Tales of a Cloud Migration

Using Apache Kafka

Arnaud Esteve - 2023 - arnaud.esteve@gmail.com

0. Context

Some background

- Personal experience: World-Scale Retailer (Decathlon)
  - Cloud Migration
  - Monolith to micro-services
- Confluent Customers:
  - Retail, Banking, IoT, …
  - Different contexts, timelines

⇒ Merged into a generic “we”

0. Context

Why?

- “We” could have saved so much time
- A lot of common patterns we’ll explore
- What we missed

⇒ “What I wish someone told us”

Agenda

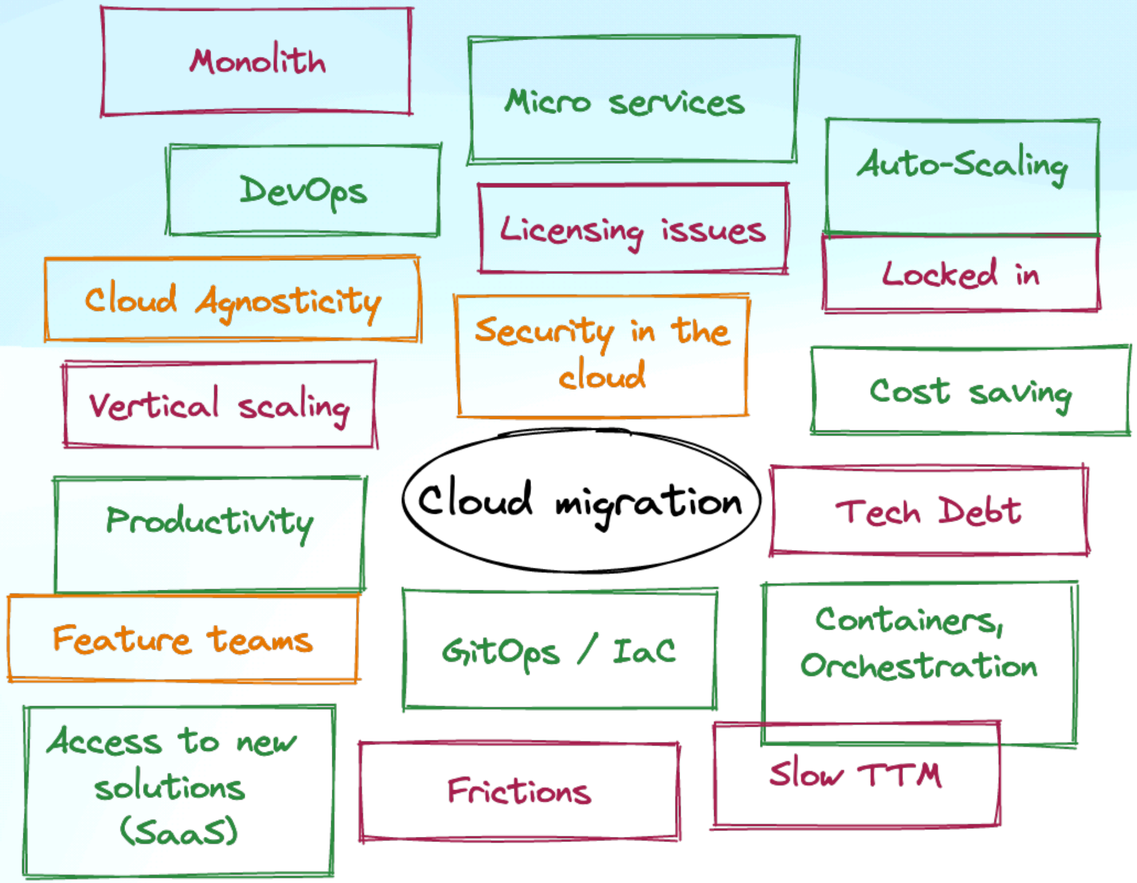
1. 🗺️ Describing the problem(s)
2. 🤔 Tackling Confusion
3. ♟️ The tactics
4. 🧪 The tooling
5. 🔮 A word on what was next

Mapping the problems

The starting line

1. Mapping the problems

Confusion, confusion everywhere



1. Mapping the problems

Analysis Paralysis

*Analysis paralysis (or **paralysis by analysis**) describes an individual or group process where overanalyzing or **overthinking** a situation can cause forward motion or decision-making to become "paralyzed", [...]*

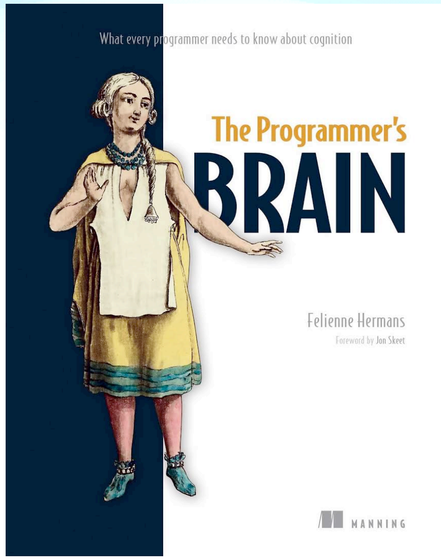
A situation may be deemed too complicated and a decision is never made, or made much too late, due to anxiety that a potentially larger problem may arise.

1. Mapping the problems

Identifying the problem

“It’s too big to reason about”

<= Overloaded working memory



- Solutions?
  - Chunk the problem 😕
  - Filter-out noise 😒

👉 Build simpler mental models: **“Schemata”**

II

PAUSE

Where do we start from?

