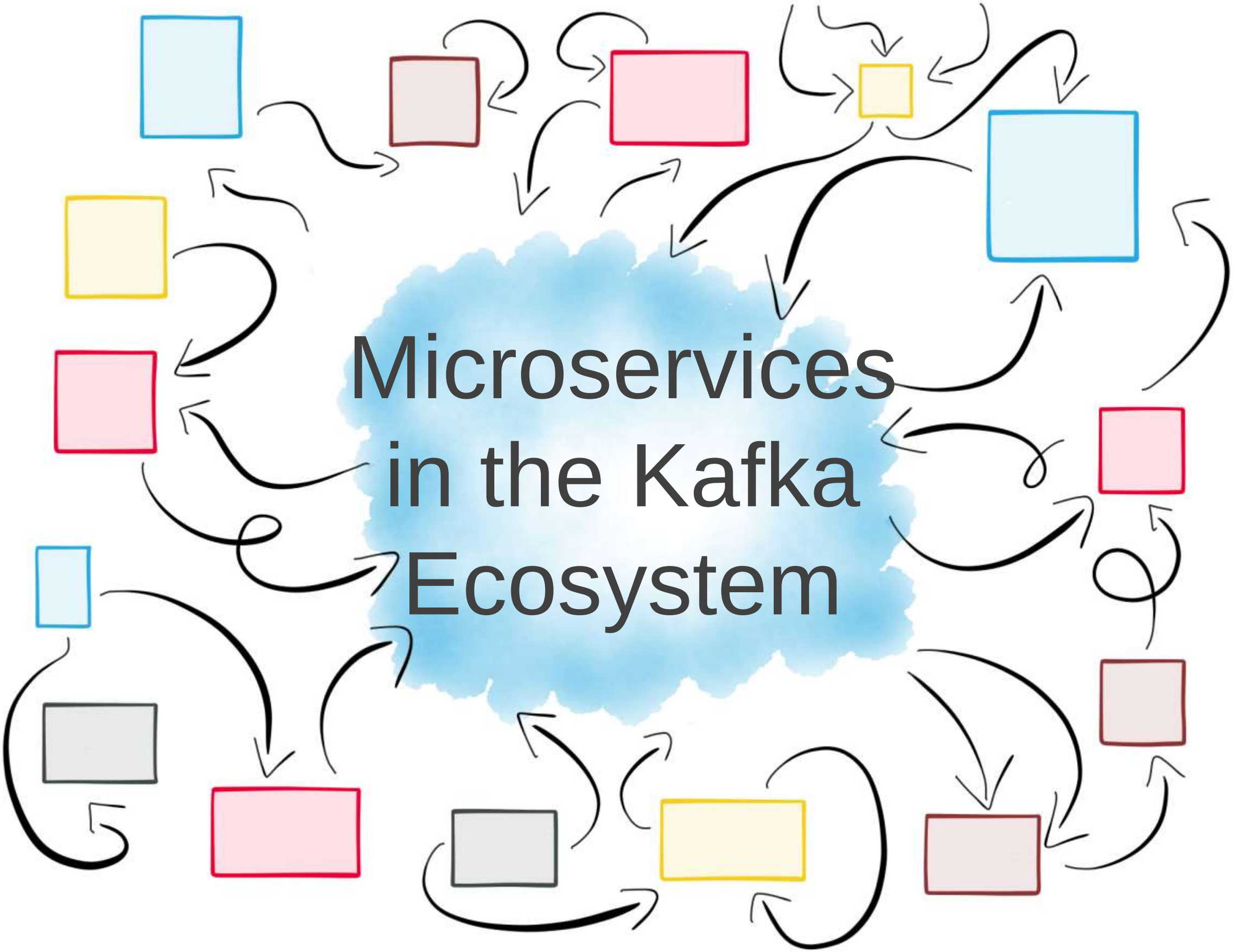
Ten Principals for Effective Event-Driven Microservices

Ben Stopford
(Office of the CTO)
@benstopford

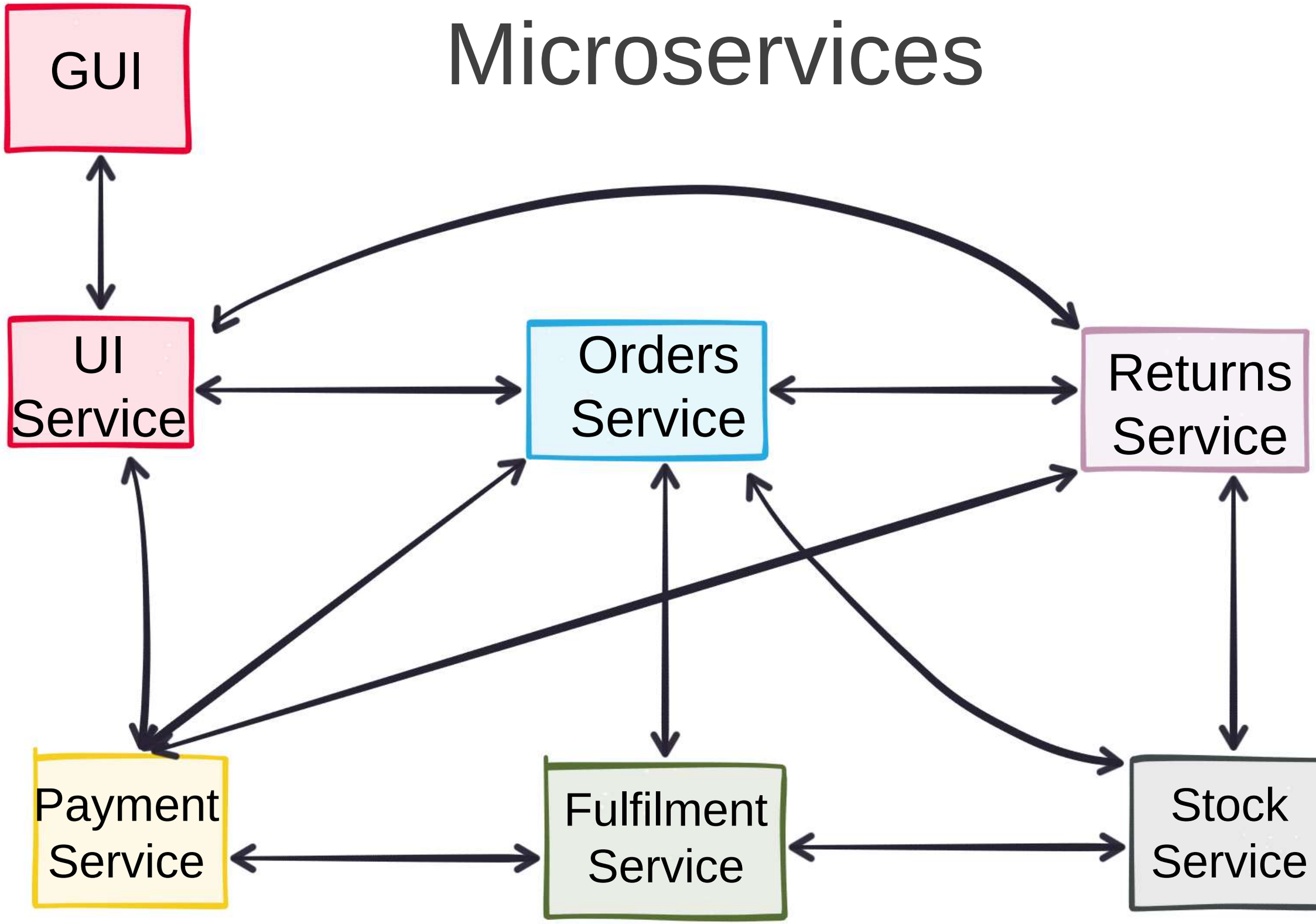
What we'll cover

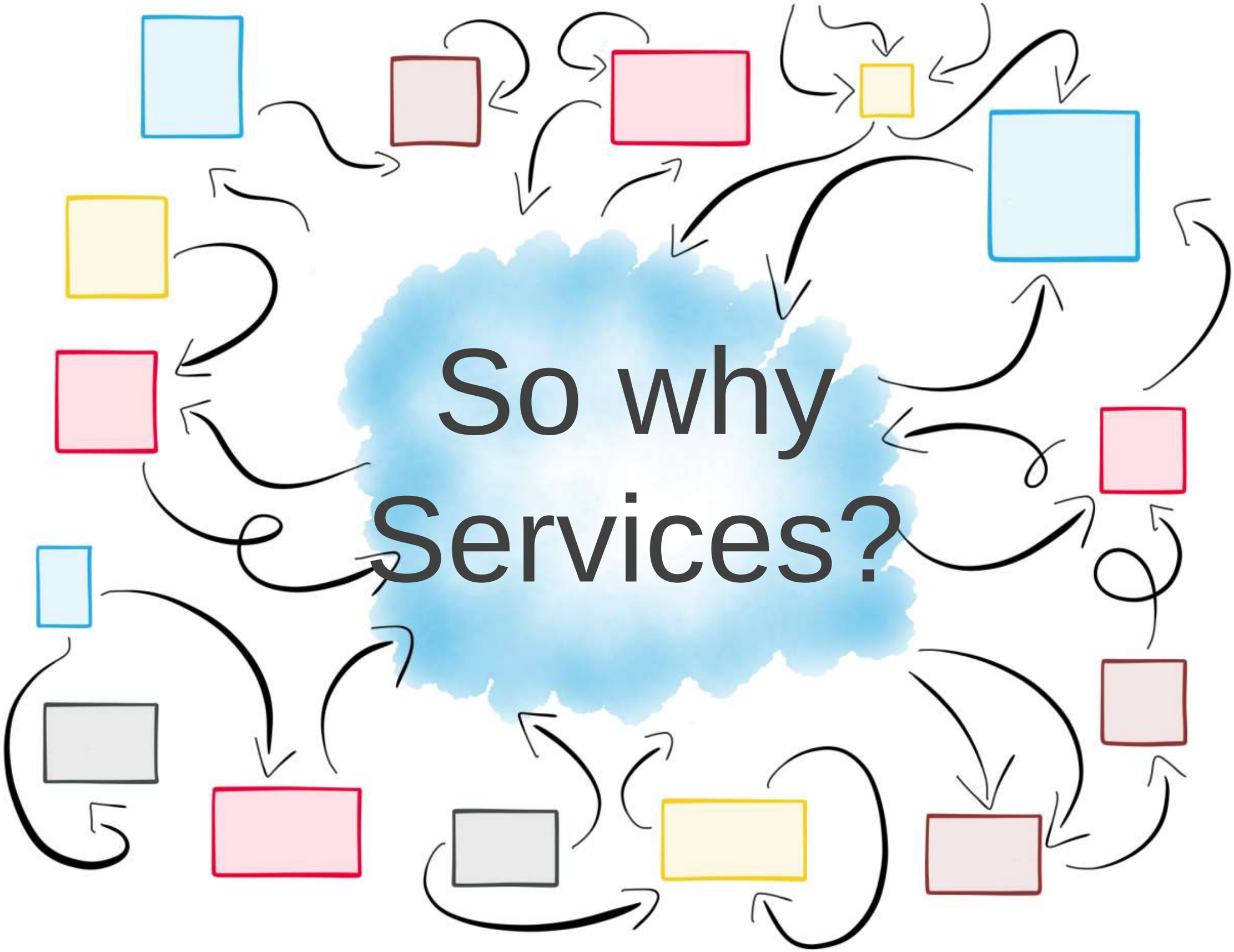
- Event Driven Microservices
- The toolset: Kafka, KStreams, Connect
- 10 Principals for Streaming Services



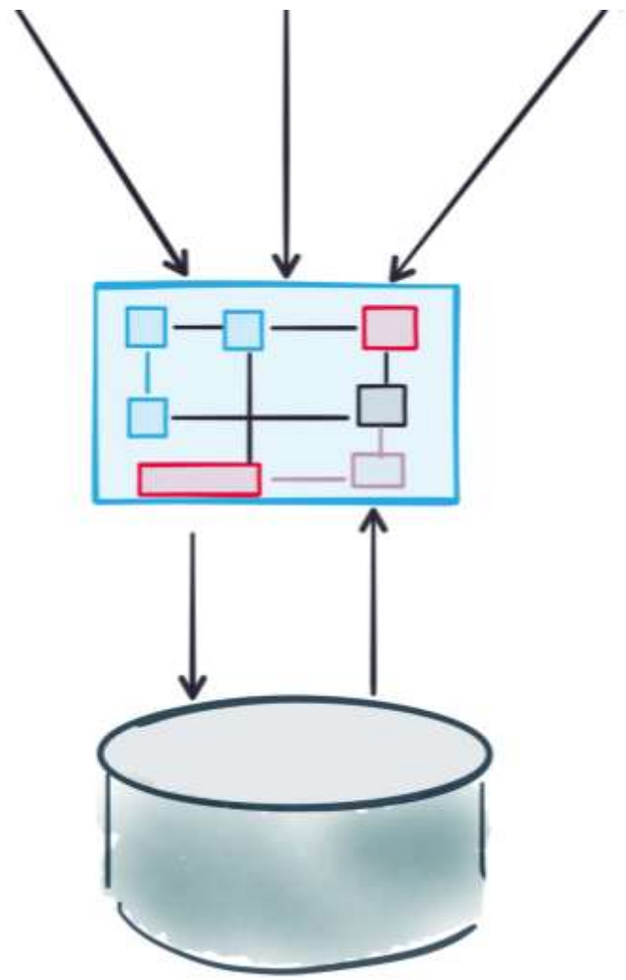
Microservices in the Kafka Ecosystem

Microservices



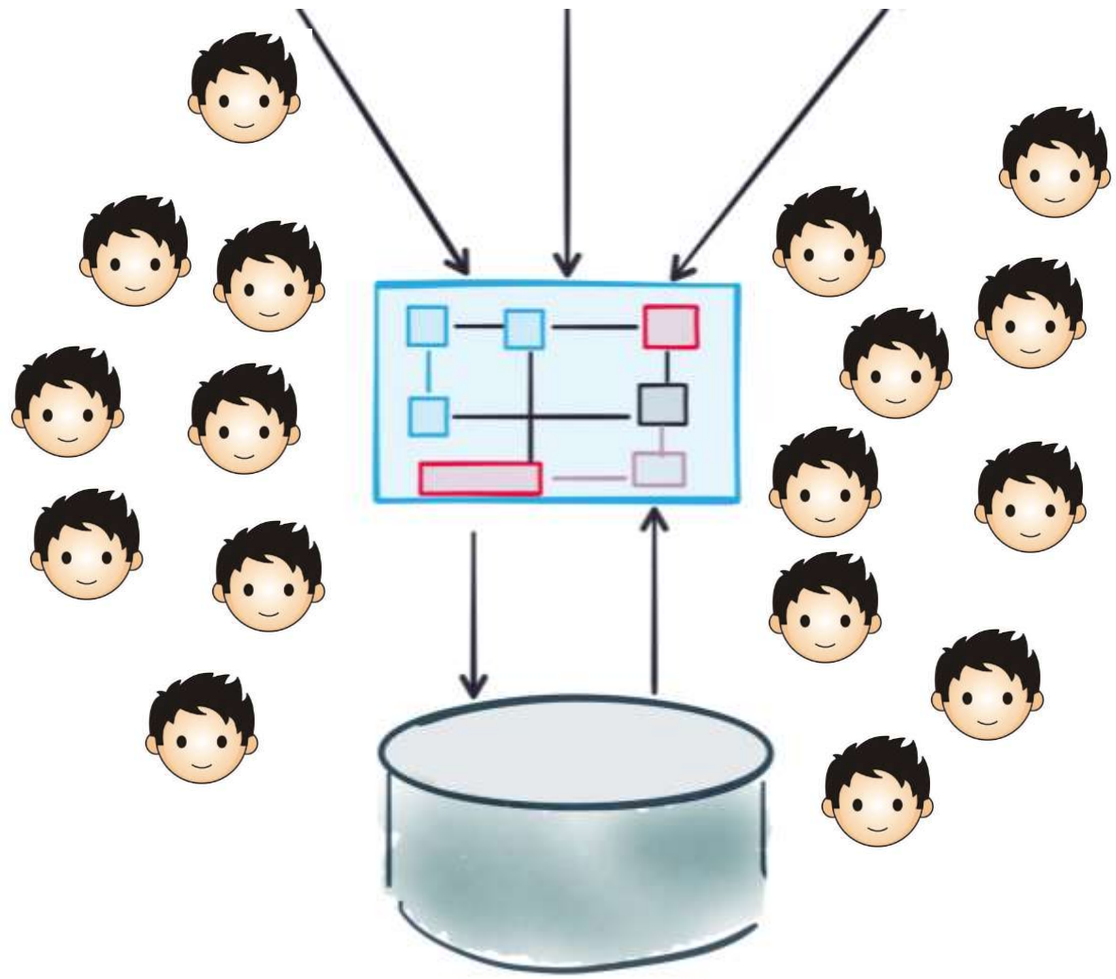

So why
Services?

The Monolith



Can we do reuse, encapsulation?

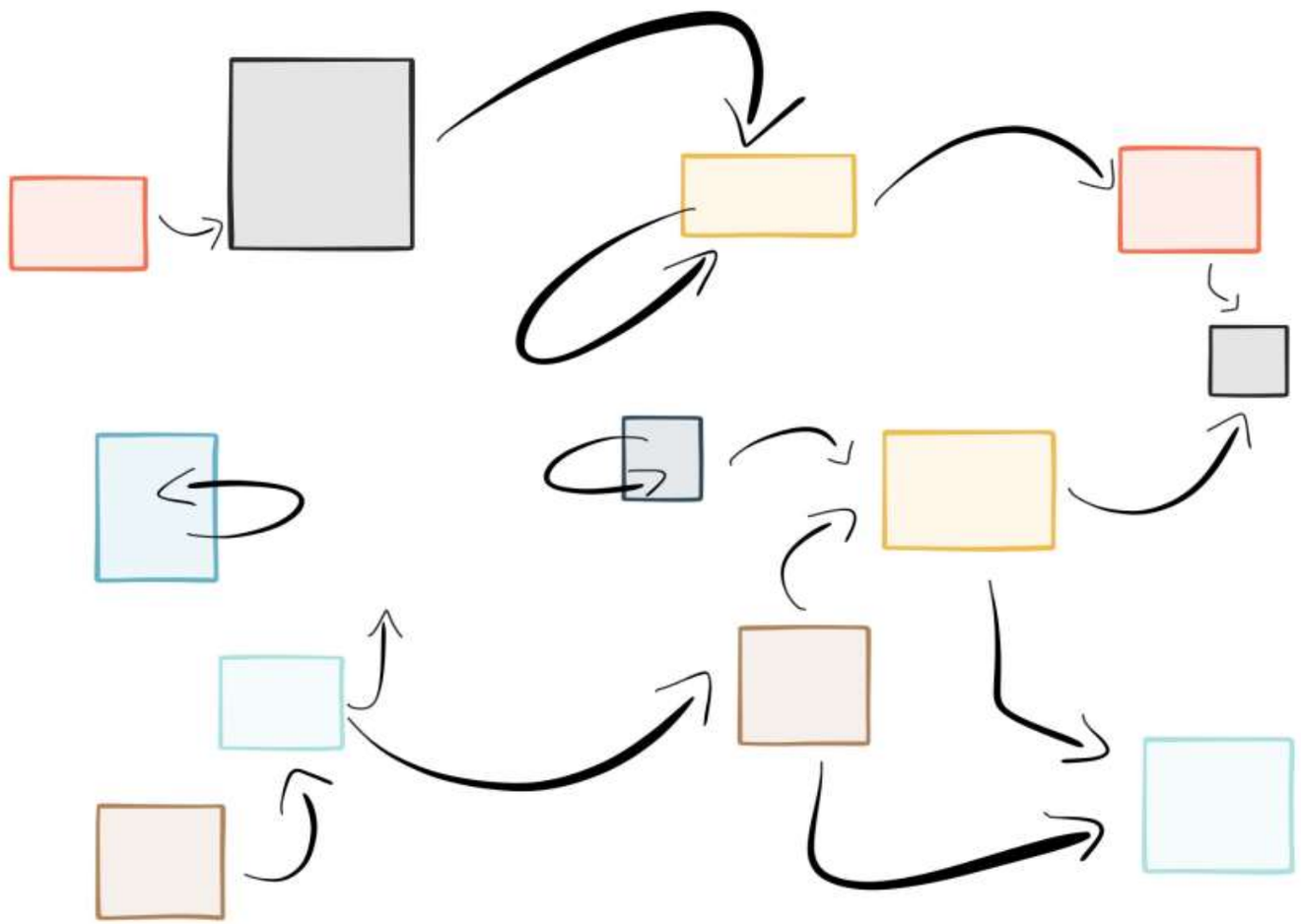
The Monolith



What happens when we grow?

Companies are inevitably a collection of applications

They must work together to some degree

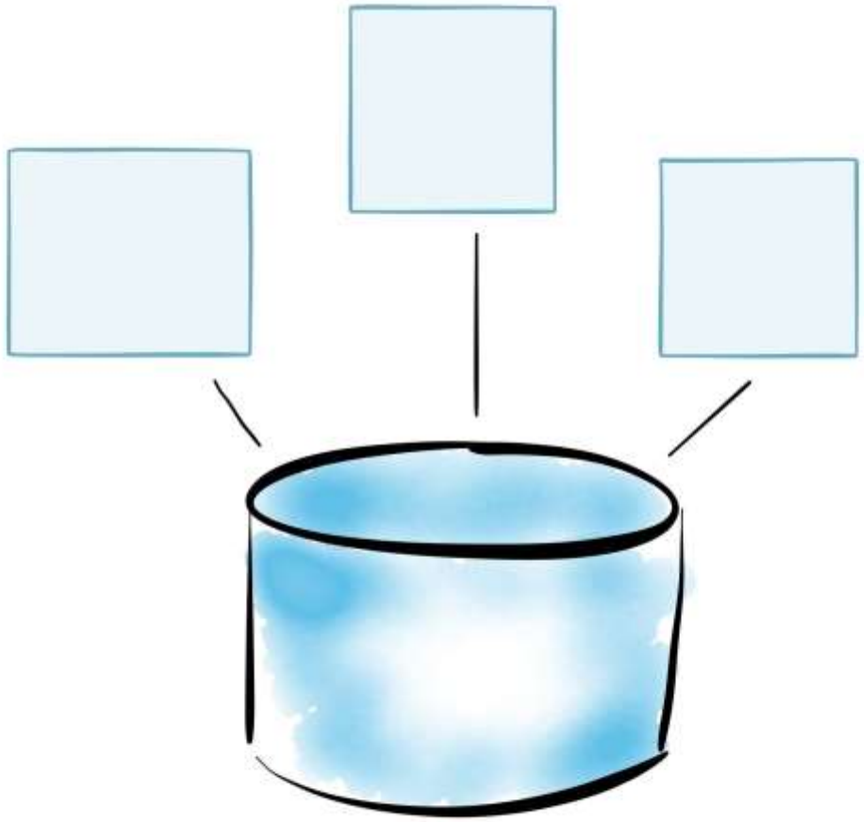


Inverse Conway Maneuver

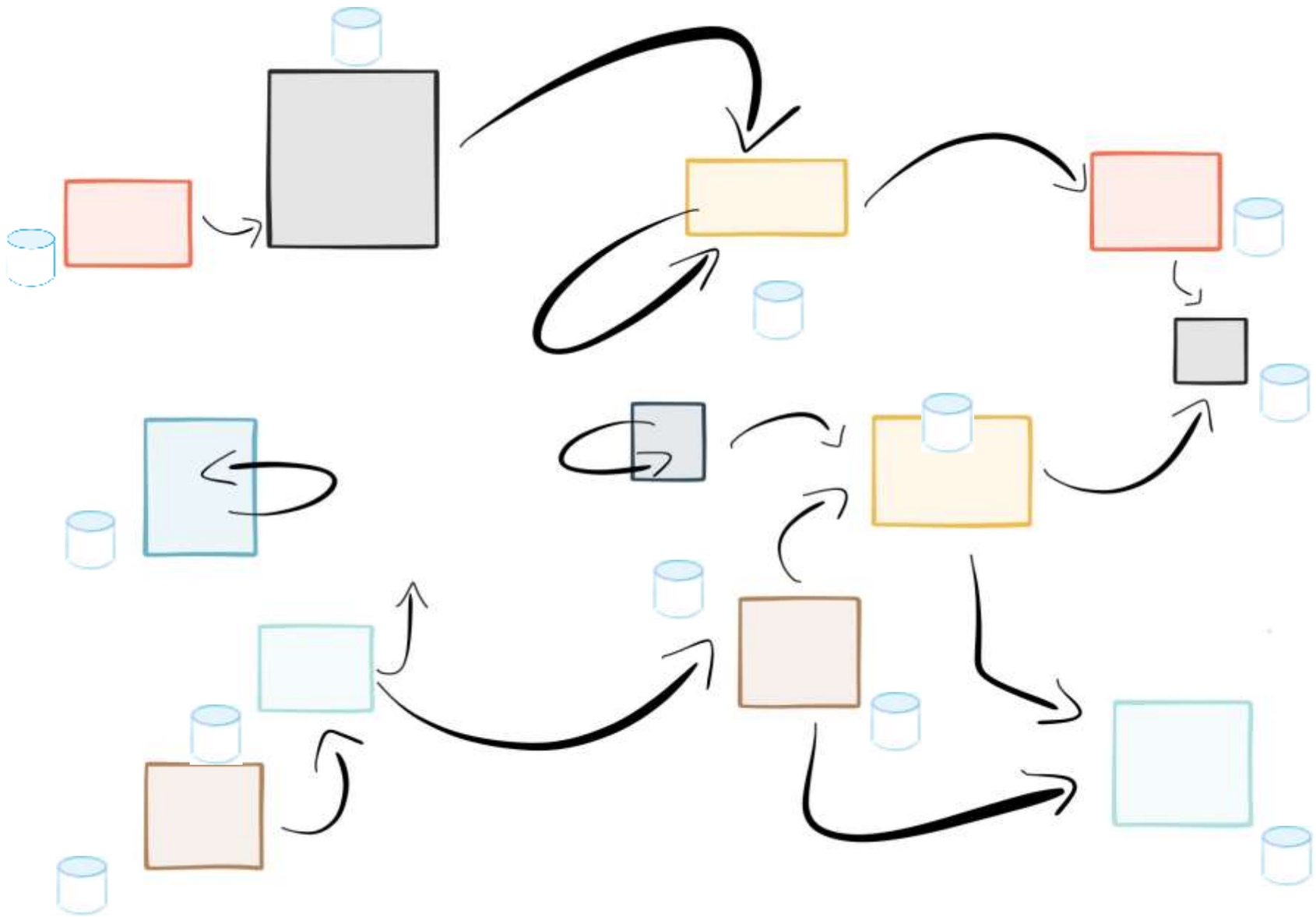
Org Structure

The 'Inverse Conway Maneuver' recommends evolving your team and organizational structure to promote your desired architecture.

