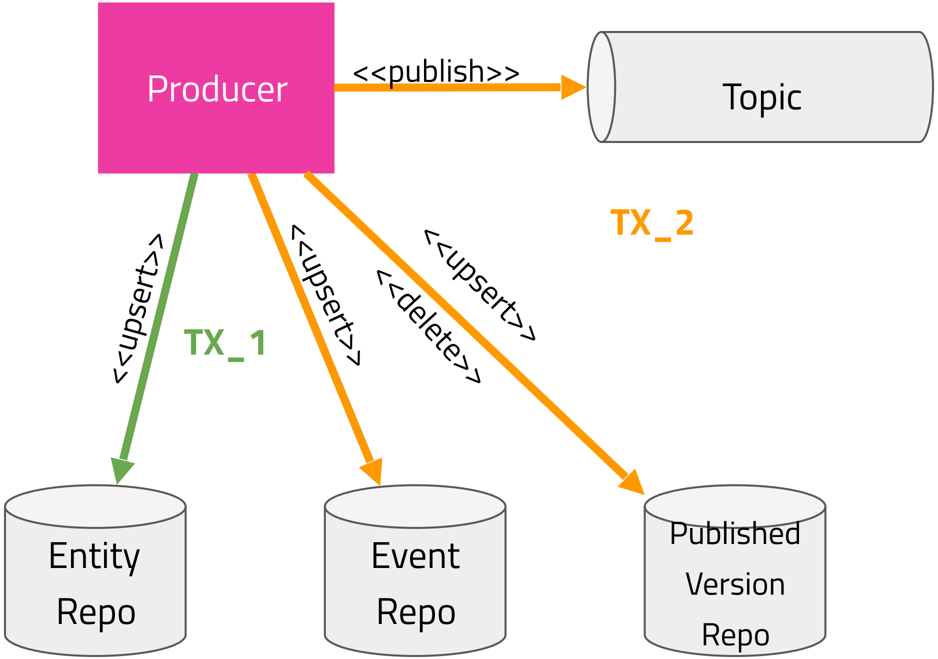
Producers

- Every service which owns a resource should publish those resource-entities to a topic
- Use only one producer or make sure there are no issues about the order of events
- To enable log-compaction use a partitioner that ensures an event with the same key is always sent to the same partition
- All producers should be able to republish all entities on request



Producers

- The producer has to make sure that the message is delivered and committed
- Therefore we store the raw event in a database to enable retries until it’s committed to the cluster
- Scheduled jobs can take care of retries and cleanup



Consumers

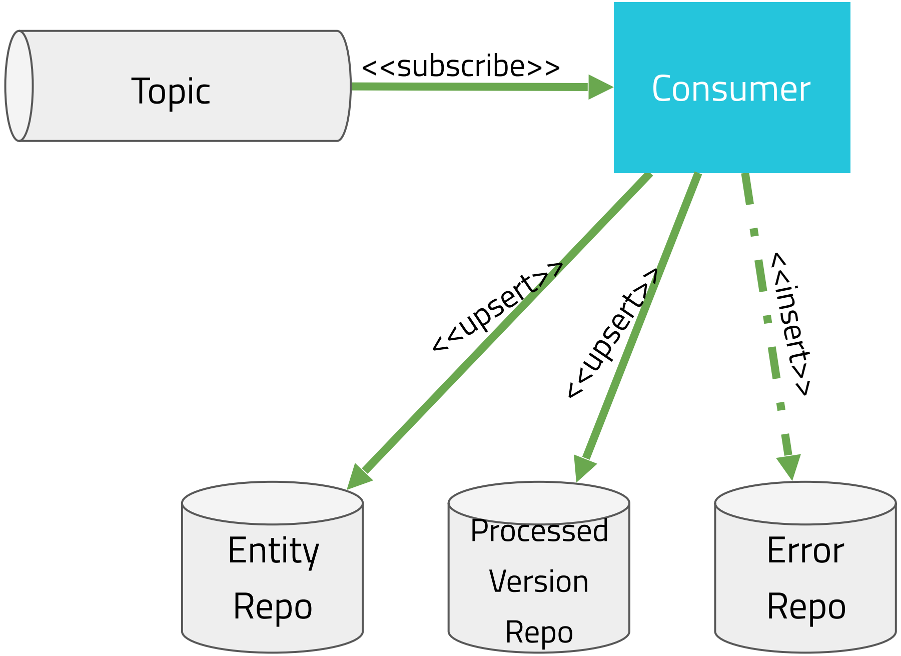
- Every service can consume every available data and should consume all data it needs to fulfil a request - **having data at request time** is better than trying to get it from another service
- The consumer has to process events idempotently. An event could be consumed more than once. The infrastructure ensures **at-least-once** delivery
- Consumers have to take care of deployment specialties like blue/green
- Consumers should be able to re-consume from the beginning.
  For instance if local data-model needs changes or additional data
- Consumers only should persist the data they really need

