


A workshop background with various tools and machinery. The scene is filled with metal parts, pipes, and tools, creating a sense of industrial activity. The lighting is warm and focused on the foreground tools.

# Change Data Streaming Patterns With Debezium & Apache Flink

**Gunnar Morling**

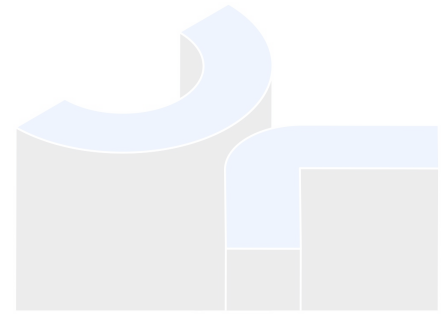
Senior Staff Software Engineer, Decodable

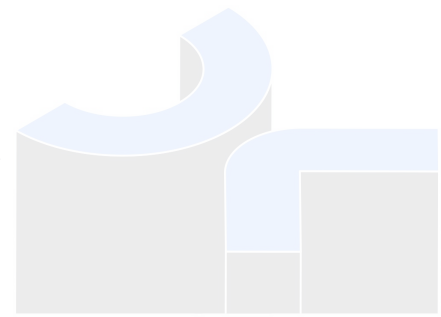
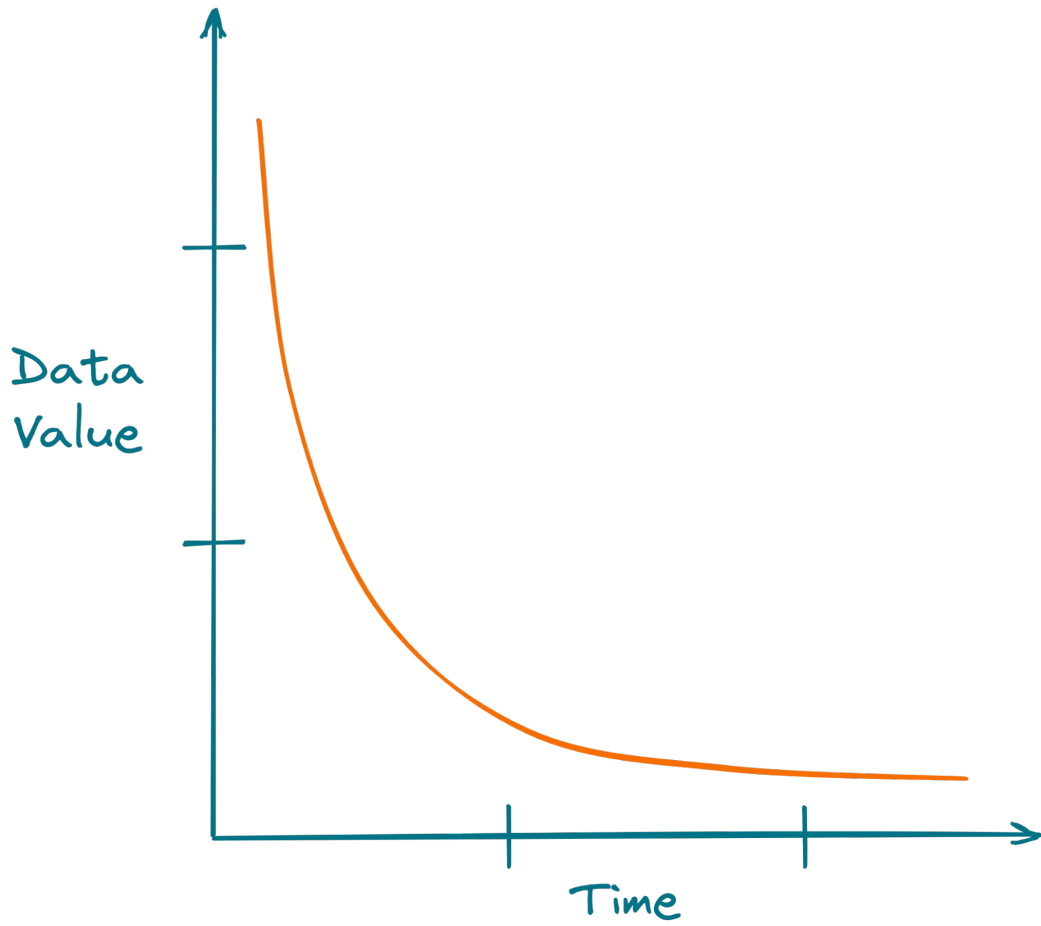
 @gunnarmorling



*The world is  
**real-time.***

*So should be  
**your data.***







# Today's Mission

Learn About...

Outbox  
Pattern

Strangler Fig  
Pattern

Audit  
Logs

...implemented with change data capture.



# Gunnar Morling



- Software engineer at **Decodable**
- Former project lead of **Debezium**
- **kcctl** 🧸, JfrUnit, ModiTect, MapStruct
- Spec Lead for Bean Validation 2.0
- Java Champion



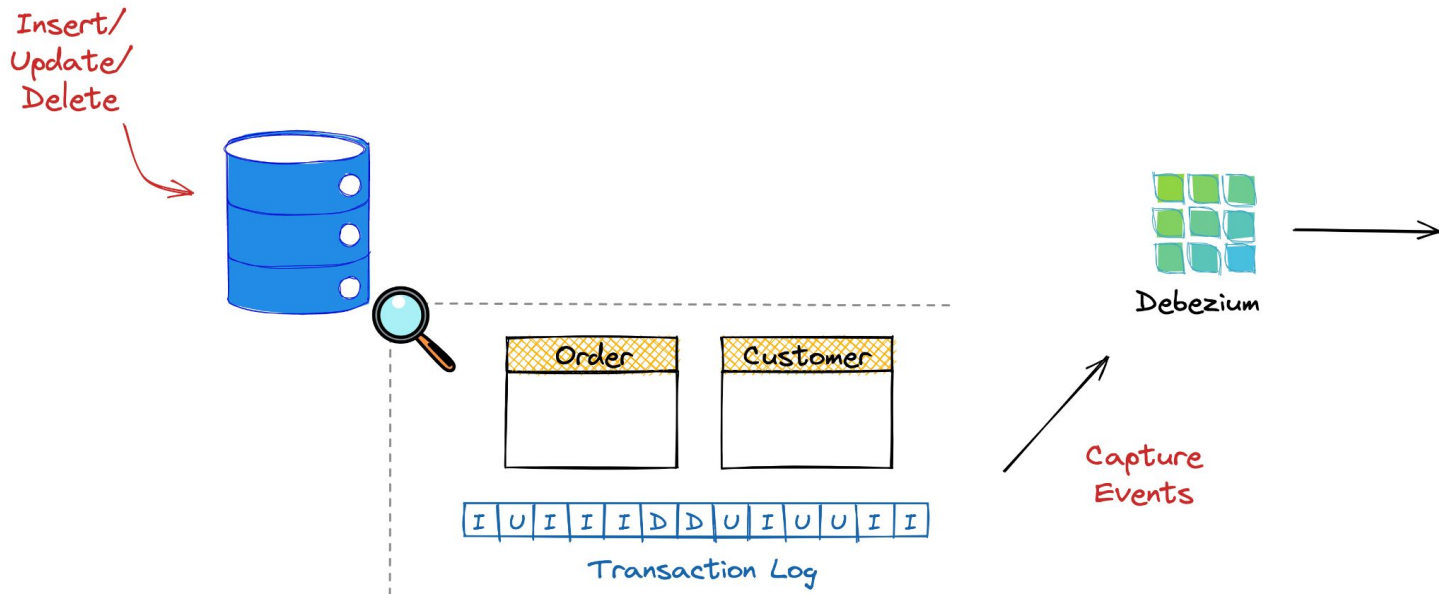


# The Tools



# Debezium

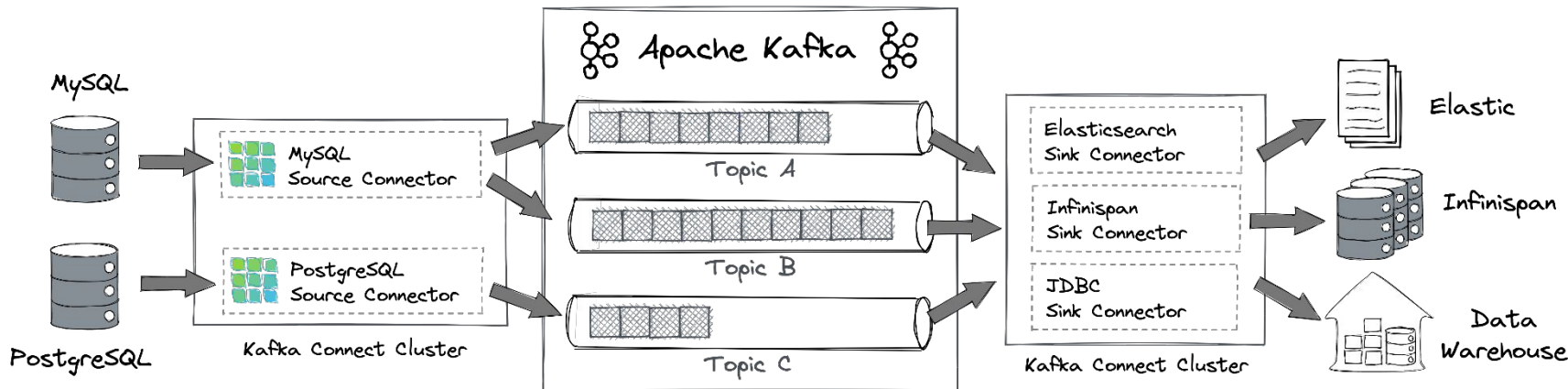
## Log-Based Change Data Capture





# Debezium

## Log-Based Change Data Capture



- **Taps into TX log** to capture INSERT/UPDATE/DELETE events
- Typically propagated to consumers via **Apache Kafka**





# Debezium

## Open-Source Change Data Capture

- A CDC Platform
  - Based on **transaction logs**
  - Snapshotting, filtering, etc.
  - Outbox support
  - Web-based **UI**
- Fully **open-source**, very active community
- Large production deployments





# Debezium: Data Change Events

- **Old** and **new row state**
- **Metadata** on table, TX id, etc.
- **Operation type**, timestamp

```
{
  "before": {
    "id": 1,
    "first_name": "Anne",
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "after": {
    "id": 1,
    "first_name": "Anne",
    "last_name": "Grant",
    "email": "annek@noanswer.org"
  },
  ...
}
```



# Debezium: Data Change Events

- **Old and new row state**
- **Metadata** on table, TX id, etc.
- **Operation type**, timestamp

```
{
  "before": {
    "id": 1,
    "first_name": "Anne",
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "after": {
    "id": 1,
    "first_name": "Anne",
    "last_name": "Grant",
    "email": "annek@noanswer.org"
  },
  ...
}
```



# Debezium: Data Change Events

- **Old** and **new row state**
- **Metadata** on table, TX id, etc.
- **Operation type**, timestamp

```
{
  ...
  "source": {
    "version": "1.6.0.Alpha1",
    "connector": "postgresql",
    "name": "PostgreSQL_server",
    "ts_ms": 1559033904863,
    "snapshot": true,
    "db": "postgres",
    "schema": "public",
    "table": "customers",
    "txId": 555,
    "lsn": 24023128
  },
  "op": "u",
  "ts_ms": 1559033904863
}
```



# Debezium

## Becoming the De-Facto CDC Standard

The screenshot shows the Google Cloud Spanner documentation page for building change streams connections to Kafka. The page is titled "Build change streams connections to Kafka" and is part of the "Guides" section. It includes a "Core concepts" section, a "Debezium" section, a "Kafka connector" section, and a "Kafka connector output" section. The page explains how to use the Kafka connector to consume and forward Cloud Spanner change streams data. It also provides a list of "Core concepts" and "Debezium" details.