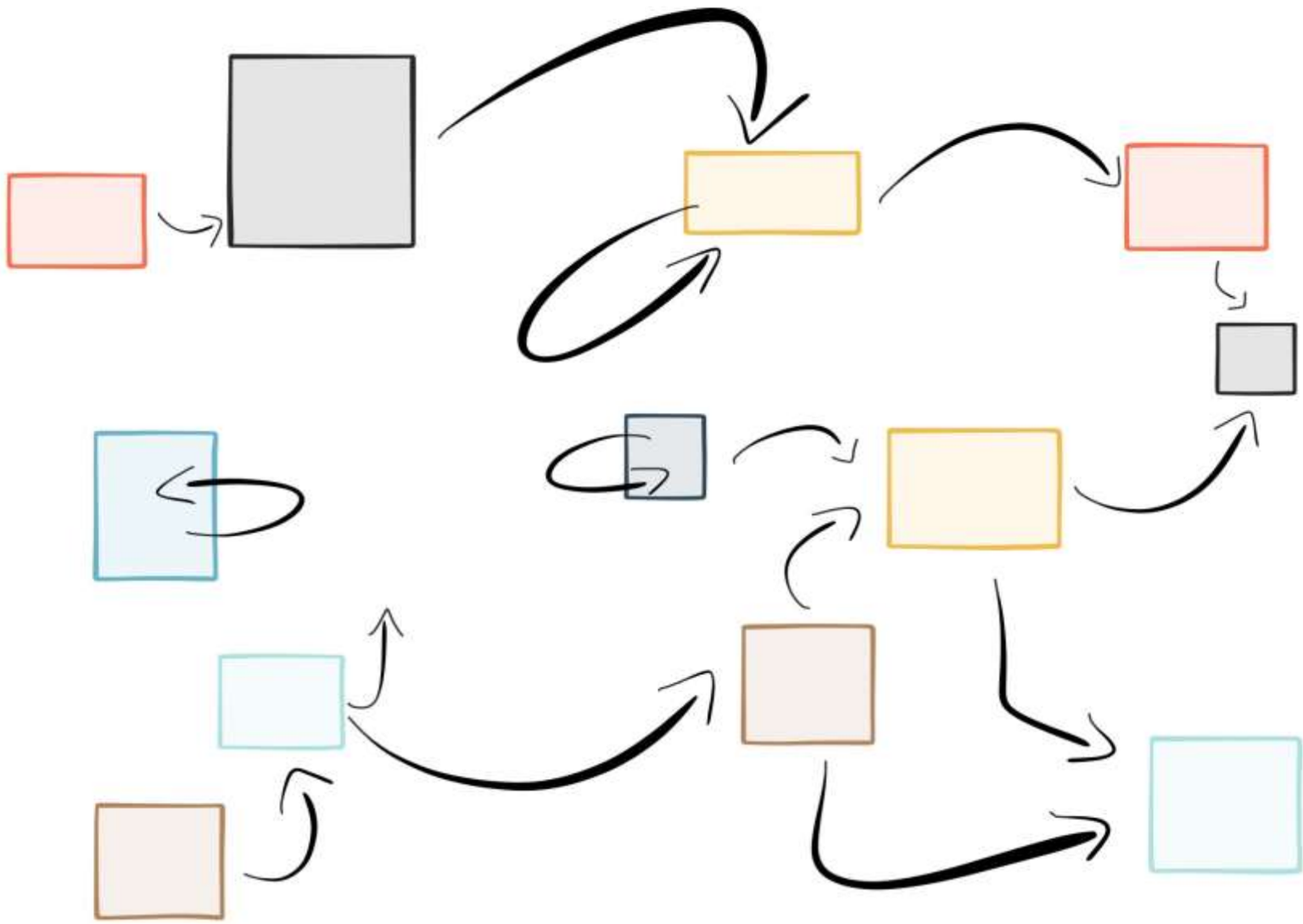
Eschew shared, mutable state



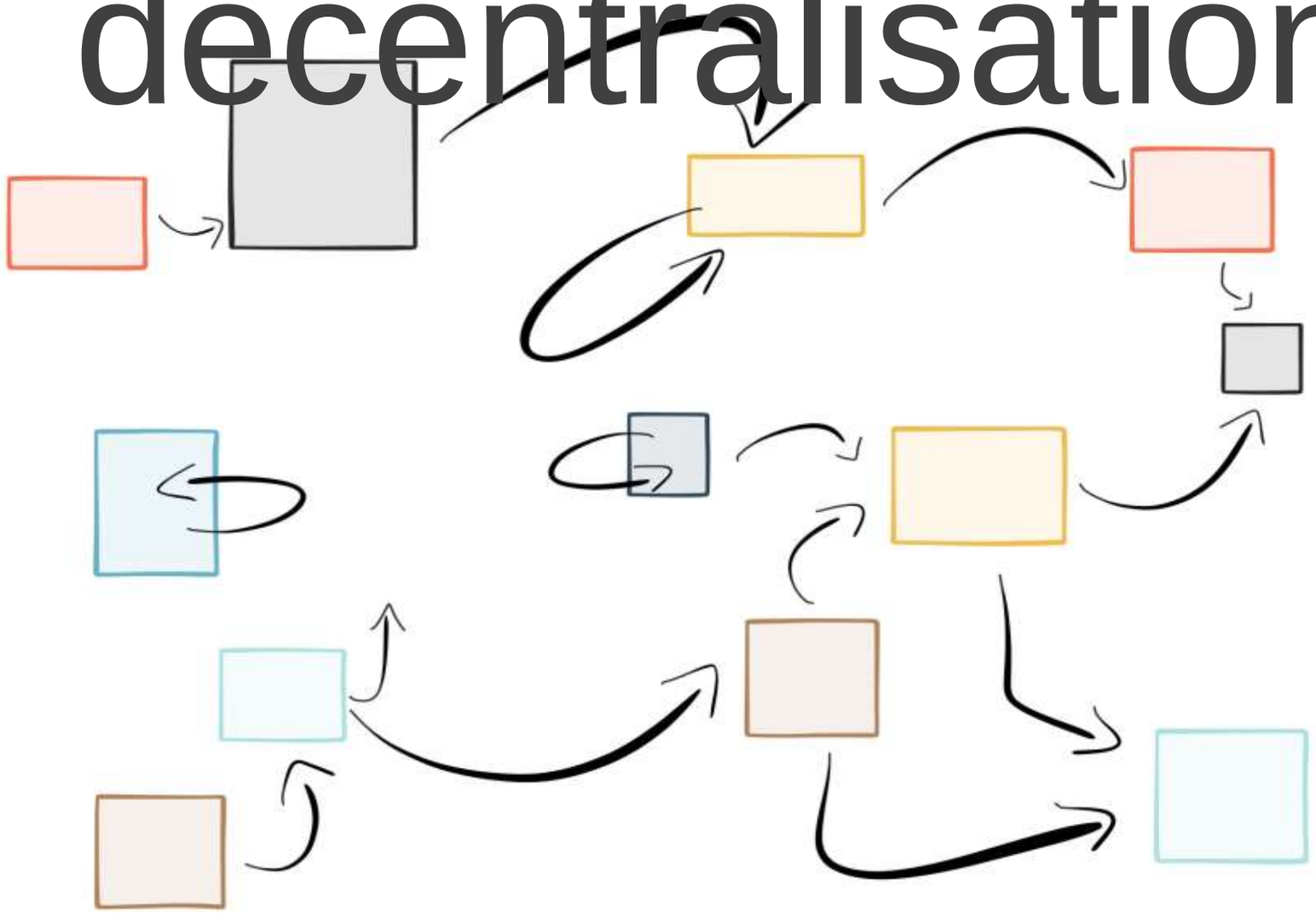
Service based approaches separate state



But state must inevitably be shared between services



Use a toolkit that embraces decentralisation

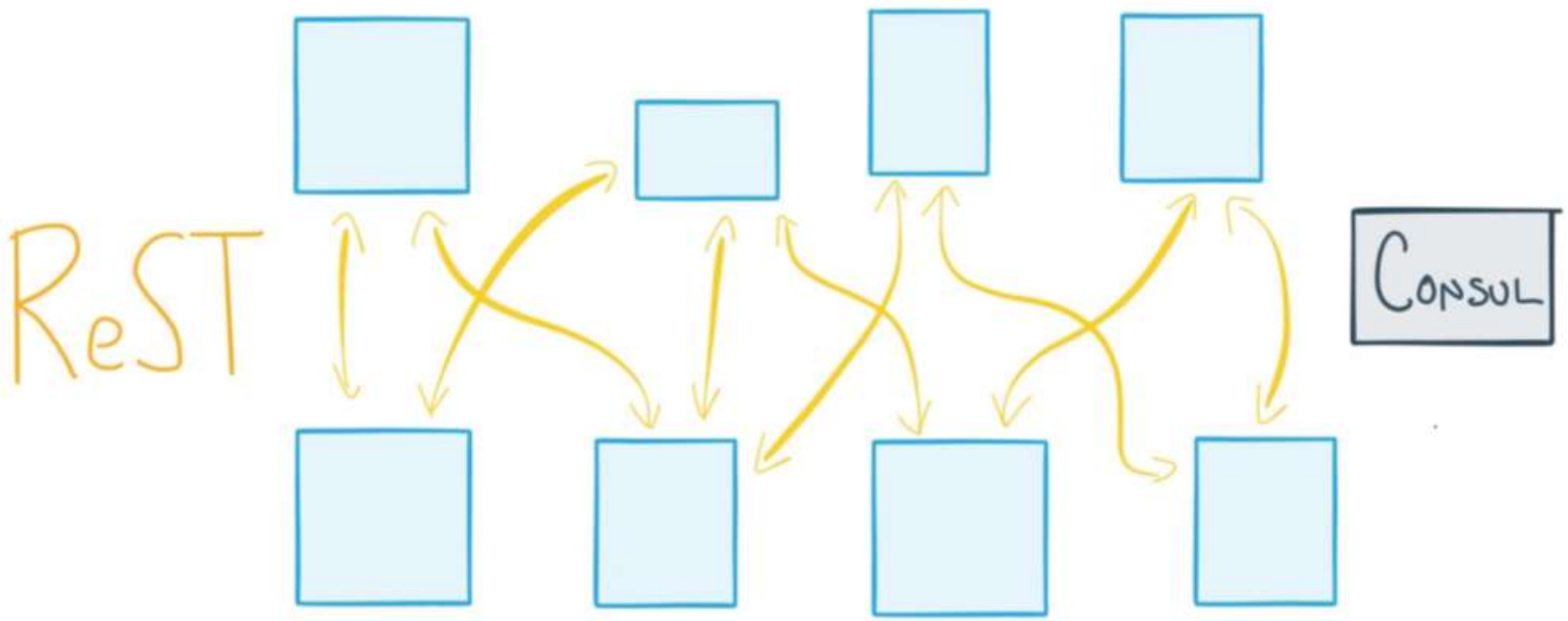


THIS IS ME

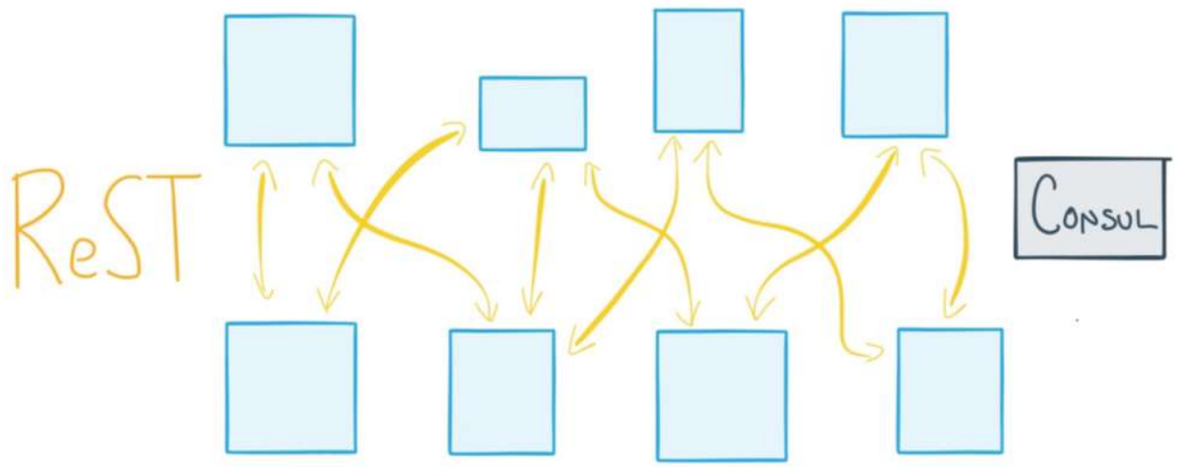
- ENGINEER AT CONFLUENT
- EX THOUGHTWORKS + UK FINANCE

Some simple patterns of distributed systems

Request / Response

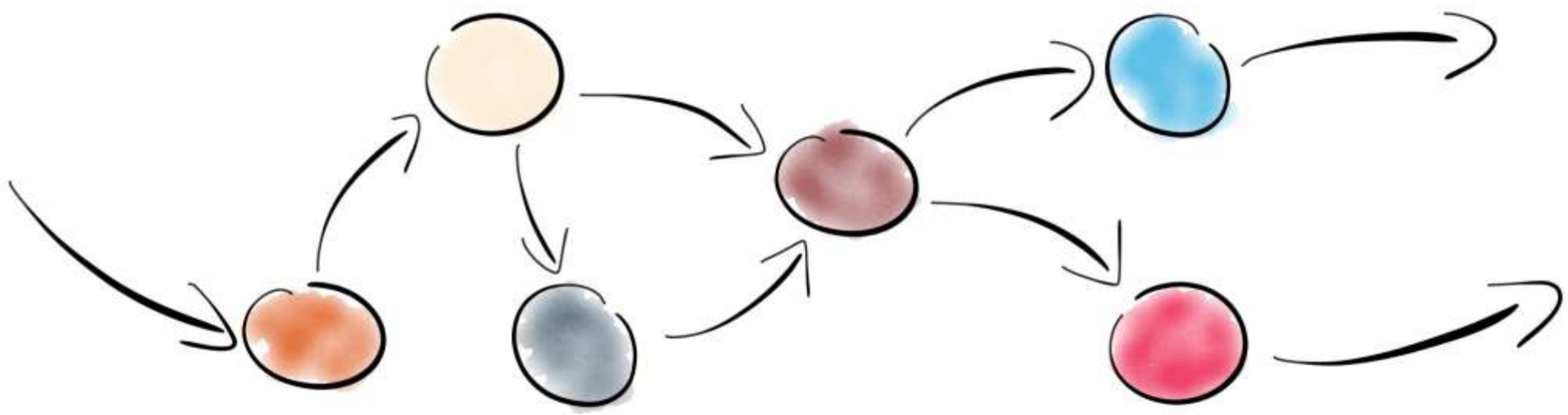


When do we need Request Response?



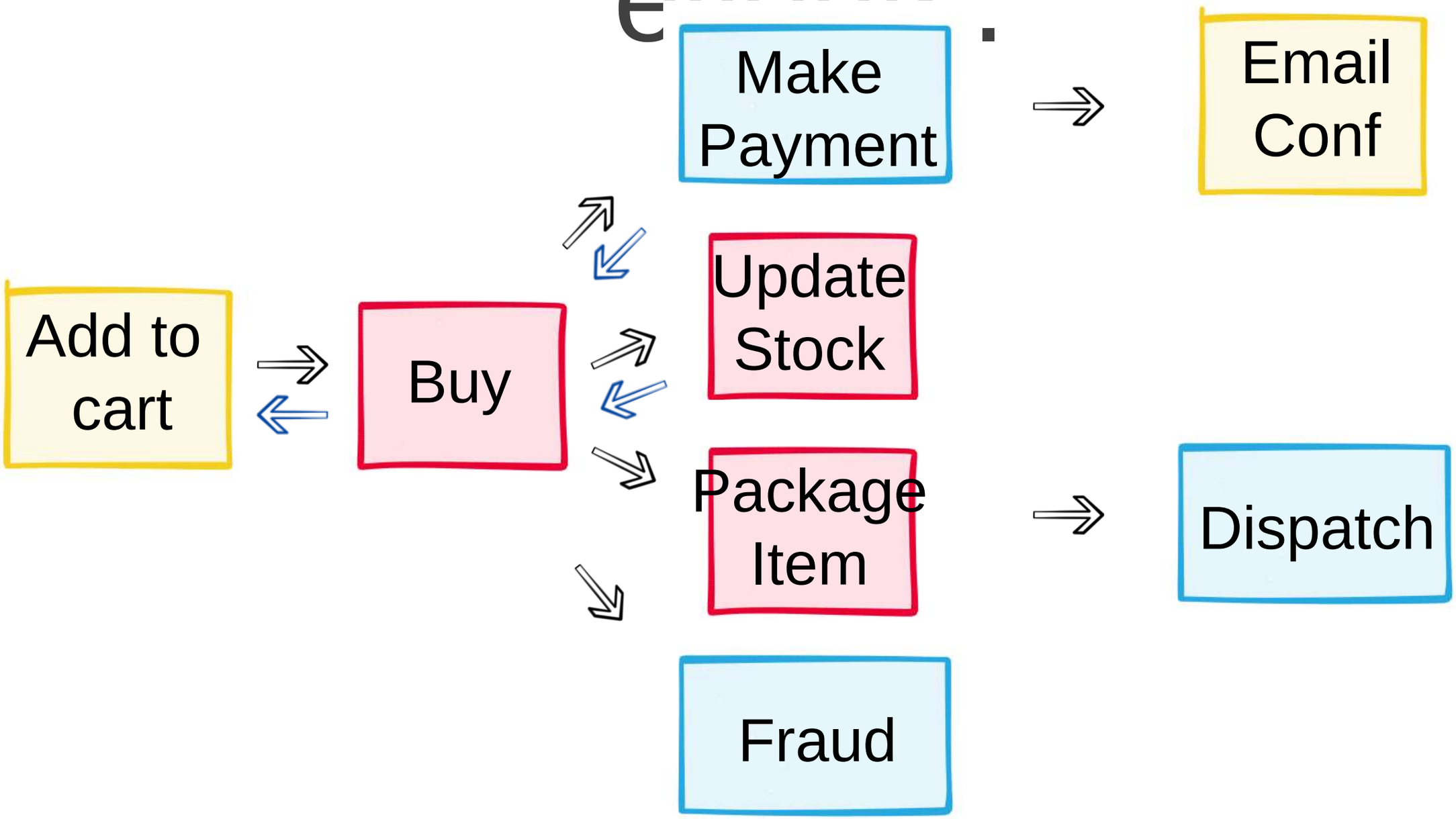
Looking things up

Event Driven



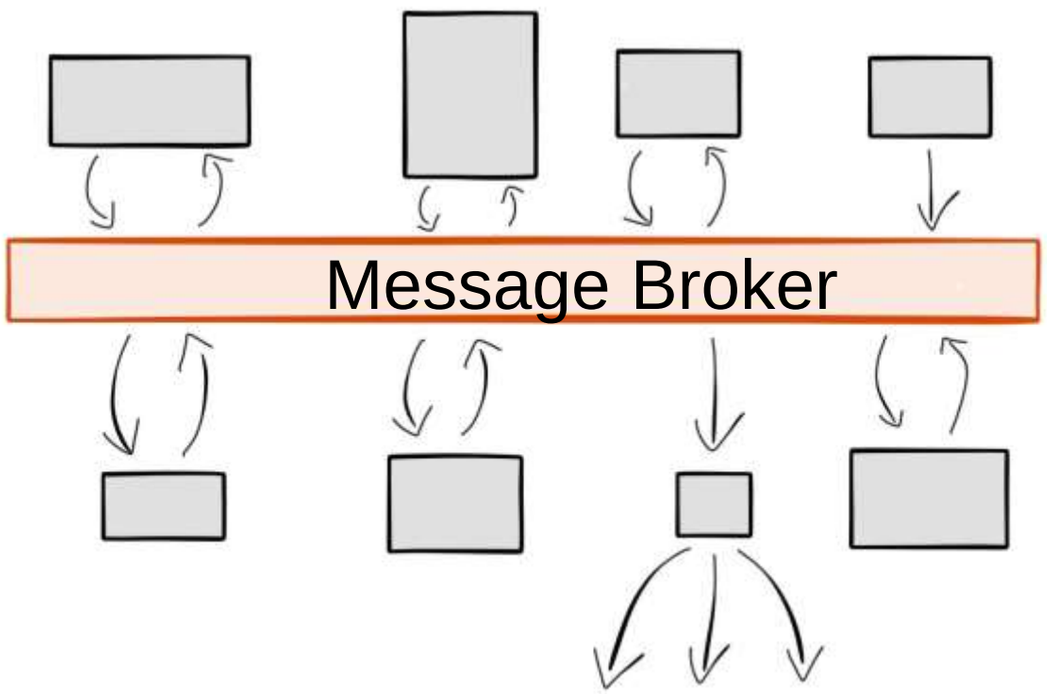
Async / Fire and Forget / Brokered

Businesses are often modeled as a sequence events:

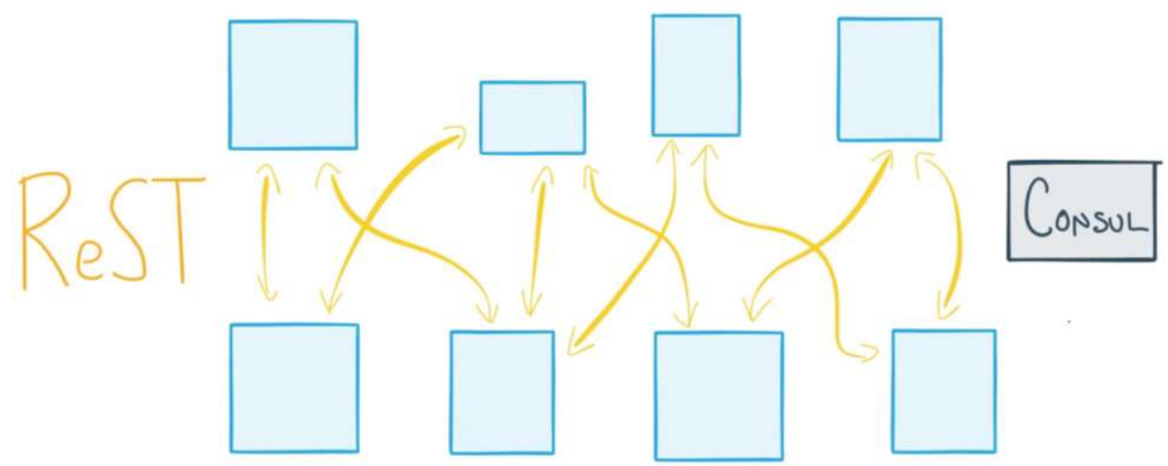


When do we need Event Driven?

