Microservices Lessons Learned From a Startup Perspective

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 Archtitect,

former Netflix Chief Cloud Architect

Affecting Circumstances

Team

- Size
- Skillset
- Structure

Legacy-System

- Maintenance effort
- Environment

Strategy

- New Features
- Timeline/Milestones

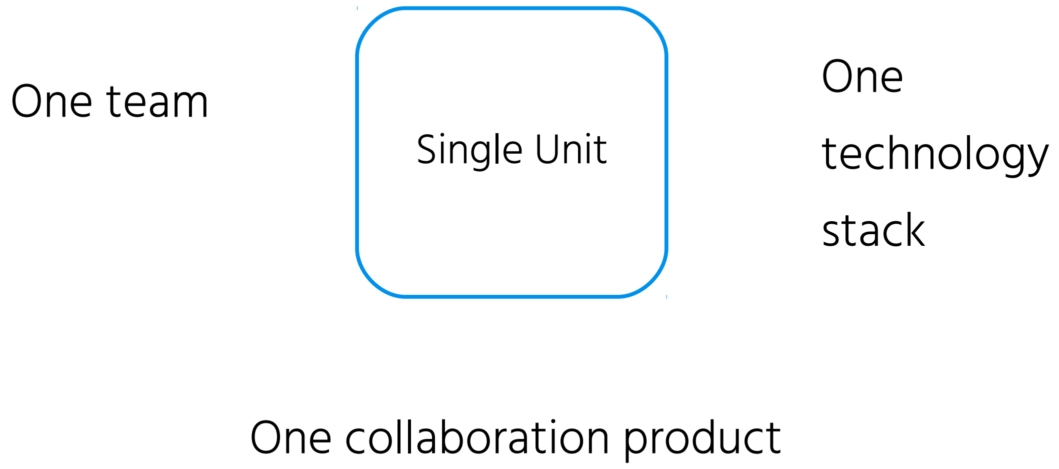
Background

JUST SOCIAL

JUST PAGE Social Network	JUST CONNECT Real-time collaboration
JUST TASKS Task Management	JUST DRIVE Document Sharing
JUST PEOPLE User Management	

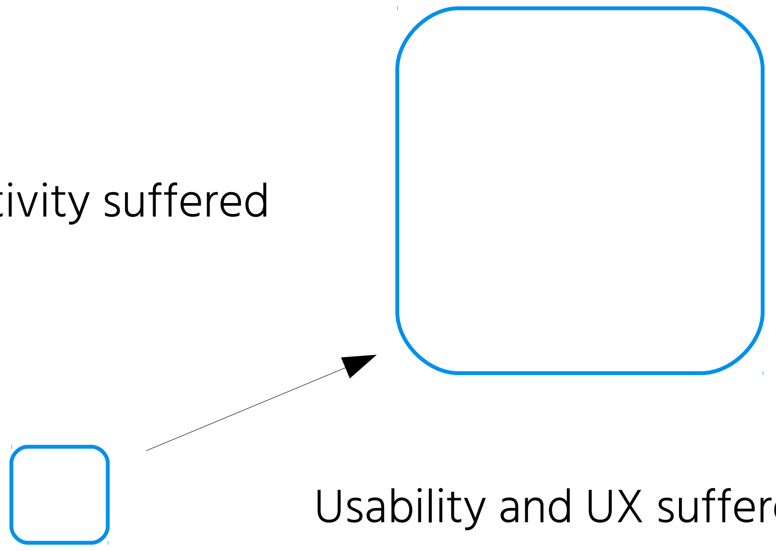
Background

At The Beginning … A Monolith In Every Aspect



Background

After An Evolving Time ...



