



A Microservices Journey

Susanne Kaiser

@suksr

EX - CTO at Just Software

@JustSocialApps



Each Journey is Different

“People try to copy Netflix,
but they can only copy what they see.
They copy the results, not the process.”

Adrian Cockcroft, AWS VP Cloud Architect,
former Netflix Chief Cloud Architect

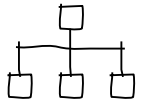


Each Journey is Different

Affecting Circumstances



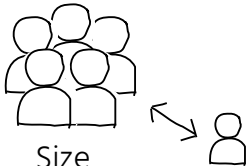
Team



Structure



Skillset



Size



Journey

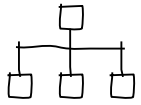


Each Journey is Different

Affecting Circumstances



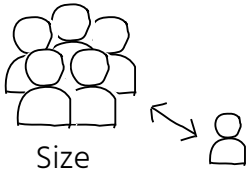
Team



Structure



Skillset



Size



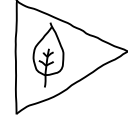
Journey



Legacy



Maintenance
effort



Runtime
environment

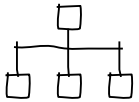


Each Journey is Different

Affecting Circumstances



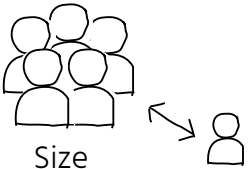
Team



Structure



Skillset



Size



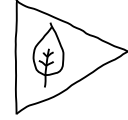
Journey



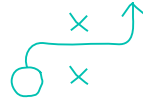
Legacy



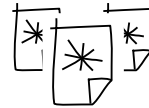
Maintenance
effort



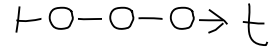
Runtime
environment



Strategy



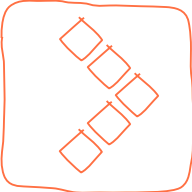
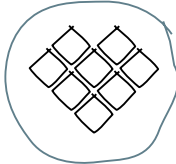
New Features



