Building event-driven Microservices with Kafka Ecosystem

Guido Schmutz
Zurich, 8.3.2018

@gschmutz

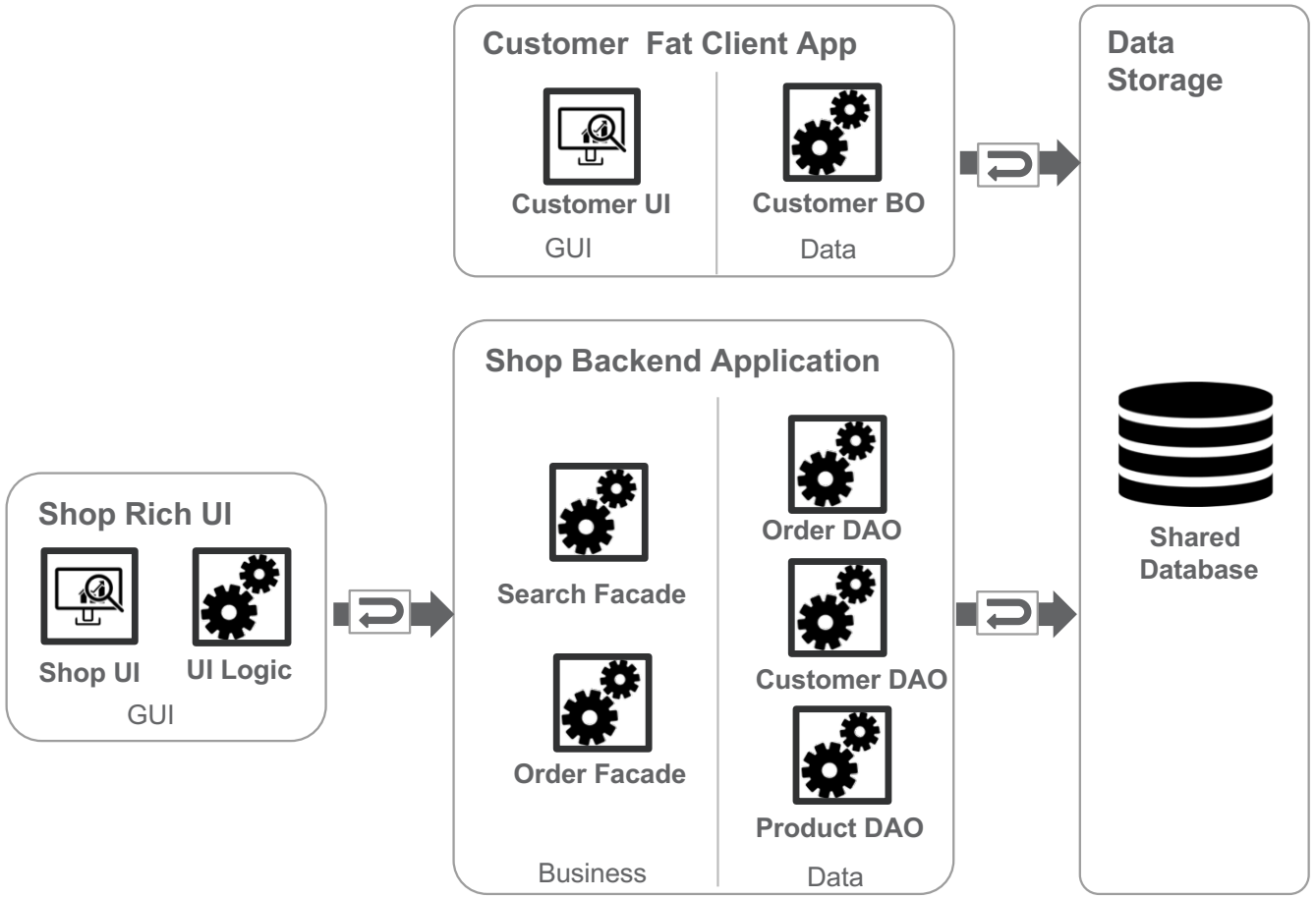
guidoschmutz.wordpress.com

Agenda

1. Where do we come from?
2. What are Microservices?
3. Why not Event Driven?
4. What about streaming sources?
5. What about integrating legacy applications?
6. CQRS and Event Sourcing
7. What about (historical) data analytics?
8. Why Kafka for Event-Driven Microservices?
9. Summary

Where do we come from?

Traditional Approach



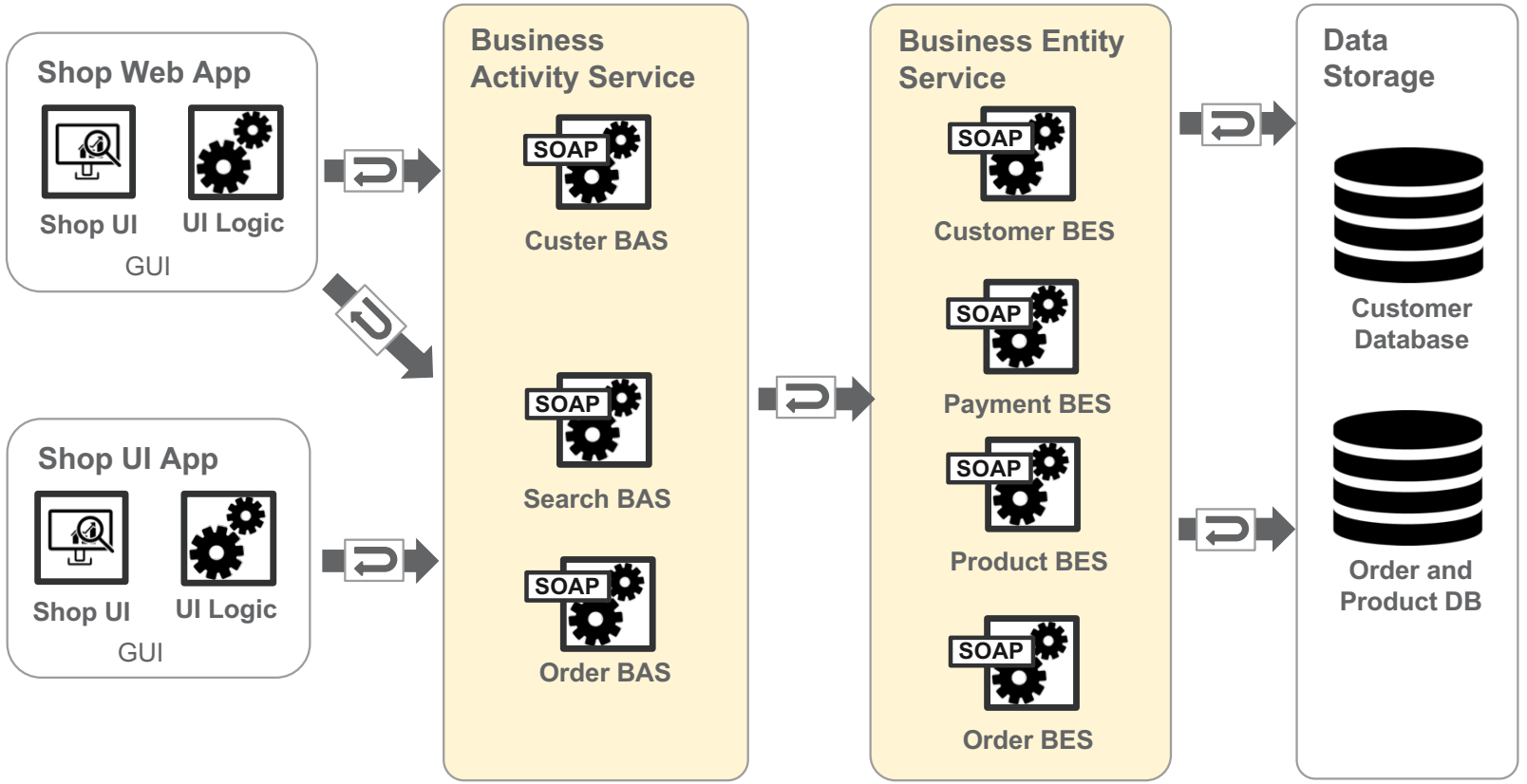
SOA Approach

Contract-first Web Services

Technical layers offer their own interfaces

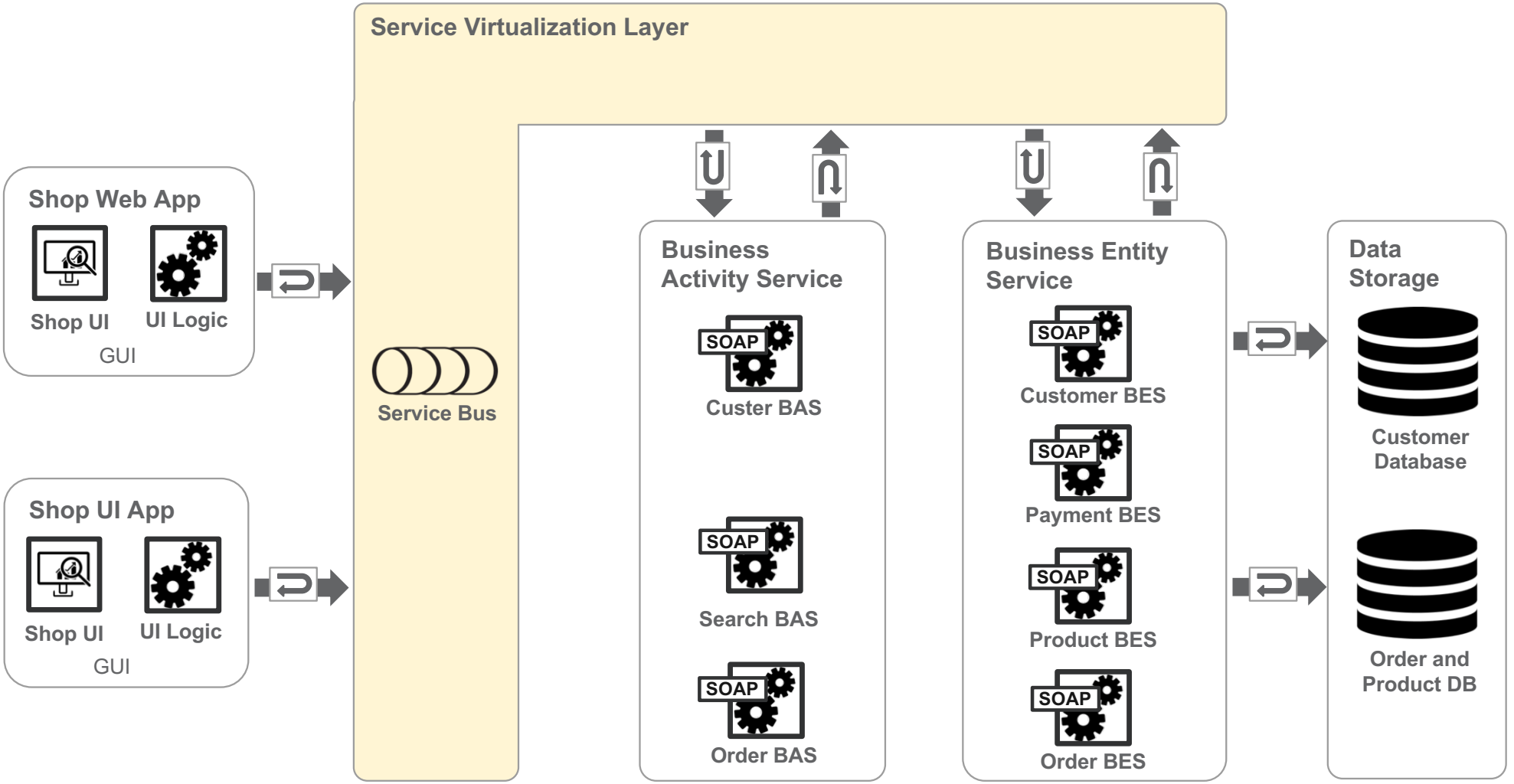
Reuse on each level

Lower layer often wraps legacy code



trivadis makes IT easier.

Virtualized SOA Approach



What are Microservices?

What are Microservices?