Tackling Confusion

Defining the schemata


Data Gravity

2. Tackling Confusion

DATA GRAVITY

- Where the **data is**, the **services are**,
  - Where the **services are,** the **value is**

The **better** the data,

The **better** the service,

The **most** value

2. Tackling Confusion

DATA GRAVITY

- We need to **move the data FIRST**
  - Everything should follow organically
  - The sooner the better
  - In a reliable way
  - No need to “make people migrate”
    - Teams should ask for it

KiSS

2. Tackling Confusion

KiSS

- In case of doubt: **Simpler is better** (less is more)
- Simple != Simplistic, but:
  - **Understand > Implement** (DIY for learning purposes only)
  - No **single solution** to rule them all (Monolith, etc.)
  - Instead:
    - **Single standard** / protocol
    - **Single approach** / way of thinking


Continuously
Better

2. Tackling Confusion

Continuously Better

- It is going to be **long**, accept it
  - No “Big Bang”
    - Unrealistic
    - Too **risky**
    - + Tunnelling effect
- But **BETTER**:
  - No 1-1 migration => Find **value**


Opportunistic

2. Tackling Confusion

Opportunistic

- No matter what: it **must bring value**
  - Avoid the “tech for techies”
- Exercises:
  - Police lineup (usual suspect)
  - Or the fridge analogy (eat what you have)
- Embrace opportunities
  - You know them, you’ll find them along the way


Dogfooding

2. Tackling Confusion

Dogfooding

- “Are we doing this right”

  ➡️ Most FAQ ➡️ Mental Charge

- A way to build convictions:
  - “Does this work for us?”
  - “Do we trust ourselves as a provider?”
- Quality matters:
  - Build exigence
  - (Internal+External) Audit


Decisions
must NOT matter

2. Tackling Confusion

Decisions must not matter

- Standards, standards, standards
- OSS
- Documentation (transparent) > implementation
  - ADRs
  - …
- Favour *reversible* / *agnostic* when possible

=> piggy-backed in KiSS

Our schemata

2. Tackling Confusion

Our pillars

- Move the data (Data Gravity)
- Only the reliable part of it, cleaned (Better)
- Using standards, in a simple way (KiSS)
- Progressively (Continuous)
- not sure where we'll store it (Decisions / Opportunities)
- but we'll use it ourselves (Dogfooding)

2. Tackling Confusion

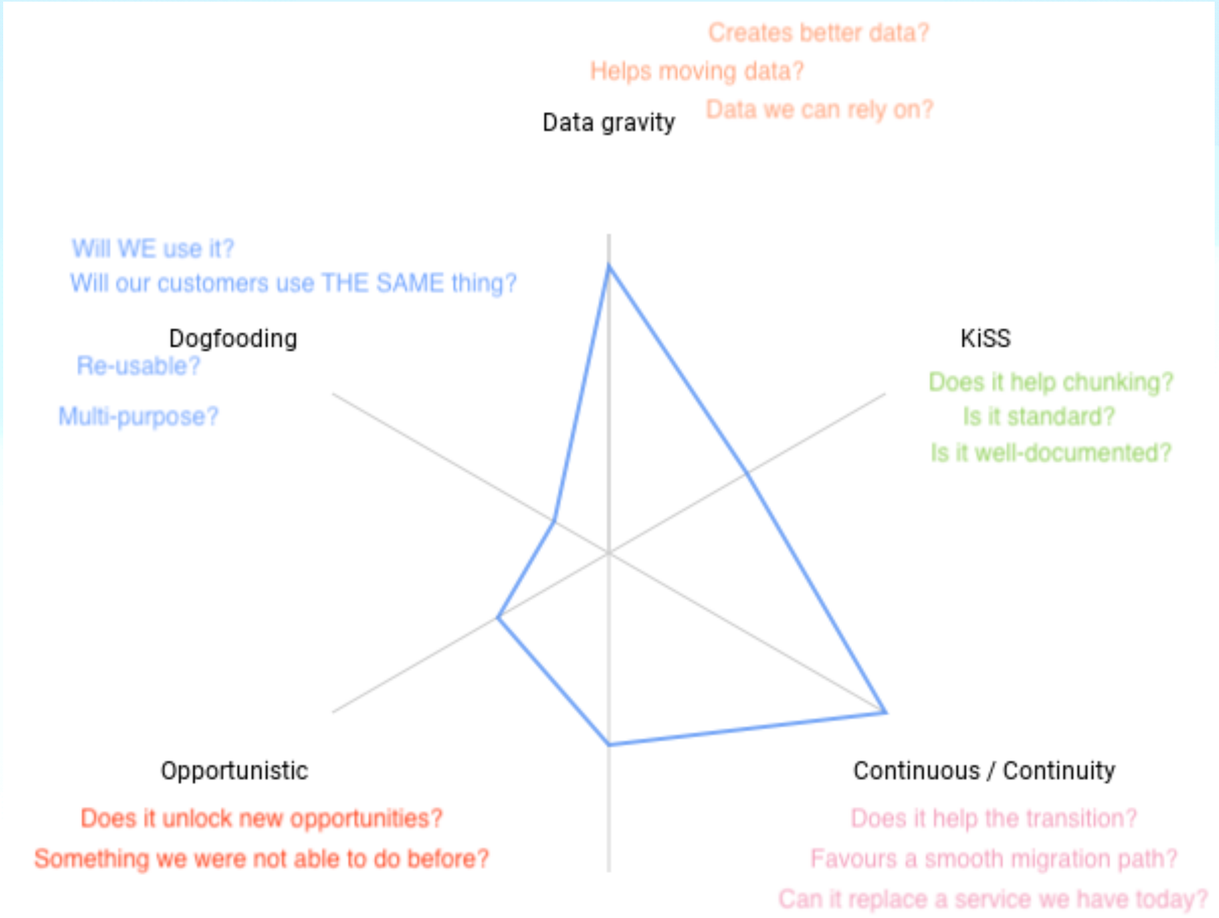
Build your own decision framework

- Your own schemata
- Whatever helps your teams

Ready to screen the migration candidates?