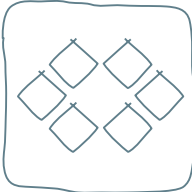
Timeline / Milestones

Background

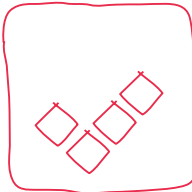
JUST SOCIAL



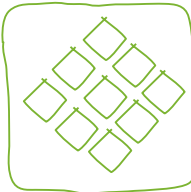
JUST DRIVE



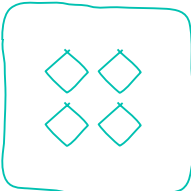
JUST CONNECT



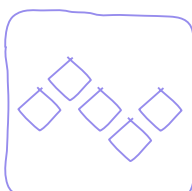
JUST LIST



JUST WIKI

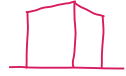


JUST PEOPLE

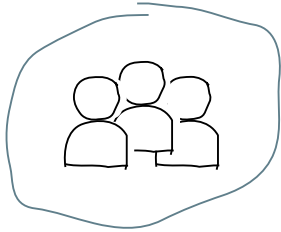


JUST NEWS

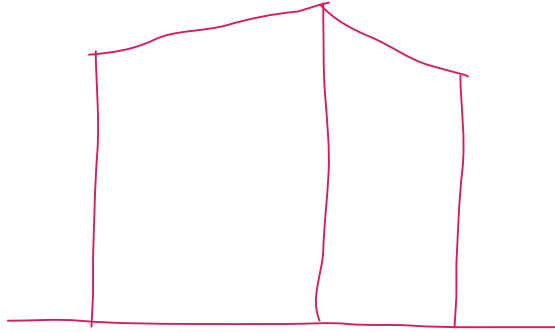
At the Beginning



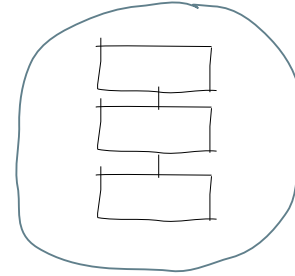
A Monolith in Every Aspect



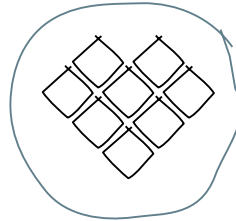
One team



Single Unit



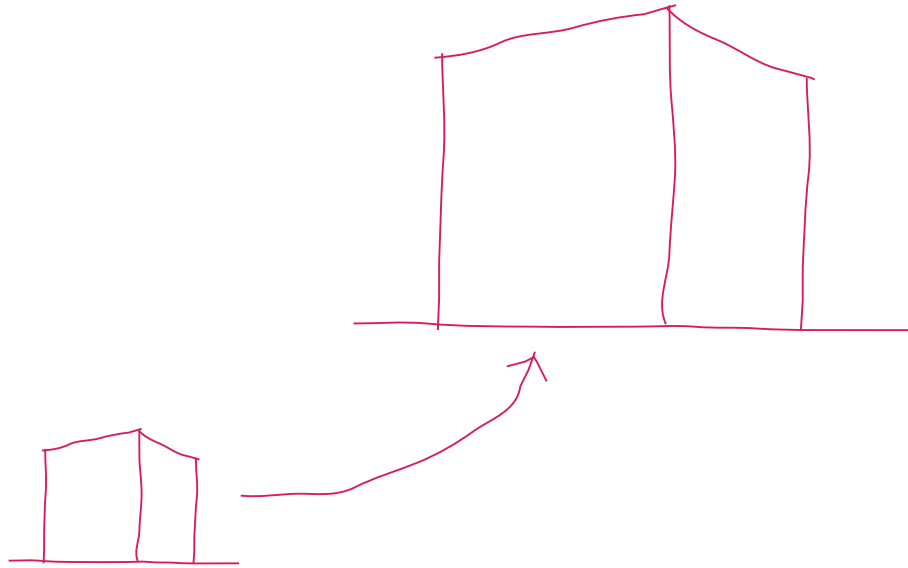
One technology stack



One collaboration product

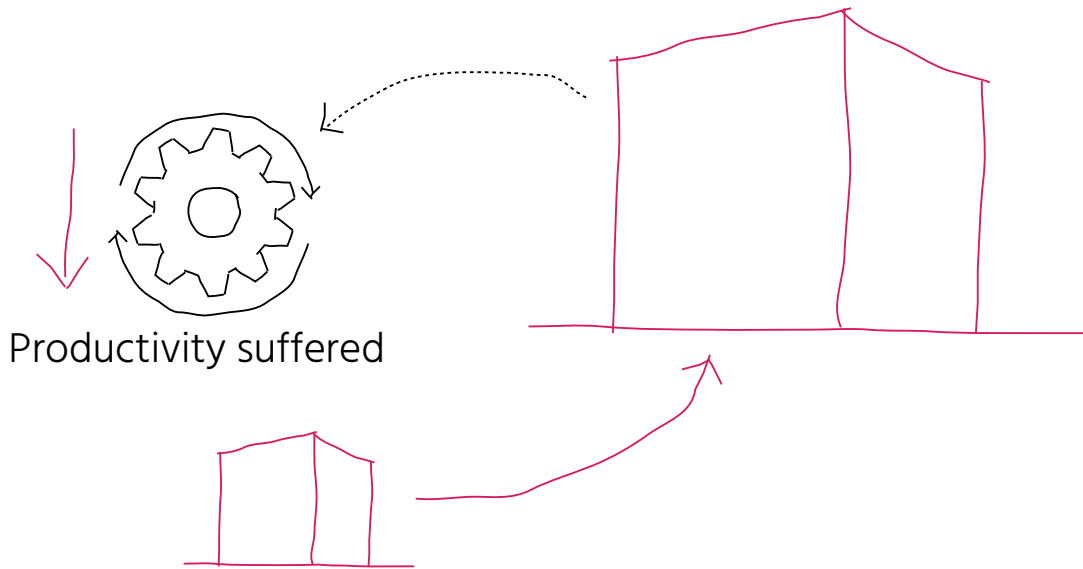
Background

$t \rightarrow$ After an Evolving Time



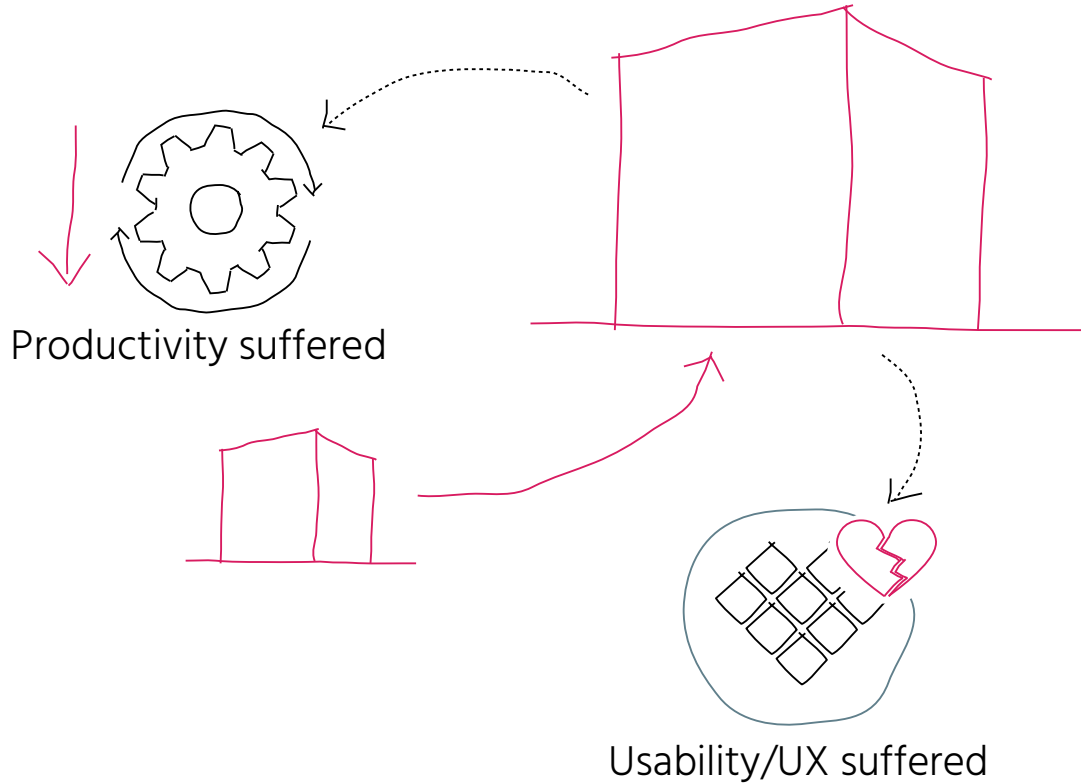
Background

→ t After an Evolving Time



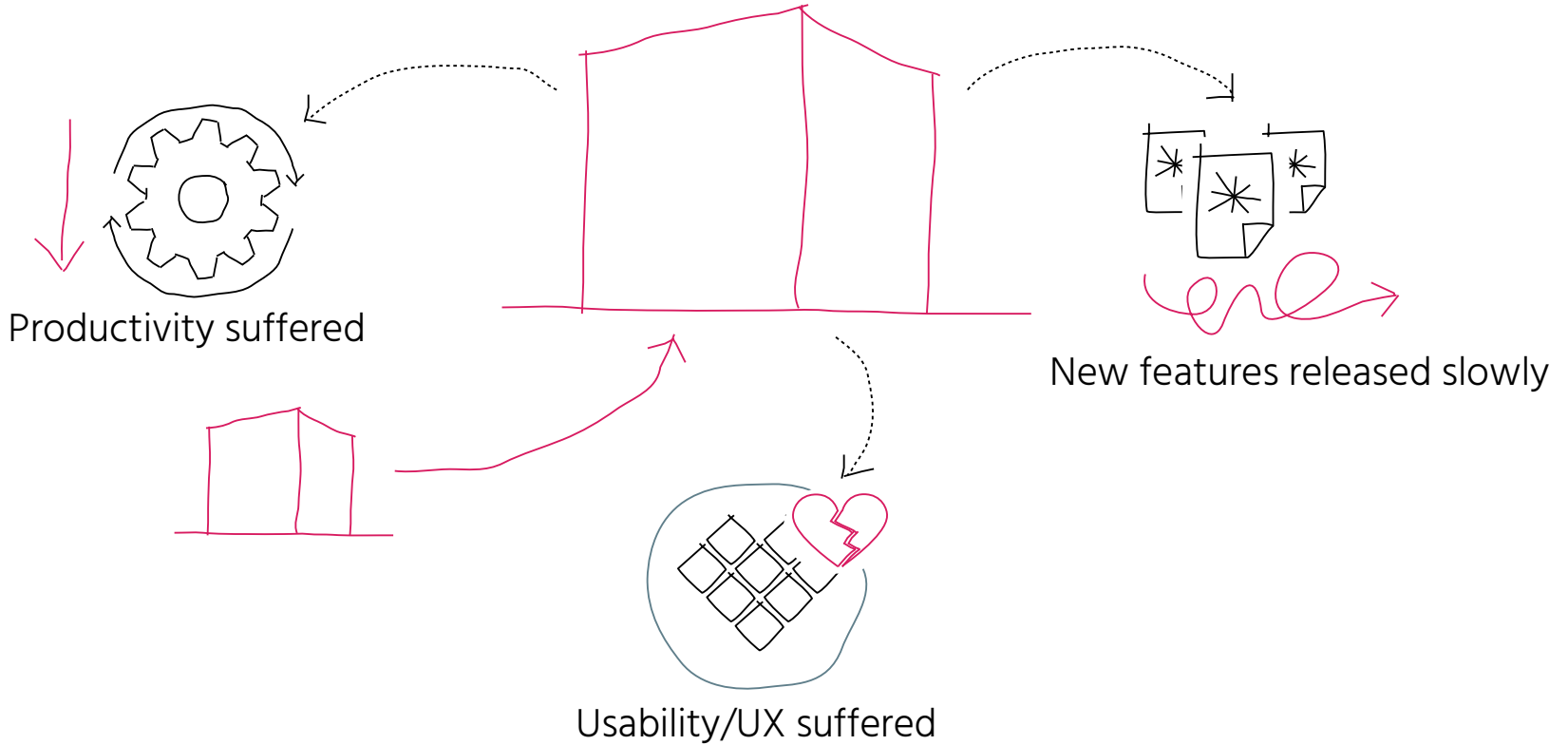
Background

→ t After an Evolving Time



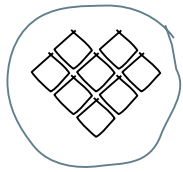
Background

→ t After an Evolving Time

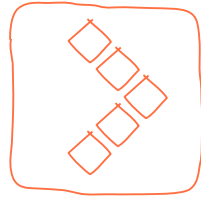


Background

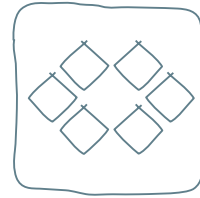
Separate Collaboration Apps



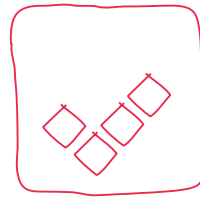
JUST SOCIAL



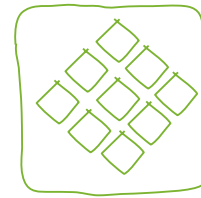
JUST DRIVE



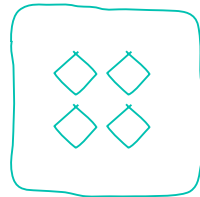
JUST CONNECT



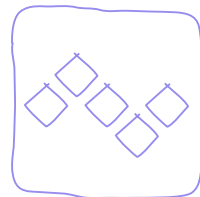
JUST LIST



JUST WIKI



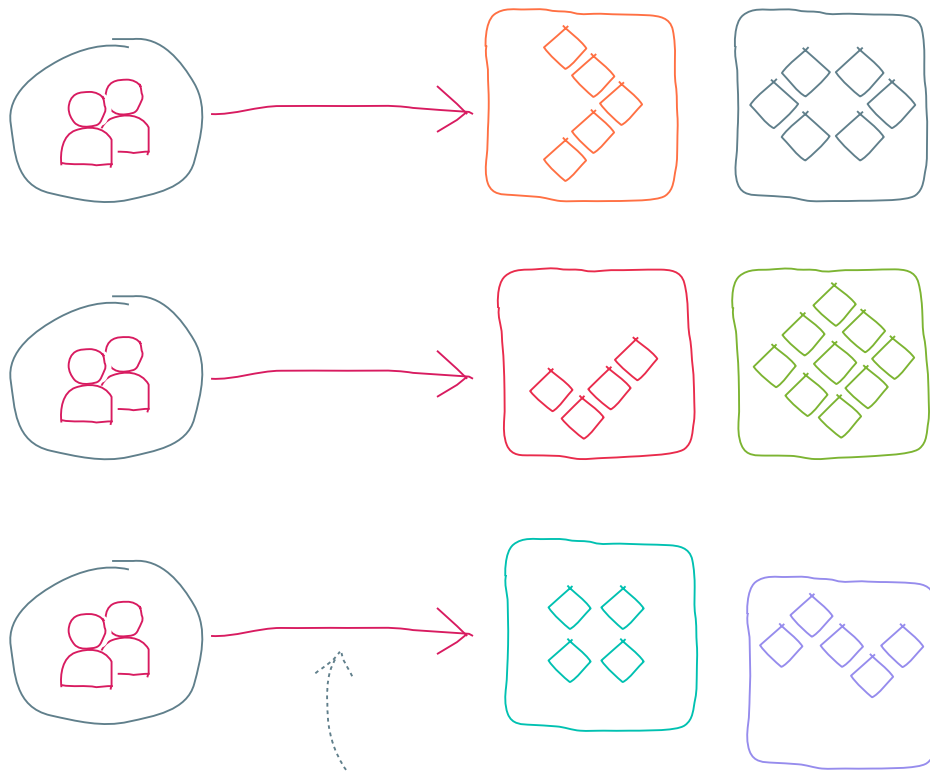
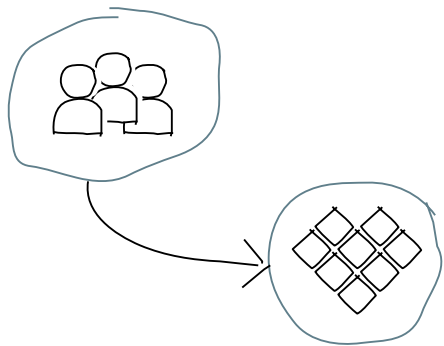
JUST PEOPLE



