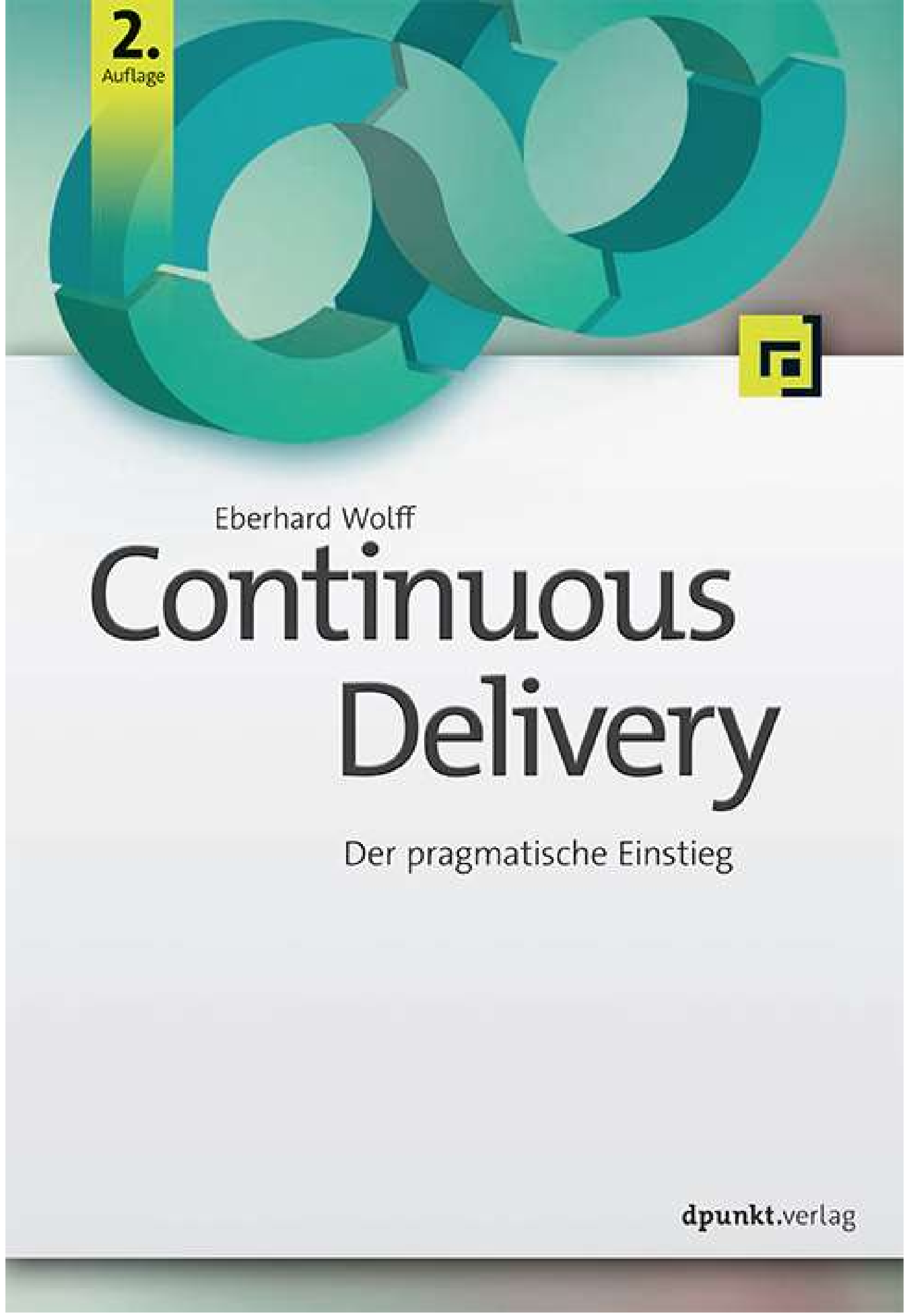
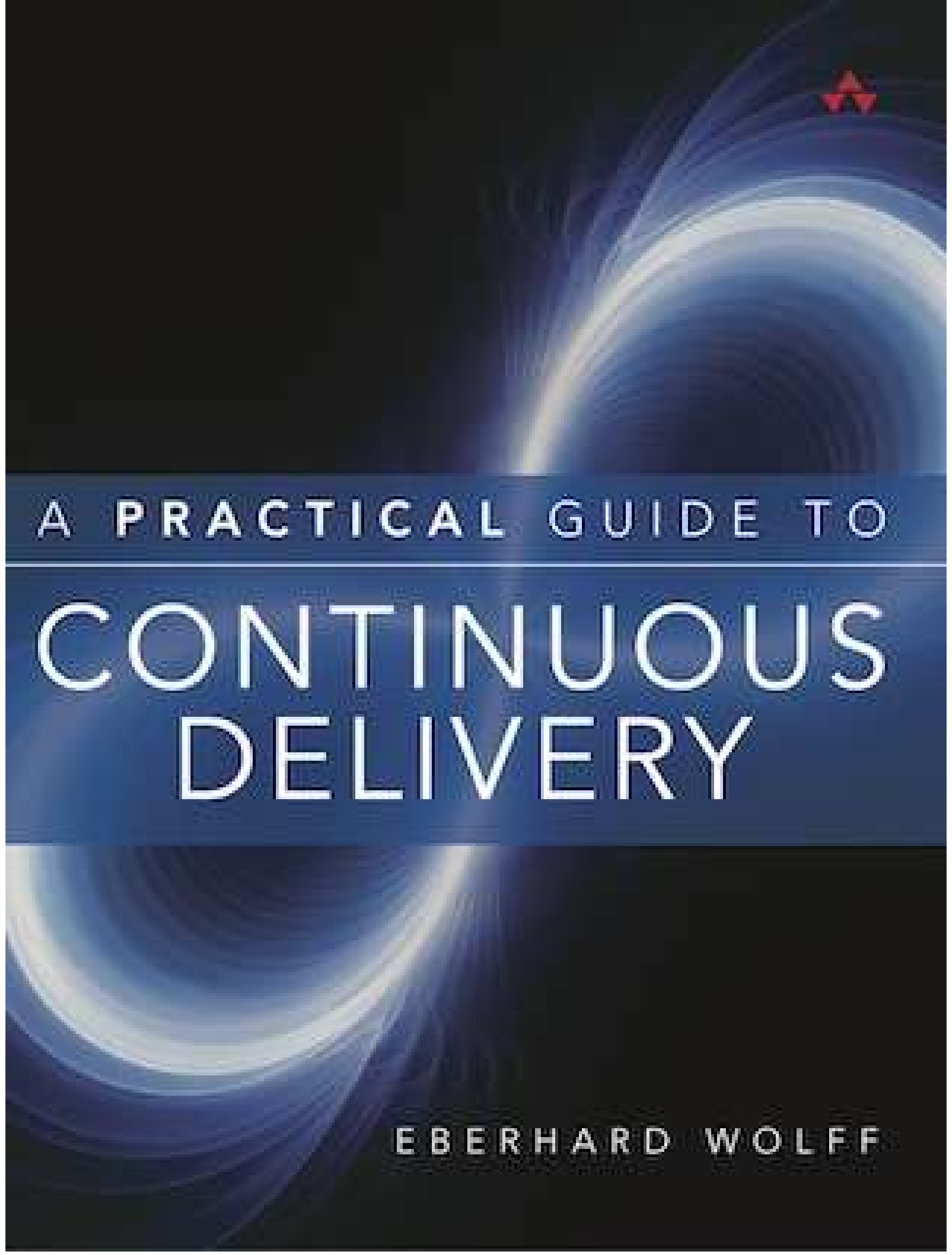
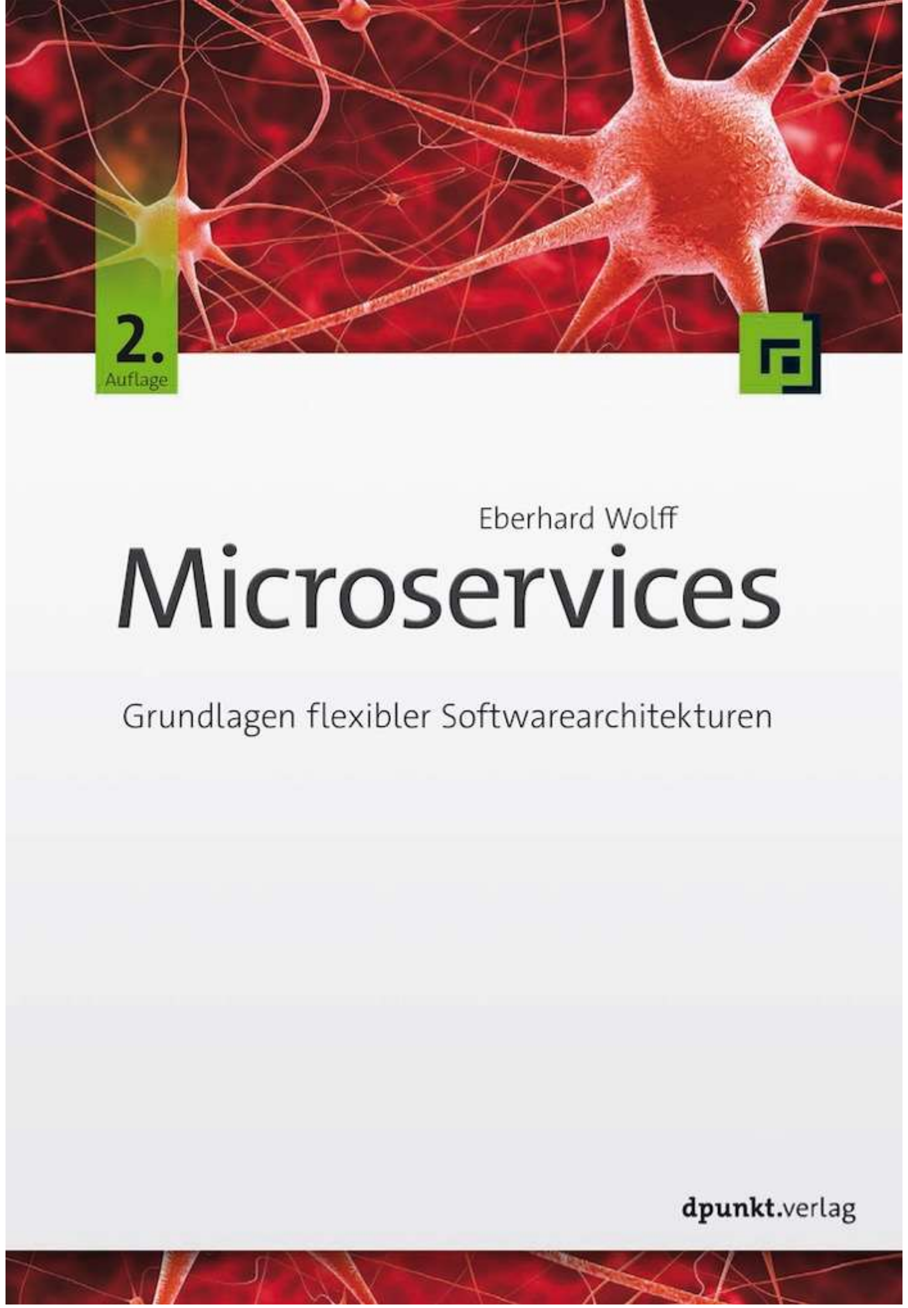
Migrating Monoliths to Microservices: A Survey of Approaches

Eberhard Wolff
Fellow
INNOQ
@ewolff
http://ewolff.com

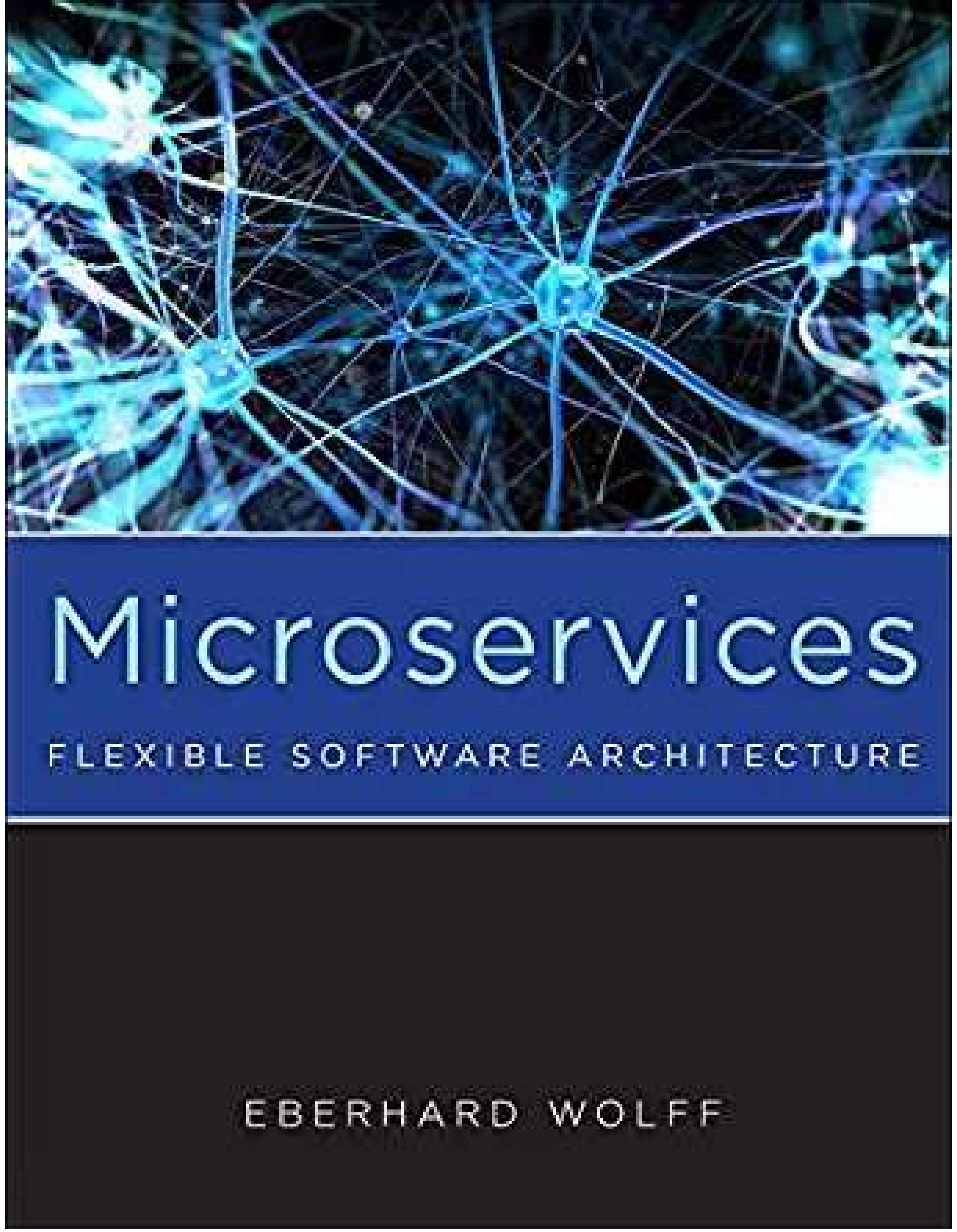




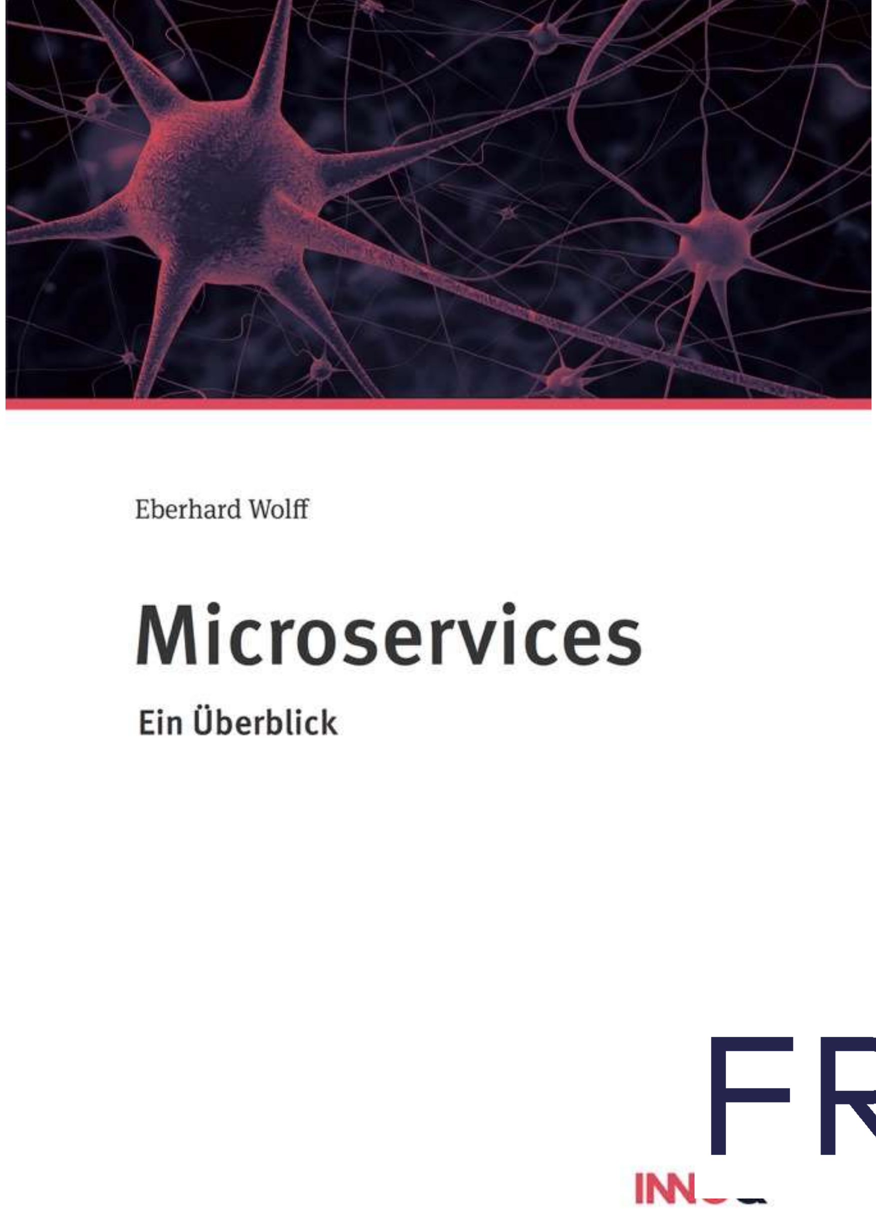
http://continuous-delivery-buch.de/ http://continuous-delivery-book.com/