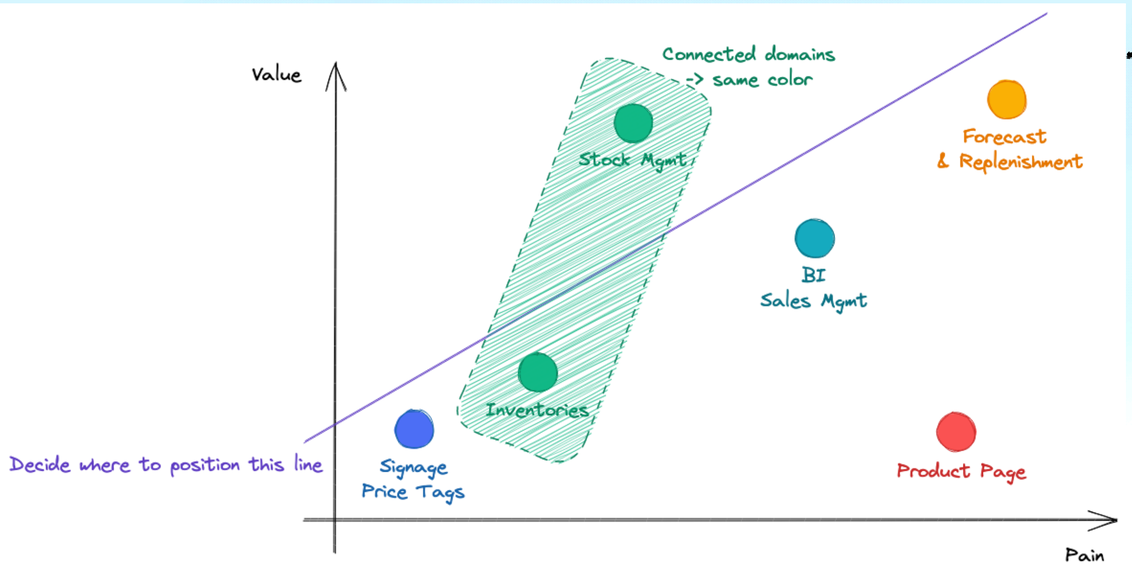
2. Tackling Confusion

WHAT **WE** TRIED



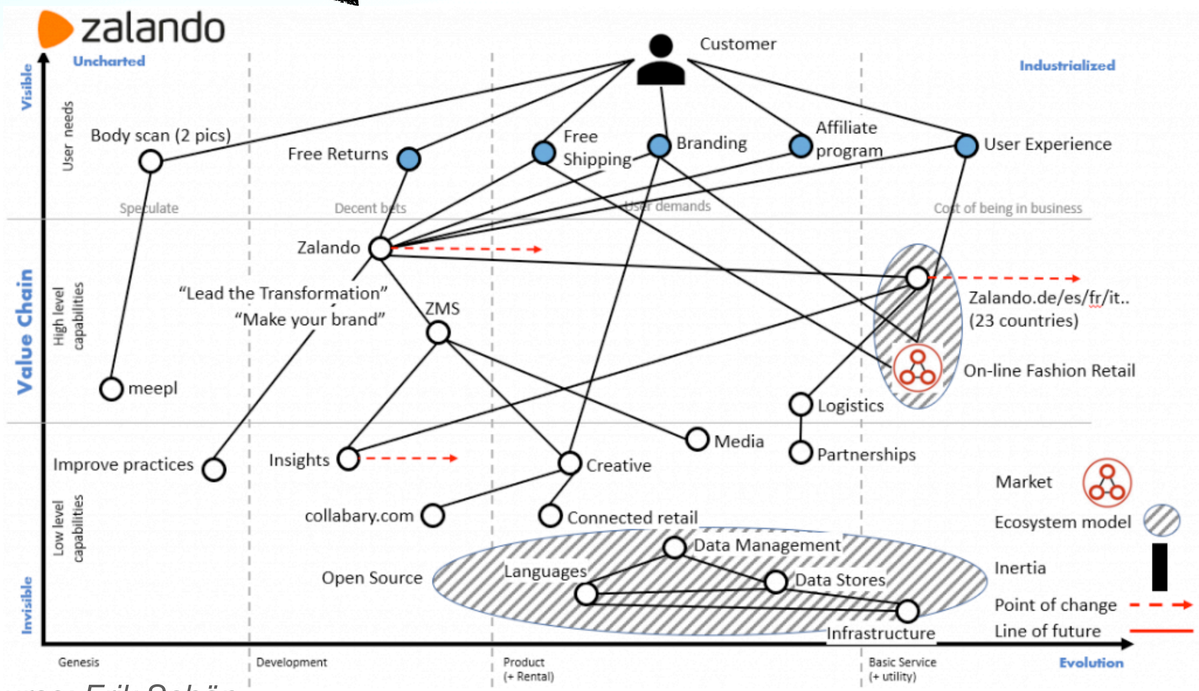
2. Tackling Confusion

WHAT WORKS FOR **YOU**



From simple
(home made domain map)

To formal
(Wardley Map)



Source: Erik Schön

II

PAUSE

Where do we start from?

Move data,
In the simplest way,
Continuously, better,
Bringing value along the way,
For us first,
=> *Domain*

What do we start with?

Event Streaming

The tactics

3. Event Streaming

Move what exactly?

- Model#1: “Move the data”

  ➡️ Which data?

- Identified domain, but broadly

  eg: Stocks, what stocks?

3. Event Streaming

Streaming what?

Stock Management example

Movement

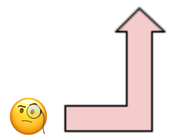
```
product: 123
store: 456
date: 2022-10-12
quantity: -1
kind: sales
```

Few usage (mostly internal)
Displayed in the UI
Flowed to supply teams
No real time use