Productivity suffered

New features

released slowly

Usability and UX suffered

Background

Separate Collaboration Apps

JUST PAGE
Social Network

JUST CONNECT
Real-time collaboration

JUST TASKS
Task Management

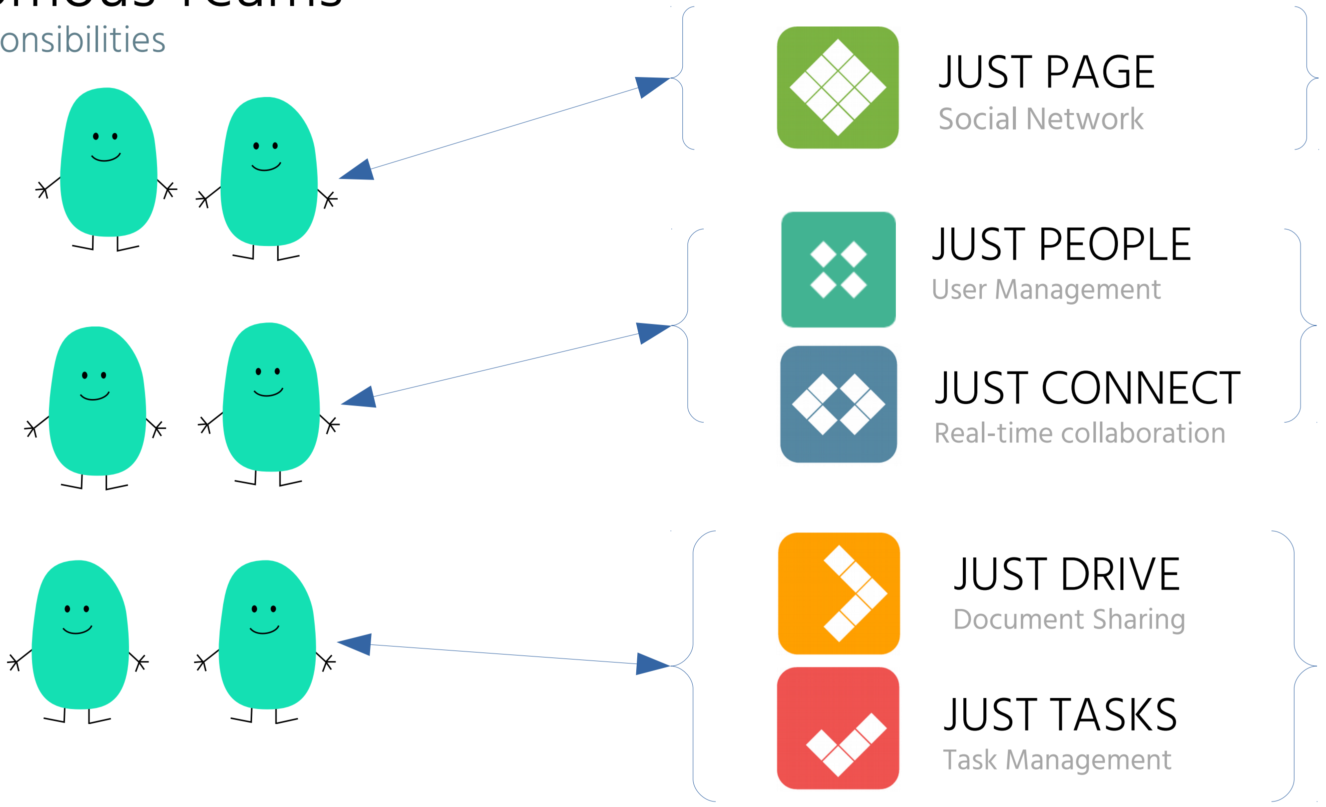
JUST DRIVE
Document Sharing

JUST PEOPLE
User Management

Background

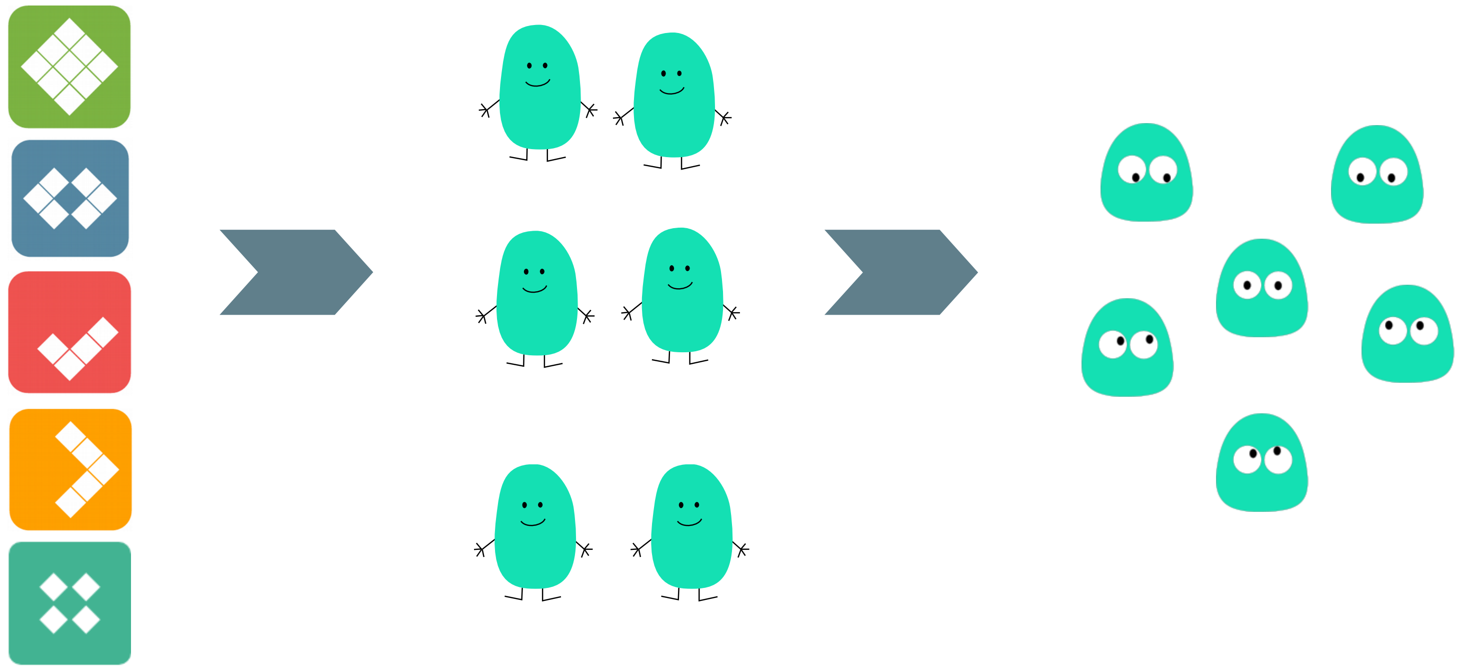
Small, Autonomous Teams

With well-defined responsibilities



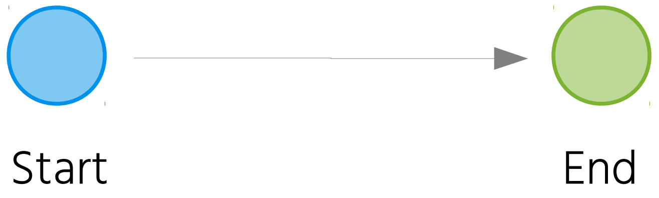
Background

In The Long Run …

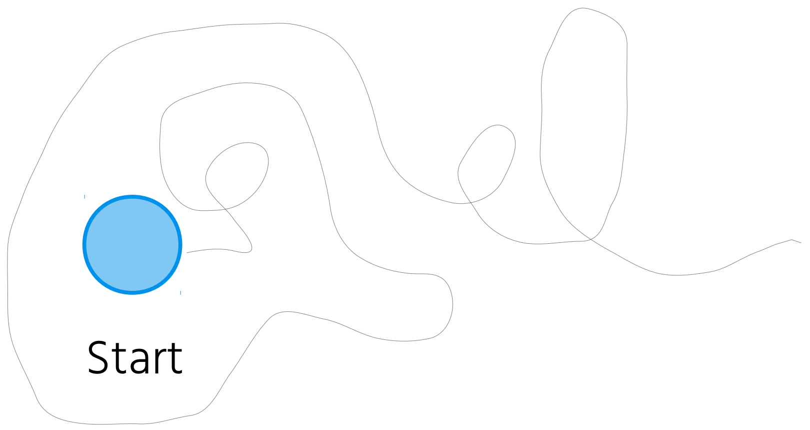


Straightforward Process?

No Straightforward Process!

