Kafka 101/Event Driven Architecture

Amanda Gilbert

Event Driven Systems

Event Driven Systems are **asynchronous**, allowing **decoupled** services to process events in a scalable, flexible way and to **replay** data when needed.

STATE

Customer’s current email address is jane.doe@gmail.com

Customer’s current balance is $450

There are currently 112 size medium t-shirts in inventory

The patient’s temperature is 98.6 degrees fahrenheit

EVENT

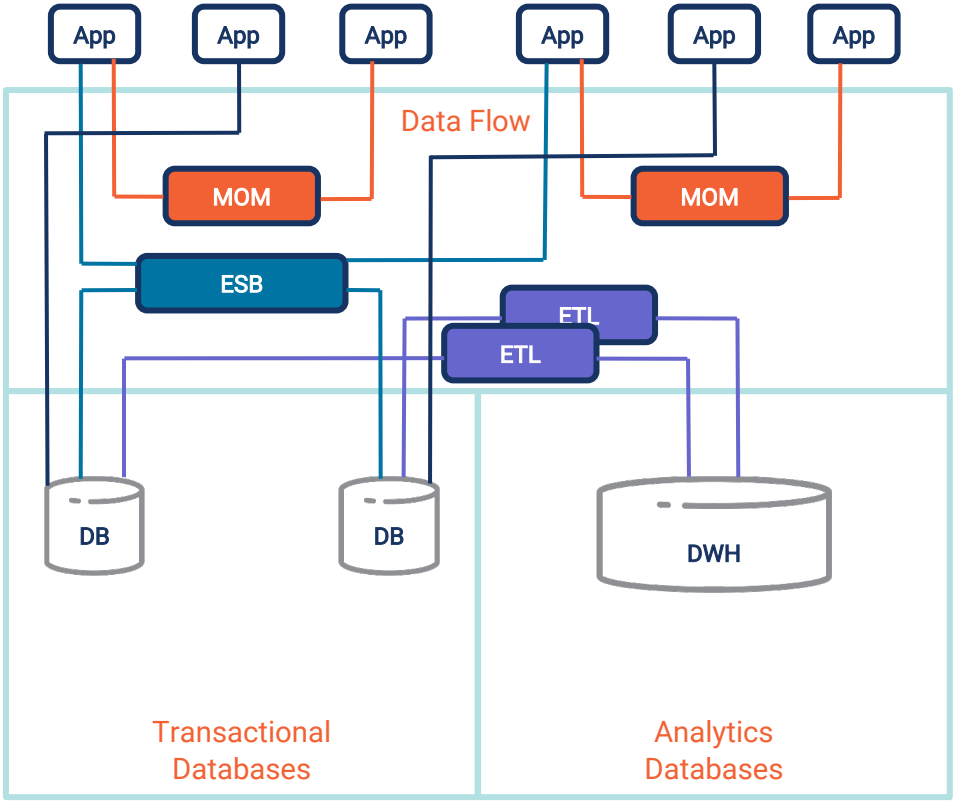
Customer changes email address to jane.doe2@gmail.com

Customer spends $16 dollars at Target.com

One medium size t-shirt is bought by an online customer

The patient's temperature raises by .8 degrees

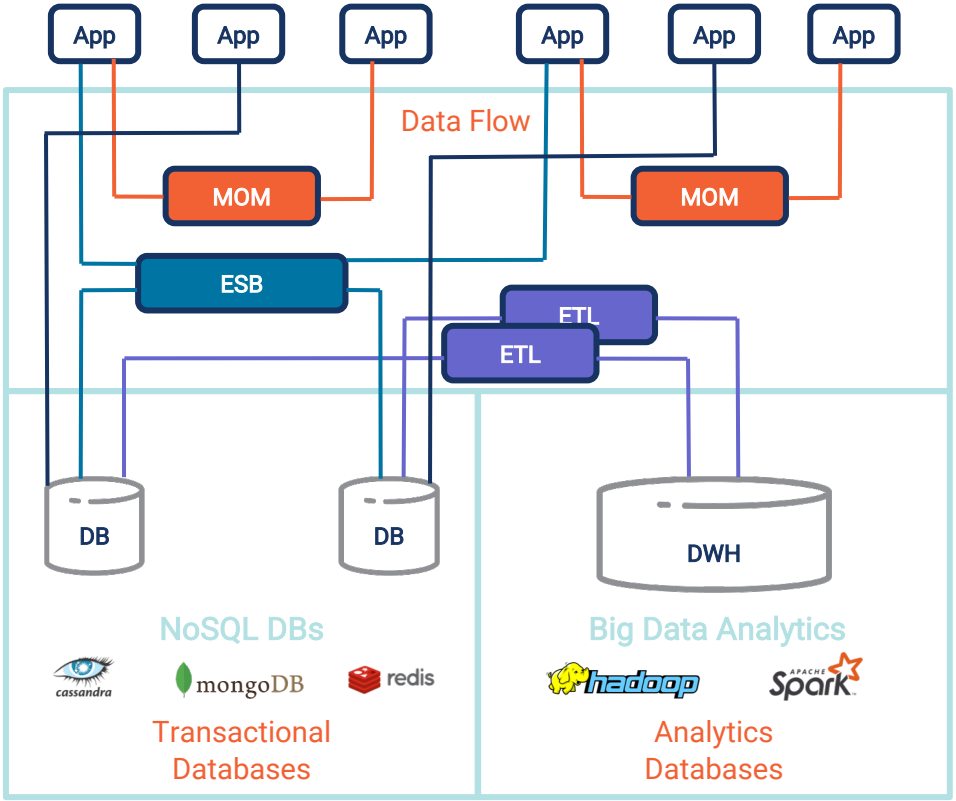
Legacy Data Infrastructure Solutions Have Architectural Flaws



These solutions can be

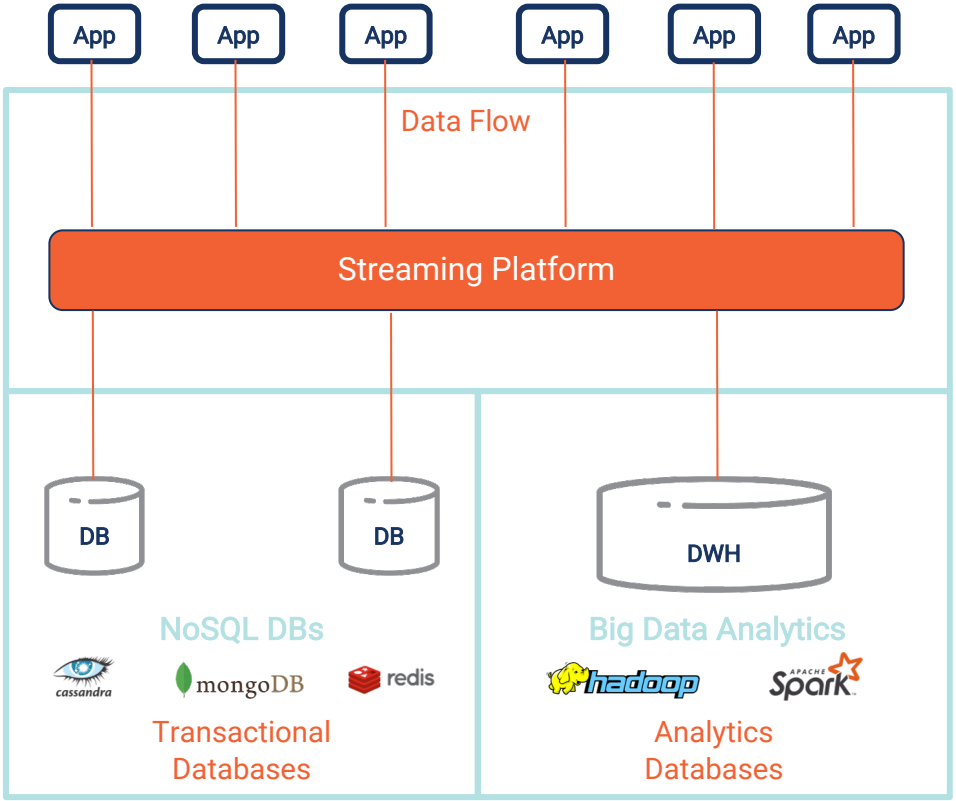
- Batch-oriented, instead of event-oriented in real time
- Complex to scale at high throughput
- Connected point-to-point, instead of publish / subscribe
- Lacking data persistence and retention
- Incapable of in-flight message processing

Modern Architectures are Adapting to New Data Requirements



But how do we revolutionize data flow in a world of exploding, distributed and ever changing data?