The screenshot shows the ScyllaDB documentation page for the Scylla CDC Source Connector. The page is titled "Scylla CDC Source Connector" and is part of the "Getting Started" section. It includes a "Scylla CDC Source Connector" section, a "Getting Started" section, and a "Capabilities" section. The page explains that the Scylla CDC Source Connector is a source connector capturing row-level changes in the tables of a Scylla cluster. It is a Debezium connector, compatible with Kafka Connect (with Kafka 2.6.0+) and built on top of scylla-cdc-java library. The source code of the connector is available at GitHub.

Google Cloud Spanner

ScyllaDB

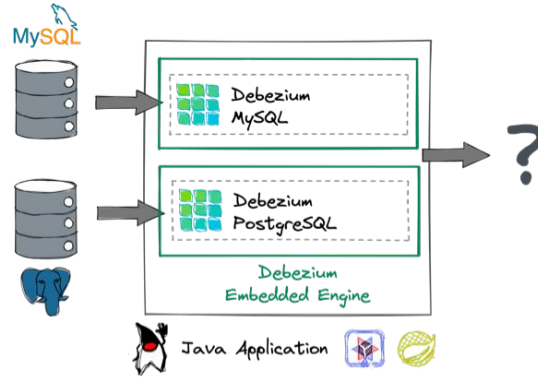


# Debezium

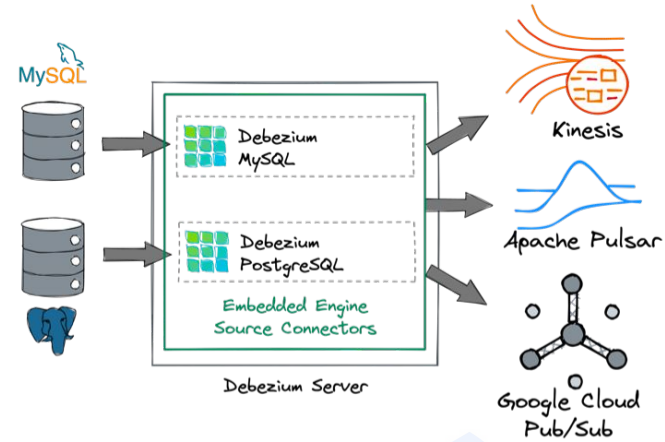
## Deployment Options



Kafka Connect



Debezium Embedded Engine



Debezium Server

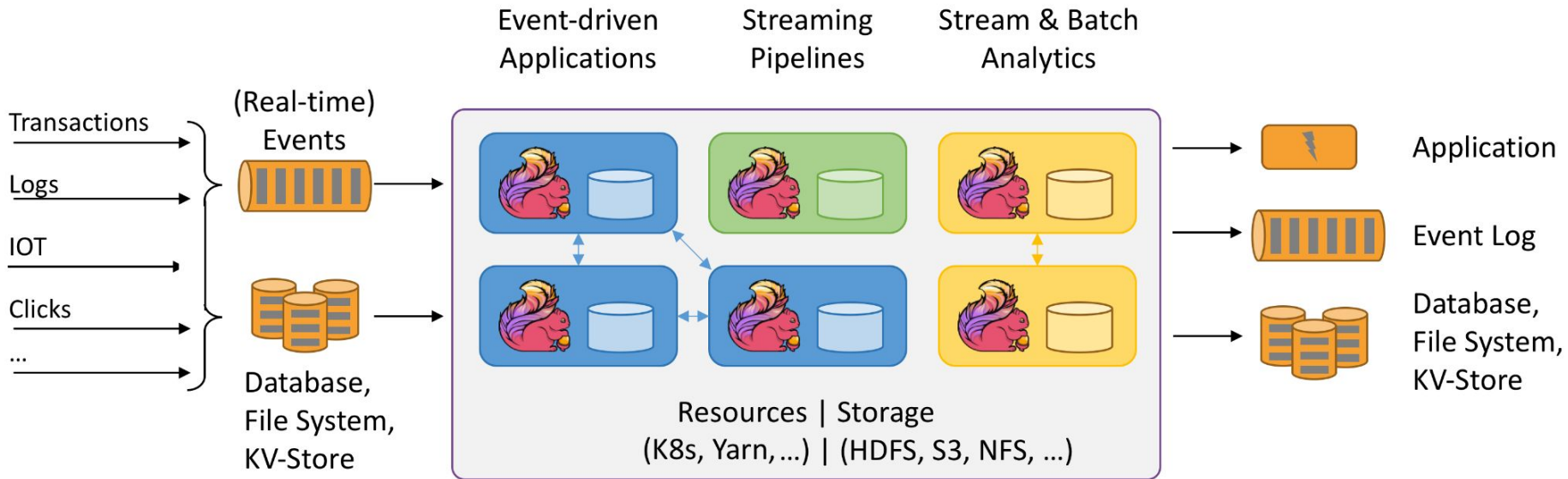


# Apache Flink



# Apache Flink

## Stateful Computations over Data Streams



<https://flink.apache.org/>





# Apache Flink

## APIs for Application Development

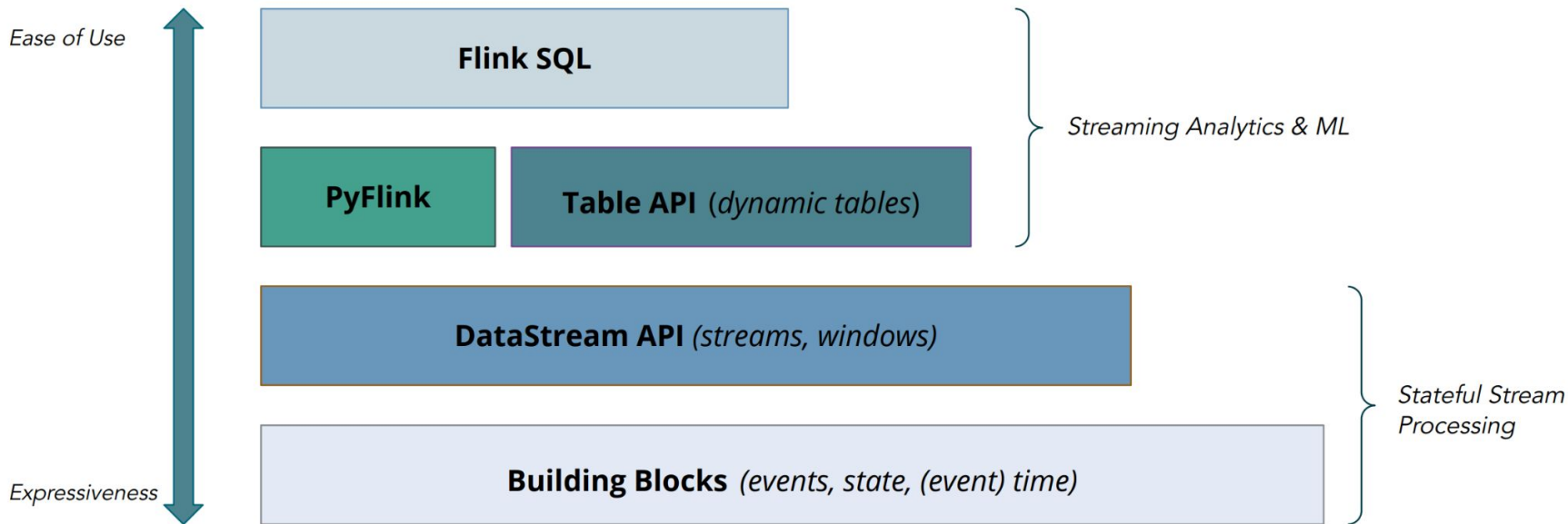
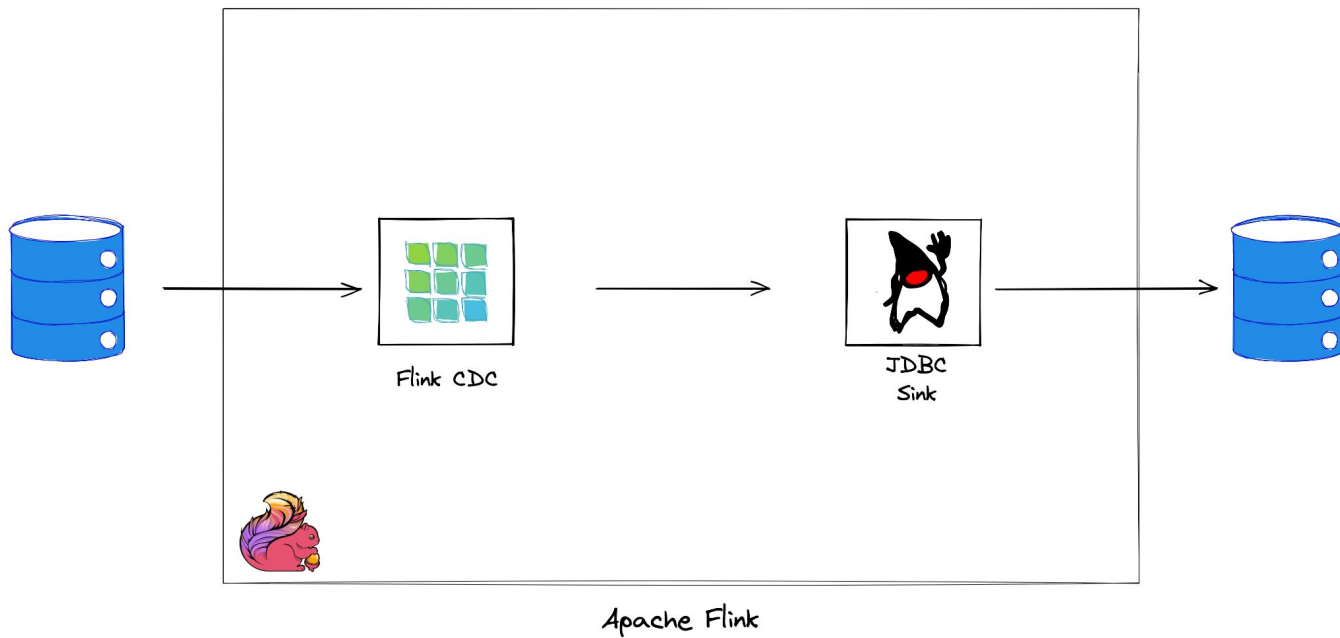


Image source: "Change Data Capture with Flink SQL and Debezium, by Marta Paes at DataEngBytes  
(<https://noti.st/morsapaes/liQzgs/change-data-capture-with-flink-sql-and-debezium>)