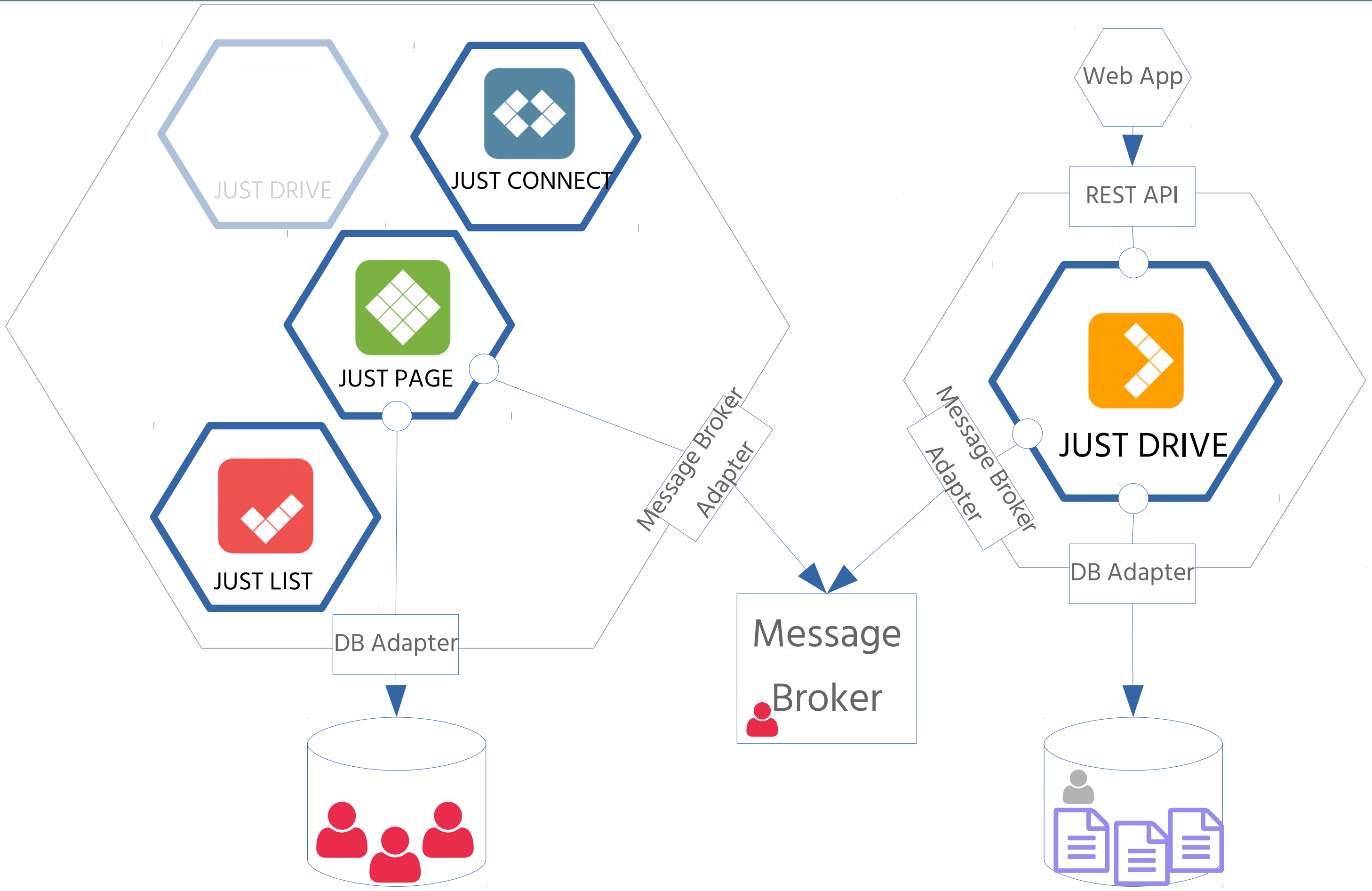


Theory



Reality

First Approach As Co-Existing Service

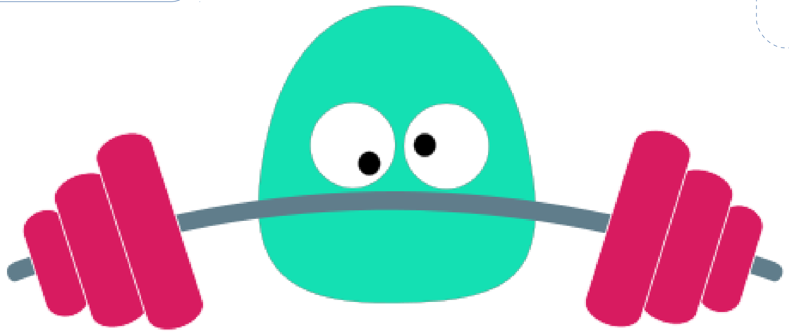


Lesson #1: Too Many Steps At Once Slow You Down

New UI

Maintain & run current system

New data structure



Timelines

More features

Start With One Manageable Step At A Time

Easy to extract

Changing frequently

Different resource requirements

Split in steps, e.g. top/down

Lesson #2: Deferring Solving Authz Handling Hurts

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

Not there, yet. Sorry!

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

Feeding the monolith

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

Not there, yet. Sorry!

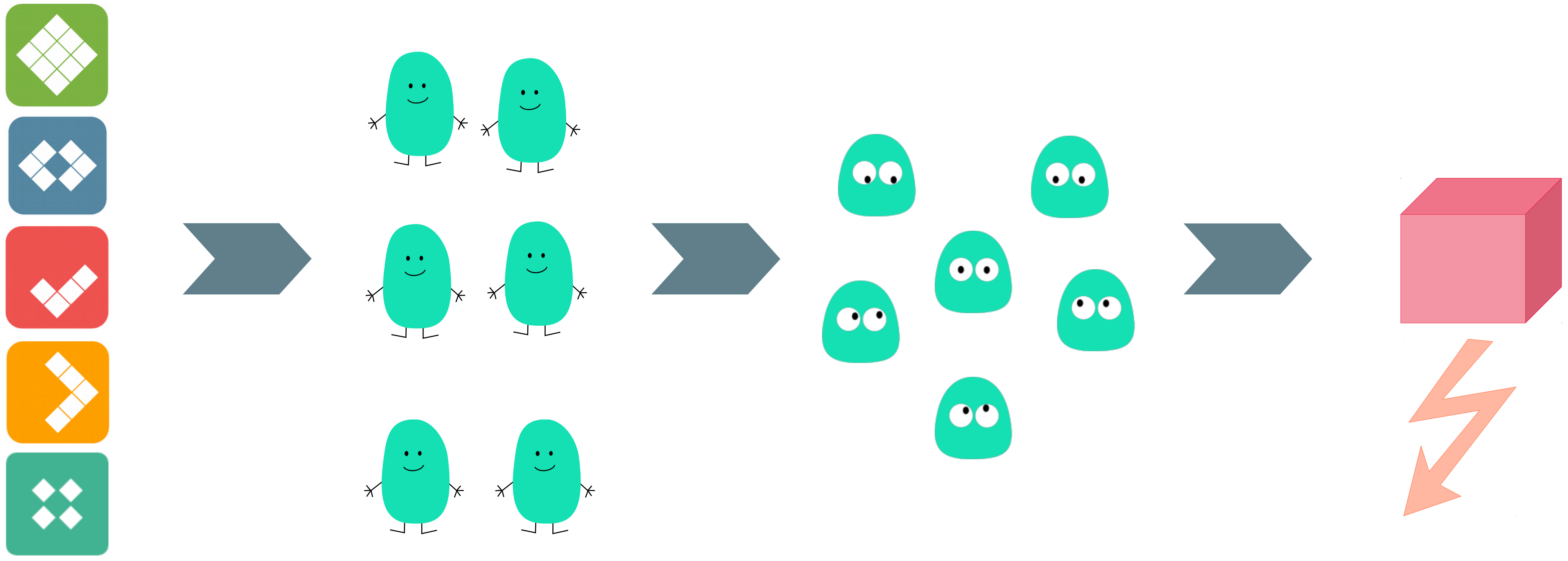
Ok, than I am implementing authz in my local service.

Re-implementing authz w/ every service

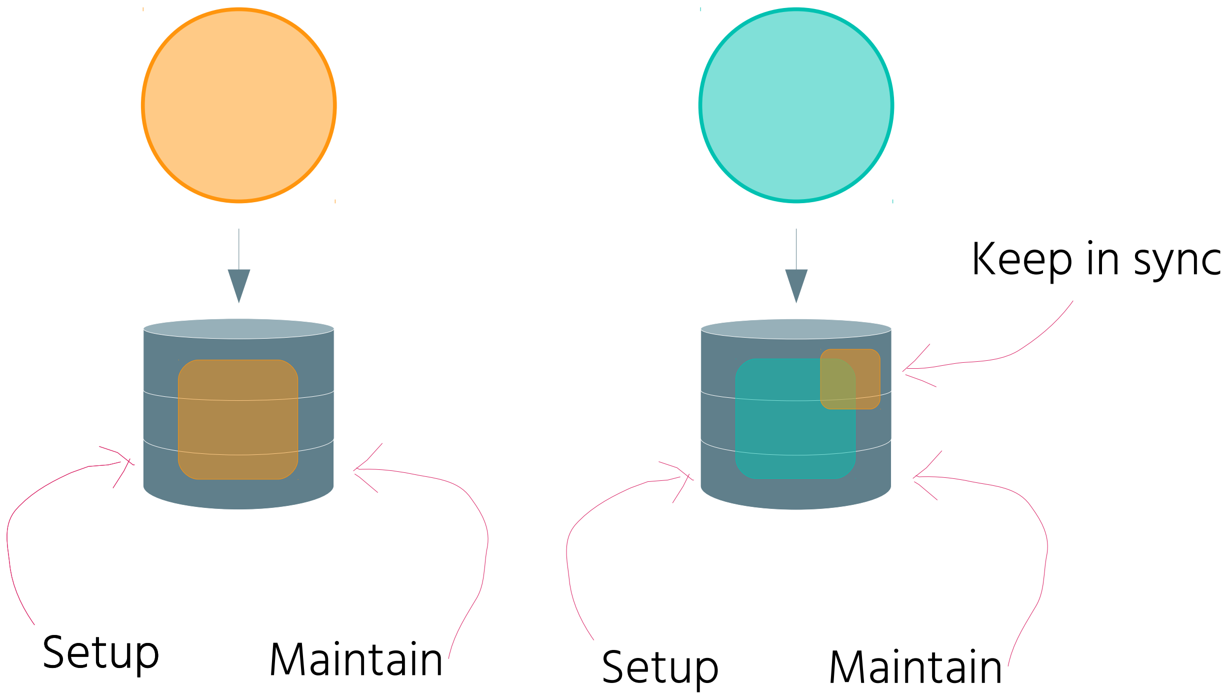
Lesson #2: Deferring Solving Authz Handling Hurts

Solve Authz Handling Early!