Tightly Scoped behind clear interfaces

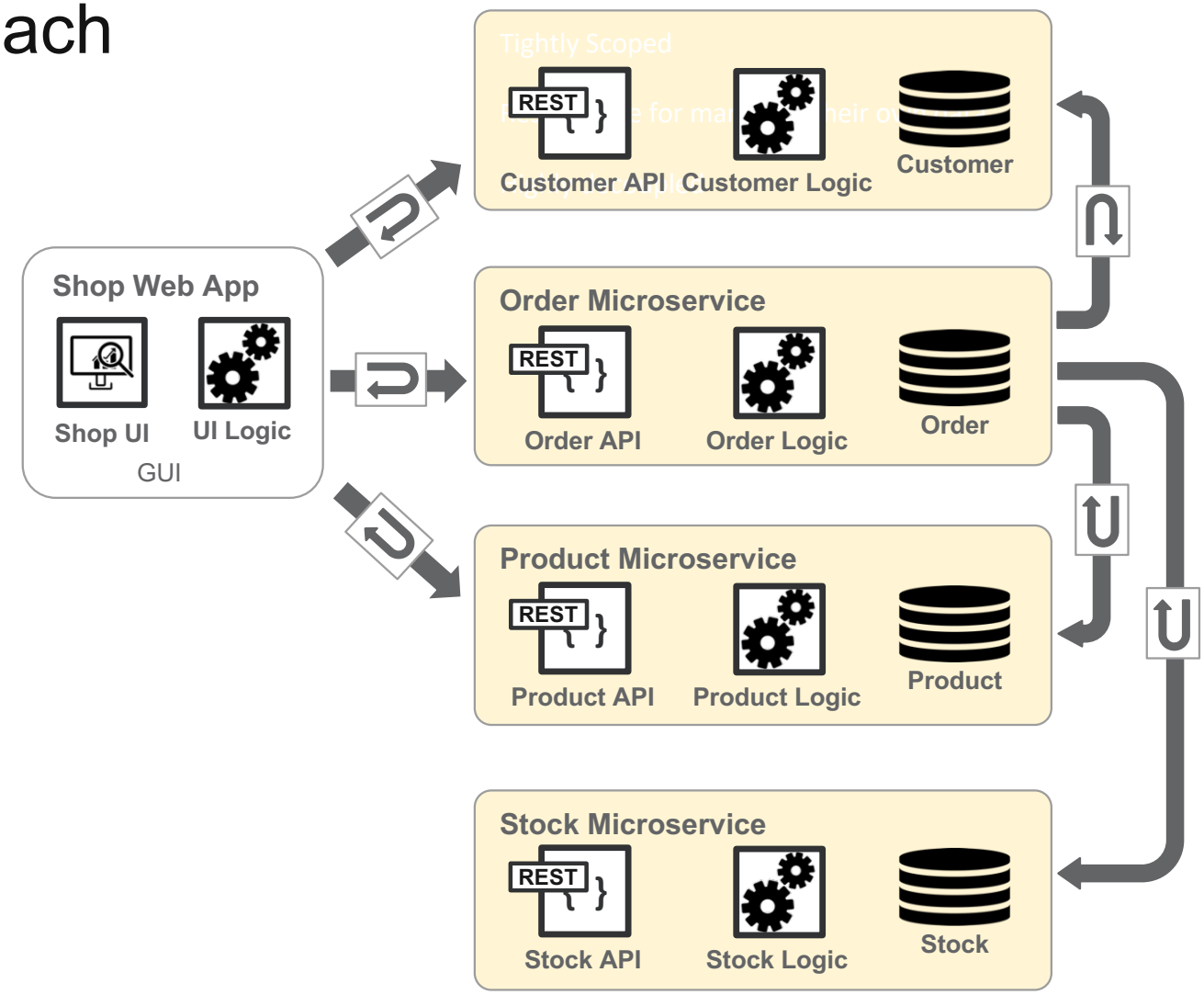
Responsible for managing their own data (not necessarily the infrastructure)

Should be highly decoupled

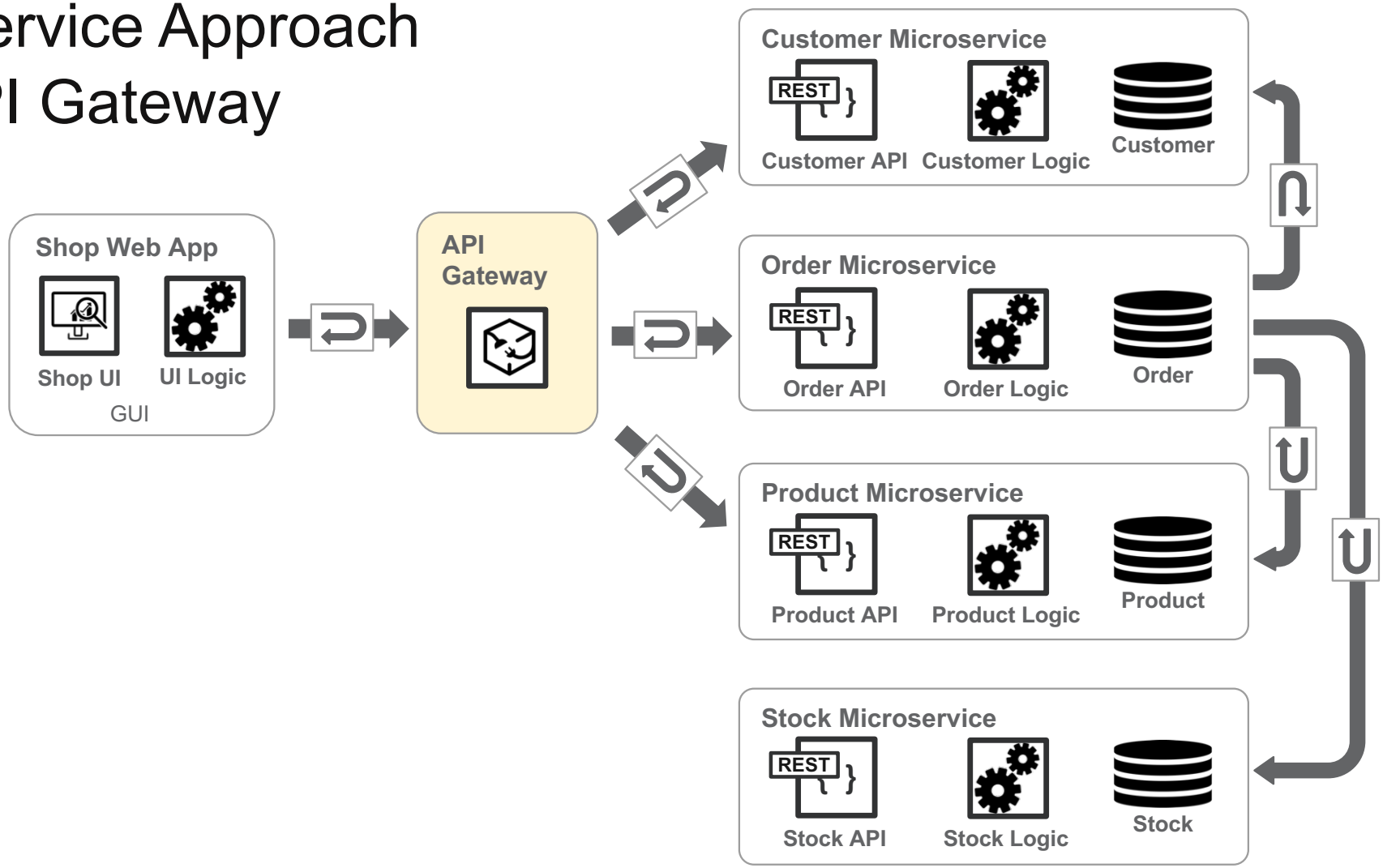
Independently deployable, self-contained and autonomous

SOA done right ?!