# Apache Flink

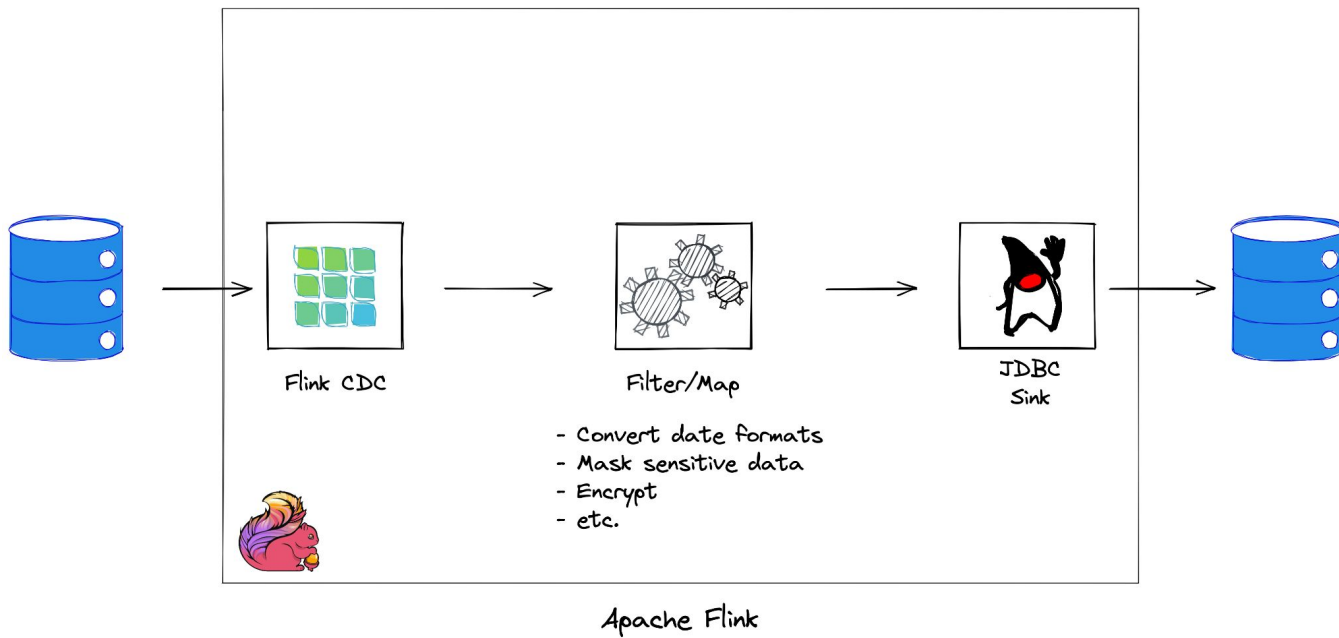
## Stream Processing of Change Data Events





# Apache Flink

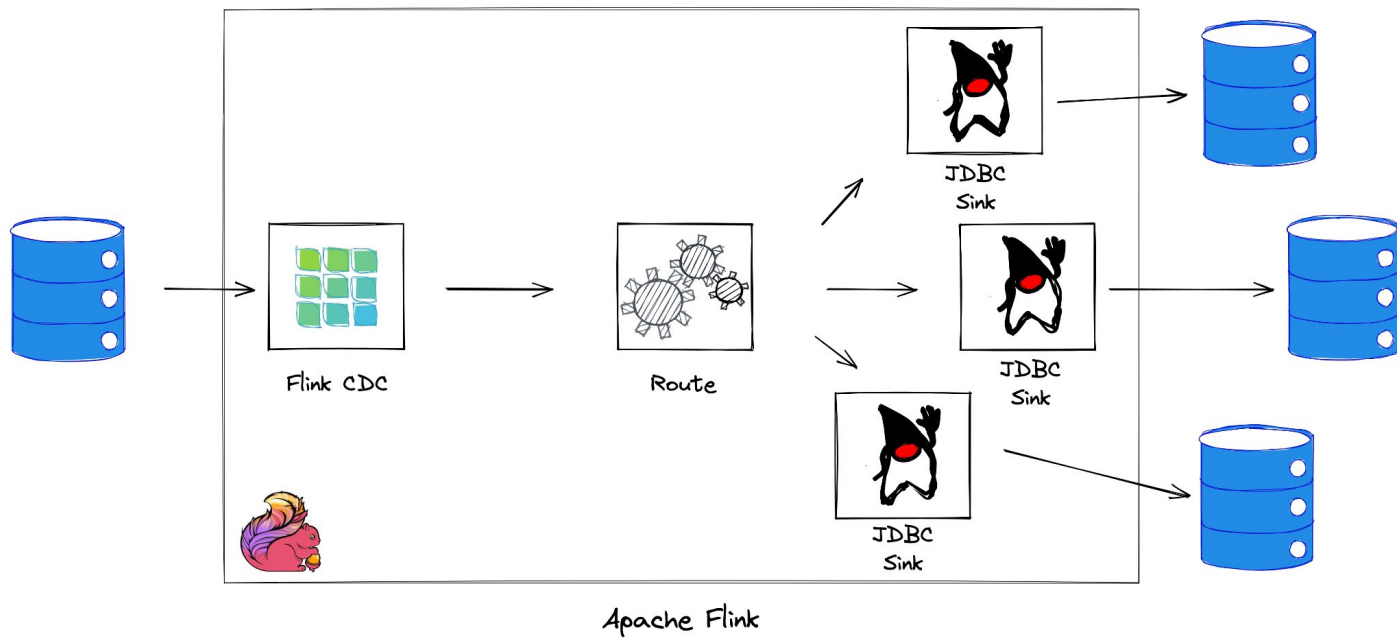
## Stream Processing of Change Data Events





# Apache Flink

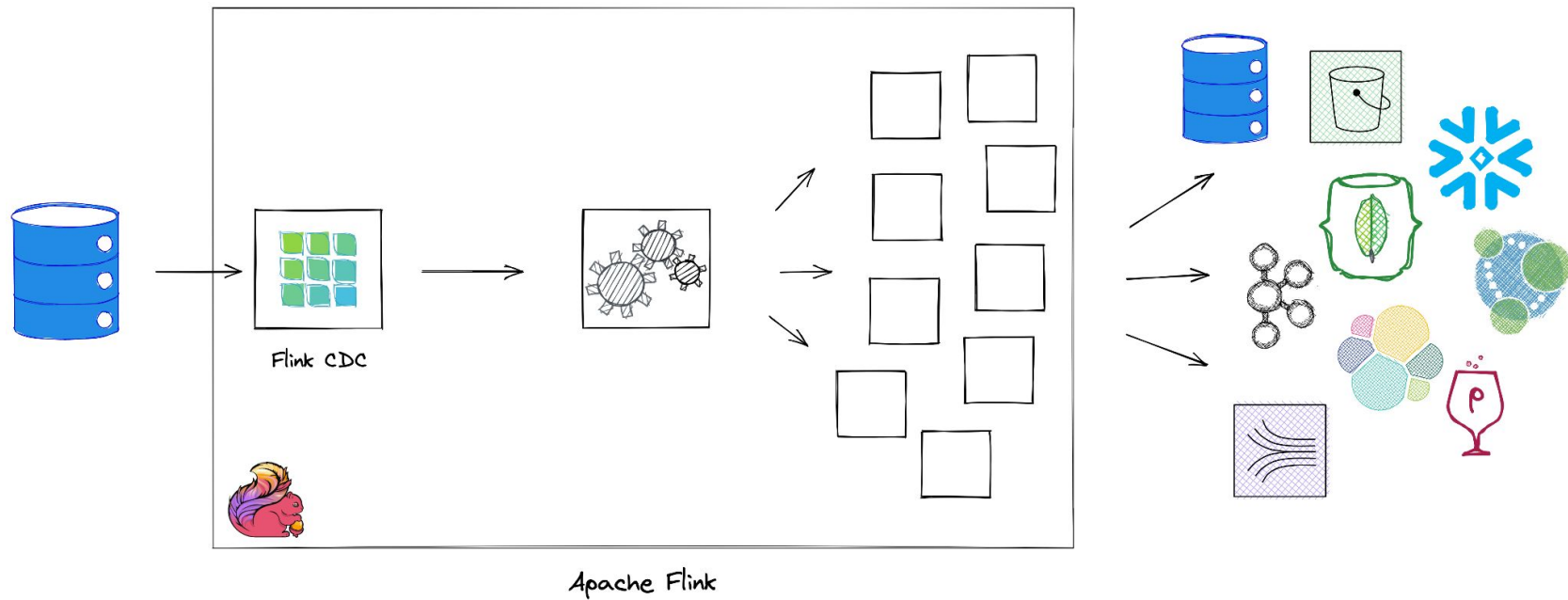
## Stream Processing of Change Data Events





# Apache Flink

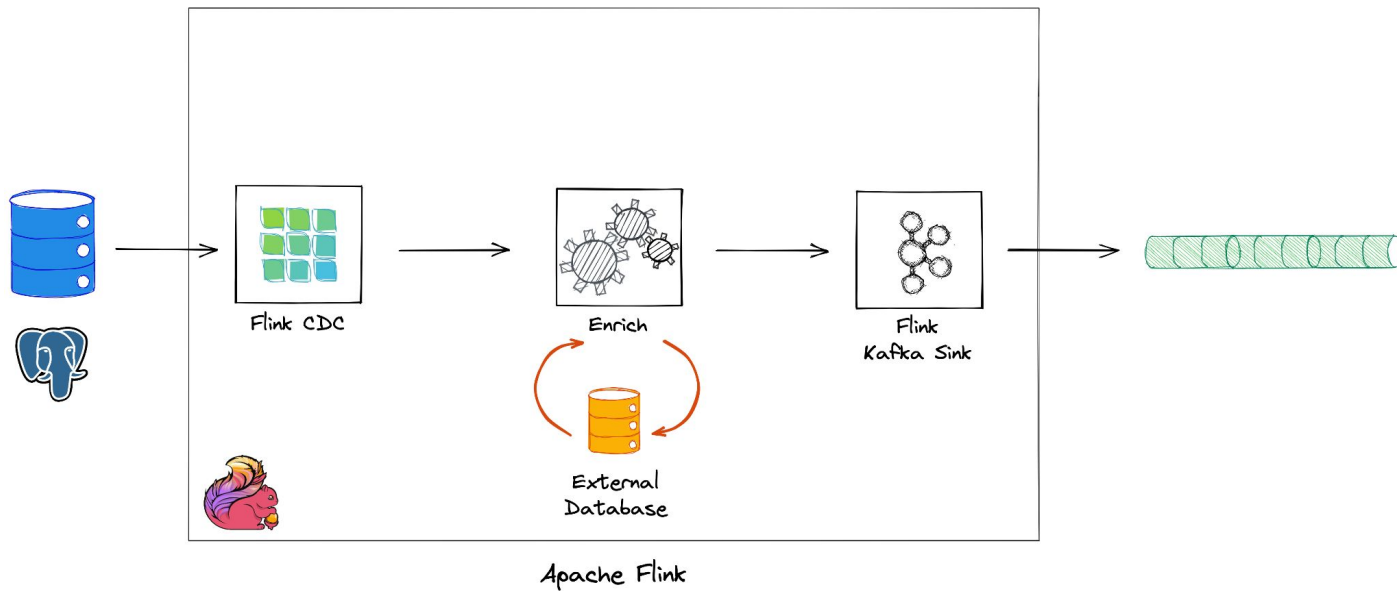
## Stream Processing of Change Data Events