Lesson #3: Less Aligned Strategy Is Expensive



Lesson #4: Data Related Overhead

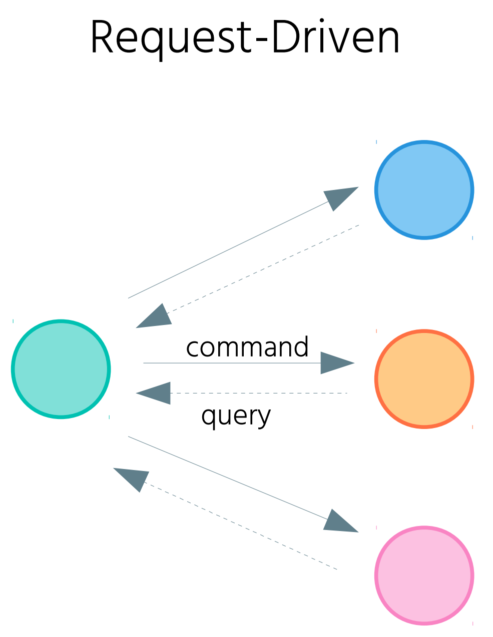


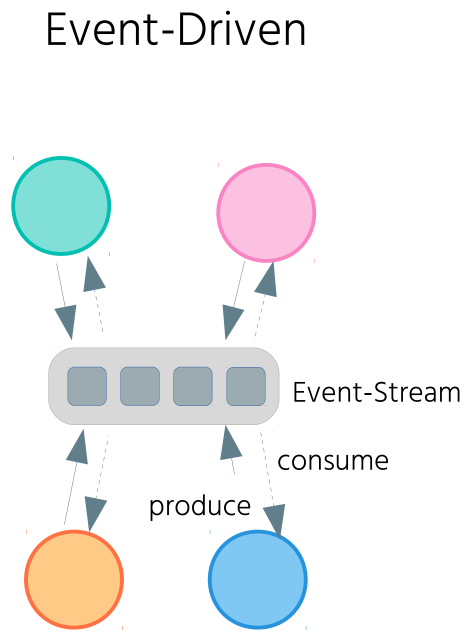
Lesson #4: Data Related Overhead

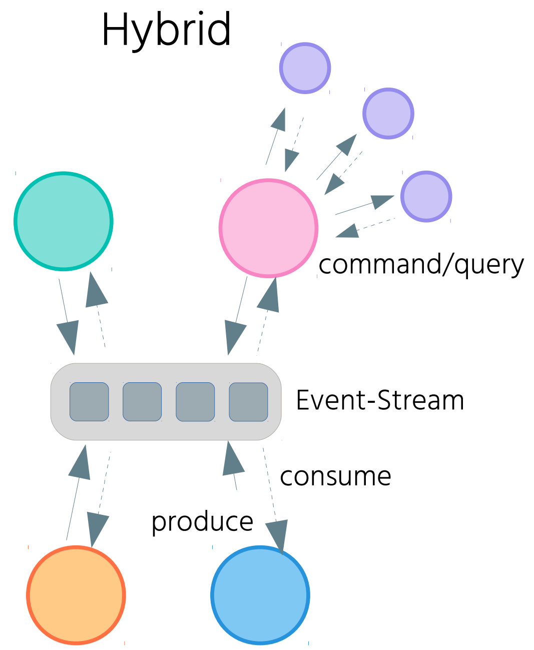
APACHE kafka®

Streams

How To Interact Between Services?

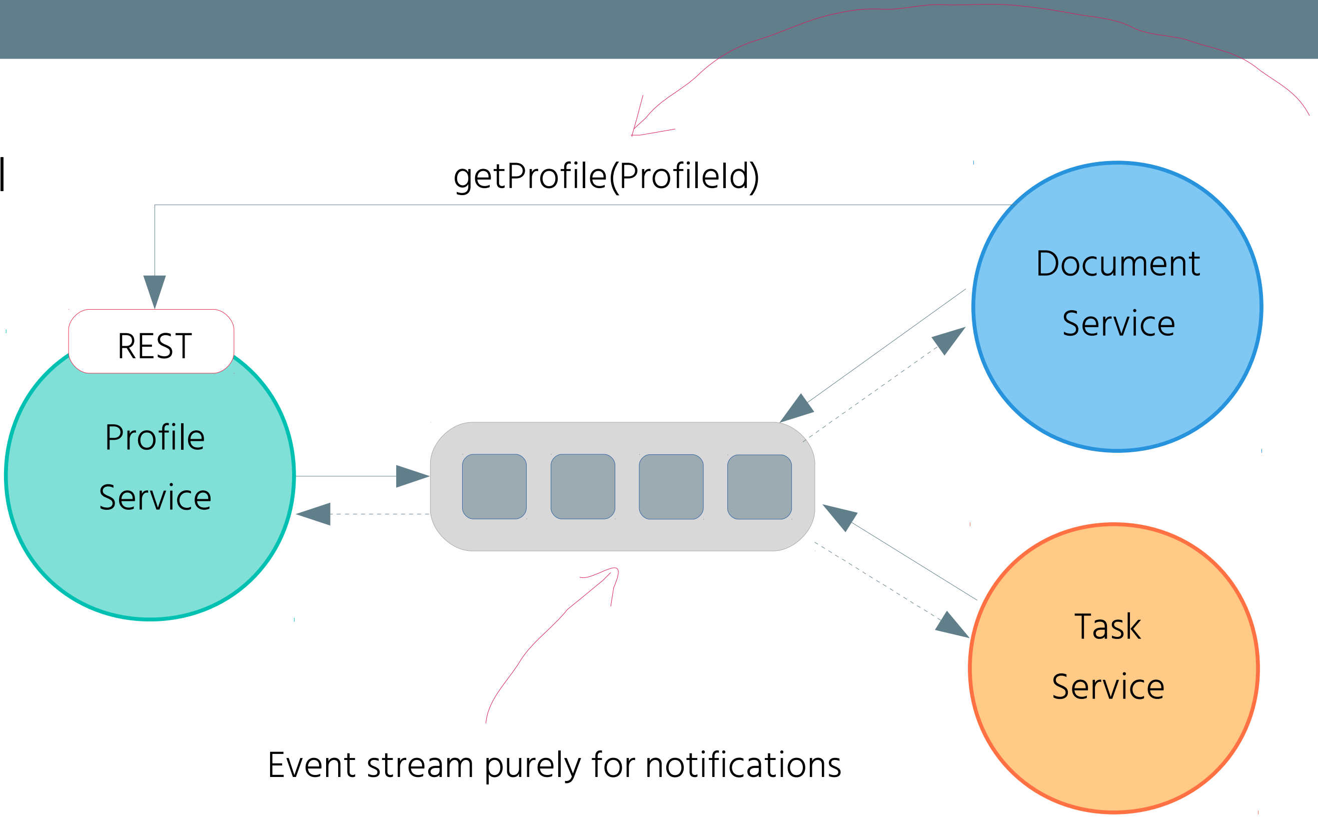






How To Manage Data?