The Solution is a Streaming Platform for Real-Time Data Processing



A Streaming Platform provides a single source of truth about your data to everyone in your organization

Event Driven Systems pt 2 - EDA Models

Event Notification

Event-Carried State Transfer

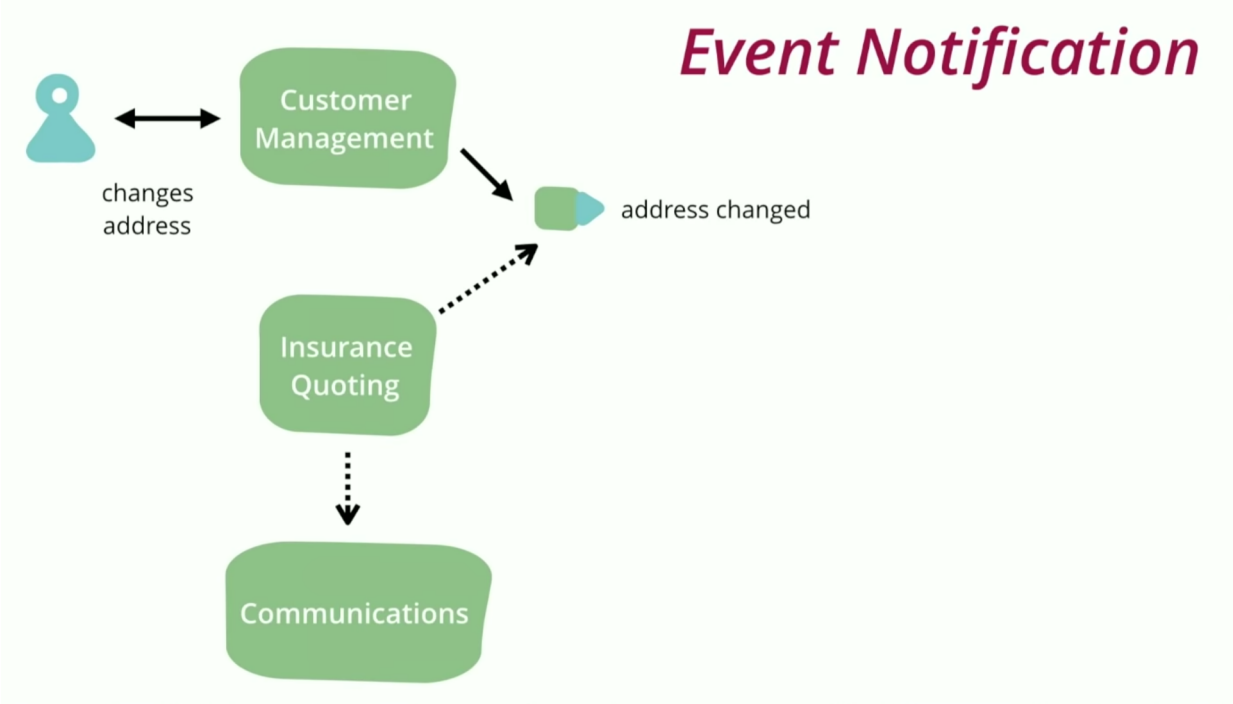
Event Sourcing

CQRS

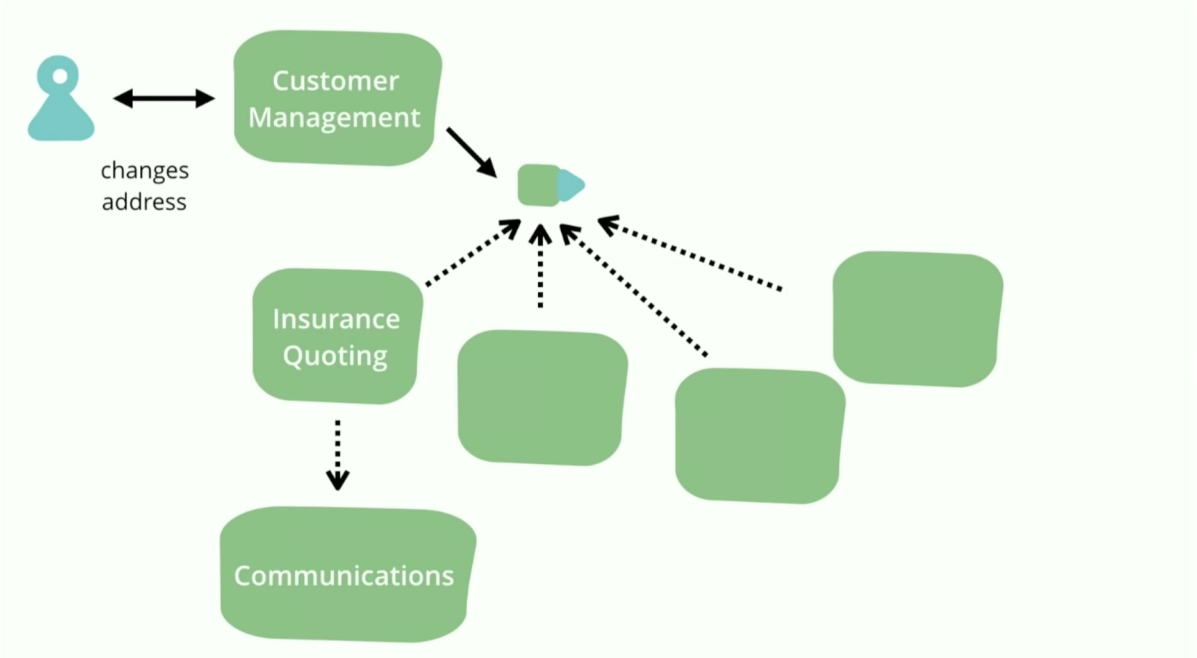
Event Notification

- Systems notify other systems of events that happen
- Loosely coupled systems, “Fire and forget”
- Usually just metadata transferred

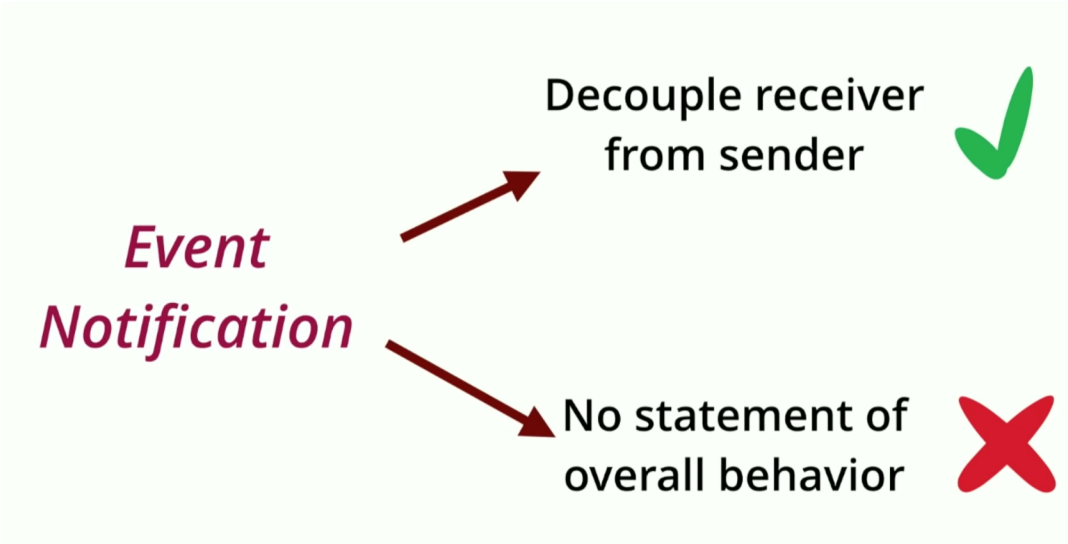
Use Case: Change customer address resolving dependencies



