Change Data Streaming Patterns for Microservices With Debezium

Modern Integration and Application Development Day, Milano, April 3 2019

Gunnar Morling
@gunnarmorling

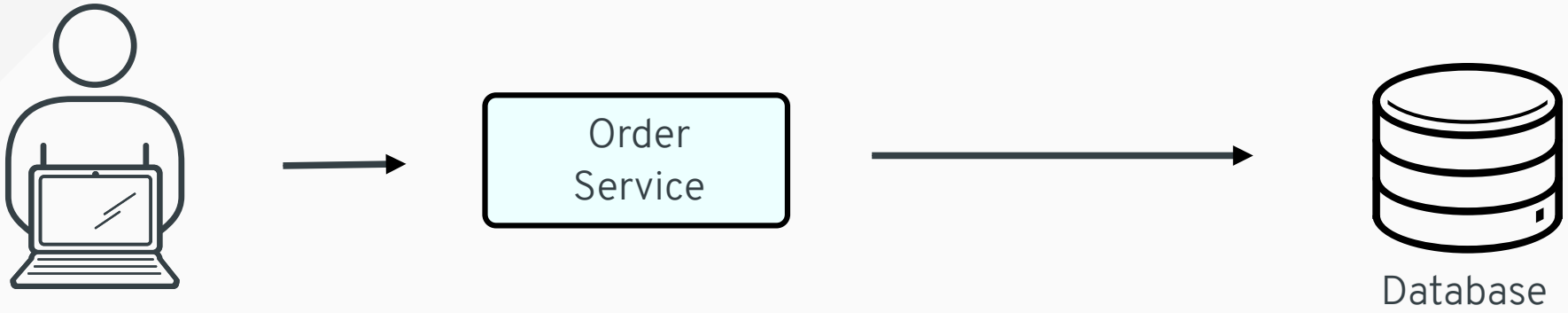
Gunnar Morling

- Open source software engineer at Red Hat
  - **Debezium**
  - Hibernate
- **Spec Lead** for Bean Validation 2.0
- Other projects: **Deptective**, MapStruct