Hybrid Model

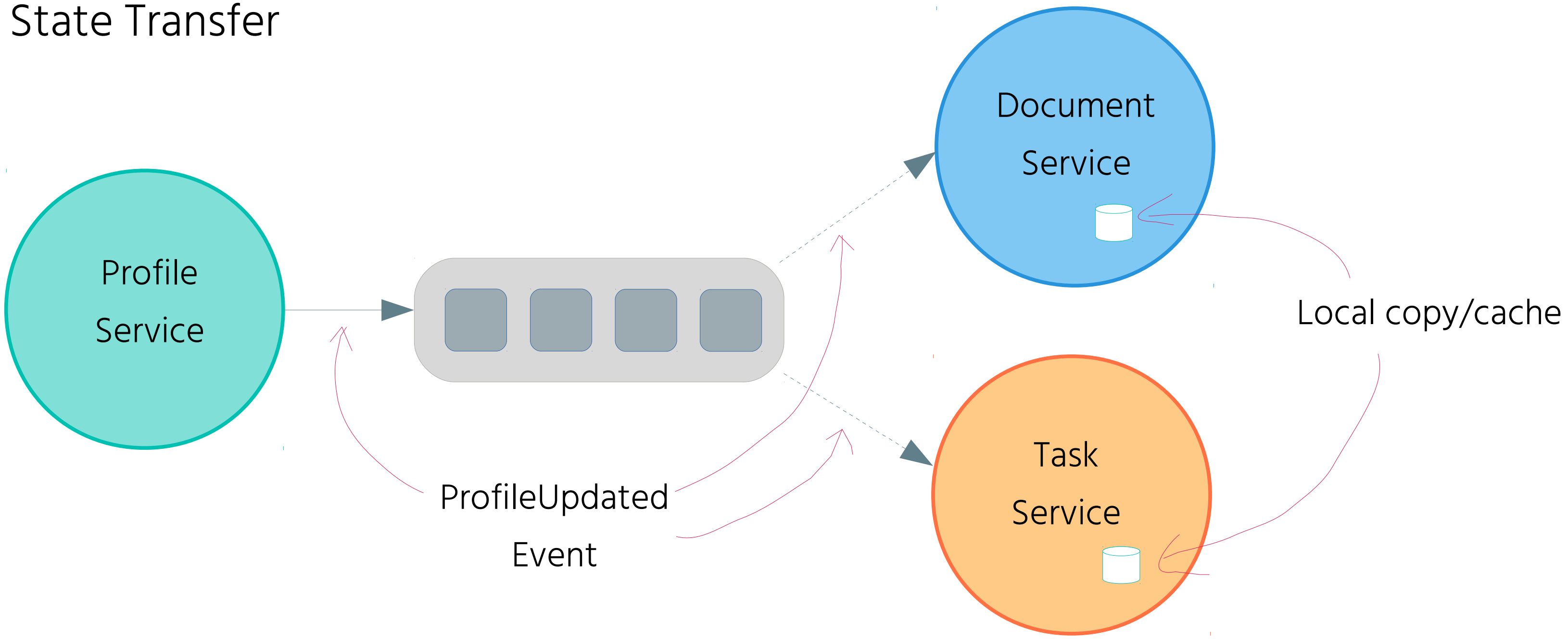


Remote query directly to source

Event stream purely for notifications

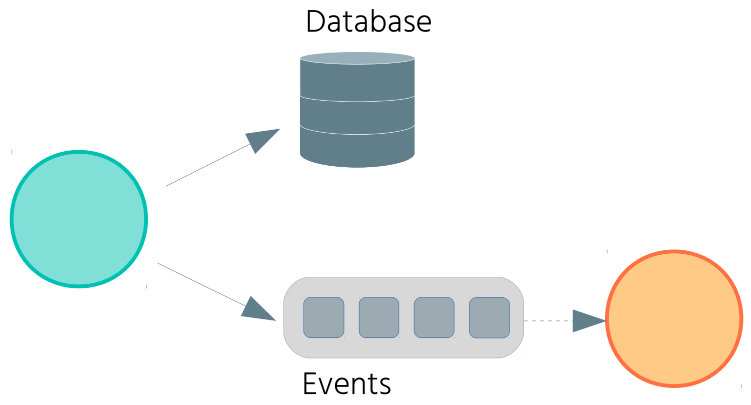
How To Manage Data?

Event-Driven State Transfer



How To Manage Data?

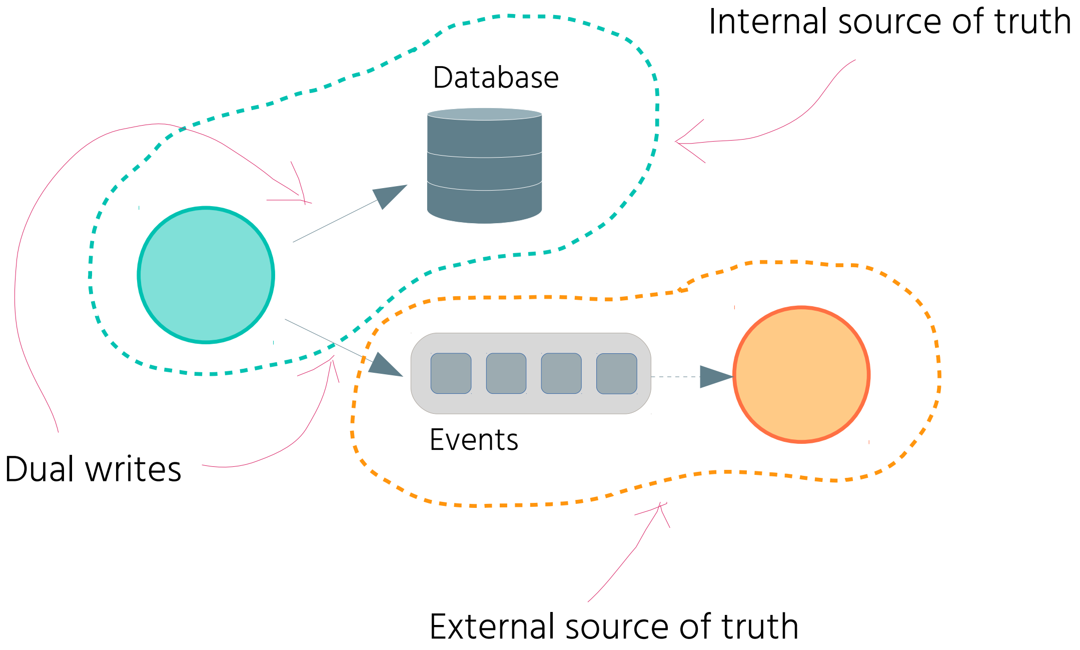
Source Of Truth



“Traditional” Event-Driven System

How To Manage Data?