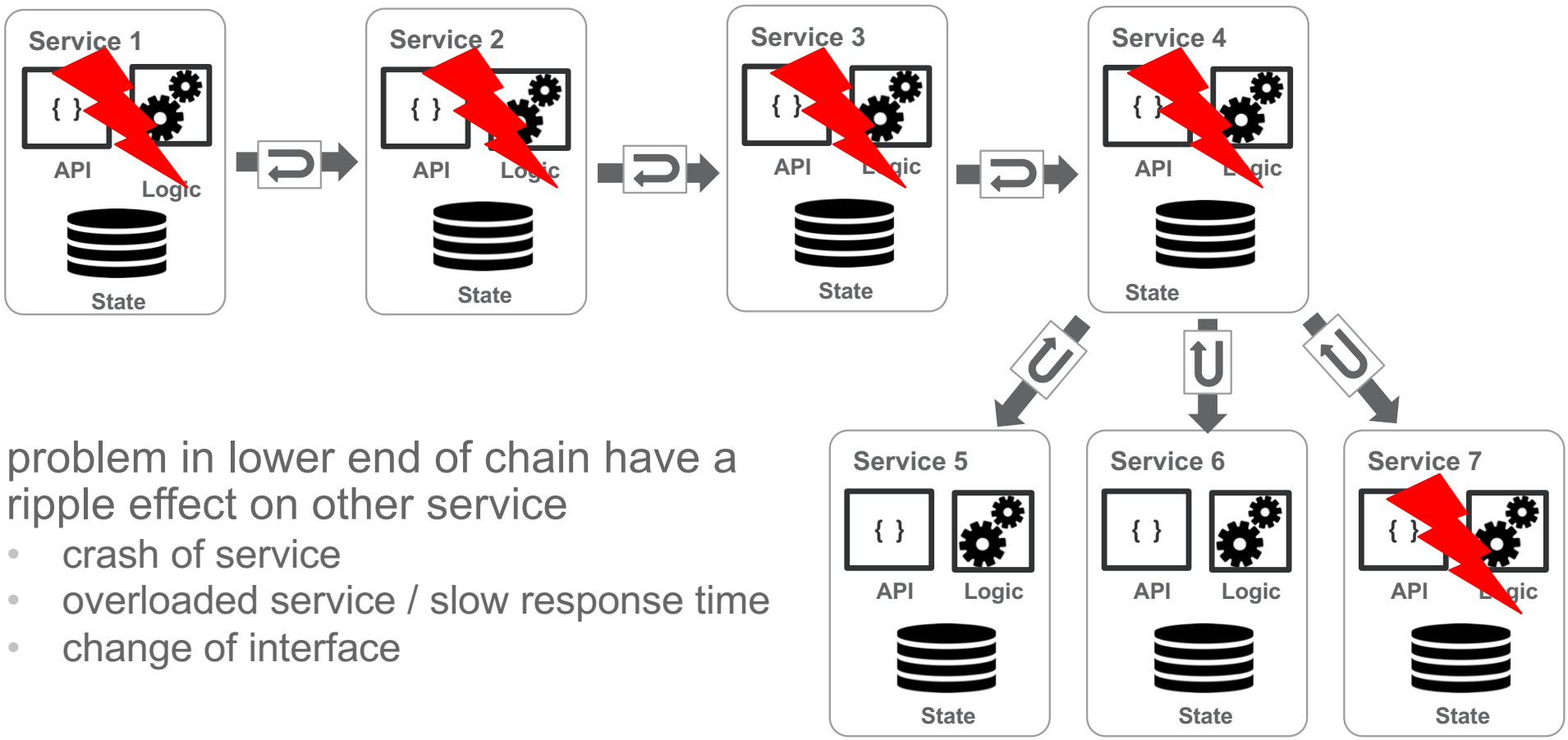
Microservice Approach



Microservice Approach with API Gateway



Synchronous World of Request-Response leads to tight, point-to-point couplings

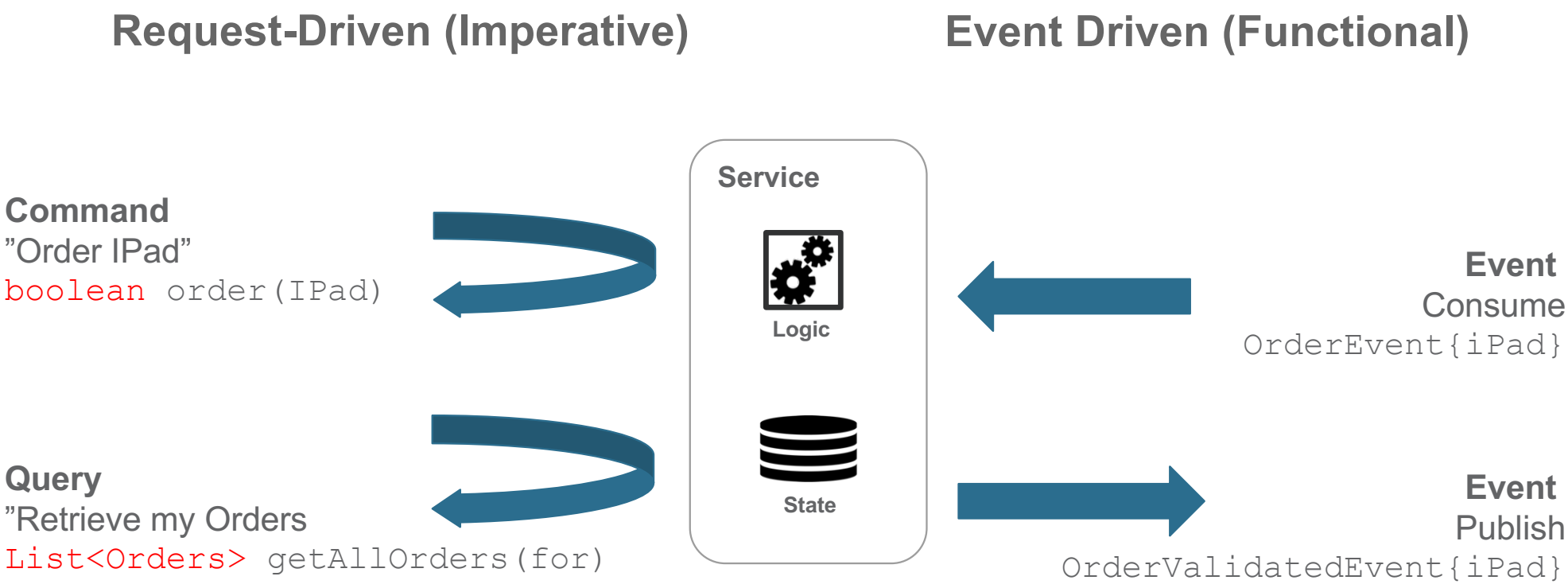


problem in lower end of chain have a ripple effect on other service

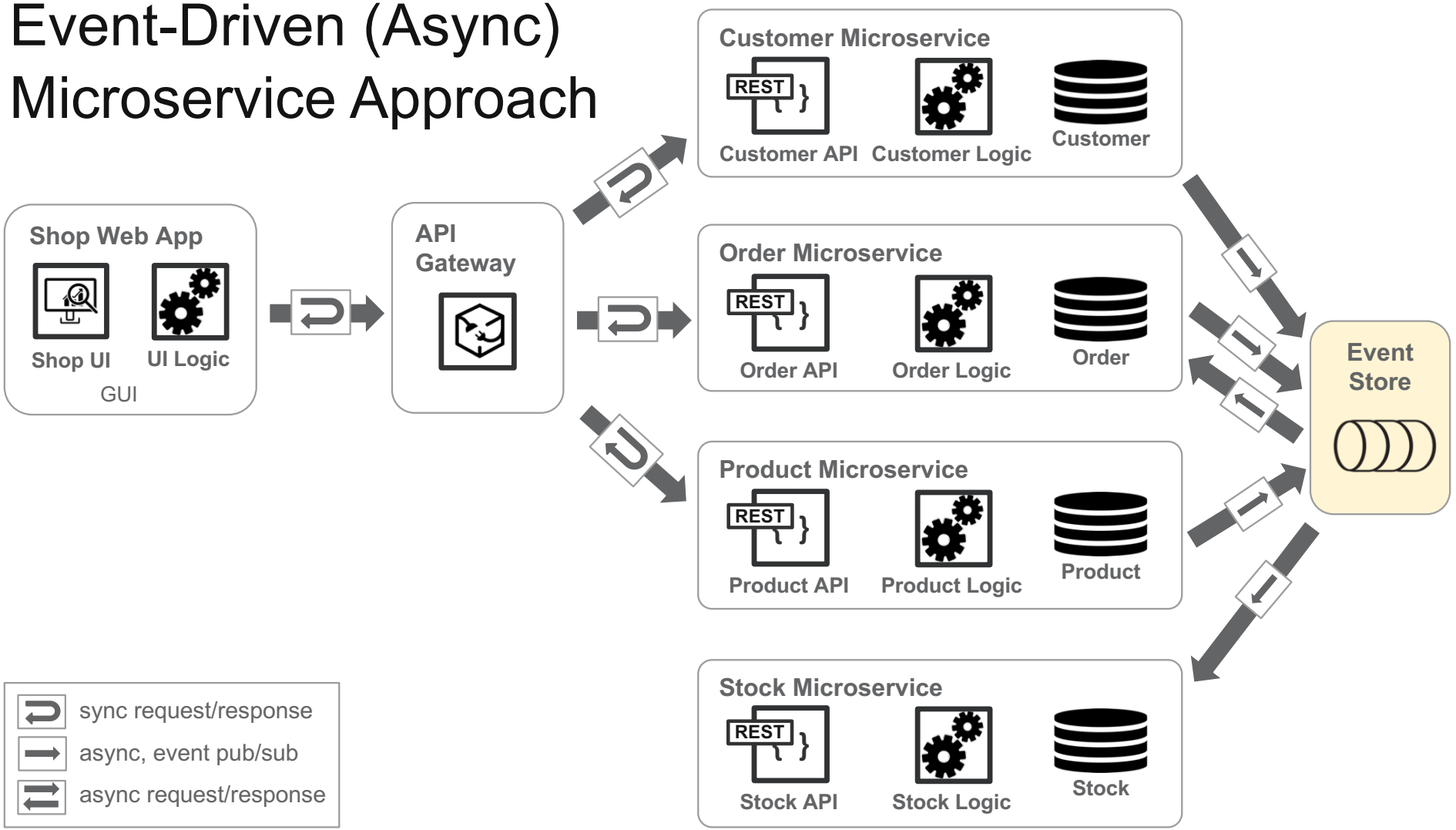
- crash of service
- overloaded service / slow response time
- change of interface

Why not Event-Driven?

3 mechanisms through which services interact

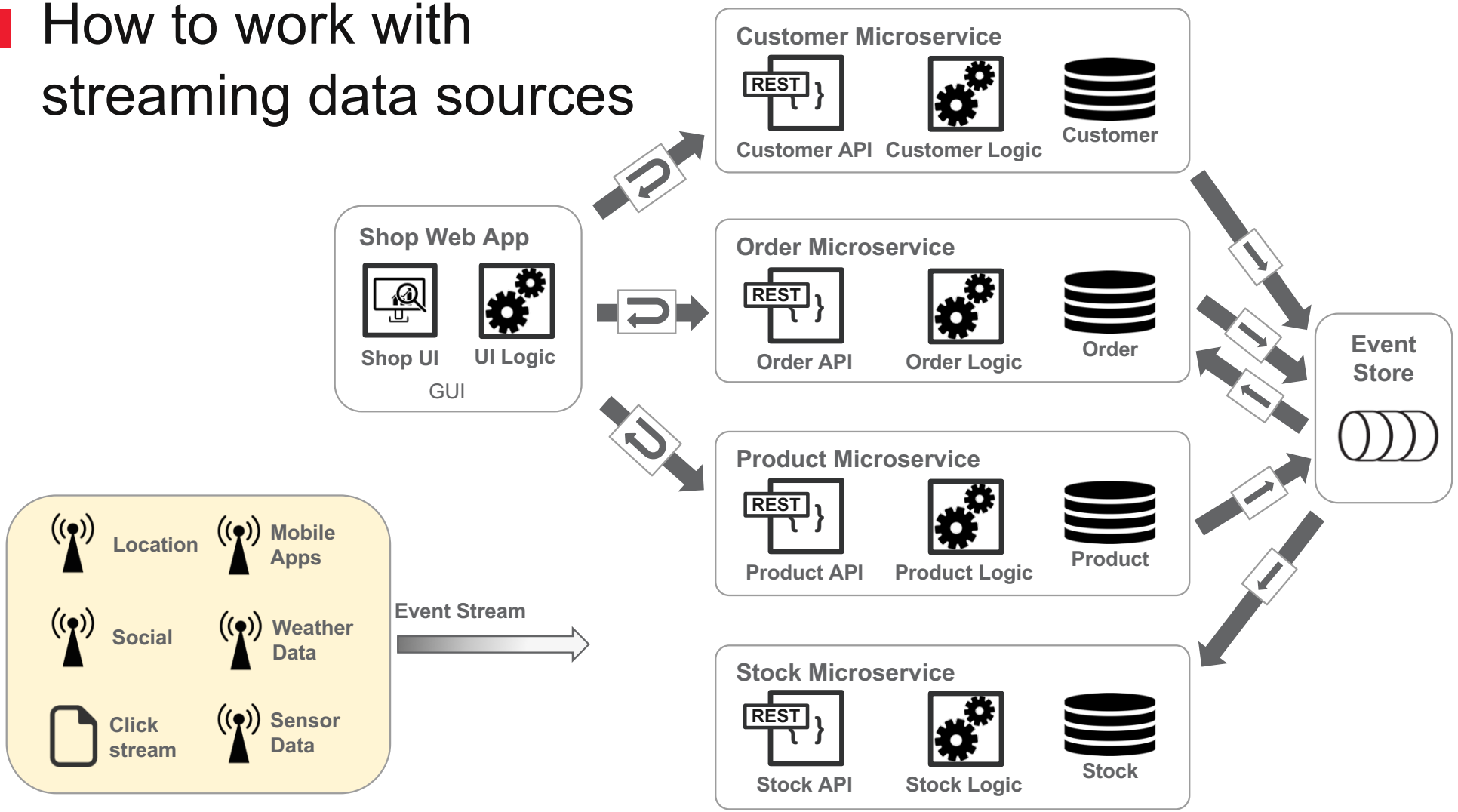


Event-Driven (Async) Microservice Approach

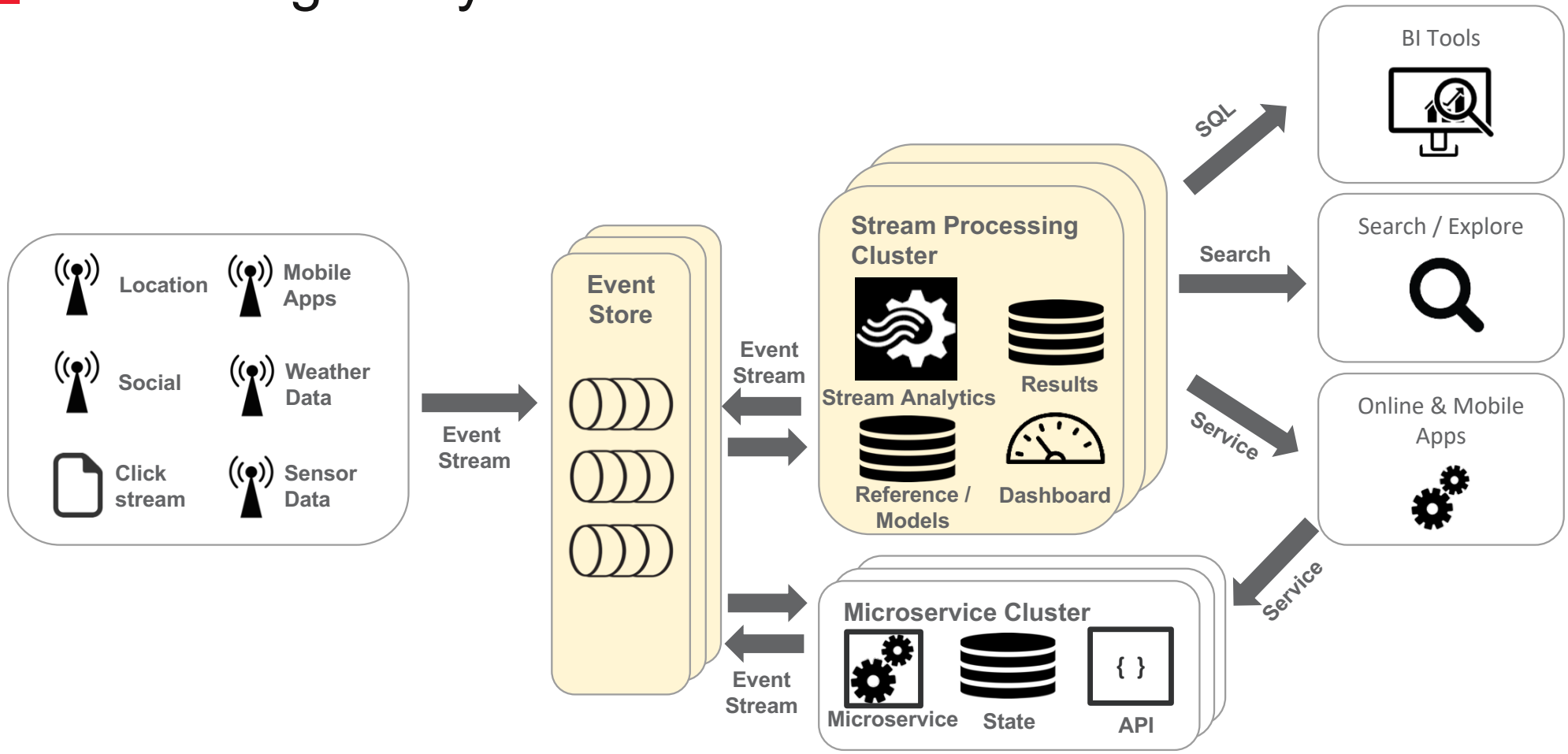


What about streaming sources?