gunnar@hibernate.org
@gunnarmorling
http://in.relation.to/gunnar-morling/

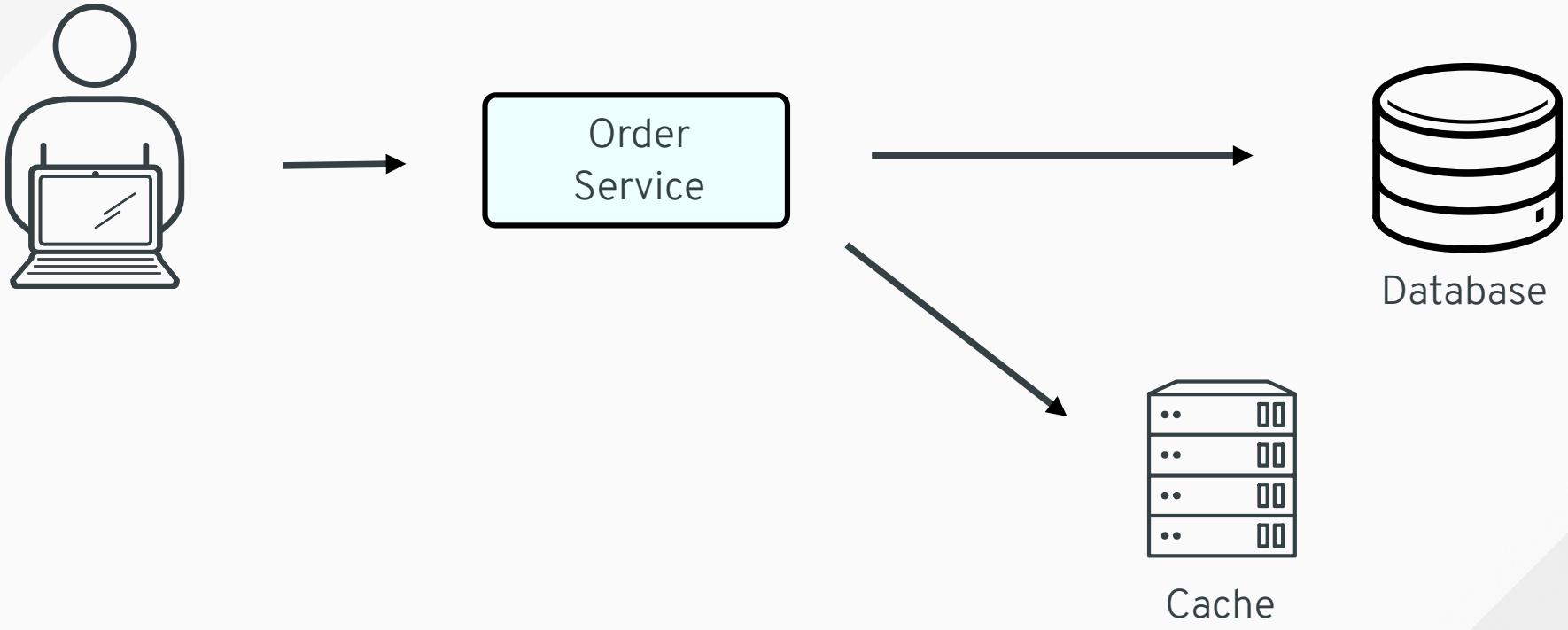
A Common Problem

Updating Multiple Resources



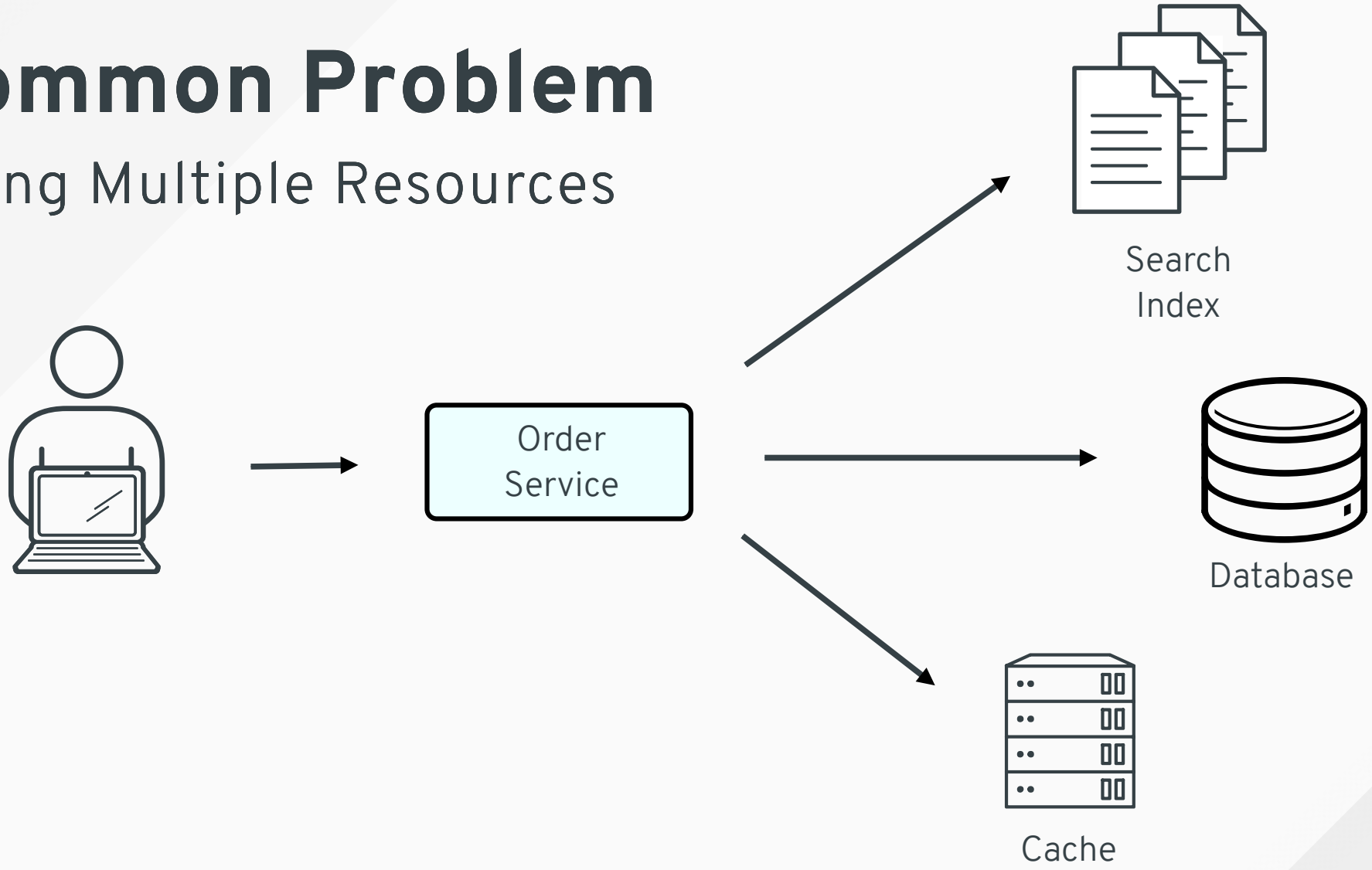
A Common Problem