JUST NEWS

Background

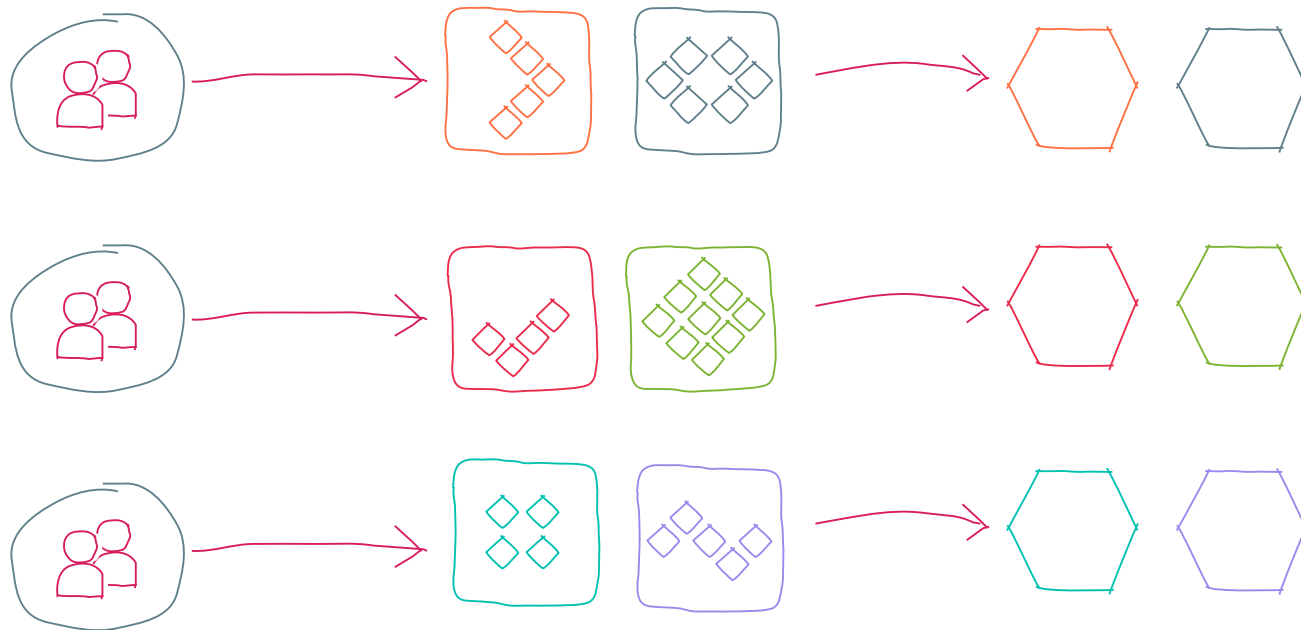
Separate, Autonomous Teams



Well-defined responsibilities

Background

In The Long Run



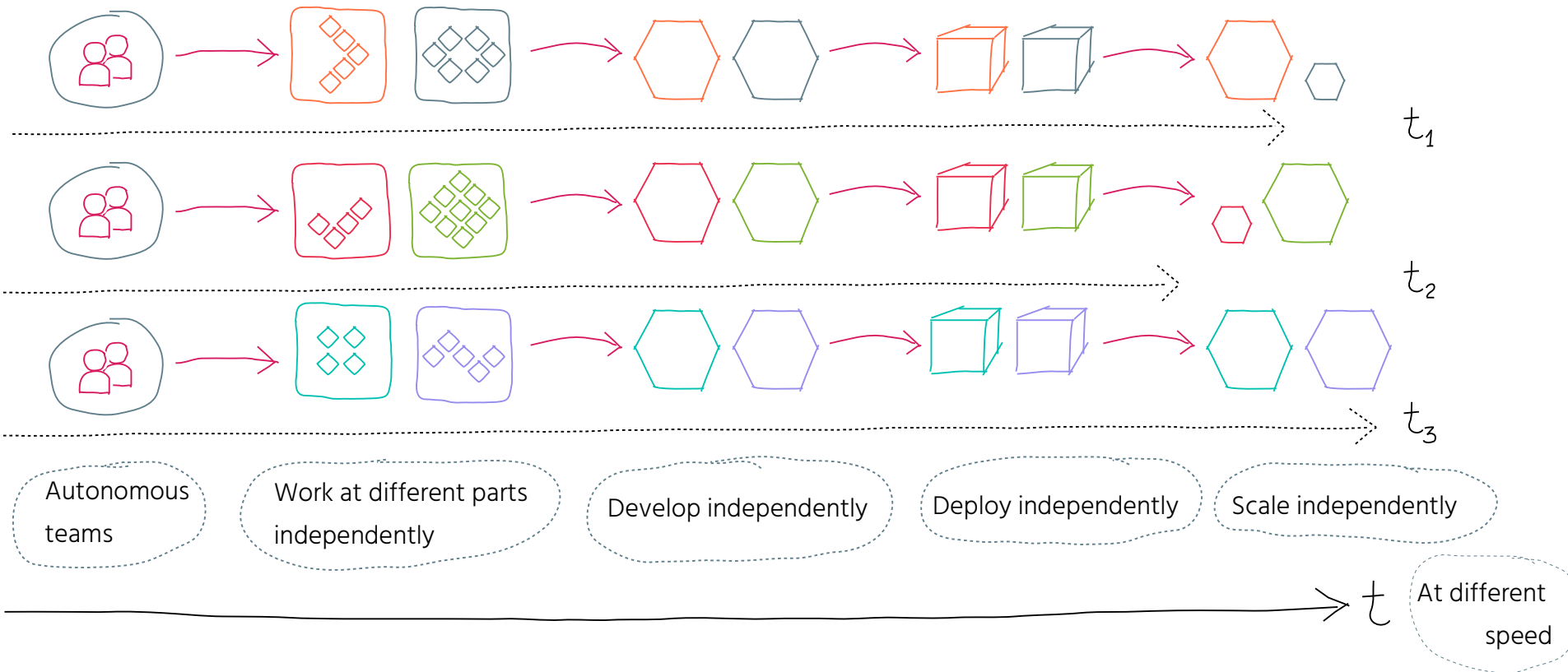
Organisation

Product

Software

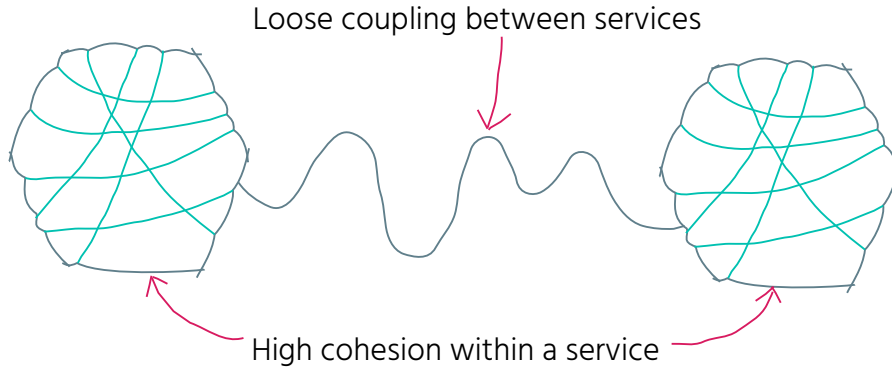
Background

Our Motivation for Microservices



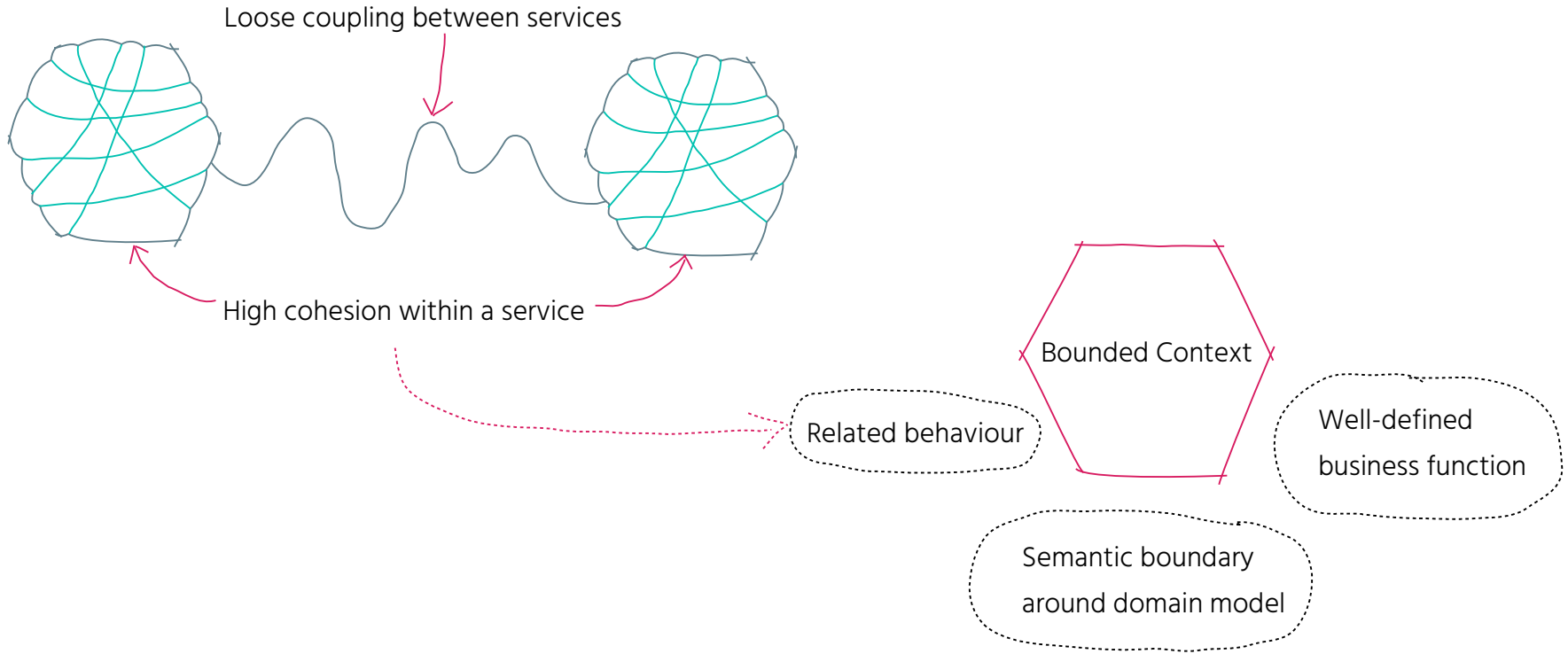
Decomposition Strategy

Identify Bounded Contexts



Decomposition Strategy

Identify Bounded Contexts

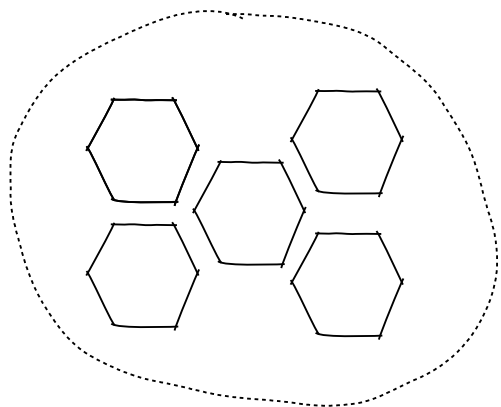


Decomposition Strategy

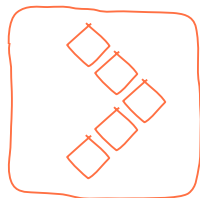


Identify Bounded Contexts

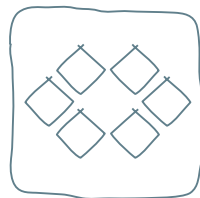
Bounded Contexts



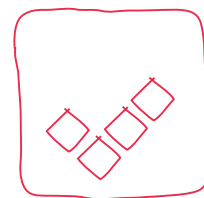
=