Picture

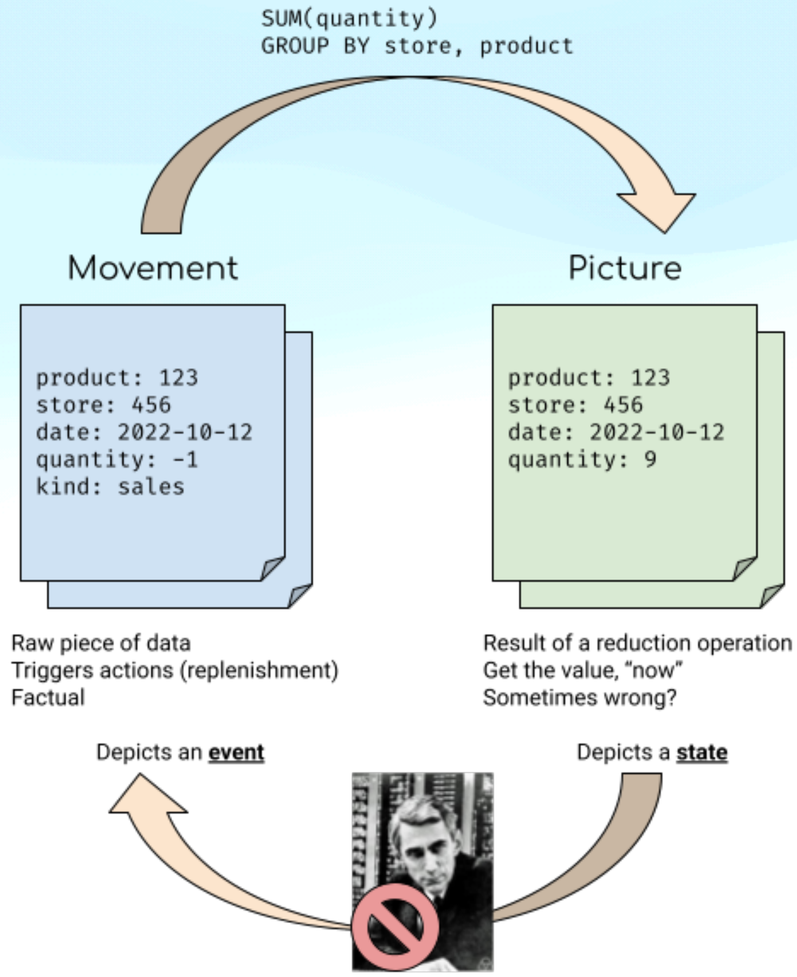
```
product: 123
store: 456
date: 2022-10-12
quantity: 9
```

Most used API in the company
Everyone needs it (e-comm, etc)
Highly critical
High source of problems



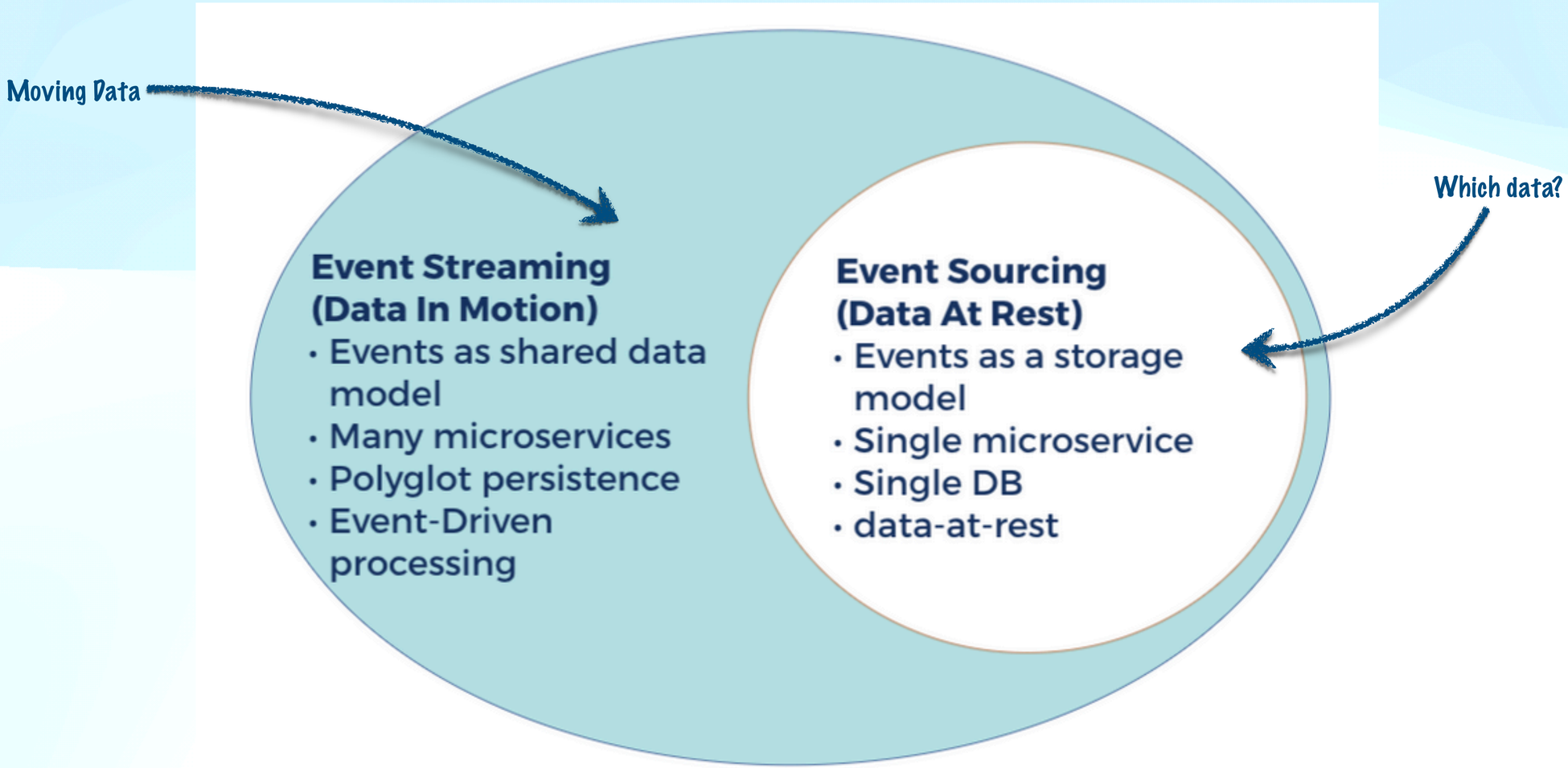
3. Event Streaming

Getting to the source of truth



3. Event Streaming

Continuous = Streaming / What = Event



3. Event Streaming

A common pattern, across common industries

- Events are everywhere:
  - Financial transactions
  - IoT sensors
  - Traceability
  - etc. => Choose your **DOMAIN EVENTS**
    - (and remember Claude)

3. Event Streaming

From **means** to **end**

- Teams loved it
  - Build a local cache (**Continuously**, **Better**)
  - Notification systems (**Opportunities**)
  - “Let us deal with this” == “Send us the event” (Chunking => KiSS)
  - Become a generic event provider => simpler scope (KiSS)
- Customers loved it (**Dogfooding**)
  - ~~Arbitrary delays~~ => Events in real time
  - “Tell me what’s happening” / “Notify me”

II

PAUSE

Where do we start from?