Updating Multiple Resources



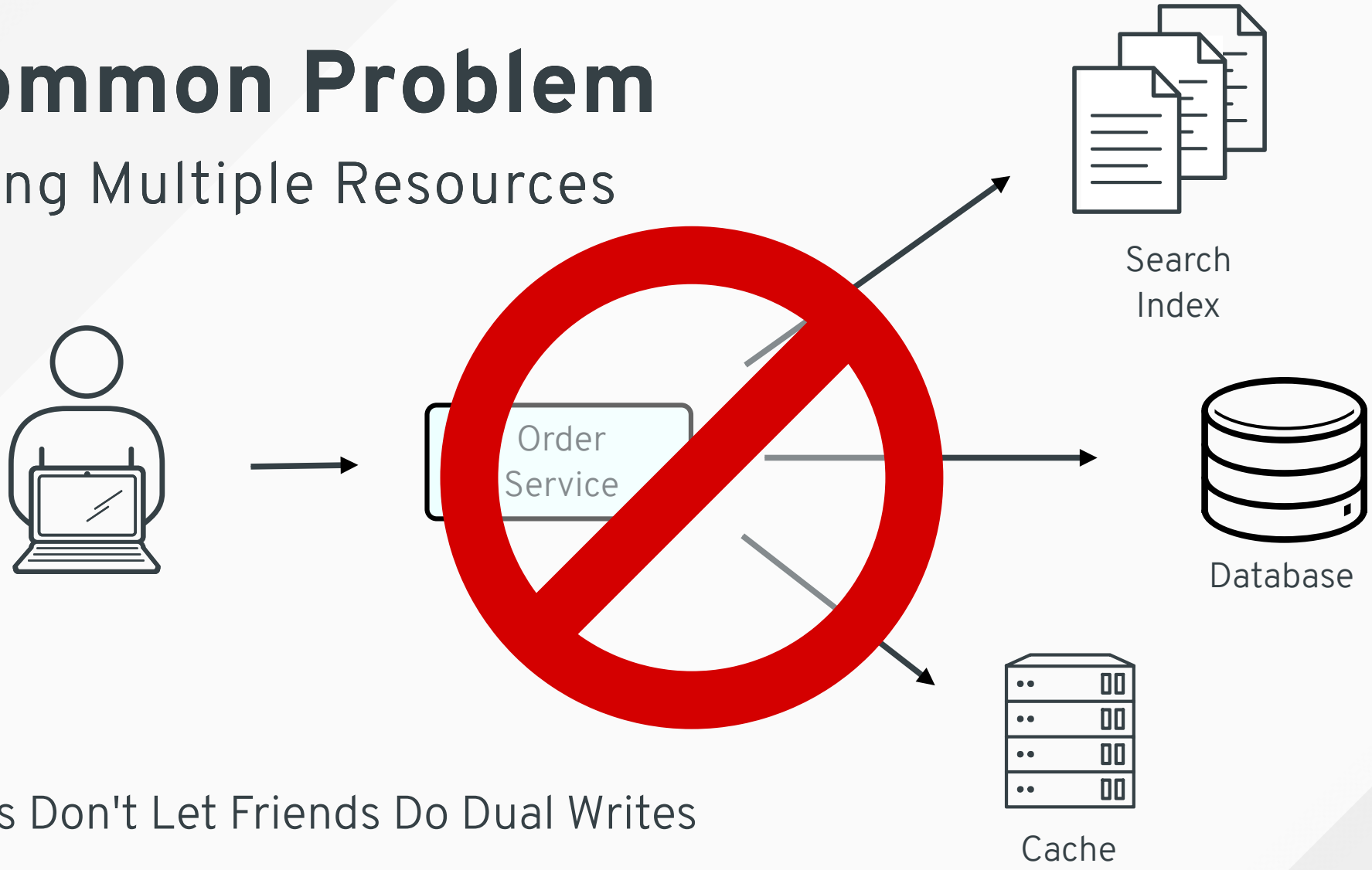
A Common Problem

Updating Multiple Resources



A Common Problem

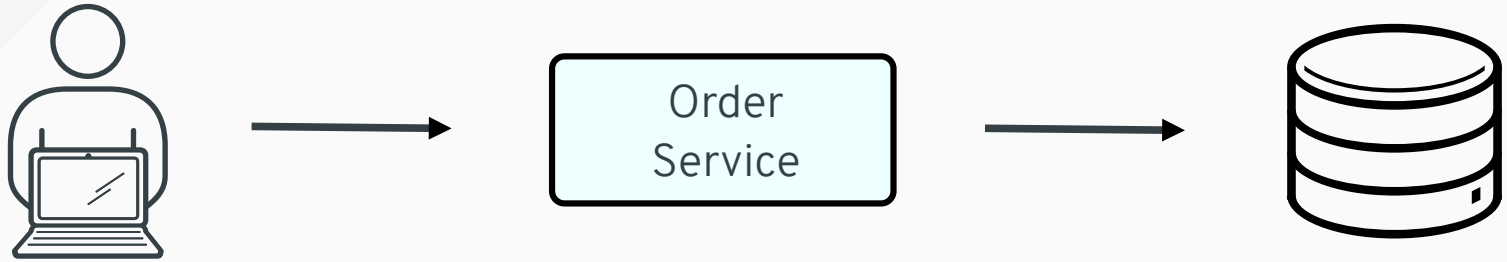
Updating Multiple Resources



“ Friends Don't Let Friends Do Dual Writes