http://microservices-buch.de/

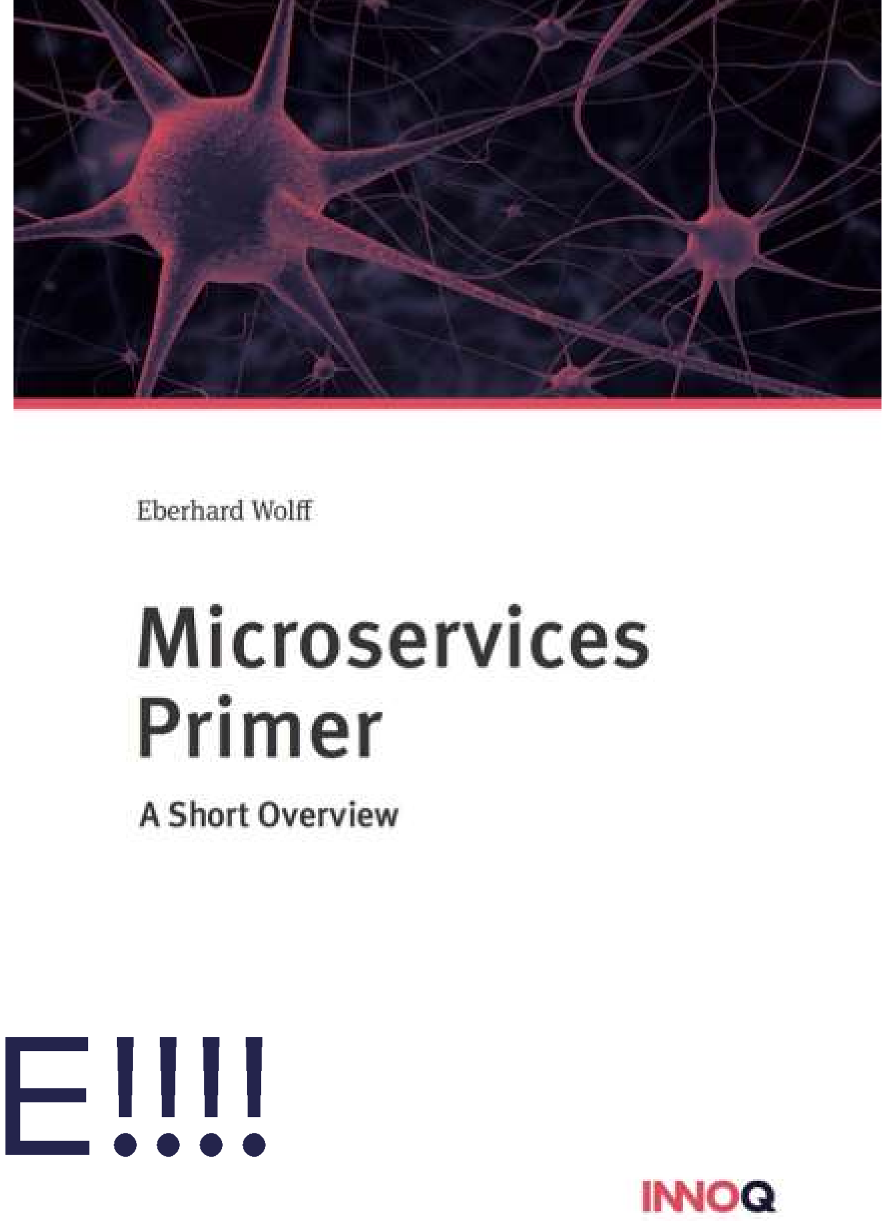


http://microservices-book.com/


Eberhard Wolff

Microservices

Ein Überblick


Eberhard Wolff

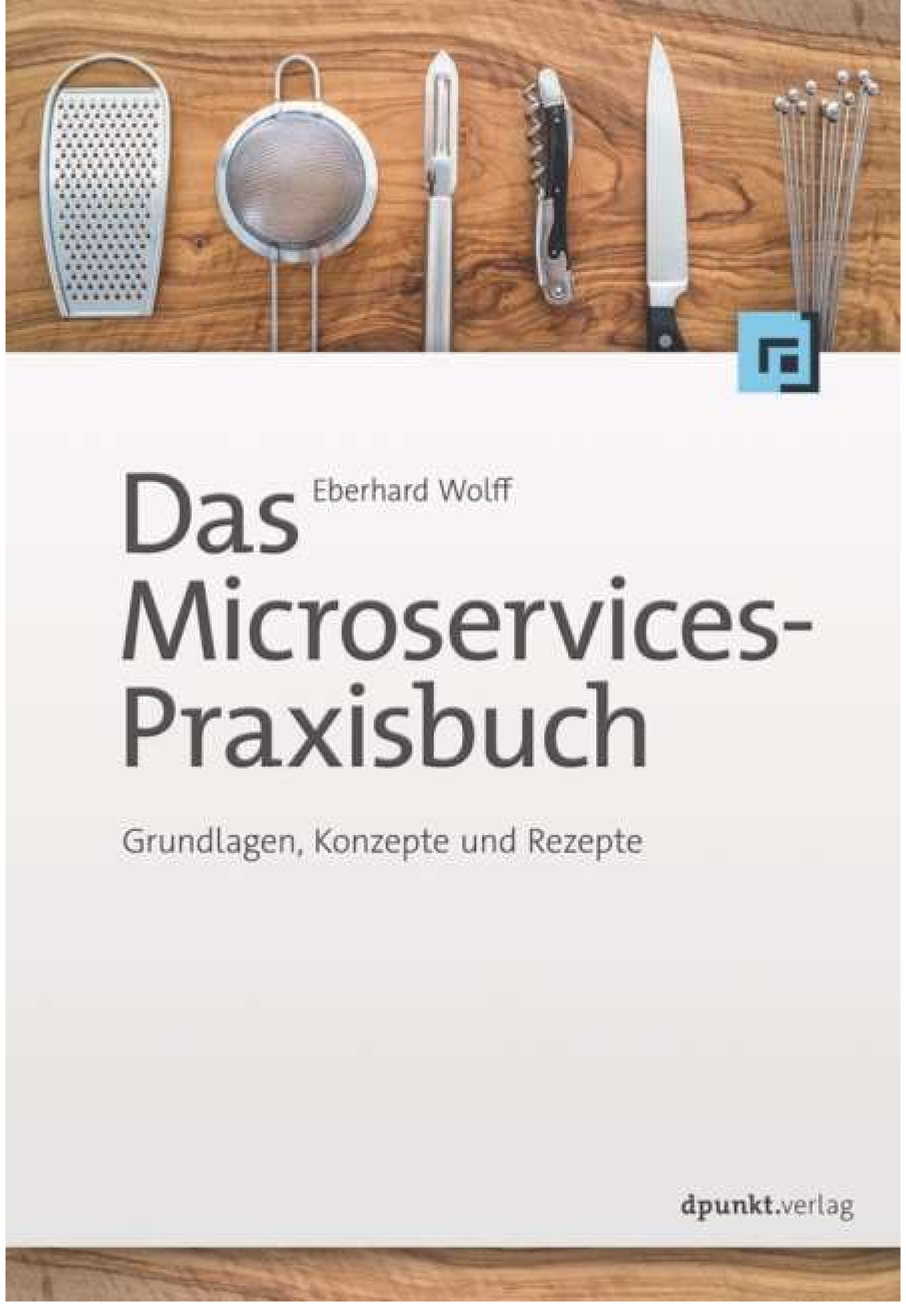
Microservices Primer

A Short Overview

FREE!!!!

http://microservices-buch.de/ueberblick.html

http://microservices-book.com/primer.html



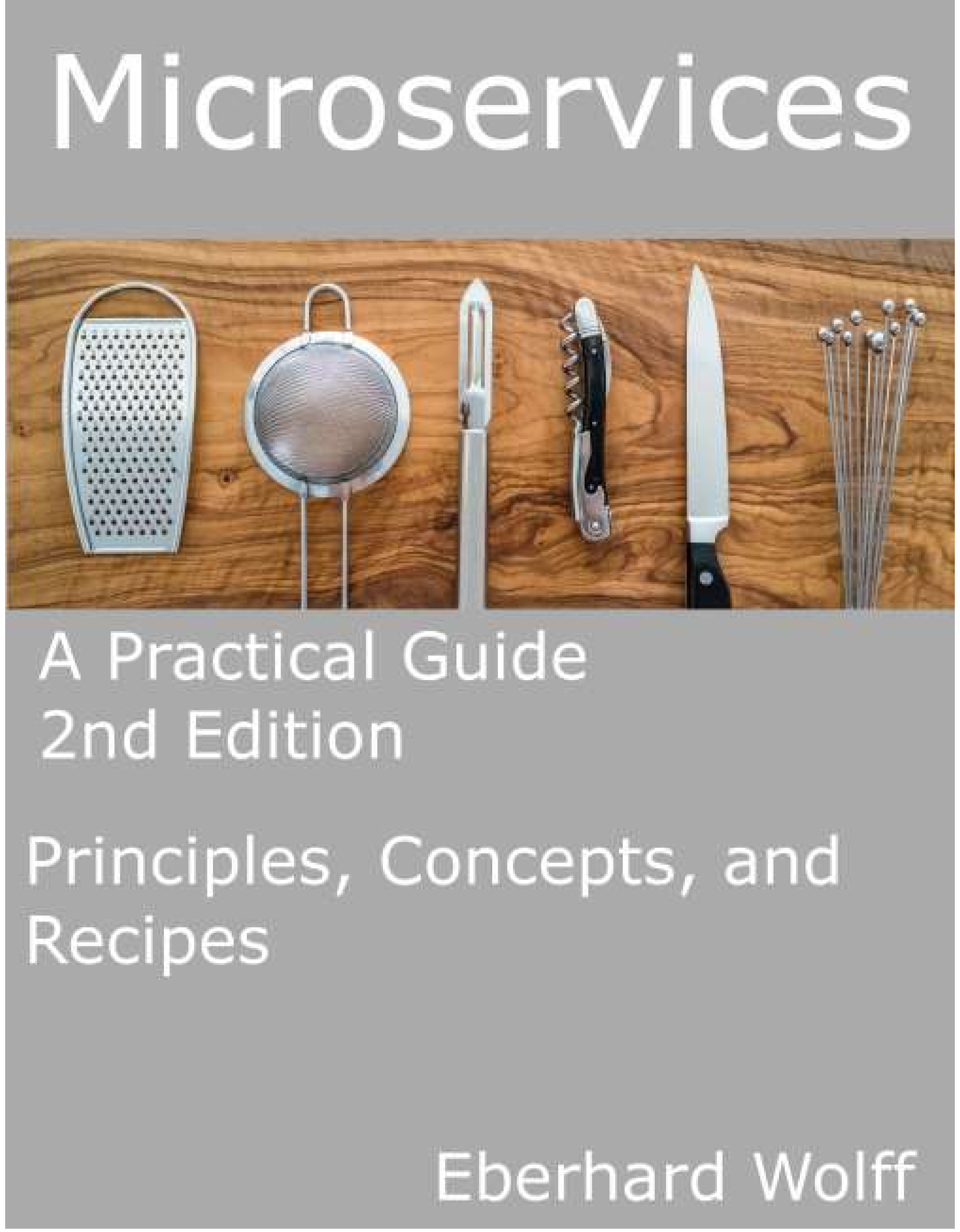
Eberhard Wolff

Das Microservices-Praxisbuch

Grundlagen, Konzepte und Rezepte

dpunkt.verlag

http://microservices-praxisbuch.de



Microservices

A Practical Guide
2nd Edition

Principles, Concepts, and Recipes

Eberhard Wolff

http://practical-microservices.com/

Eberhard Wolff

Microservices Rezepte

Technologien im Überblick

INNOQ

Eberhard Wolff

Microservices Recipes

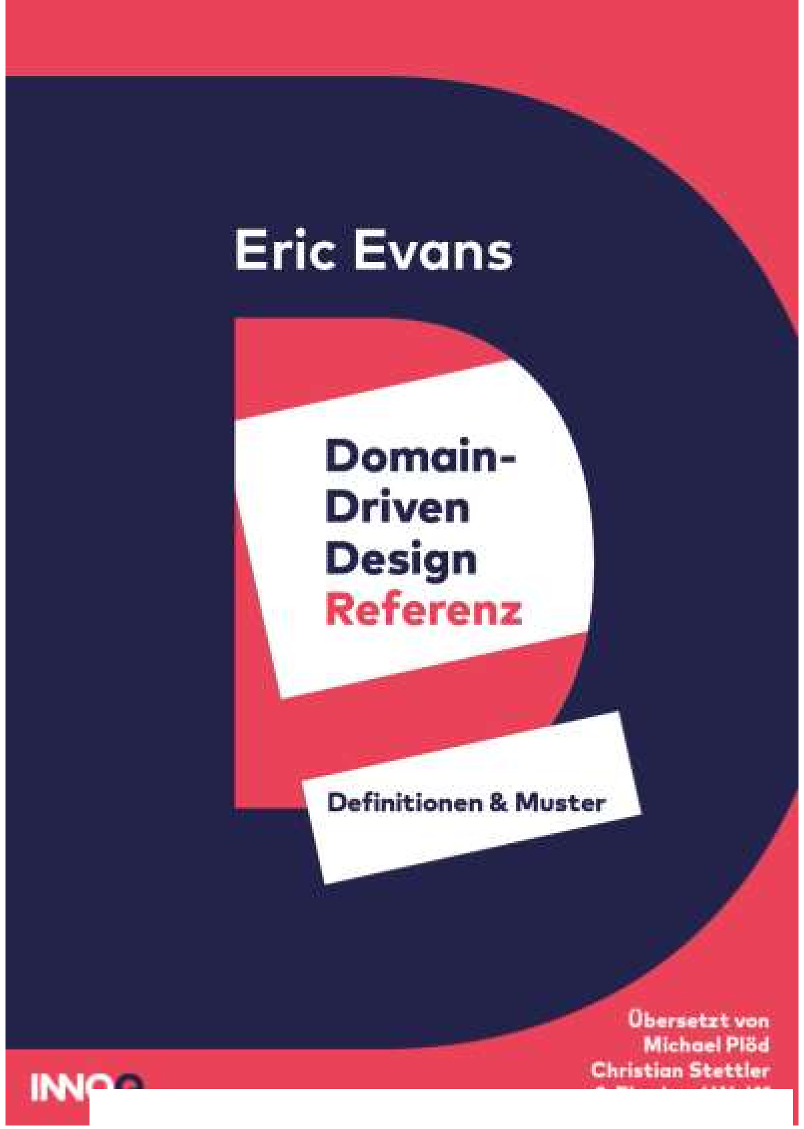
Technology Overview

INNOQ

FREE!!!!