SOA / Microservices



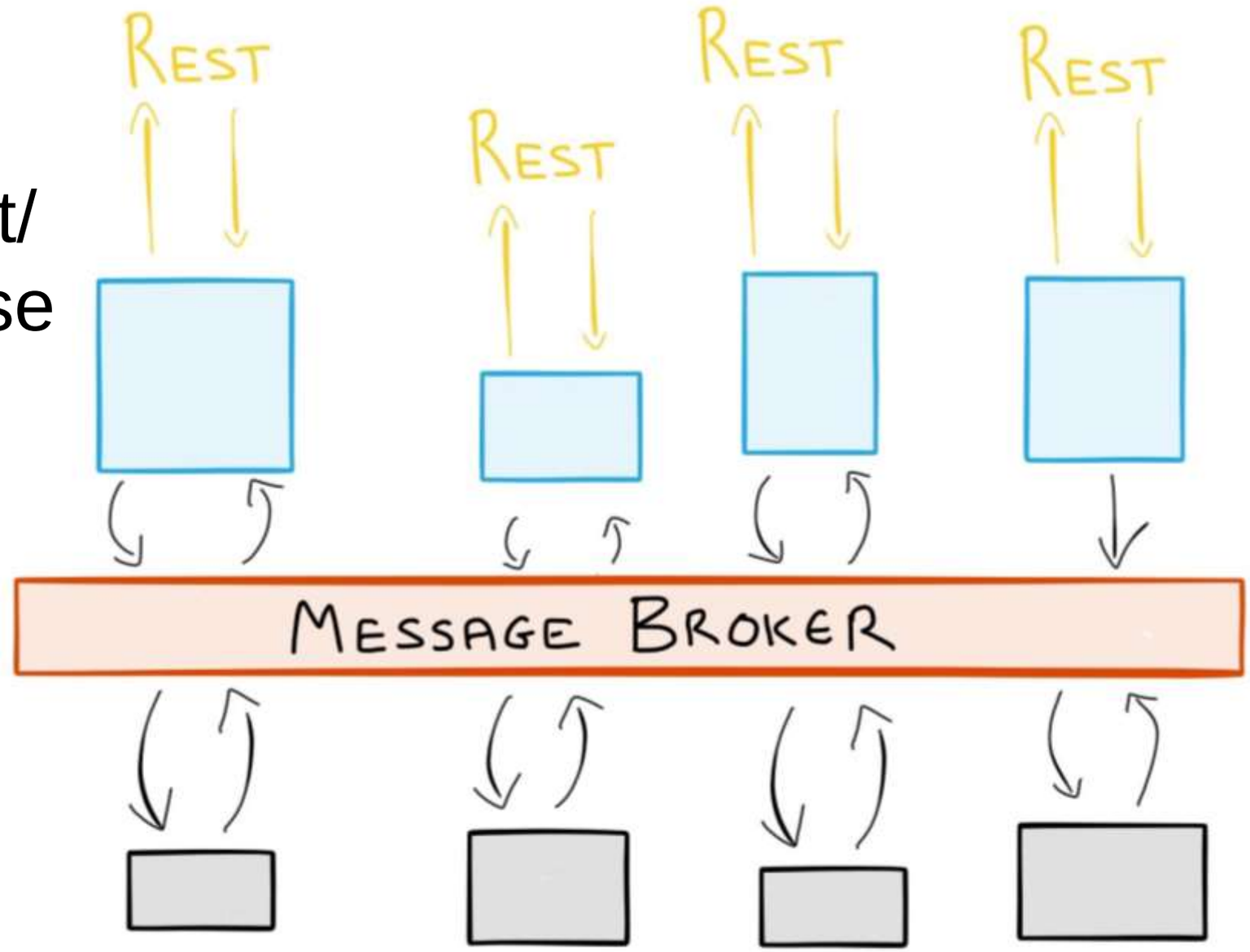
Event Based



Request/Response

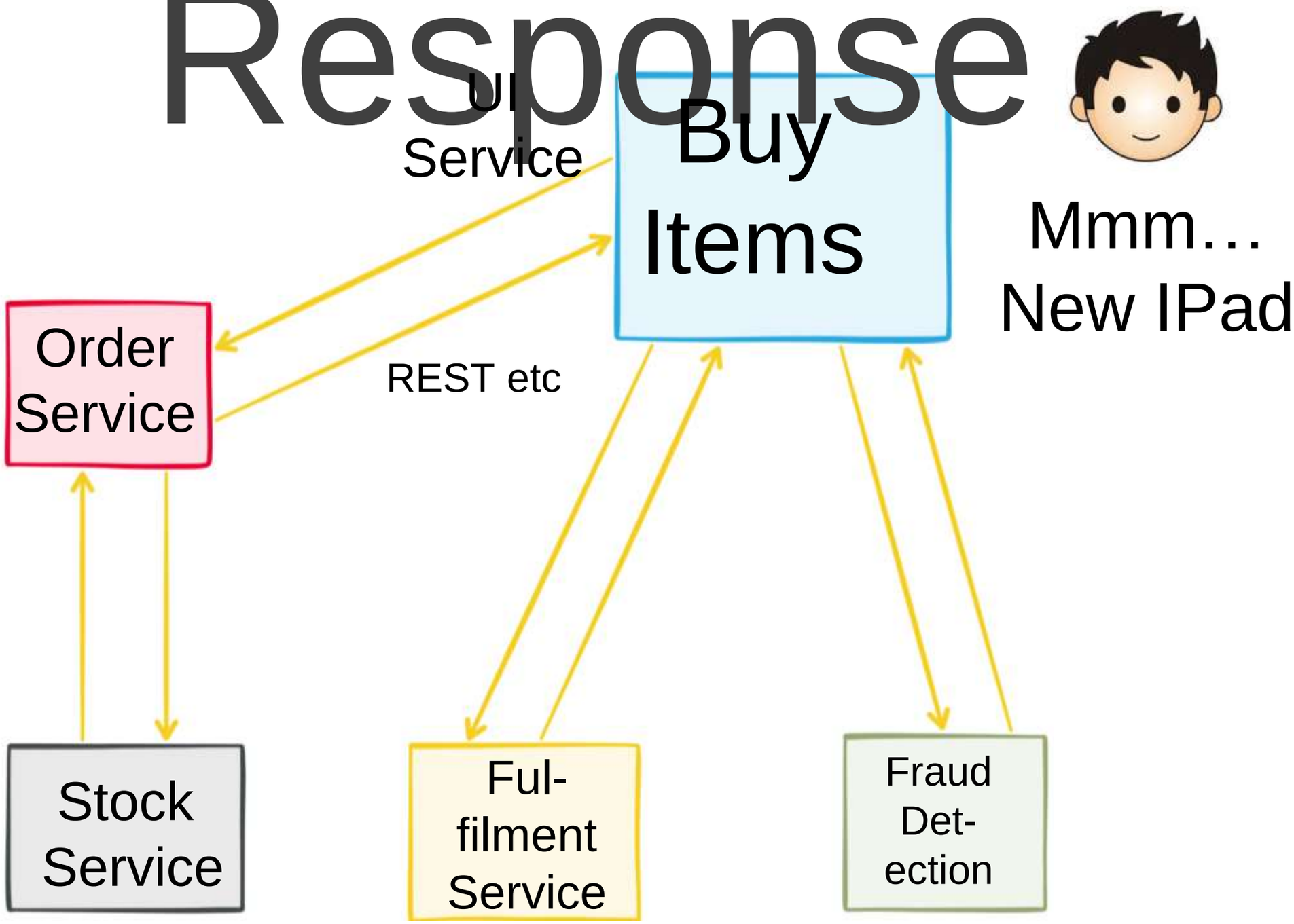
Hybrids

Request/
Response

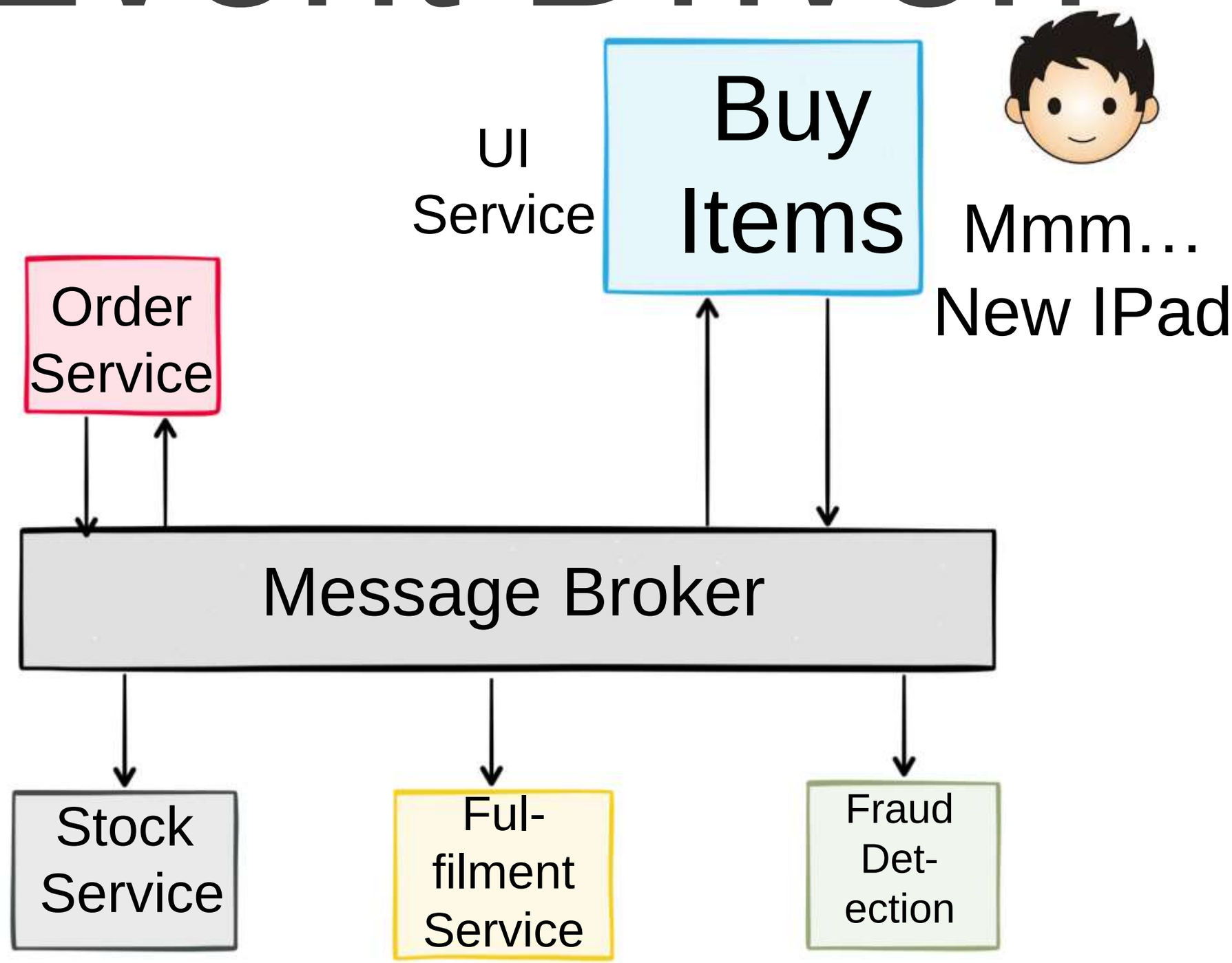


Event-
Based

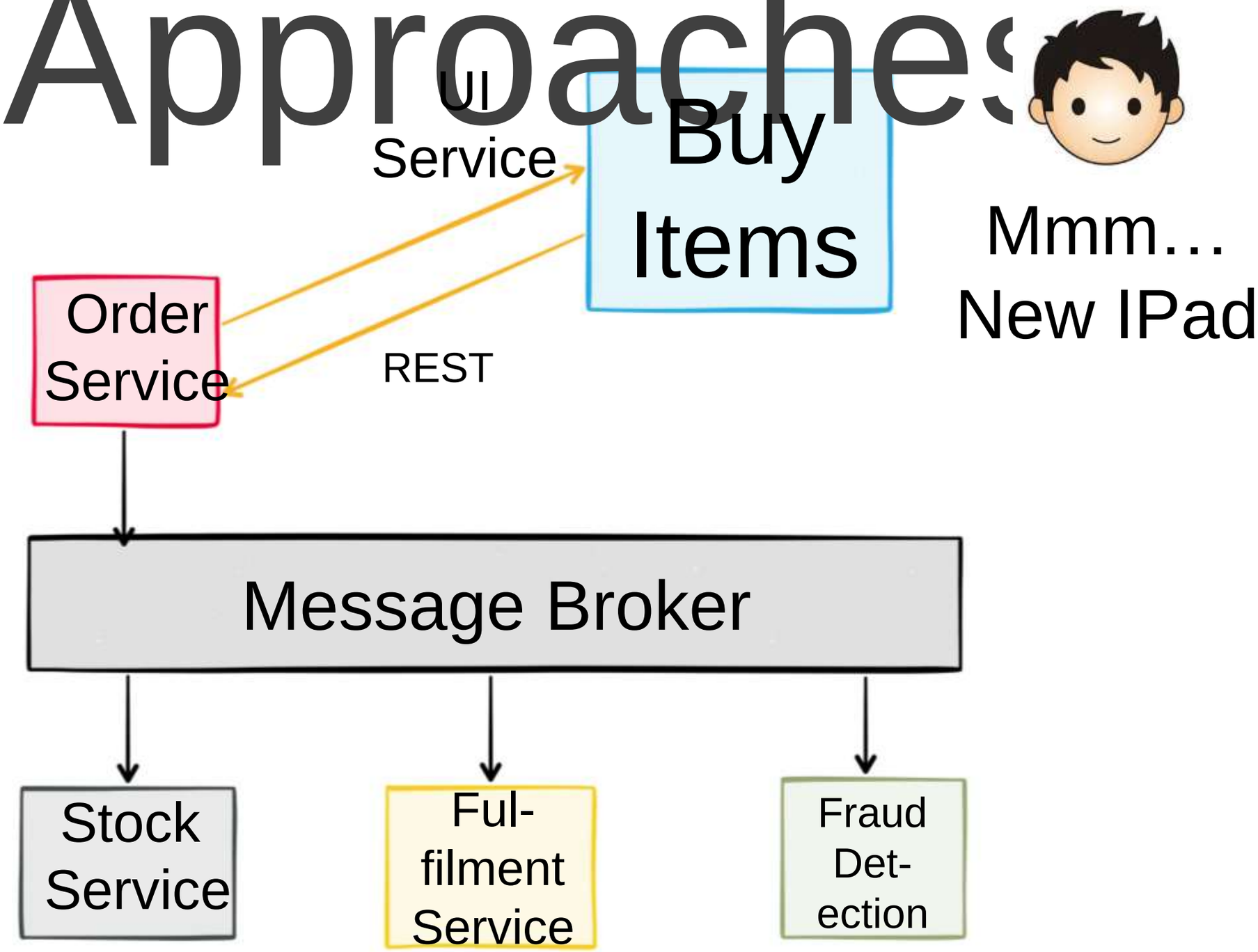
Request Response



Event Driven



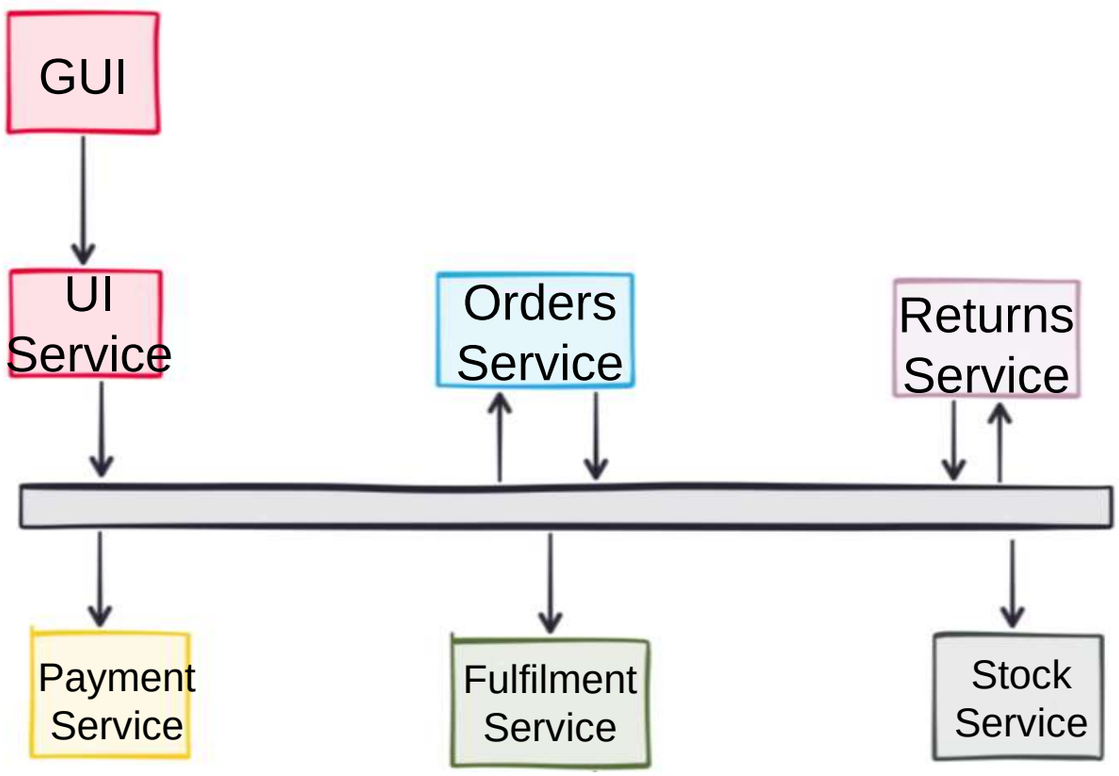
Hybrid Approaches



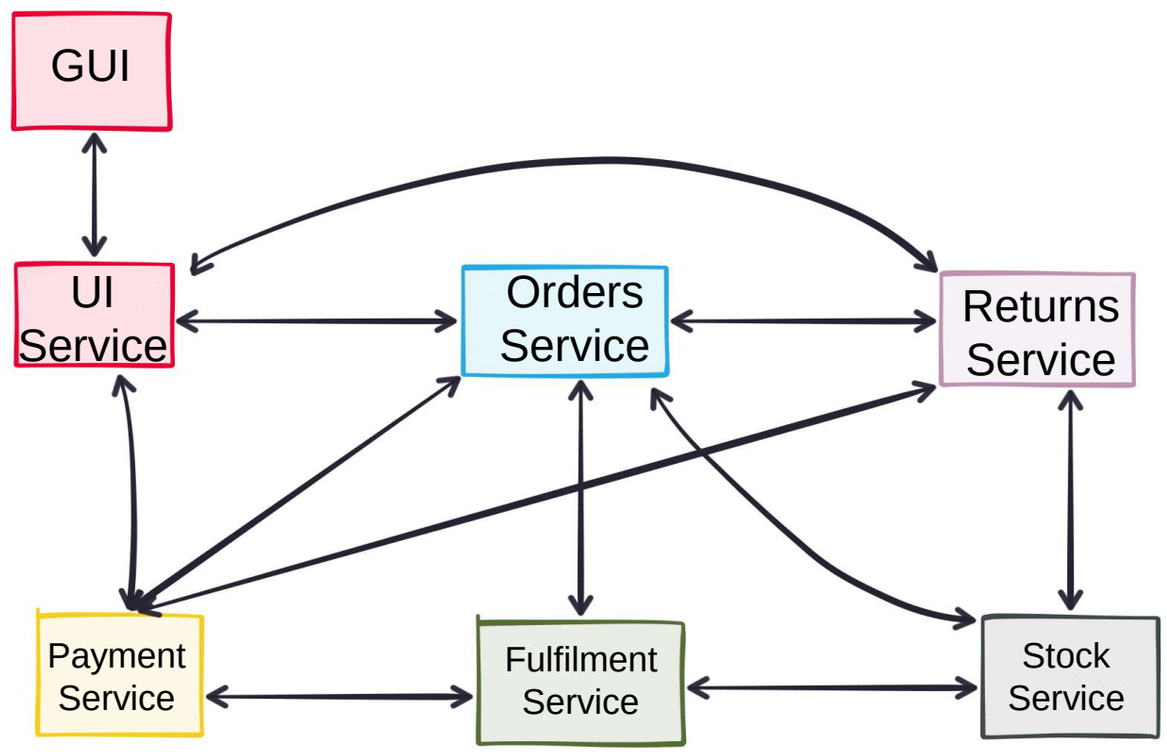
As software engineers we are inevitably affected by the tools we surround ourselves with.

Languages, frameworks, even processes all act to shape the software we build.

The tools we choose have a big effect on our architecture

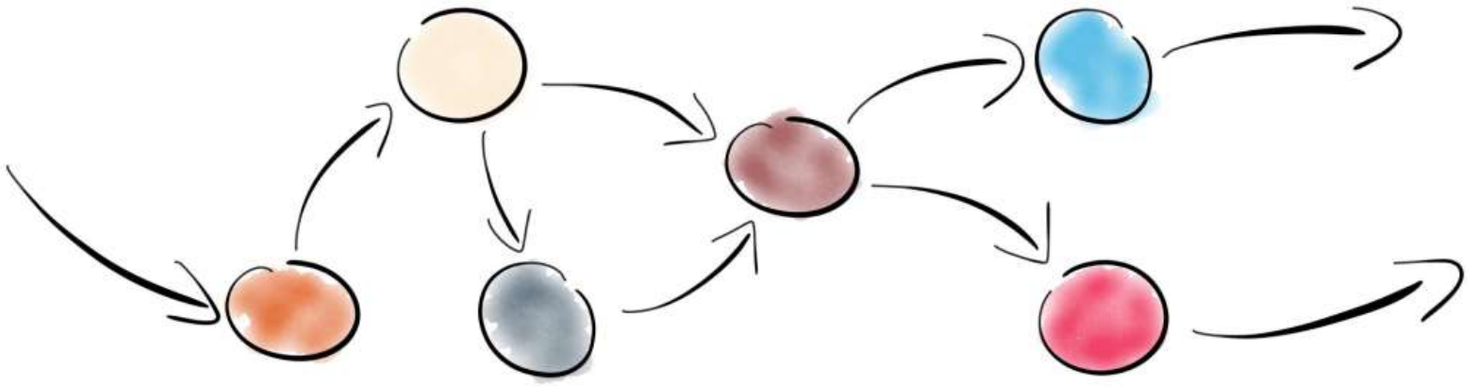


Event Driven



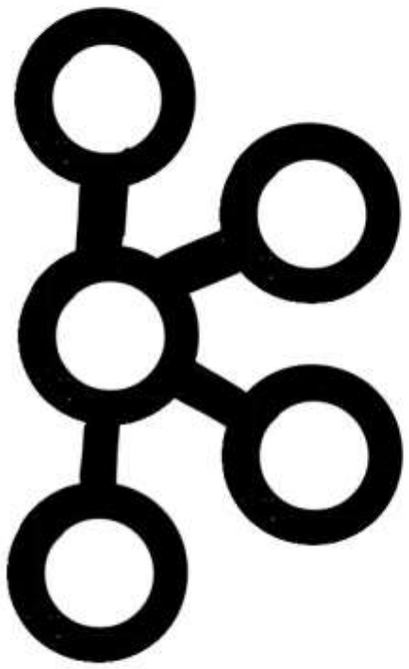
Request Response

Kafka is well suited to Event Driven Architectures



It will lead you down that path

The Tool Set

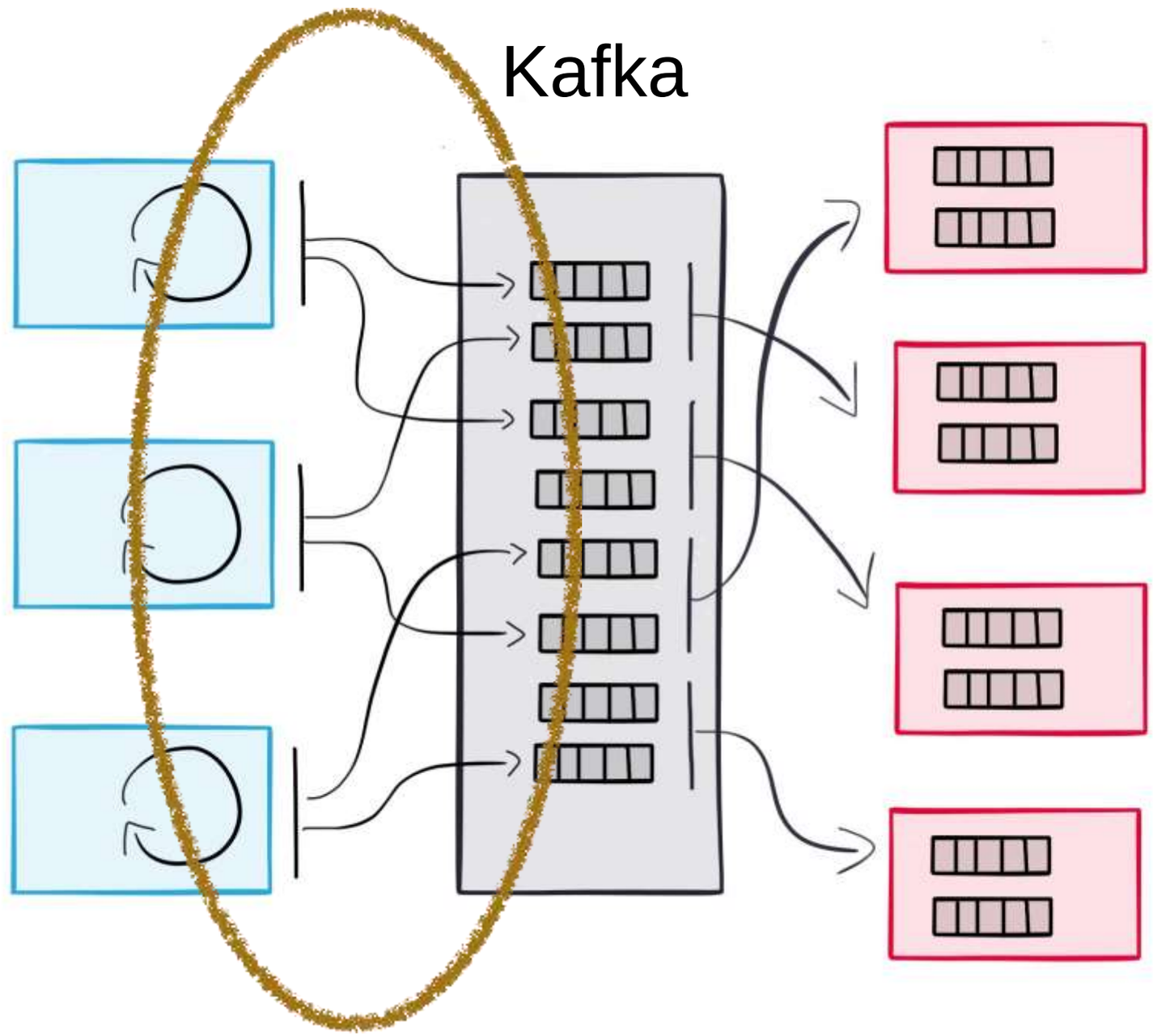


Kafka

a Distributed Log

What is a Distributed Log?