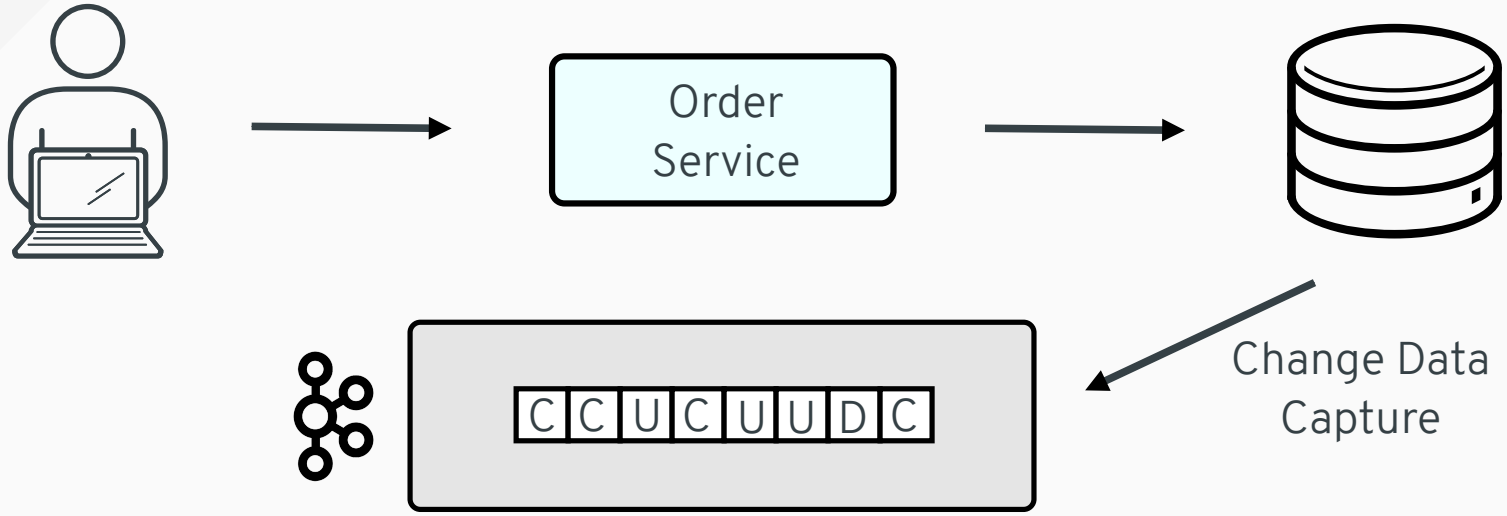
A Better Solution

Streaming Change Events From the Database



A Better Solution

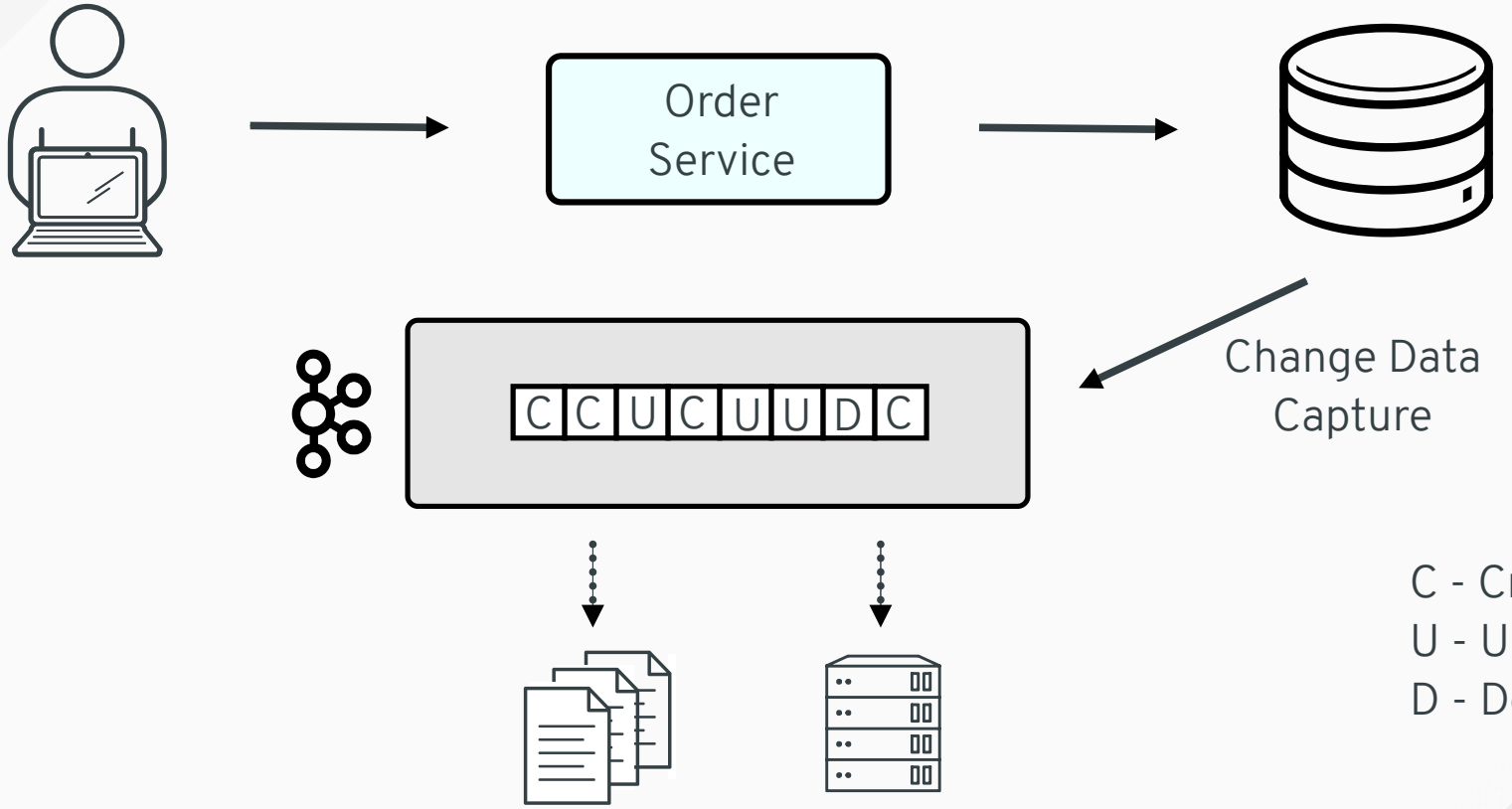
Streaming Change Events From the Database