Use Case: Change customer address, Events or Commands



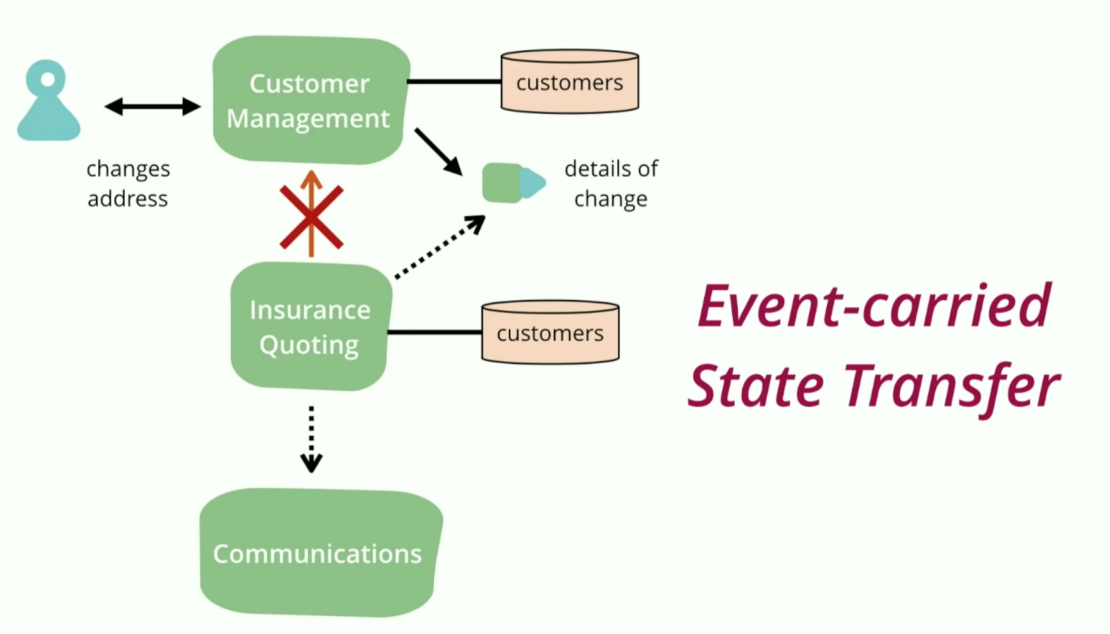
Notification Events



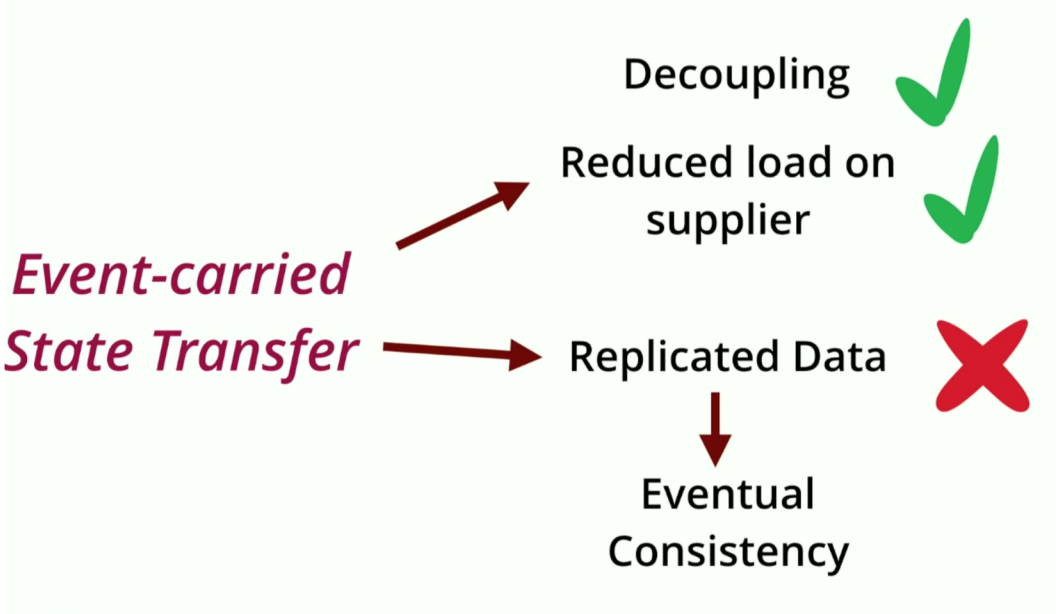
Event-Carried State Transfer

- Upon a state change, data is transferred to other systems through events
- Greater decoupling as systems are not reliant on each other
- Lower latency, lower load on underlying systems

Event-carried State Transfer



Event-carried State Transfer



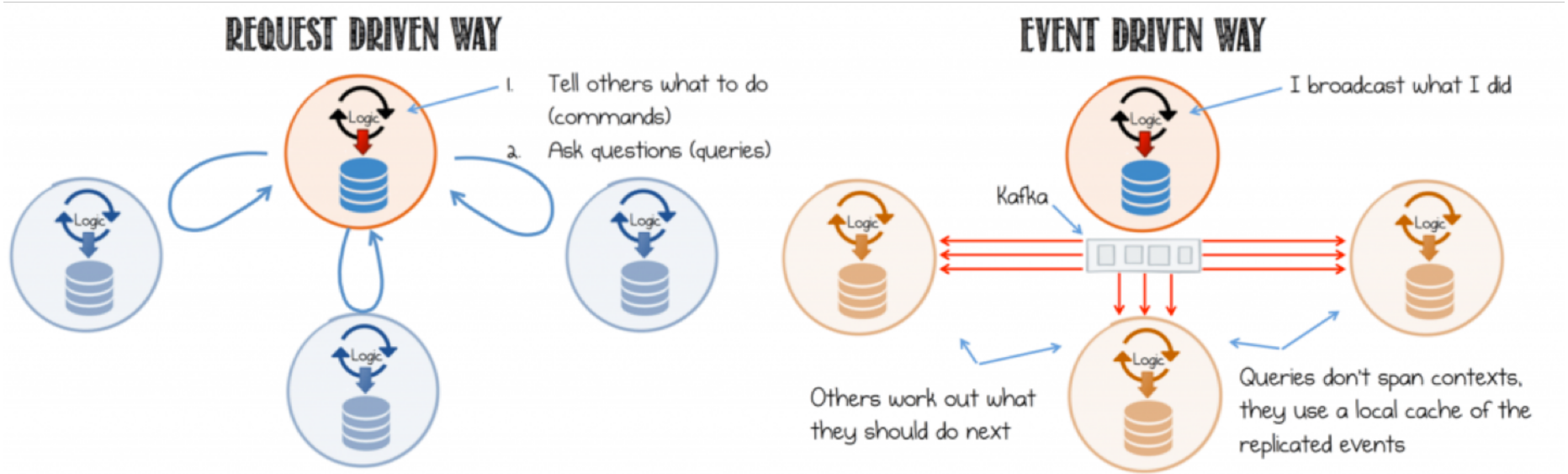
Event Sourcing

- Whenever a change is made it is logged as an event
- System state can be rebuilt at any time from the event layer
- Large amounts of storage required at the kafka layer, doesn't work for use cases where a MV is required

CQRS

- Separate read and write
- Every method is either a command or a query
- Kafka can be used as a middle layer (write to kafka, pull the data out and write to a db to be read)