Shard on the way in

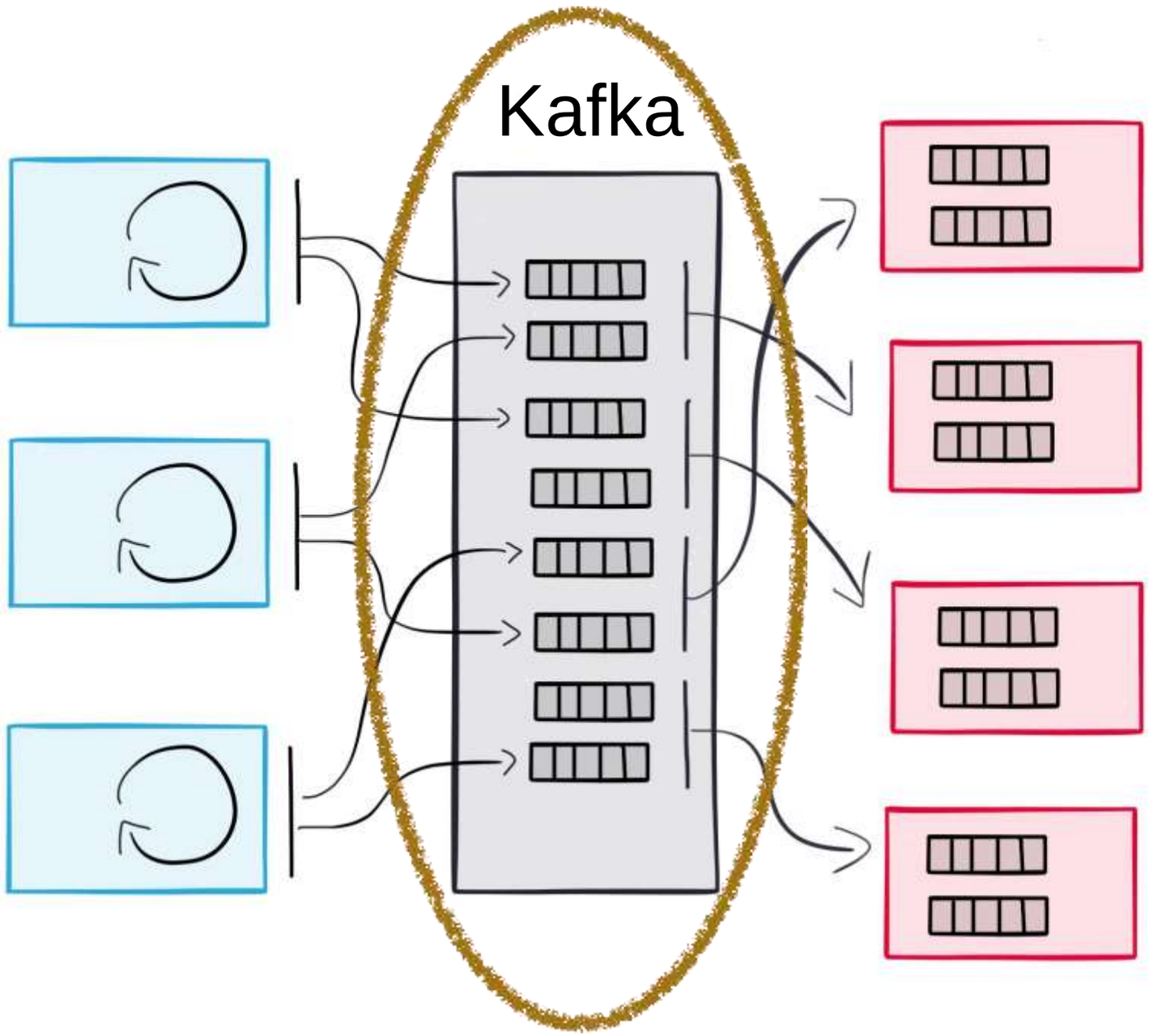


Producing Services

Consuming Services

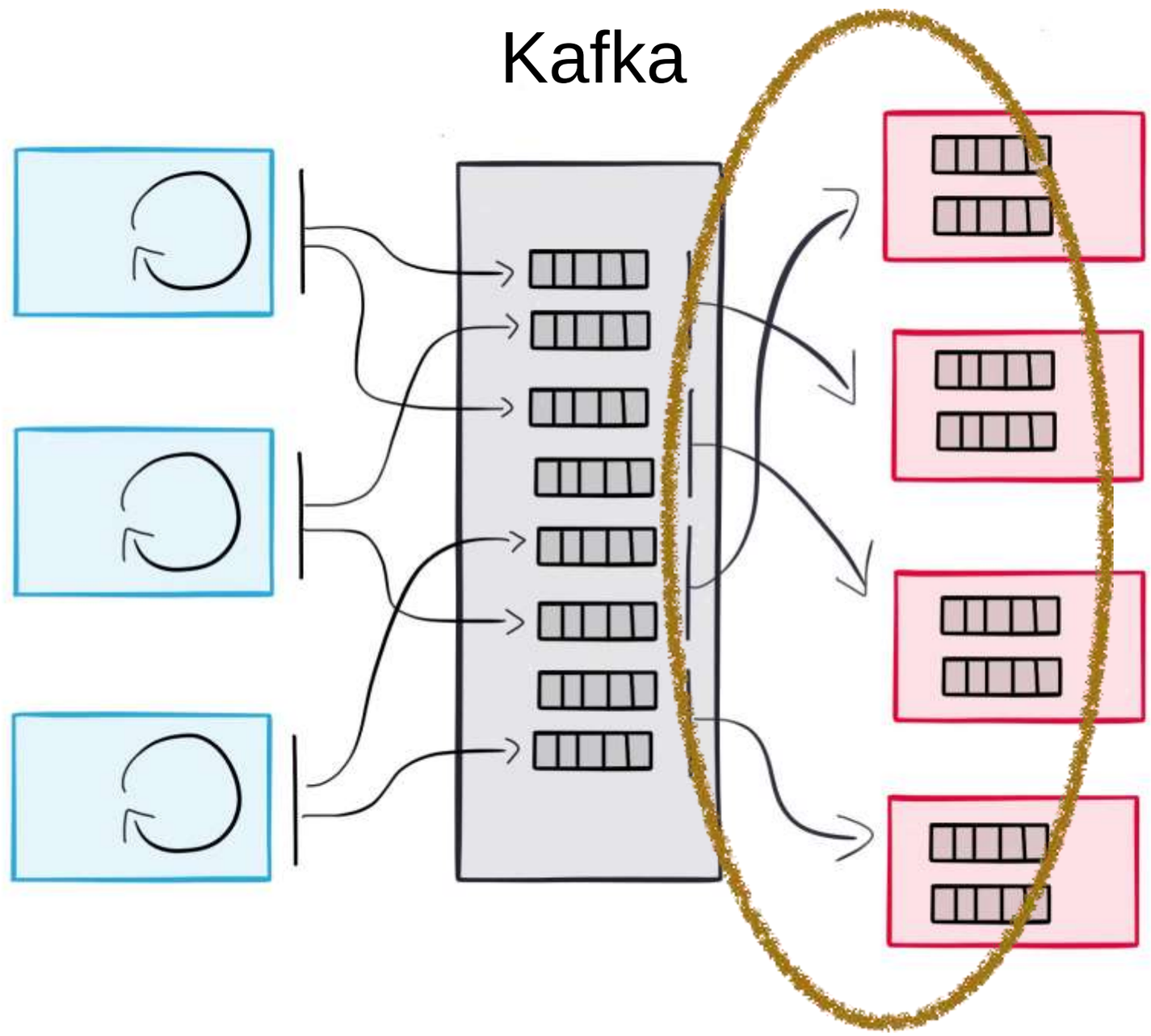
Each shard is a queue

Producing Services

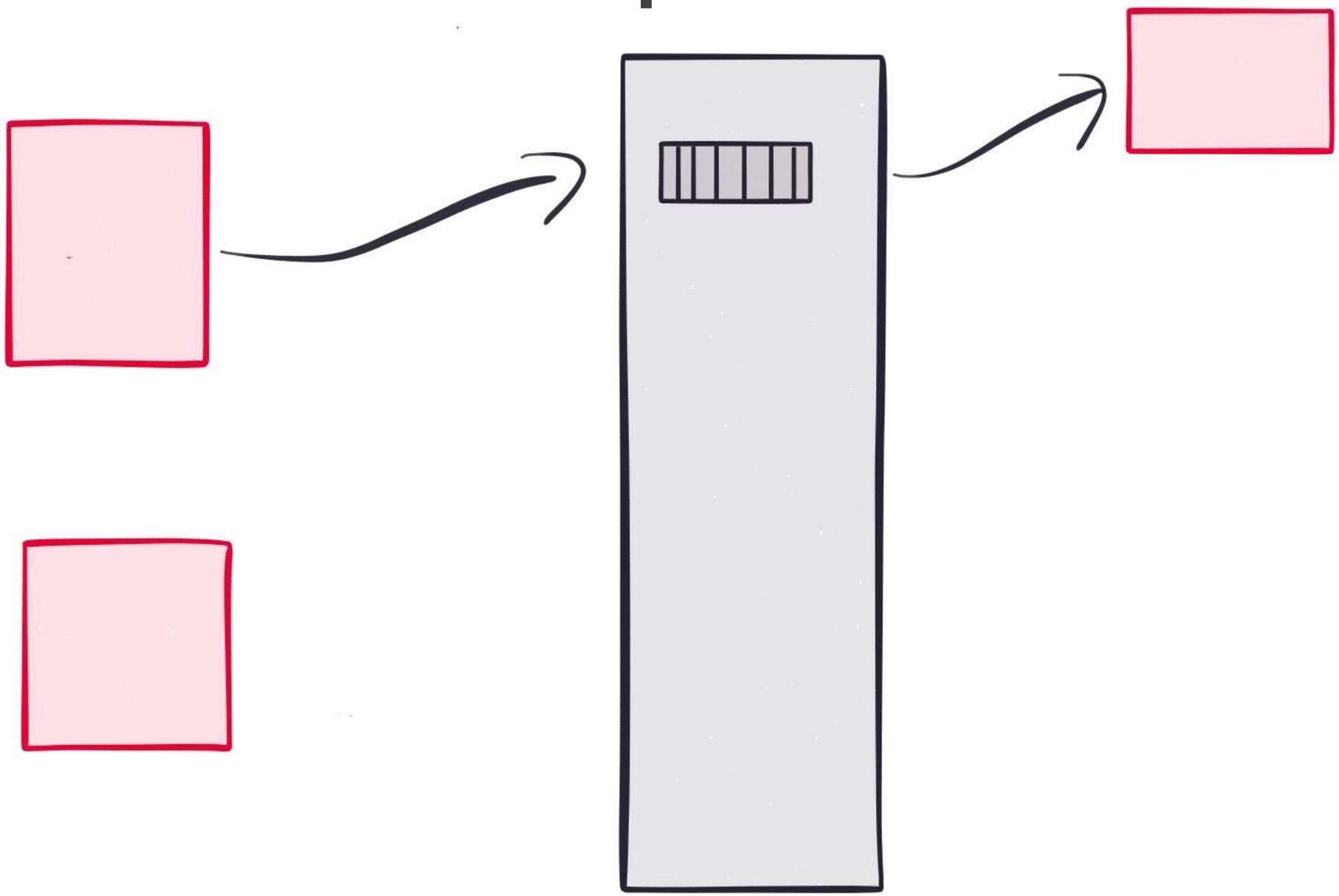


Consuming Services

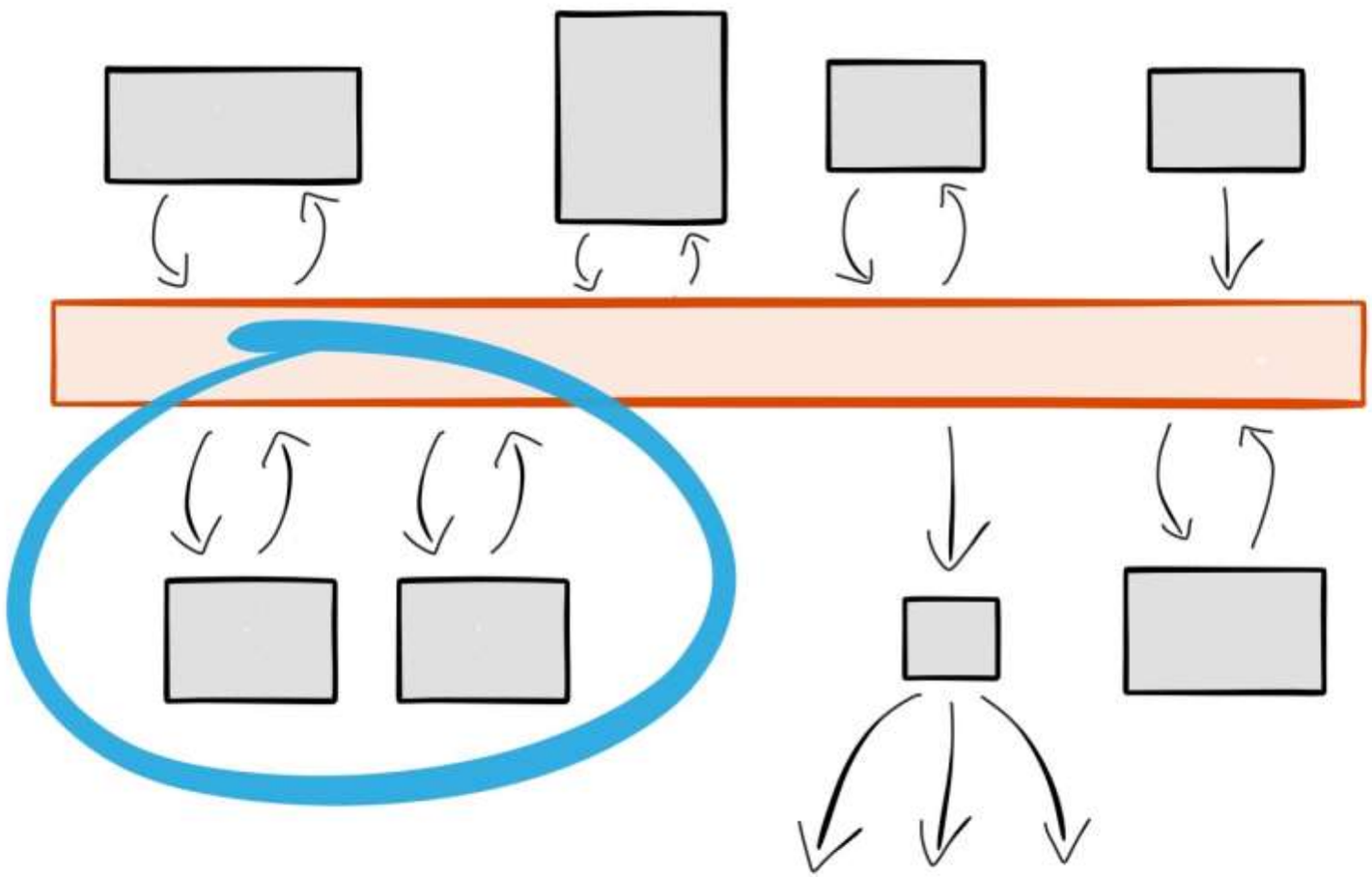
Consumers share load



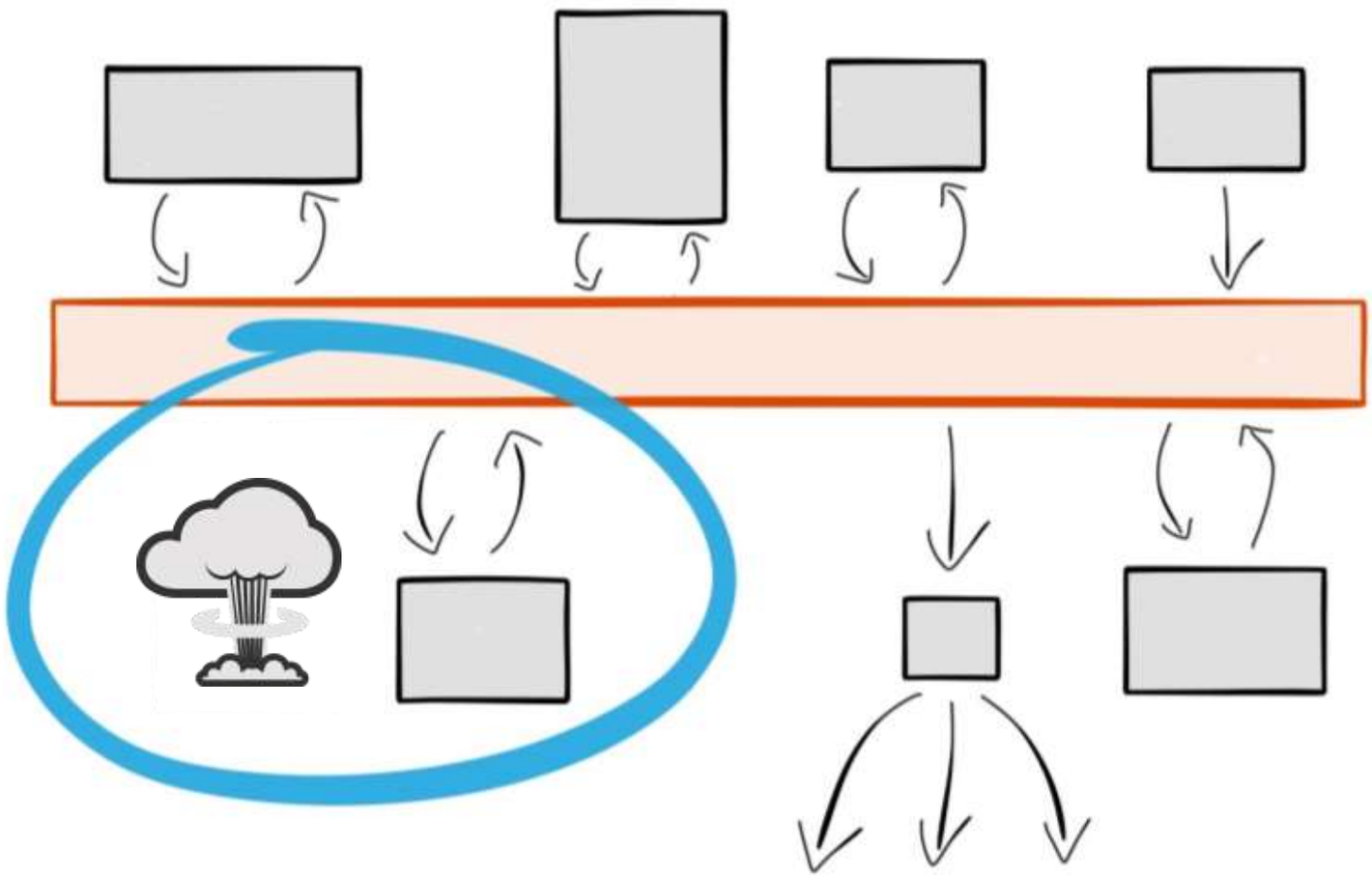
Reduces to a globally ordered queue



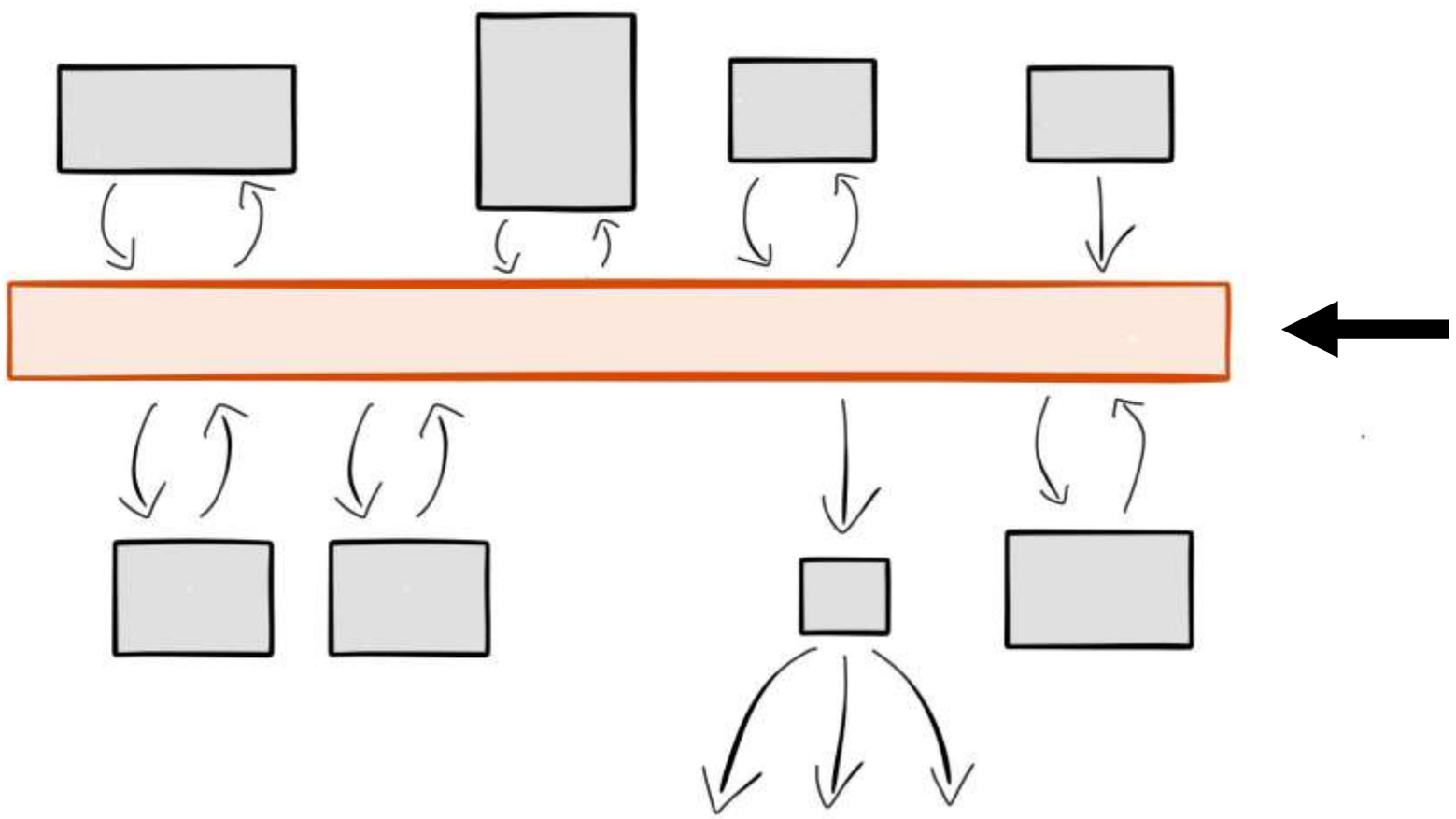
Load Balanced Services



Fault Tolerant Services

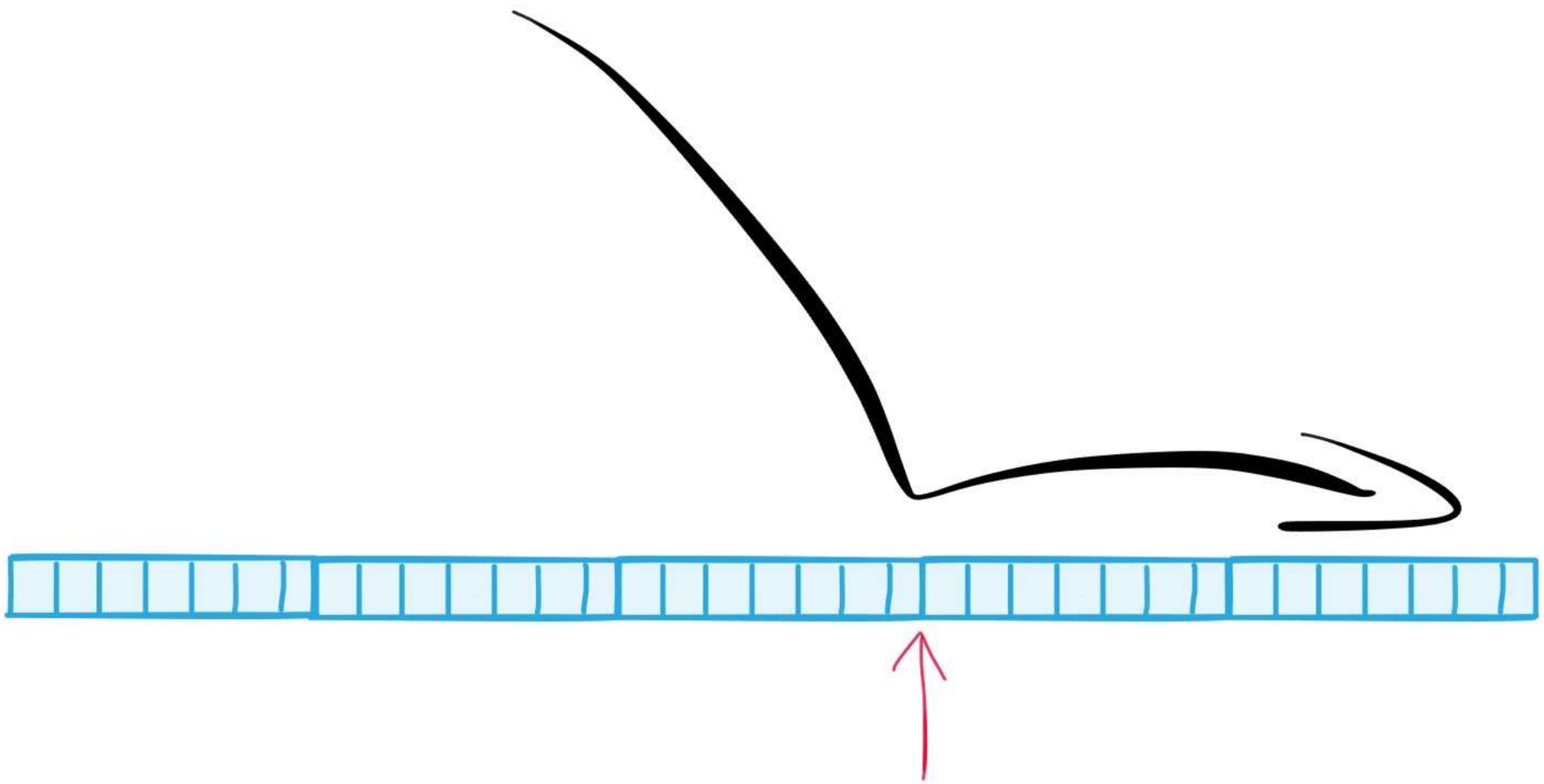


Build ‘Always On’ Services



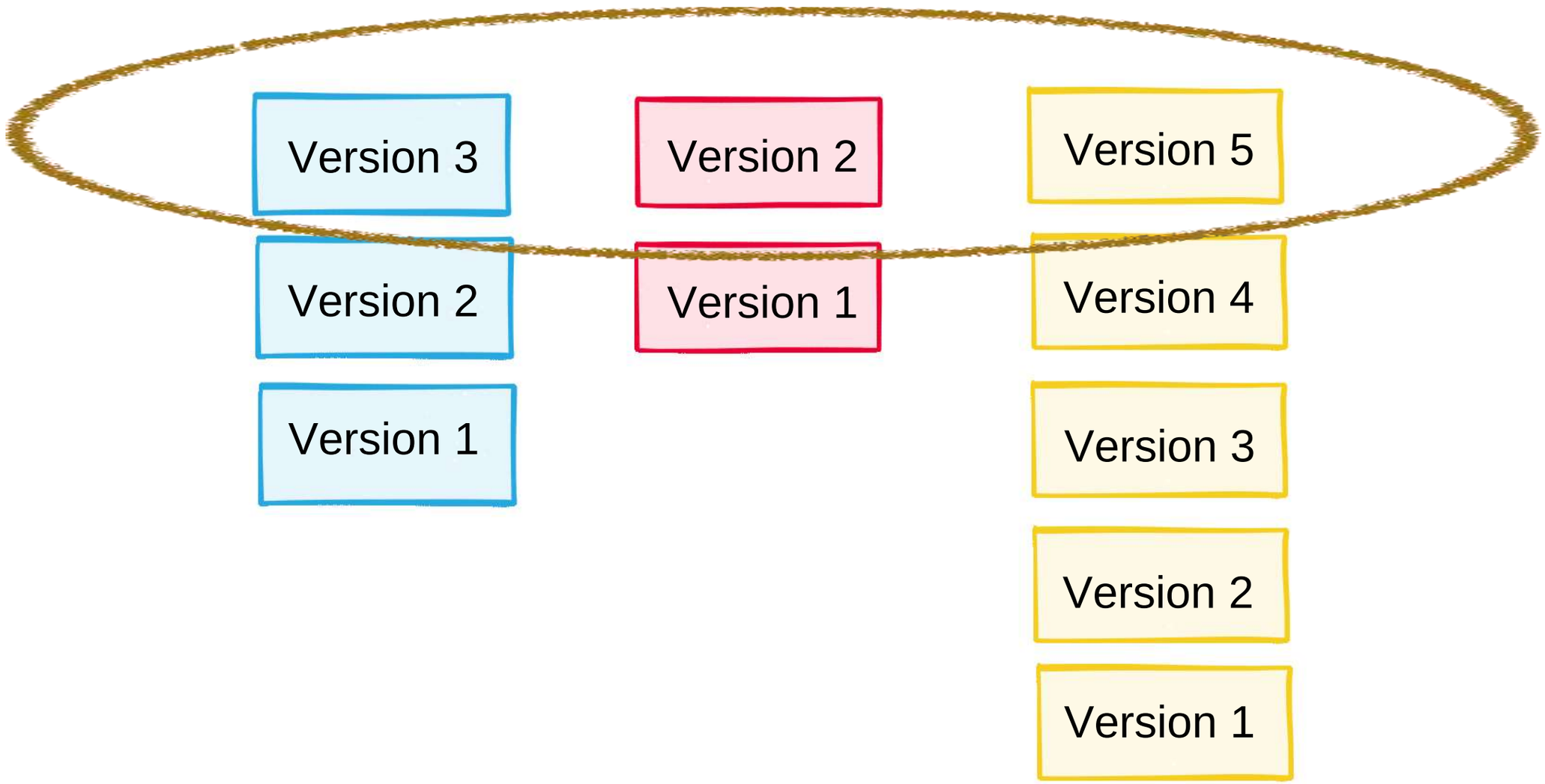
Rely on Fault Tolerant Broker

Services can “Rewind & Replay” the log



Rewind & Replay

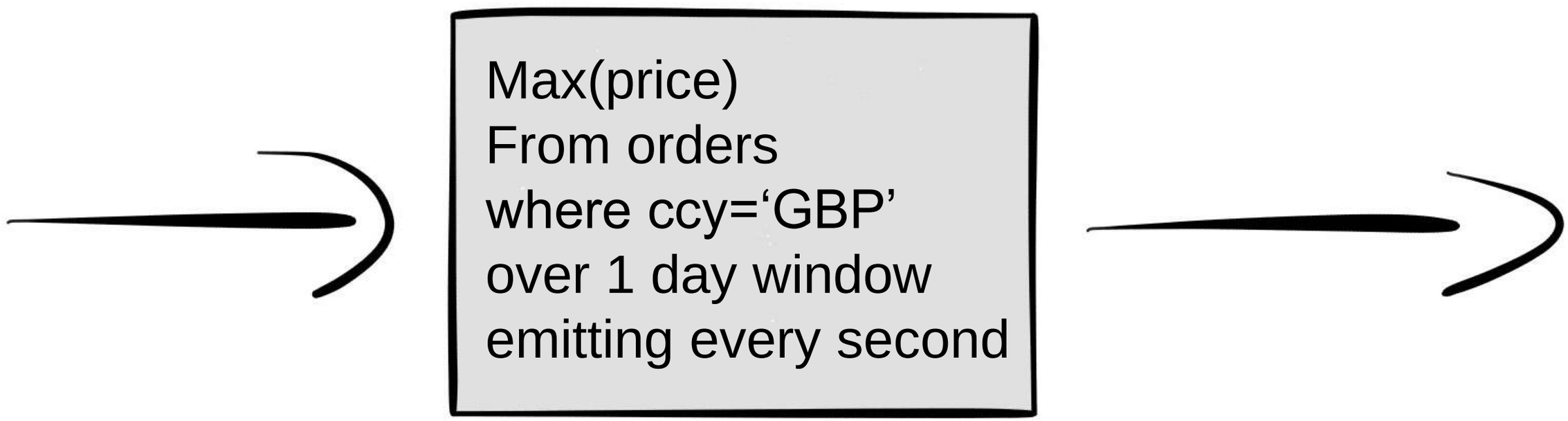
Compacted Log
(retains only latest version)



