

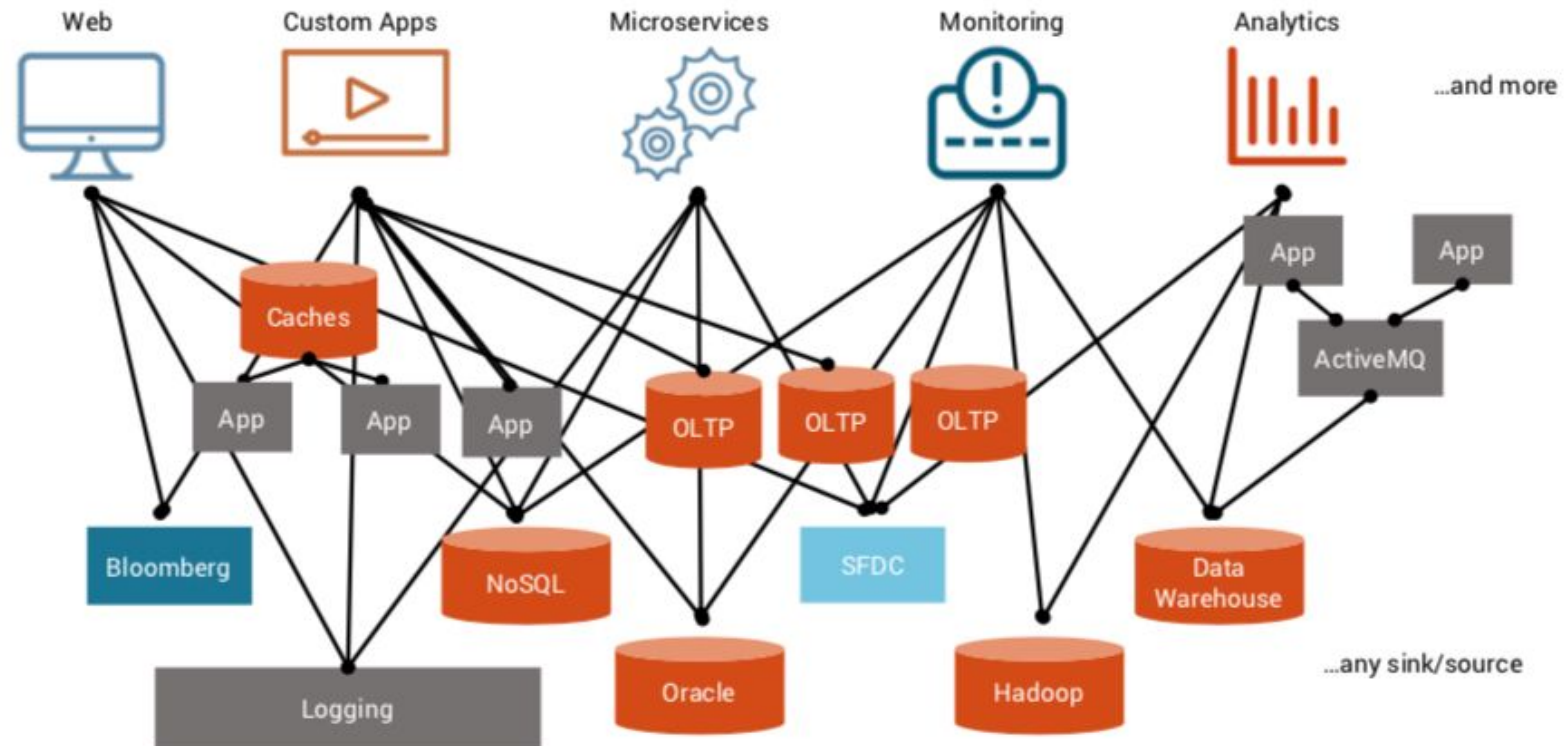
# Intro to Kafka

Ankush

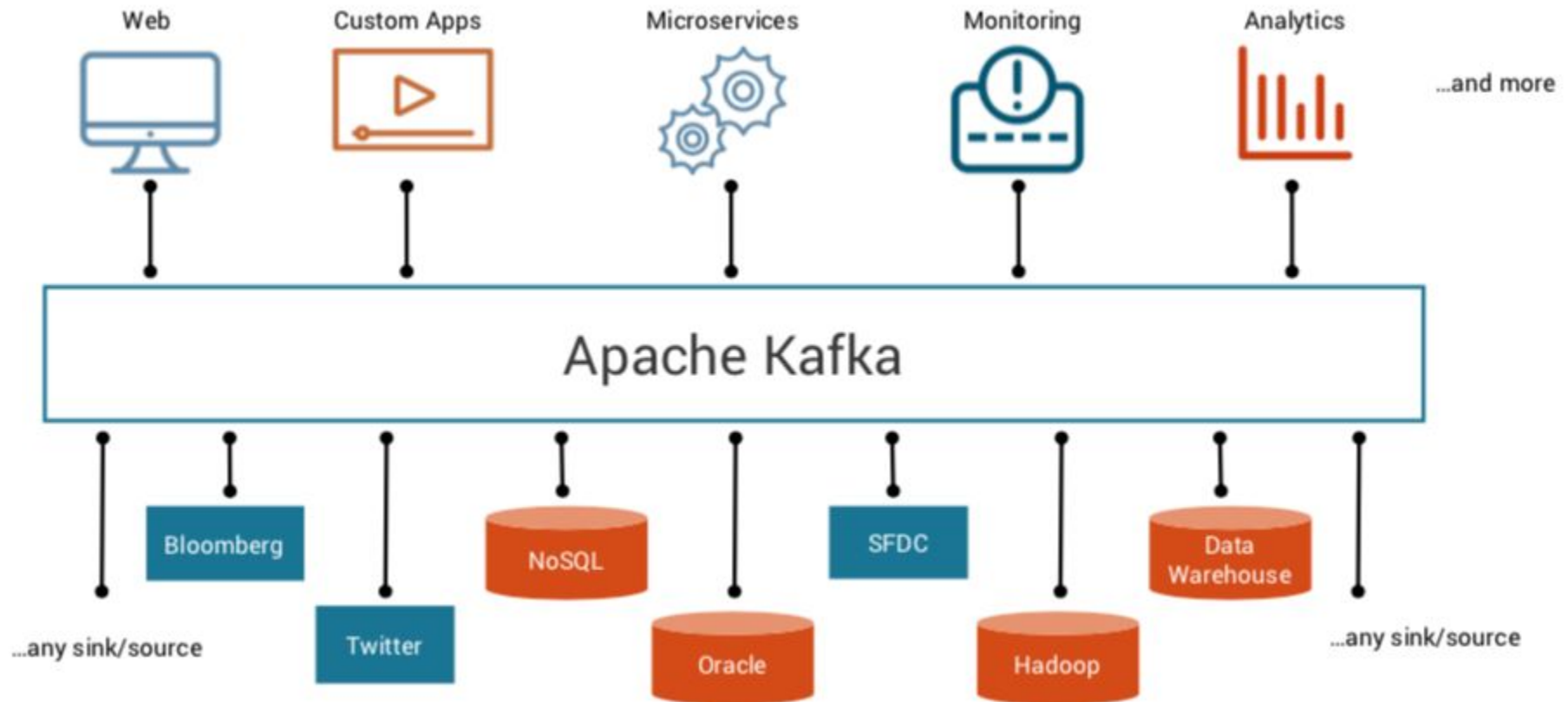
# Index

- What is kafka
- Basic components
- Avro and Schema Registry
- Kafka Connect
- Kafka Streams
- What questions to ask?