A visual summary of commands, events, and queries



Introduction to Kafka

Apacke Kafka®: Open Source Streaming Platform Battle-Tested at Scale

LinkedIn

The birthplace of Apache Kafka

More than 1 petabyte of data in Kafka

Over 4.5 trillion messages per day

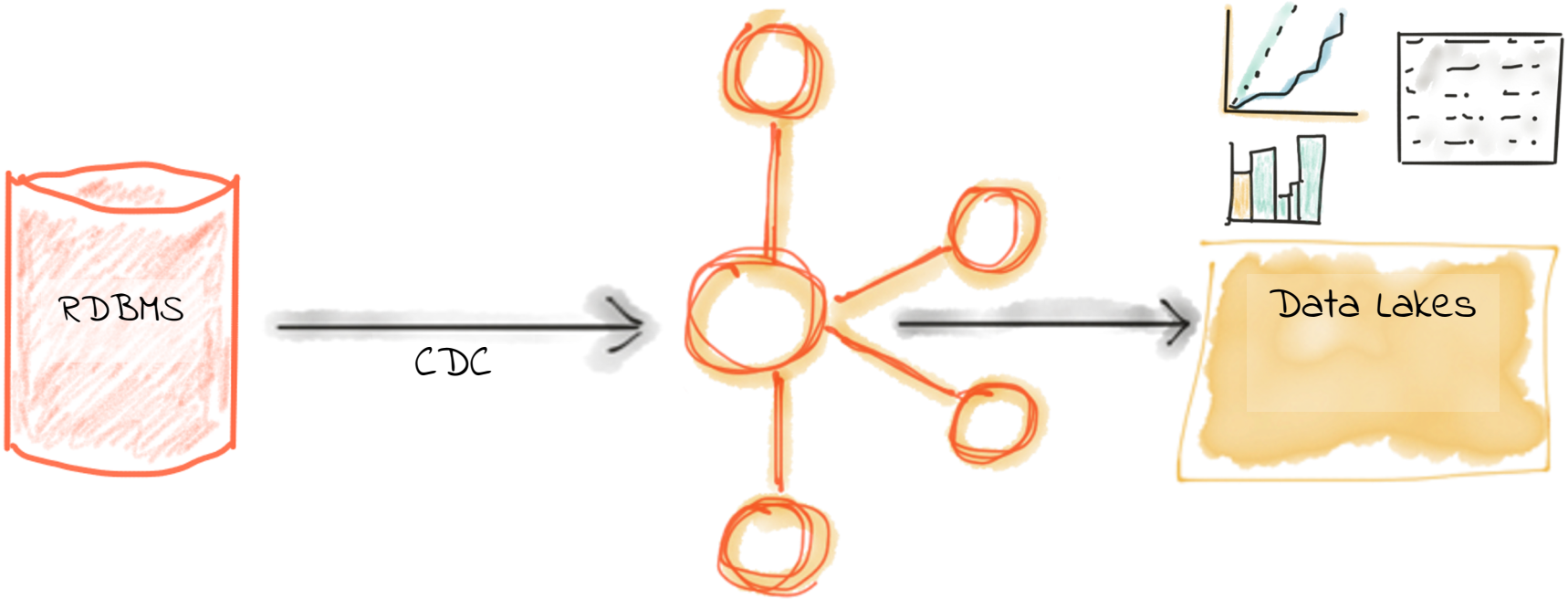
60,000+ data streams

Source of all data warehouse & Hadoop data

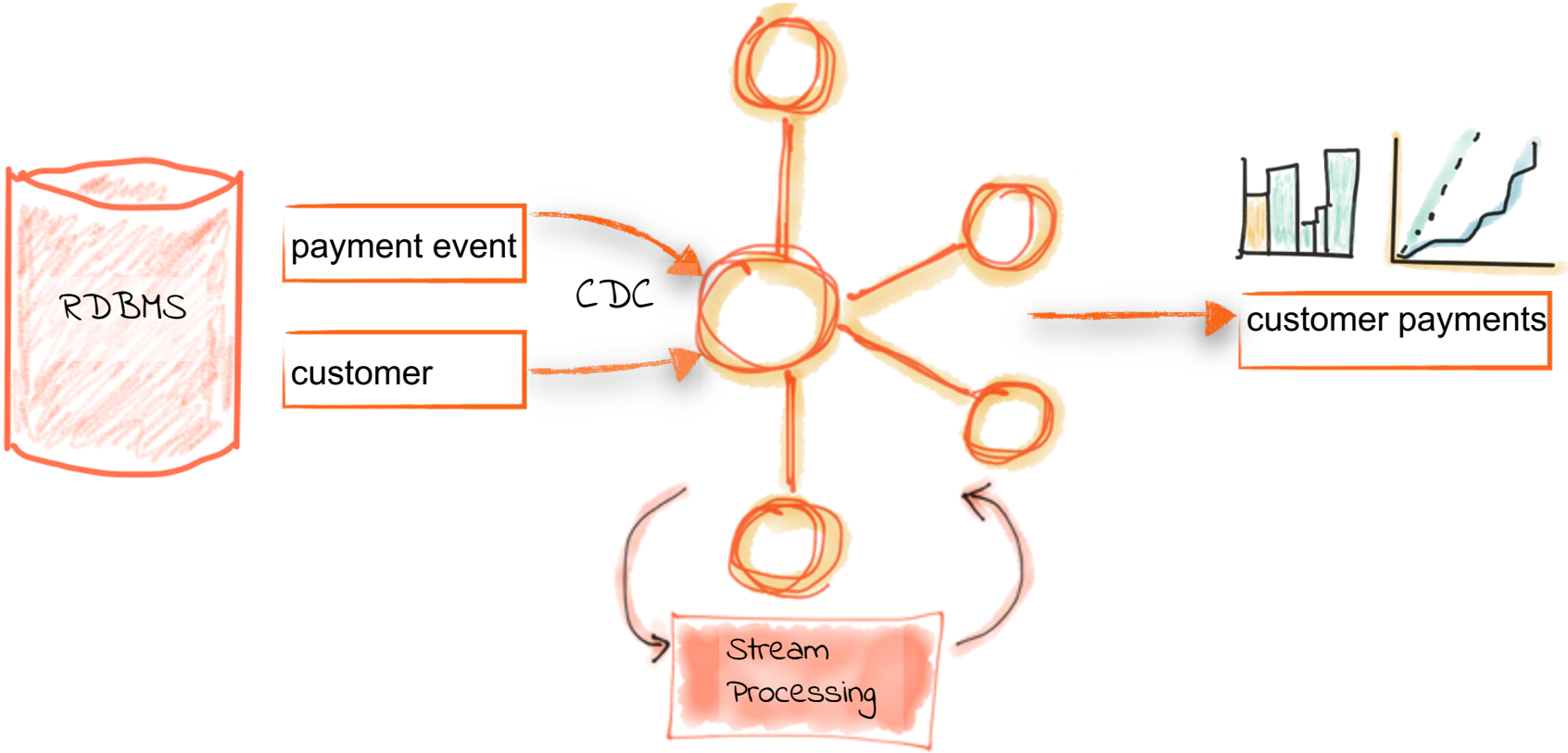
Over 300 billion user-related events per day