Consumers

- The consumer is responsible for a manual commit only after a successful processing of the event. Successful can mean:

- Needed data from an event is saved in the services data-store

- The event can’t be processed and is stored in a private error queue / table



Kafka & Ops

Pros

- **Each service has its own database:**

  This impacts / supports migrations, query tuning, database usage
- **Topic replication / mirror:**

  The replication of topics to different brokers offer support for a second datacenter or migration to different environments
- **Asynchrony:**

  Services don't need no do synchronous calls to share their data with other services

Contras

- **Another super important service:**

  Kafka is the hub of your business data. Take care about this.
- **Redundancy of data:**

  Your databases will store the same data, or subsets, more than once
- **Asynchrony:**

  A consumer may not be up-to-date with some topics, this might lead to inconsistencies, e.g. in the frontend

Kafka & Ops

Eventing benefits for Operation

By using the concept of “Kafka-mirrors”,

you can push selected topics to a different Kafka-Cluster (one-way).

This way you easily can setup services as *consumers* at a different datacenter.

For *producers* you’d shut down the producer, switch the direction

of the “kafka-mirror” and the start the producer “at the other side”.

Optionally: delete the topic, create and fill it anew.

Possible alternative:

create Kafka-Clusters spreading over datacenters and use „rack-awareness“

Helpful things

What helped us most?

- Strong Architecture-Guild:

  - Eventing-Guide

  - API-Guide

  - ... and many more

- Active Communication of changes & constraints
- Monthly / Bi-monthly Bootcamps for (new) colleagues

What helped us most?

Continued ...

- Introduction of Eventing (with Kafka)
- Make development teams analyse logs & metrics on their own
    - Strong usage of ELK
    - Strong usage of Prometheus
- External traffic (Web, mobile App, partners)

  always has to get routed through a gateway (service)

What we Learned

What did we learn?

- Communication is a key factor
- Automation pays off
- Eventing with Kafka is cool
- Temporary solutions last *very* long
- The knowledge / distribution of RACI-model helps (RACI-matrix)
- UBIURI (you build it, you run it) is not only an option

What did we learn?

Continued …

We did try to scoop out the Monolith. —> That was not a good idea.

Perhaps better:

Put a gateway in front of your legacy-application and switch resource by ressource.

Every service must have an owner!

The Future

The Future

… will be different in many ways.

- UBIURI / You build it, you run it
- SRE-pattern ?
- No more Devs + Ops but DevOps?

We’ll see…

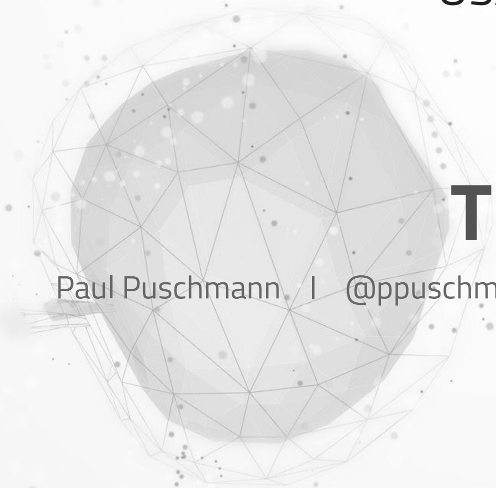
The Future

Google Cloud Platform

kubernetes

From Monolith to Microservices

OSAD 2018, Munich



Thank You!

Paul Puschmann | @ppuschmann | www.rewe-digital.com | @rewedigitaltech