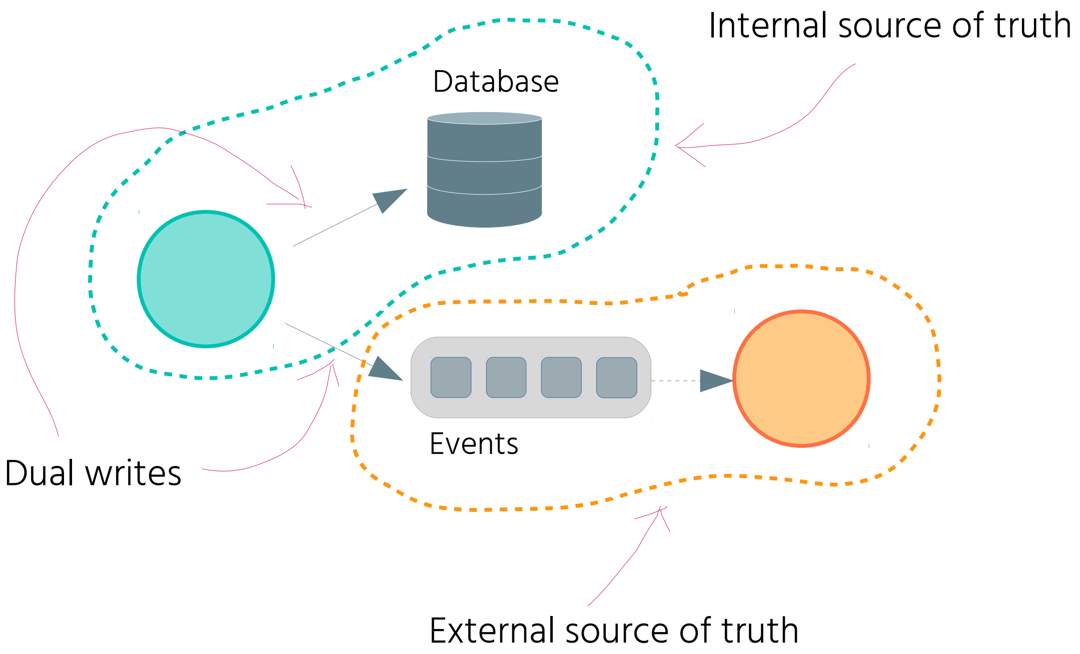
Multiple Sources Of Truth



“Traditional” Event-Driven System

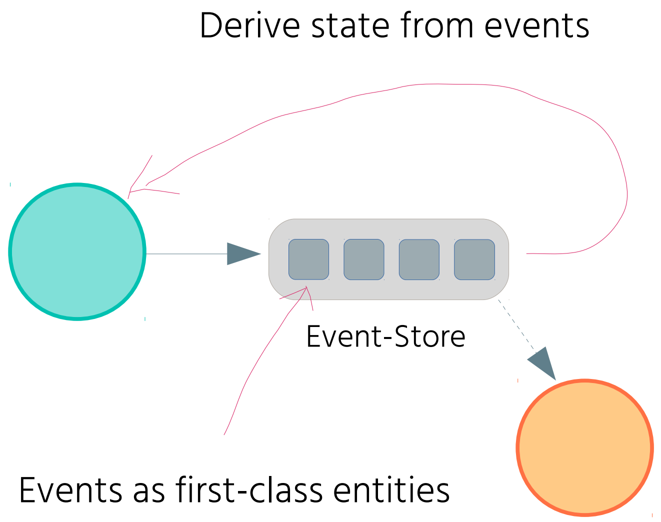
How To Manage Data?