Architectural Patterns with Apache Kafka

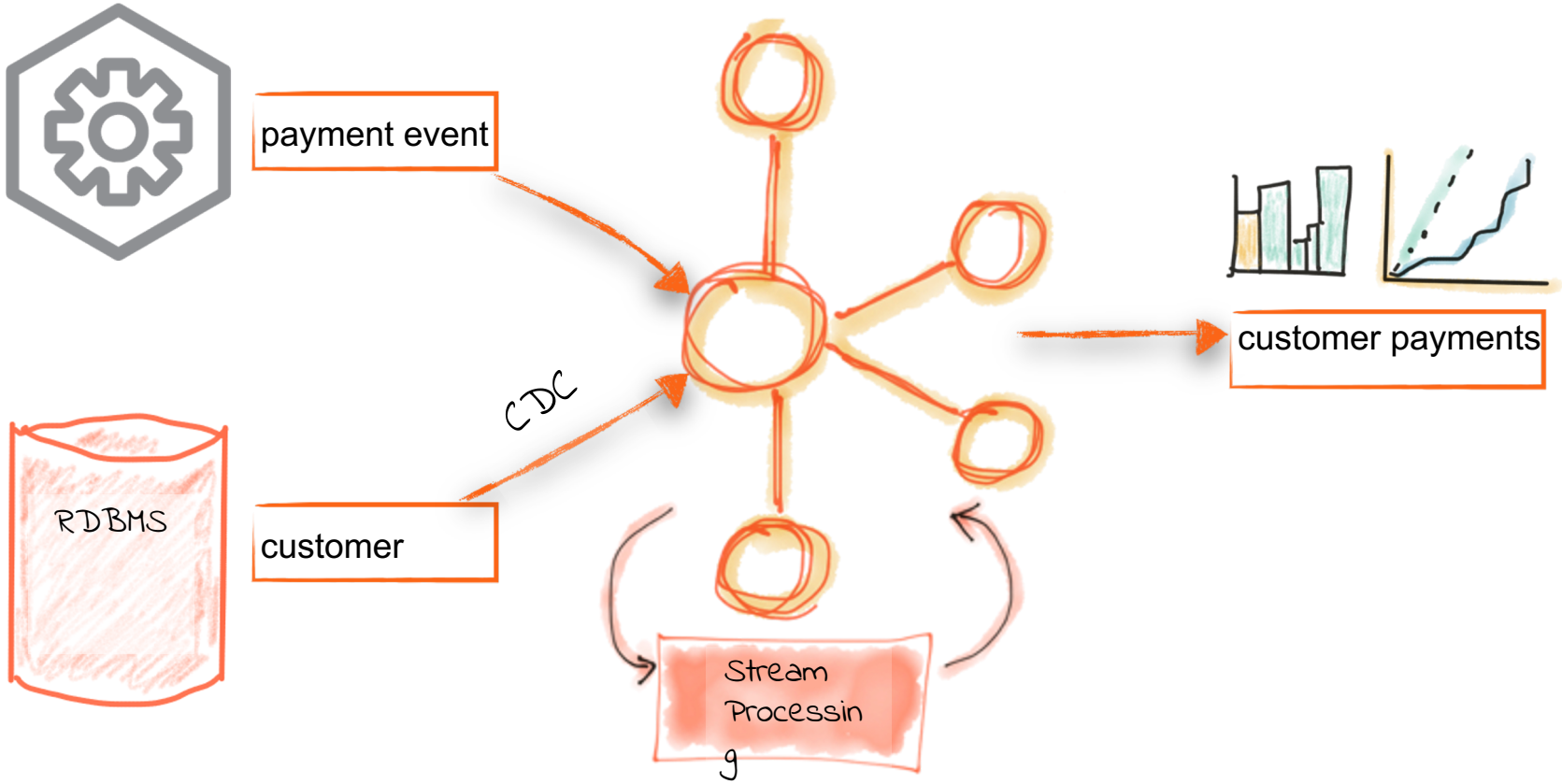
Analytics - Database Offload

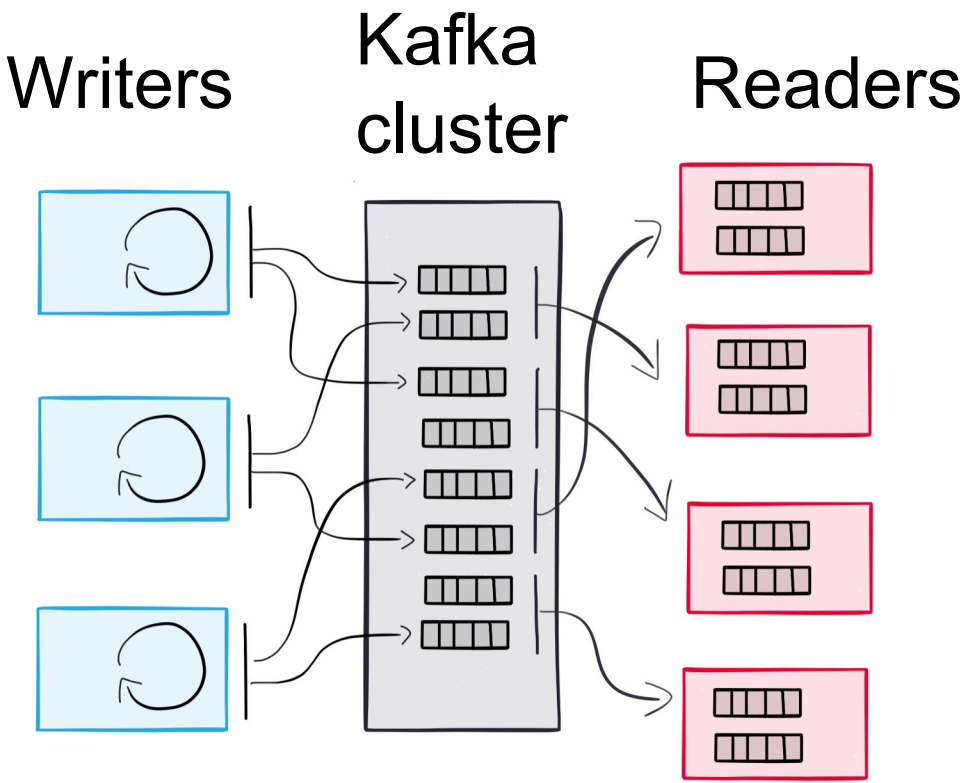


Stream Processing with Apache Kafka and KSQL



Transform Once, Use Many



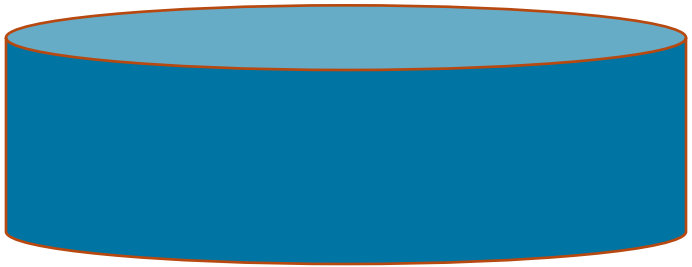

Writers
Kafka cluster
Readers

Scalability of a filesystem

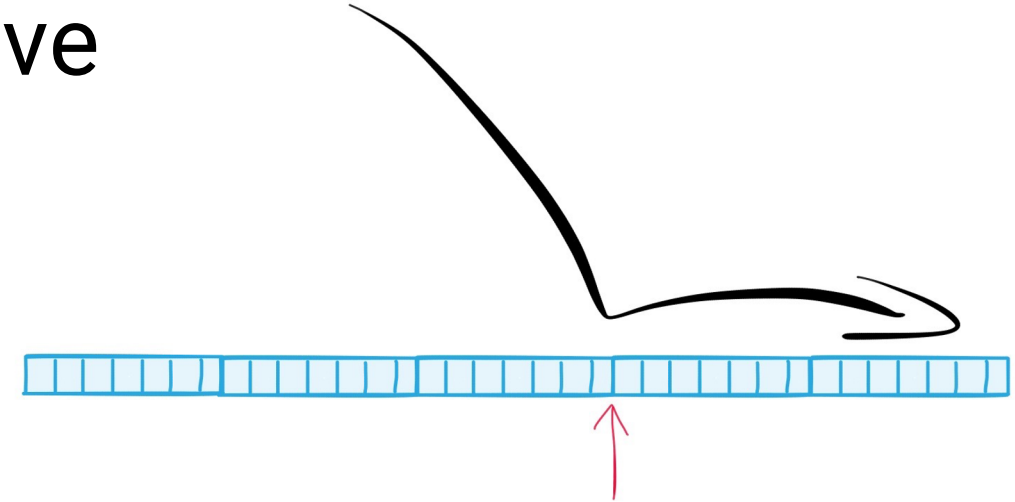
- hundreds of MB/s throughput
- many TB per server
- commodity hardware