C - Create
U - Update
D - Delete

A Better Solution

Streaming Change Events From the Database

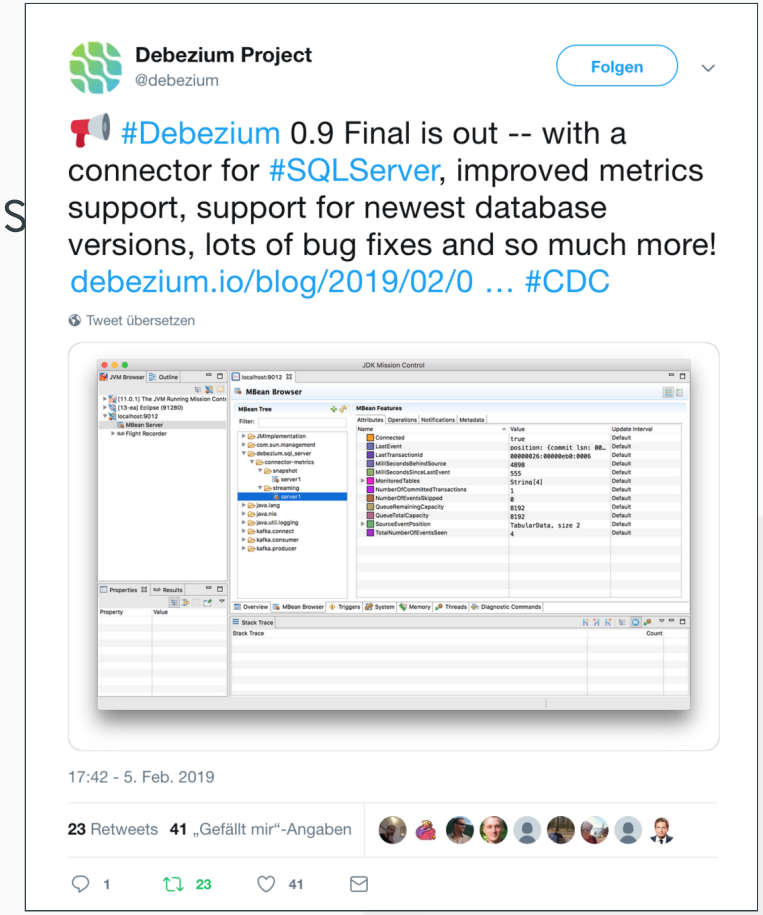



Change Data Capture
With Debezium

Debezium

Change Data Capture Platform

- Retrieves change events from **TX logs** from different DBs
  - Transparent to writing apps
  - Comprehensive **type support** (PostGIS etc.)
  - **Snapshotting**, Filtering etc.
- Fully open-source, very **active community**
- Latest version: 0.9 (based on **Kafka 2.1**)
- Production deployments at multiple companies (e.g. WePay, Trivago, BlaBlaCar etc.)



Advantages of Log-based CDC

Tailing the transaction log

- **All data changes** are captured
- **No polling delay** or overhead
- **Transparent** to writing applications and models
- Can **capture deletes**
- Can capture **old record state** and further meta data
- Different formats/APIs, but Debezium deals with this

Debezium

CDC Use Cases

- Update or invalidate **caches**
- Enable **full-text search** via Elasticsearch, Solr etc.
- Data **replication**
- **Microservices** data exchange
- **Auditing**/historization
- Update **CQRS** read models
- Enable **streaming queries**

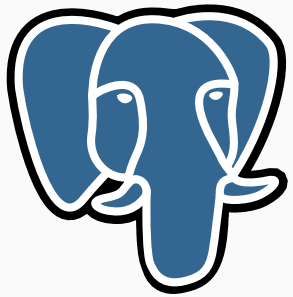
Change Event Structure

- Key (PK of table) and Value
- Payload: **Before** state, **After** state, **Source** info
- Serialization format:
  - JSON
  - Avro (with Confluent Schema Registry)

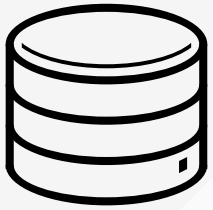
```
{
  "schema": {
    ...
  },
  "payload": {
    "before": null,
    "after": {
      "id": 1004,
      "first_name": "Anne",
      "last_name": "Kretchmar",
      "email": "annek@noanswer.org"
    },
    "source": {
      "name": "dbserver1",
      "server_id": 0,
      "ts_sec": 0,
      "file": "mysql-bin.000003",
      "pos": 154,
      "row": 0,
      "snapshot": true,
      "db": "inventory",
      "table": "customers"
    },
    "op": "c",
    "ts_ms": 1486500577691
  }
}
```

Debezium Connectors

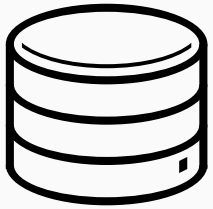
- **MySQL**
- **Postgres**
- **MongoDB**
- **SQL Server**
- **Oracle** (Tech Preview, based on XStream)
- Possible future additions
  - Cassandra?
  - MariaDB?