http://microservices-praxisbuch.de/rezepte.html

http://practical-microservices.com/recipes.html



Eric Evans

Domain-Driven Design Referenz

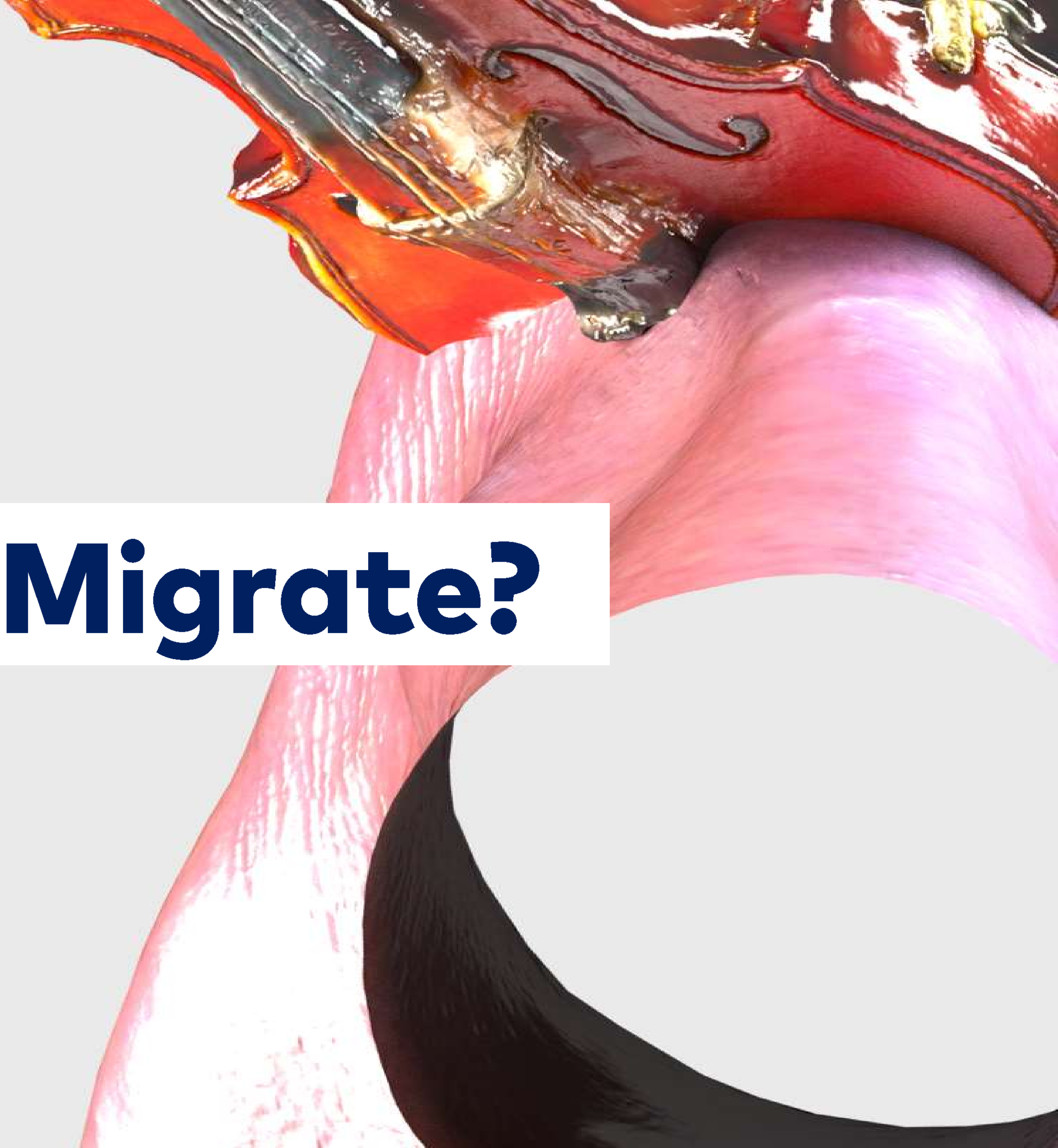
Definitionen & Muster

Übersetzt von Michael Plöd Christian Stettler

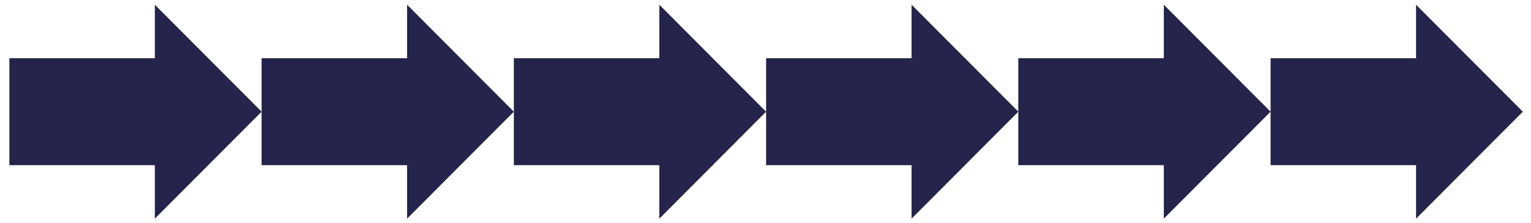
FREE!!!!

http://ddd-referenz.de/
https://domainlanguage.com/ddd/reference/

How to Migrate?



Big Bang



Stepwise

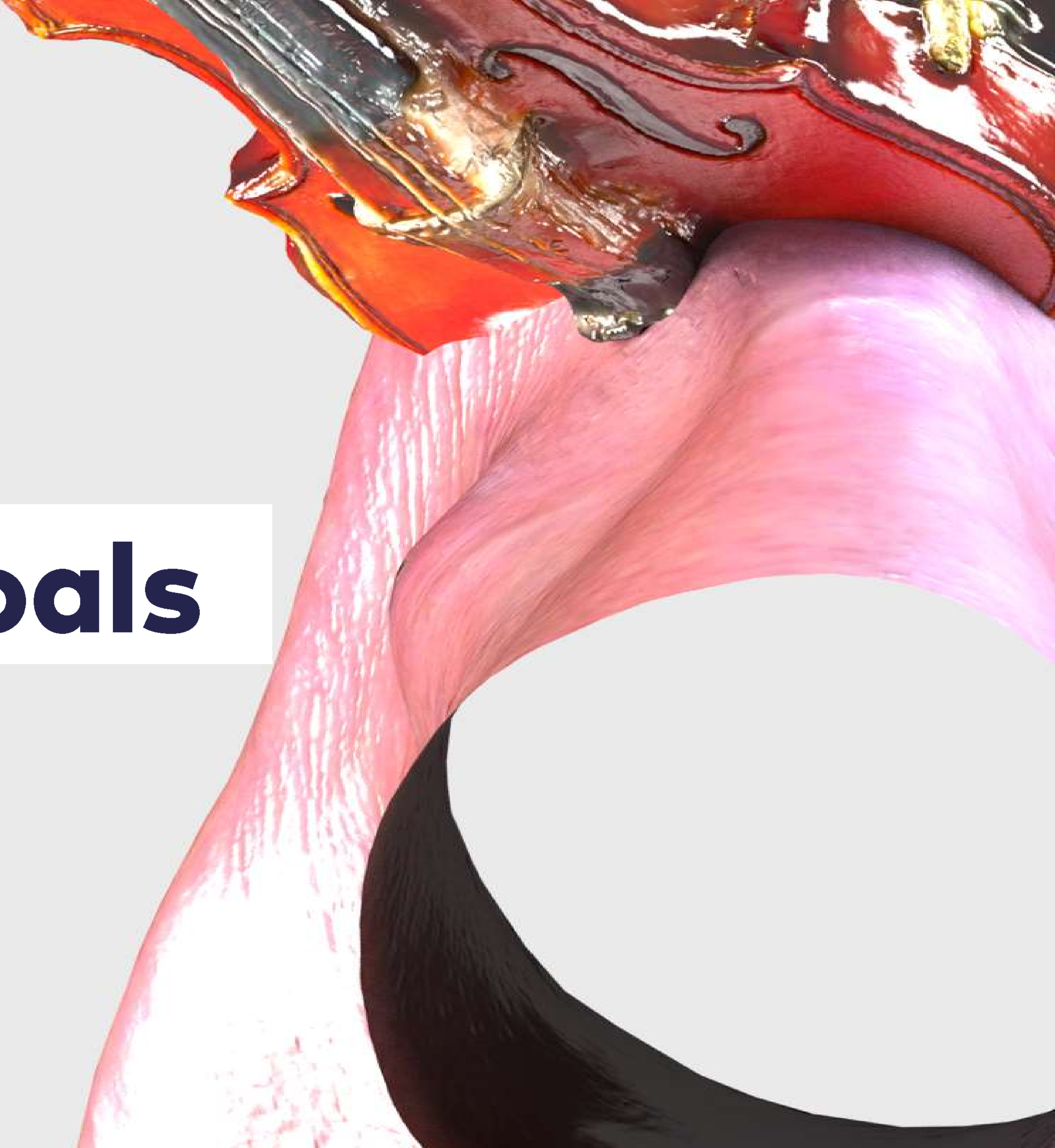
Prefer Stepwise Migration!

Less risk

Faster return on investment

Easier to change priorities

Goals



Goals

Microservices have many benefits:

Independent teams for a large project

Independent scalability

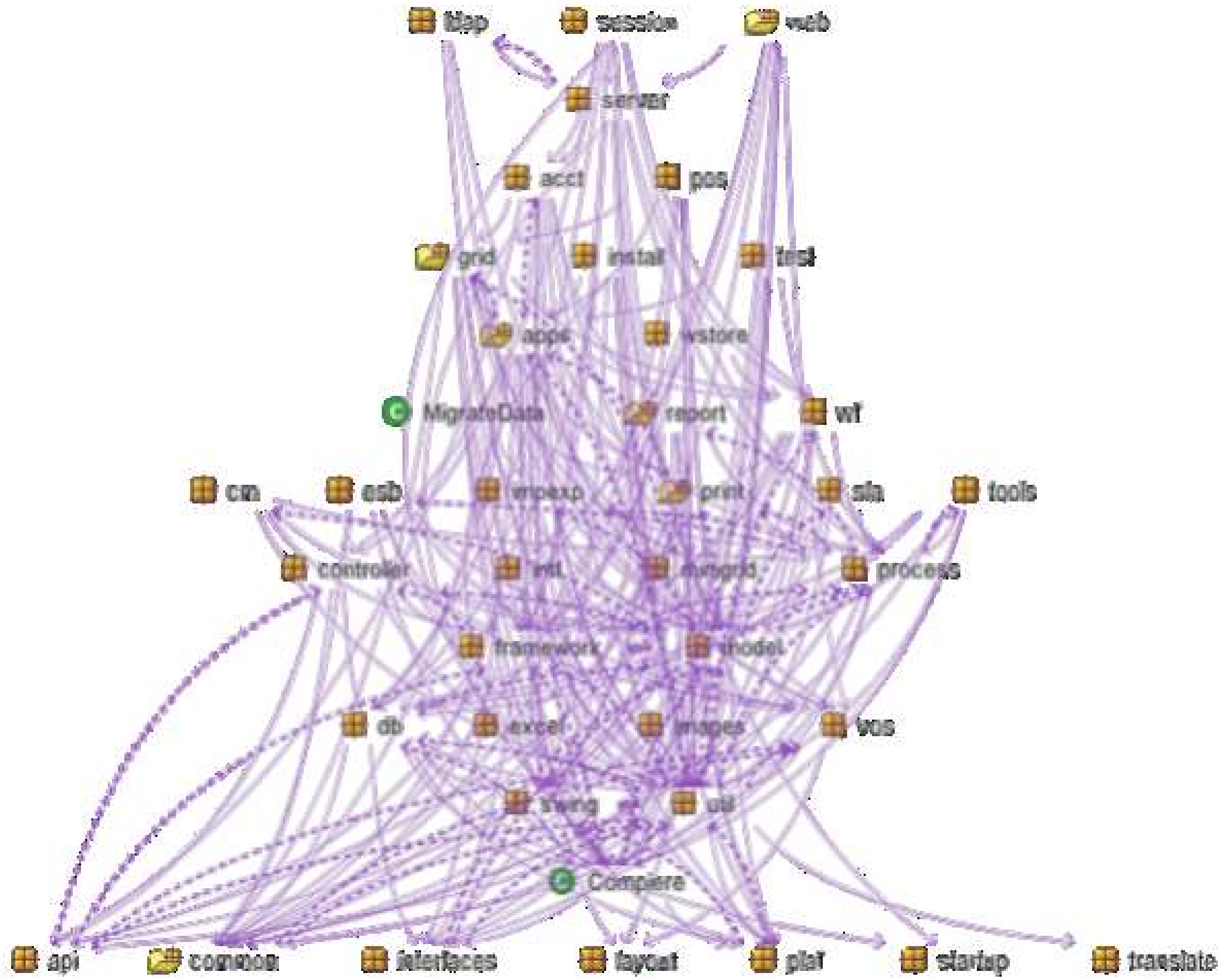
Independent deployment

Security – can add firewalls between microservices

Stability – independent crashes

Typical Goal: Changeability

Typical Legacy System



Constraints: "Black Box Migration"

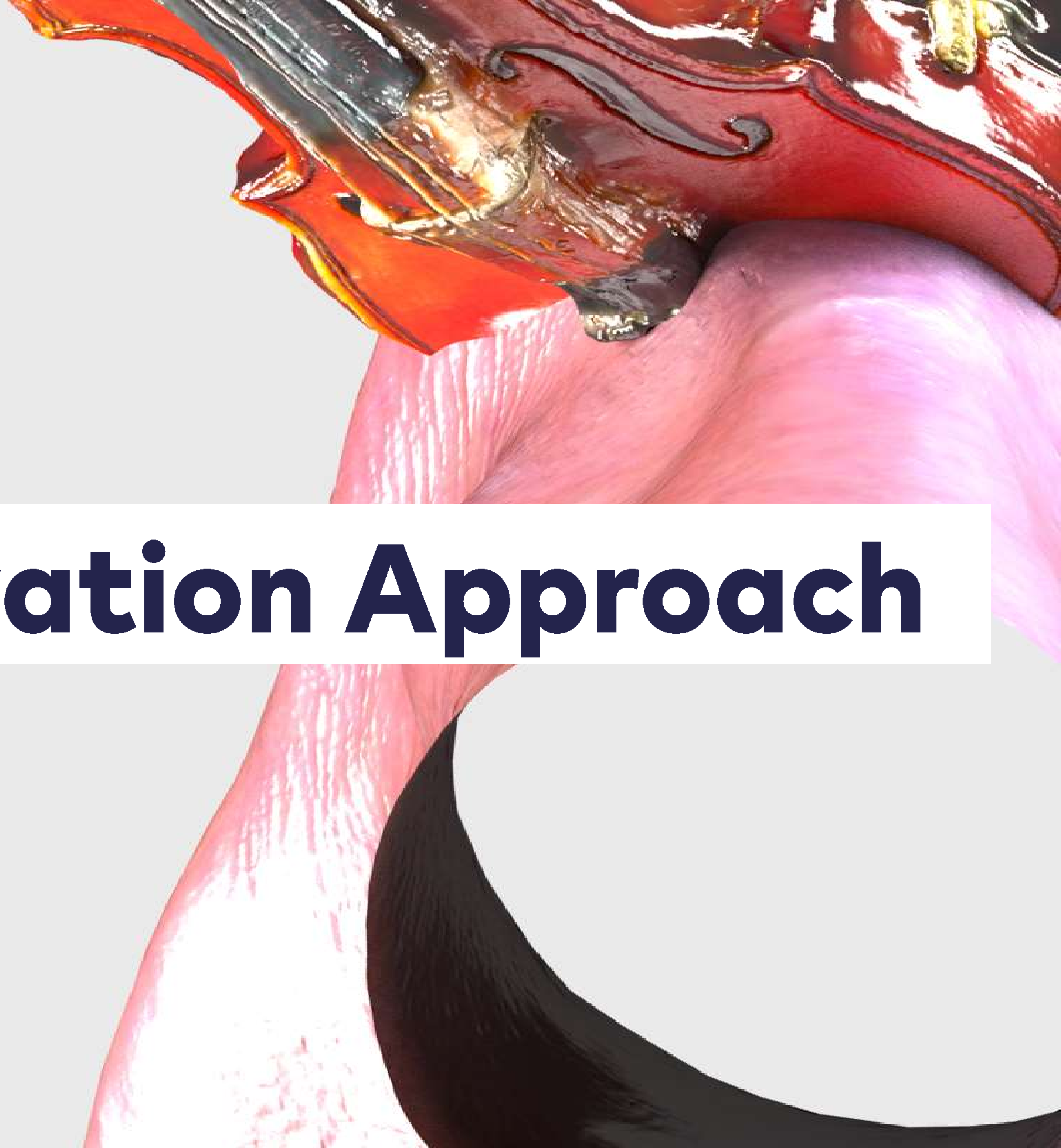
Try to understand as little of the monolith as possible!