Guarantees of a Database

- Strict ordering
- Persistence



Reset to any point in the shared narrative

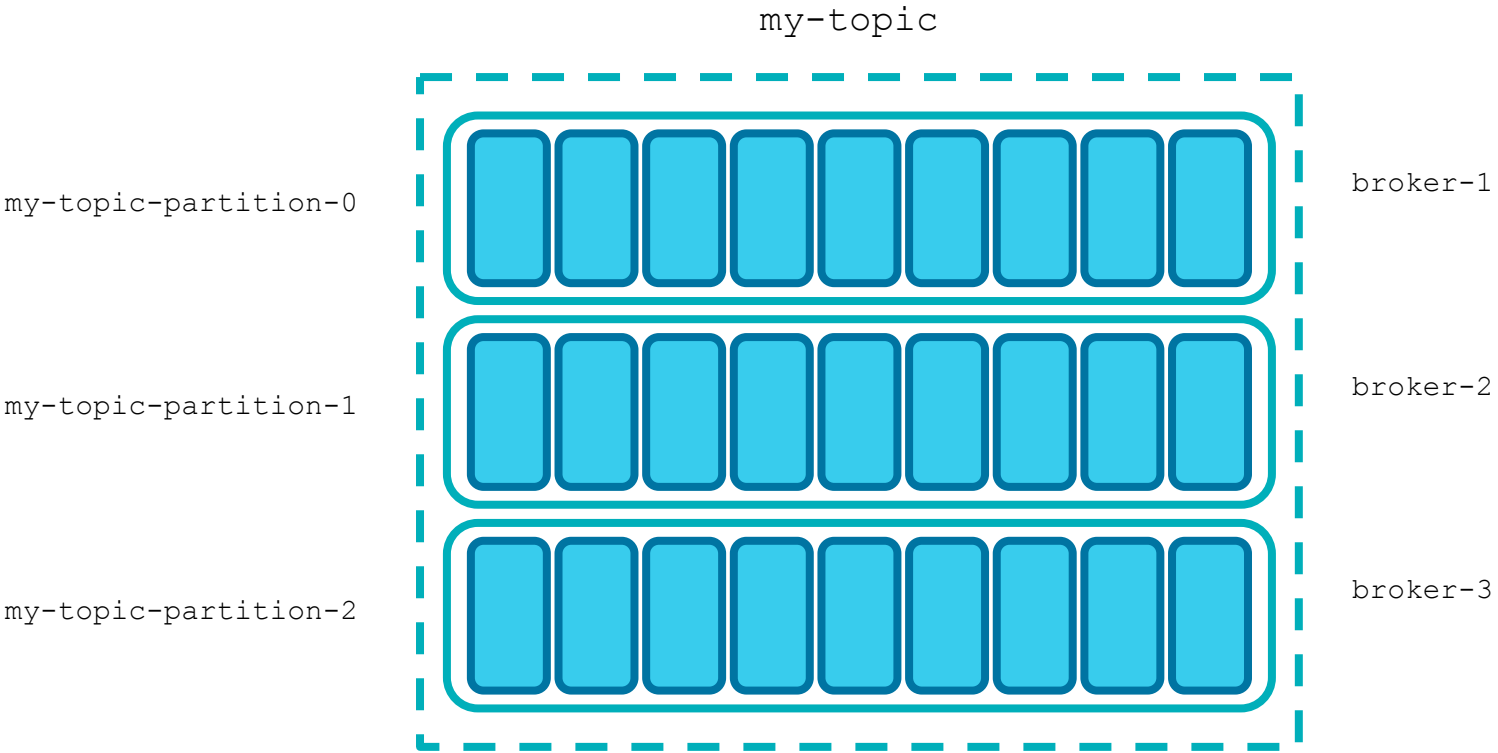


Rewind & Replay

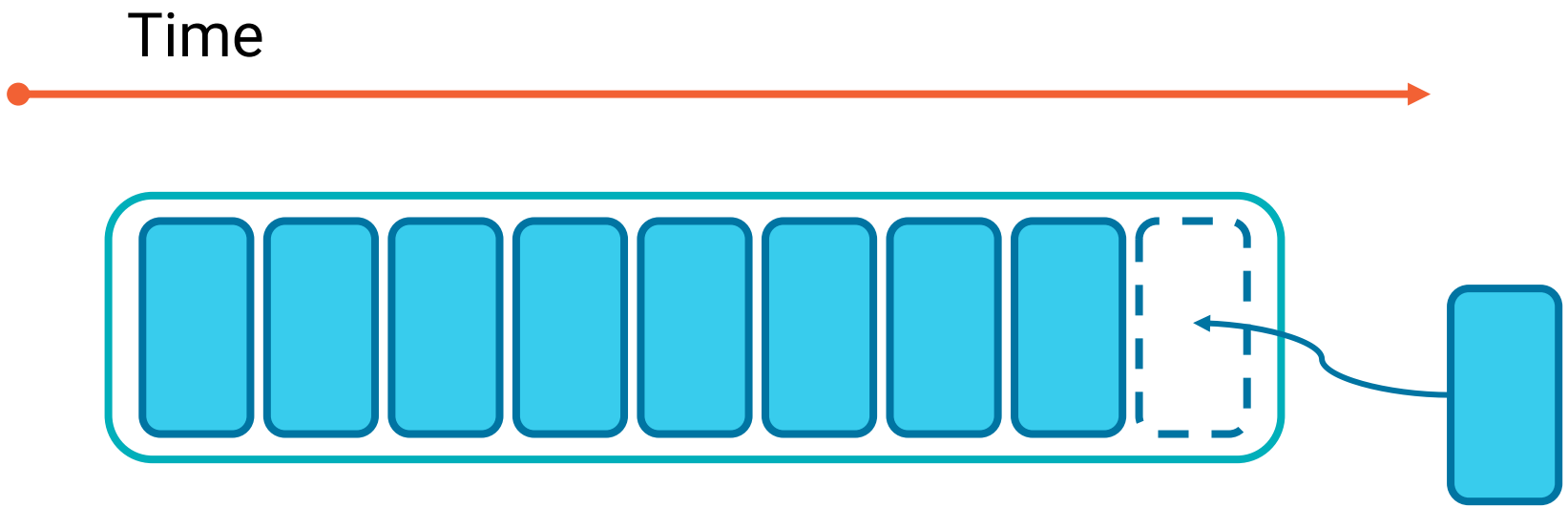
Distributed by design

- Replication
- Fault Tolerance
- Partitioning
- Elastic Scaling

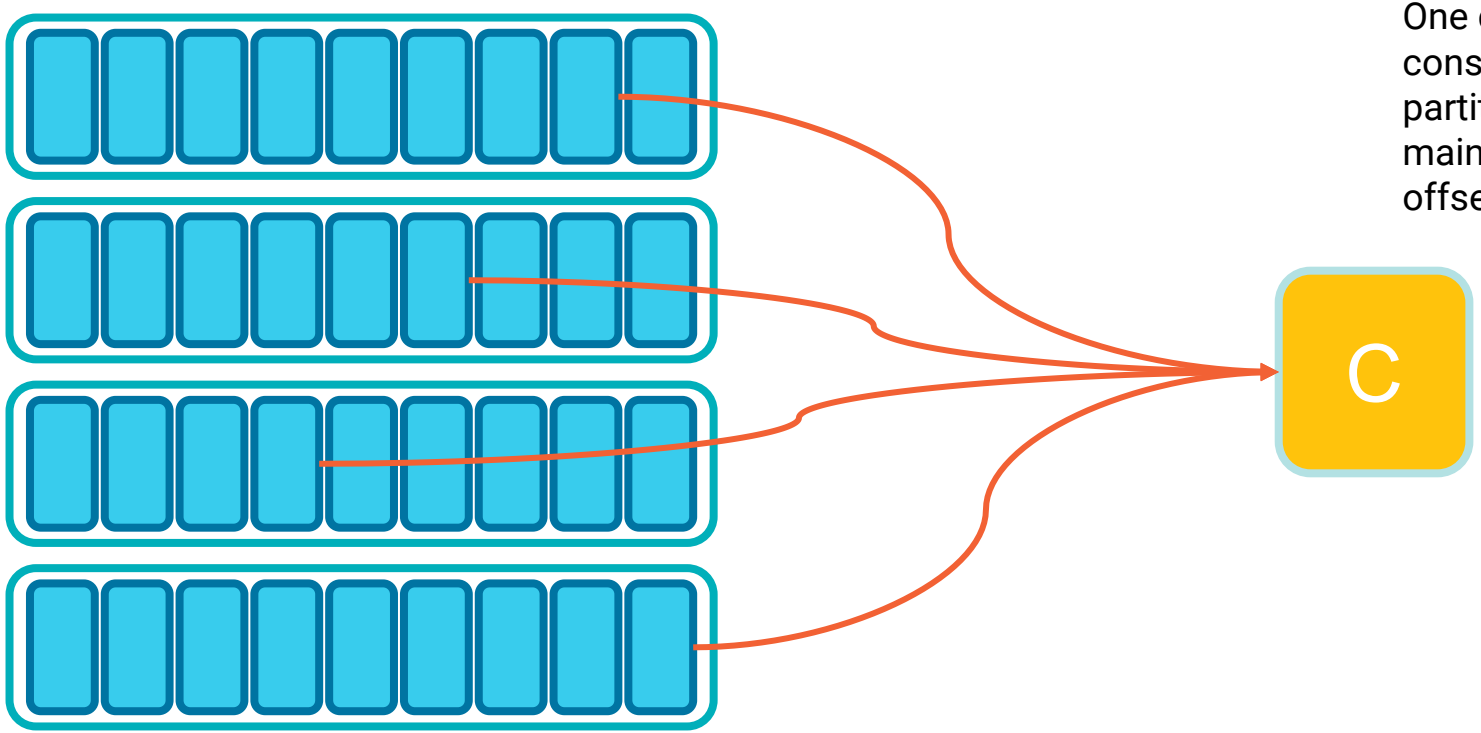
Kafka Topics