# Apache Flink

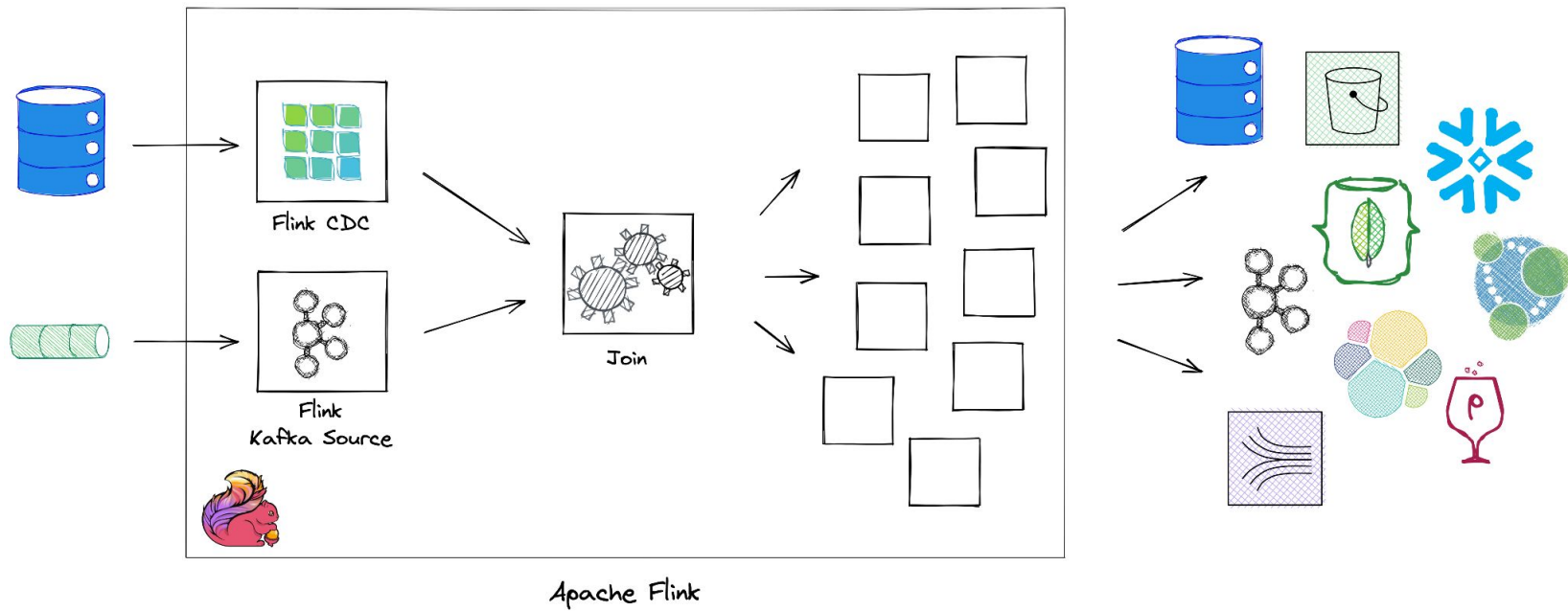
## Stream Processing of Change Data Events





# Apache Flink

## Stream Processing of Change Data Events



# Outbox Pattern

POSTBRIEFKASTEN

POST'S'SE  
Berliner Straße 17-25  
Filiale 21481 Leuenburg

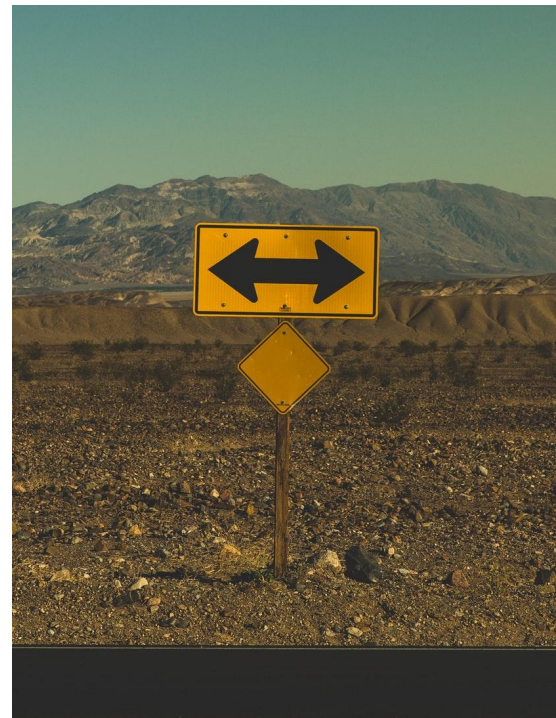
|                                                  |                                                  |                                            |  |
|--------------------------------------------------|--------------------------------------------------|--------------------------------------------|--|
| Montag-Freitag<br>Samstag<br>Sonntag             | letzte<br>Tageslieferung<br>15:15<br>11:30       |                                            |  |
| Nächster Briefkasten mit<br>späteren Lieferungen | Berliner Straße 17-25<br>Filiale 21481 Leuenburg | Nachbildung eines<br>Briefkastens von 1896 |  |





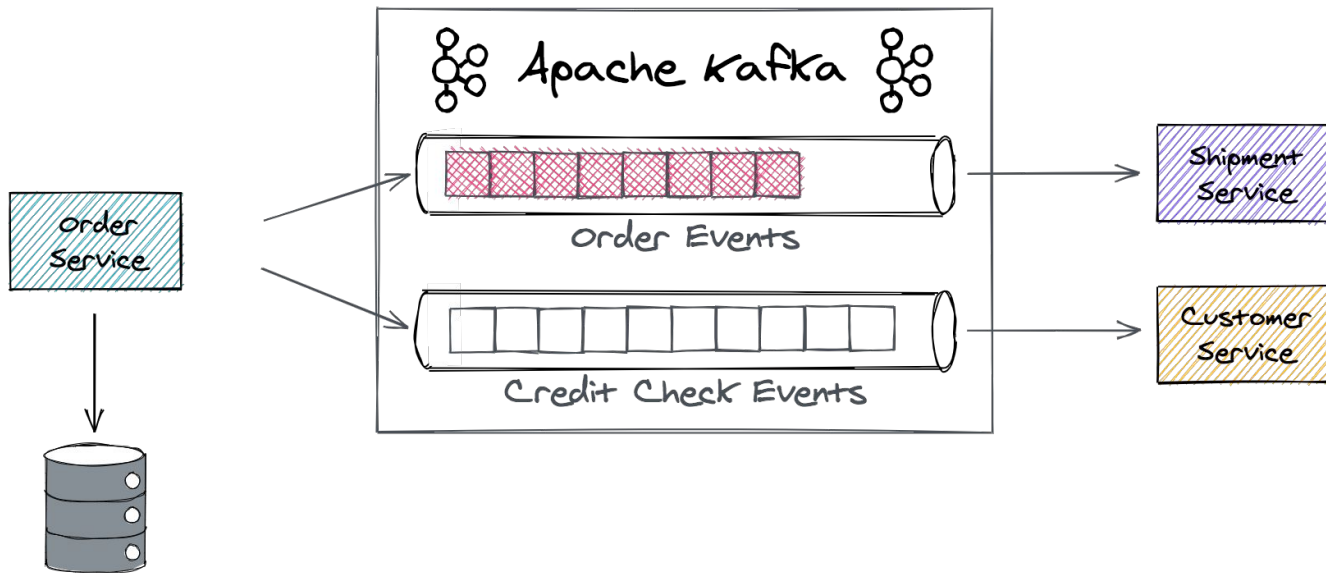
# Challenge: Microservices Data Exchange

- Services need to update their **database**,
- send messages to **other services**,
- and that **consistently!**