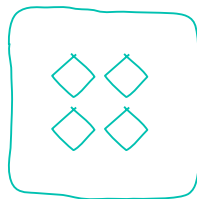
JUST DRIVE



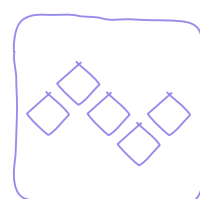
JUST CONNECT



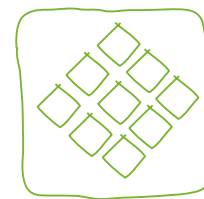
JUST LIST



JUST PEOPLE



JUST NEWS

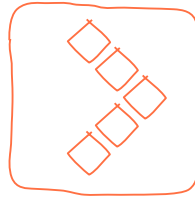


JUST WIKI

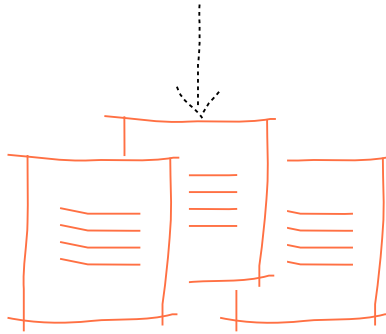
Decomposition Strategy



Co-Existing Service From Scratch

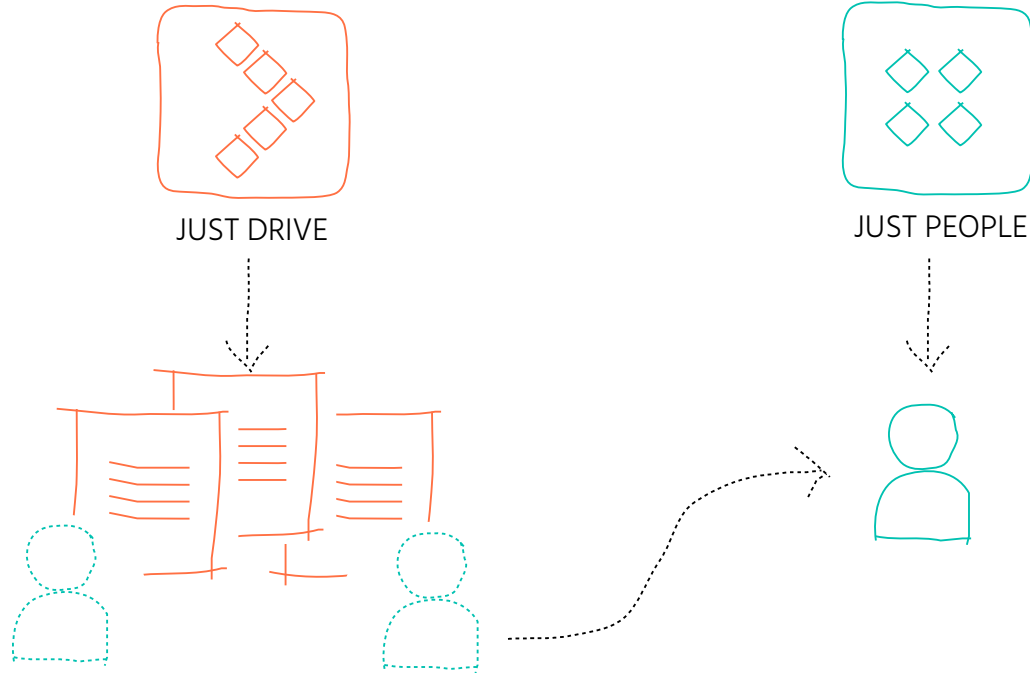


JUST DRIVE



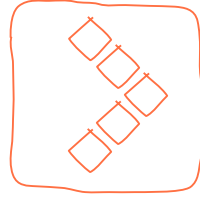
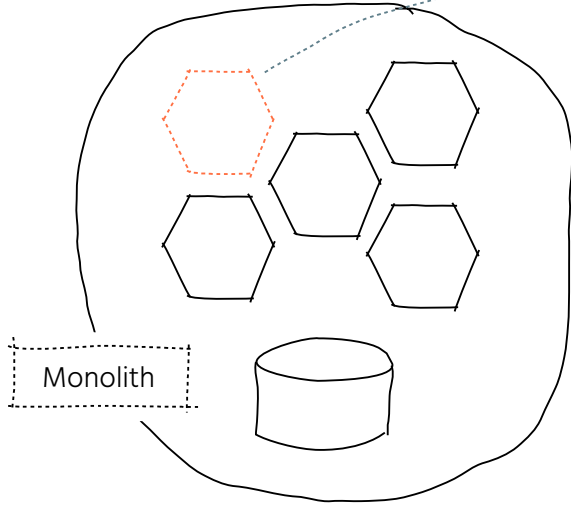
Decomposition Strategy

 Co-Existing Service From Scratch

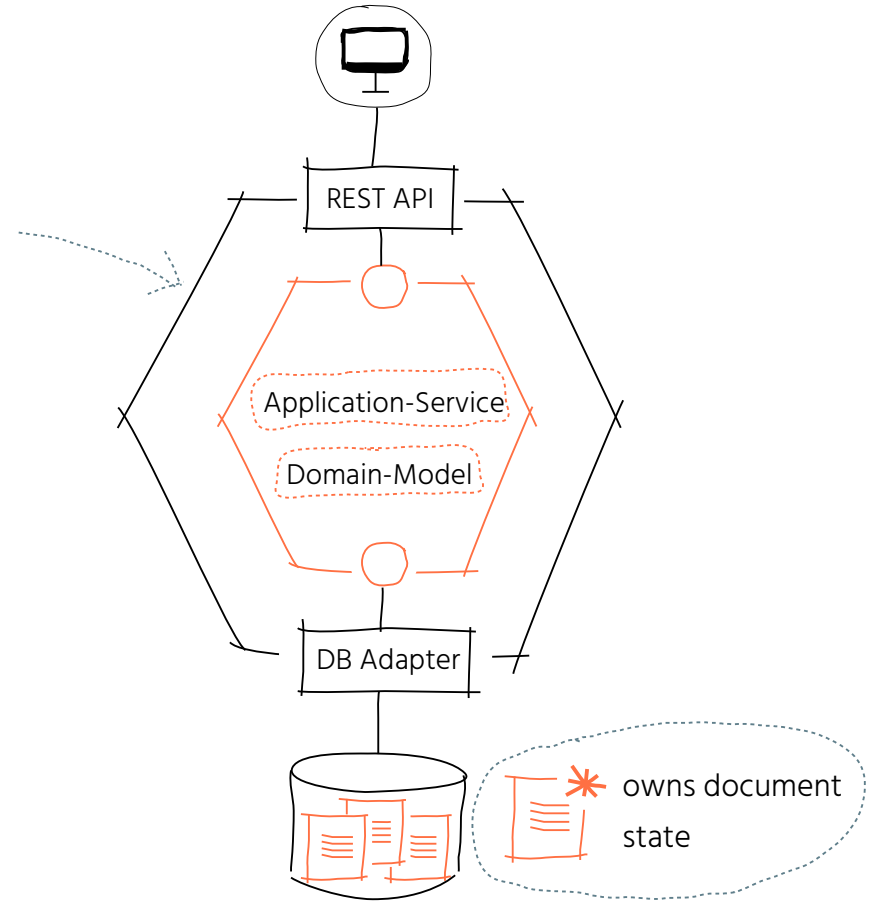


Decomposition Strategy

 Co-Existing Service From Scratch

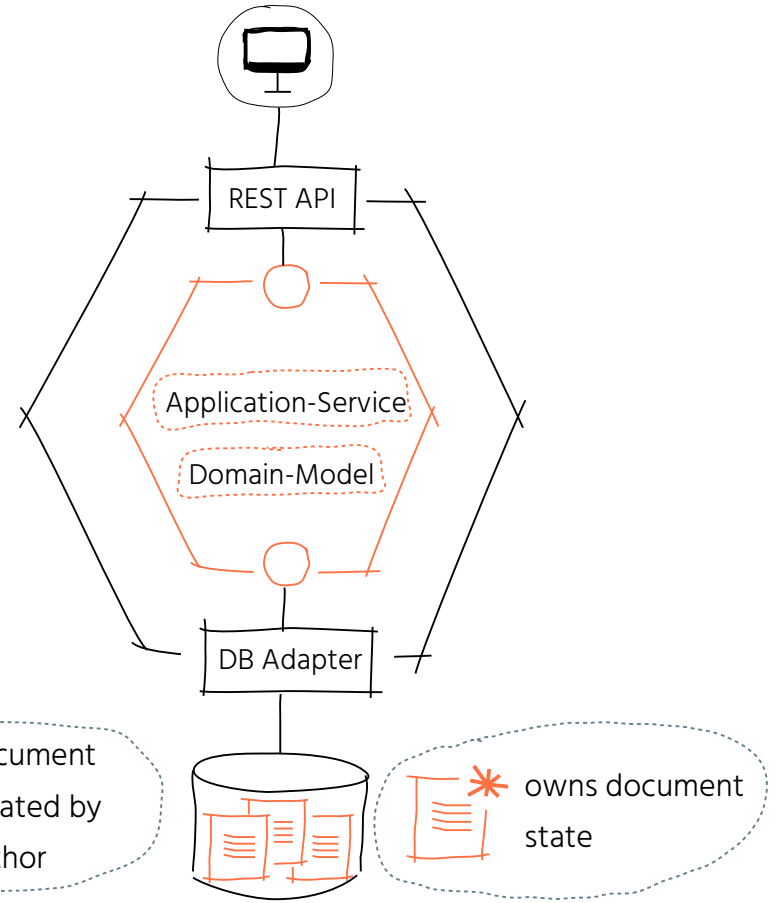
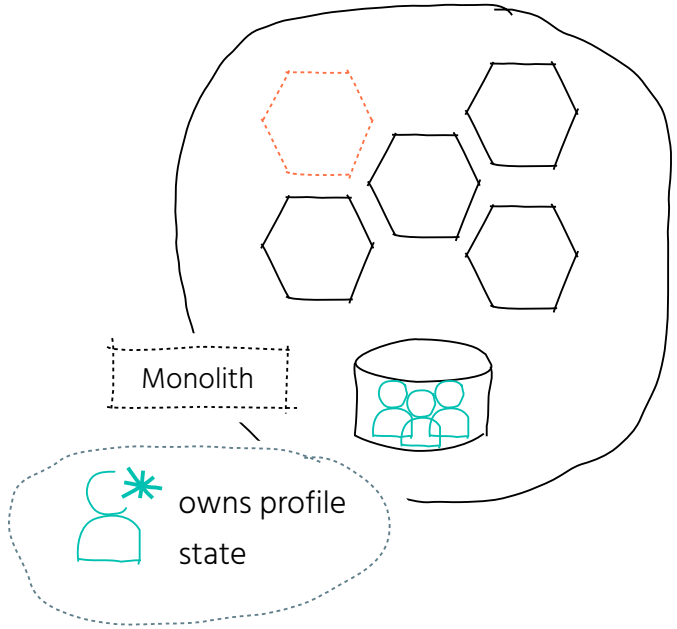