Producing to Kafka

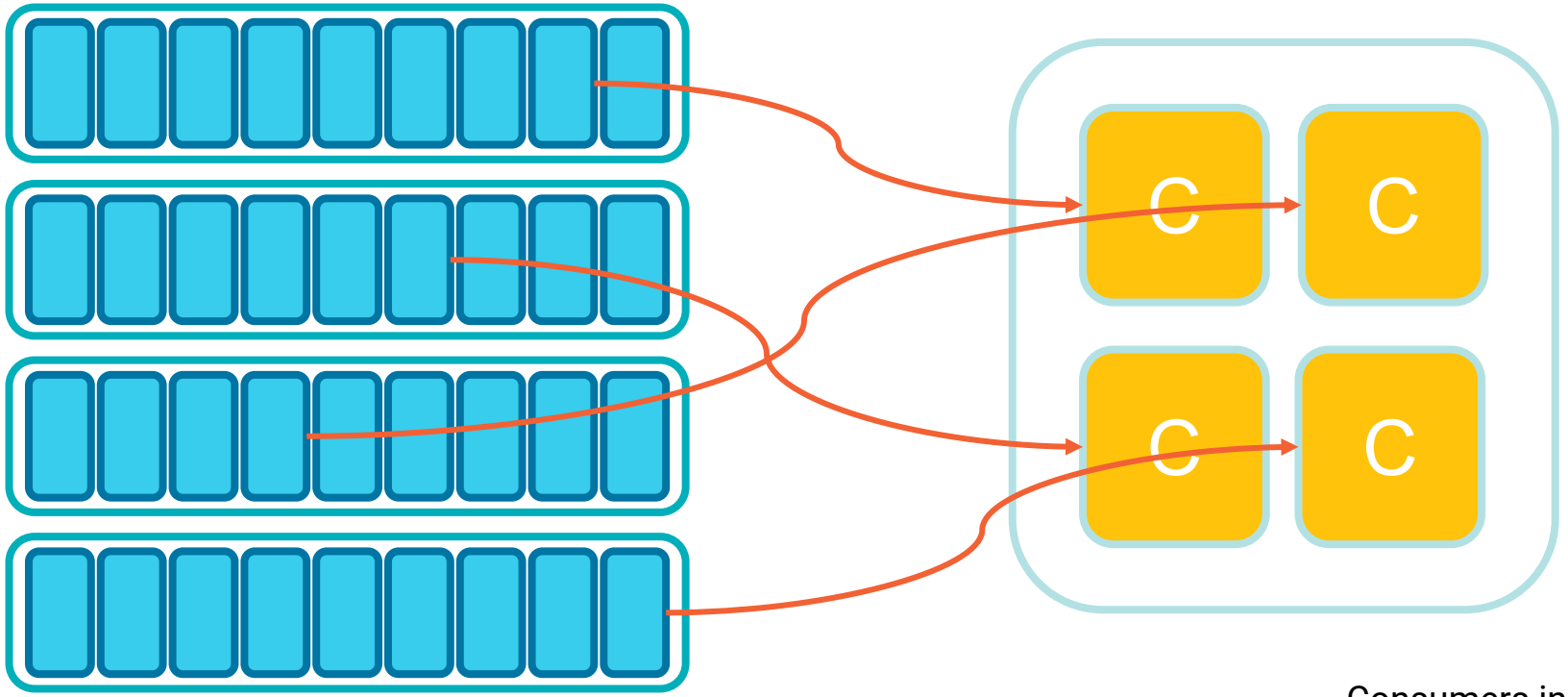


Consuming From Kafka - Single Consumer



One consumer will consume from all partitions, maintaining partition offsets

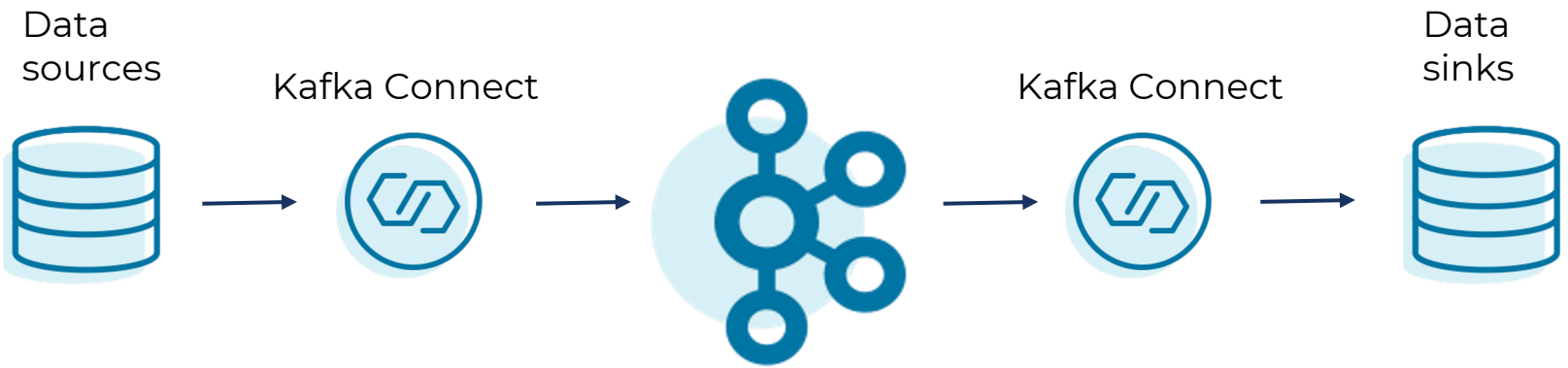
Consuming From Kafka - Grouped Consumers



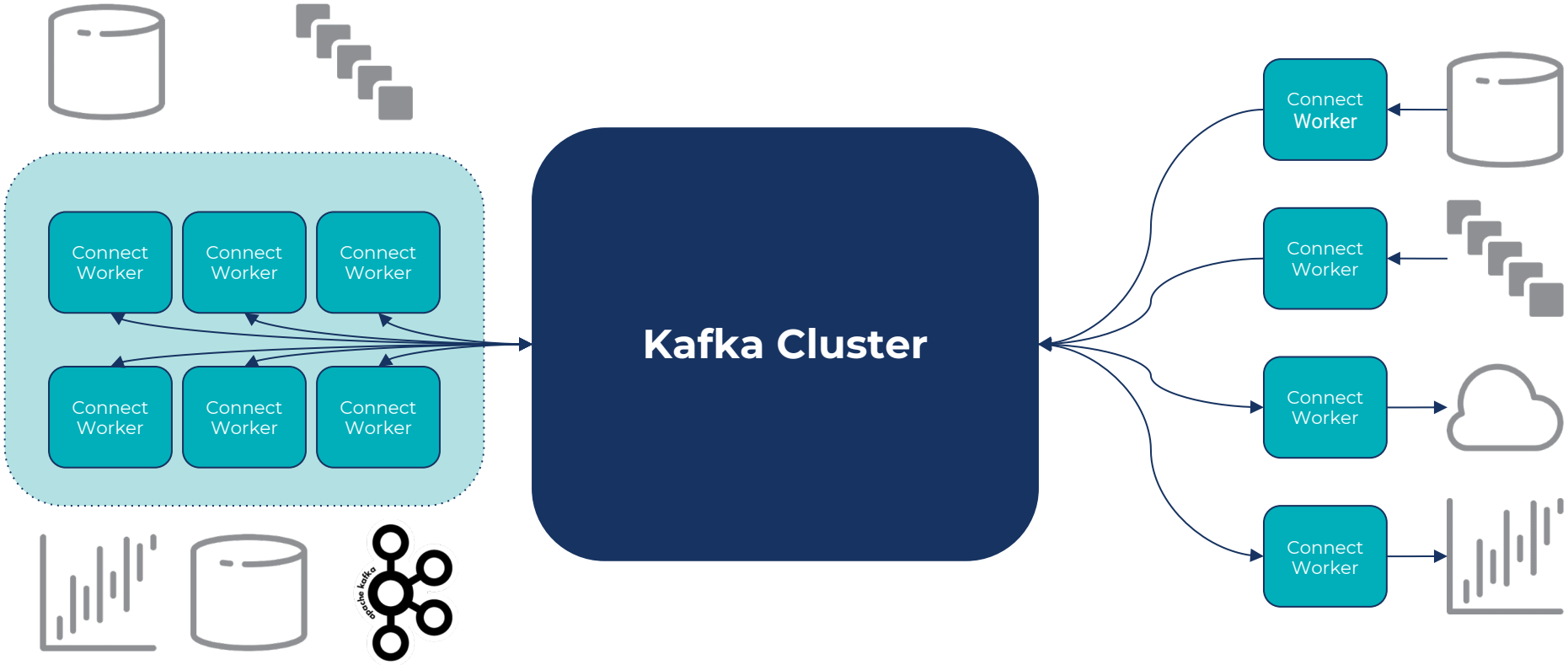
Consumers in a consumer group share the workload