# Architecture w/o kafka?



# Architecture with Kafka?



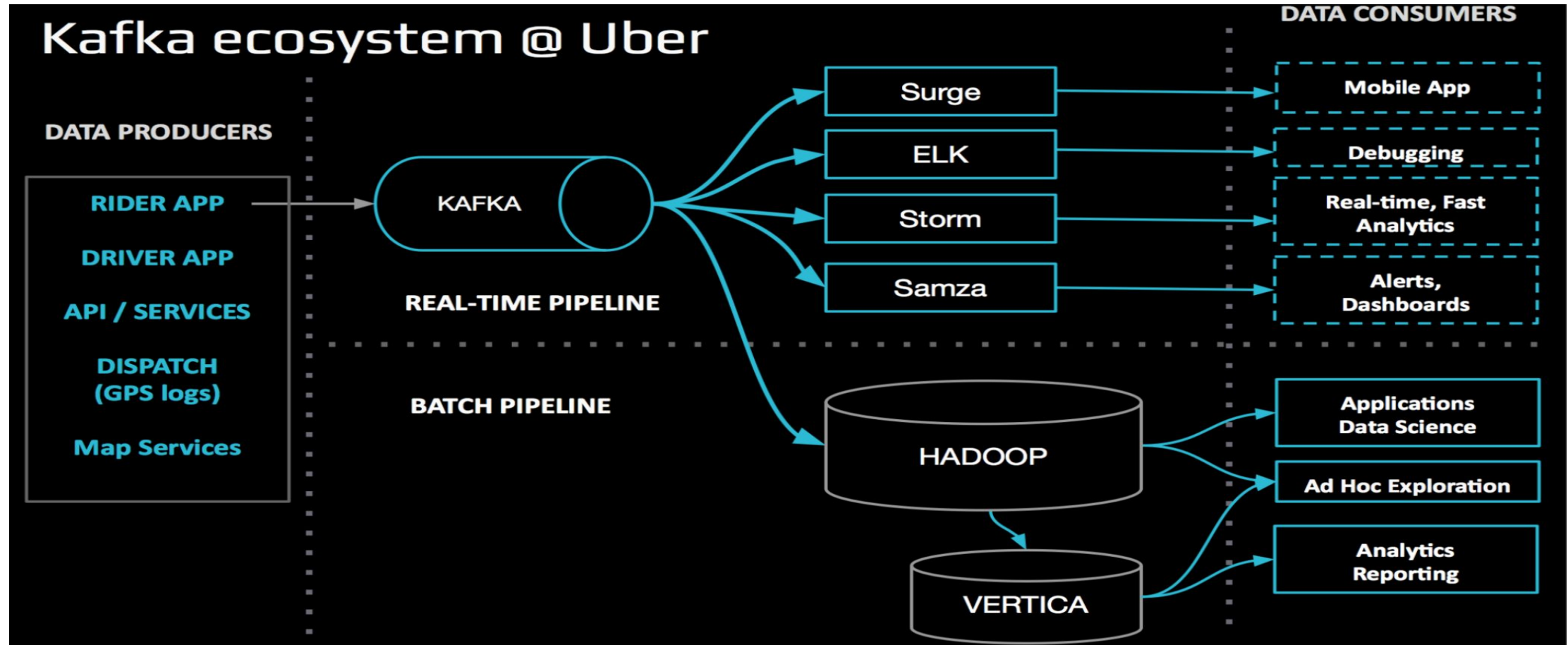
# Kafka everywhere

 DiDi





# Kafka everywhere



# Basic terms

# Messages in kafka

- Kafka Message
  - Key
  - Value
  - Timestamp



# Topic in Kafka

- Producer pushes messages to a topic
- Consumer consumes messages from a topic

# Kafka Broker

- Kafka broker => Physical machine on which Kafka is running
- Kafka Cluster => Multiple Kafka brokers => Multiple machines working together