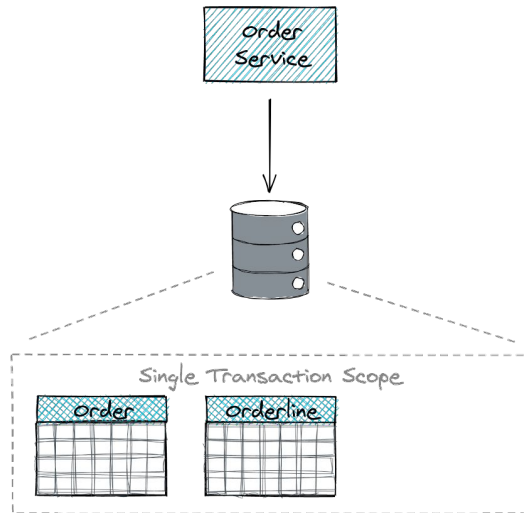
# Outbox Pattern



"Dual writes" are **prone to inconsistencies!**

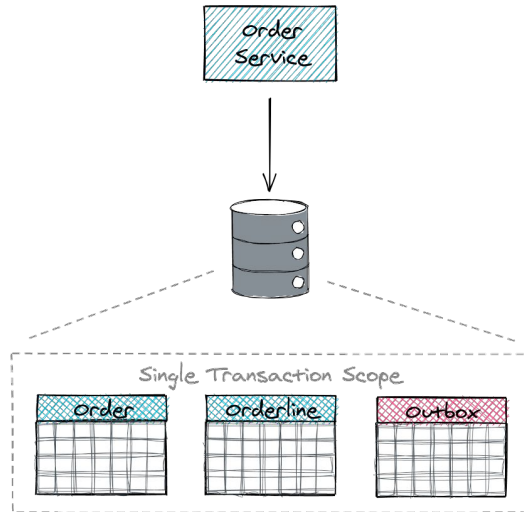


# Outbox Pattern



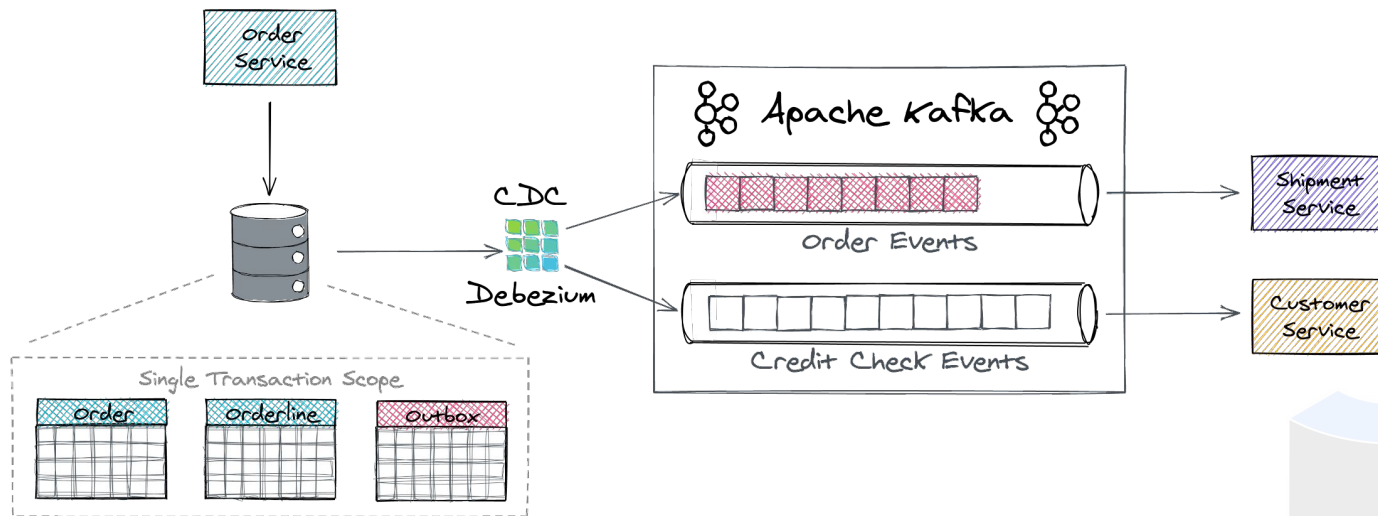


# Outbox Pattern





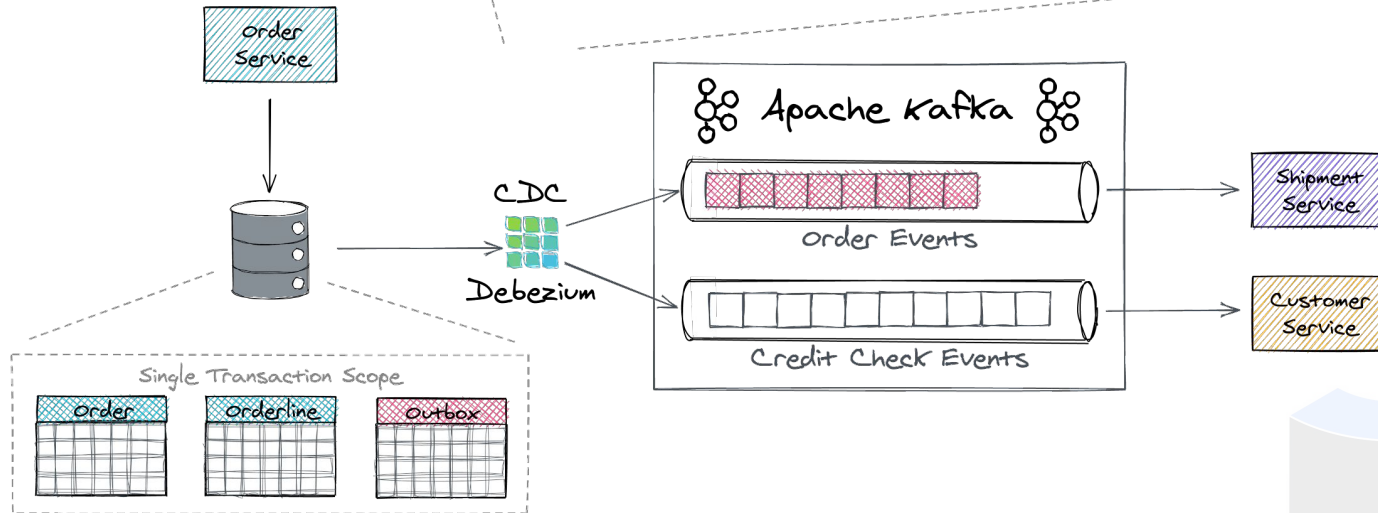
# Outbox Pattern





# Outbox Pattern

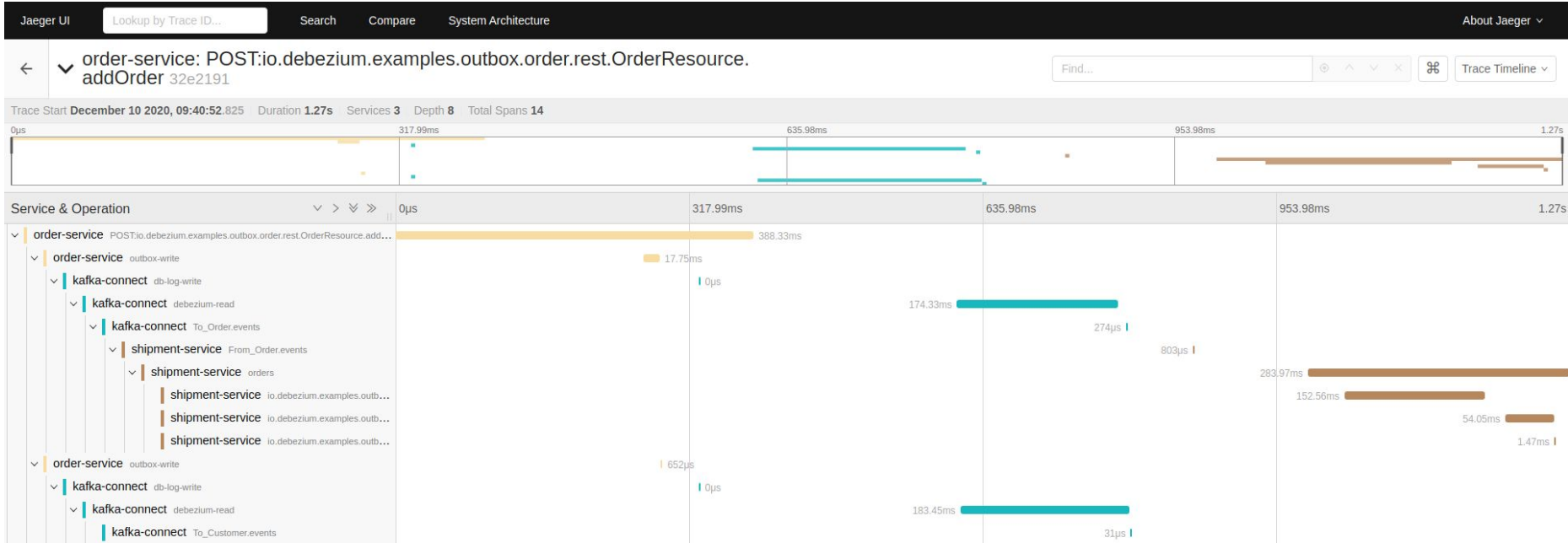
| Outbox Table |               |             |                  |                 |
|--------------|---------------|-------------|------------------|-----------------|
| Id           | AggregateType | AggregateId | EventType        | Payload         |
| ec6e         | Order         | 123         | orderCreated     | {"id": 123,...} |
| 8af8         | Order         | 456         | orderLineChanged | {"id": 456,...} |
| ...          |               |             |                  |                 |





# Outbox Pattern

## Observability





## Variation on Postgres

### pg\_logical\_emit\_message()

- Directly writing arbitrary messages **to the WAL**
- No need for an outbox table

```
SELECT pg_logical_emit_message(  
  true,  
  'outbox-messages',  
  '{ "id" : "ec6e", "aggregateType" : "Order", ... }'  
);
```



A low-angle photograph of a massive, ancient tree in a dense forest. The tree's trunk is thick and gnarled, with a complex network of aerial roots hanging down from the canopy. The roots are light brown and appear to be in various stages of growth, some still thin and others thicker. The canopy is filled with lush green leaves, and the sky is visible through the branches. The overall scene is one of a well-preserved, old-growth forest.

# Strangler Fig Pattern



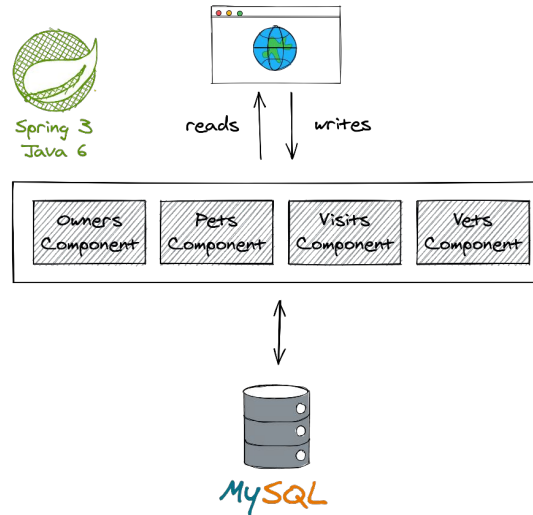
# Challenge: Migrating Systems

- **Gradually evolve** from old into new
- Support temporary **coexistence**
- **Avoid big bang** cut-over