Move data,
In the simplest way,
Continuously, better,
Bringing value along the way,
For us first,
=> A domain

What do we start with?

Streaming Events,
Shaping a ***source of truth***

How do we do that?

Picking the right Tech

Apache Kafka

4. Picking the right tech

Does Kafka help streaming events? 🙄

- Events can be represented as logs
  - Factual <=> Immutable
  - Chronological <=> Ordered
    - Index <=> Offset
    - Bookmark <=> Consumer offset

4. Picking the right tech

Key features in the context of a migration

- Replay-ability is key (Better, Safer)
- AK (as a protocol) is a standard (Decisions -> KiSS)
- Suits other patterns as well: CQRS, ... (KiSS)
- Vibrant ecosystem (Data Gravity, Better, Decisions -> KiSS)
- AK is not simple, but well documented (Decisions -> KiSS)
- Stream Processing (Better, Opportunities)
- Advanced Governance features (Better, Safer)

4. Picking the right tech

Evaluating Kafka

👍 **Industry Support**:
- Good Client libs
- Same solution for many cases

👍/👎 **External use**
- Direct customer integration?
- “Protocol Bridging”
- => connectors help, http-sink

👎 **Defining standards**
- “Pick the right auth”
- Conventions, etc.

👎 **Self-Managing**
- Hard and costly

👍 **Moves Data**
- Fast
- Scalable
- Resilient
- Built for this

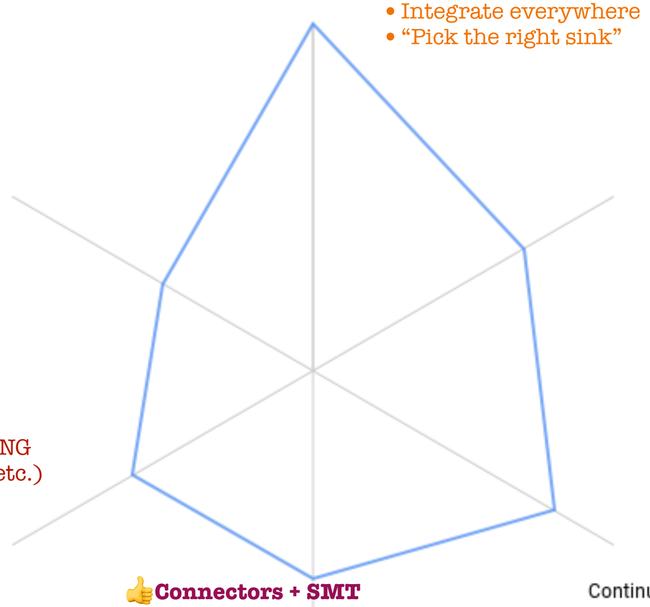
👍 **Connectors**
- Help extracting the data
- Integrate everywhere
- “Pick the right sink”

👍**Standard protocol**
- Widely adopted
- Widely documented

👍**Literature**
- Blogs, videos, trainings, certifications
- Events, Meet-ups
- See “patterns”

👎 **Entry-Barrier**
- Training, training, training
- “Don’t you have an API instead?”

👎**Hard work**
- Extracting from the source
- Risky & costly to lose
- SPoF??



👍 **Ecosystem**
- Connectors for everything
- Event Streaming -> can rebuild ANYTHING
- Integrates everywhere (Analytics, ML, etc.)

👍 **Multi-Purpose**
- (Audit) Logs
- Metrics
- “World is event-driven”

👍 **Stream Processing**
- Solves a lot of problem, easily
- Wide range of solutions

👎 **Stream Processing**
- Entry-barrier + Limitations

👍**Connectors + SMT**
- Filter, transform => better

👍**Strong Decoupling**
- Separation of concerns
- Isolation

👍 **Well-documented Patterns**
- Dead Letter Stream
- Event Gateway
- Outbox pattern, ...

👎 **Governance & Standards**
- Lack of opinionated view
- Governance topics
- Event formats, enveloppe, ...
- Encryption, PII, ACLs

👍 **Distributed, reliable**
- Stable connectors
- Optimised
- No cron, no batch
- Simply flowing, streaming

👍 **Decoupled**
- Producer / Consumers
- Consumer Groups scalability
- Different languages, techs

👎 **Hard work required**
- Extracting from the source
- Upstream vs. Downstream

II

PAUSE

Where do we start from?

Move data,
In the simplest way,
Continuously, better,
Bringing value along the way,
For us first,
=> A domain

What do we start with?

Streaming Events,
Shaping a ***source of truth***
That can be used by many

How do we do that?

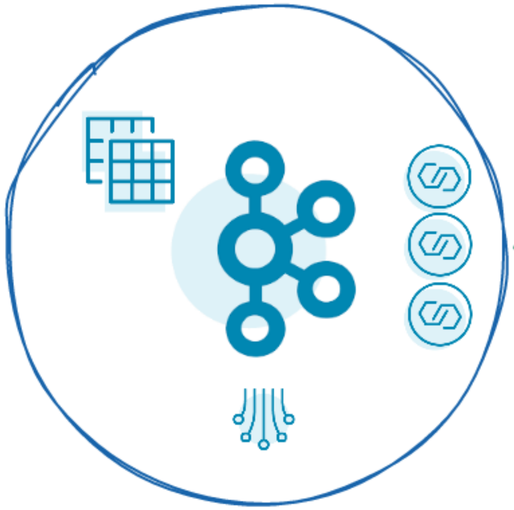
Using Apache Kafka
AND ITS ECOSYSTEM

Lessons Learned

And what's next?