JUST DRIVE



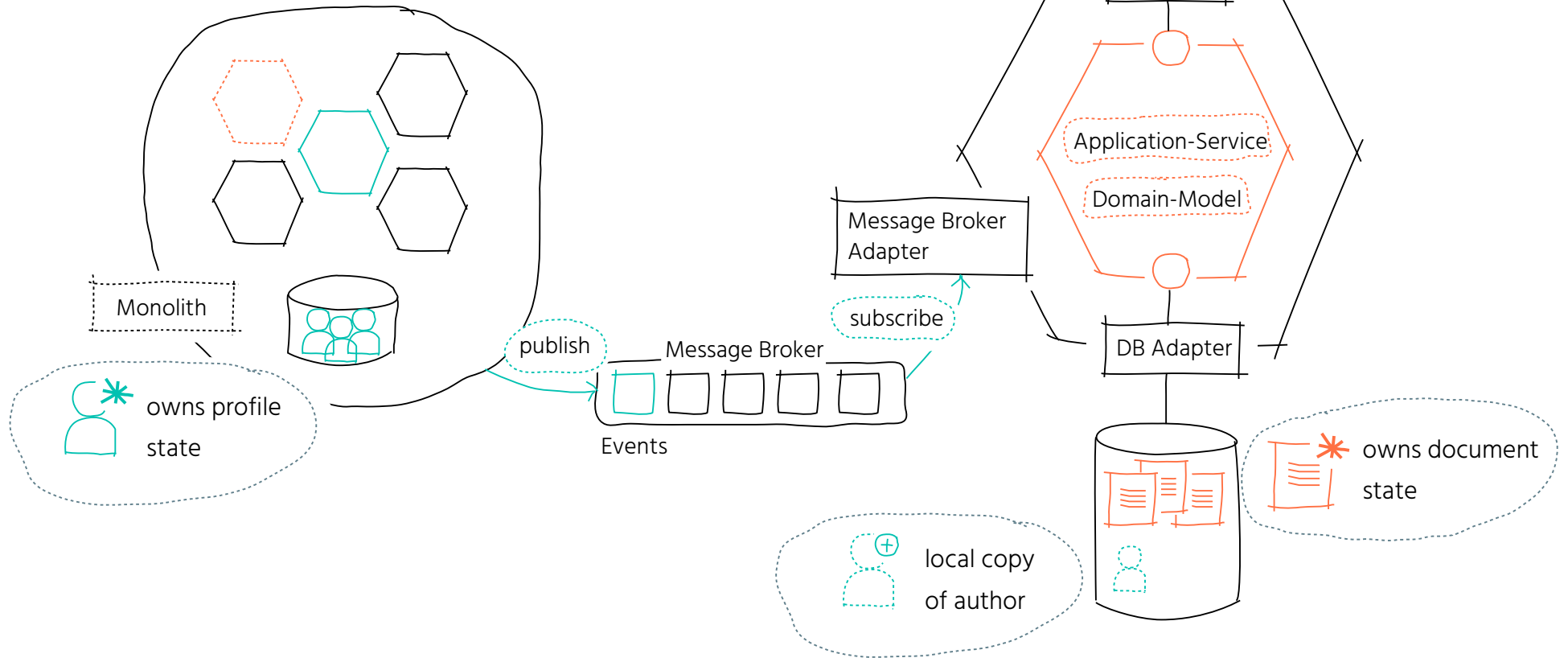
Decomposition Strategy

 Co-Existing Service From Scratch



Decomposition Strategy

Co-Existing Service From Scratch

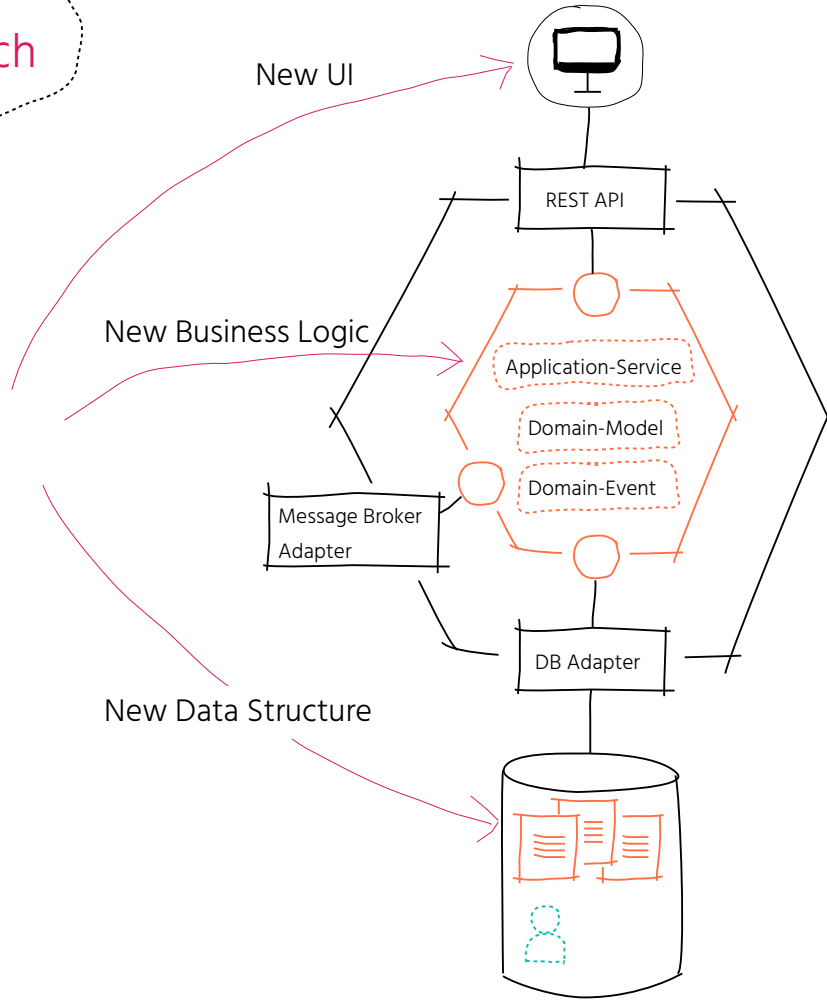
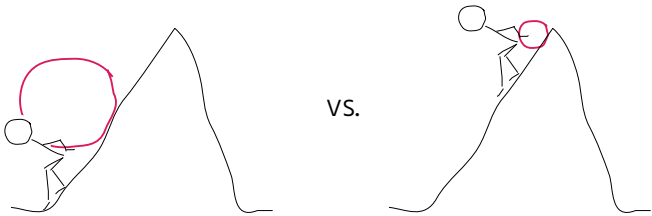


Decomposition Strategy

 Co-Existing Service From Scratch

Good approach in general,
but we did **too many steps at once**

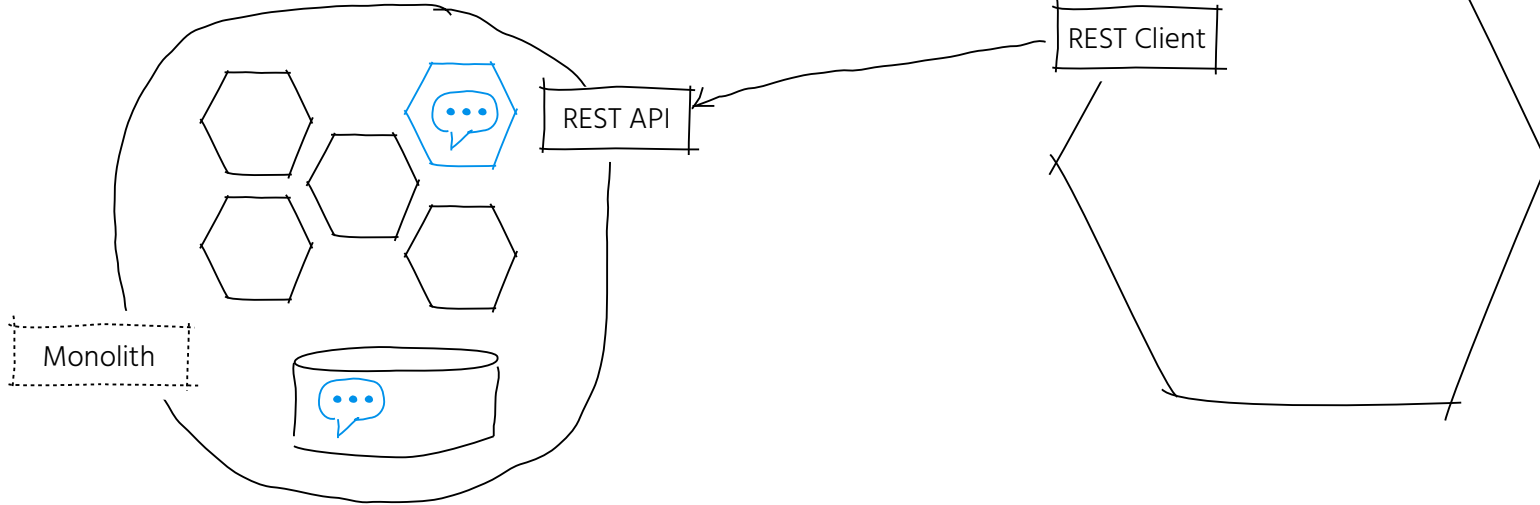
=> Not optimal to start with



Decomposition Strategy



Incremental Top Down



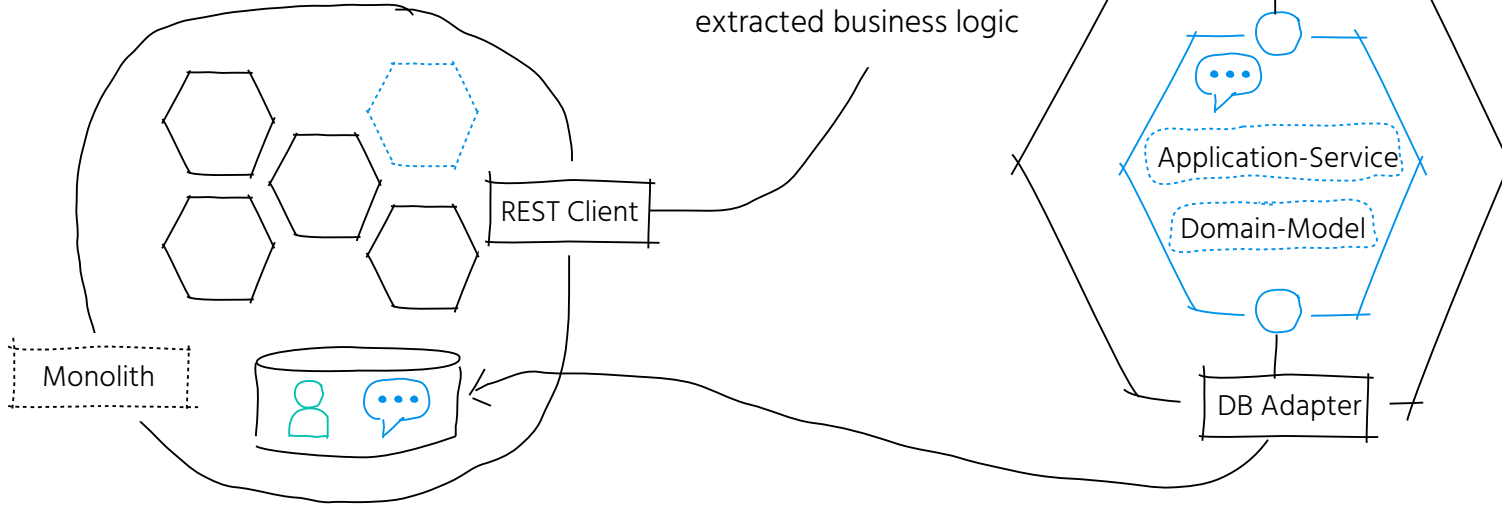
1.

Extracting Web App

Decomposition Strategy



Incremental Top Down



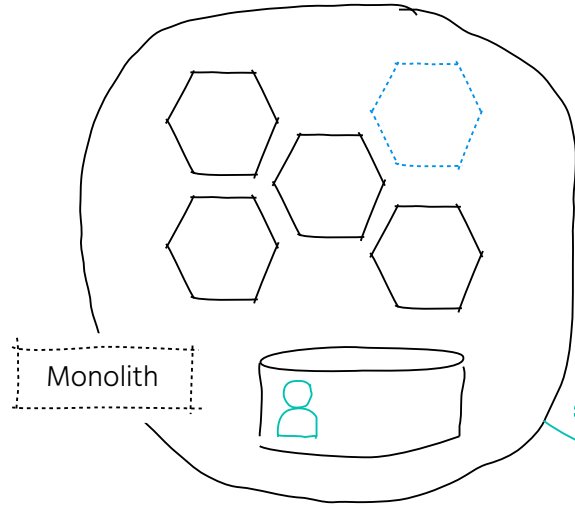
2.