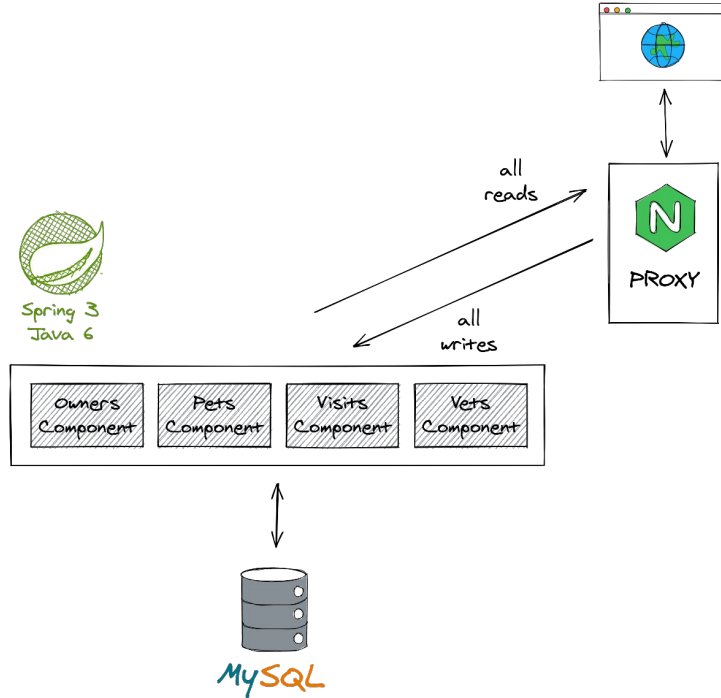


# CDC-based Strangler Fig Pattern



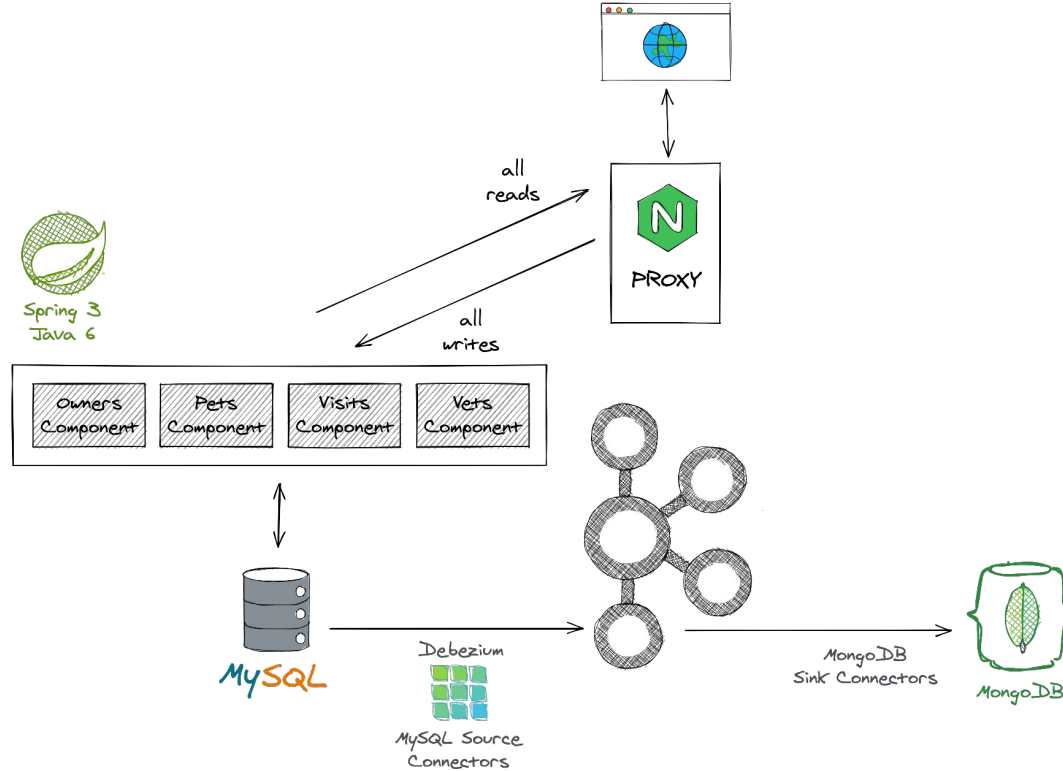


# CDC-based Strangler Fig Pattern



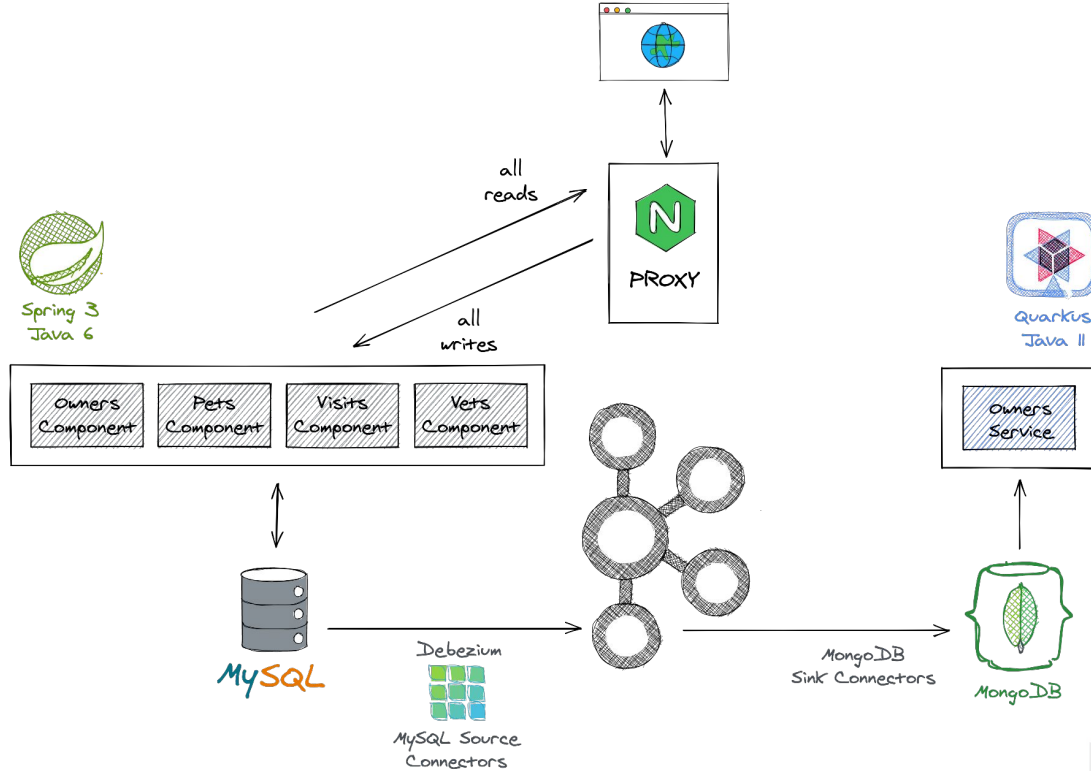


# CDC-based Strangler Fig Pattern



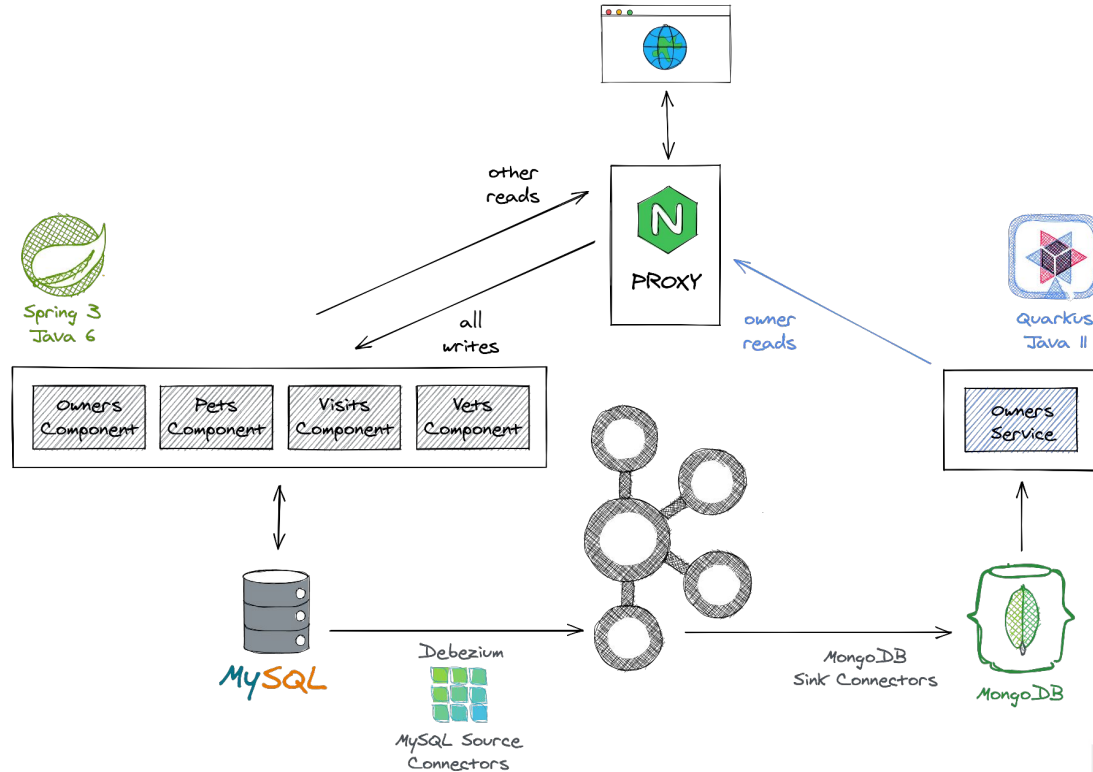


# CDC-based Strangler Fig Pattern



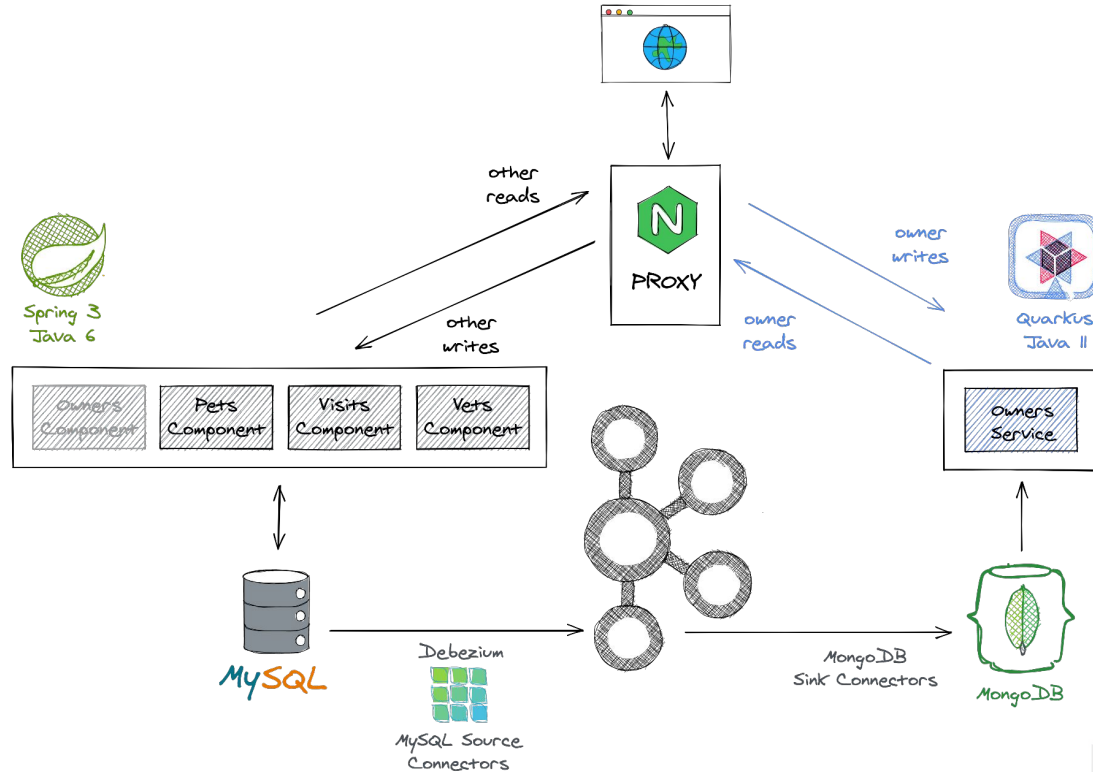


# CDC-based Strangler Fig Pattern





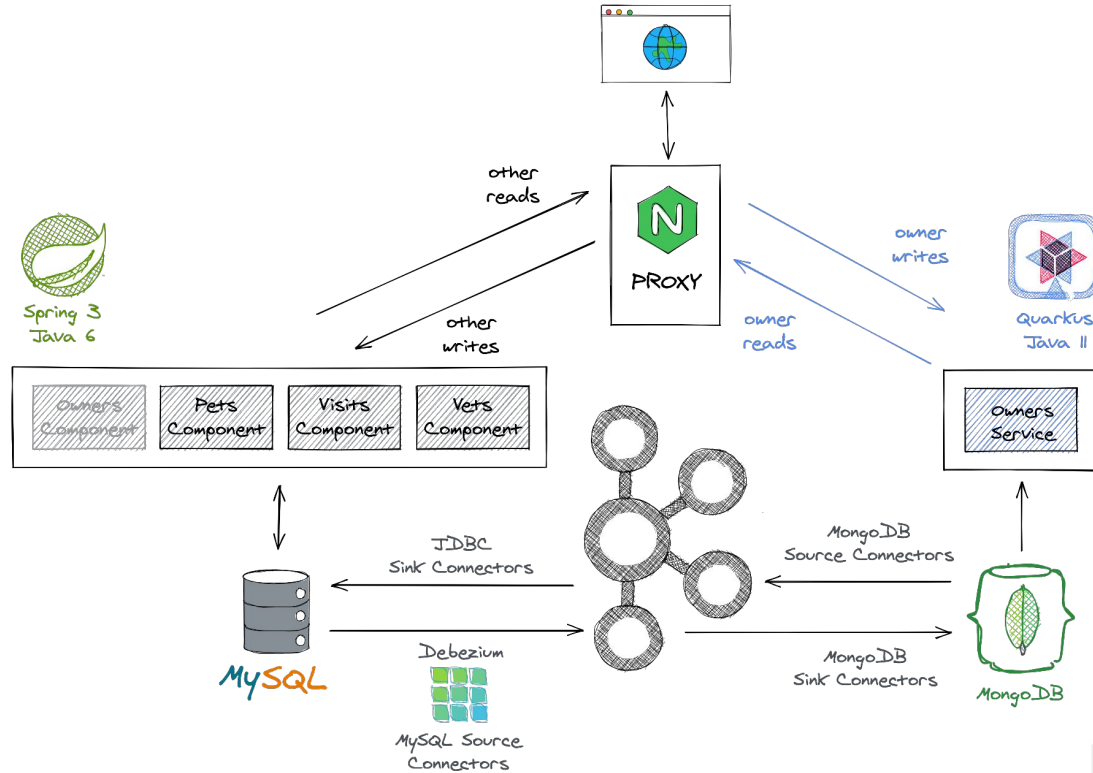
# CDC-based Strangler Fig Pattern





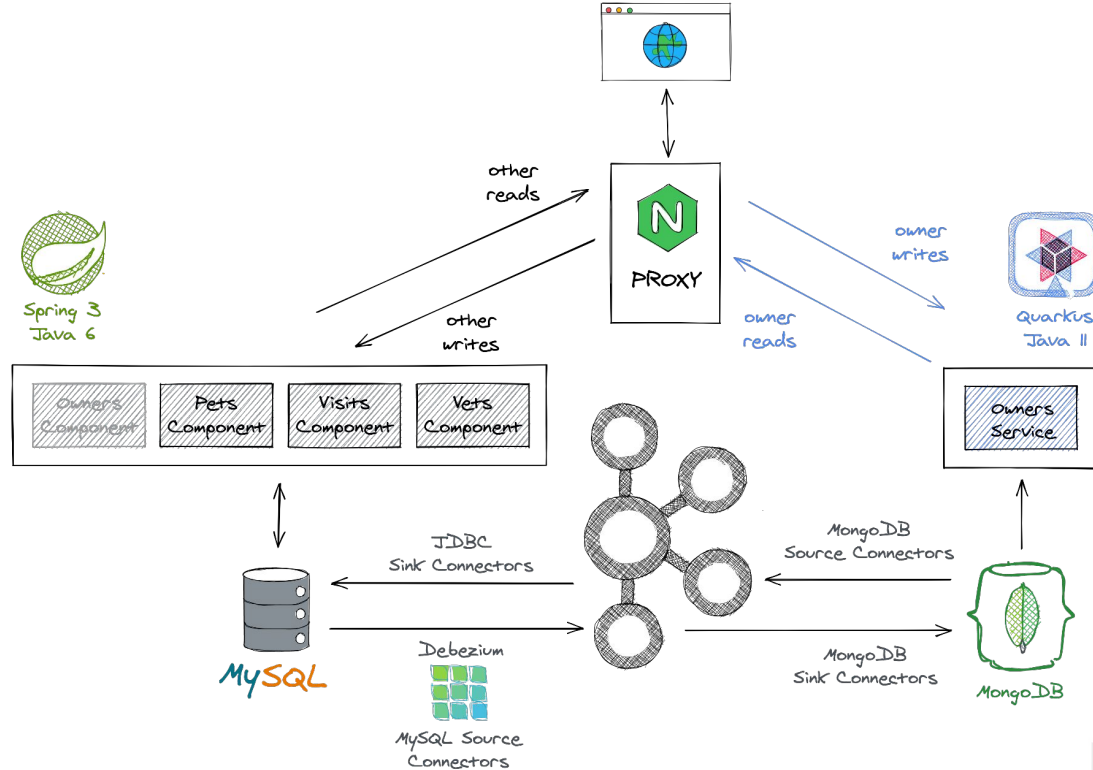