Use as little of the monolith as possible!

Ideally: New technology and new architecture

If technology and architecture are great
– why migrate to microservices?

Blueprint Migration Approach



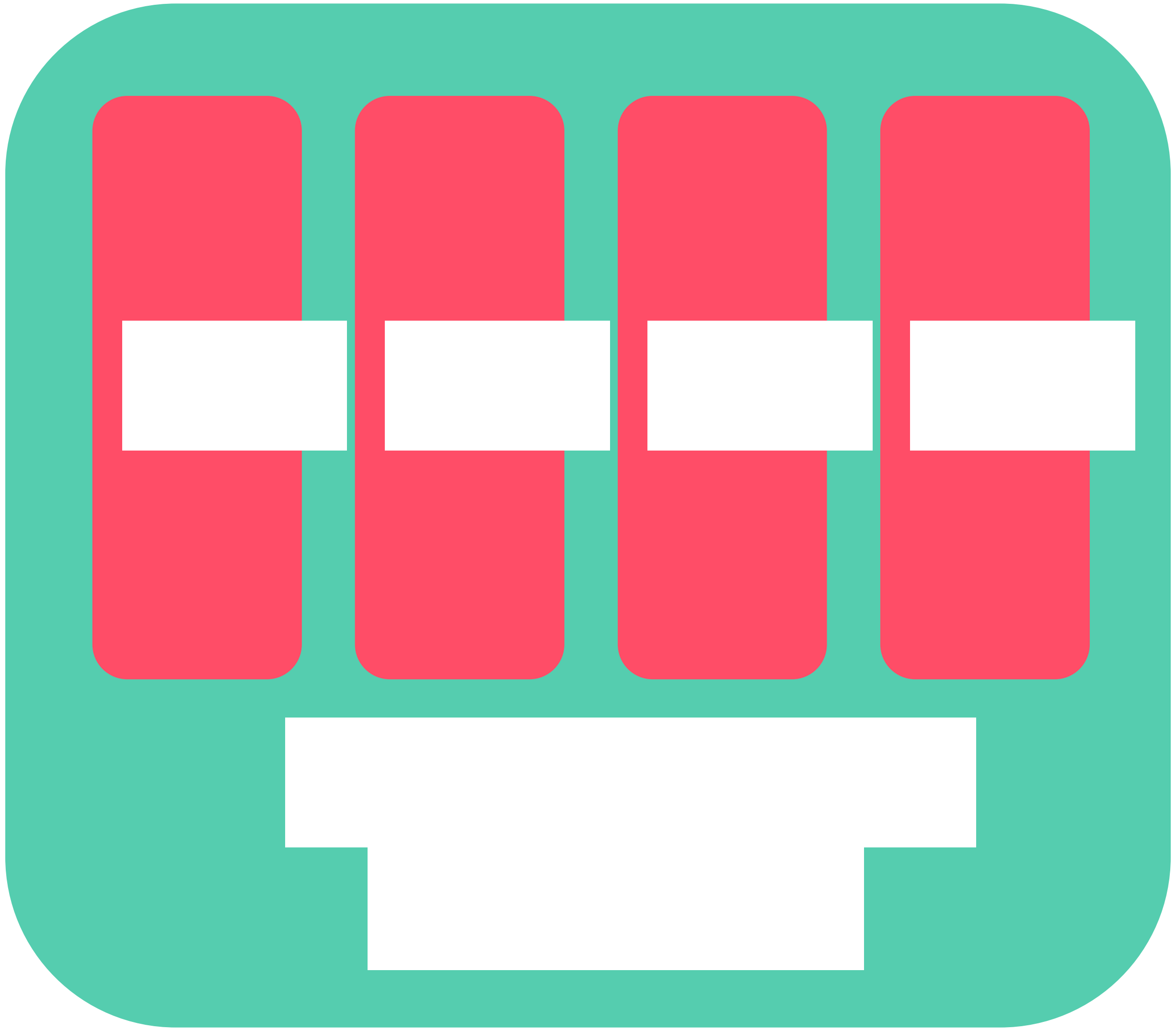
CUSTOMIZE!

Blueprint Migration Approach

Two parallel tasks

Both incremental

Identify Bounded Contexts



Identify Bounded Contexts

From a user's perspective

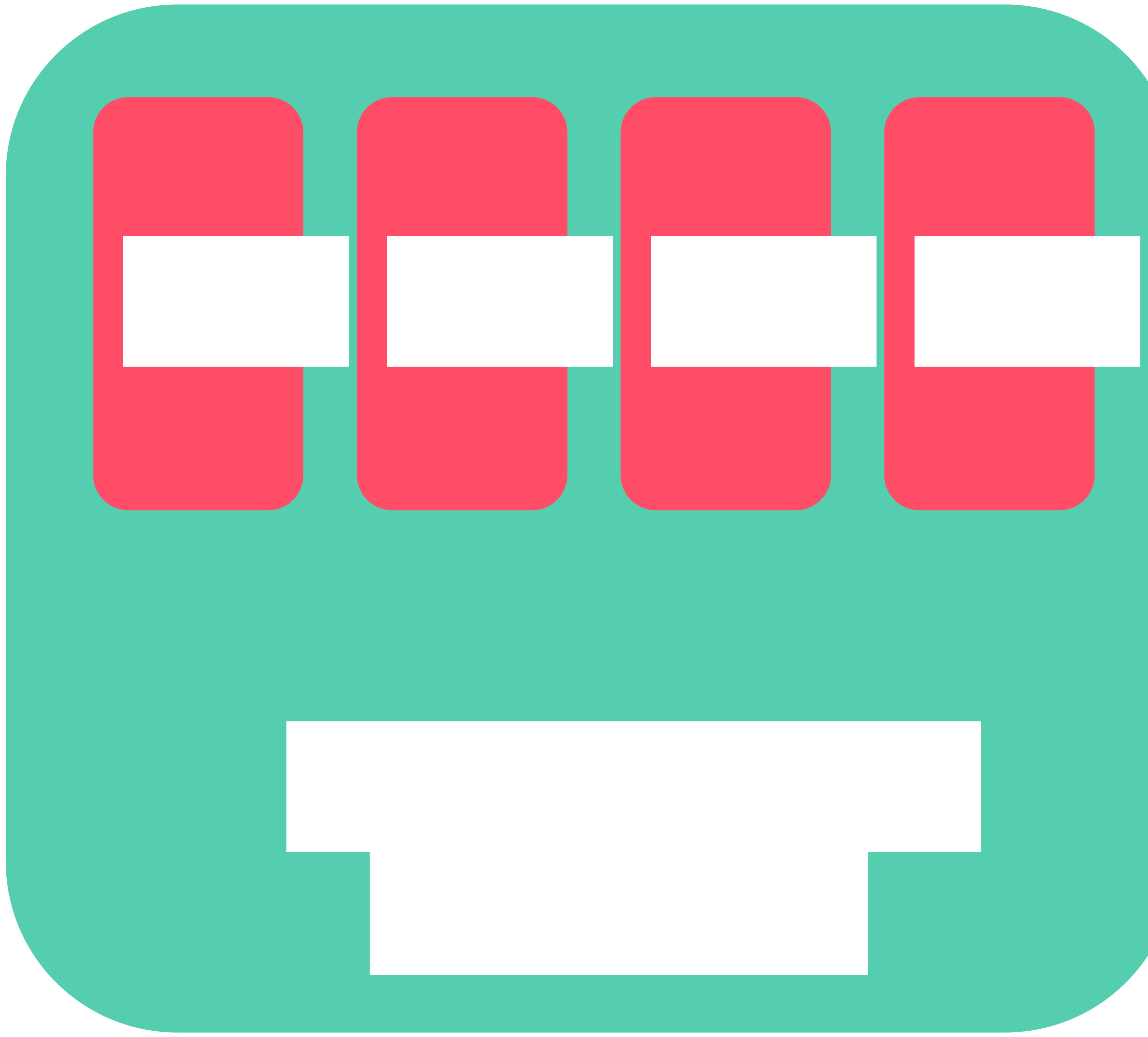
Gives rough idea about ideal architecture

Iterative: the next step is important

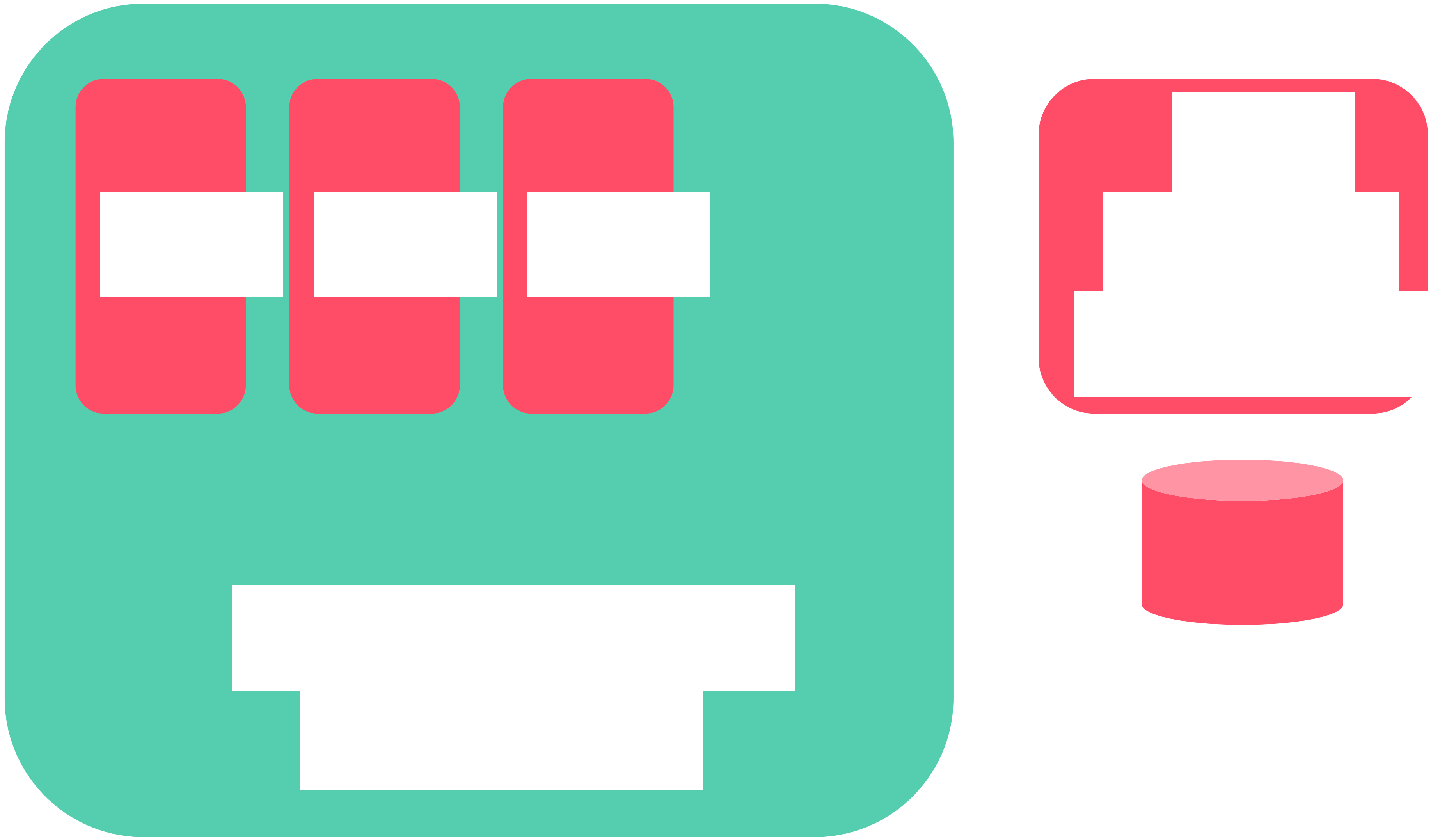
...not the desired end result

"Black box migration": avoid reverse engineering code, database schemas...

Migrate a Bounded Context



Migrate a Bounded Context



Migrate a Bounded Context

Bounded context = separate domain model

Separate domain = separate database schema

Ideally microservice =

bounded context including UI

Better Yet: New Bounded Context

New requirements might justify a new Bounded Context

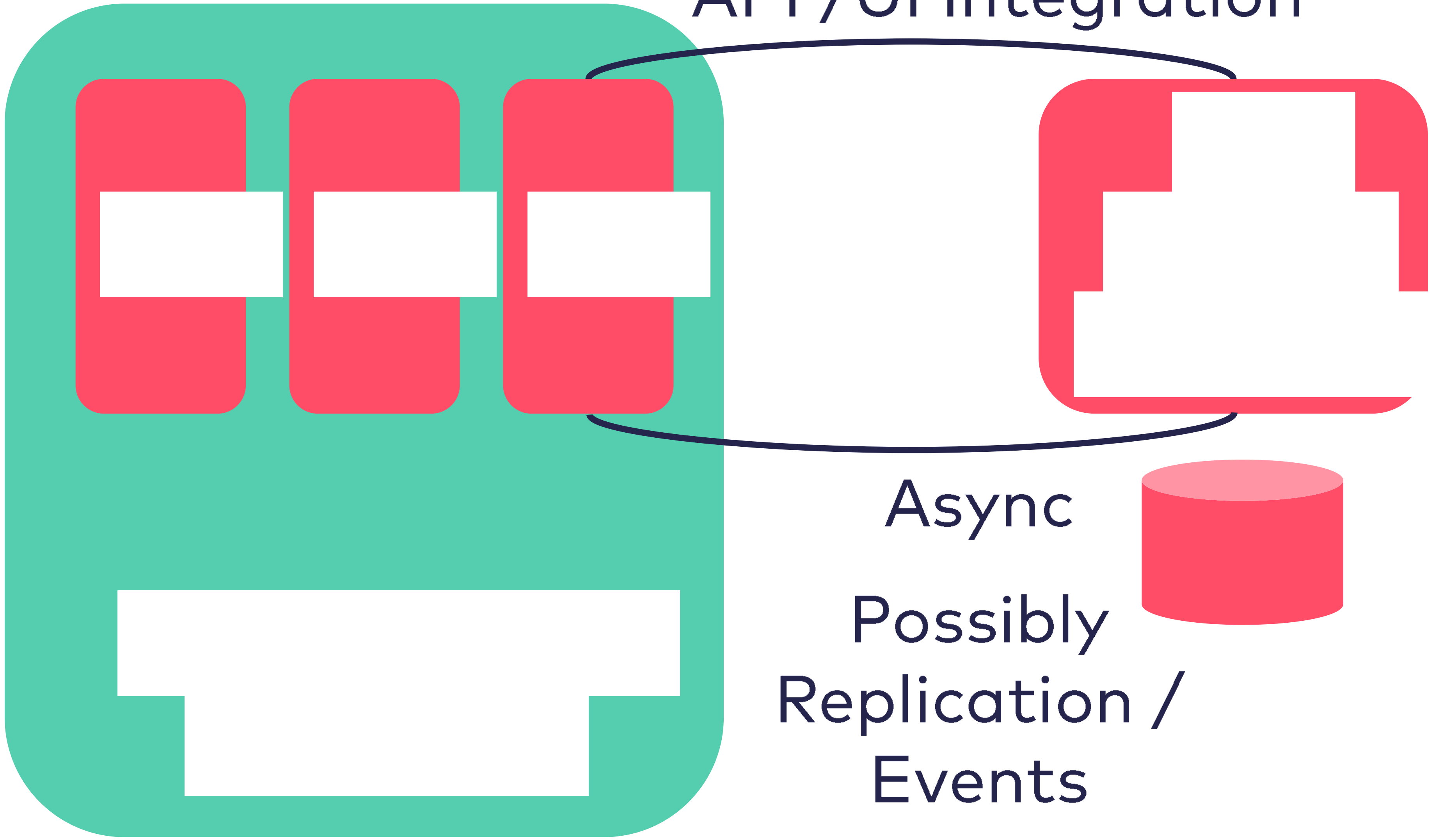
No need to understand old business logic

More support from business experts

Direct pay-off from new microservice

Adding a microservice is even better than a migration!