How to work with streaming data sources



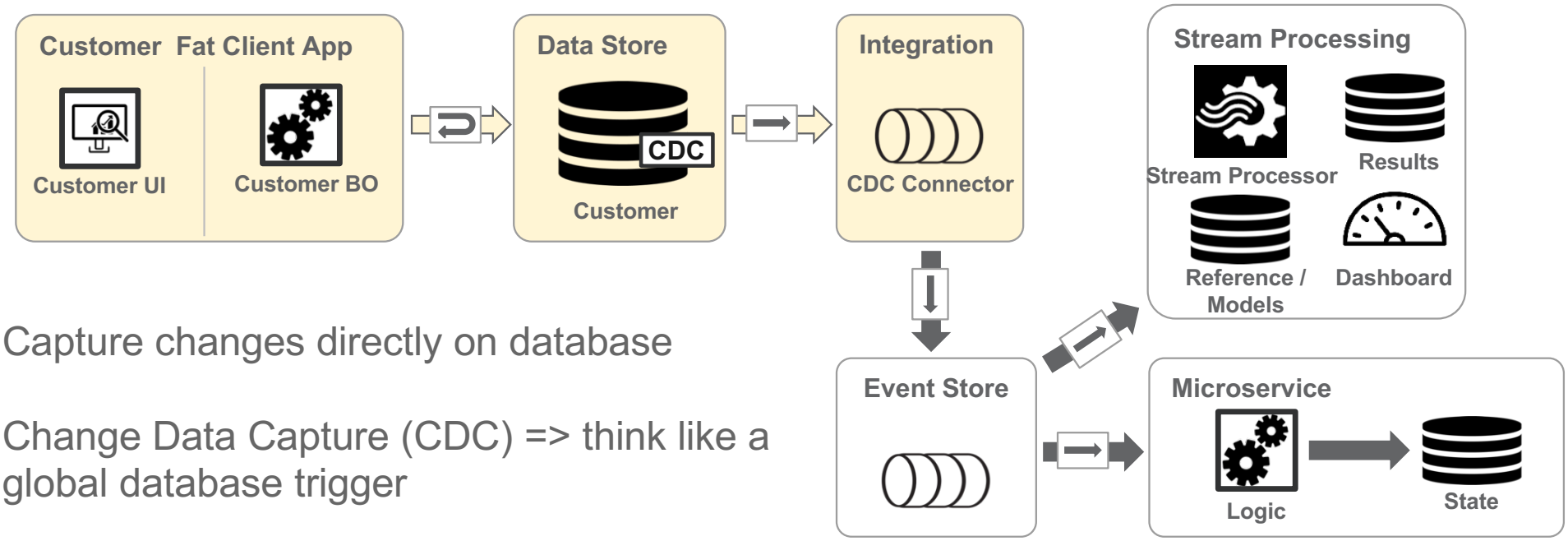
Streaming Analytics Architecture



trivadis
makes IT easier.

What about integrating legacy applications?

Integrate existing systems through CDC

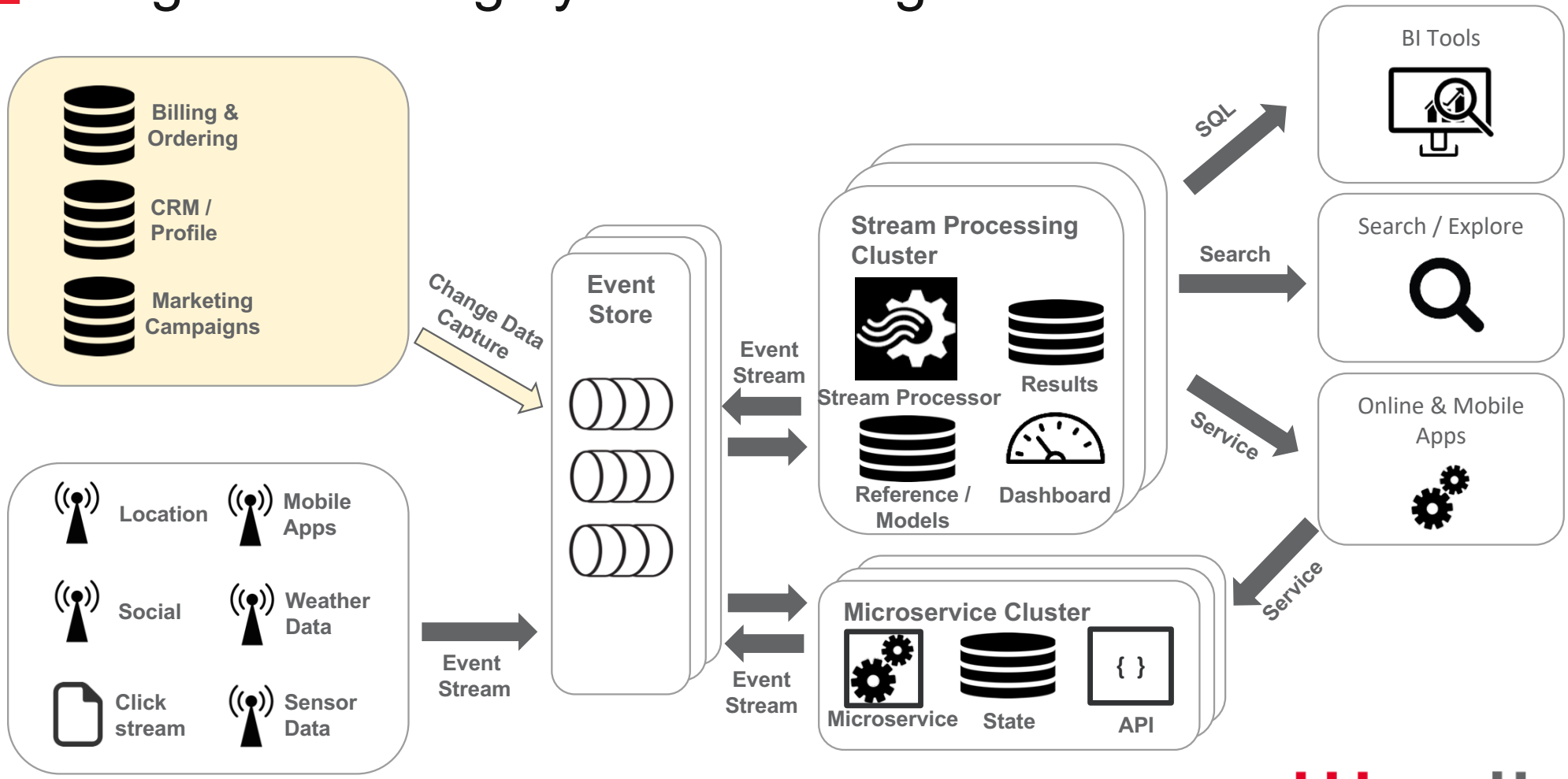


Capture changes directly on database

Change Data Capture (CDC) => think like a global database trigger

Transform existing systems to event producer

Integrate existing systems through CDC



trivadis
makes IT easier.

CQRS and Event Sourcing

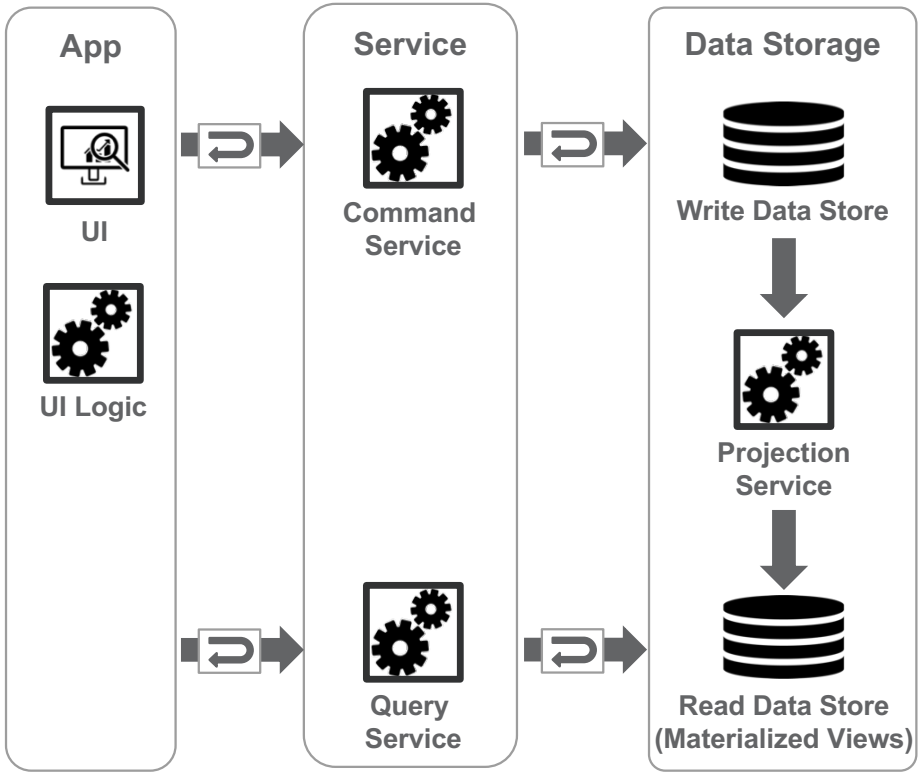
Command Query Responsibility Segregation (CQRS)

Optimize different nonfunctional requirements for **read** and **write** behavior

split between

- **commands** that trigger changes in state
- **queries** that provide read access to the state of resources

support services with **higher performance and capacity requirements** for reading data than for writing data



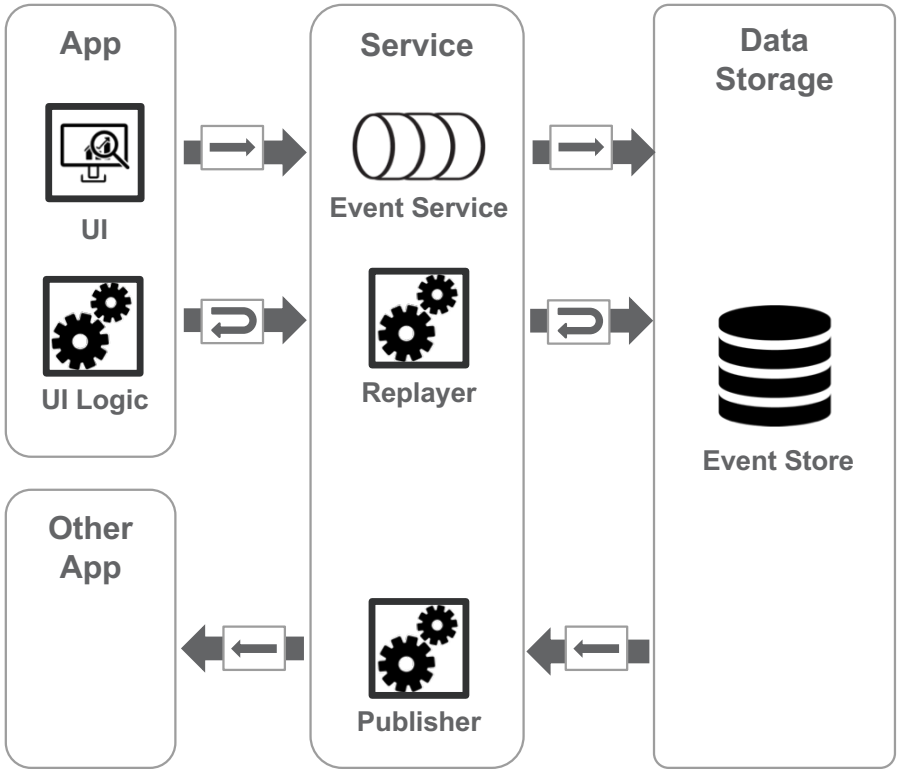
Event Sourcing

persists the state of a business entity as a **sequence of state-changing events**

Whenever state of business entity changes, a new event is appended to the list of events

Saving an event is a single operation and is inherently atomic

The application reconstructs an entity's current state **by replaying the events**