# CDC-based Strangler Fig Pattern





# CDC-based Strangler Fig Pattern



Demo Repo: <https://bit.ly/ff21-sfp>



# Benefits

- Incremental migration → **“baby steps”**
- **Pause or stop** migration without losing spent efforts
- Migration steps ideally **reversible**

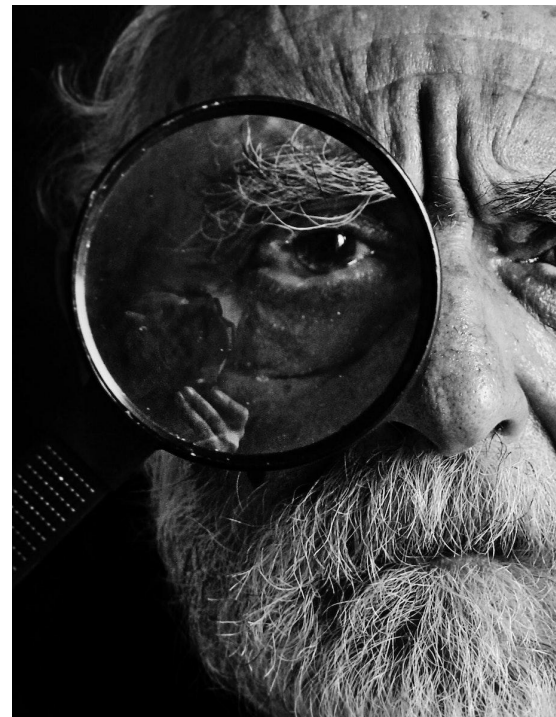
Rationale: ⚠ **minimize risk** ⚠





# CDC Pipeline Considerations

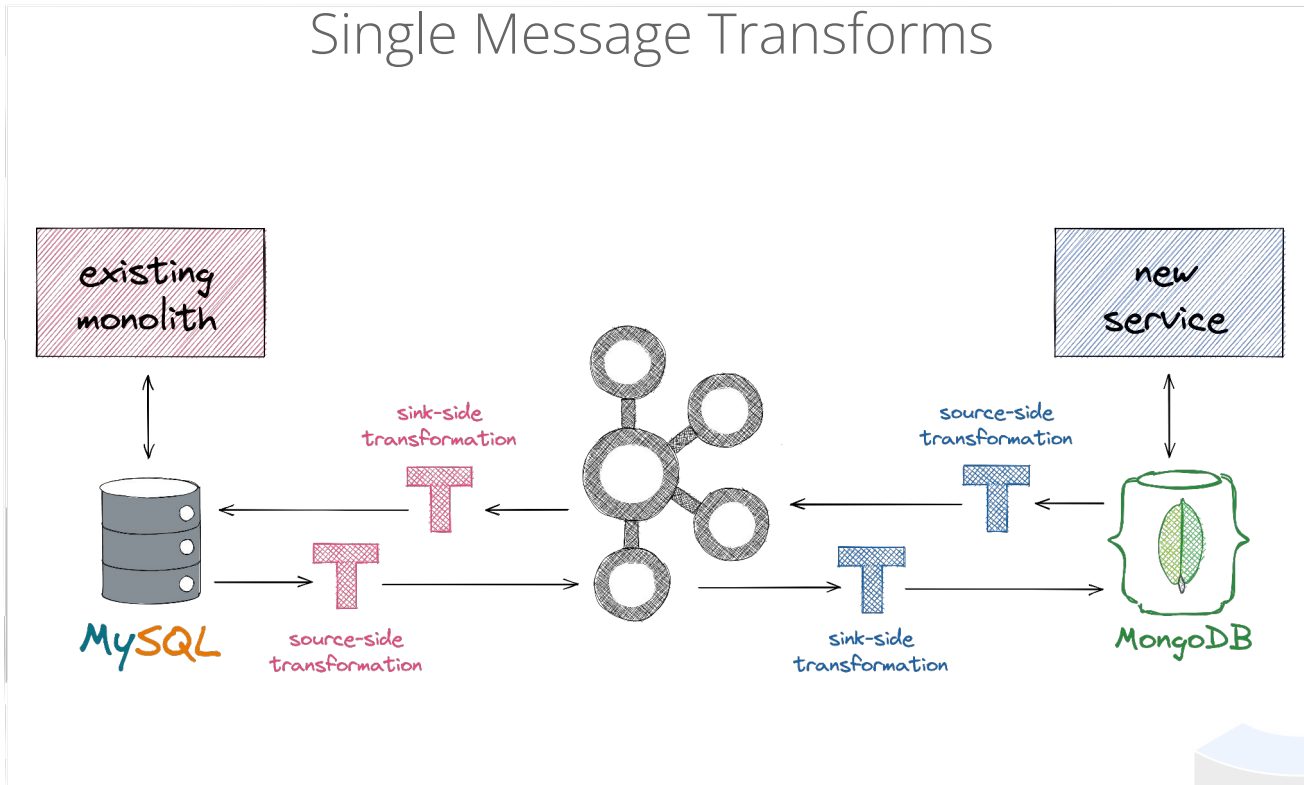
- **data model leaking** from monolith ?
- ~~"1:1 replication"~~ → **building aggregates ?**