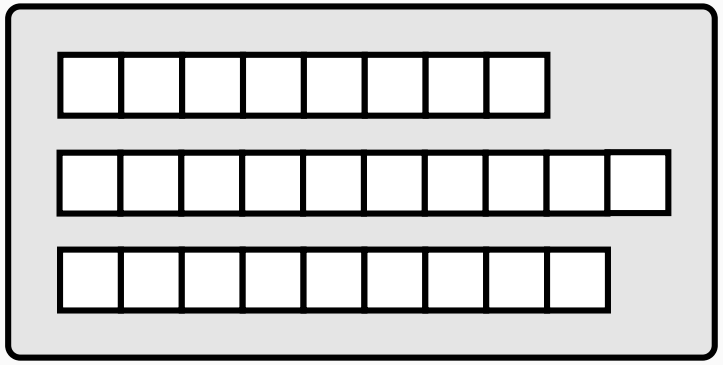
CDC with Debezium and Kafka Connect



MySQL

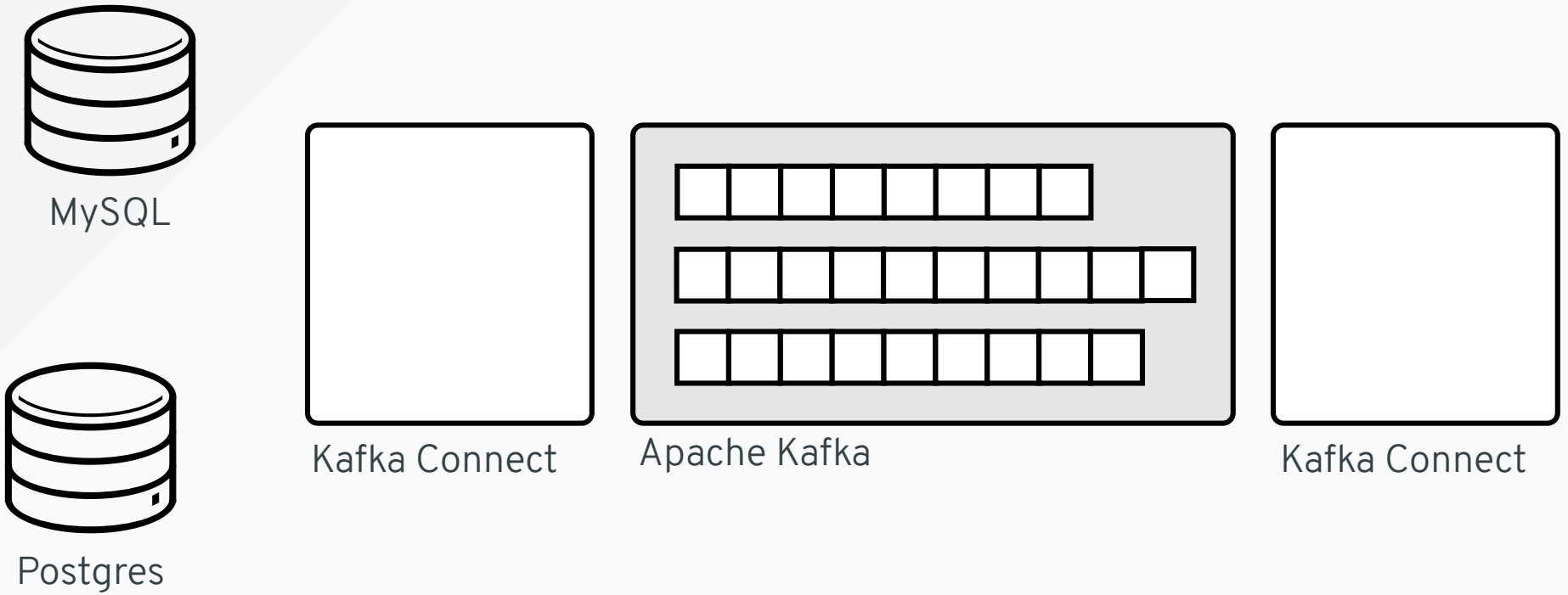


Postgres

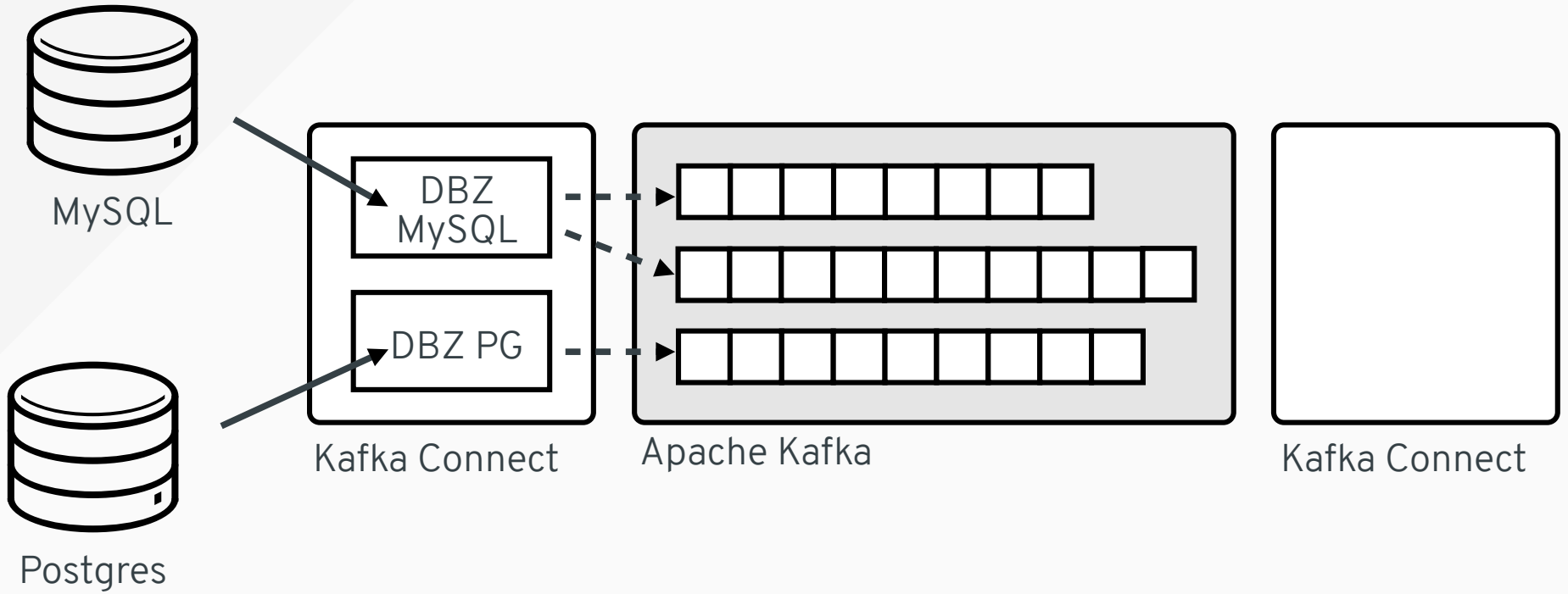


Apache Kafka

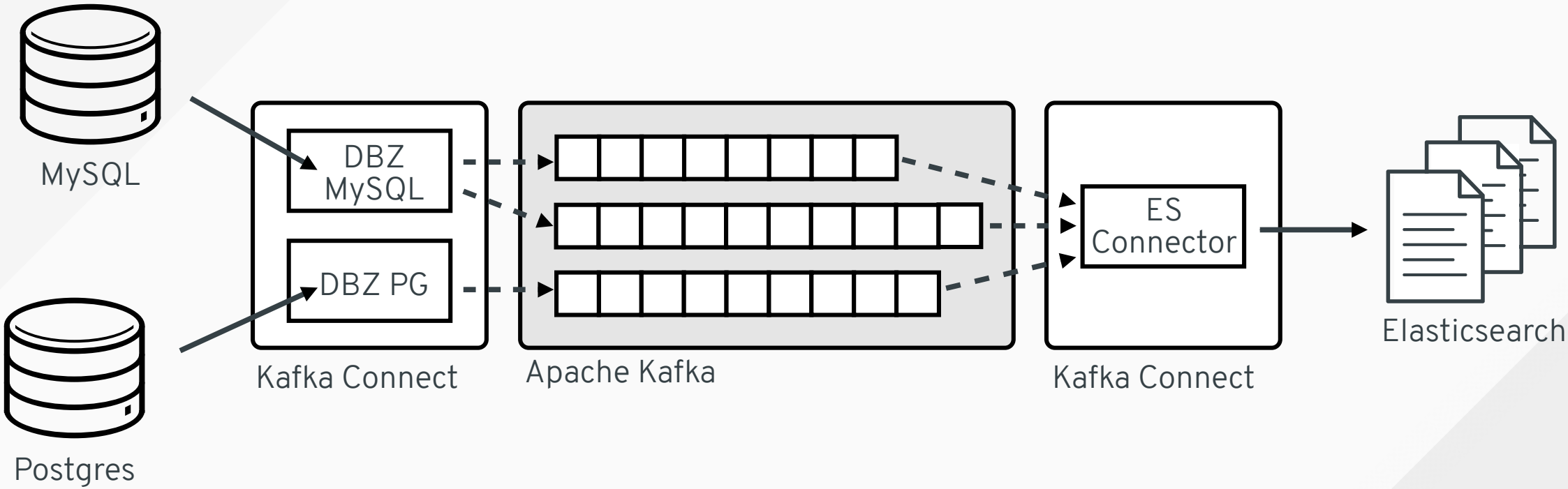
CDC with Debezium and Kafka Connect



CDC with Debezium and Kafka Connect