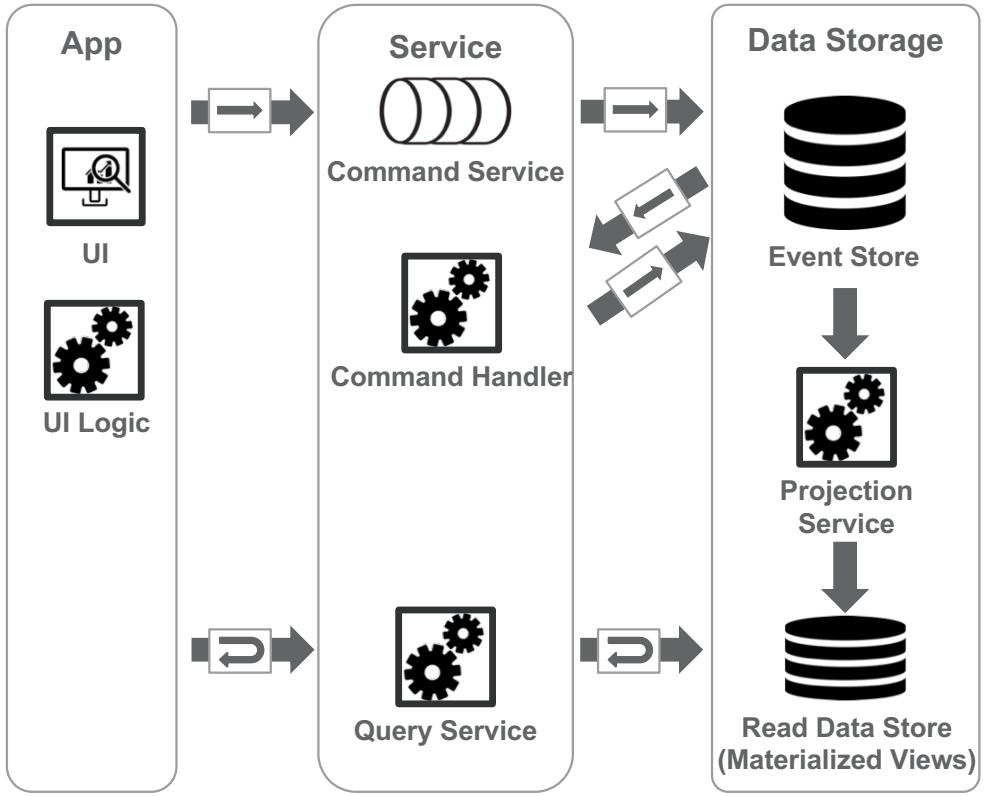


Event Sourcing & CQRS

Event sourcing is commonly combined with the CQRS pattern

materializing views from the stored events

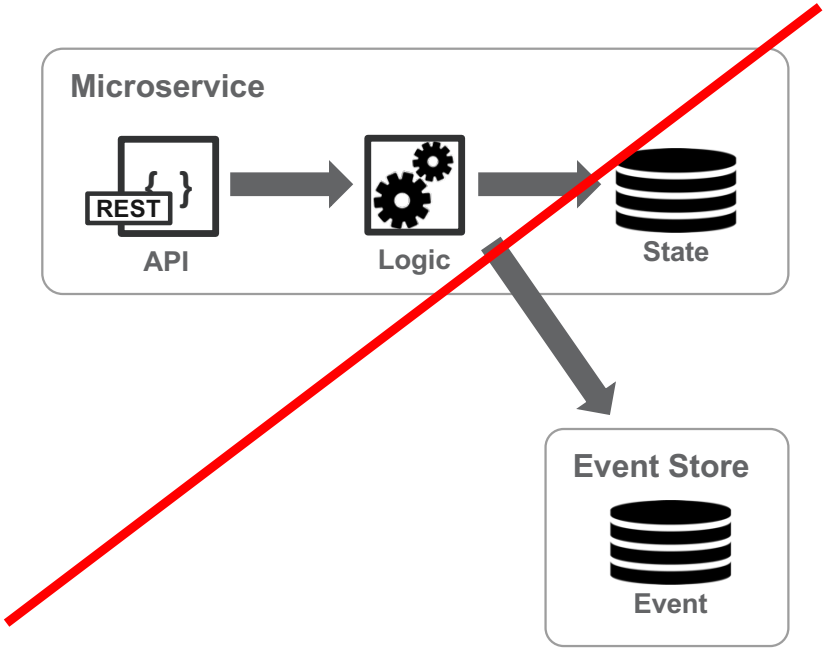
Optionally Commands can be stored in event store and transformed into events by the command handler



Have only one „source of truth“

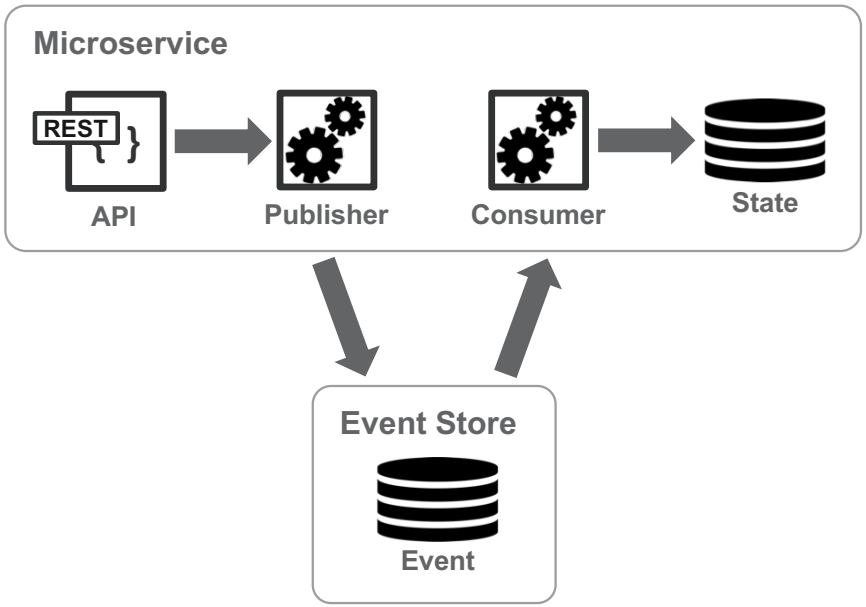
Avoid double write!

- Would need distributed transactions



Write Event first then consume it from same micro service

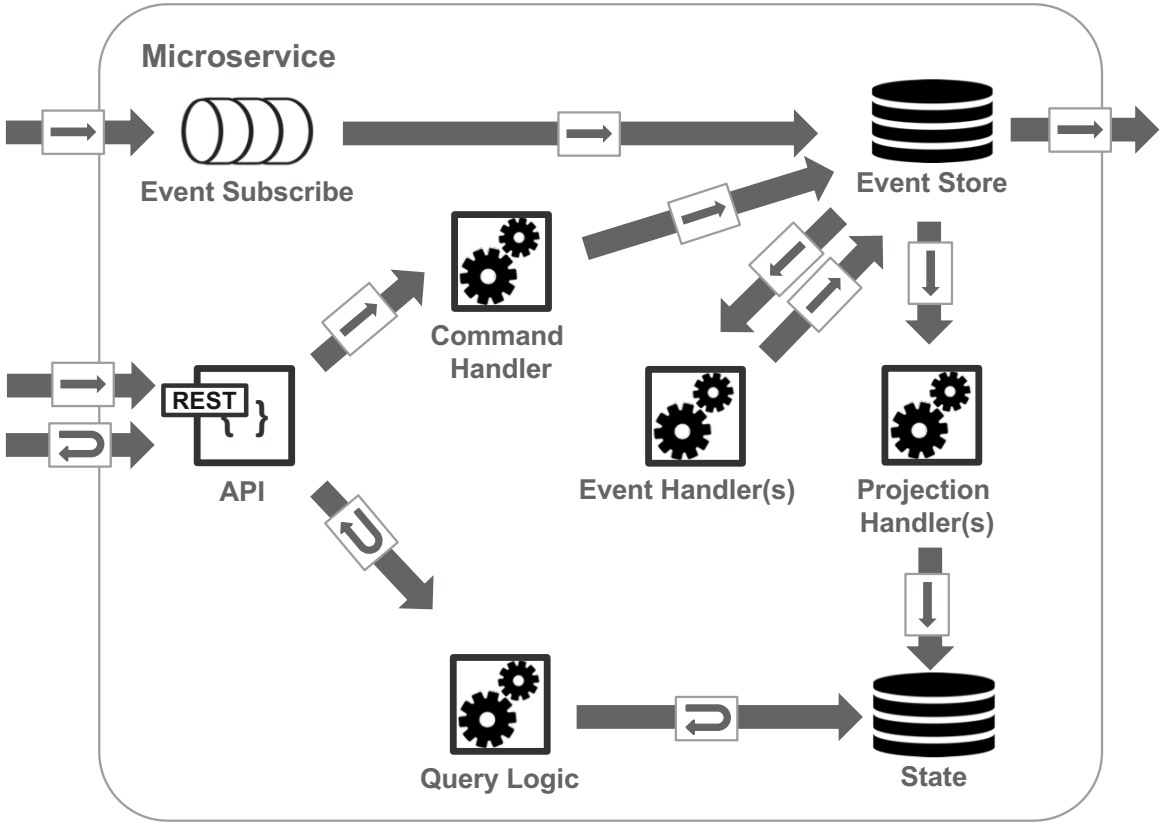
- “eat your own dog food”



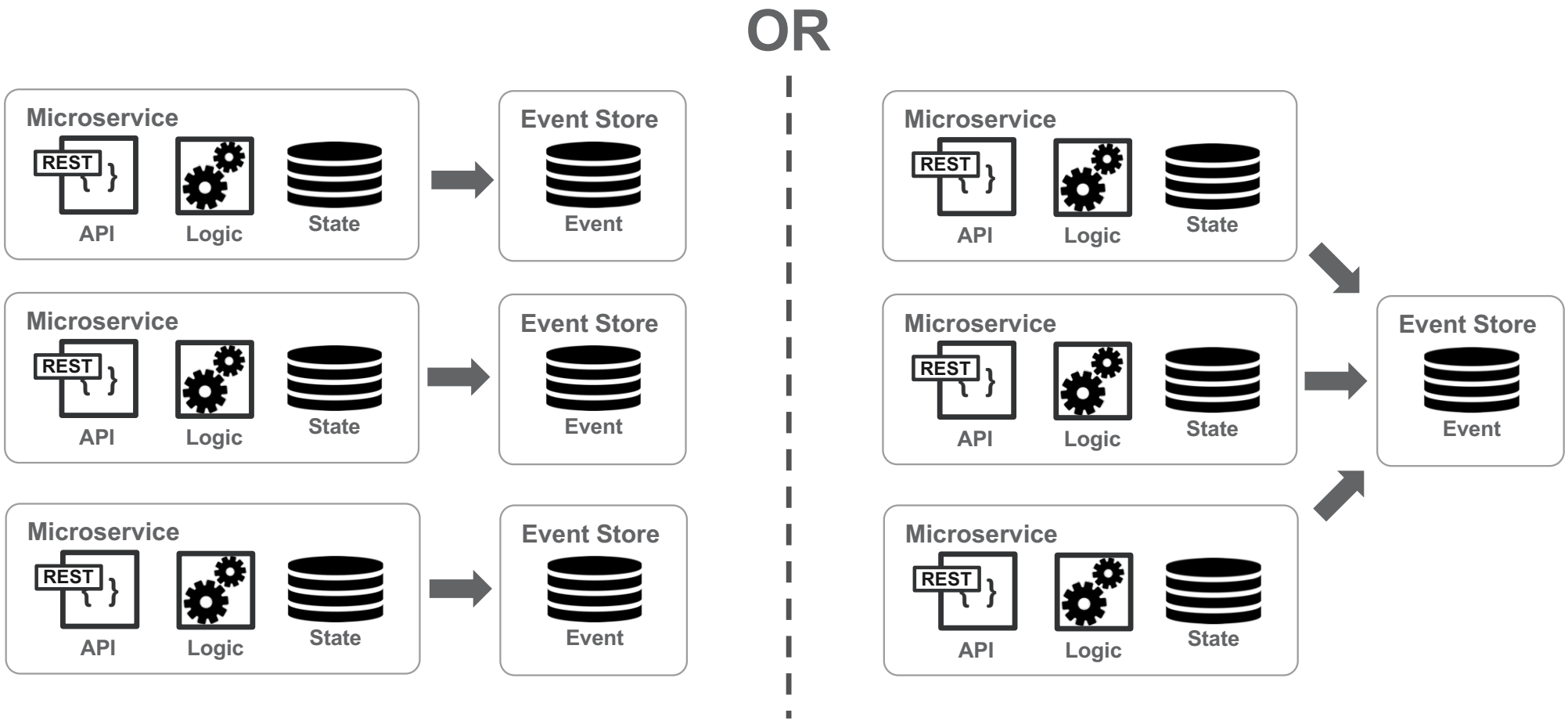
Using Event Sourcing with Microservices

"Event sourcing enables building a forward-compatible application architecture—the ability to add more applications in the future that need to process the same event but create a different materialized view."