Integrate Microservice and Deployment Monolith



Integrate Microservice and Deployment Monolith

Asynchronous integration = decoupling

UI integration (e.g. links) provides loose coupling

API integration: Route some requests to the new system

Repeat

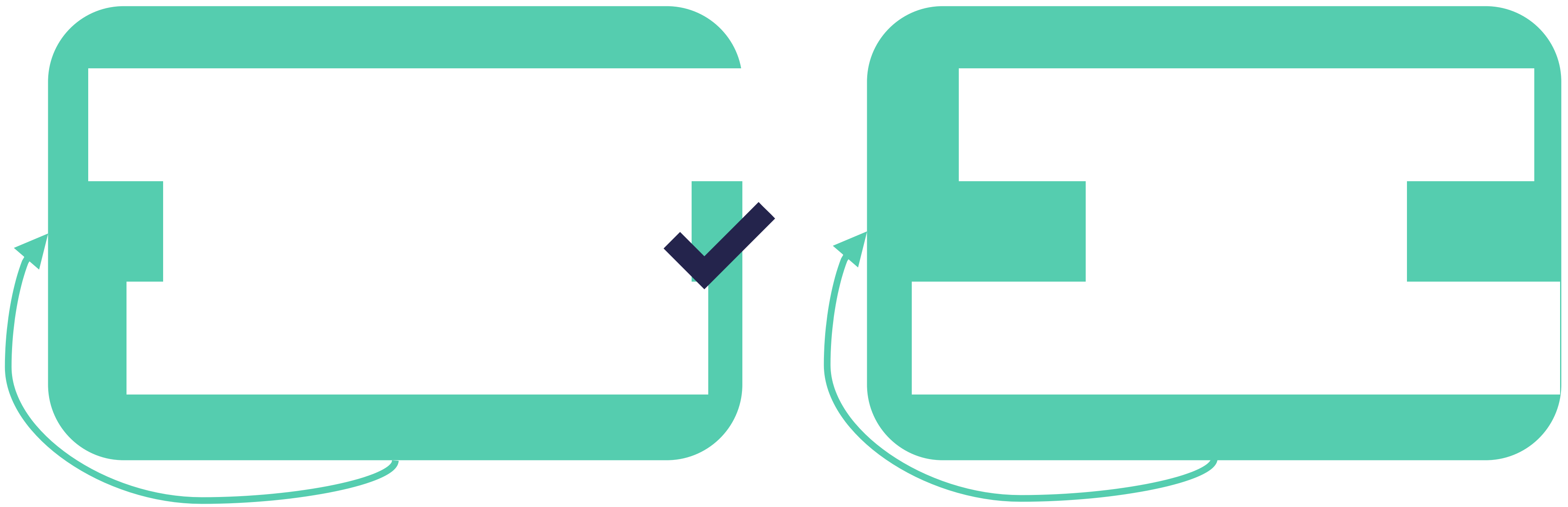
Choose the next bounded context

Note: Migration might never terminate

Great news!

Why migrate bounded context where risk is too high and / or pay-off too low?

Blueprint Migration Approach



Build Infrastructure

Challenge: Operate a large number of microservices

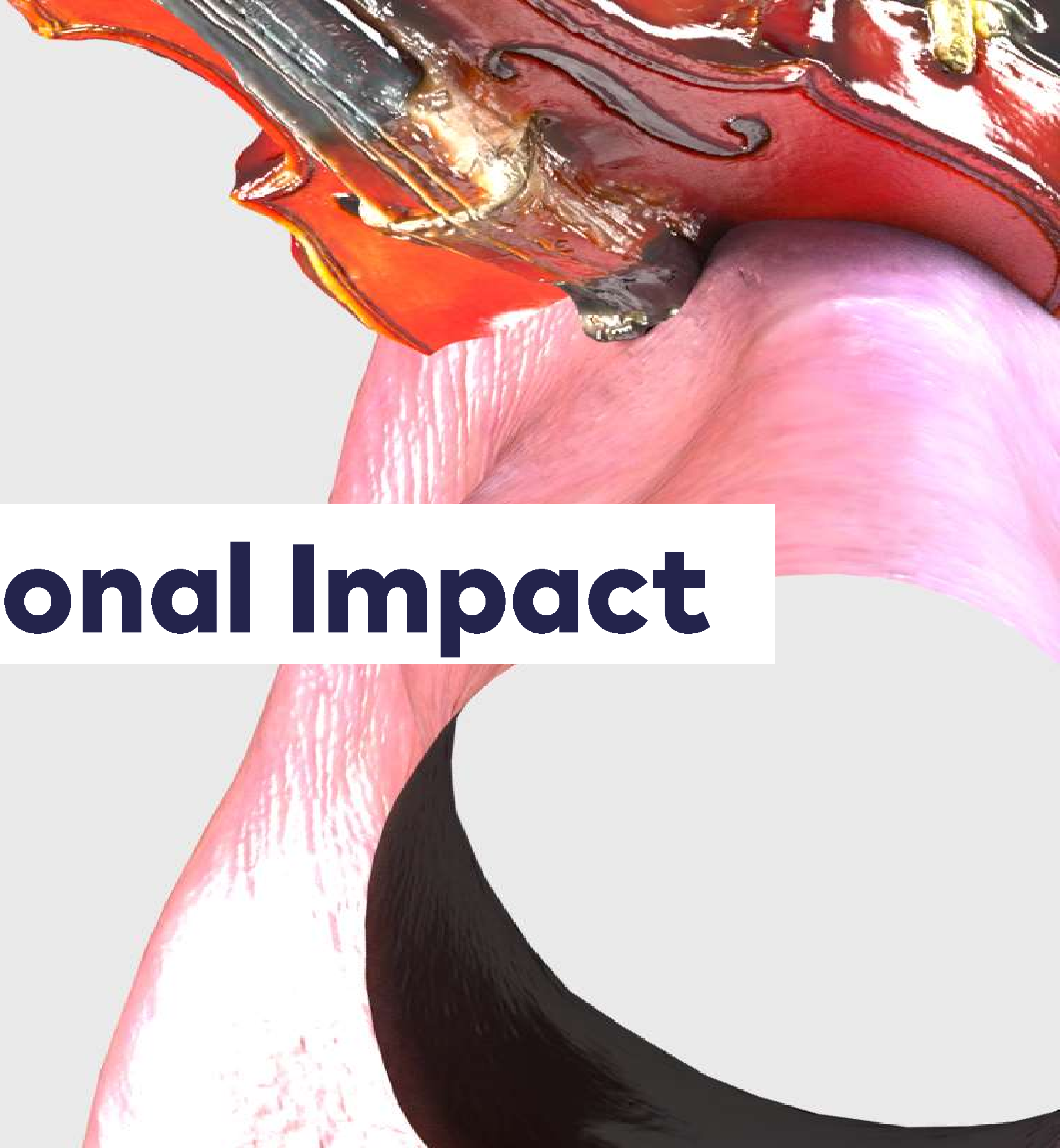
No challenge for one microservice

i.e. build infrastructure when needed

...but not later

Operating 10 or 100 is quite different from 1.

Organizational Impact



Organizational Impact

Independent microservices enable independent teams

Delegate technological responsibilities to teams!

Teams should be responsible for a part of the domain!

Choose team member who want to support the migration!

Other Strategies

Other Strategies

Fit Organization:

Compromise architecture to keep organization

Change by Extension:

New code only in microservices

Strangler:

generic

Other Strategies

Fit Organization:

Compromise architecture to keep organization

Change by Extension:

New code only in microservices

Strangler:

generic

No Bounded Context

Other Strategies

More:

https://speakerdeck.com/
ewolff/
monolith-to-microservices-a-comparison-of-strategies

Conclusion

Consider goals!

Blueprint: bounded contexts, infrastructure parallel

Consider just adding new microservices

Blueprint does not always fit

Consider organization, too

Migration might not terminate...

Why Microservices Fail: An Experience Report

Eberhard Wolff
Fellow
INNOQ
@ewolff
http://ewolff.com

Common Data Model

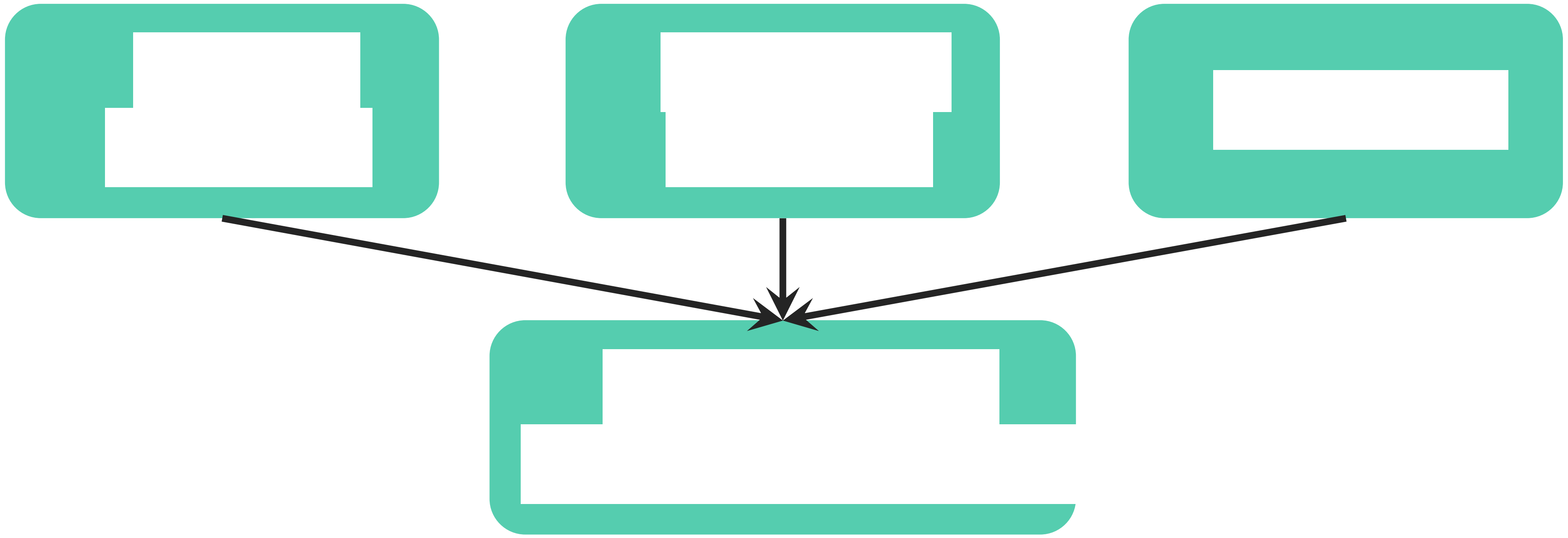
"The services need some common data."

Common Data Model: Communication

Common data model for communication only

Data model = common library

All services must use latest version of library

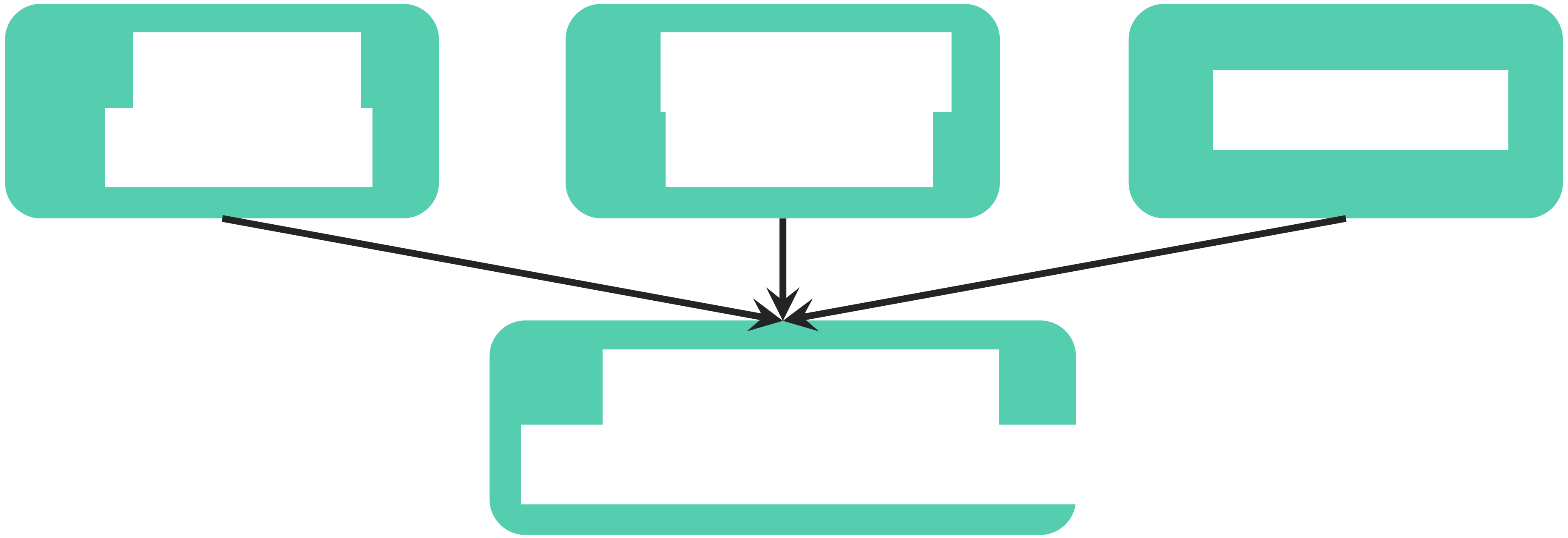


Common Data Model: Communication

Change -> redeploy all services

No decoupled deployment

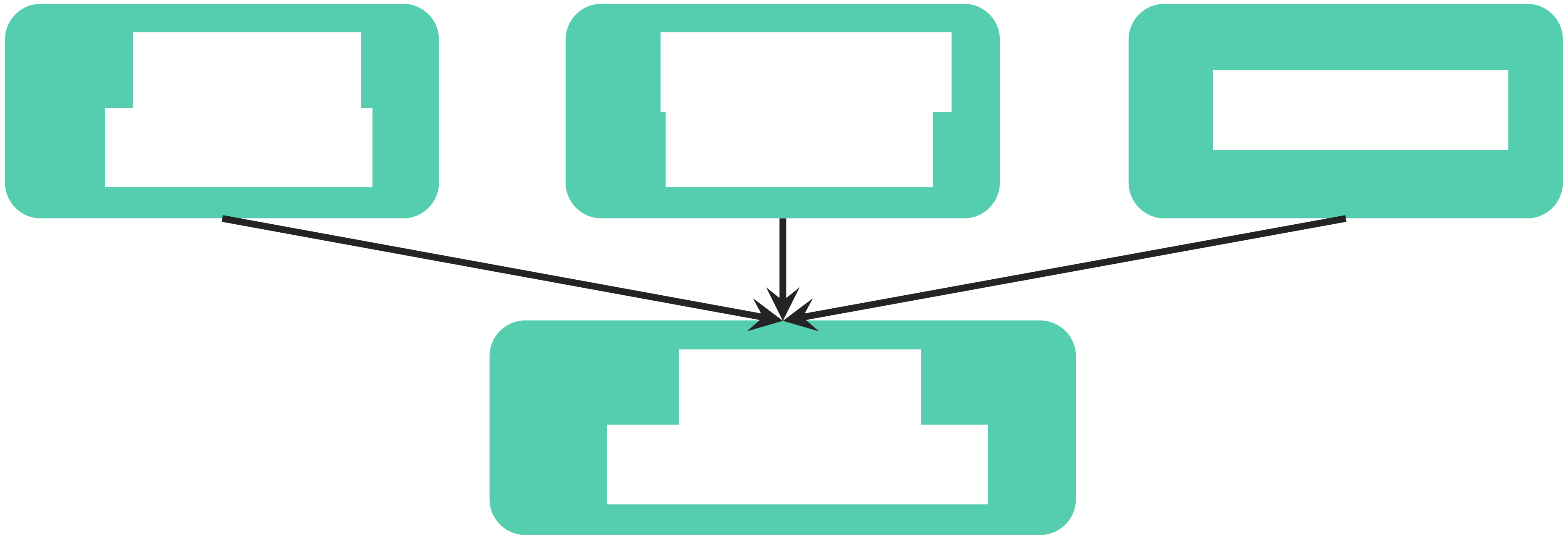
Deployment monolith with microservices challenges