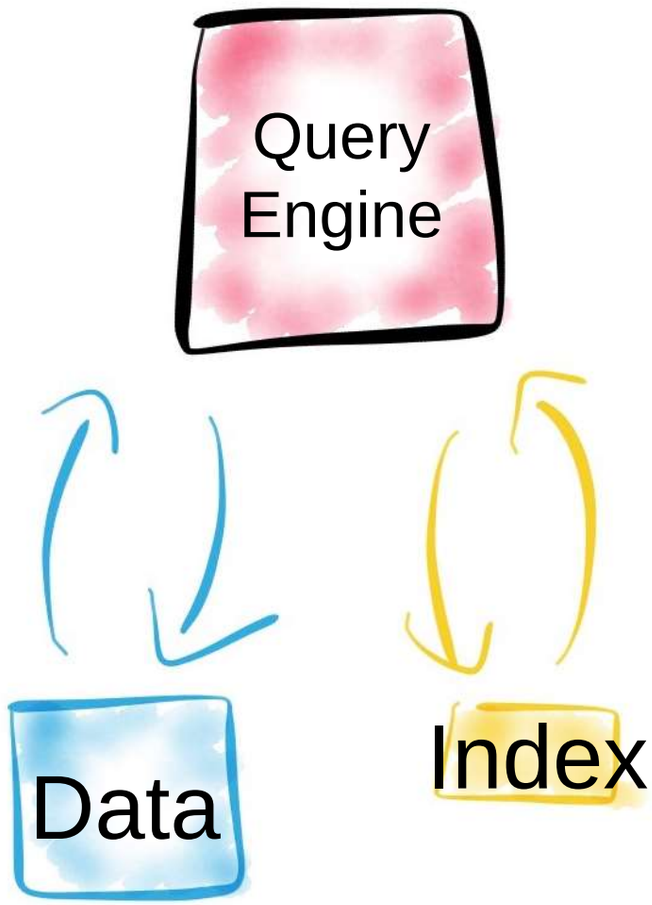
KSTREAMS

A database engine for data-in-flight

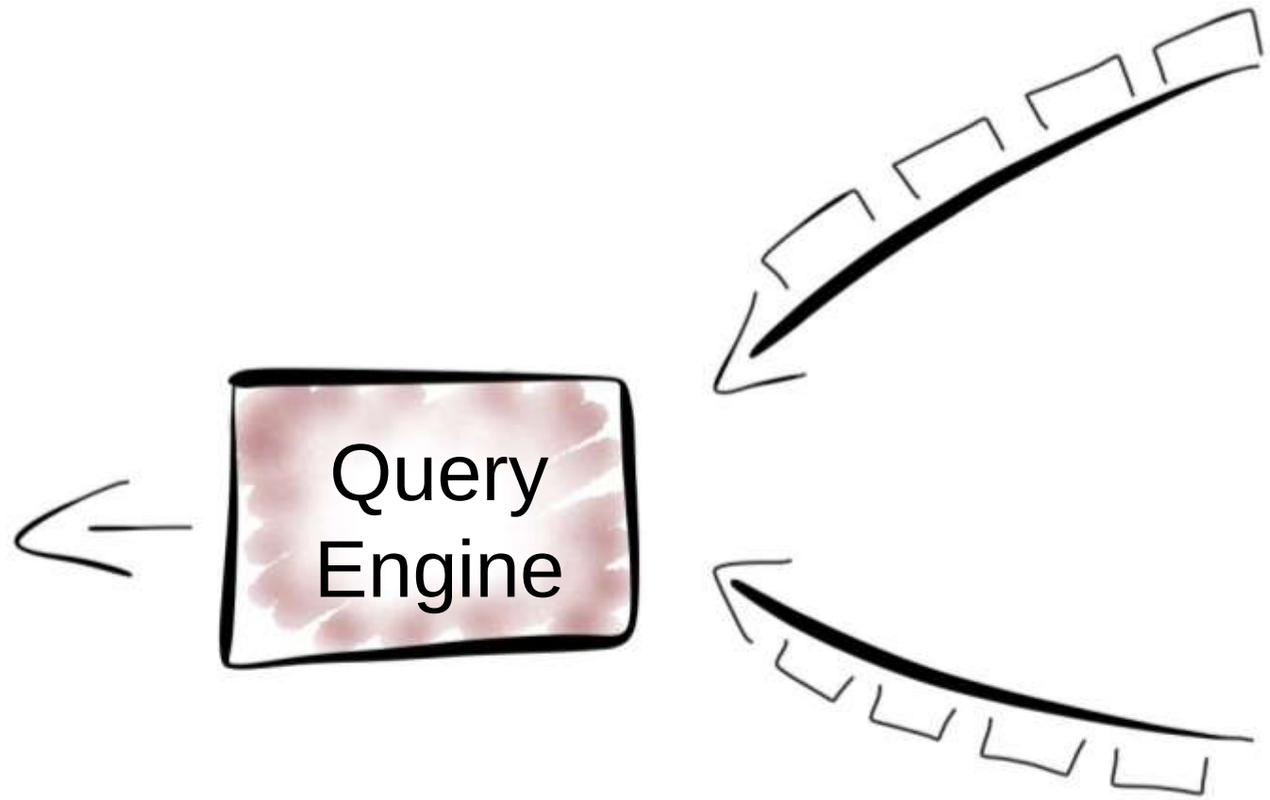
Continuously Running Queries



What is stream processing engine?



vs



Database

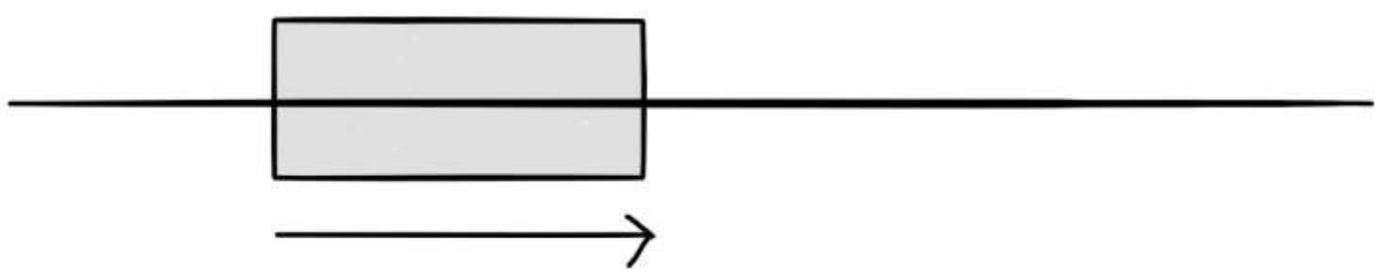
Finite source

Stream Processor

Infinite source

Windowing

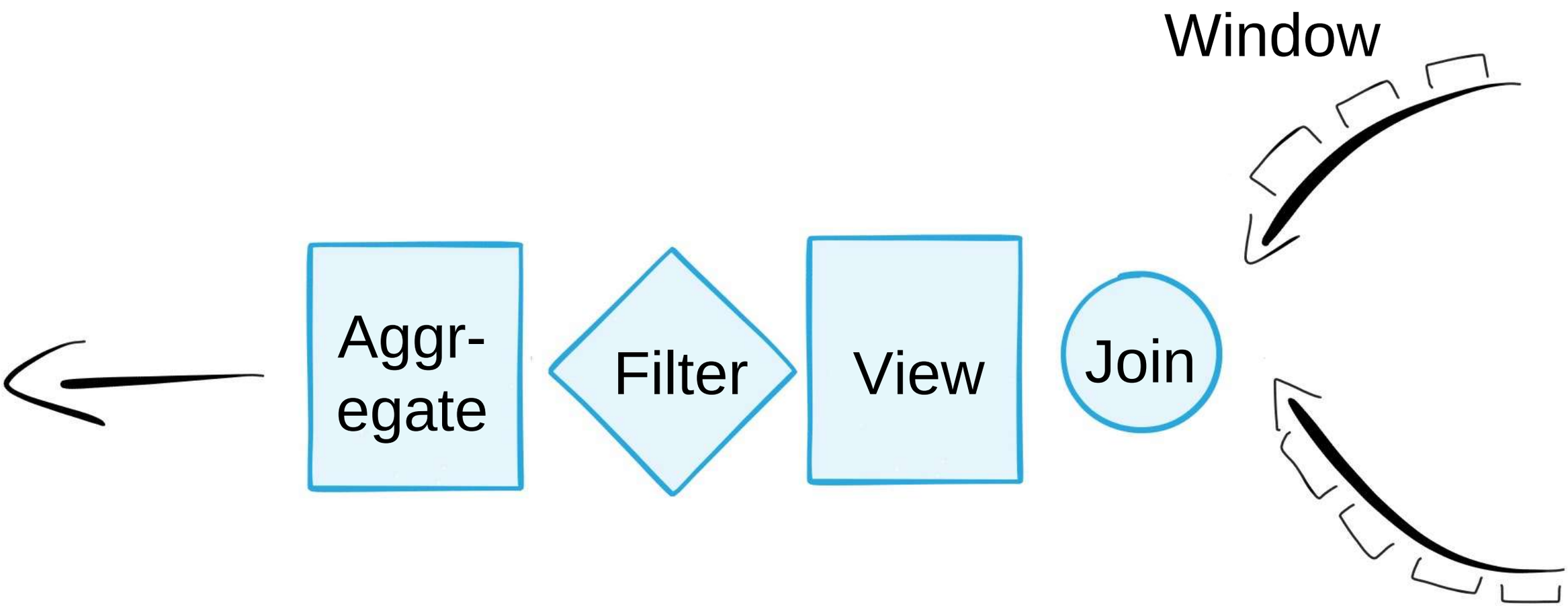
Fixed (tumbling)



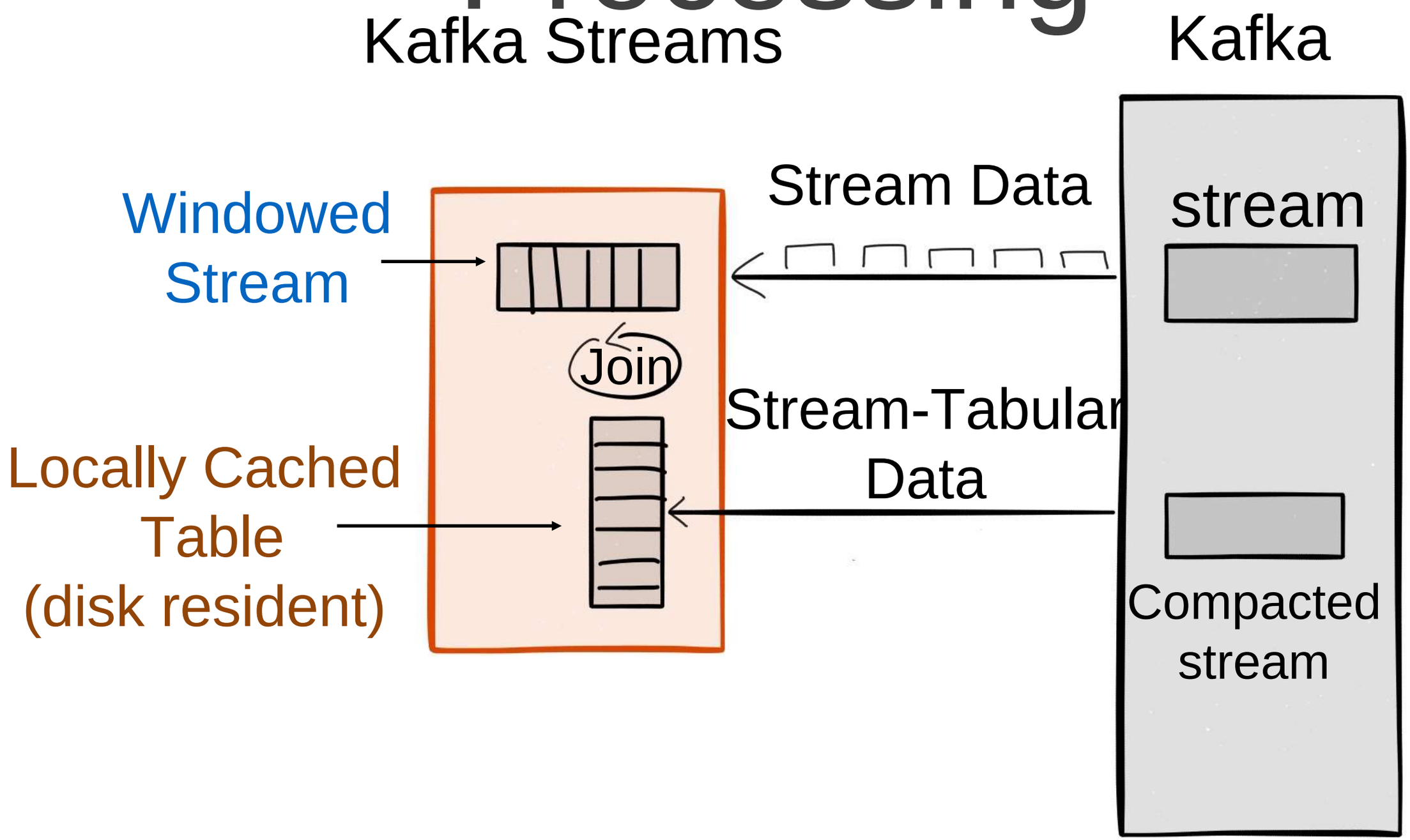
Sliding

For unordered or unpredictable streams

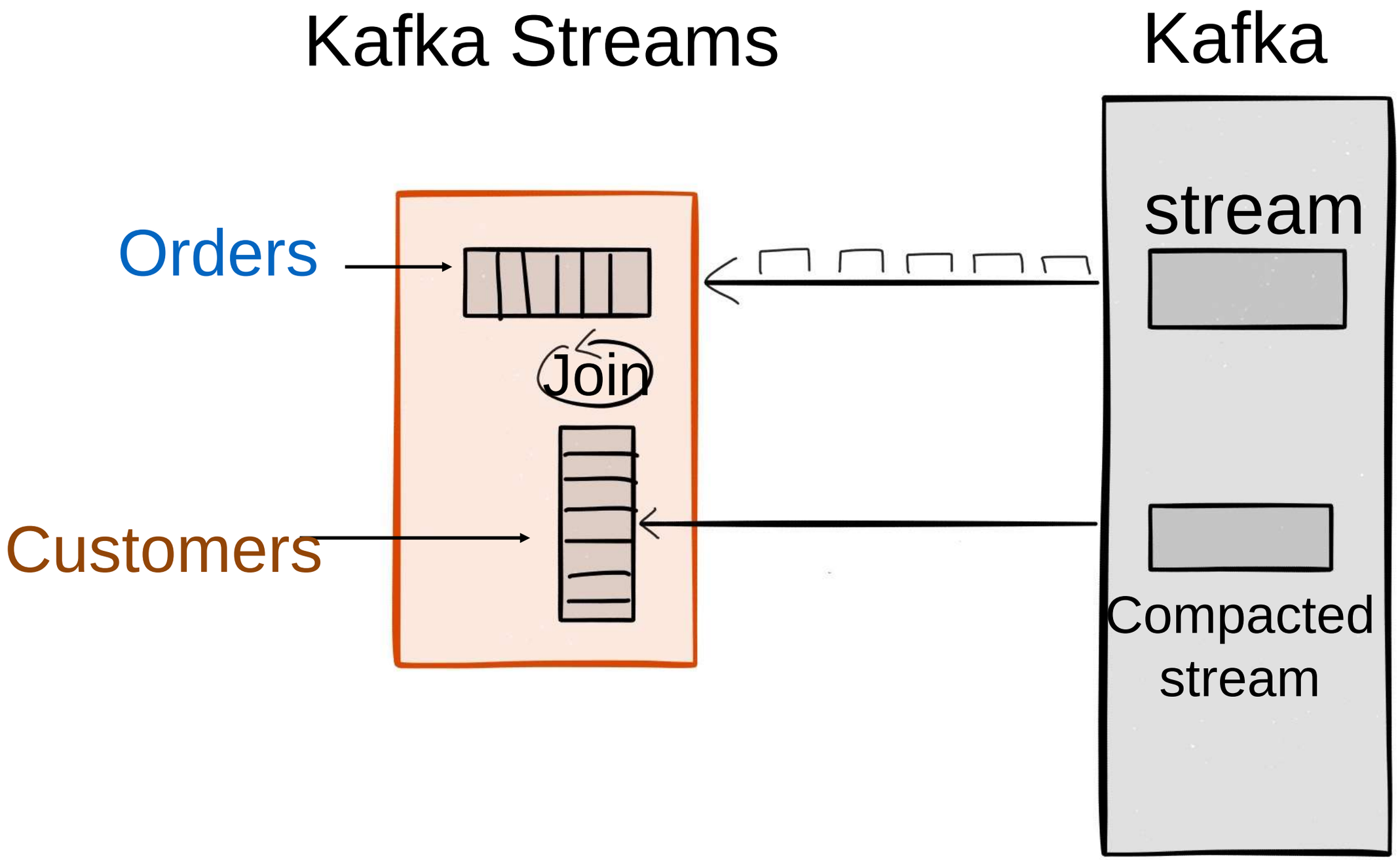
Features: similar to database query engine



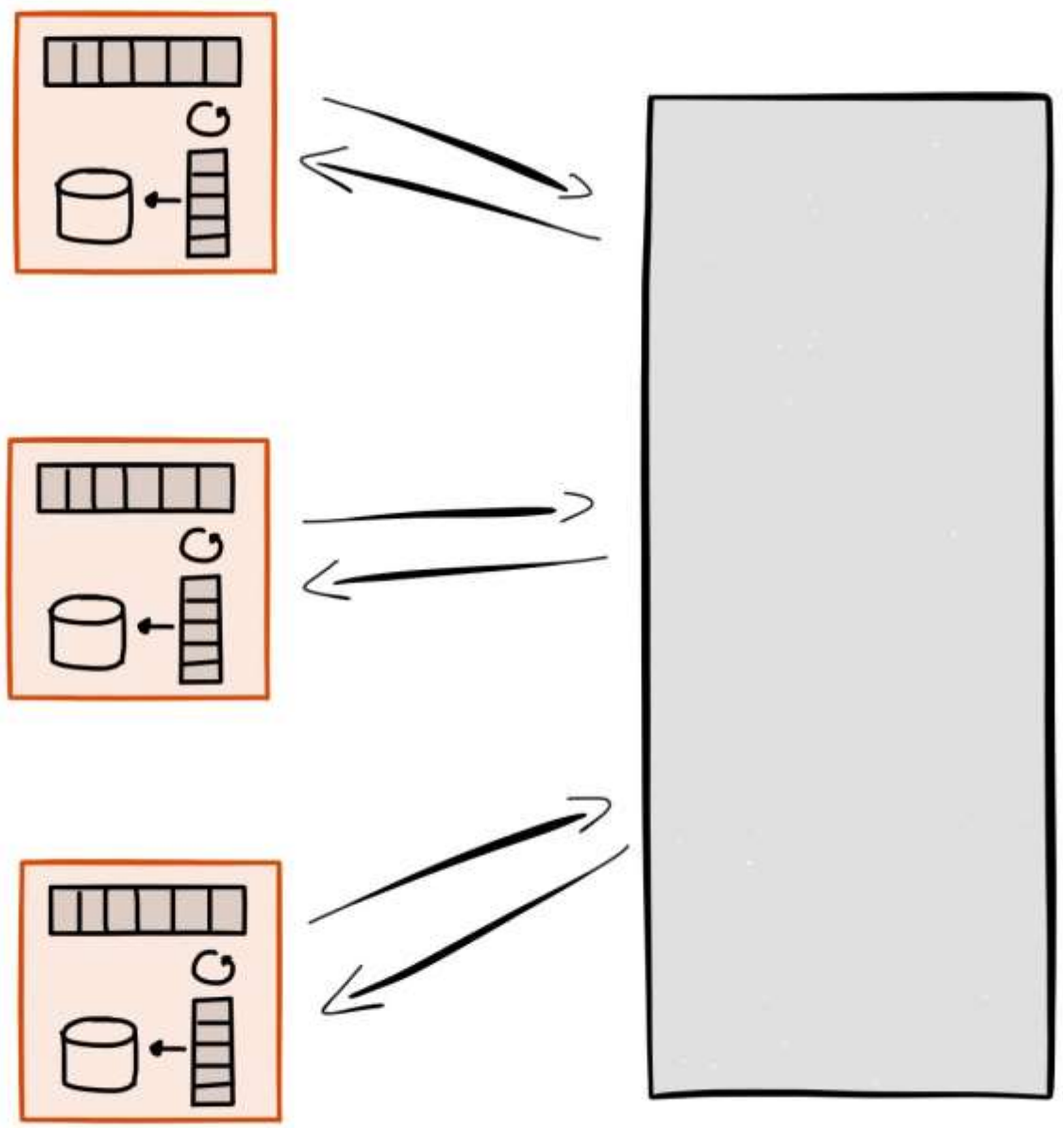
Stateful Stream Processing



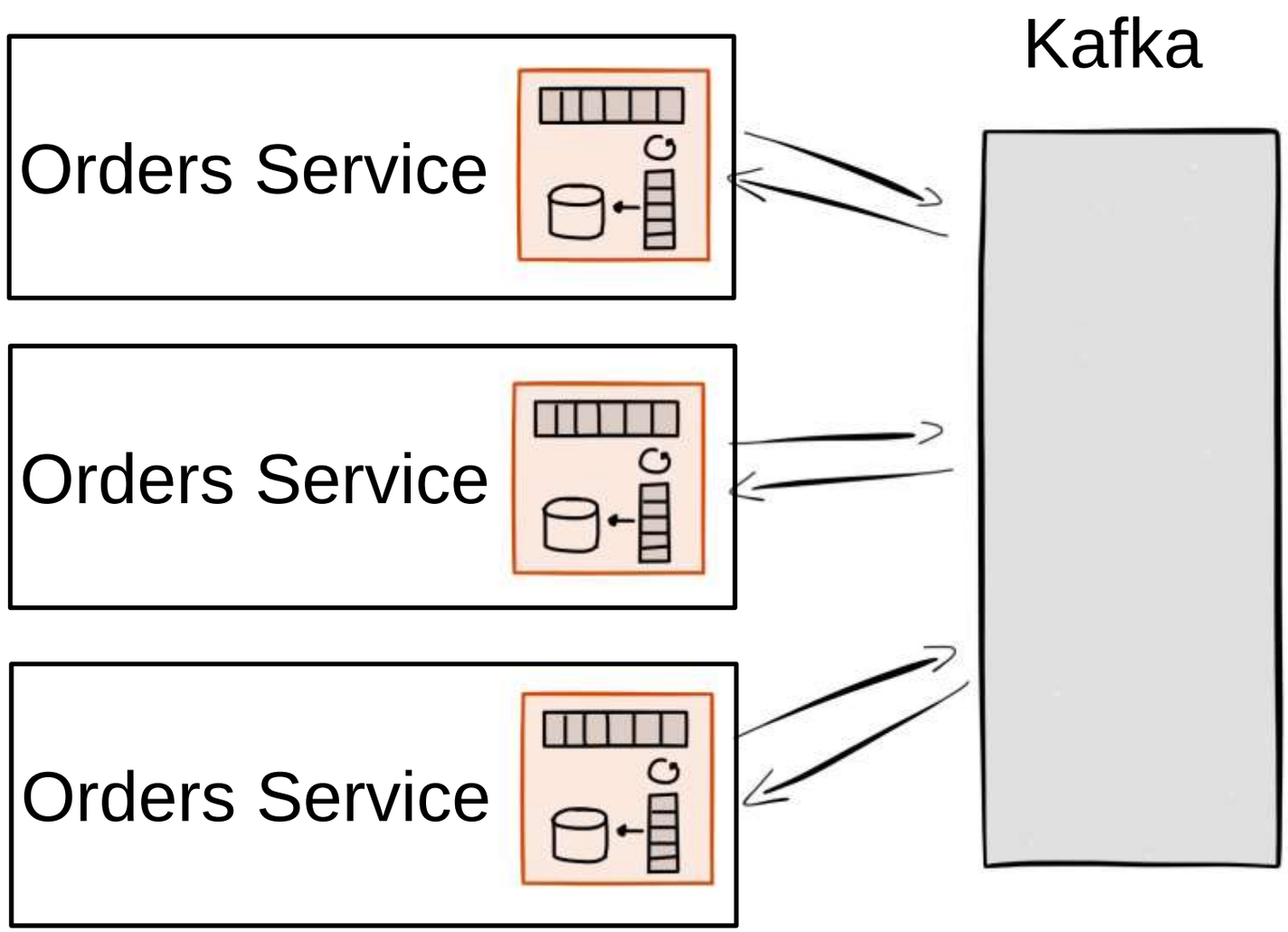
Useful for Enrichment



Scales Out



Embeddable



Kafka Connect

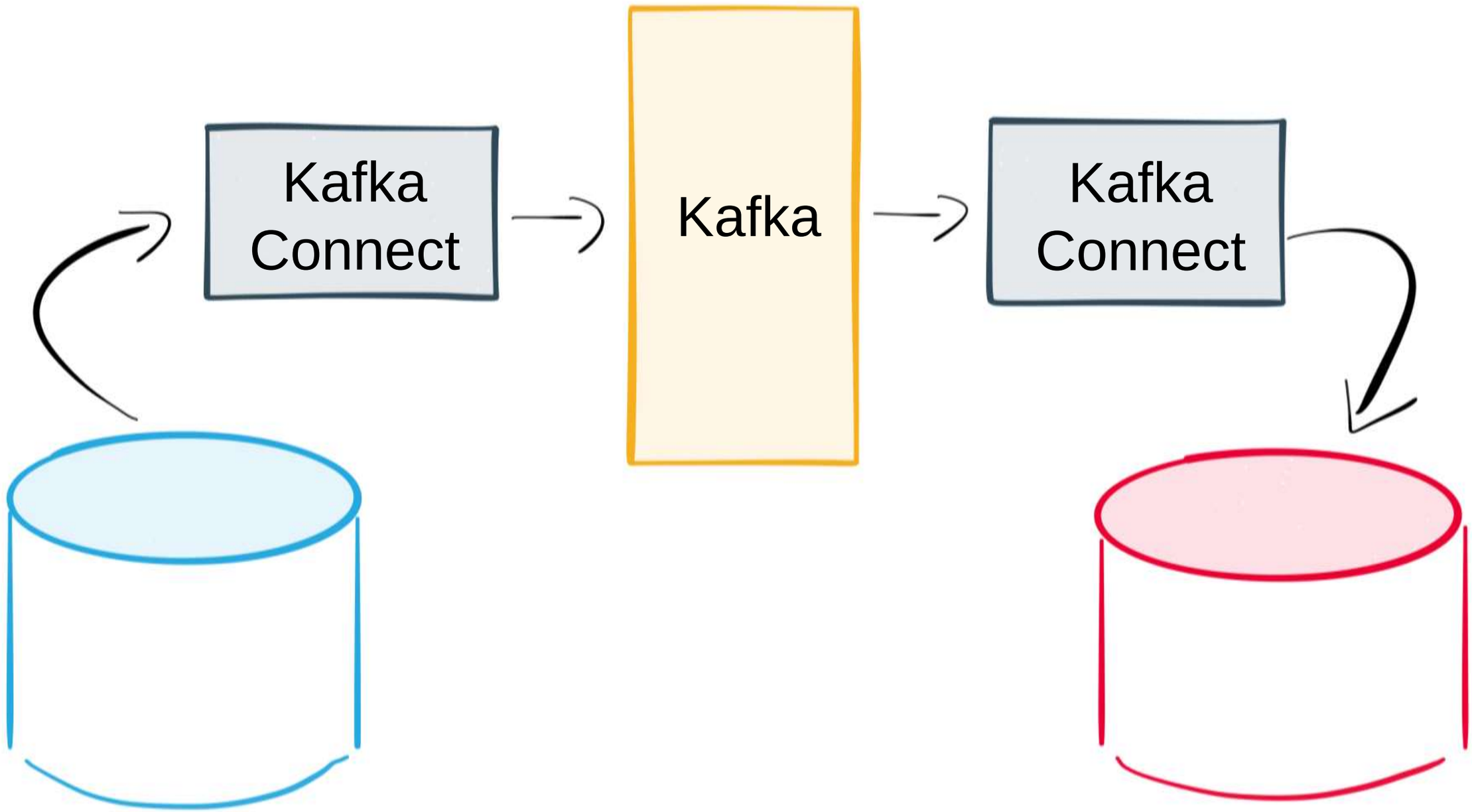
View Replication

Sometimes you need to physically move data

Replicate it, so both copies are identical

Iterate via regeneration

Kafka Connect



So...

Service Backbone

Scalable, Fault Tolerant, Concurrent, Strongly Ordered, Stateful

Embeddable tool for data manipulation

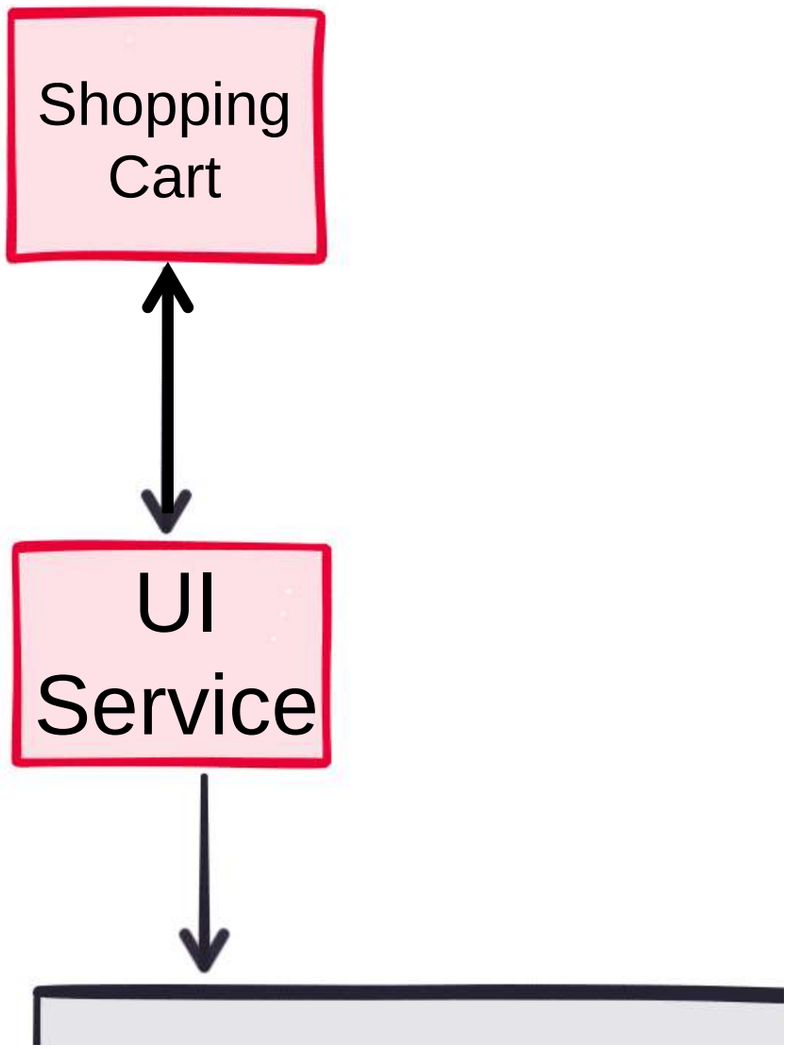
Adapt data streams (transformation, stream maintenance, analytic function)

Replicate Data Sources Exactly

Create Regenerable, Streaming Views

How do we actually do this?