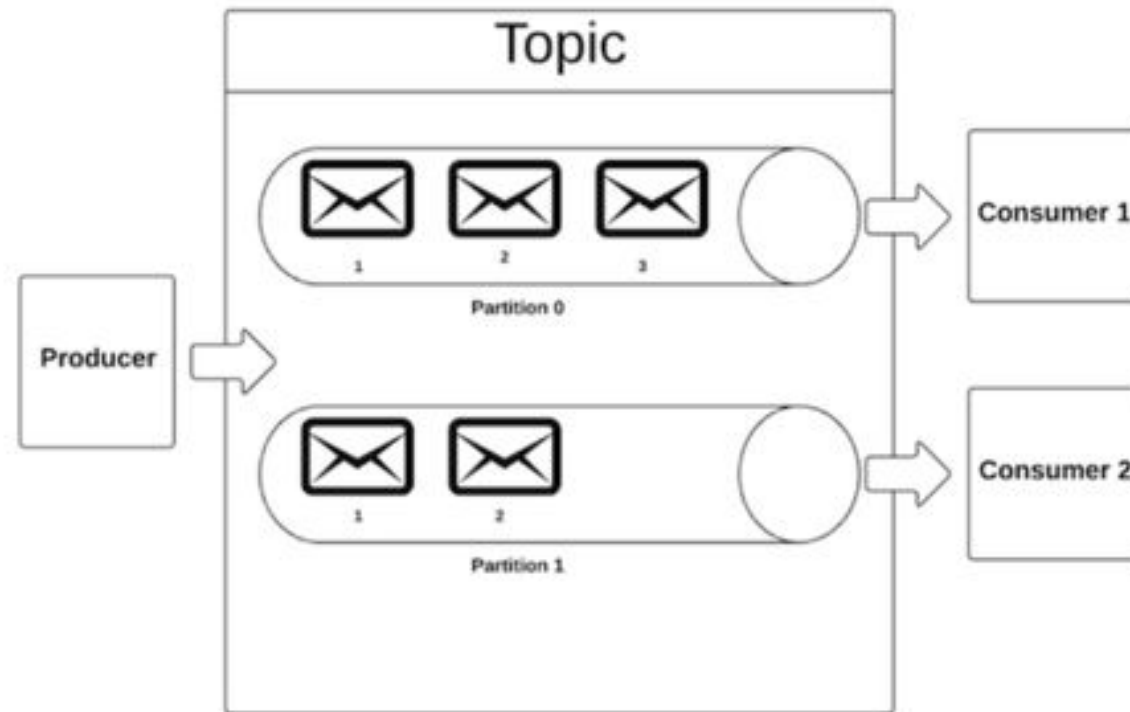
# Logs

- Data segments present in your disk
- Stores messages in a order fashion
  - Assigns sequence id to each message before storing in logs