5. Lessons Learned

👍👎 A divided world



Focus efforts upstream
KiSS is strategic here
“Temporary” is acceptable

Move the data, at **all** costs

Low effort (connectors, etc.)
“Dogfooding” is strategic here
“Temporary” is unacceptable

Use the data, at **low** cost

5. Lessons Learned

👍 Reduced Friction

- An answer to a lot of problems
  - compacted topics, stream processing, connectors, etc.
- Isolation of concerns
- Examples:
  - “Can you change your API?” <= most common question ever
    - “What about you consume the data?”

5. Lessons Learned

👎 It’s not that easy

- Lots of standards, lack of dogma
  - AuthN / AuthZ
  - Hard to make decisions

- Traps (See “AK: Tips and Tricks I wish I had known before”)
- Self-Maintaining Distributed systems is hard

5. Lessons Learned

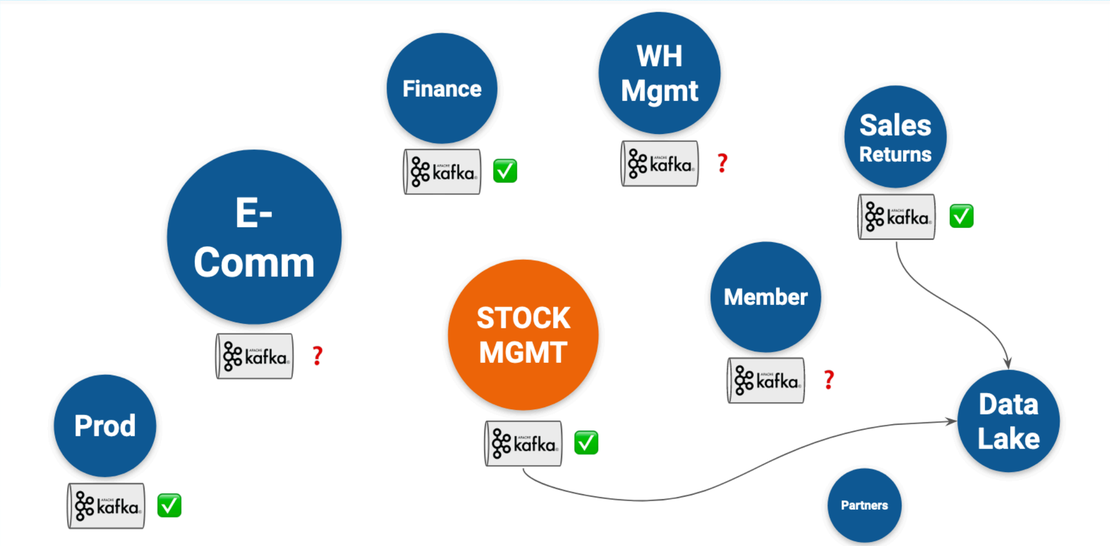
👎 Many hard decisions at scale

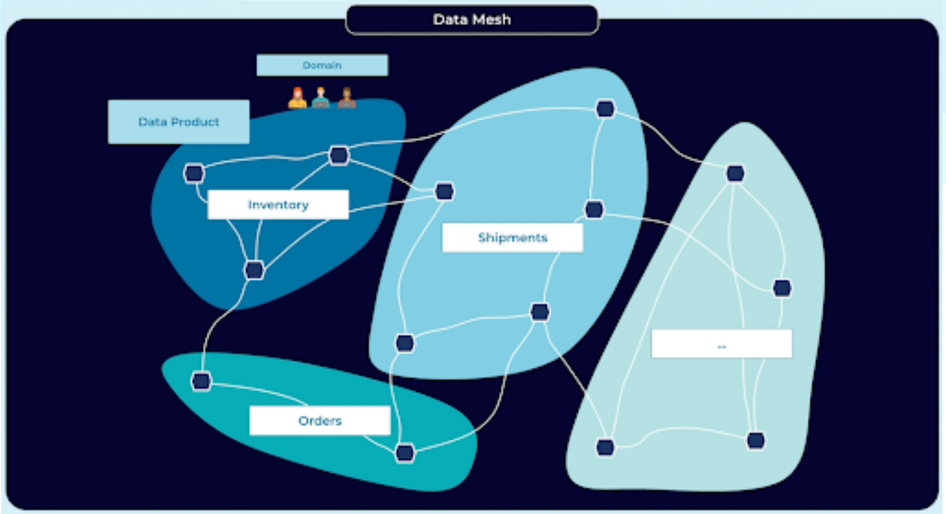
- Internal Standards
  - Naming conventions,
  - Data format / schema, …

- Internal Guidance / Training / Knowledge
  - Dimensioning / Sizing (partition count, clusters)
  - **Document** the traps faced, errors made