10 (opinionated) principals for Streaming Services

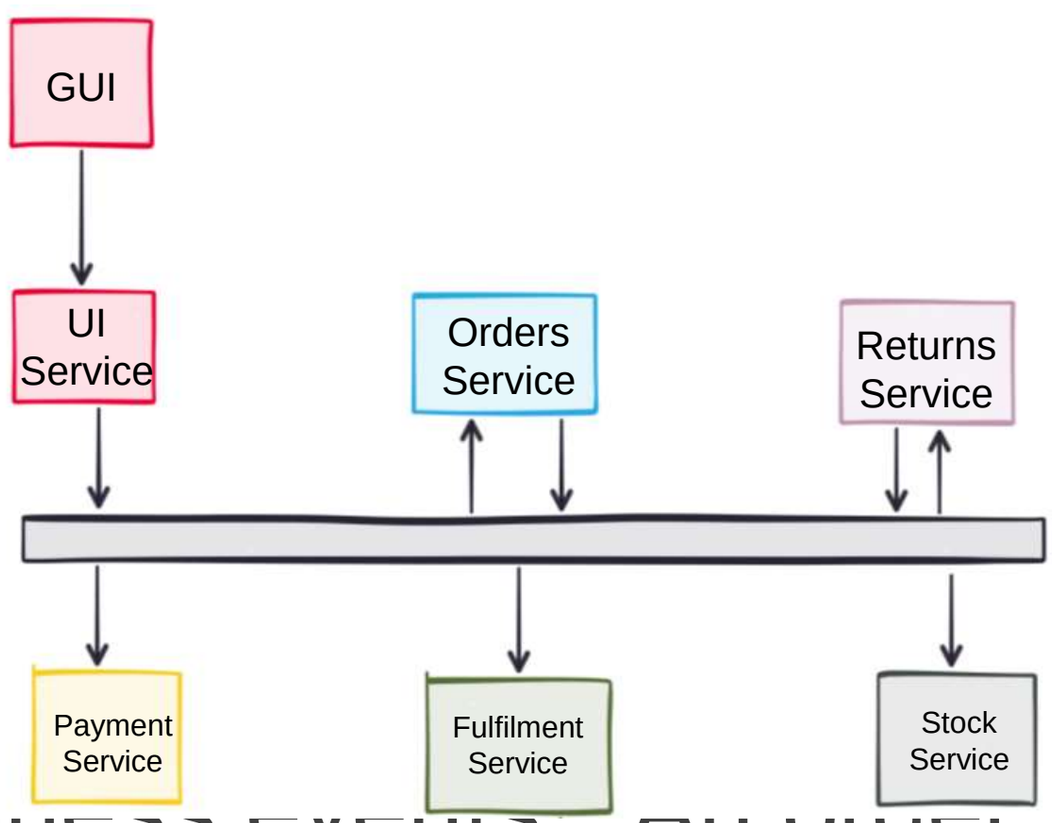


1. Don’t use Kafka for shopping carts!

(OK, you can, but use sparingly)

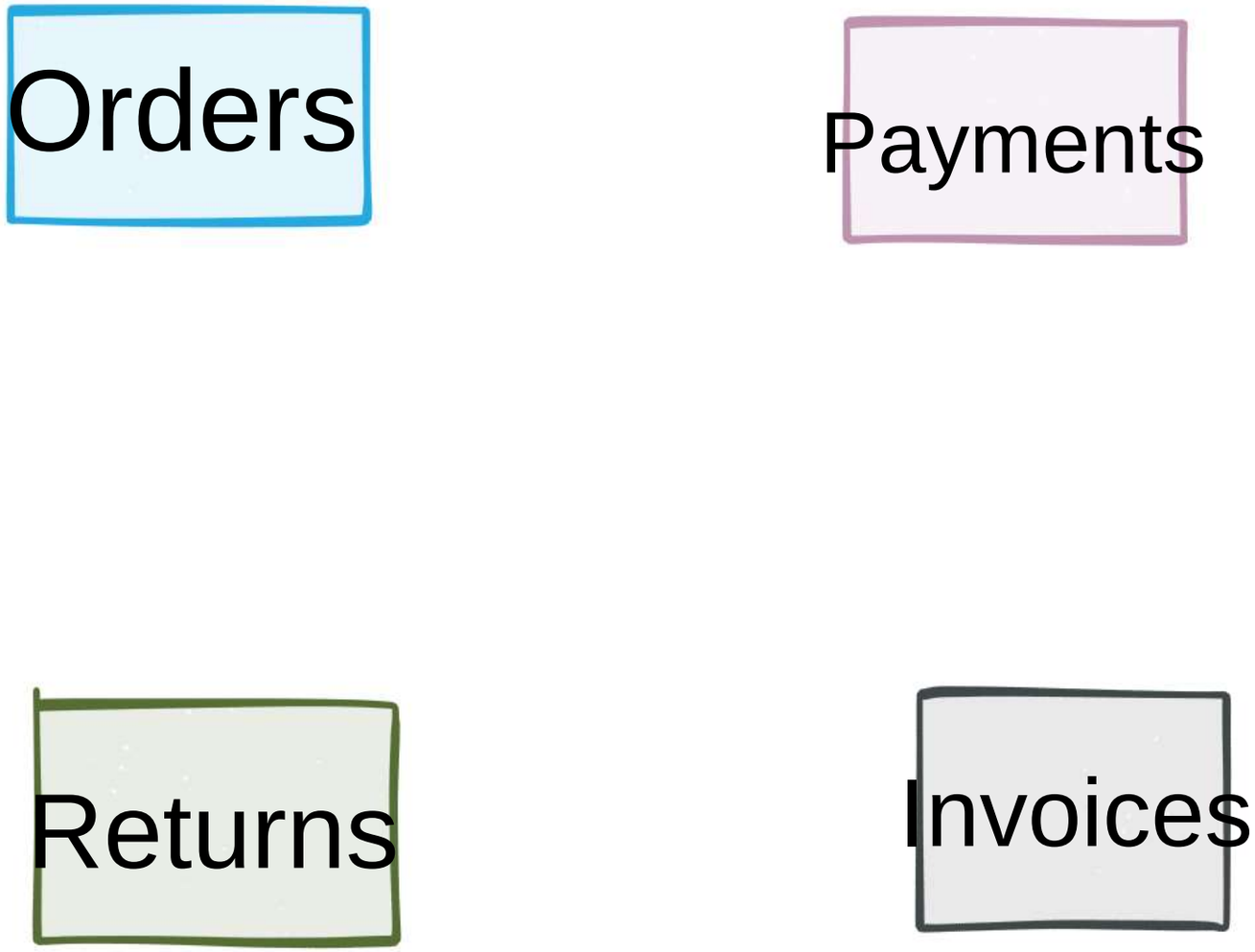
Broker/durability/broadcast add little to request response

Do use Kafka for event driven archtectures.



Think “business events”. An order was created, a payment was received, a trade was booked etc. Pub/Sub.

2. Pick Topics with Business Significance



Give your messages meaningful IDs and version them

rdersService1-Order-1234-

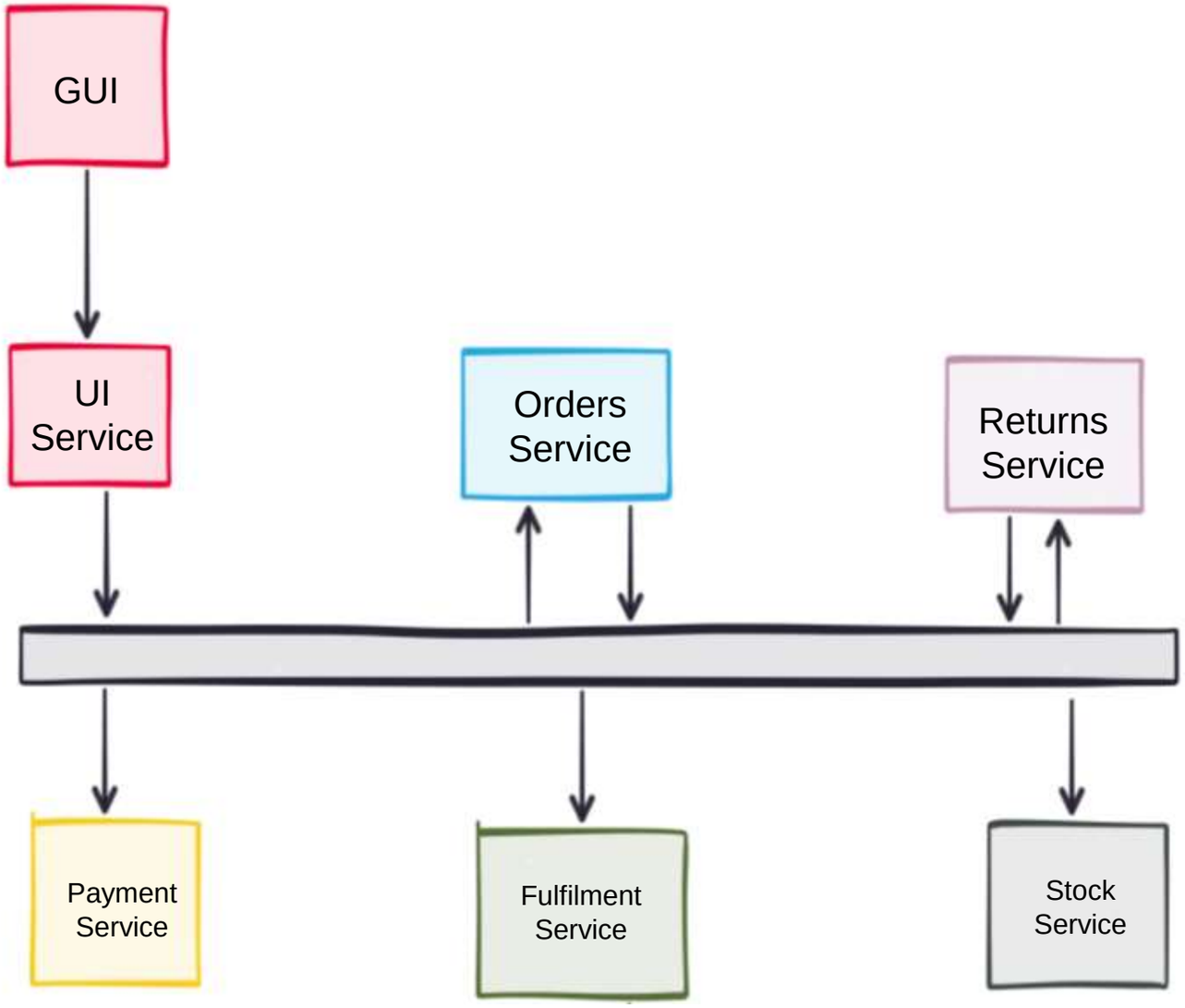
Include the service name

Should relate to the real world

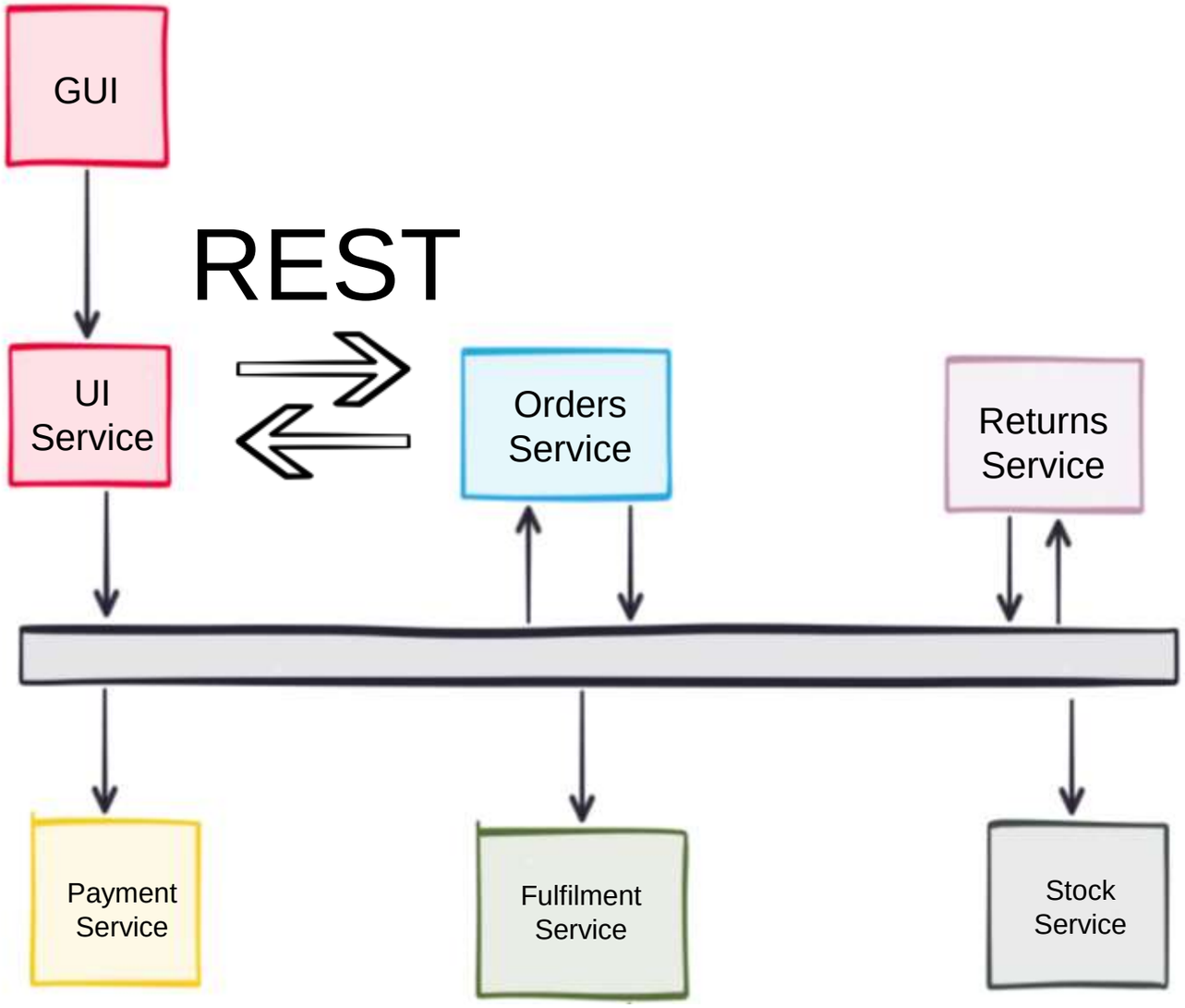
Should be Versioned (if mutable)

Note the key used for sharding in Kafka may not be this key

3. Decouple publishers from subscribers

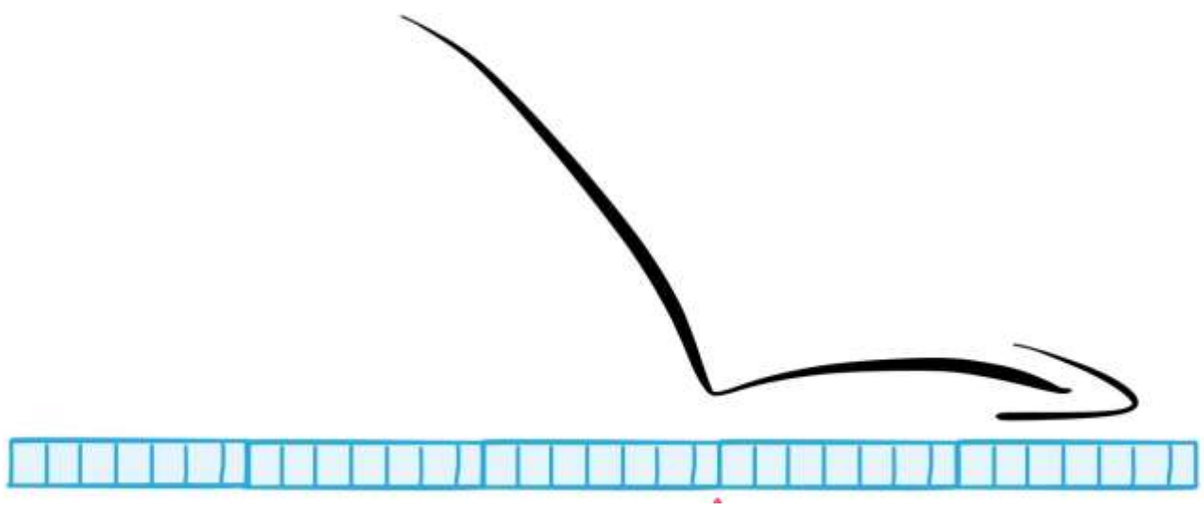


Add Request/Response only where needed

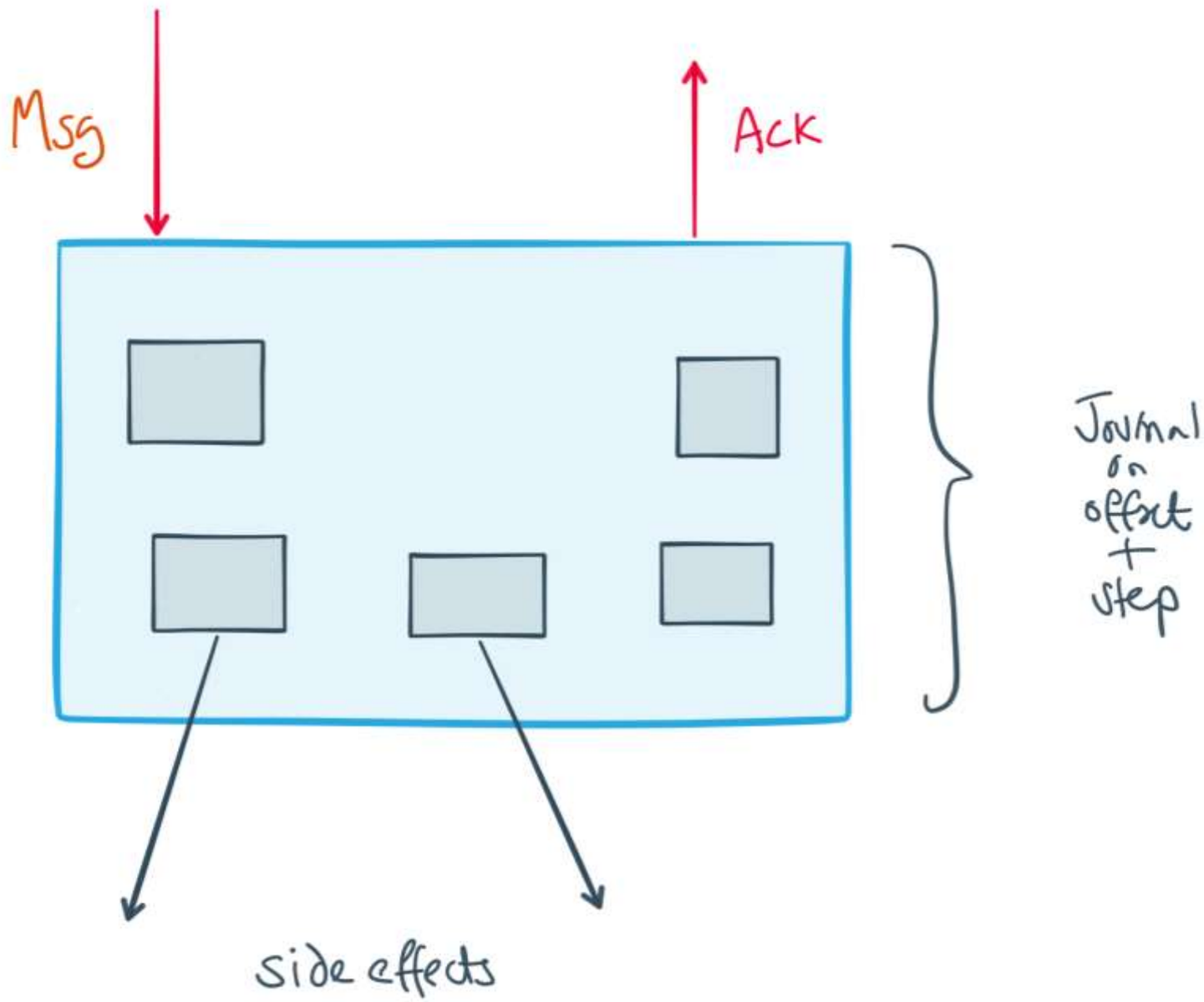


4. Use the log to regenerate state

Avoid journaling incoming events

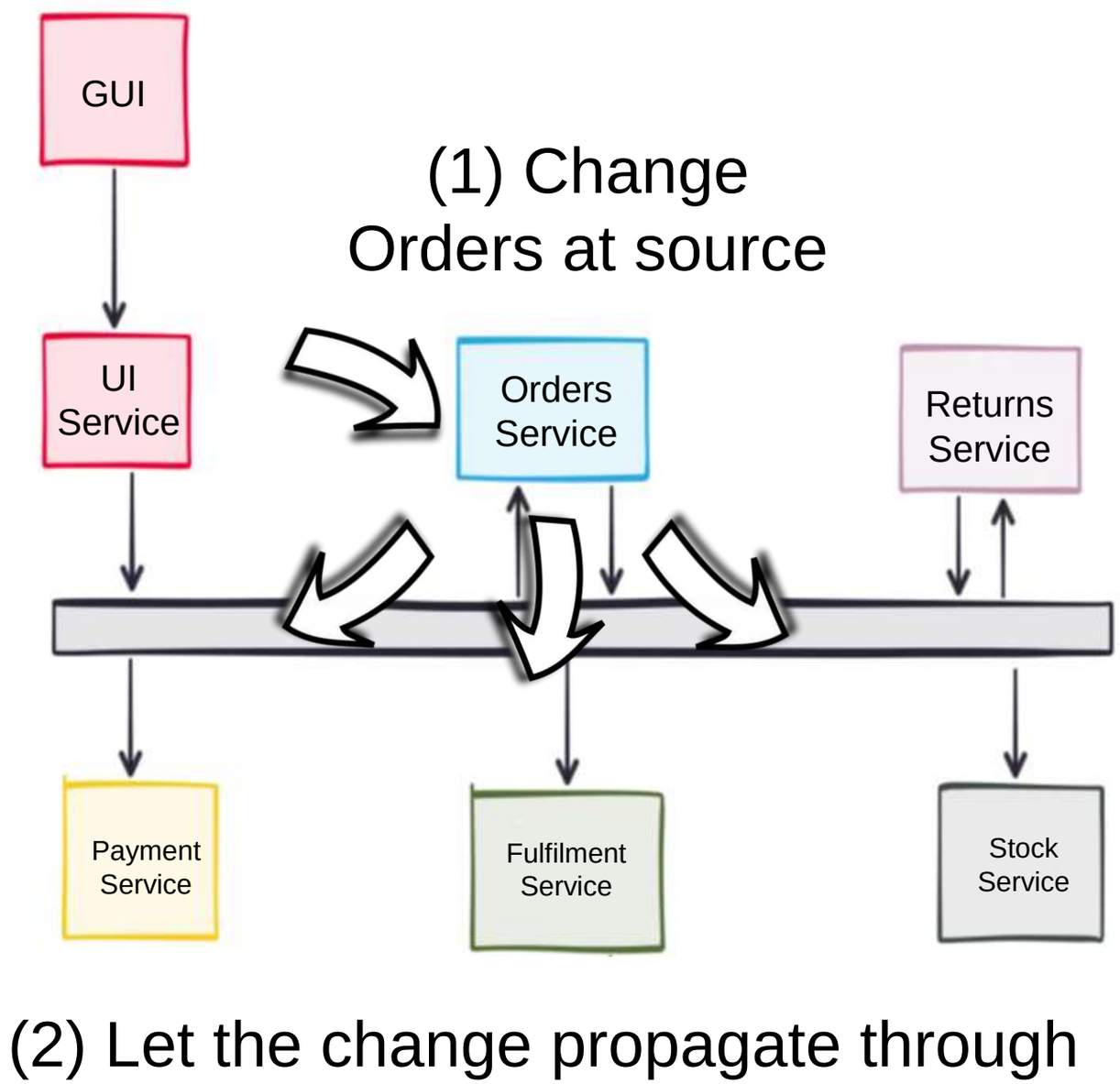


Event Source side effects



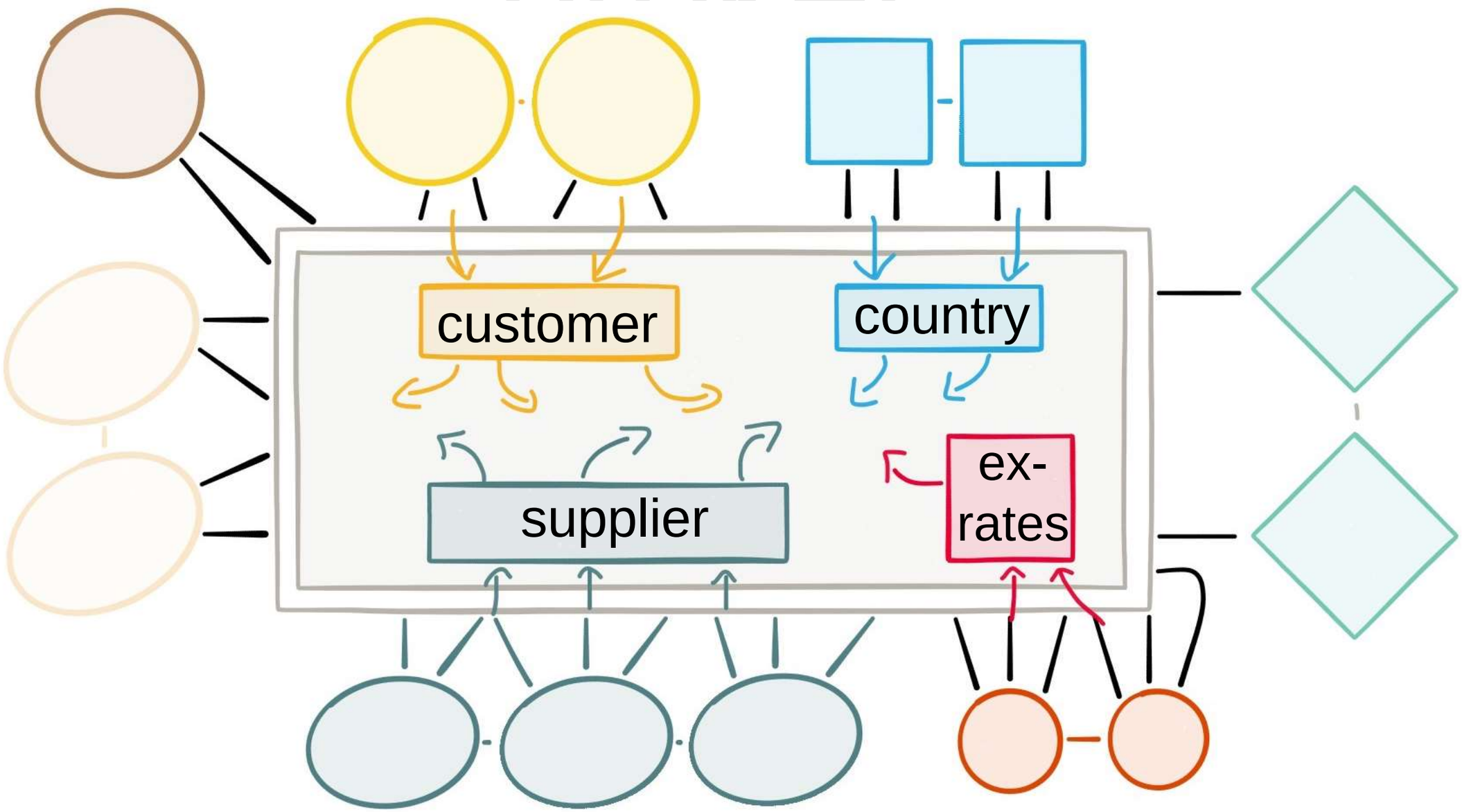
- Use offsets to tie these back to the stream
- Store in:
  - Kafka
  - Kstreams state store
  - Other DB