Multiple Sources Of Truth



"Traditional" Event-Driven System

Single Source Of Truth



Event Sourcing

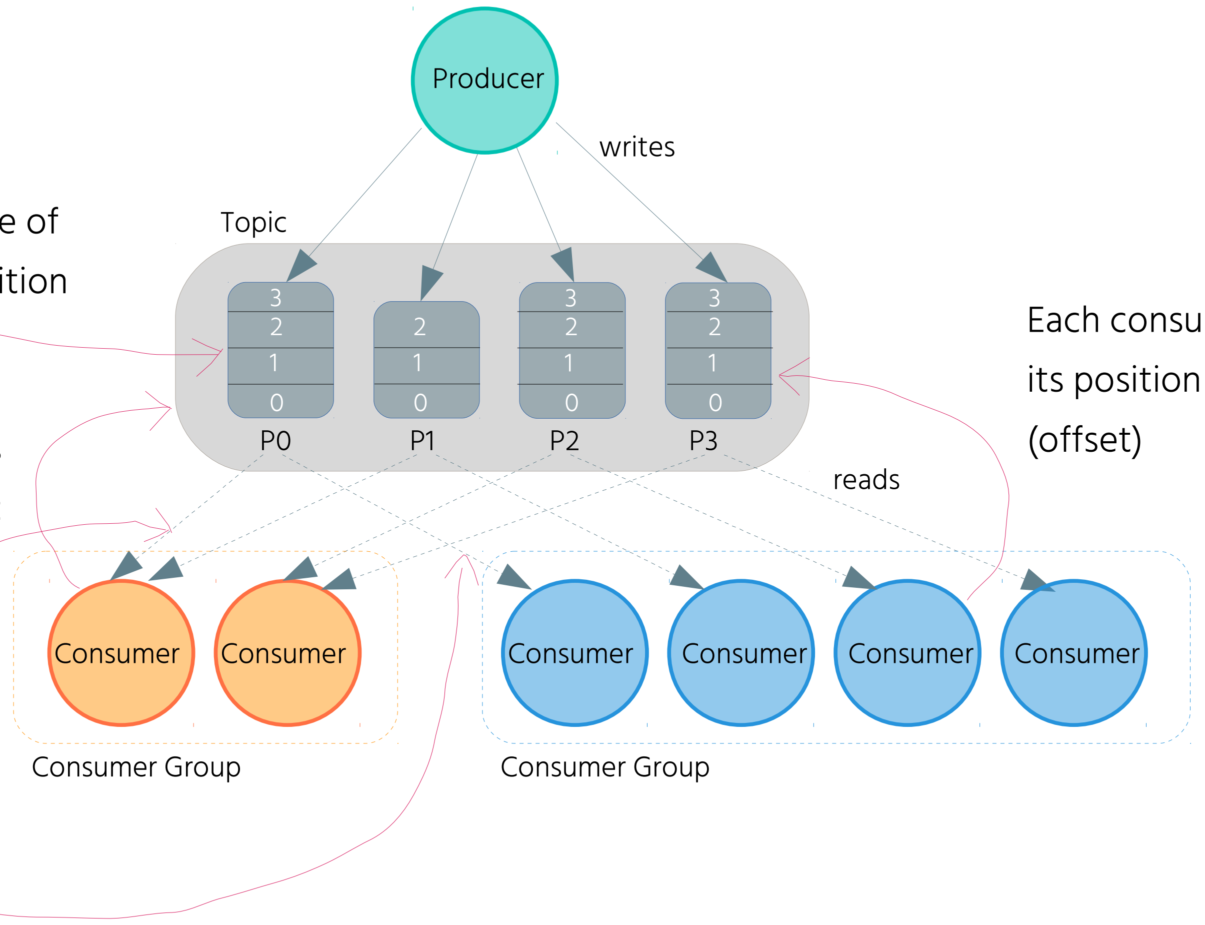
Apache Kafka

Immutable, ordered sequence of records per partition

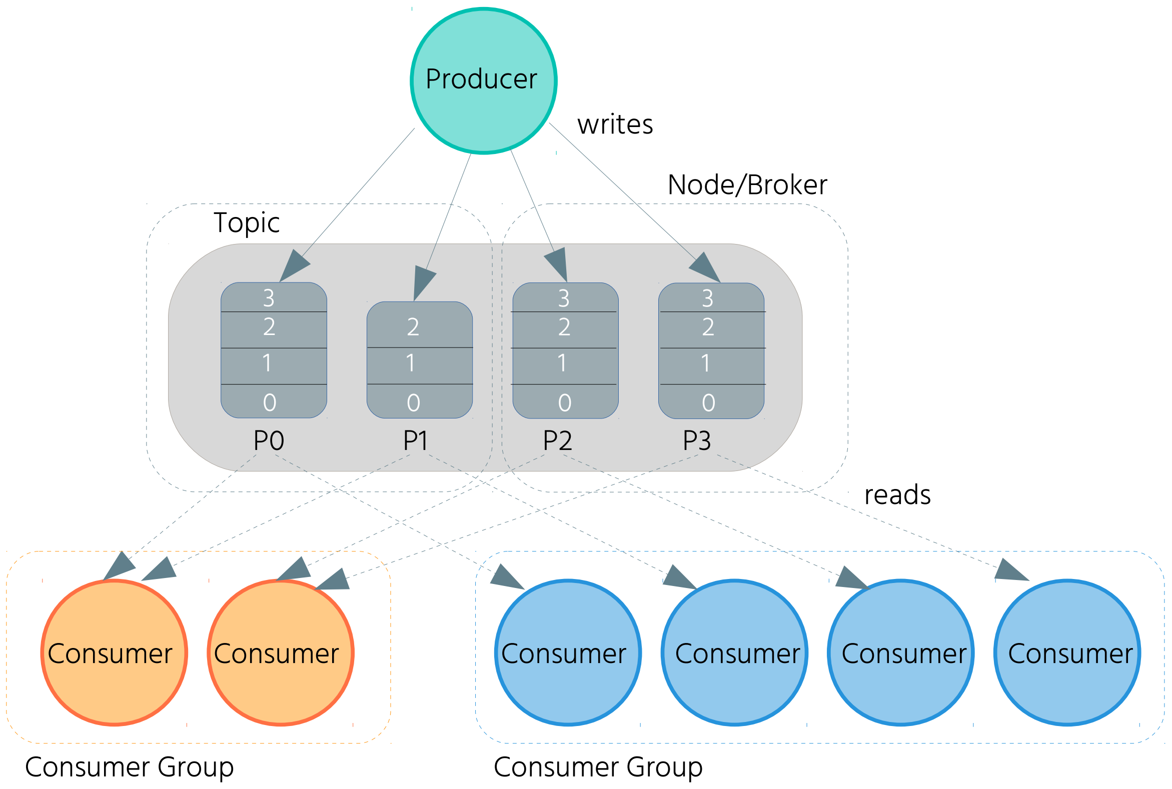
Each consumer subscribes to a topic

Each partition is assigned to one consumer within a consumer group

Each consumer controls its position per partition (offset)



Apache Kafka



Scalable

- Topic can be scaled out to several nodes
- Messages load-balanced between consumer of one group
- Add partitions for more parallelism
- Adding capacity with 0 downtime

Fault-tolerant

- Data stored to disk
- Replicated partitions
- Consumer conrols its offset

Fast

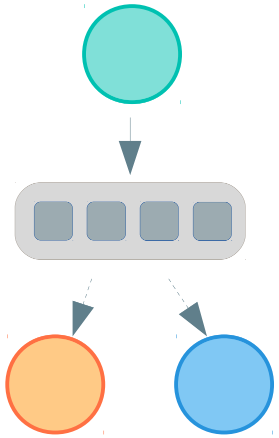
- O(1) to append messages
- LinkedIn 2016: 1.4 trillion messages/day across over 1400 brokers