# Enhanced CDC Processing

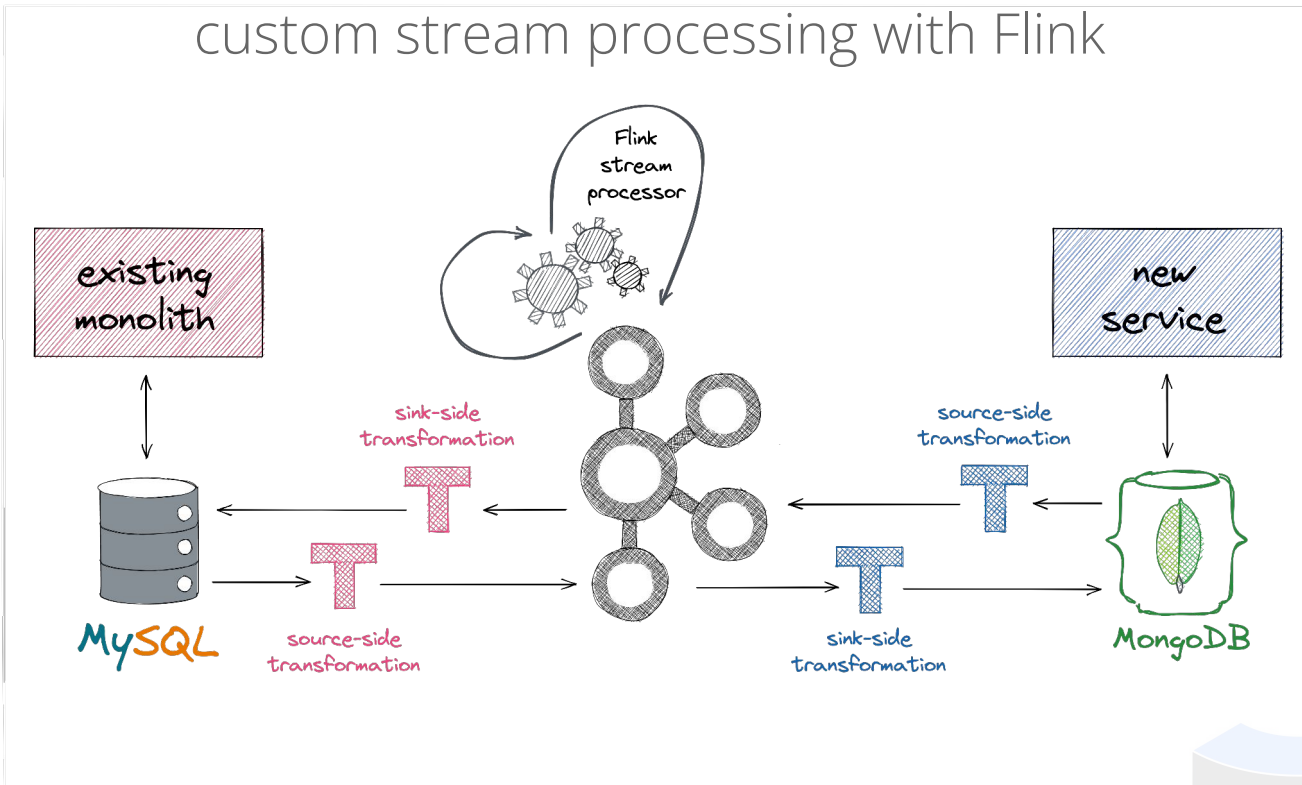
## Single Message Transforms





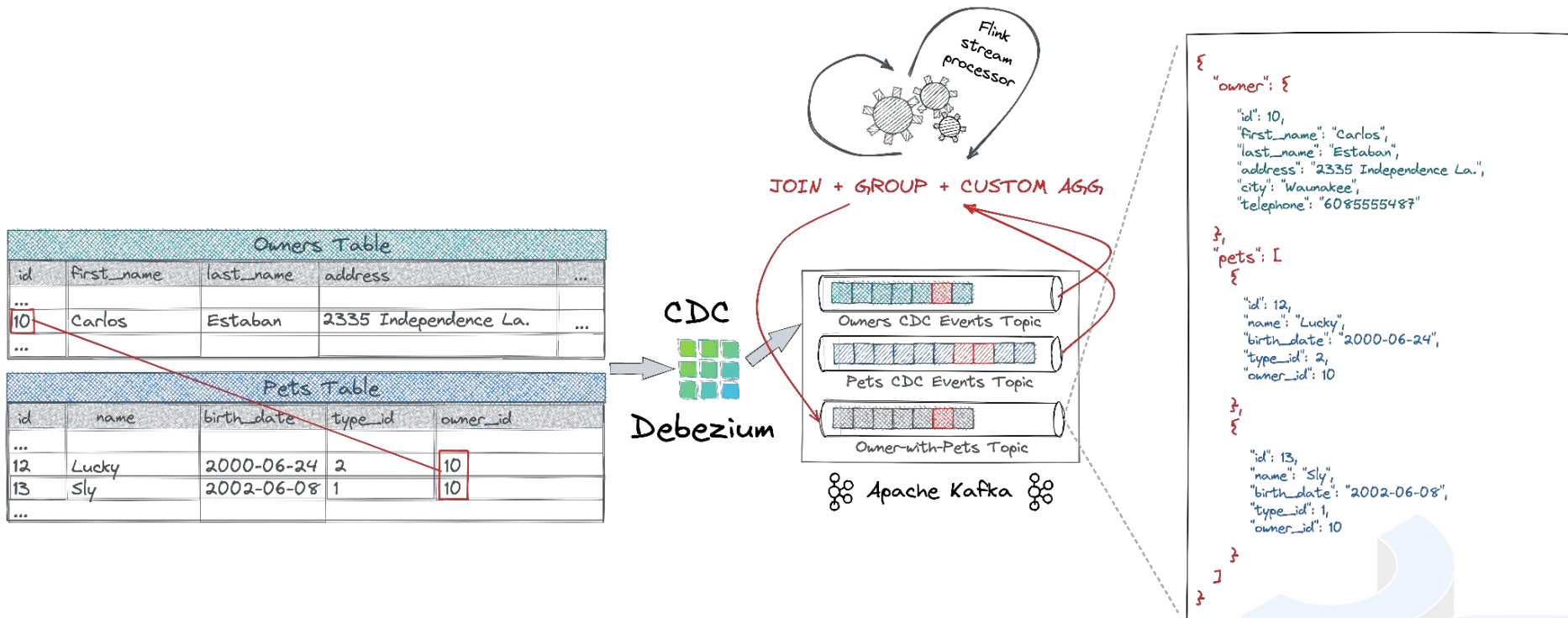
# Enhanced CDC Processing

custom stream processing with Flink





# Example: Join With Custom Aggregation





# Example: Join With Custom Aggregation

## Flink SQL

```
INSERT INTO flink_owner_with_pets
SELECT owp.owner.id as key,owp.owner,owp.pets FROM (
  SELECT ff21_agg_owner_pets(owner,pet) as owp FROM (
    SELECT
      ff21_to_owner(o.id, first_name, last_name, address, city, telephone) as owner,
      ff21_to_pet(p.id,name,birth_date,type_id,owner_id) as pet
    FROM dbz_mysql_owners o
    JOIN dbz_mysql_pets p ON p.owner_id = o.id
  ) as t1
  GROUP BY pet.owner_id
) as t2
;
```



# Audit Logs





## Challenge: Capturing Intent

pg\_logical\_emit\_message()

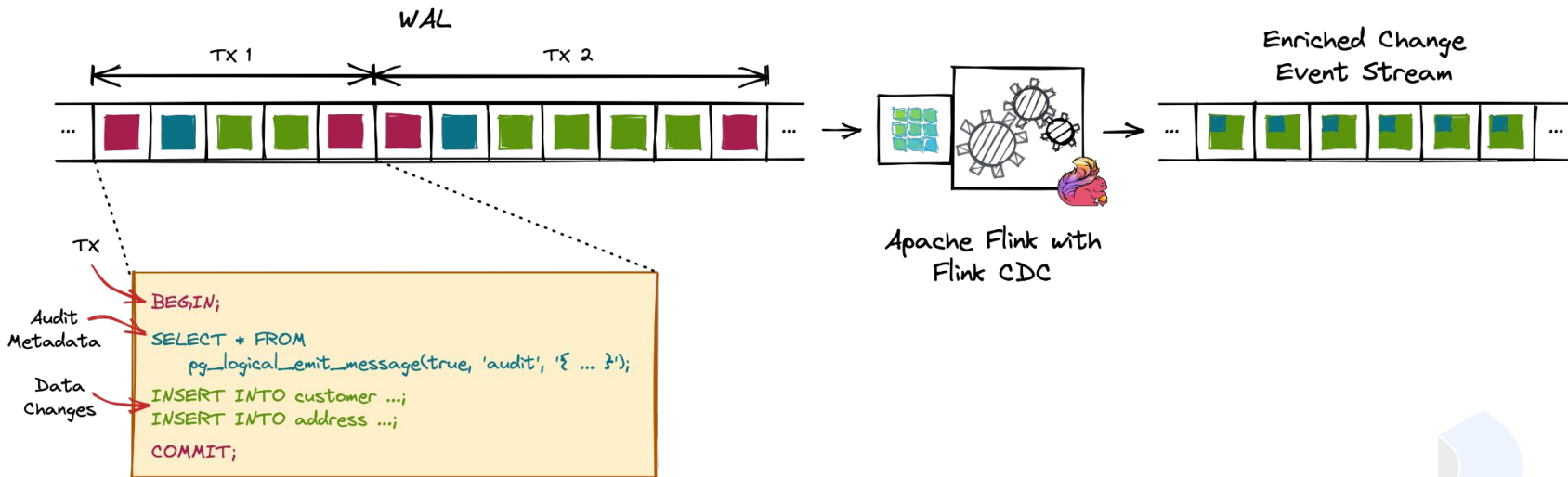
- Pure CDC events **lack metadata** like business user, device id, etc.
- Solution: emit at TX begin, enrich events using Flink

```
SELECT pg_logical_emit_message(  
  true,  
  'audit-data',  
  '{ "client_date" : "2019-08-22 08:12:31", "usecase" : " CREATE VEGETABLE", "user_name" : "farmerbob" }'  
);
```



# Capturing Intent

## Enriching Change Data Events with Metadata





# Capturing Intent

## Enriching Change Events Via Apache Flink

```
public void flatMap(String value, Collector<String> out) throws Exception {
    ChangeEvent changeEvent = mapper.readValue(value, ChangeEvent.class);
    String op = changeEvent.getOp();
    String txId = changeEvent.getSource().get("txId").asText();

    // decoding message
    if (op.equals("m")) {
        Message message = changeEvent.getMessage();

        // an audit metadata message -> put metadata to state
        if (message.getPrefix().equals("audit")) {
            localAuditState = new AuditState(txId, message.getContent());
            return;
        }
        else {
            out.collect(value);
        }
    }
}
```

```
// a change event -> enrich with state stored before
else {
    if (txId != null && localAuditState != null) {
        if (txId.equals(localAuditState.getTxId())) {
            changeEvent.setAuditData(localAuditState.getState());
        }
        else {
            localAuditState = null;
        }
    }

    changeEvent.setTransaction(null);
    out.collect(mapper.writeValueAsString(changeEvent));
}
}
```



**Wrap-Up**



## Take Aways

- The **fresher data** is, the more valuable it is
- **Change Data Capture** and **Debezium**: Powerful tools for realtime change event feeds
- Combining CDC with **stream processing**: Many more possibilities





# Thank You!



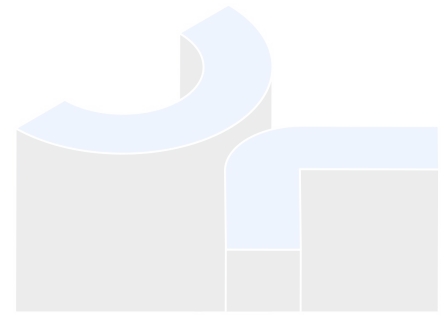
# Q & A

 [gunnar@decodable.co](mailto:gunnar@decodable.co)

 [@gunnarmorling](https://twitter.com/gunnarmorling)

The logo consists of two blue shapes: a semi-circle on the left and a chevron-like shape on the right, both pointing towards each other.

decodable







# Debezium

## Correlating Events From Same Transaction

```
{
  "before": null,
  "after": {
    "pk": "2",
    "aa": "1"
  },
  "source": {
    ...
  },
  "op": "c",
  "ts_ms": "1580390884335",
  "transaction": {
    "id": "571:53195832",
    "total_order": "1",
    "data_collection_order": "1"
  }
}
```

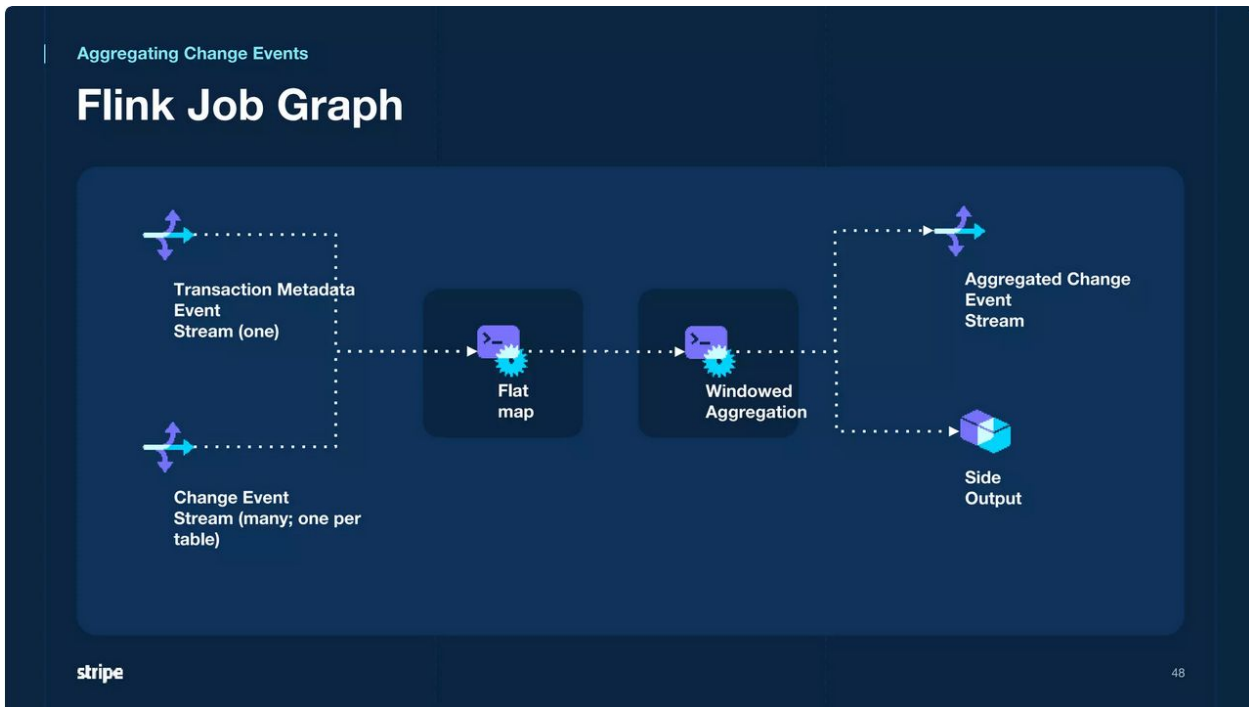
```
{
  "status": "BEGIN",
  "id": "571:53195829",
  "ts_ms": 1486500577125
}

{
  "status": "END",
  "id": "571:53195832",
  "ts_ms": 1486500577691,
  "event_count": 2,
  "data_collections": [
    {
      "data_collection": "s1.a",
      "event_count": 1
    },
    {
      "data_collection": "s2.a",
      "event_count": 1
    }
  ]
}
```



# Debezium

## Correlating Events From Same Transaction



<https://www.slideshare.net/FlinkForward/squirreling-away-640-billion-how-stripe-leverages-flink-for-change-data-capture>