Kafka Connect

No-Code way of connecting known systems (databases, object storage, queues, etc) to Apache Kafka



Workers: The JVM(s) that run **connectors** and **tasks**. Can be run in either **standalone** or **distributed** mode.



When to use Kafka's Streams API

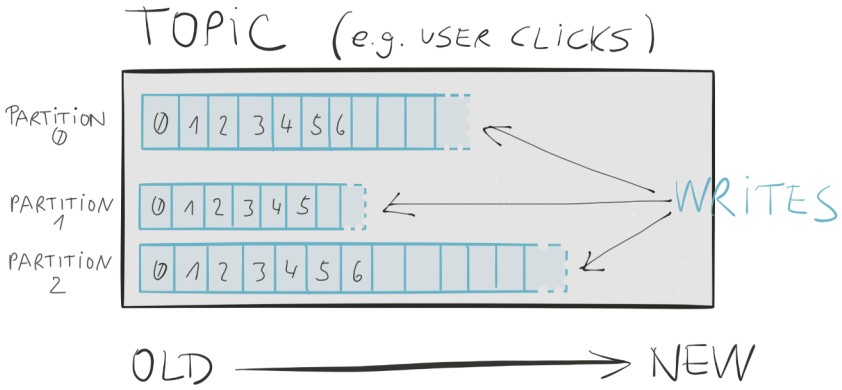
- **Mainstream Application Development**
- **To build core business applications**
- Microservices
- Fast Data apps for small and big data
- Reactive applications
- Continuous queries and transformations
- Event-triggered processes
- The "T" in ETL
- <and more>

Use case examples

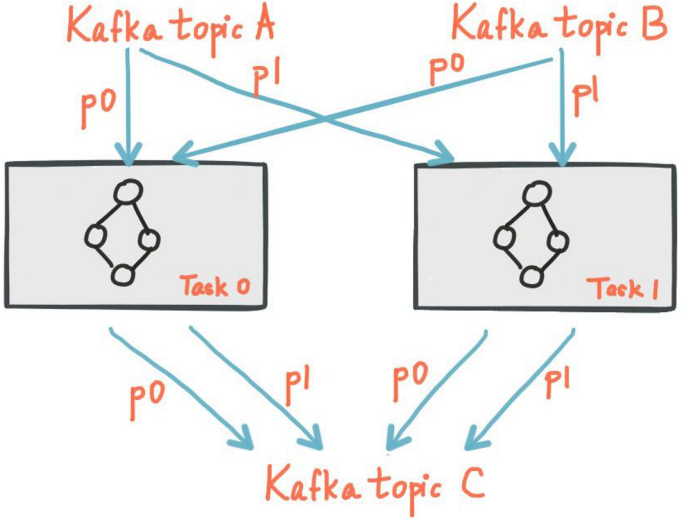
- Real-time monitoring and intelligence
- Customer 360-degree view
- Fraud detection
- Location-based marketing
- Fleet management
- <and more>

Key concepts

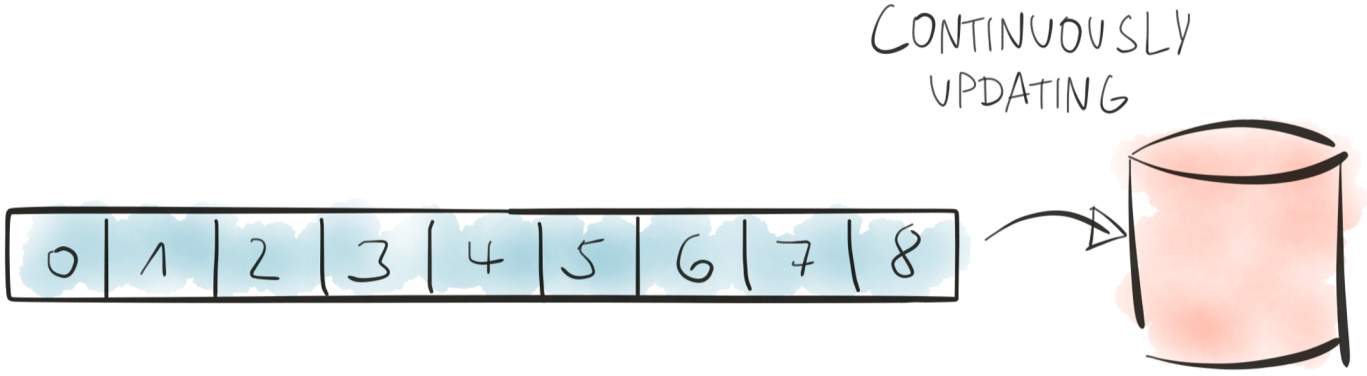
Kafka Core



Kafka Streams

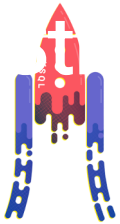


STREAM → TABLE



EXAMPLE:
CUSTOMER 360°

MATERIALIZED
VIEW



KSQL

```
CREATE STREAM fraudulent_payments AS
 SELECT * FROM payments
 WHERE fraudProbability > 0.8;
```

VS.

kafka streams

```
object FraudFilteringApplication extends App {

  val config = new java.util.Properties
  config.put(StreamsConfig.APPLICATION_ID_CONFIG, "fraud-filtering
  config.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker

  val builder: StreamsBuilder = new StreamsBuilder()
  val fraudulentPayments: KStream[String, Payment] = builder
    .stream[String, Payment]("payments-kafka-topic")
    .filter((_ ,payment) => payment.fraudProbability > 0.8)

  val streams: KafkaStreams = new KafkaStreams(builder.build(), co
  streams.start()
}
```

KSQL example use cases

Data exploration

Data enrichment

Streaming ETL

Filter, cleanse, mask

Real-time monitoring

Anomaly detection

Who should manage your Kafka cluster?

Self Managed

- Granular configuration control
- Higher security control
- High DevOps burden

Providers

- OS Kafka
- Confluent Community / Platform

Fully Managed

- Faster time to live
- Lower operational burden
- Maximize uptime
- Lower operational control

Providers

- MSK
- Confluent Cloud
- Heroku


CONFLUENT

Resources

I Heart Logs: https://www.confluent.io/ebook/i-heart-logs-event-data-stream-processing-and-data-integration/

Designing Event Driven Systems: https://www.confluent.io/designing-event-driven-systems/

Kafka: The Definitive Guide: https://www.confluent.io/resources/kafka-the-definitive-guide/

Confluent Blog: https://www.confluent.io/blog

Reference Architecture: https://www.confluent.io/resources/apache-kafka-confluent-enterprise-reference-architecture/

Kafka Quickstart: https://kafka.apache.org/quickstart

Confluent Cloud Quickstart: https://docs.confluent.io/curre ud- tml

Thank You!

agilbert@confluent.io