Neha Narkhede, Confluent Blog

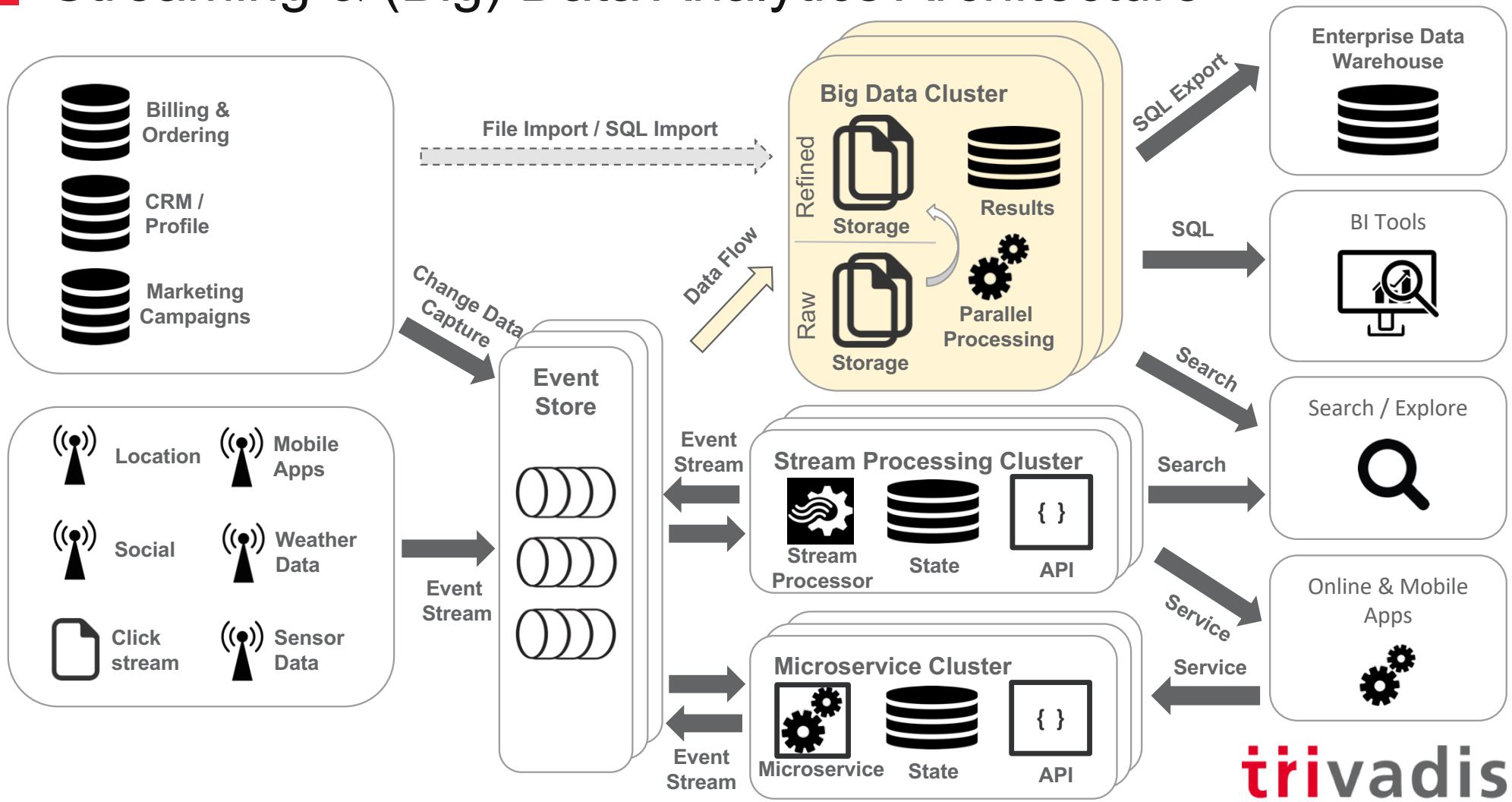


How many Event Stores do we need ?



What about (historical) data analytics?

Streaming & (Big) Data Analytics Architecture

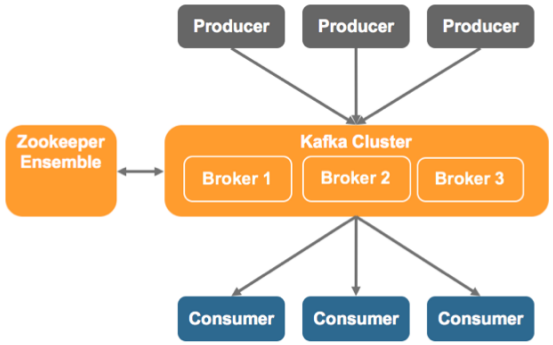


trivadis makes IT easier.

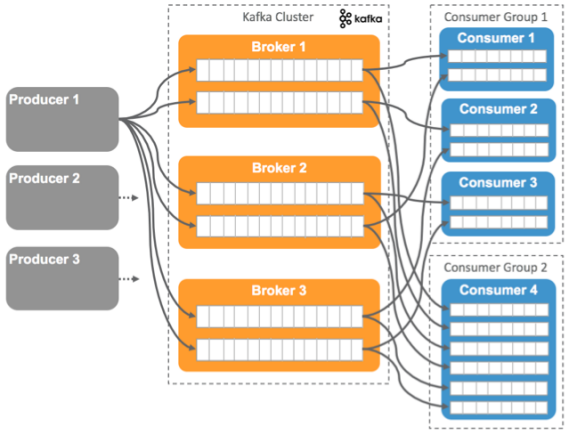
Why Kafka for Event-Driven Microservices?

Apache Kafka – A Streaming Platform

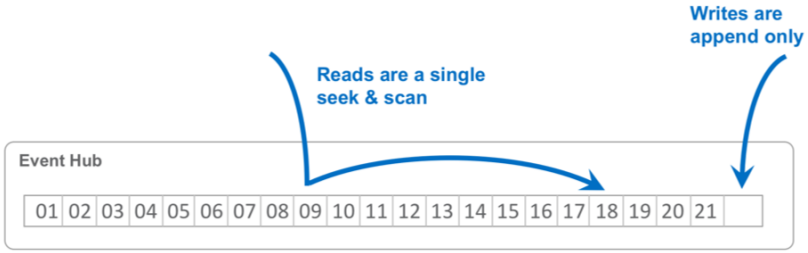
High-Level Architecture



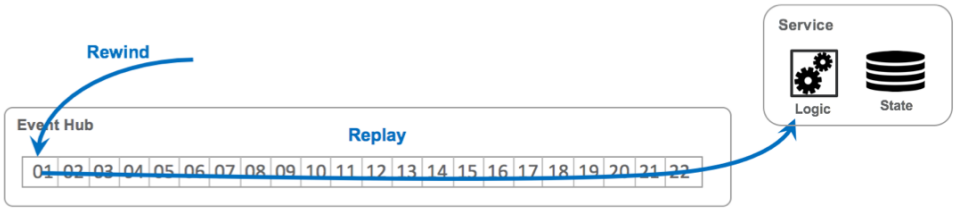
Scale-Out Architecture



Distributed Log at the Core

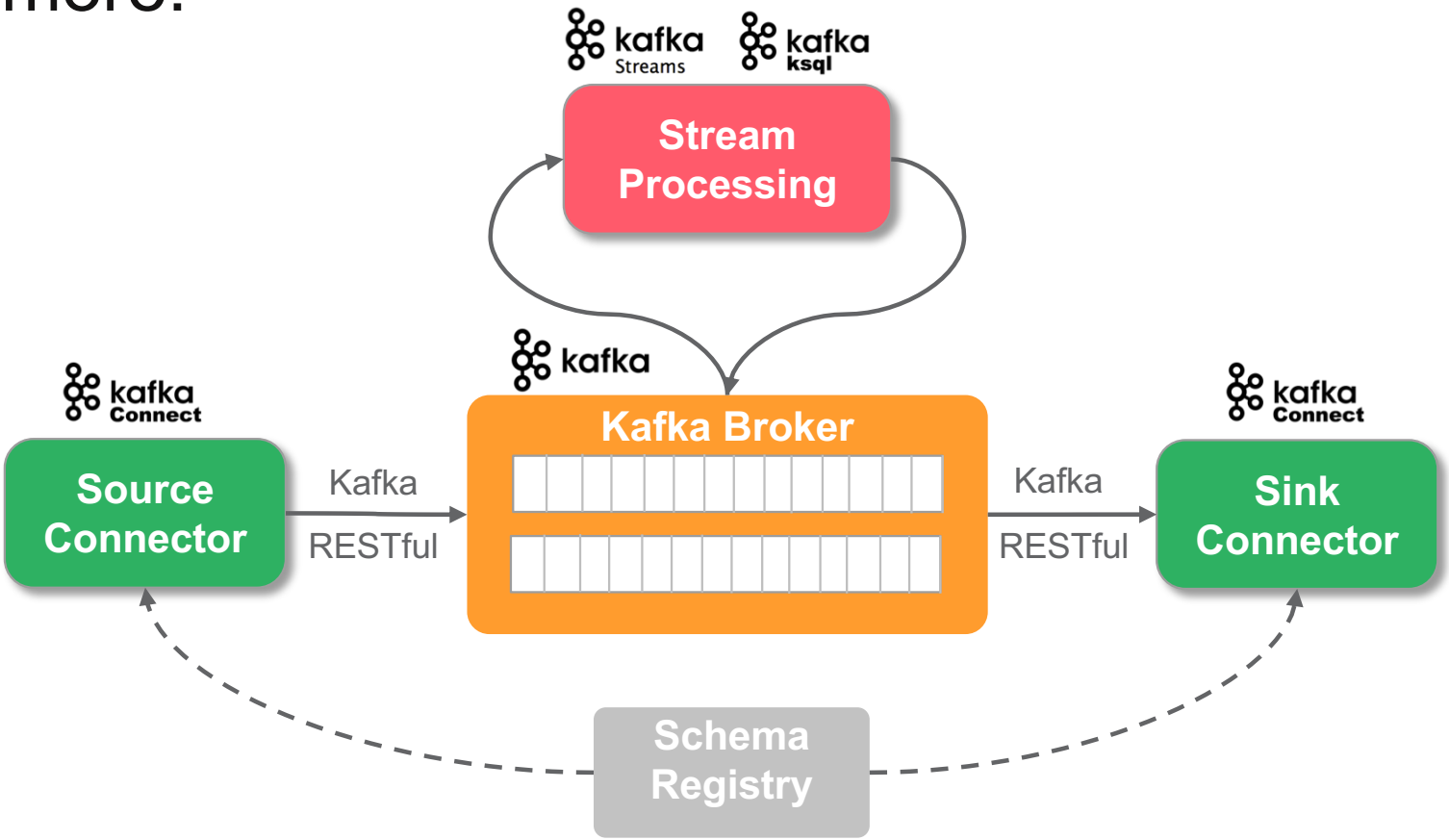


Logs do not (necessarily) forget



trivadis makes IT easier.

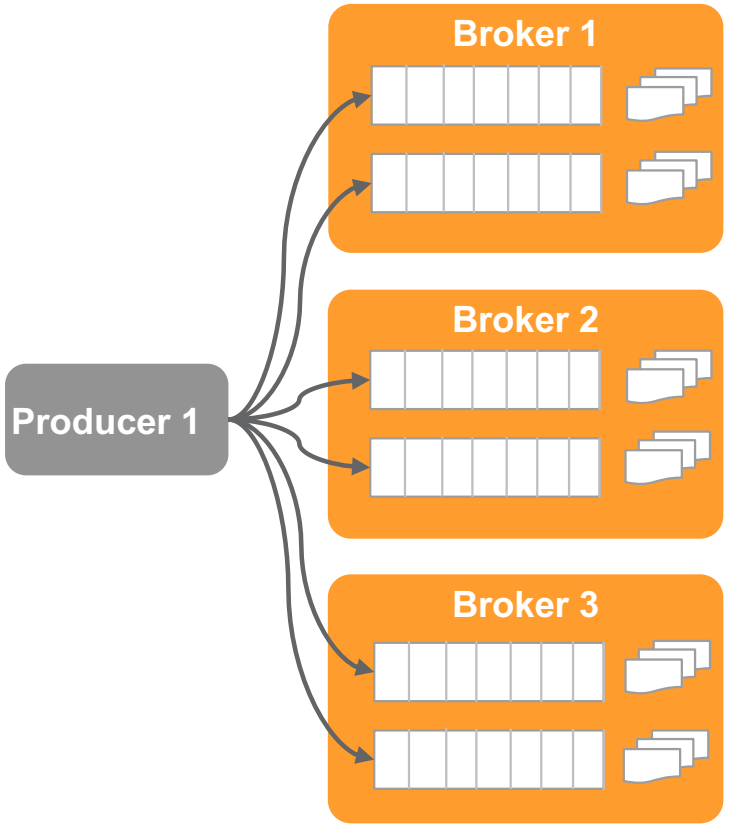
Apache Kafka – scalable message processing and more!