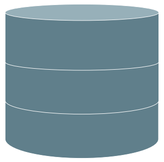
Apache Kafka Combines …

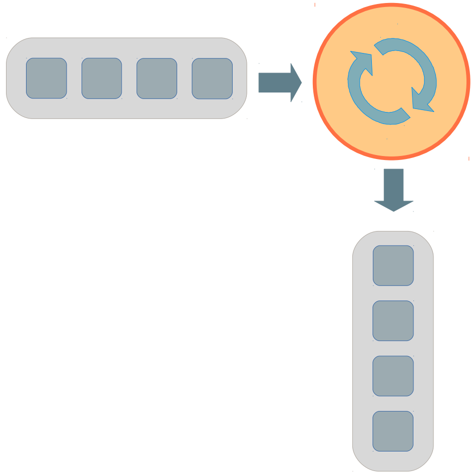
Messaging System

Storage System

Streaming Platform

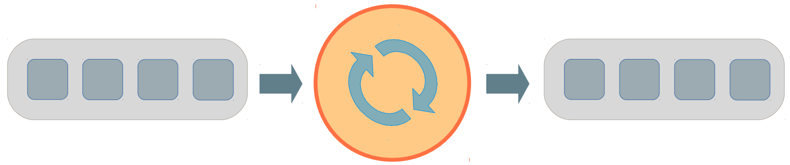




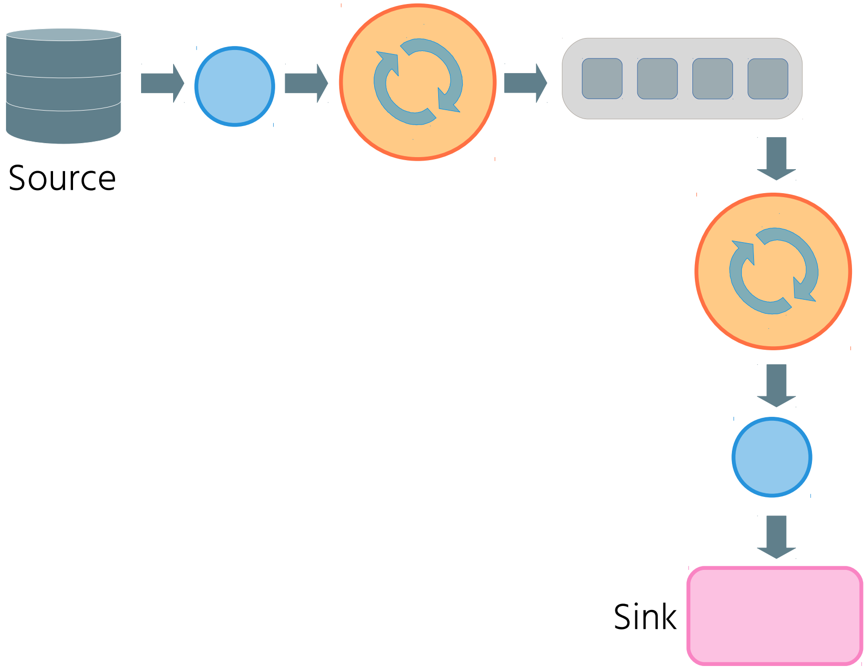


Primary Use Cases Of Streaming Platforms

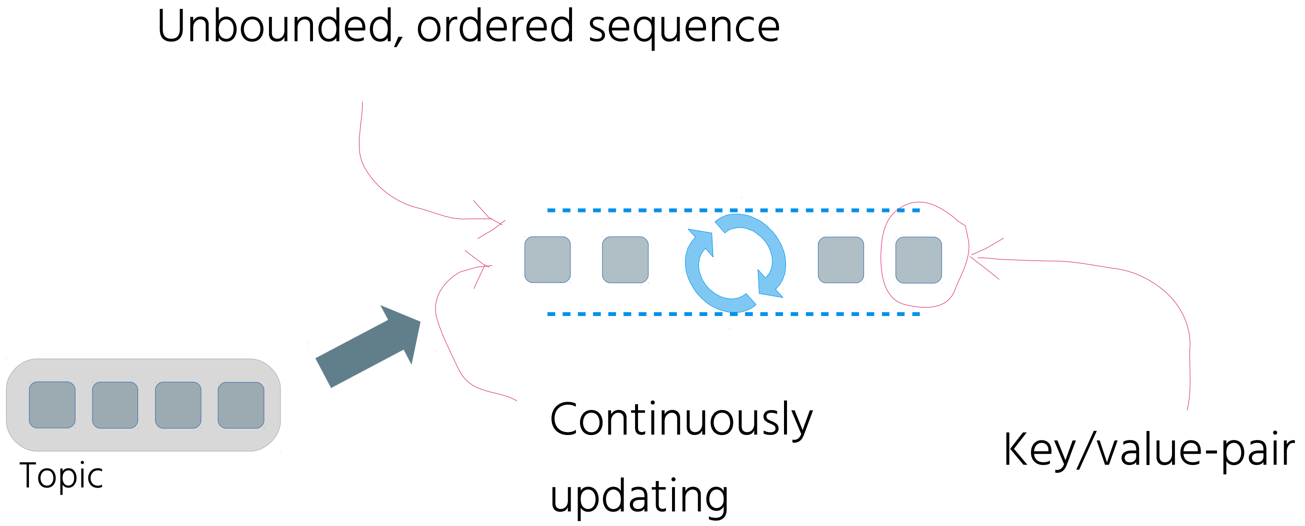
Stream Processing



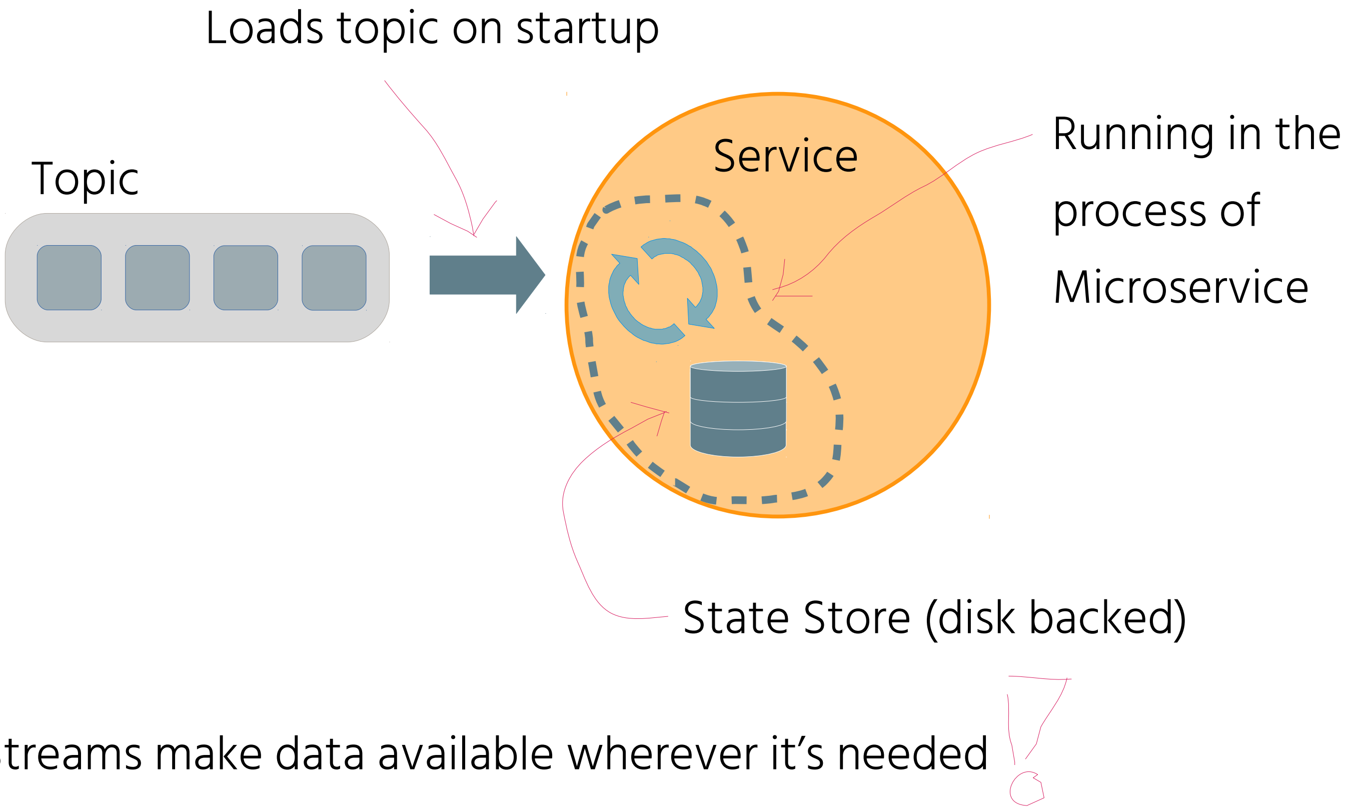
Data Integration



Kafka Streams



Kafka Streams



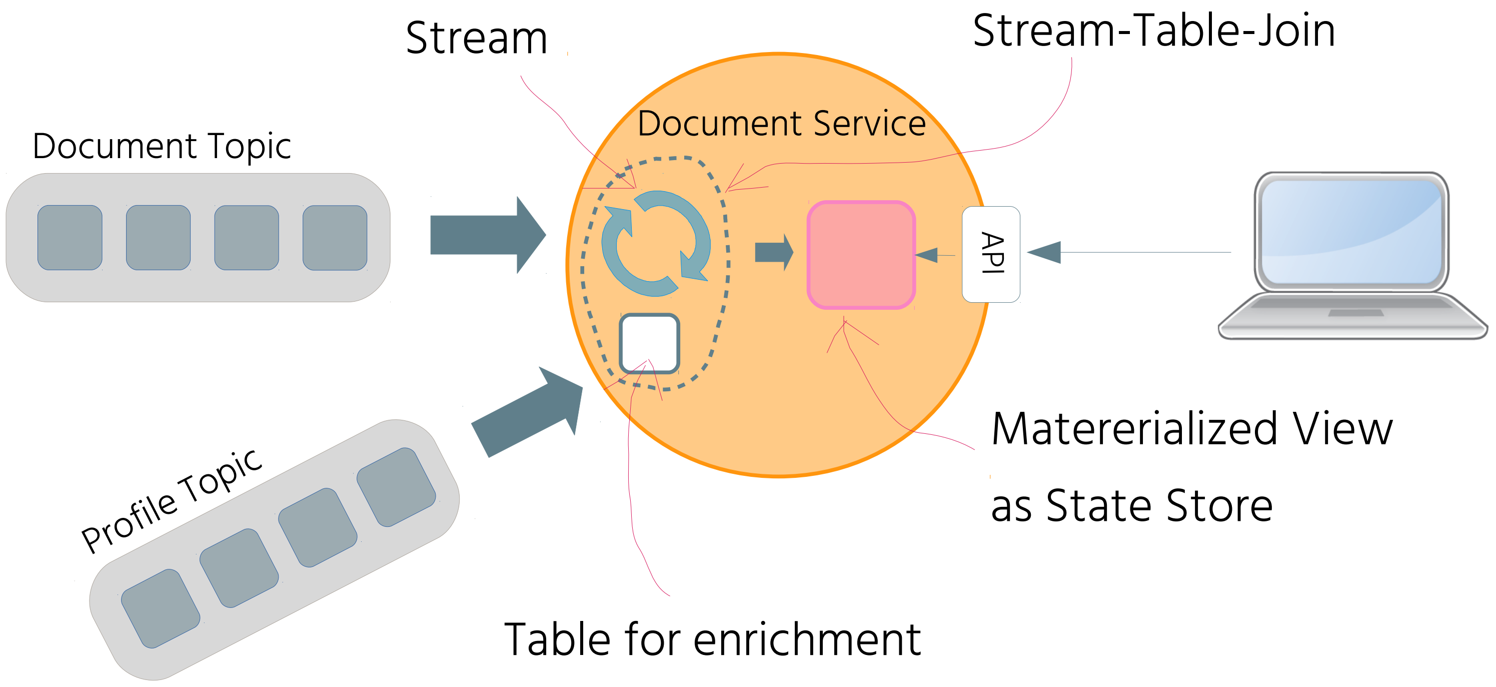
Loads topic on startup
Topic
Service
Running in the process of Microservice
State Store (disk backed)

Streams make data available wherever it's needed

Streams can be:

- Joined
- Filtered
- Grouped
- Aggregated
- etc.

Kafka Streams For Materialized Views



Low Barrier To Entry For New Service

- No separate data storage to set up
- No extra local copies / caches to set up and to keep in sync
- No remote calls
- Materialized View always up to date
- Scalable, fault-tolerant, fast

=> reducing overhead, increasing performance & autonomy

If We Could Start The Journey Again …

- Start with one manageable step at a time
- Take care of Authorization handling early
- Align strategy w/ Microservices goals
- Using Kafka Streams for Materialized Views

THANK YOU!

Susanne Kaiser

@suksr

EX- CTO at Just Software

@JustSocialApps