Extracting Business Logic

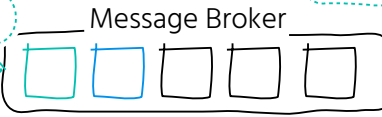
Decomposition Strategy



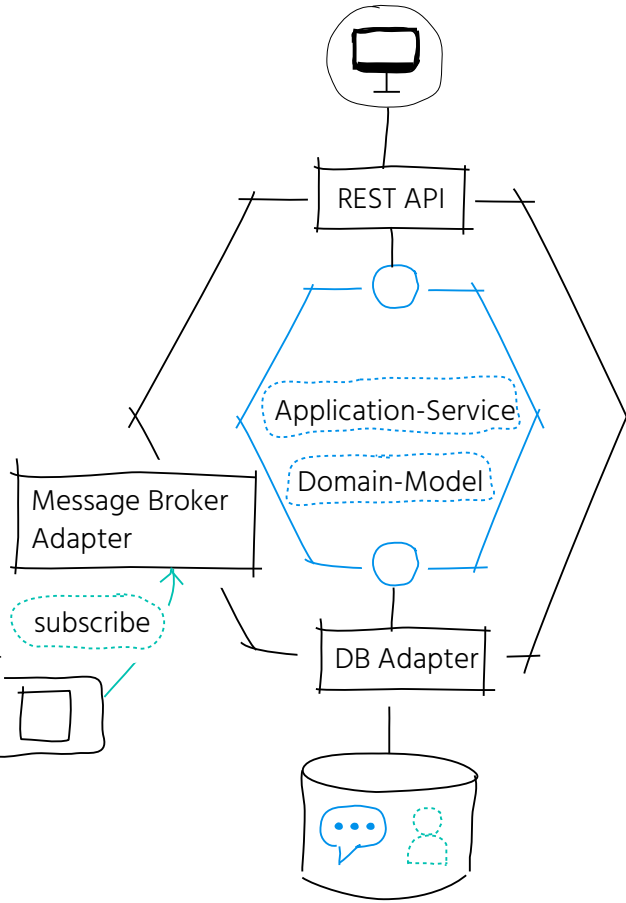
Incremental Top Down



publish



Events

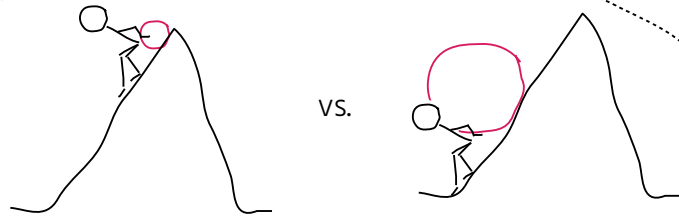


3.

Splitting Data Storage

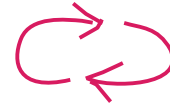
Decomposition Strategy

1. —
 2. —
 3. —
- Which One First?

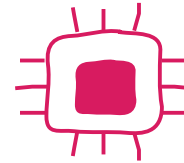


Easy to Extract

Early experiences w/ Microservices



Changing Frequently



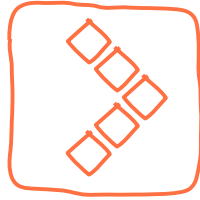
Different Resource
Consumption

Greatest benefit
after extraction

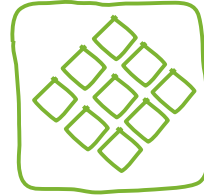
Cross-Cutting Concerns



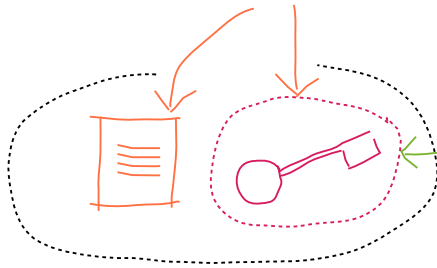
Authorization



JUST DRIVE



JUST WIKI



Fine-grained authorization

Inter-service dependency

Cross-Cutting Concerns



Authorization

I have a new service that needs authorization. Where is the authz service I could use?

Not there, yet. Sorry!

Ok, then I am putting my code to the place where authz handling exists ... to the monolith.



Feeding the monolith

Ok, then I am implementing authz in my local service.