Common Data Model: Events

Data model = events stored e.g. in Kafka

Rebuild local state from events



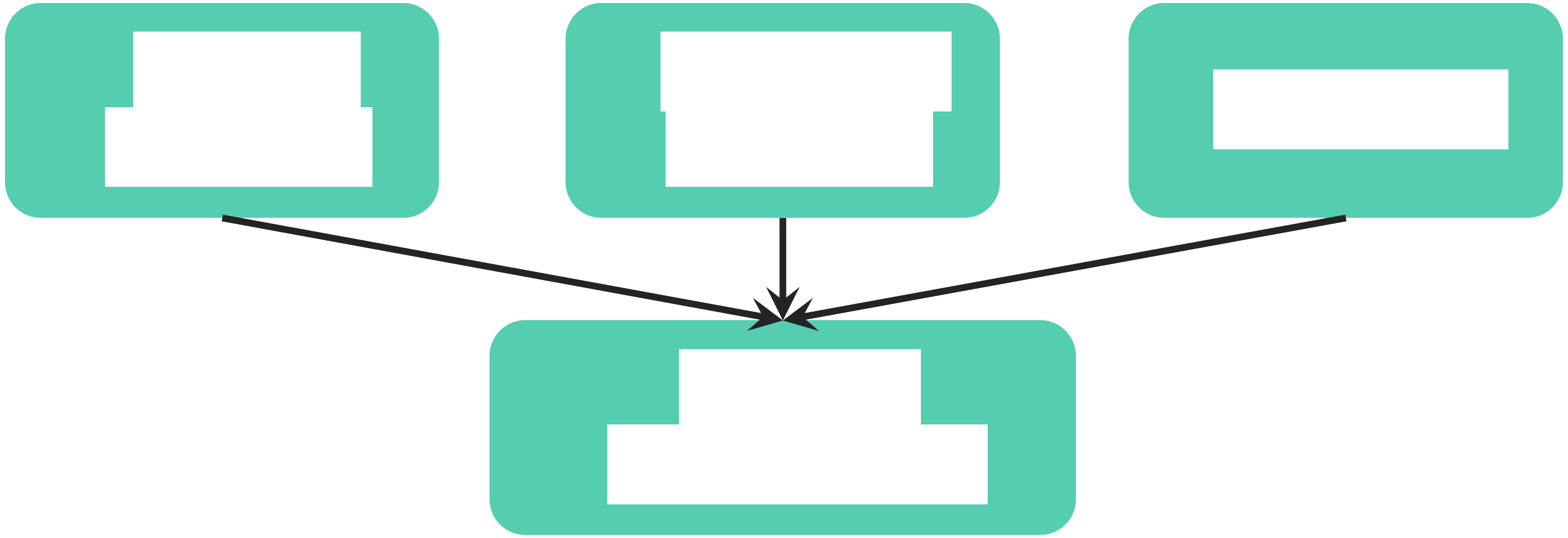
Common Data Model: Events

Many dependencies

Event data model hard to change

Particularly hard: remove an attribute

I.e. model will keep growing



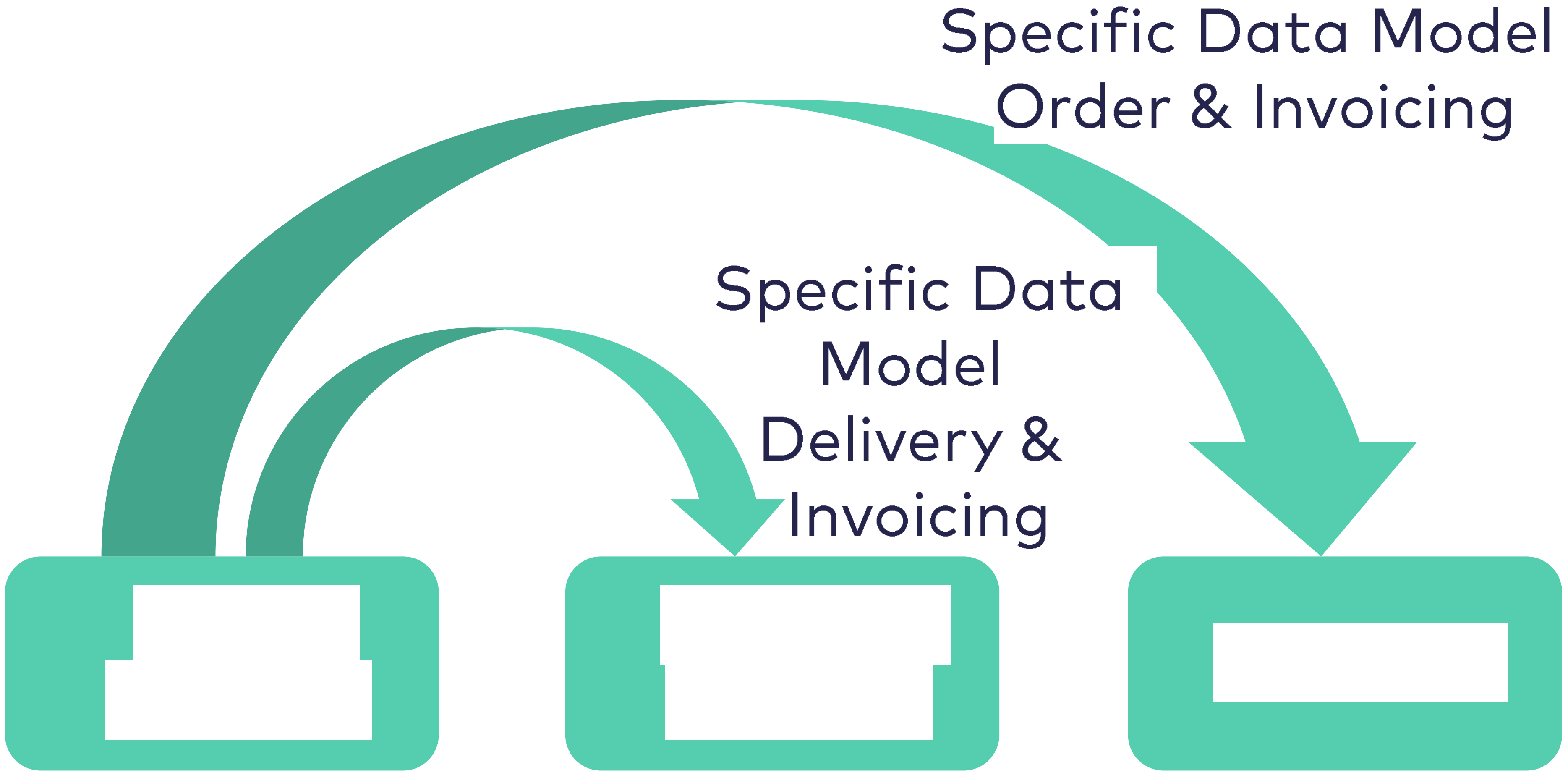
Centralized Data Model: Cure

Use separate local data models

Well-understood

Use specific data model for each interface between two microservices.

Centralized Data Model: Cure

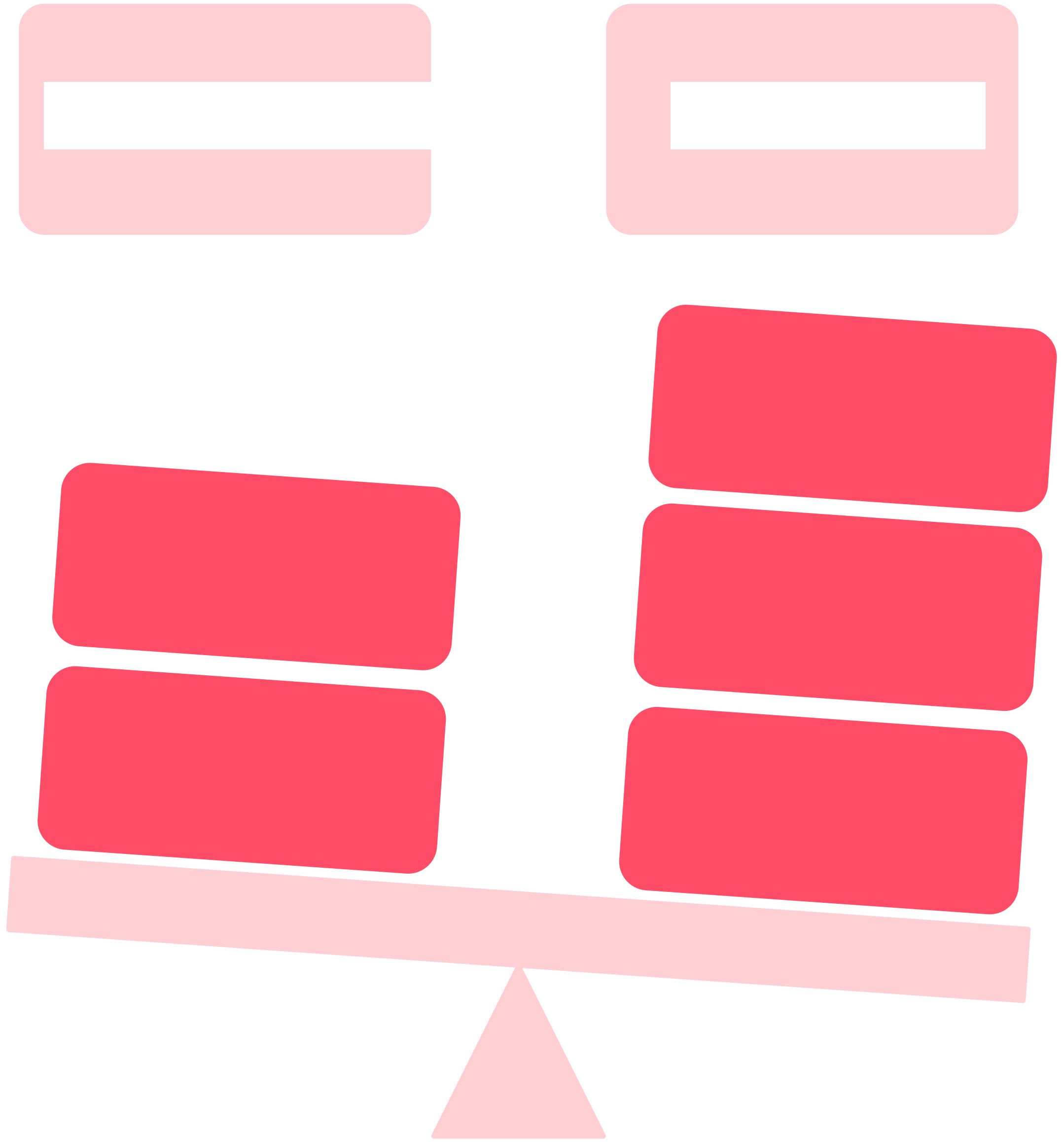


Data Model Inflation?

Independence vs. one model

Trade-off

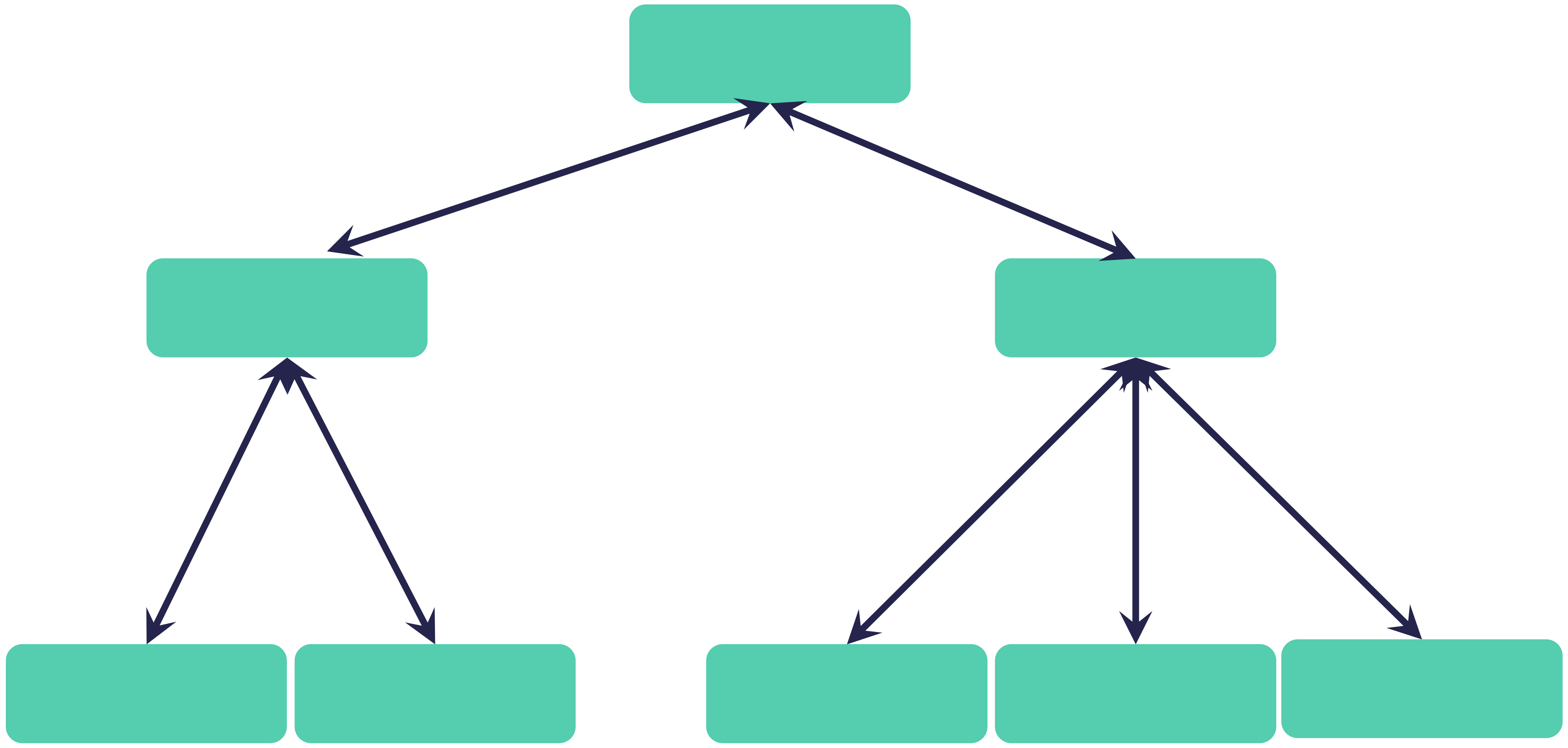
No one single best solution.



Synchronous Calls

"We do microservices the Netflix way!"