5. Apply the Single Writer Principal

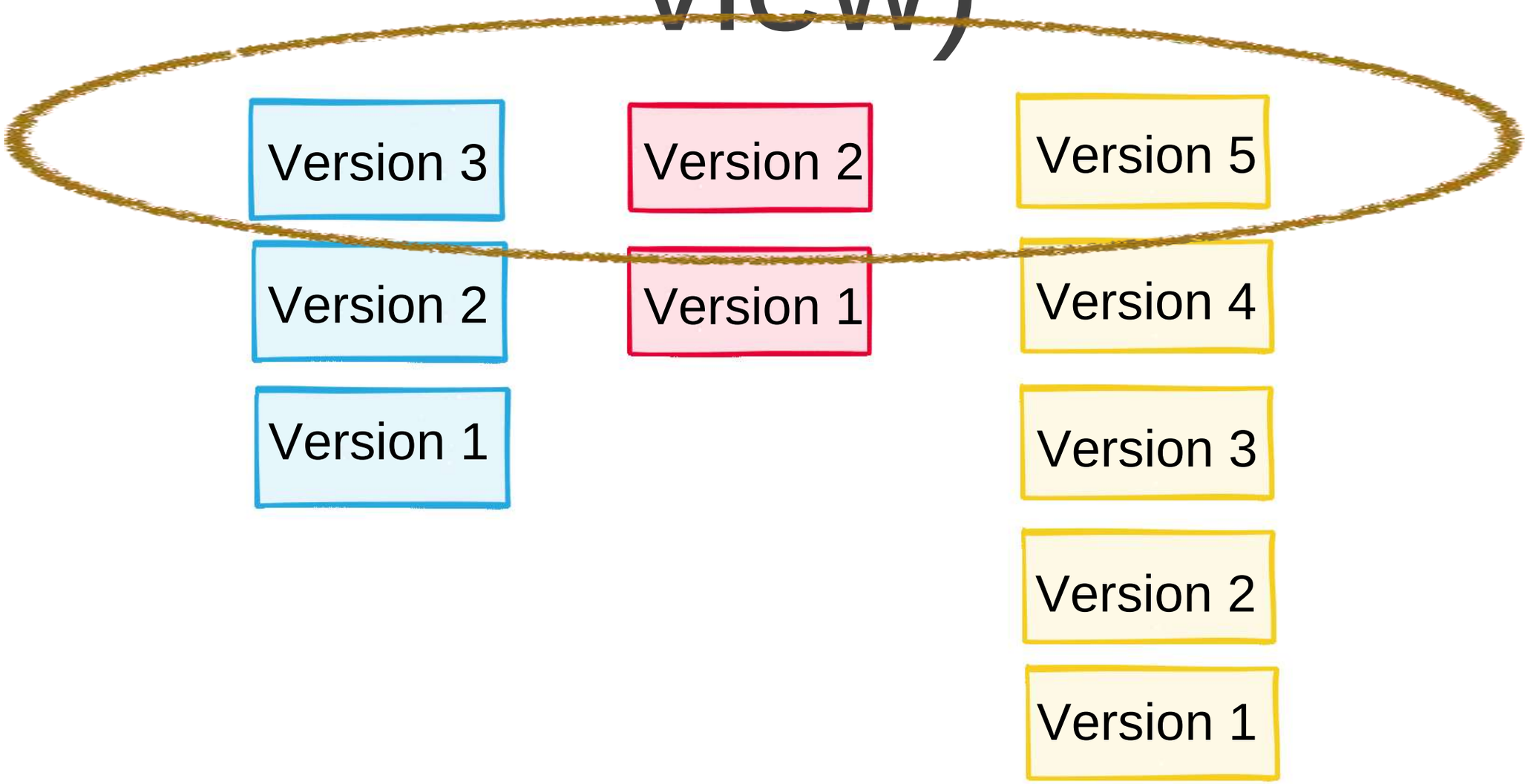


- Change at source (by calling that service)
- Let the change propagate back
- Keep local copies read only.

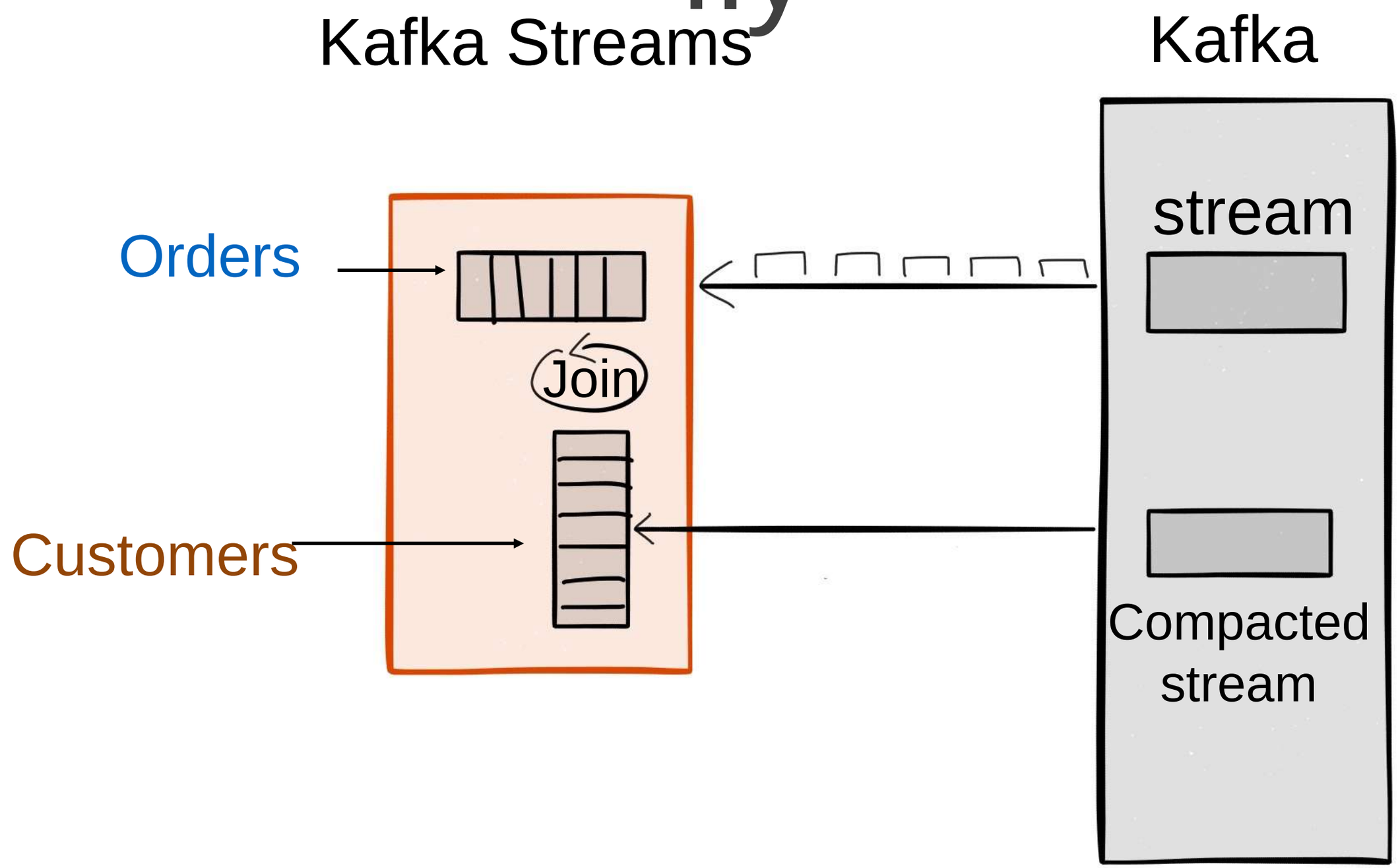
6. Leverage keeping datasets inside the broker



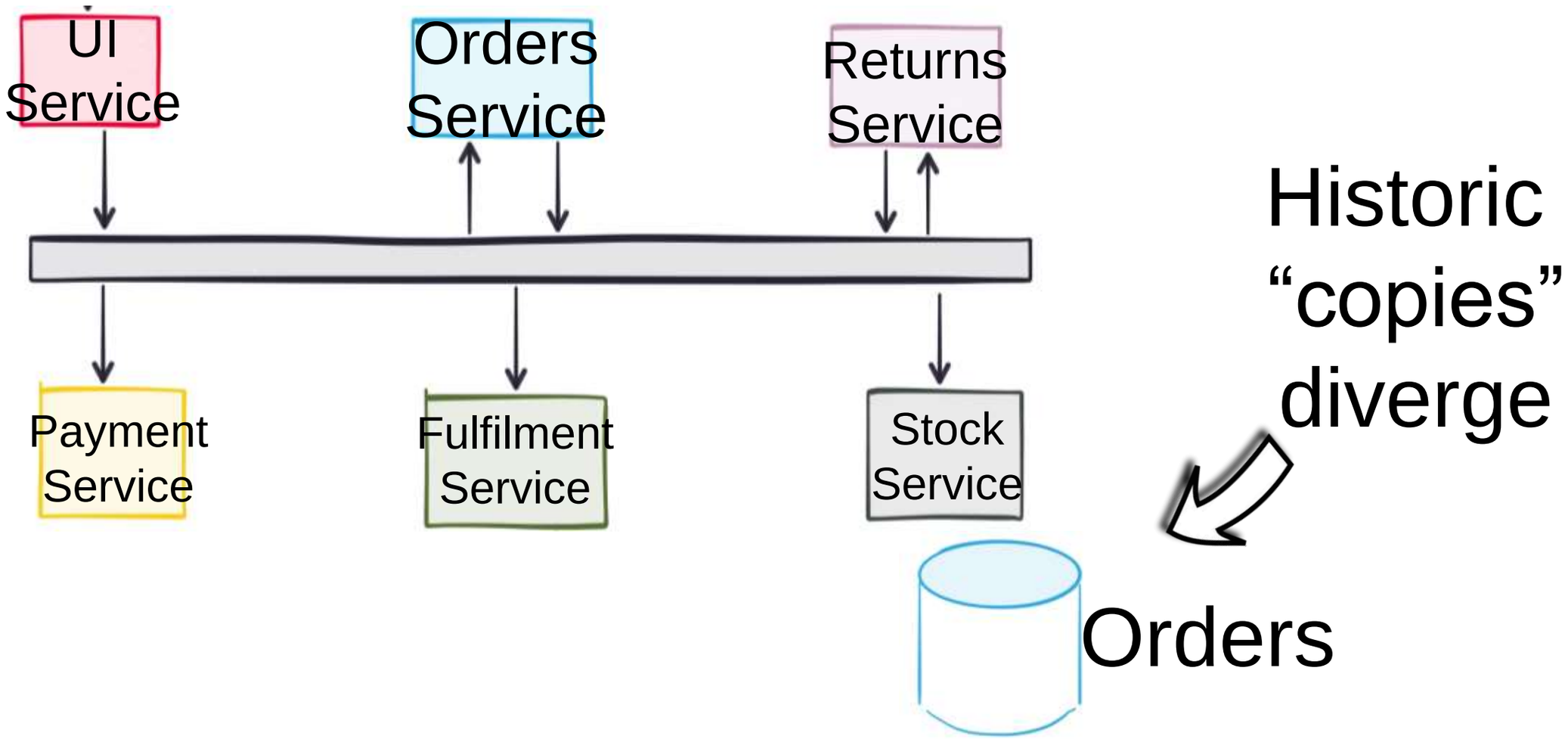
Leverage keeping only the latest version (table view)



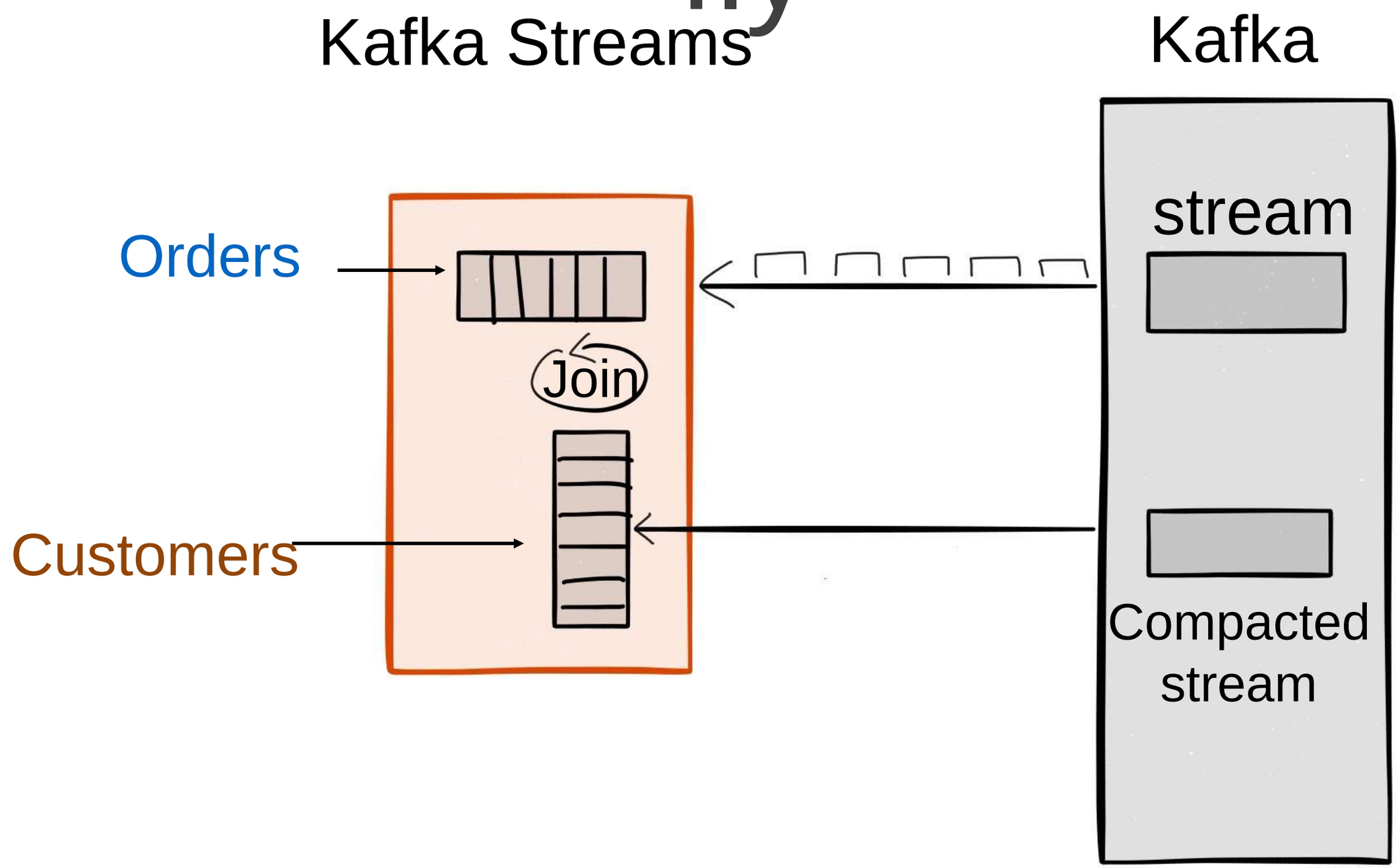
Join & Process on the fly



7. Prefer stream processing over maintaining historic views



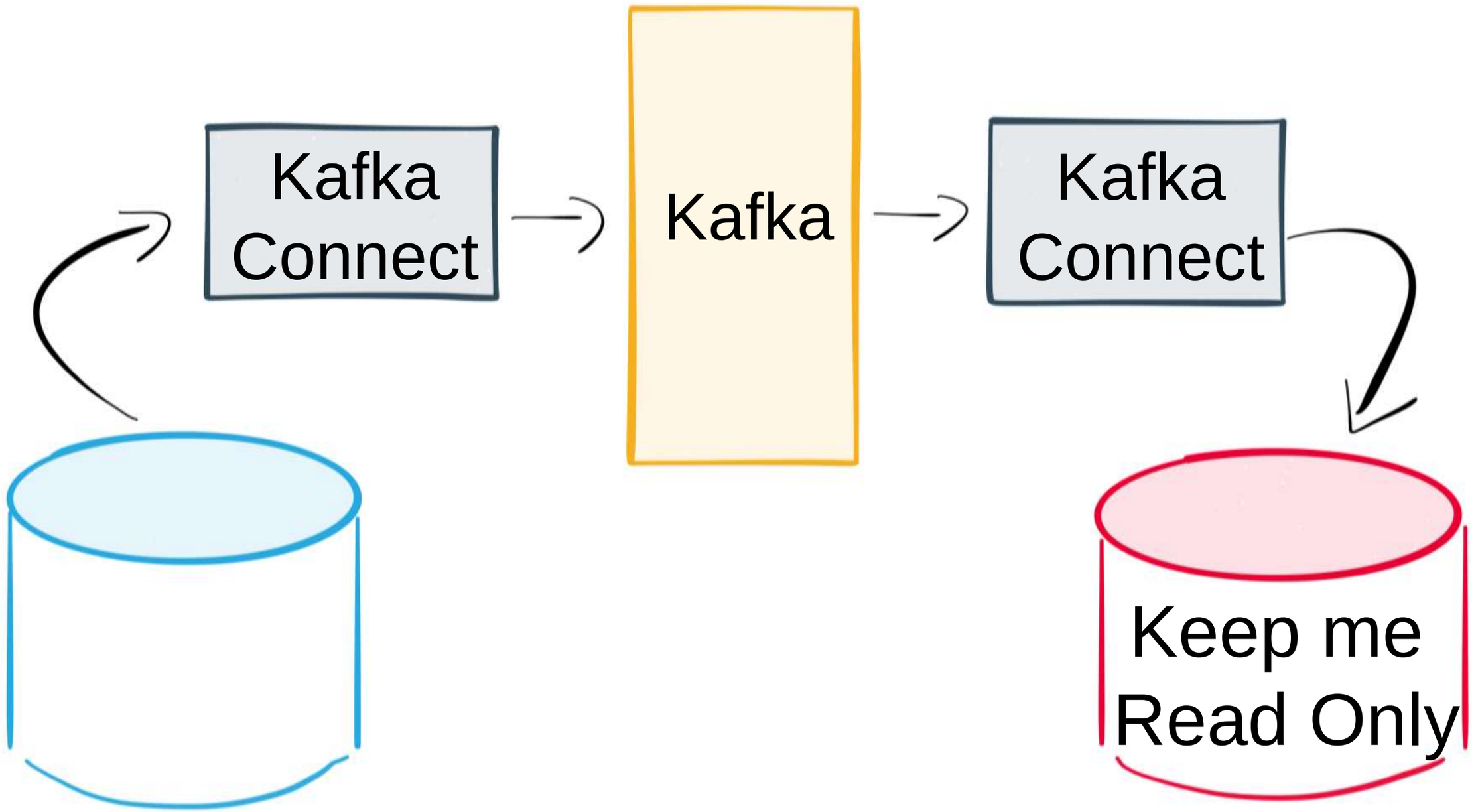
Join & Process on the fly



8. Sometimes you need historic views.

=> Replicate & Keep Read Only

Replicate



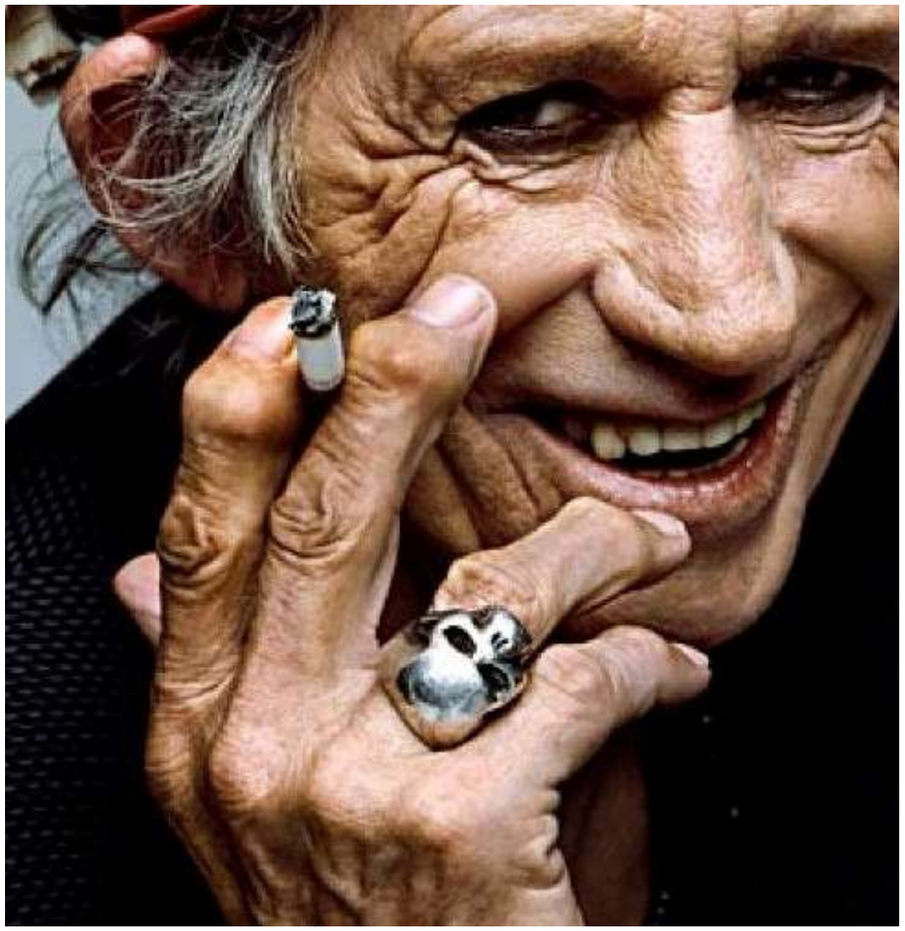
Iterate

Polyglotic Persistence

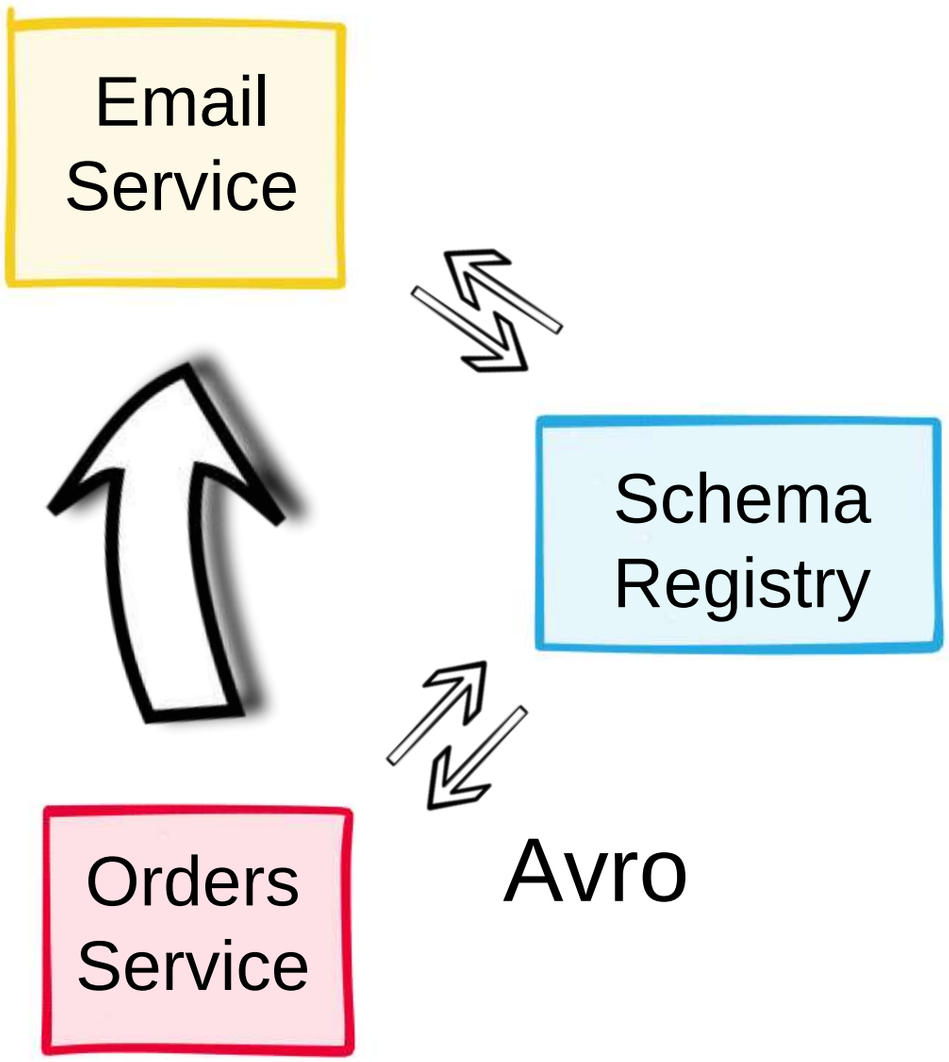
9. Use Schemas

(especially if data is retained)