Re-implementing authz w/ every new service

Cross-Cutting Concerns



Handle Them Early



Feeding the monolith



Re-implementing authz w/ every
new service

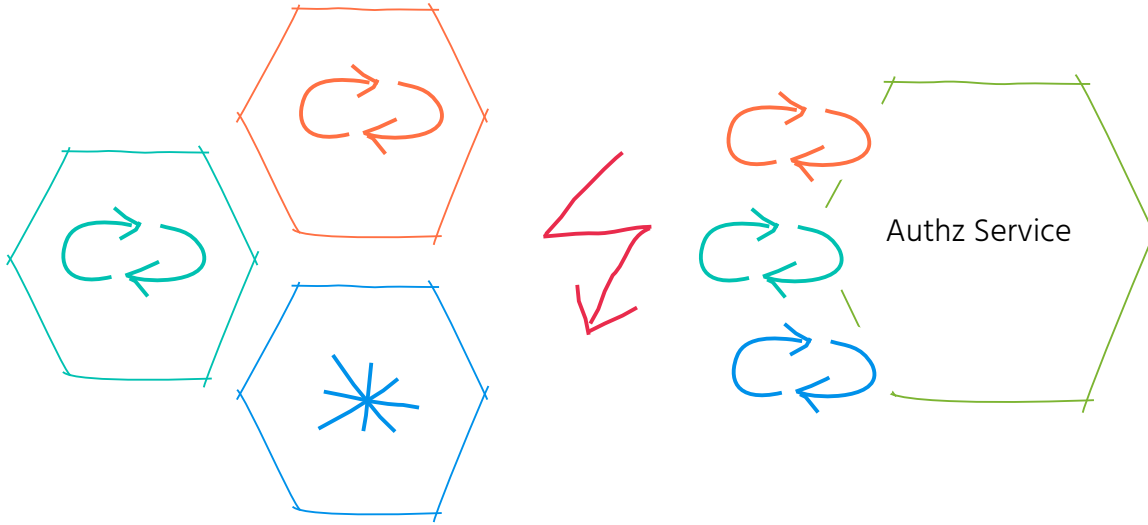


Handle Cross-Cutting Concerns Early

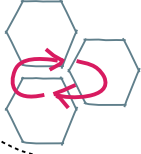
Cross-Cutting Concerns



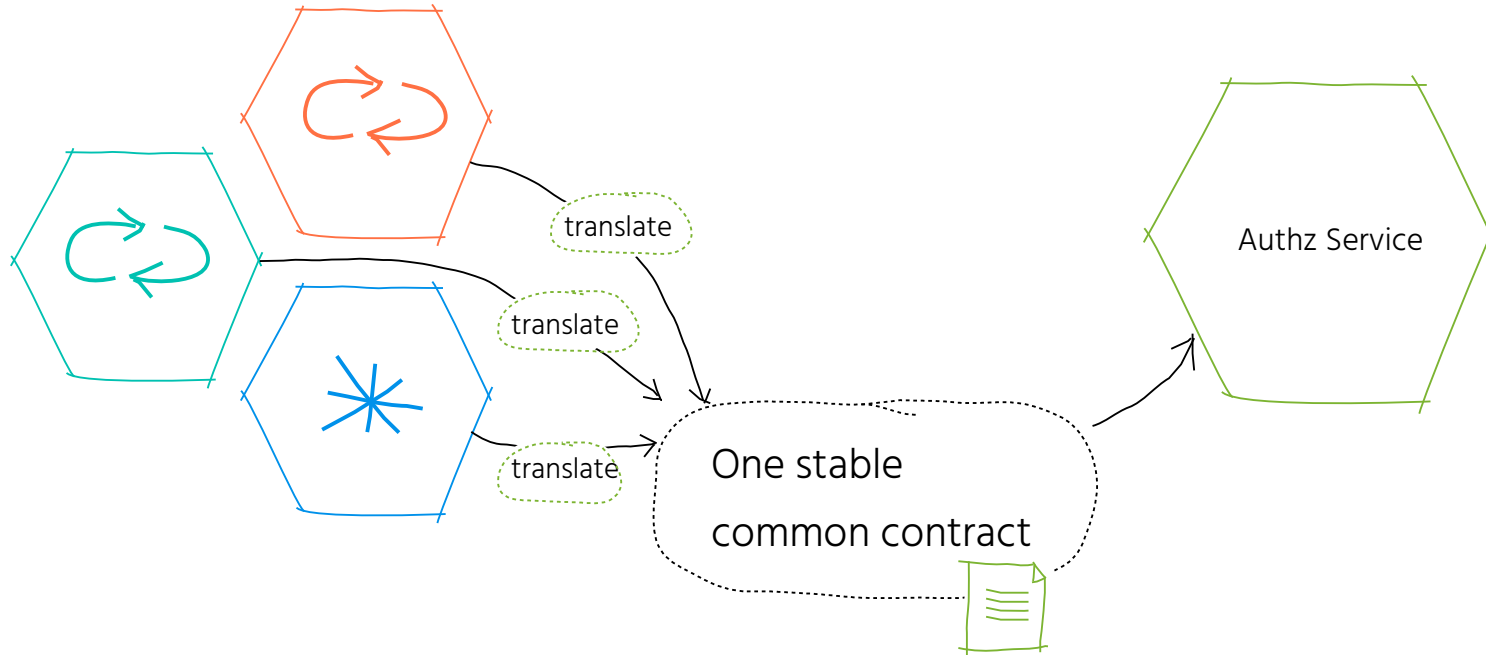
Avoid A Distributed Monolith



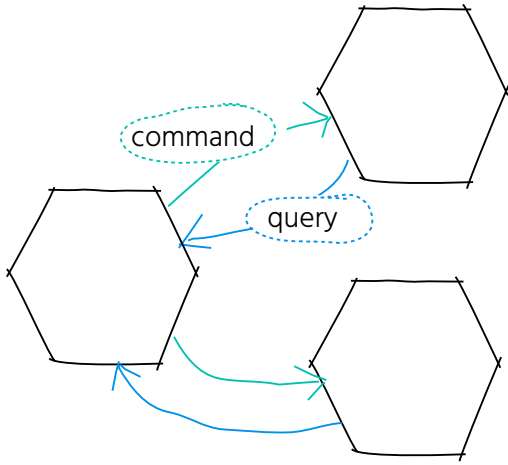
Cross-Cutting Concerns



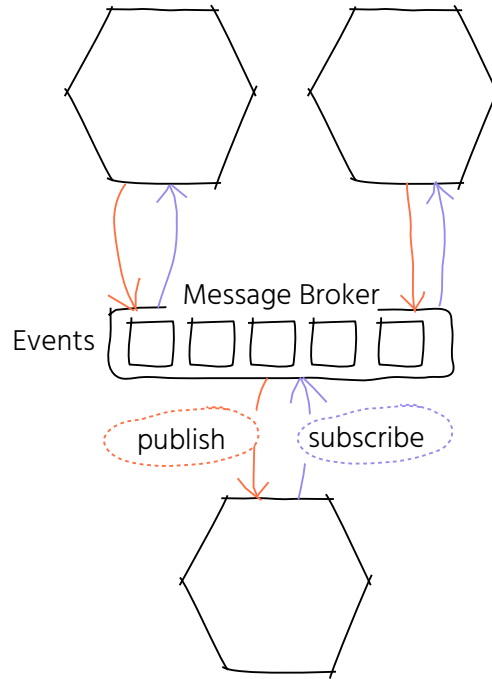
Avoid A Distributed Monolith



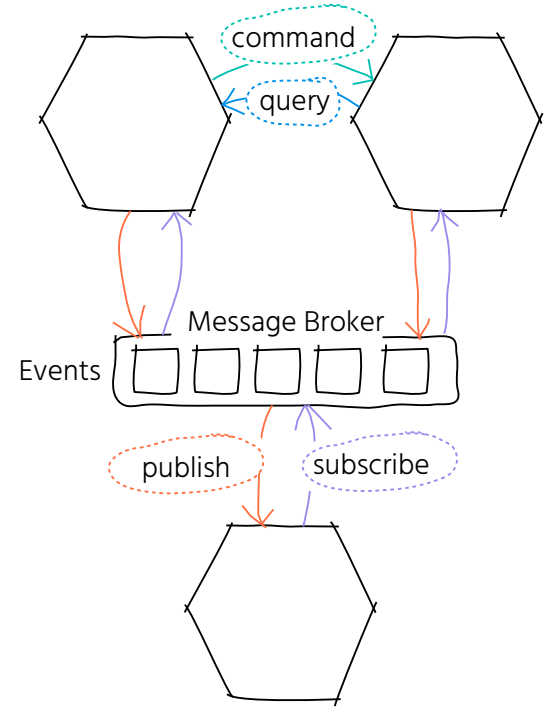
Service Interaction



Request-Driven

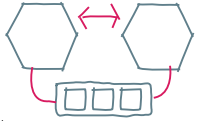


Event-Driven

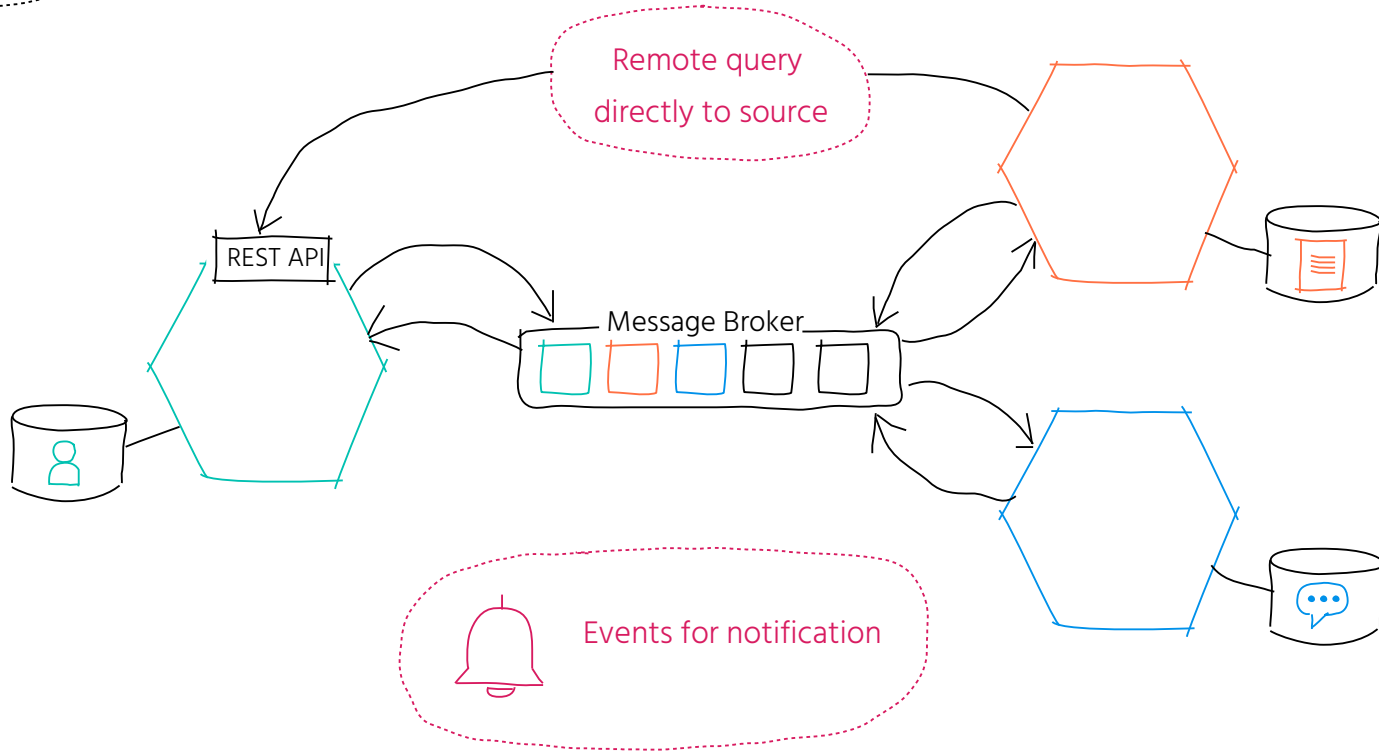


Hybrid

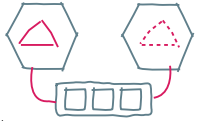
How To Manage Shared Data?



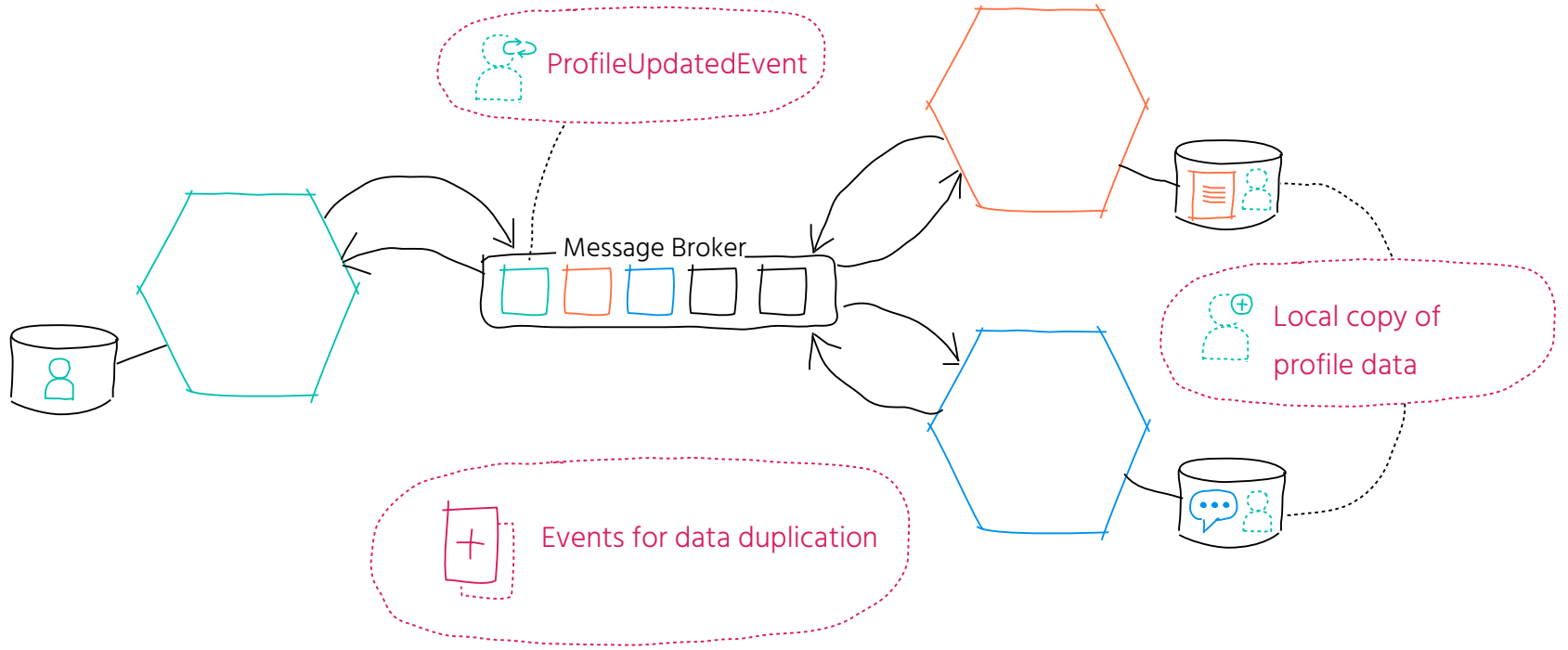
Hybrid Model



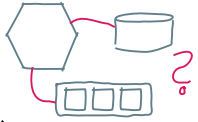
How To Manage Shared Data?



Event Driven State Transfer

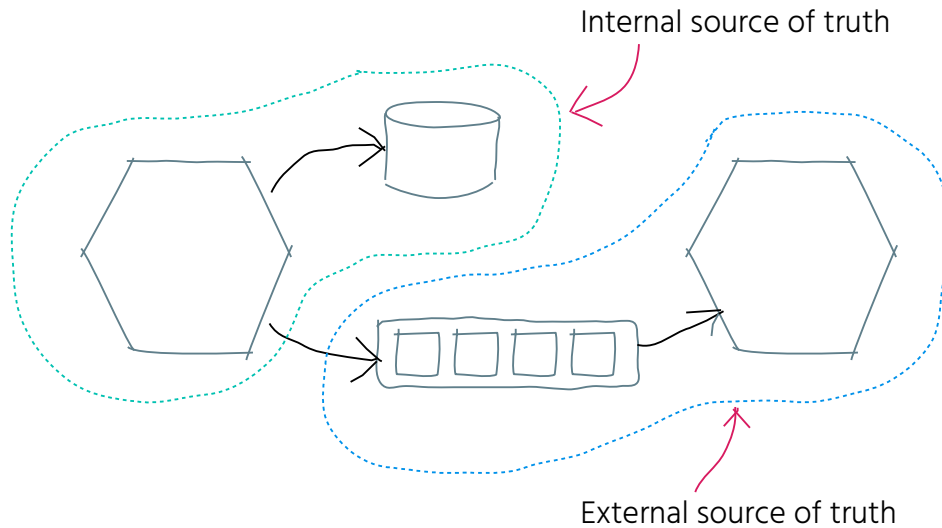


How To Manage Shared Data?



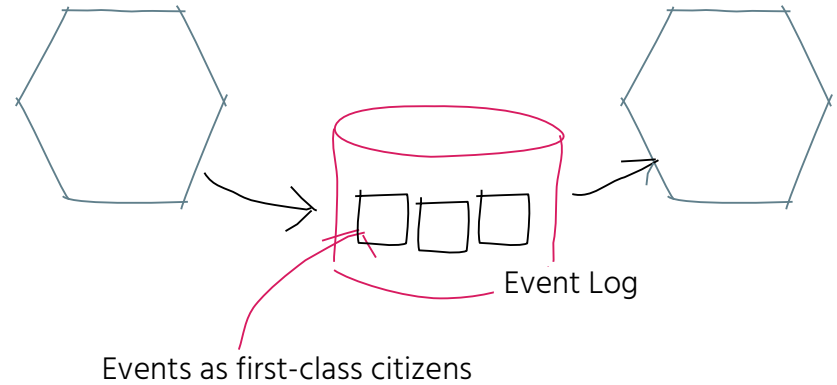
Source Of Truth

Multiple sources of truth

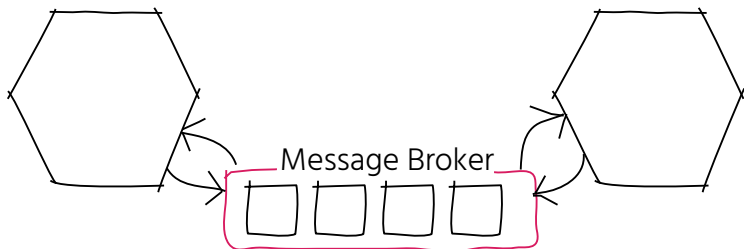


"Traditional" Event-Driven System

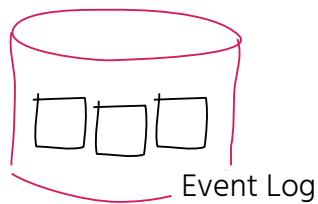
Single source of truth



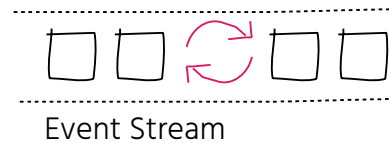
Event Store



Messaging System



Storage System



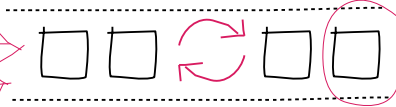
Streaming Platform

How To Manage Shared Data?