Cascading Synchronous Calls



Easy to understand

Similar to local programs

Synchronous Calls: Challenge

Latencies add up

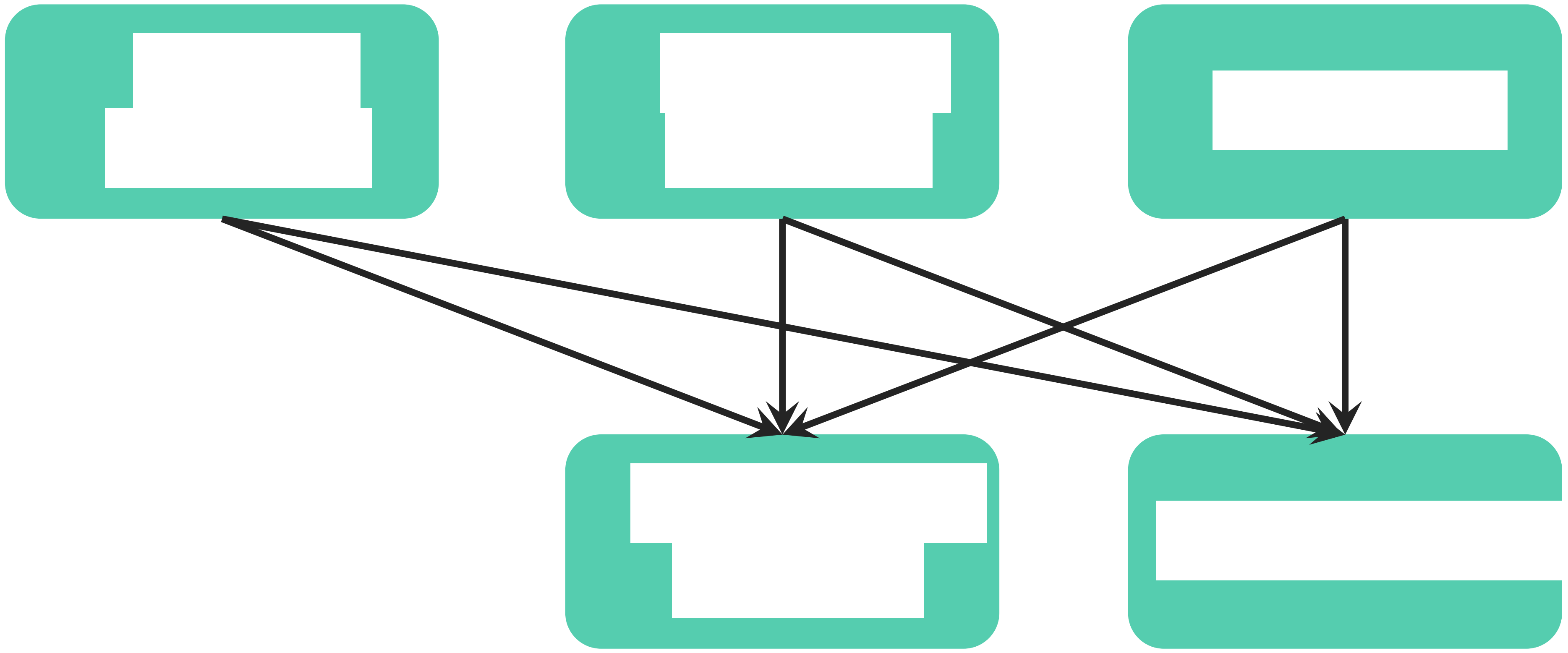
...or calls have to be in parallel

Flaky service: Hard to compensate failures

Asynchronous resilience: Messages transferred later, inconsistencies

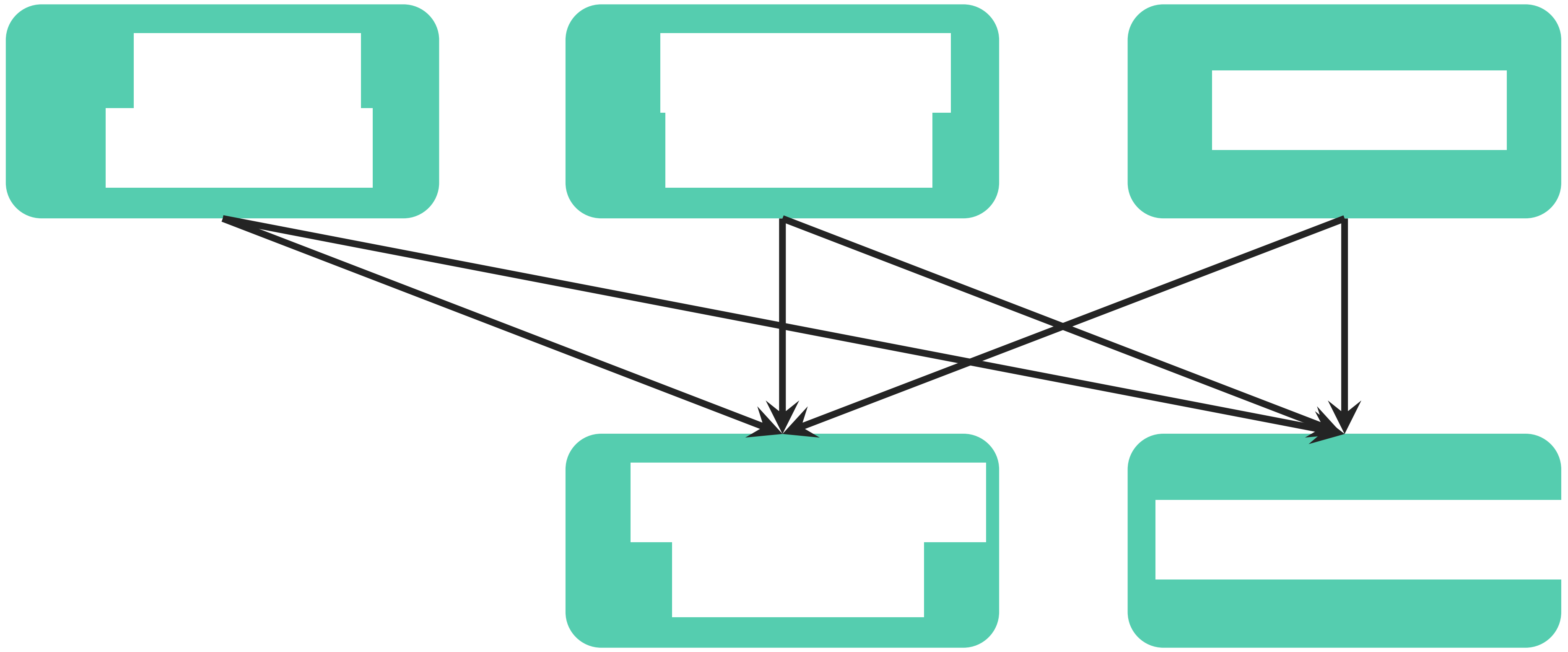
Performance issues due to network traffic

Entity Service



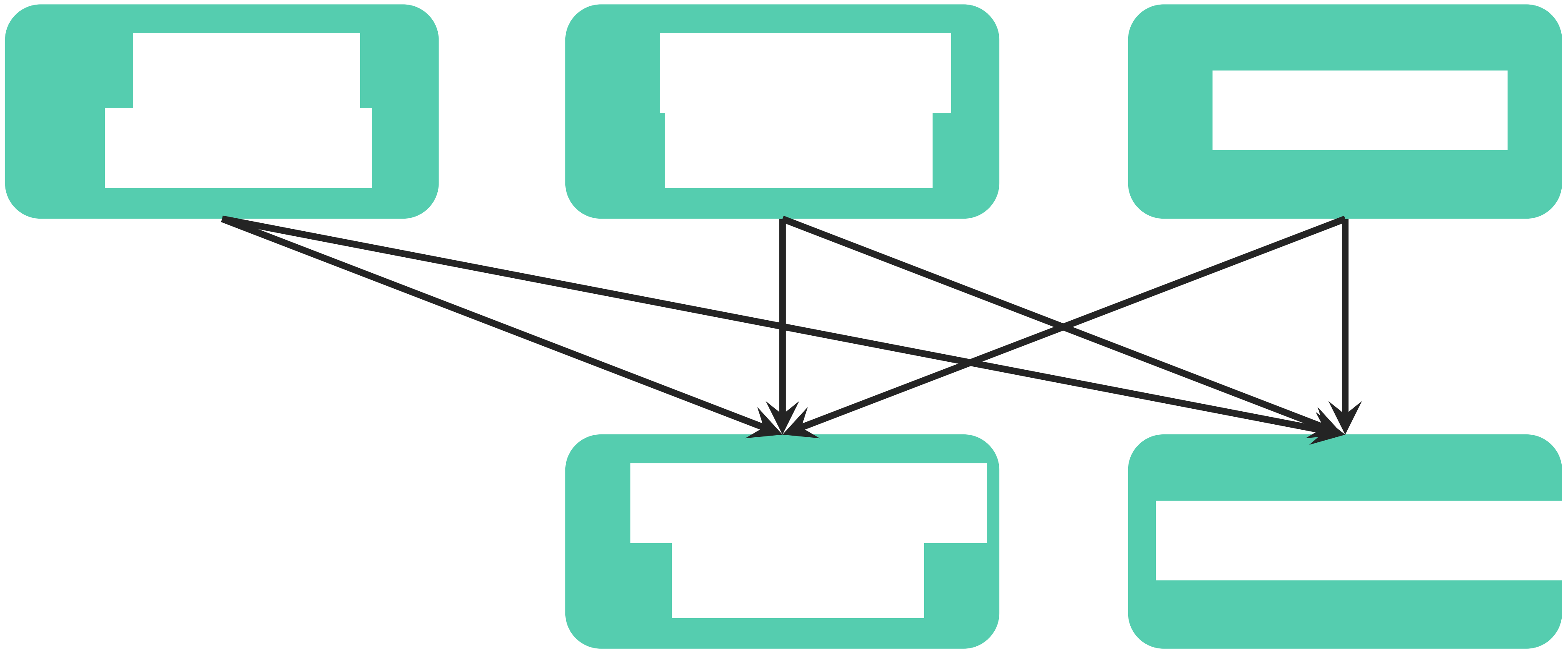
Entity Service

Can easily become a centralized data model



Entity Service

Synchronous calls

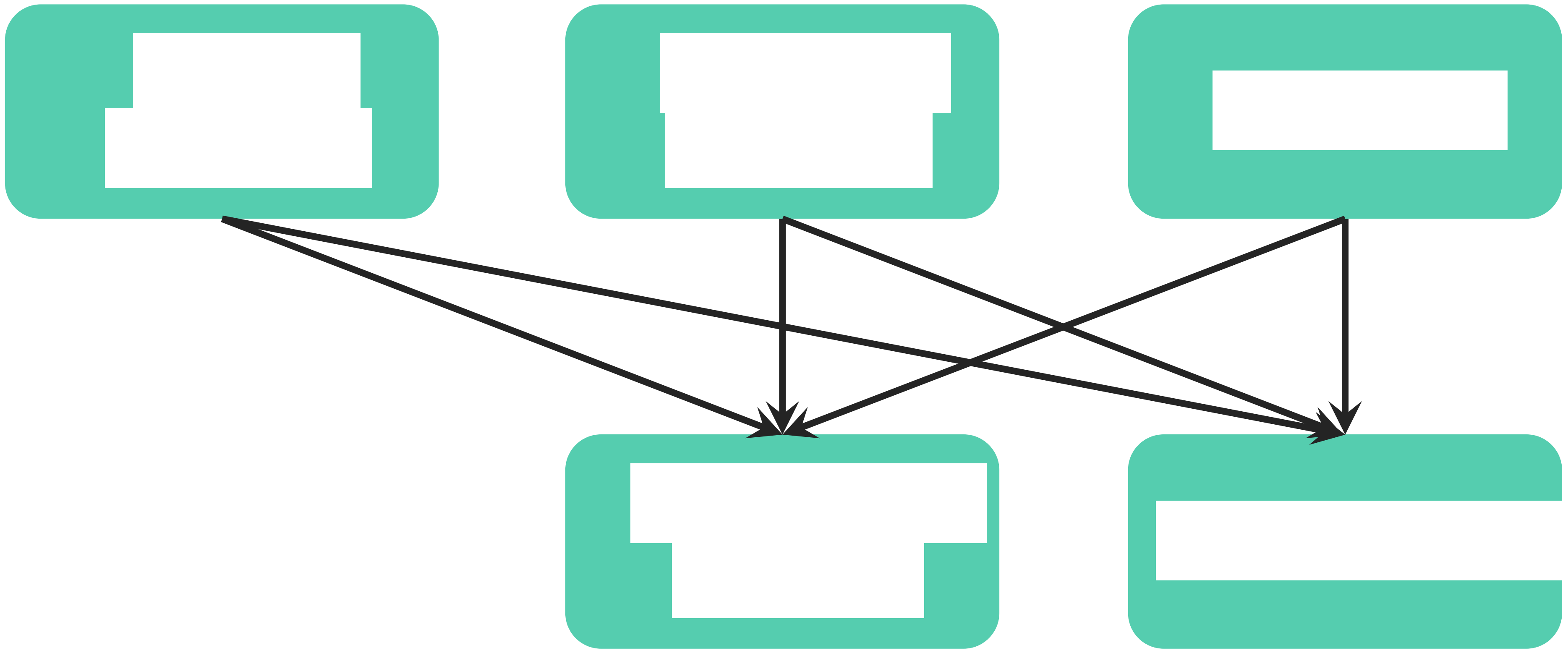


Entity Service

Every call goes through three services.

Performance

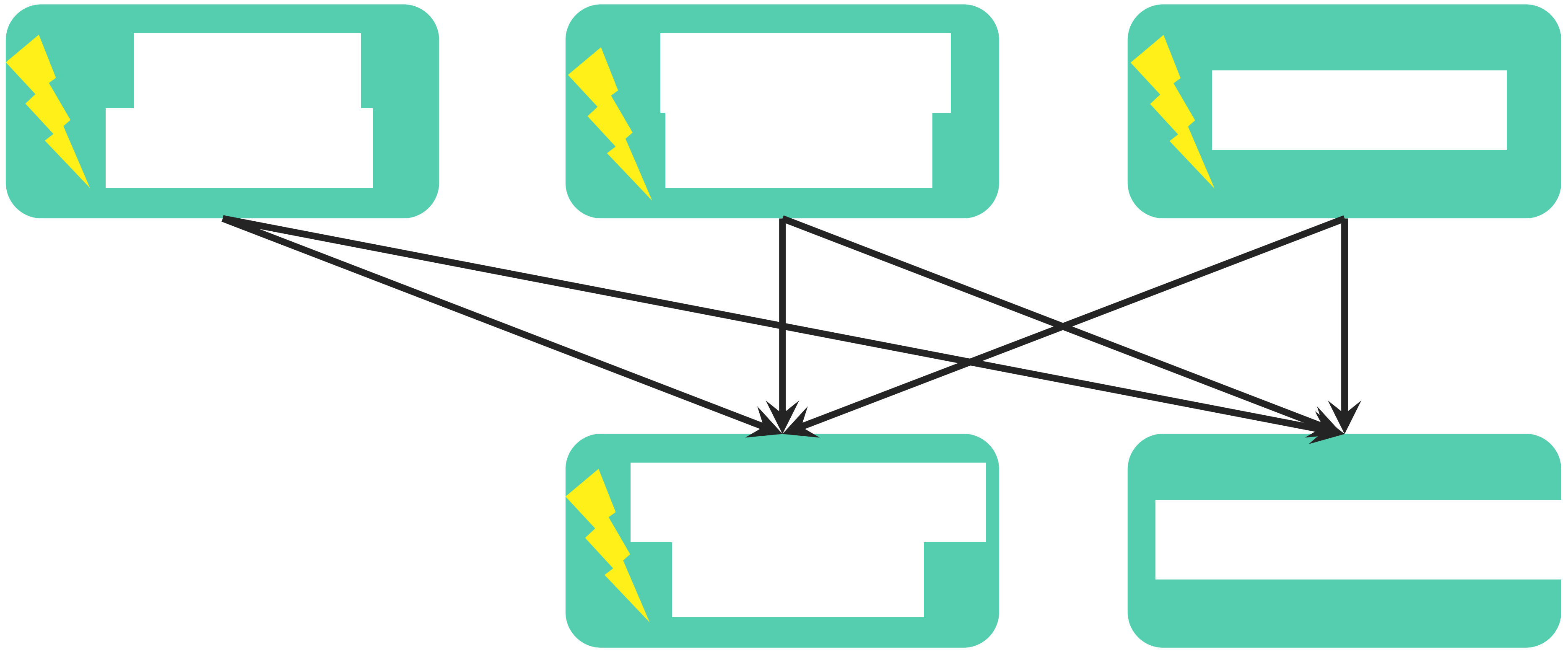
Latency



Entity Service

Failure can easily propagate.