5. Lessons Learned

👍 A Spreading Pattern 🔨

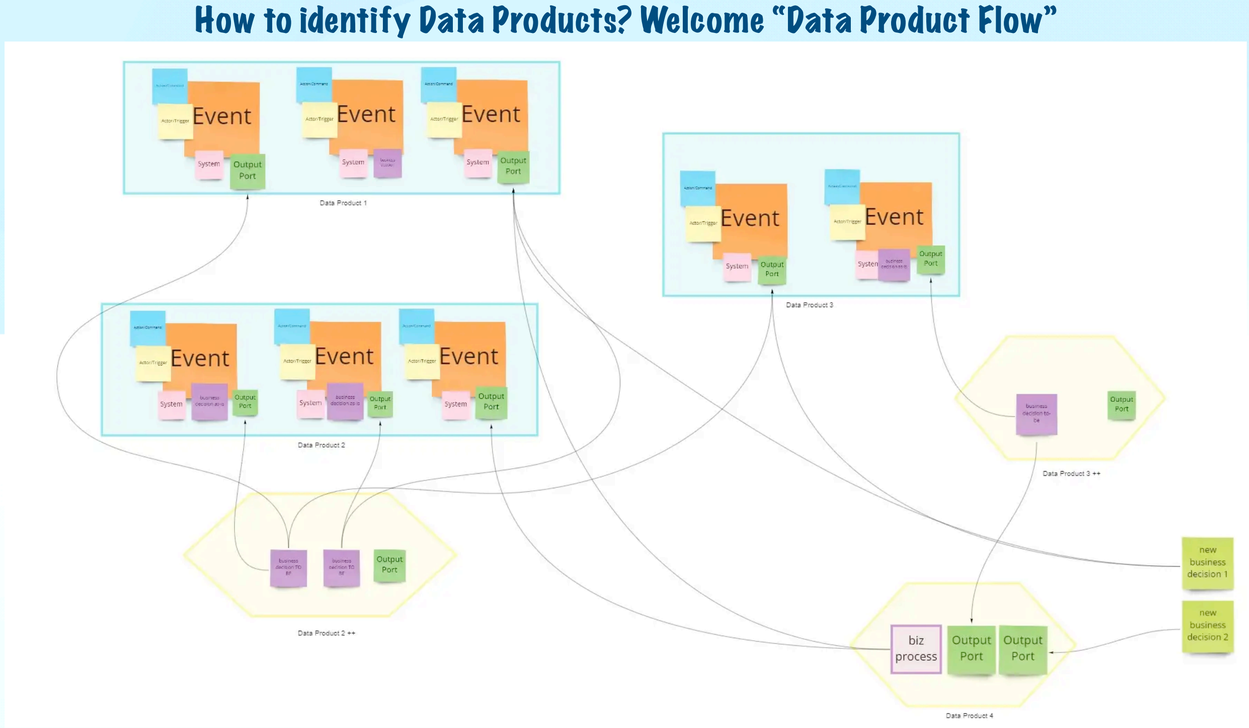




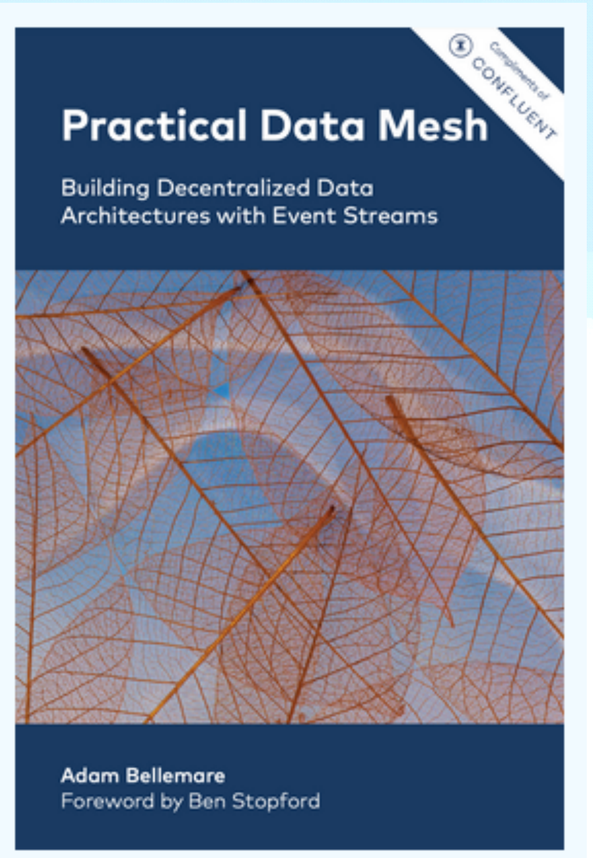
Source: https://developer.confluent.io/learn-kafka/data-mesh/

5. Lessons Learned

(Streaming) Data Products



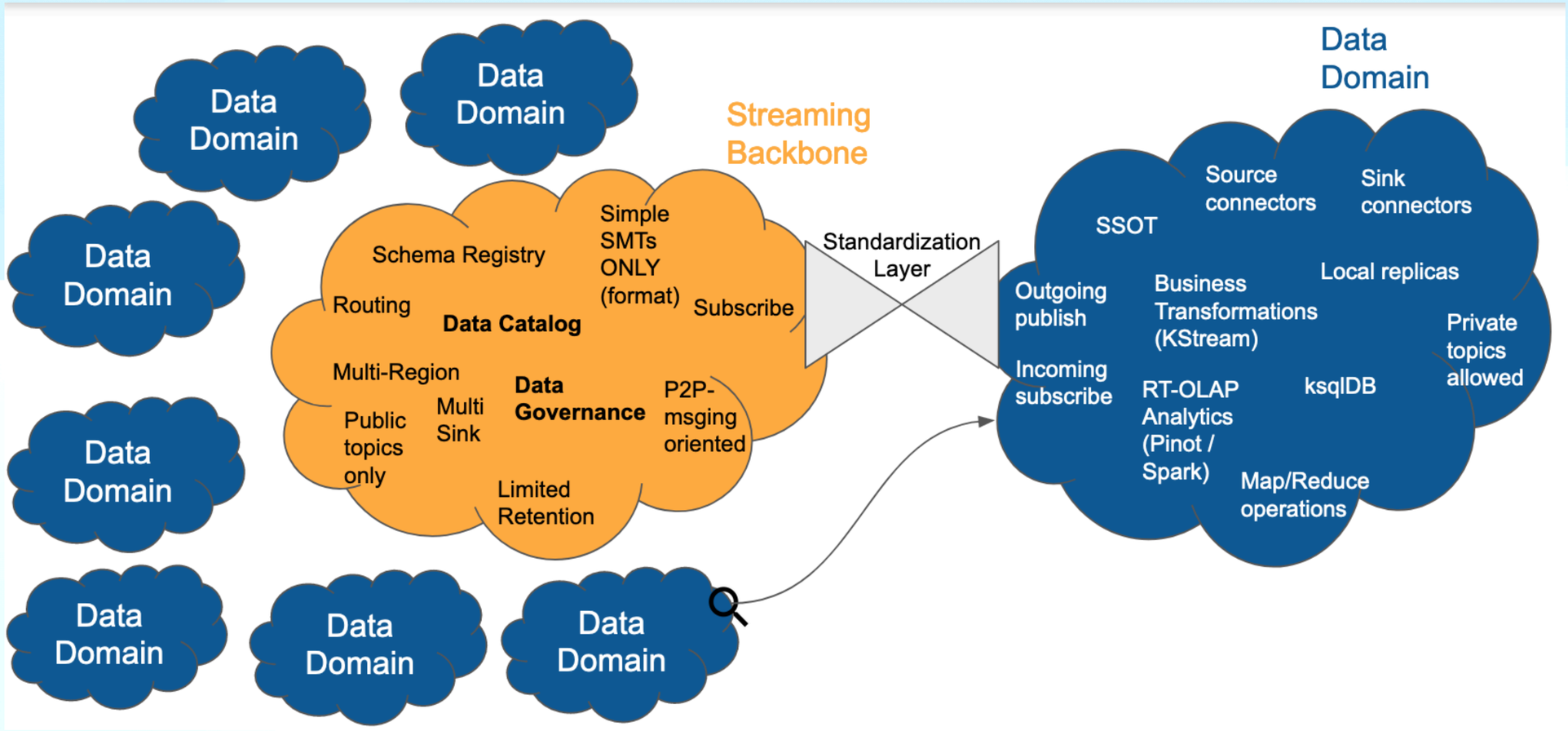
https://medium.com/agile-lab-engineering/how-to-identify-data-products-welcome-data-product-flow-76d7d85d23af