CDC with Debezium and Kafka Connect

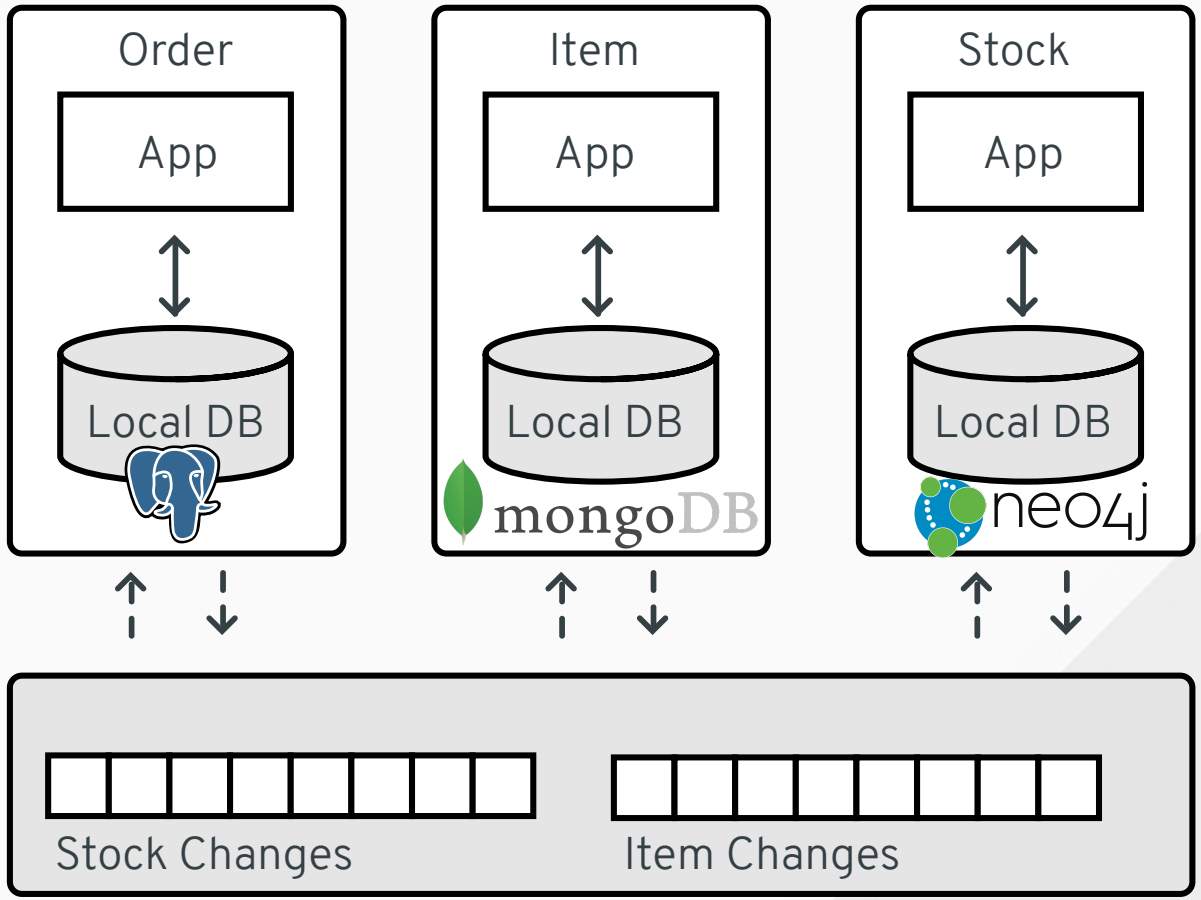



Microservice
CDC Patterns

Pattern: Microservice Data Synchronization

Microservice Architectures

- Propagate data between different services **without coupling**
- Each service keeps **optimised views locally**