Hold Data for Long-Term – Data Retention

1. Never
2. Time based (TTL)
   `log.retention.{ms | minutes | hours}`
3. Size based
   `log.retention.bytes`
4. Log compaction based
   (entries with same key are removed):

```
kafka-topics.sh --zookeeper zk:2181 \
                --create --topic customers \
                --replication-factor 1 \
                --partitions 1 \
                --config cleanup.policy=compact
```



Topic Viewed as Event Stream or State Stream (Change Log)

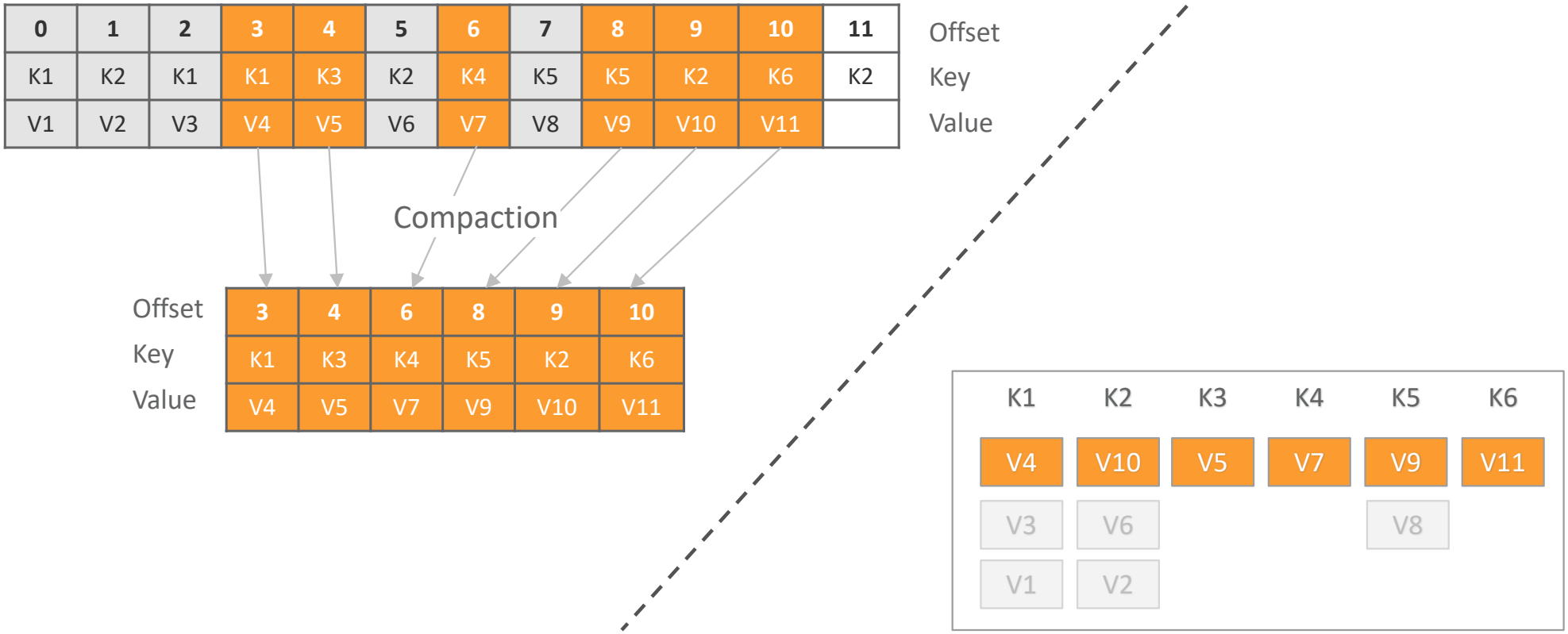
Event Stream

2015-10-02	11,Peter,Muster,3010,Berne
2016-10-04	12,Paul,Steffen,8001,Zurich
2016-12-02	21,Lisa,Meier,3043,Ittigen
2017-05-03	11,Peter,Muster,3015,Berne
2017-05-03	21,Lisa,Steffen,8001,Zurich
2017-07-03	11,Peter,Muster,3052,Zollikofen

State Stream (Change Log Stream)

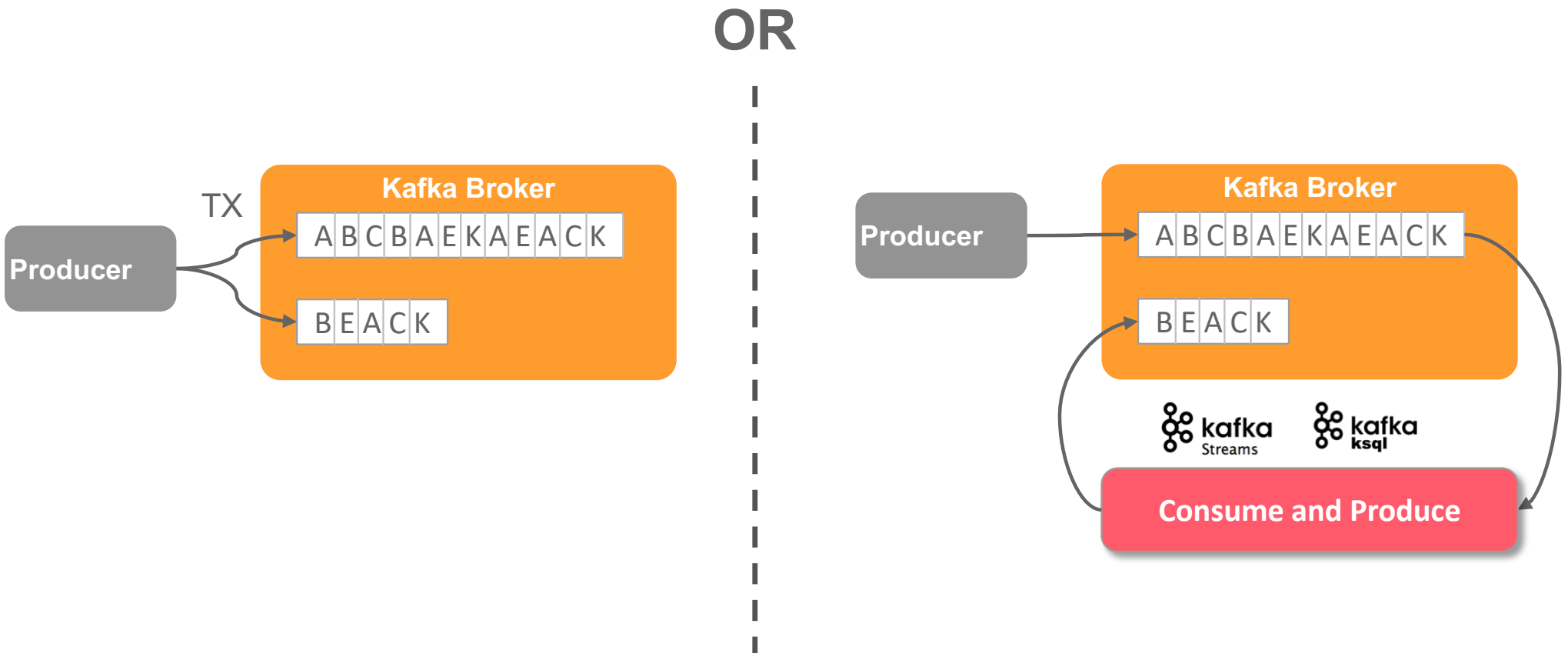
~~2015-10-02~~	~~11,Peter,Muster,3010,Berne~~
2016-10-04	12,Paul,Steffen,8001,Zurich
~~2016-12-02~~	~~21,Lisa,Meier,3043,Ittigen~~
~~2017-05-03~~	~~11,Peter,Muster,3015,Berne~~
2017-05-03	21,Lisa,Steffen,8001,Zurich
2017-07-03	11,Peter,Muster,3052,Zollikofen

Keep Topics in Compacted Form

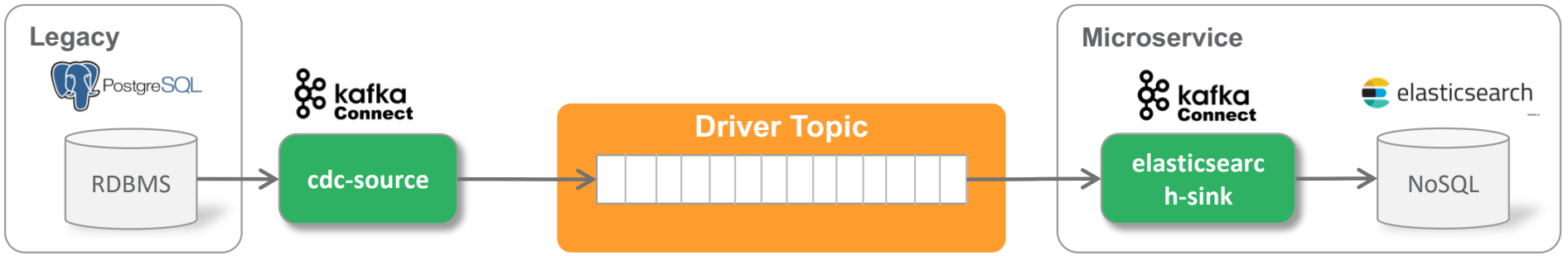


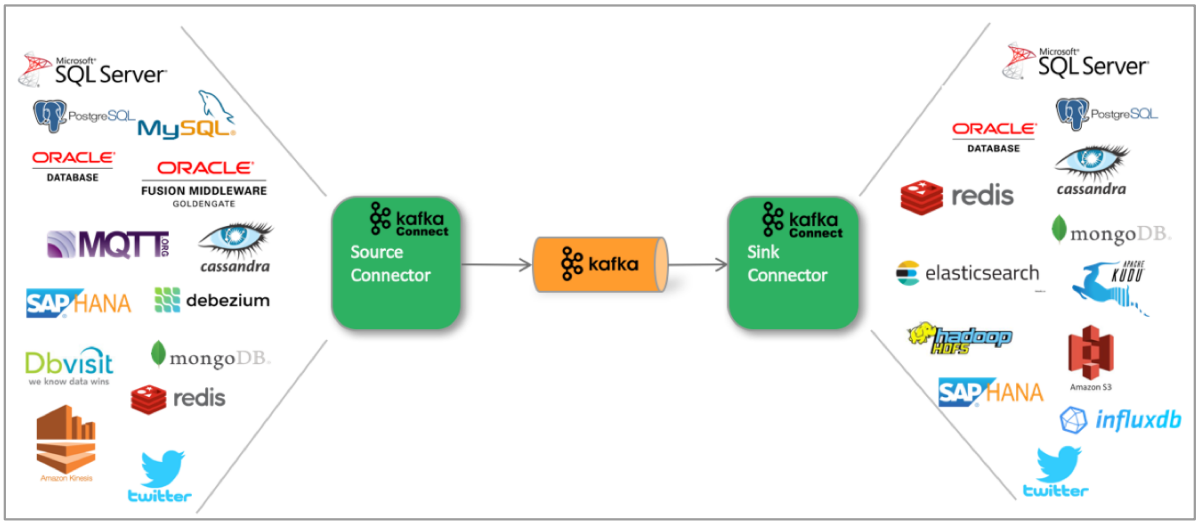
trivadis makes IT easier.

Keep Topics both in Original and Compacted Form



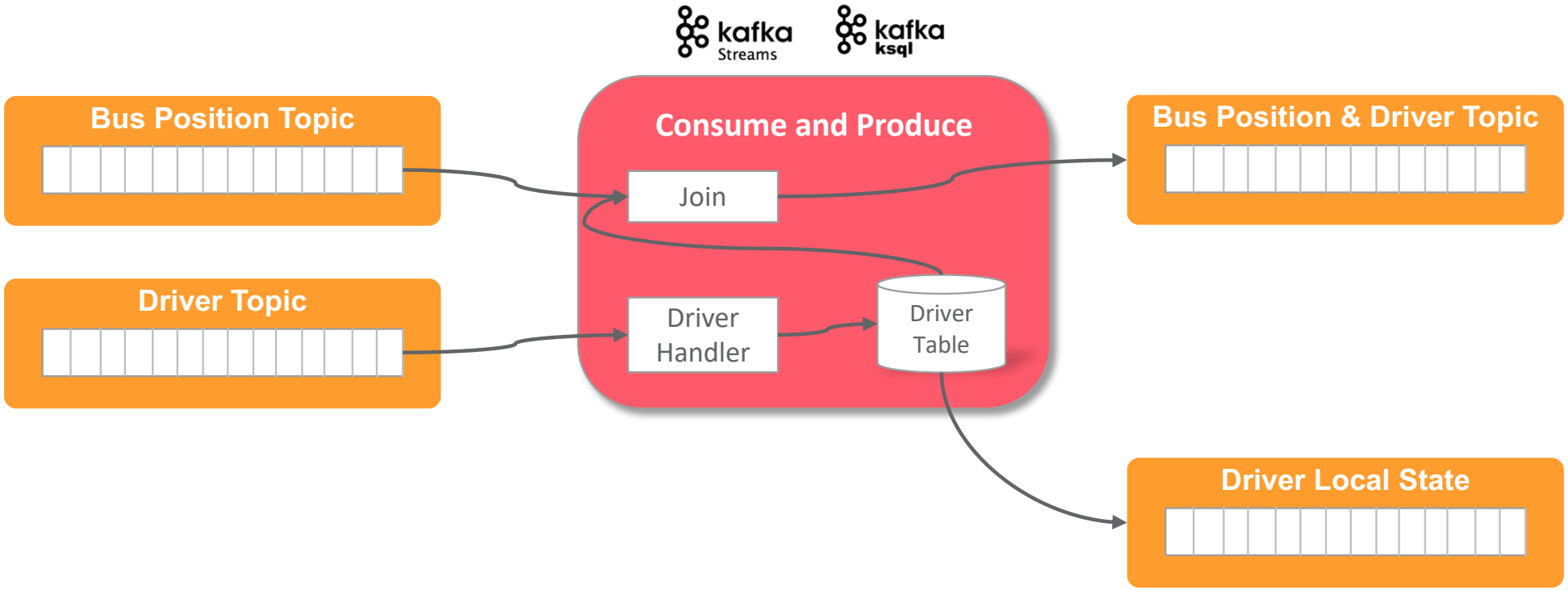
Change Data Capture (CDC)