https://developer.confluent.io/learn/data-mesh/

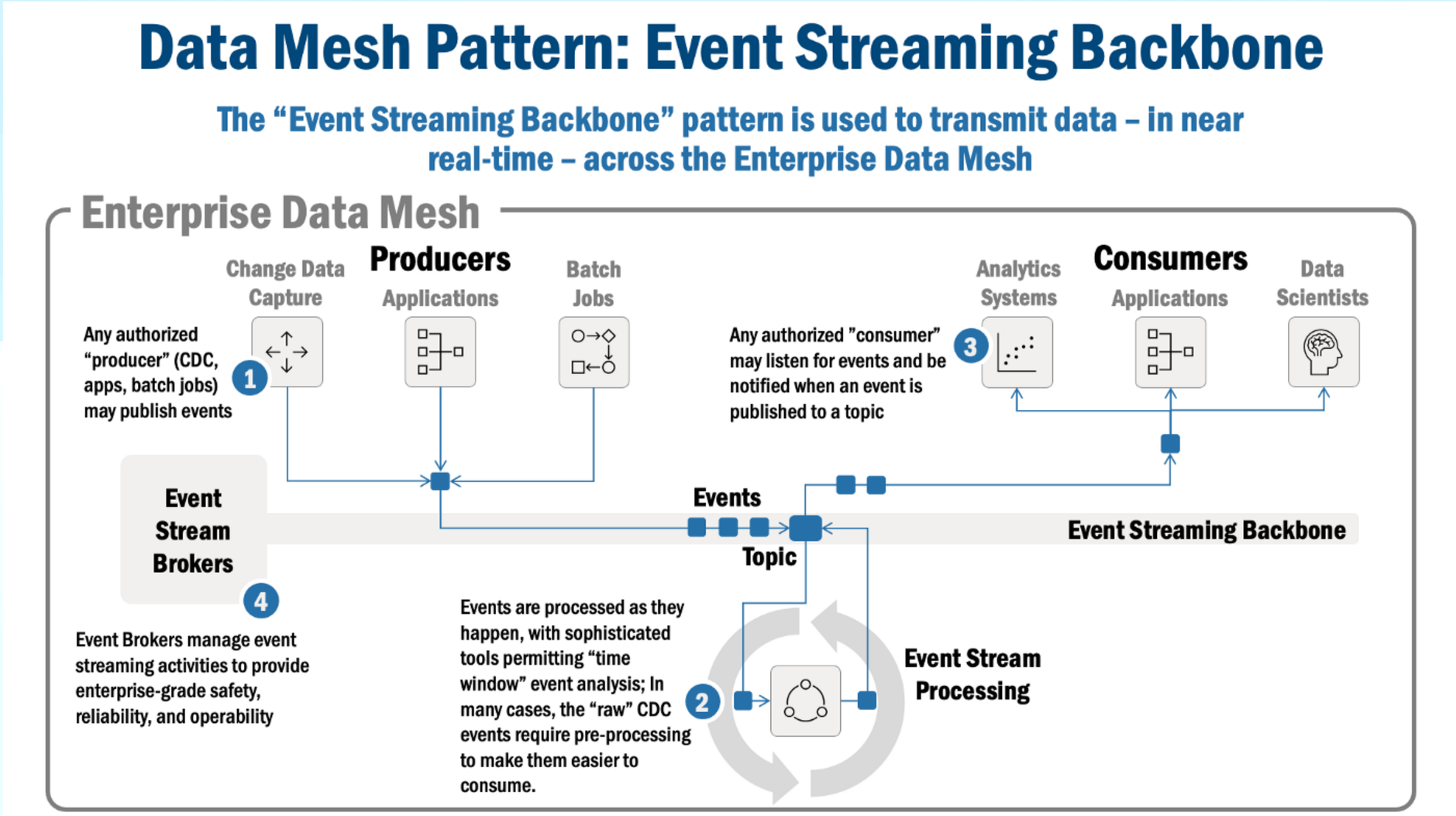
5. Lessons Learned

👍 A Spreading Pattern 🔨



5. Lessons Learned

A Spreading Pattern 🔨 - (Streaming) Data Mesh



https://towardsdatascience.com/data-mesh-pattern-deep-dive-event-streaming-backbone-99a5bb2a7cbf

Where do we start from?

Move data,
In the simplest way,
Continuously, better,
Bringing value along the way,
For us first,
=> A Domain

What do we start with?

***Events**,*
*Shaping a **source of truth***

How do we do that?

Using Apache Kafka
AND ITS ECOSYSTEM

Where does it lead us?

=> Stream Processing
=> Governance
*=> Data (Streaming) **Products***
*<= Data (Streaming) **Mesh***

Thank YOU!

arnaud.esteve@gmail.com

Also: “Apache Kafka: Tips & Tricks I wish I’d known before"