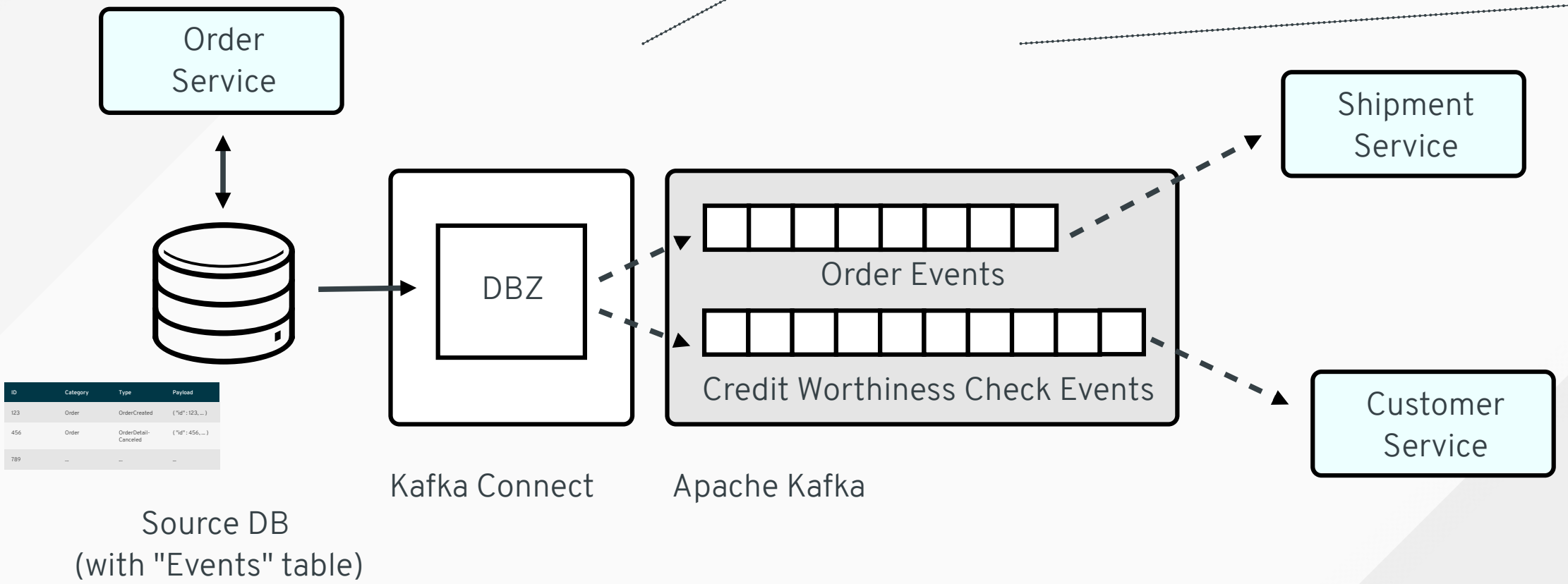


Pattern: Outbox

Separate Events Table

"Outbox" table

Id	AggregateType	AggregateId	Type	Payload
ec6e	Order	123	OrderCreated	{ "id" : 123, ... }
8af8	Order	456	OrderDetailCanceled	{ "id" : 456, ... }
890b	Customer	789	InvoiceCreated	{ "id" : 789, ... }



Pattern: Microservice Extraction

Migrating from Monoliths to Microservices

- **Extract microservice** for single component(s)
- Keep write requests **against running monolith**
- **Stream changes** to extracted microservice
- Test new functionality
- **Switch over**, evolve schema only afterwards

Pattern: Leverage the Powers of SMTs

Single Message Transformations

- **Aggregate** sharded tables to single topic
- **Keep compatibility** with existing consumers
- **Format conversions**, e.g. for dates
- Ensure compatibility with sink connectors
  - Extracting "after" state only
  - Expand MongoDB's JSON structures


Demo

Running Debezium on Kubernetes

AMQ Streams: Enterprise Distribution of Apache Kafka

- Provides
  - **Container images** for Apache Kafka, Connect, Zookeeper and MirrorMaker
  - **Operators** for managing/configuring Apache Kafka clusters, topics and users
  - Kafka Consumer, Producer and Admin clients, Kafka Streams
- Supported by Red Hat
- Upstream Community: **Strimzi**

Summary

- **CDC** enables use cases such as replication, microservices data exchange and much more
- **Debezium:** CDC for a growing number of databases
- **Contributions welcome!**
- Tell us about your **feature requests and ideas!**

Resources

- **Website:** http://debezium.io/
- **Source code**, examples, Compose files etc. https://github.com/debezium
- **Discussion group** https://groups.google.com/forum/#!forum/debezium
- **Strimzi** (Kafka on Kubernetes/OpenShift) http://strimzi.io/
- **Latest news:** @debezium