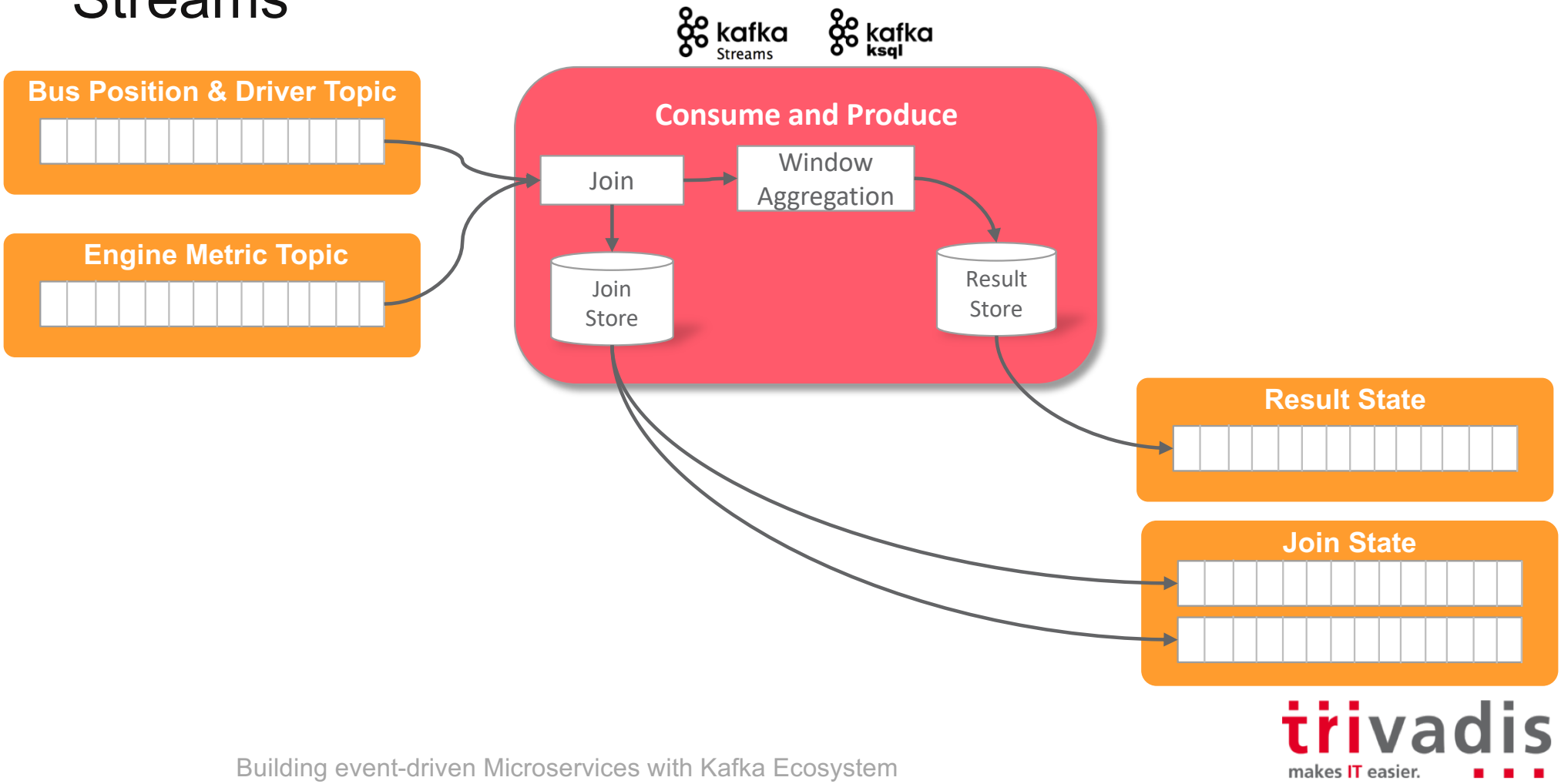


trivadis makes IT easier.

Enrich Stream with Static Data with Kafka Streams

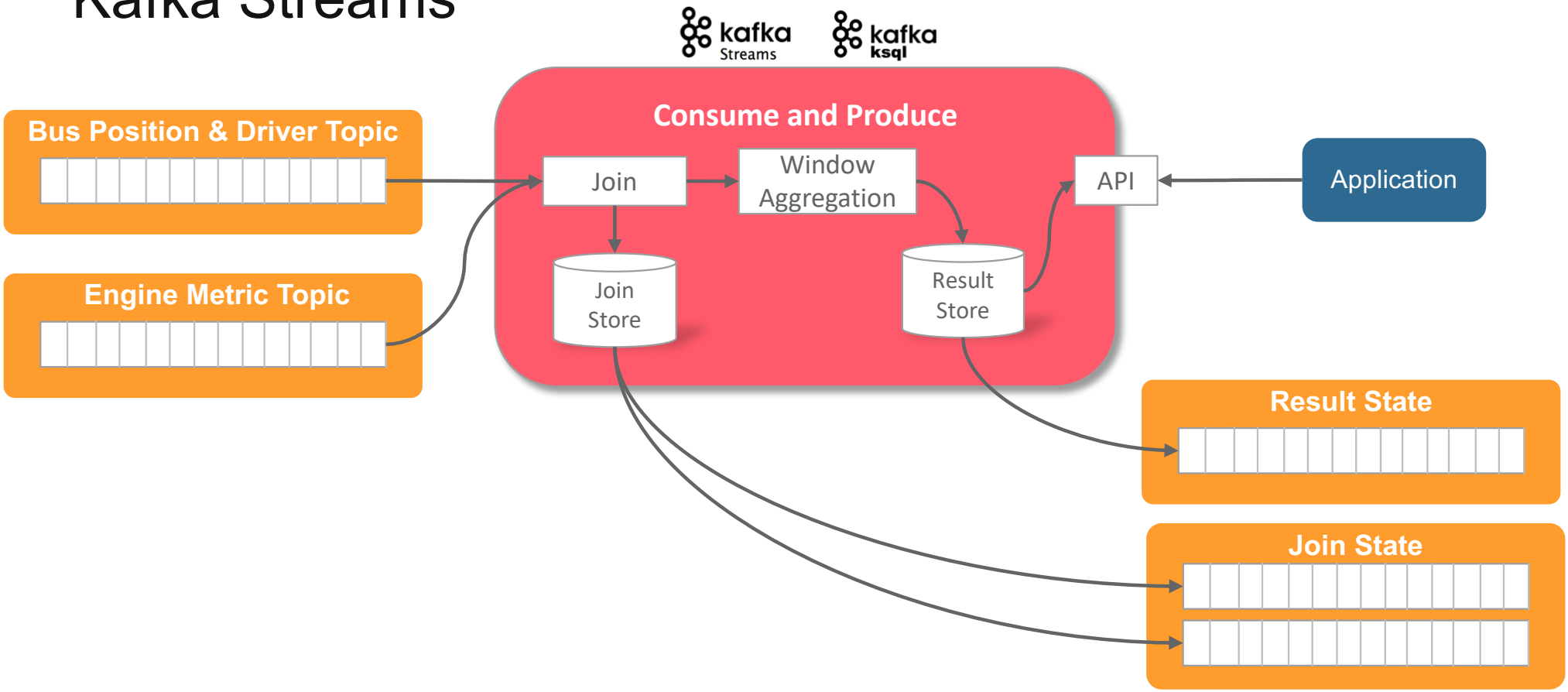


Building Embedded, Materialized View with Kafka Streams



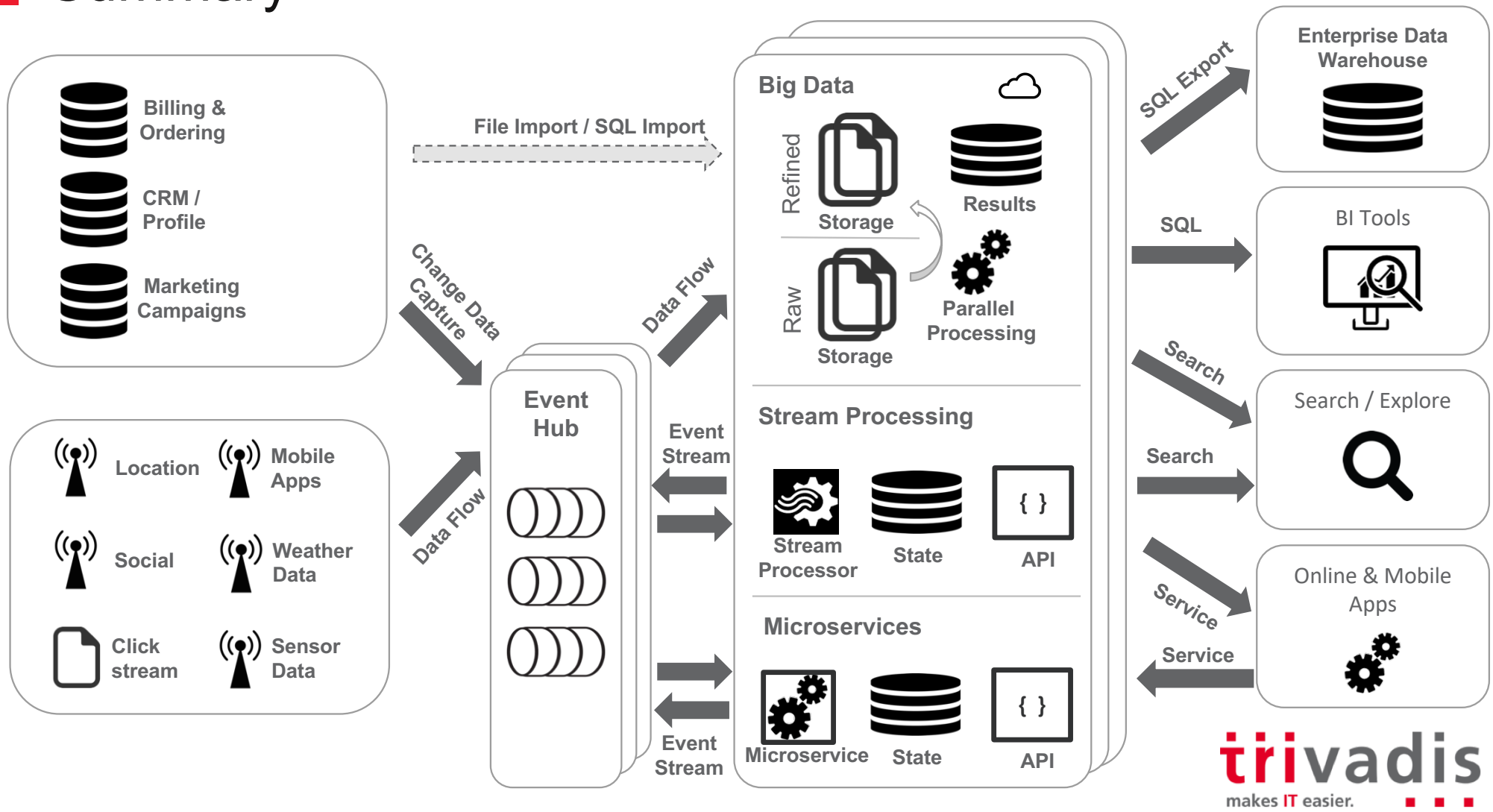
trivadis makes IT easier.

Building Embedded, Queryable Materialized View with Kafka Streams



Summary

Summary



trivadis
makes IT easier.

Summary

- service autonomy is key in a Microservices Architecture!
- not all communication need to be synchronous => separate into
  - commands
  - events
  - queries
- Kafka is well suited as an event broker / event store
  - brings many more interesting features beyond just “message passing”

References

Microservices Blog Series, Ben Stopford, Confluent:

- https://www.confluent.io/blog/tag/microservices

Apache Kafka for Microservices: A Confluent Online Talk Series:

- https://www.confluent.io/landing-page/microservices-online-talk-series/

Turning the database inside-out with Apache Samza, Martin Kleppmann, Con

- https://www.confluent.io/blog/turning-the-database-inside-out-with-apache-samza/

Event sourcing, CQRS, stream processing and Apache Kafka: What's the connection?, Neha Narkhede, Confluent:

- https://www.confluent.io/blog/event-sourcing-cqrs-stream-processing-apache-kafka-whats-connection/

Immutability Changes Everything, Pat Helland, Salesforce:

- http://cidrdb.org/cidr2015/Papers/CIDR15_Paper16.pdf

Commander: Better Distributed Applications through CQRS and Event Sourcing, Bobby Calderwood:

- https://www.youtube.com/watch?v=B1-gS0oEtYc