Flaky services

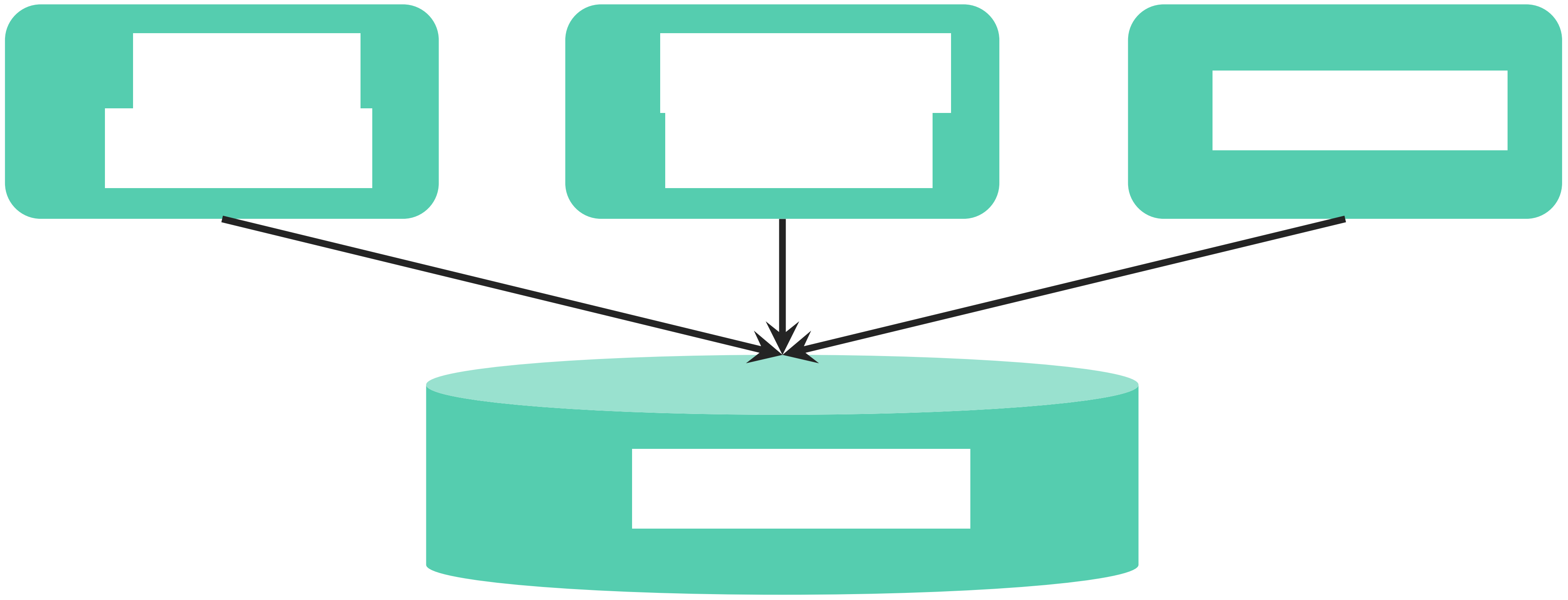


Common Database

Might be a centralized data model

Performance / latency not an issue

Shouldn't be flaky.



Entity Service: Cure

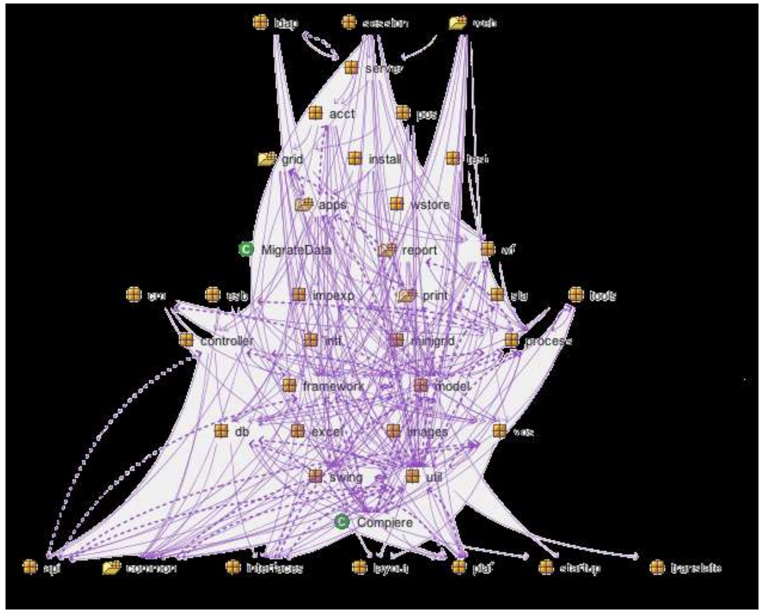
Each microservice should have its own data model

= Domain-driven Design's Bounded Context

Might share a database…

…but with separate schemas

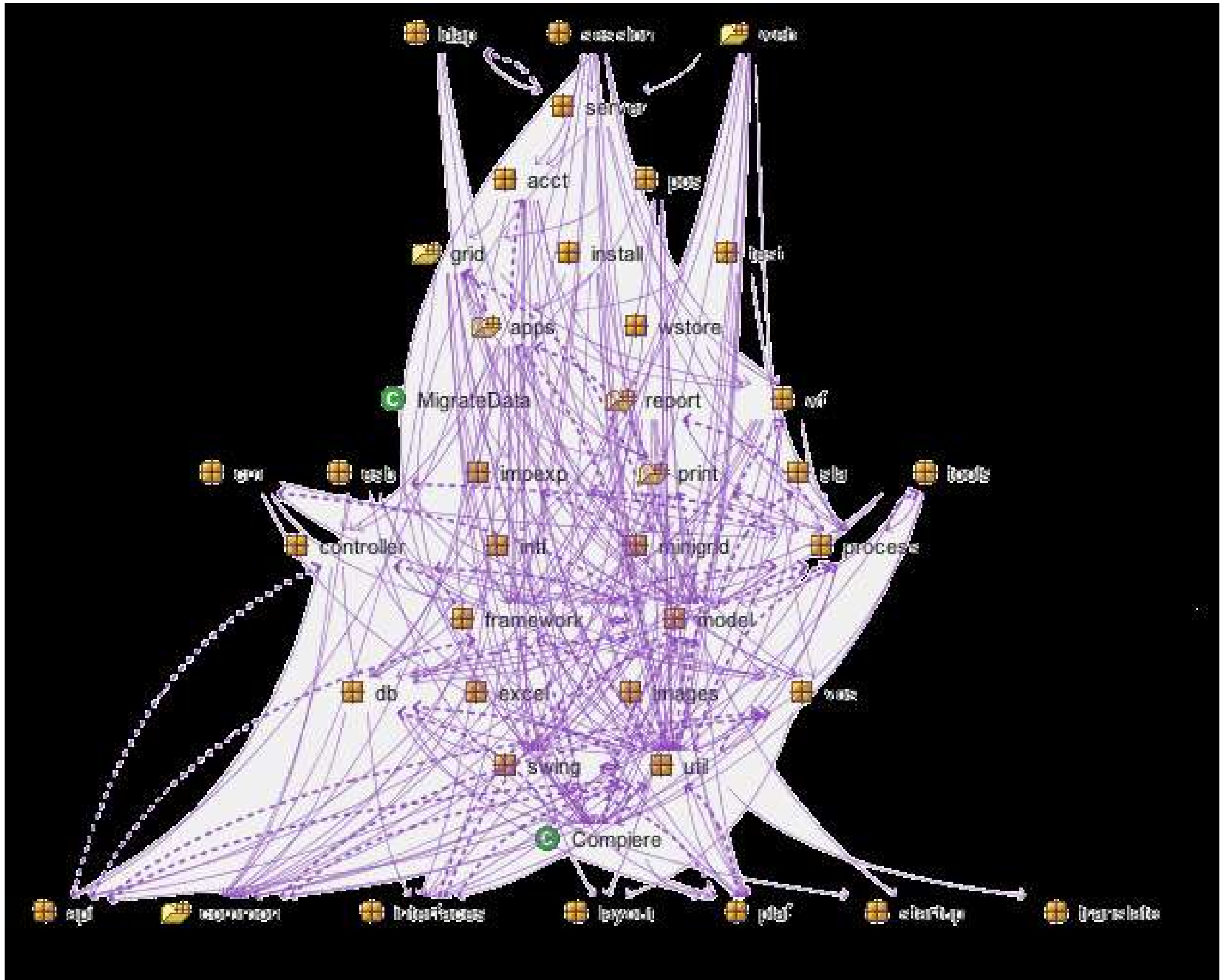
Shared database might make services flaky.



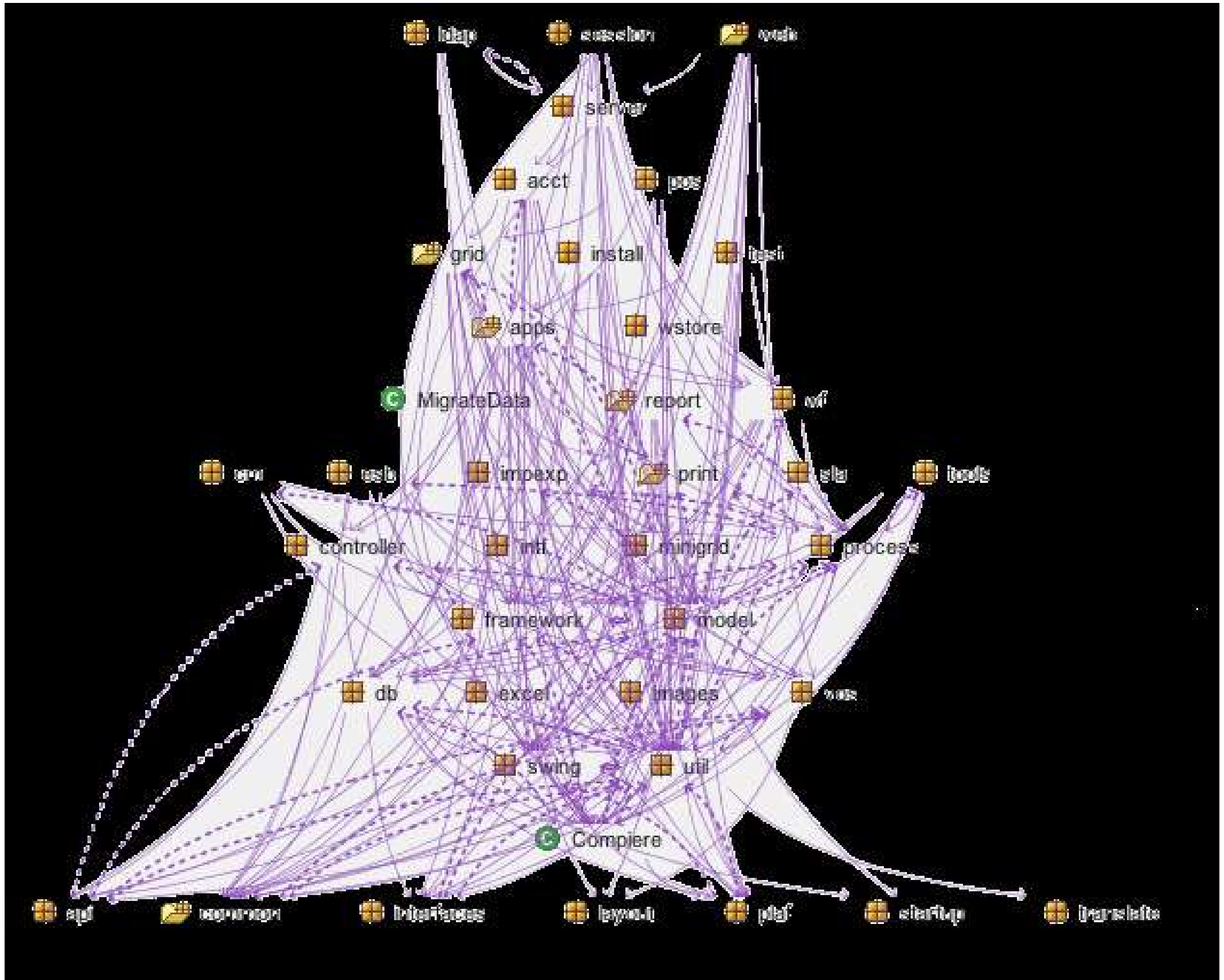
Bad Structure

"The system is flexible and maintainable – because we use microservices!"

Bad Structure: Deployment Monolith



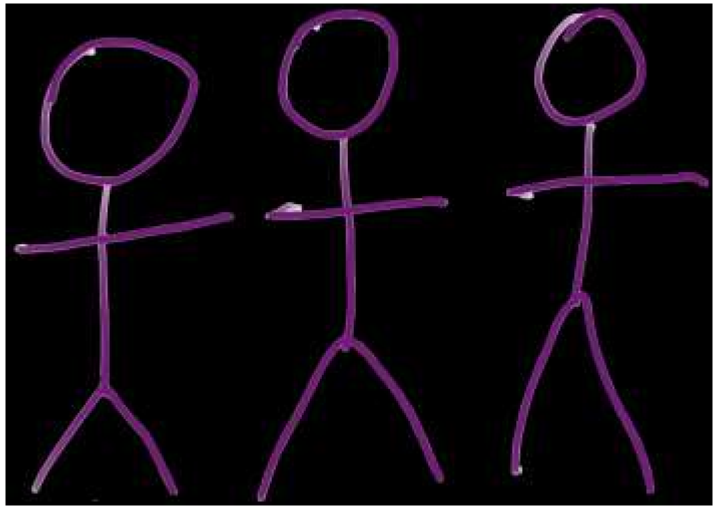
Bad Structure: Microservices



If you want to fix the structure,

If you want to fix the structure, microservices won't help.

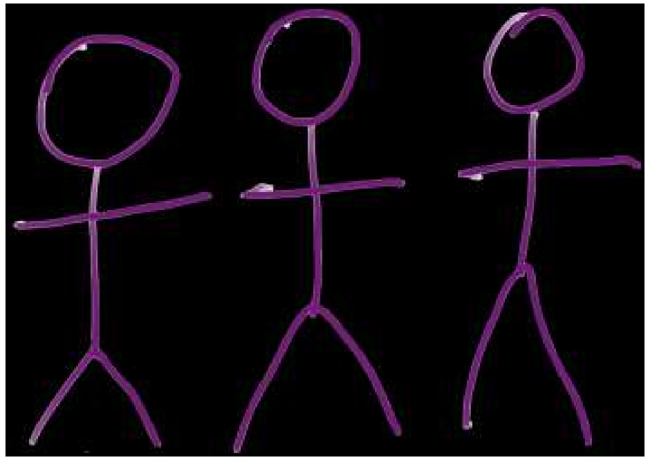
If you want to fix the structure, fix the structure.



Organization

"Architects will decide.
The teams are just not up to the challenge. ☹"

Organization: Challenge



Microservices enable

independent teams

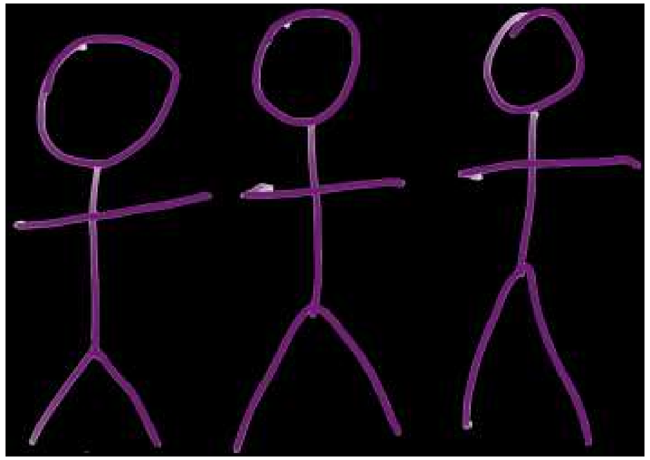
...independent technologies

...independent parts of the domain

Centralized decisions = no independent teams

Reduces the benefit of microservices

Organization: Cure



Leap of faith:

Empower teams

If you actually trust people, they behave differently.

Dev will work different if code goes to prod and not QA...

Conclusion

The problem is not microservices.

The problem is not microservices.
The problem is the right trade-off.

See paper for more challenges or https://speakerdeck.com/ewolff/why-microservices-fail

11 demos for hands-on microservices: https://ewolff.com/microservices-demos.html

Send email to
microservices-dortmund2019@ewolff.com

Slides
+ Microservices Primer DE / EN
+ Microservices Recipes DE / EN
+ Sample Microservices Book DE / EN
+ Sample Practical Microservices DE/EN
+ Sample of Continuous Delivery Book DE

Powered by Amazon Lambda
& Microservices