# Partitioning in Kafka

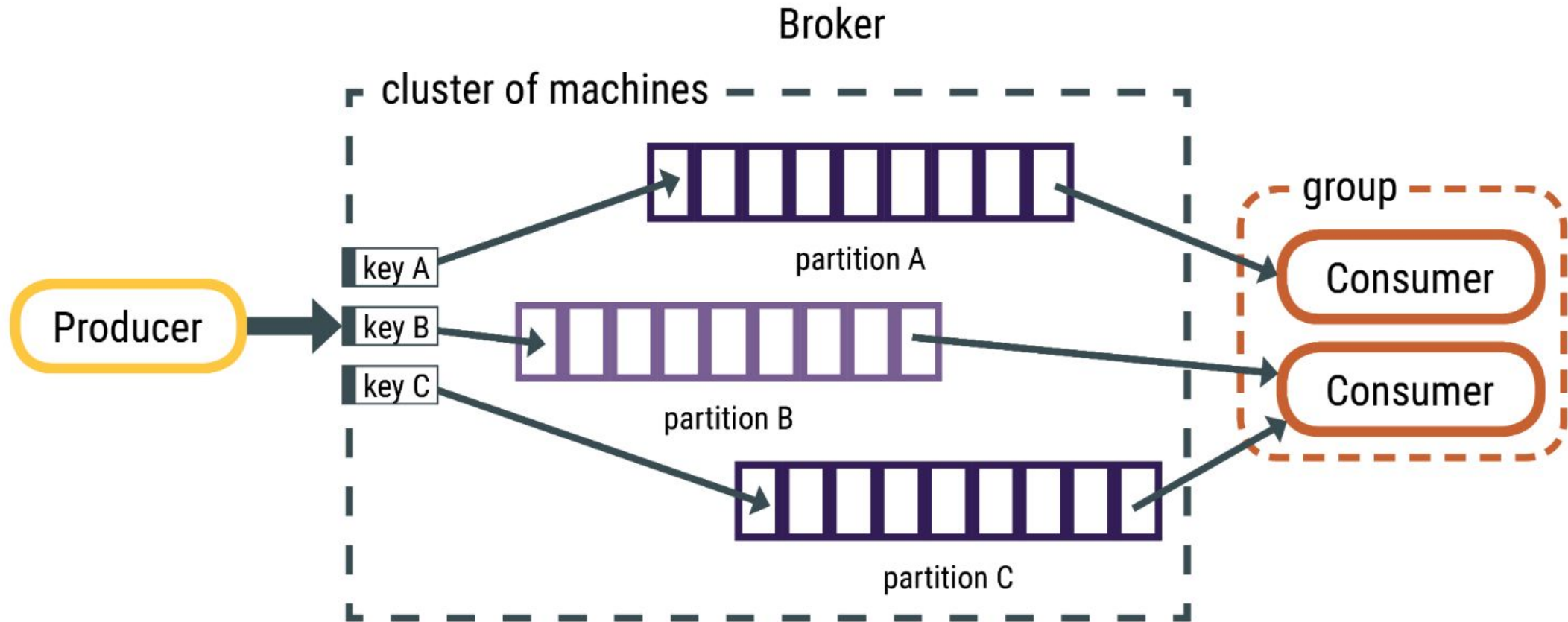
Scalability

# Topics and Partitions in Apache Kafka

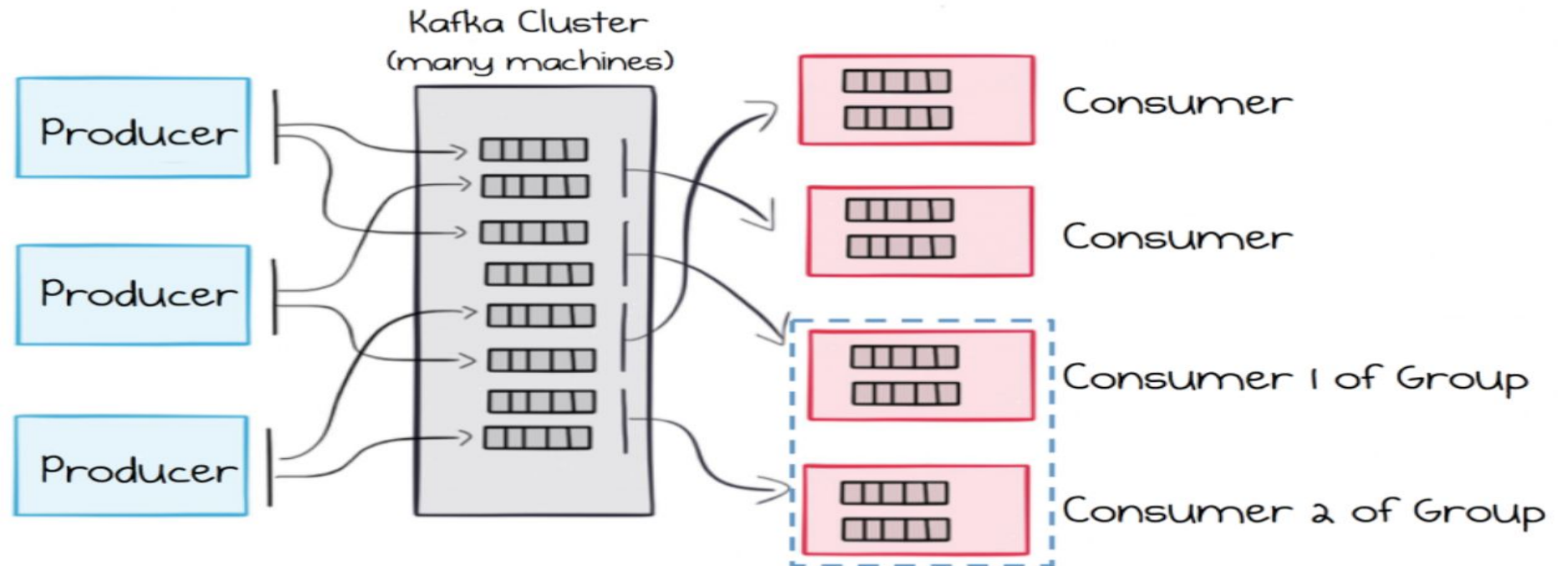


- **topic partitions are a unit of parallelism**
- **a partition can only be worked on by one consumer in a consumer group at a time**