Kafka Streams

Unbounded, ordered sequence
of data records



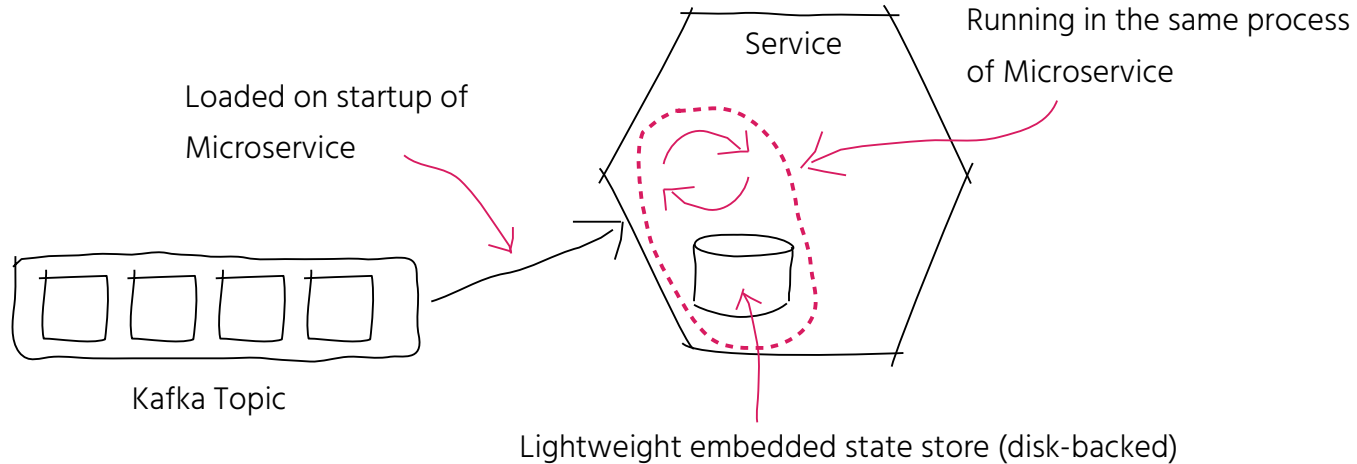
Continuously
updating

Key-value pair

How To Manage Shared Data?



Kafka Streams



Streams make data available wherever it's needed



How To Manage Shared Data?



Kafka Streams

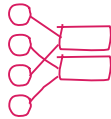
Kafka Streams API



join



filter



group by



aggregate

etc.

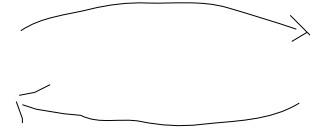
Kafka Stream-Table Duality



KStream

Changelog of state changes

(key1, value1), (key2, value2), (key 1, value 3)



KTable

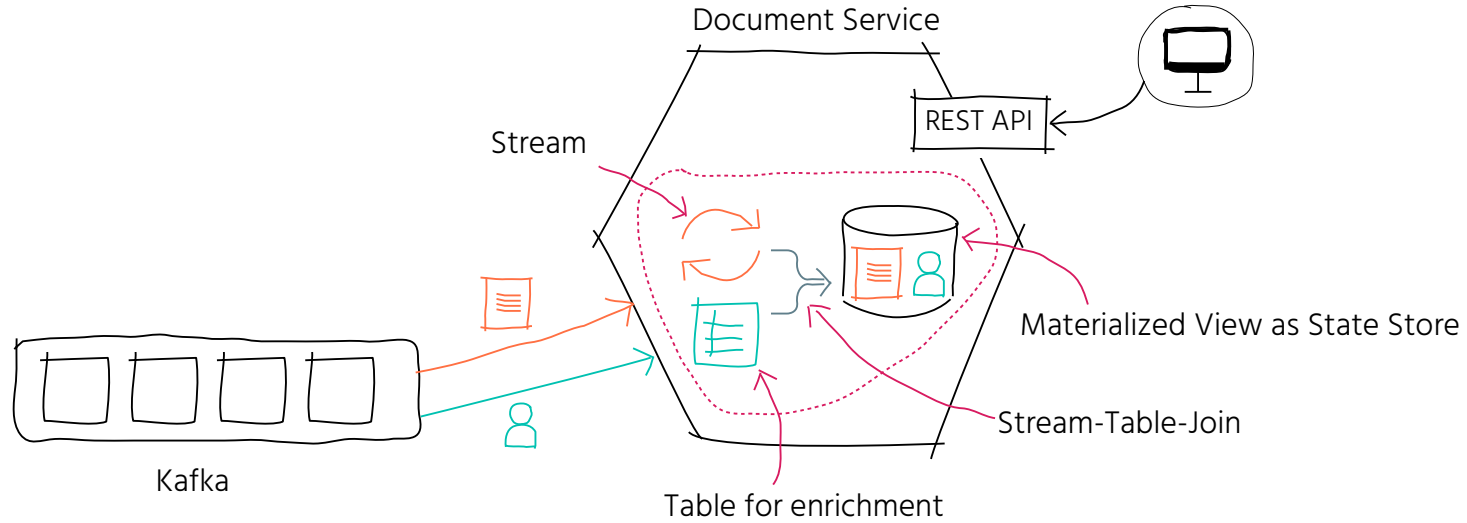
Snapshot of the latest
value for each key

key1 → value3

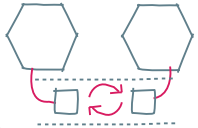
key2 → value2

How To Manage Data?

Materialized Views w/ Kafka Streams



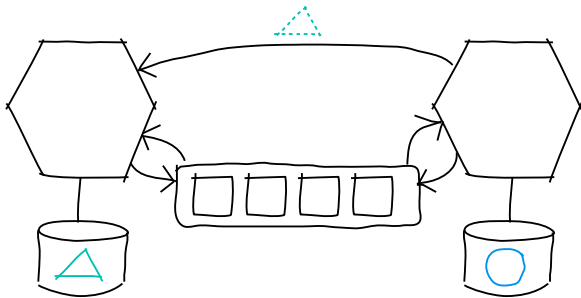
How To Manage Shared Data?



Event Streams as a Shared Source of Truth



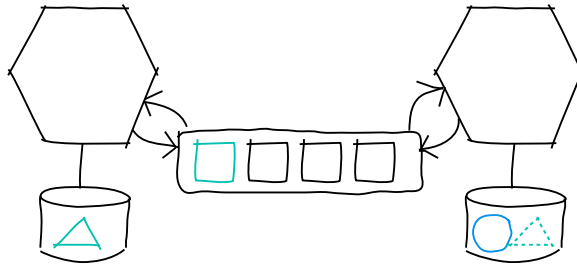
Events for notification



- Simple integration
- Remote query => increasing coupling



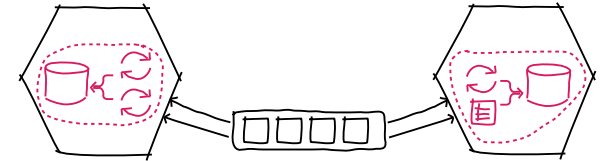
Events for data duplication



- Eliminating remote query => better decoupling
- Local copy => better autonomy
- Duplicating effort to maintain local dataset



Event streams as a shared source of truth



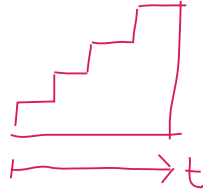
- Eliminating local copy => reduces duplicating effort
- Pushes data to where it's needed
- Increases pluggability
- Low barrier to entry for new service



Lessons Learned



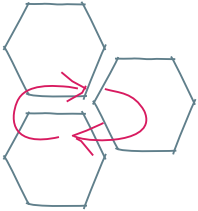
Start small



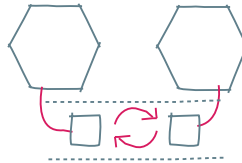
One manageable step at a time



Handle cross-cutting concerns early



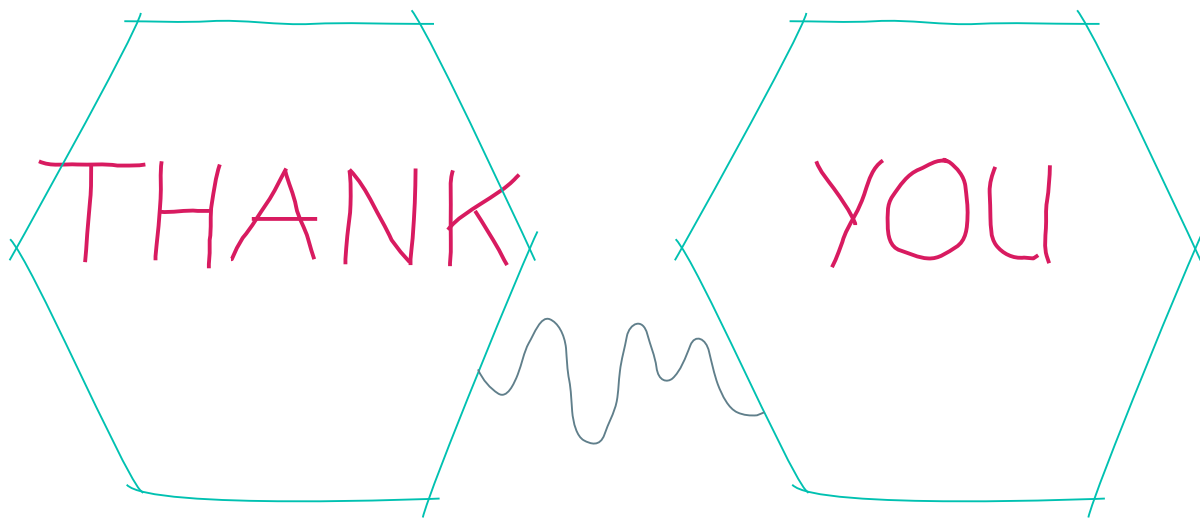
Avoid a distributed monolith



Design event-driven & consider
event streams as shared source of truth



Be aware of affecting circumstances
&
Each journey is different :)



Susanne Kaiser

@suksr

EX - CTO at Just Software

@JustSocialApps