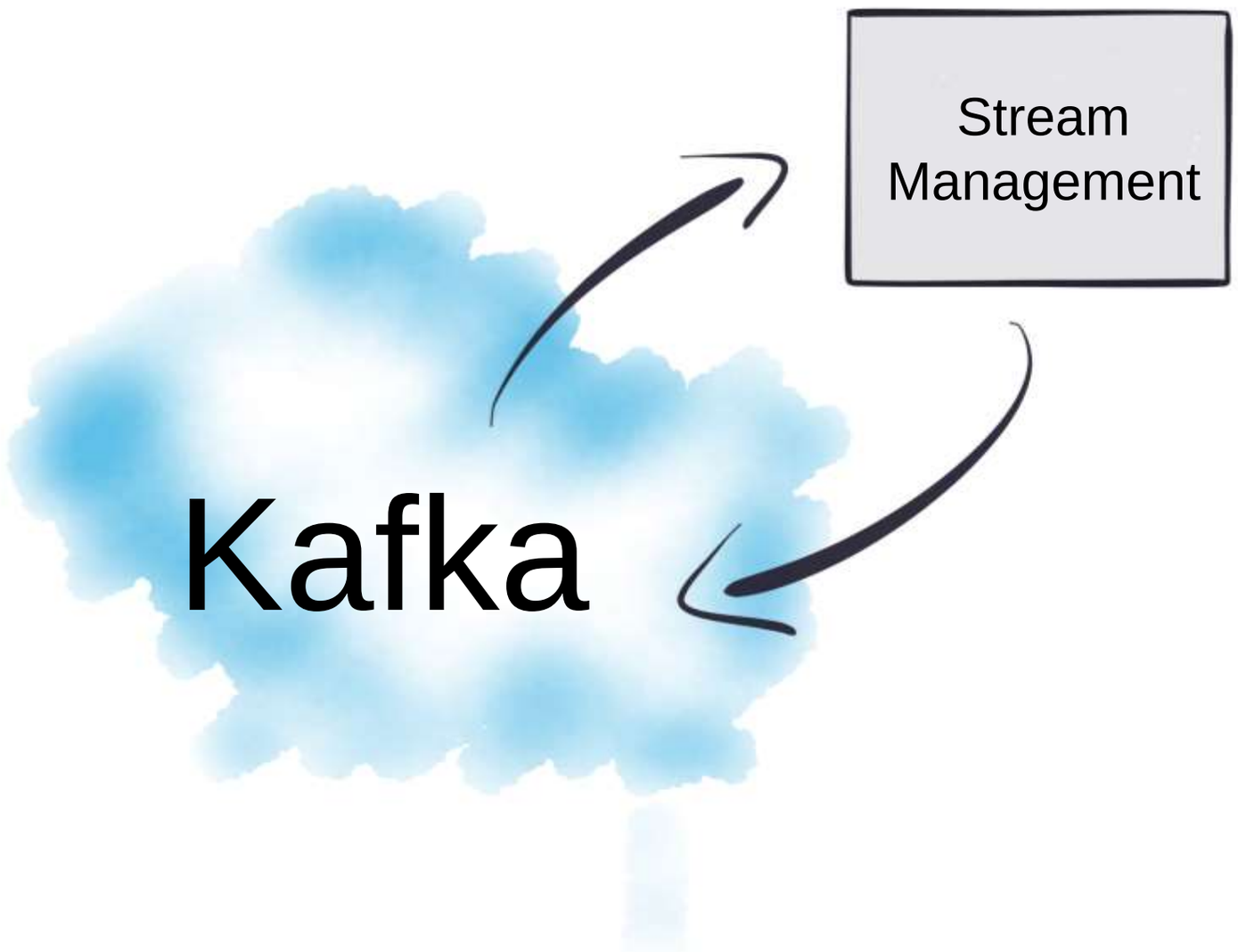
Schemaless data doesn’t age well



Confluent Schema Registry can help

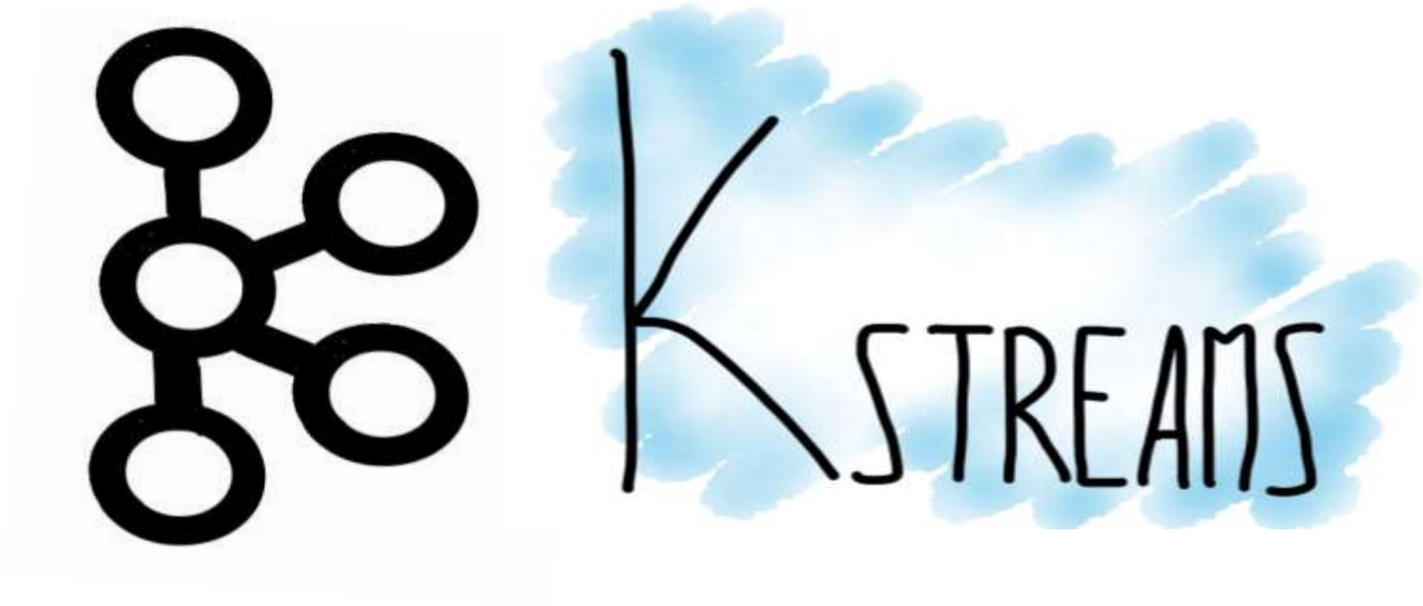


10. Consider “Stream Management” Services

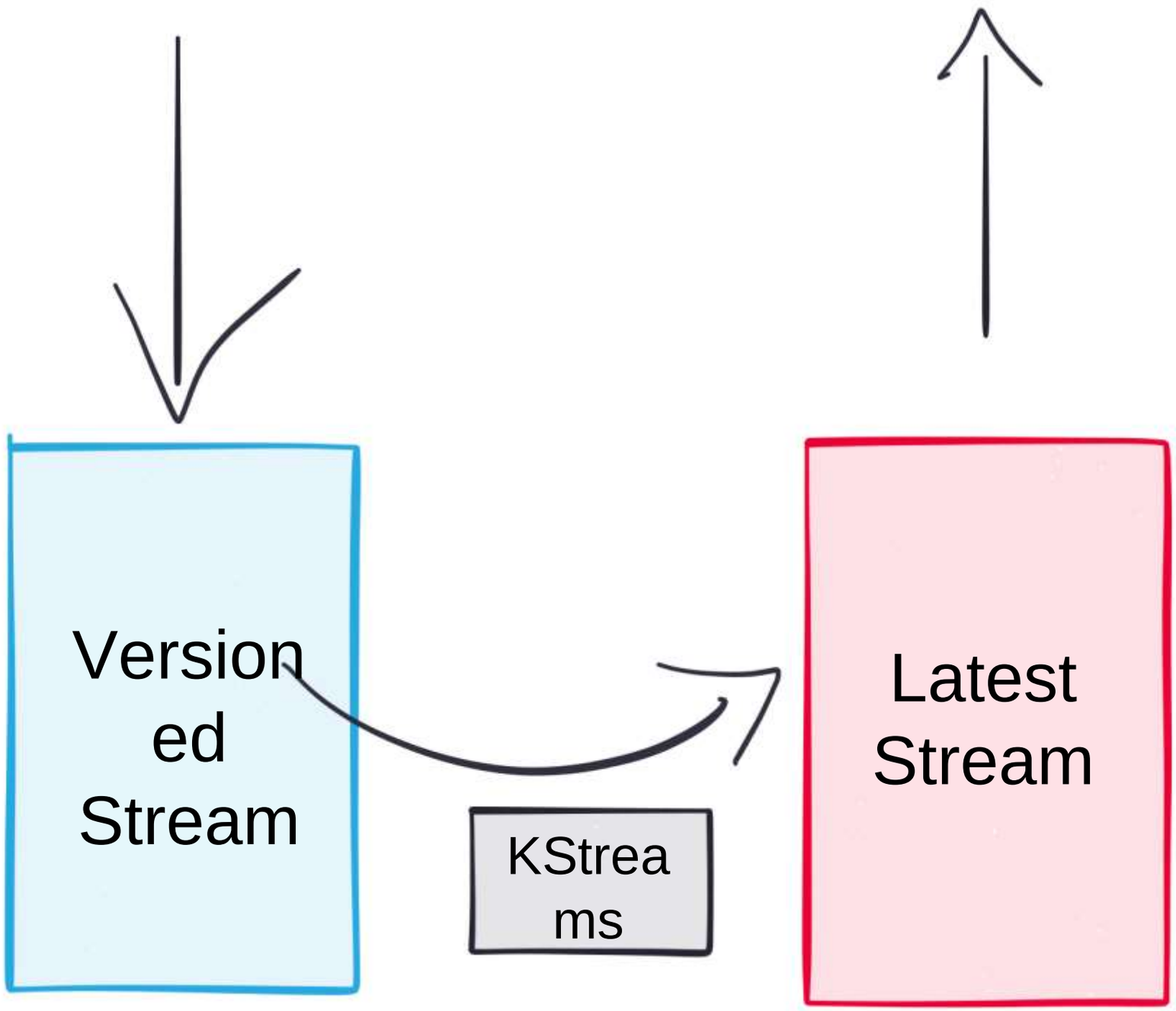


- Retaining data => Admin tasks
- Similar to the role of a DBA
  - Data Migration
  - Repartitioning
  - Latest/versioned
  - Environment Management
  - CQRS

KStreams is a good toolset for this



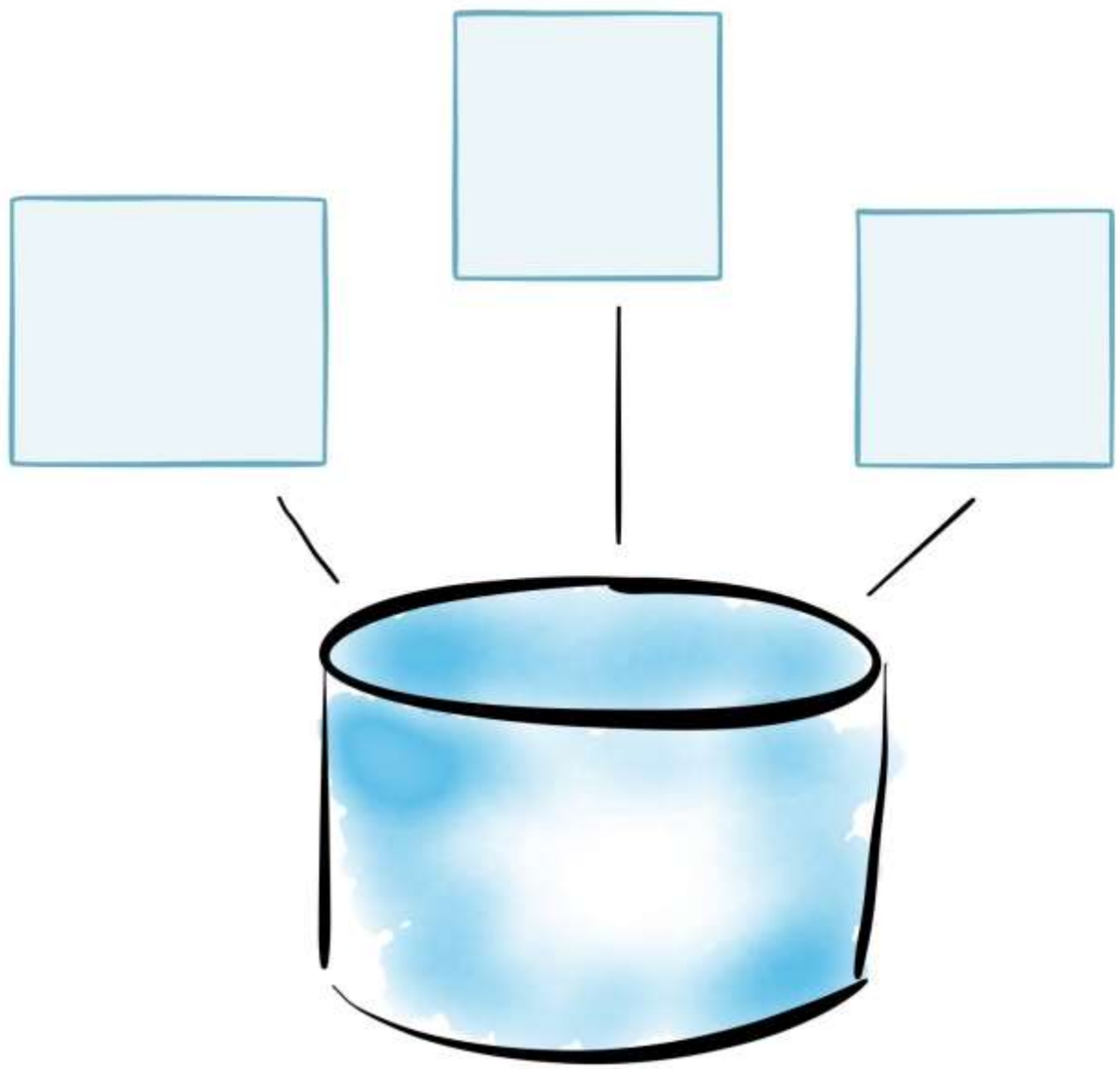
Stream Management



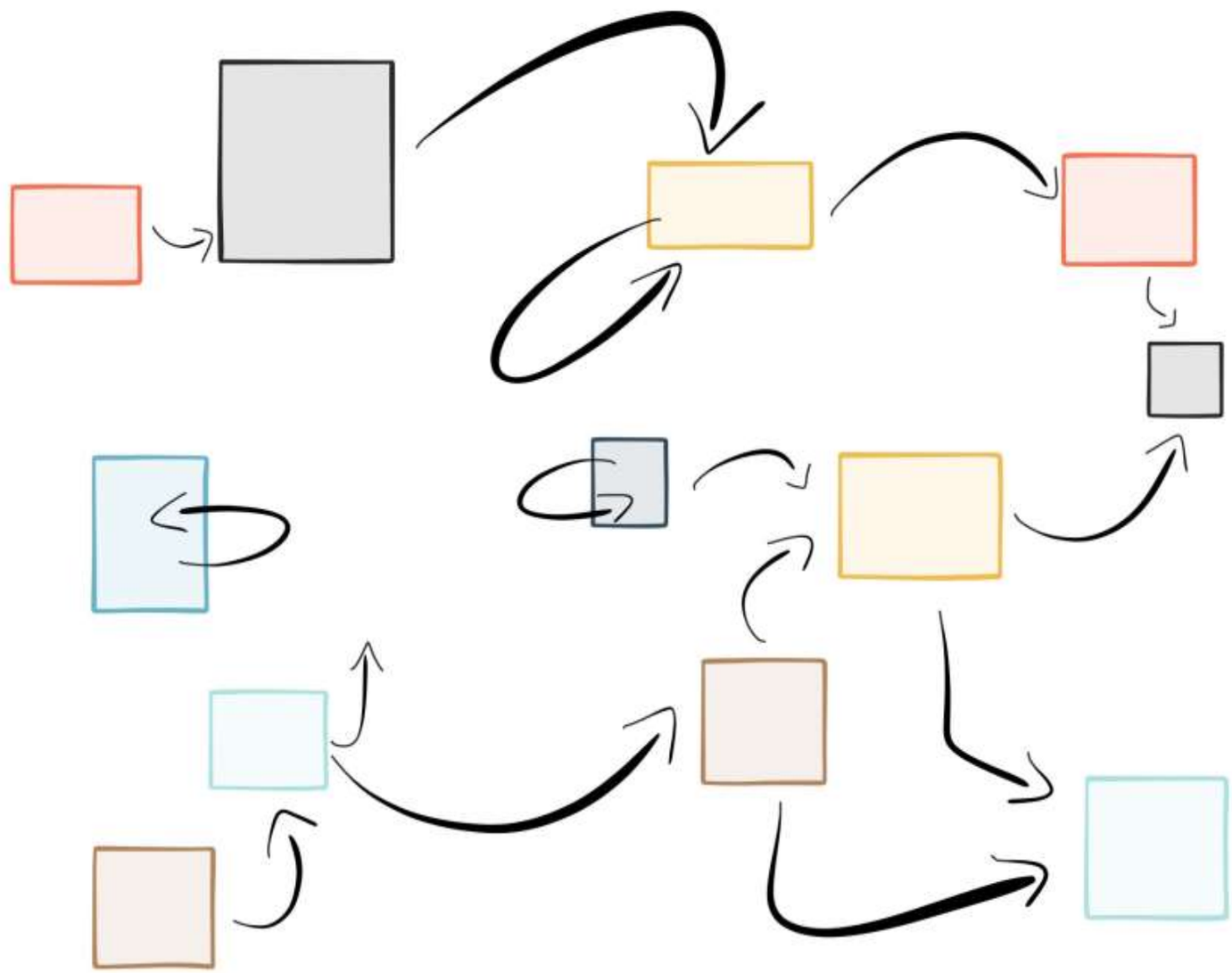
So…

1. Don’t use Kafka for shopping carts!
2. Pick Topics with Business Significance
3. Decouple publishers from subscribers
4. Use the log to regenerate state
5. Apply the Single Writer Principal
6. Leverage keeping datasets inside the broker
7. Prefer stream processing over maintaining historic views
8. Sometimes you need historic views. => Replicate Read Only
9. Use Schemas
10. Consider “Stream Management” Services

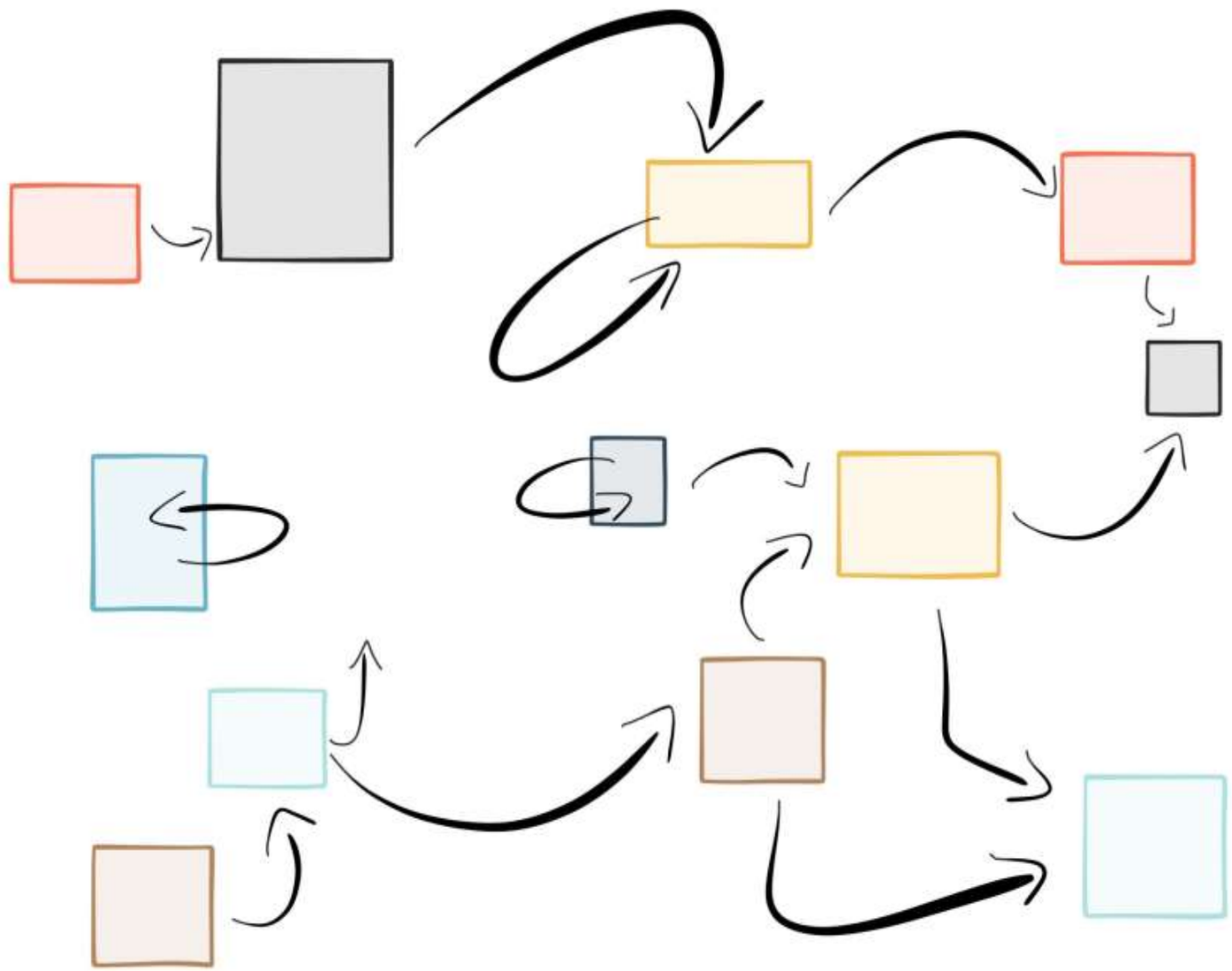
Microservices push us away from shared, mutable state



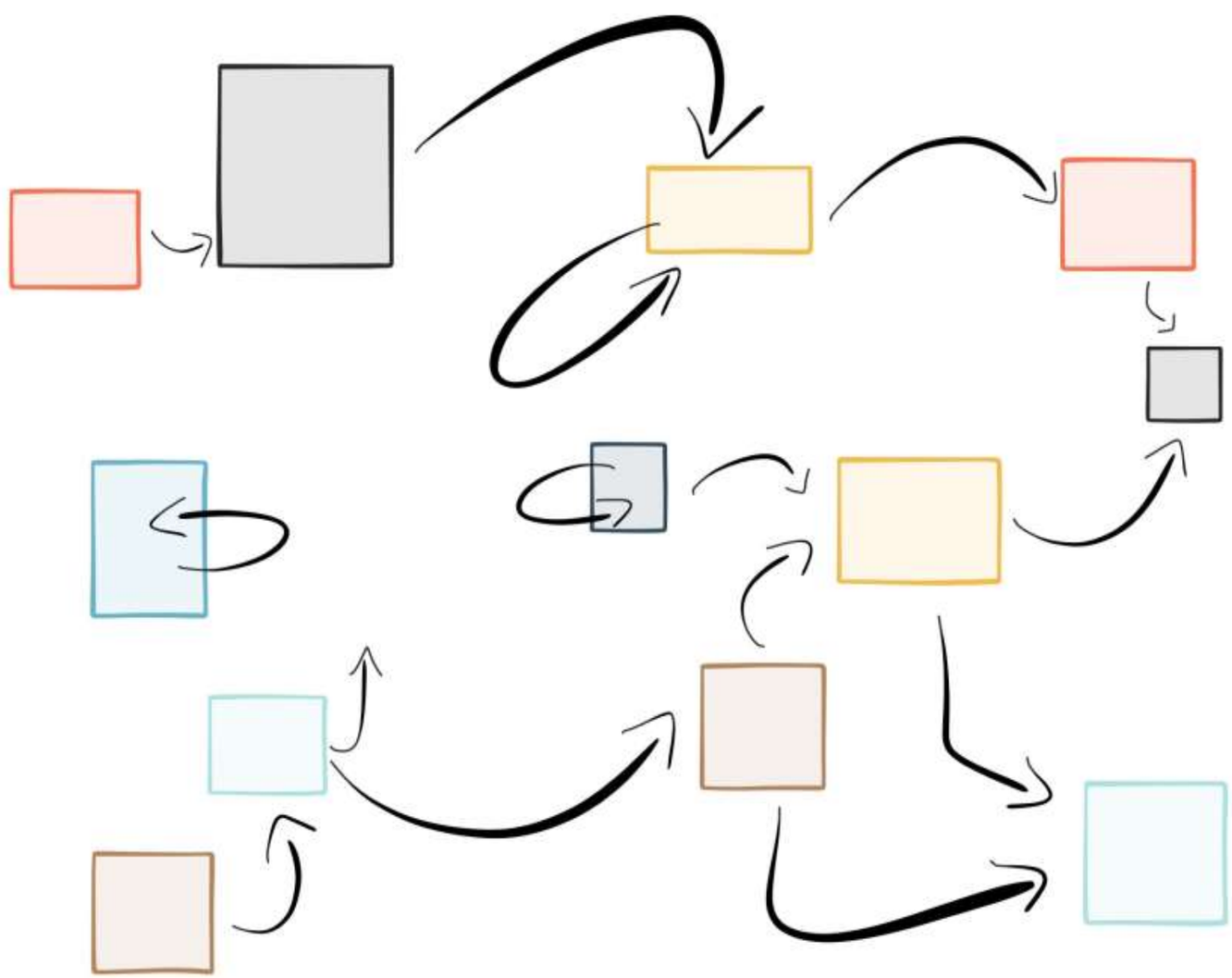
But state needs to be communicated



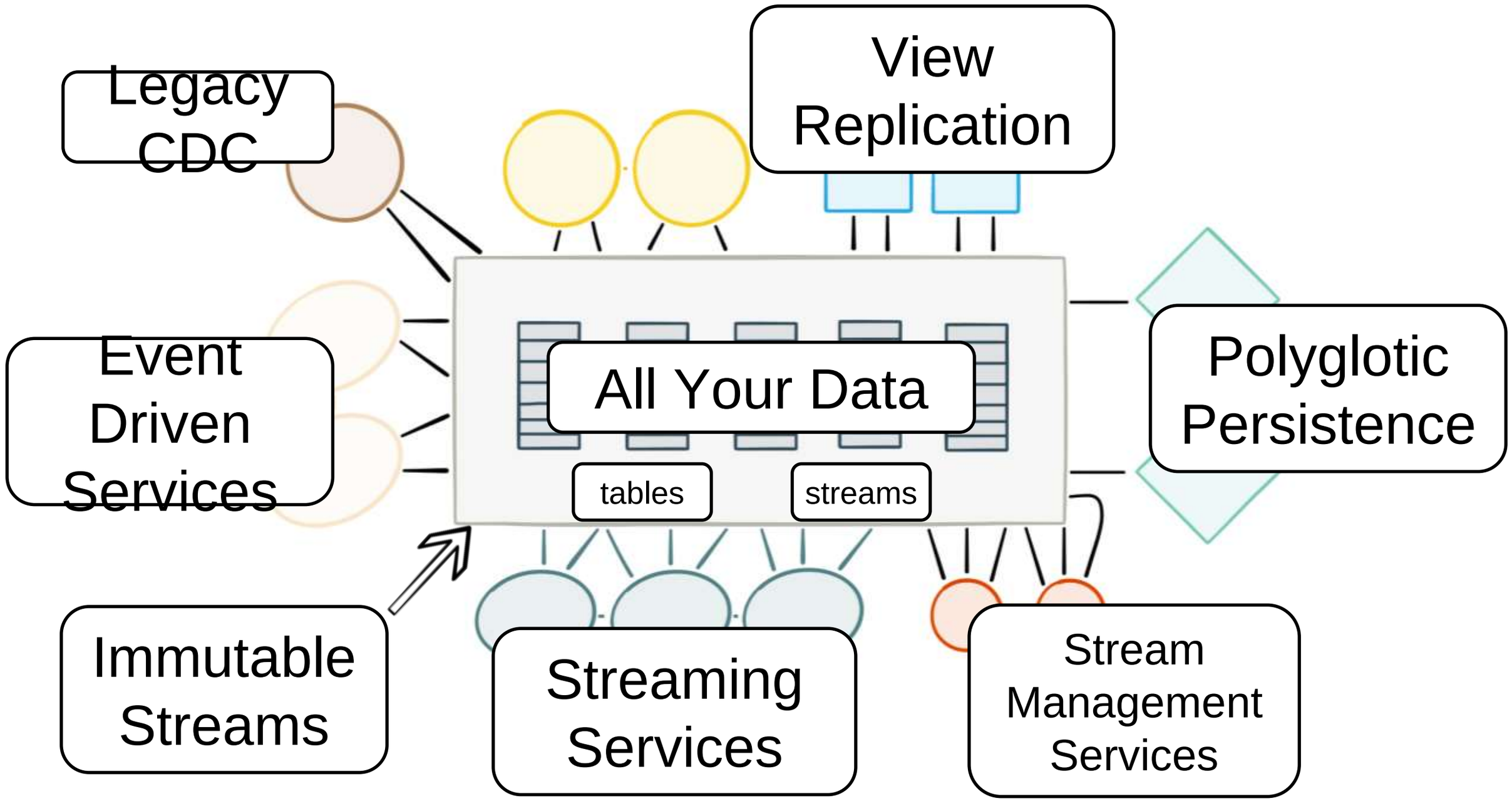
In an increasingly data-heavy world we need tools to do this efficiently



…and in real time.



We need a data-centric toolset to do this



Keep it simple,
Keep it moving

Thanks!

@benstopford