# Partitions in Apache Kafka

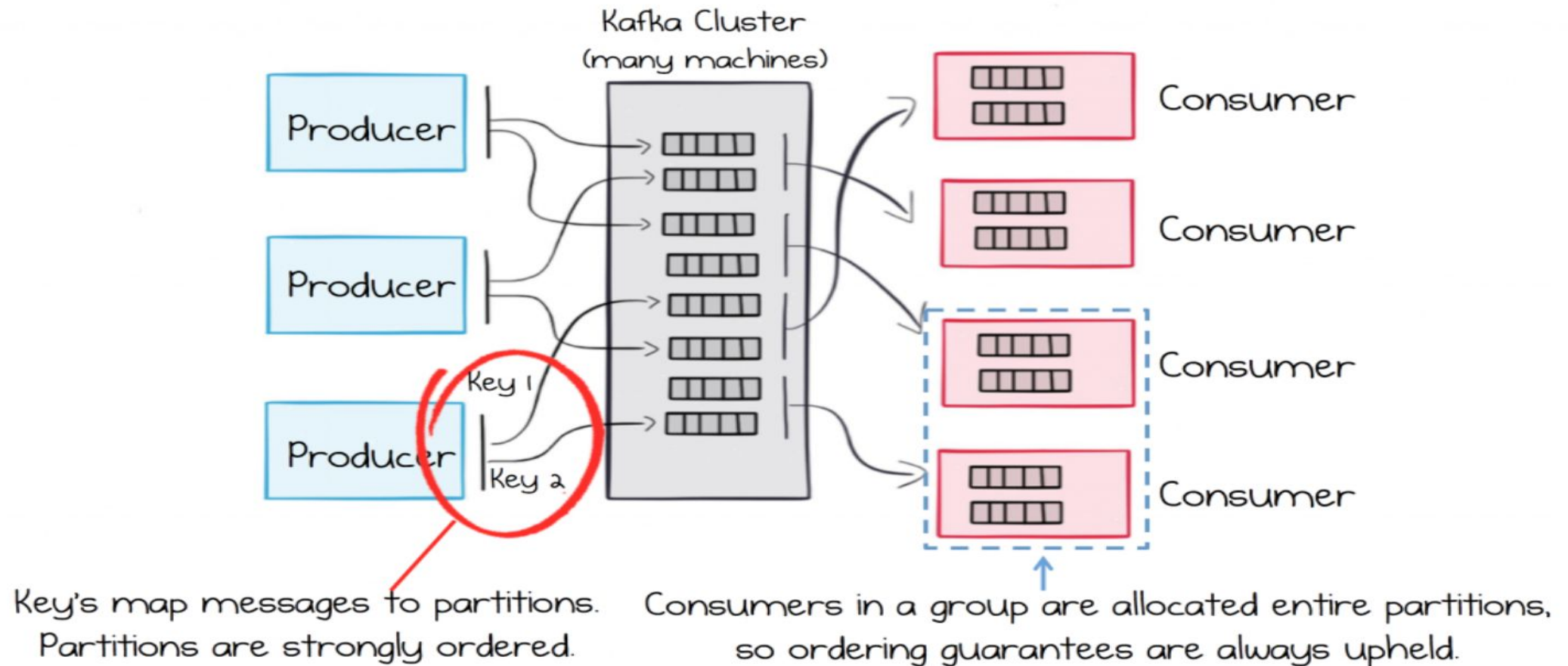


# Partitions in Apache Kafka



Producers spread messages over many partitions, on many machines, where each partition is a little queue. Load balanced consumers (denoted a Consumer Group) share the partitions between them.

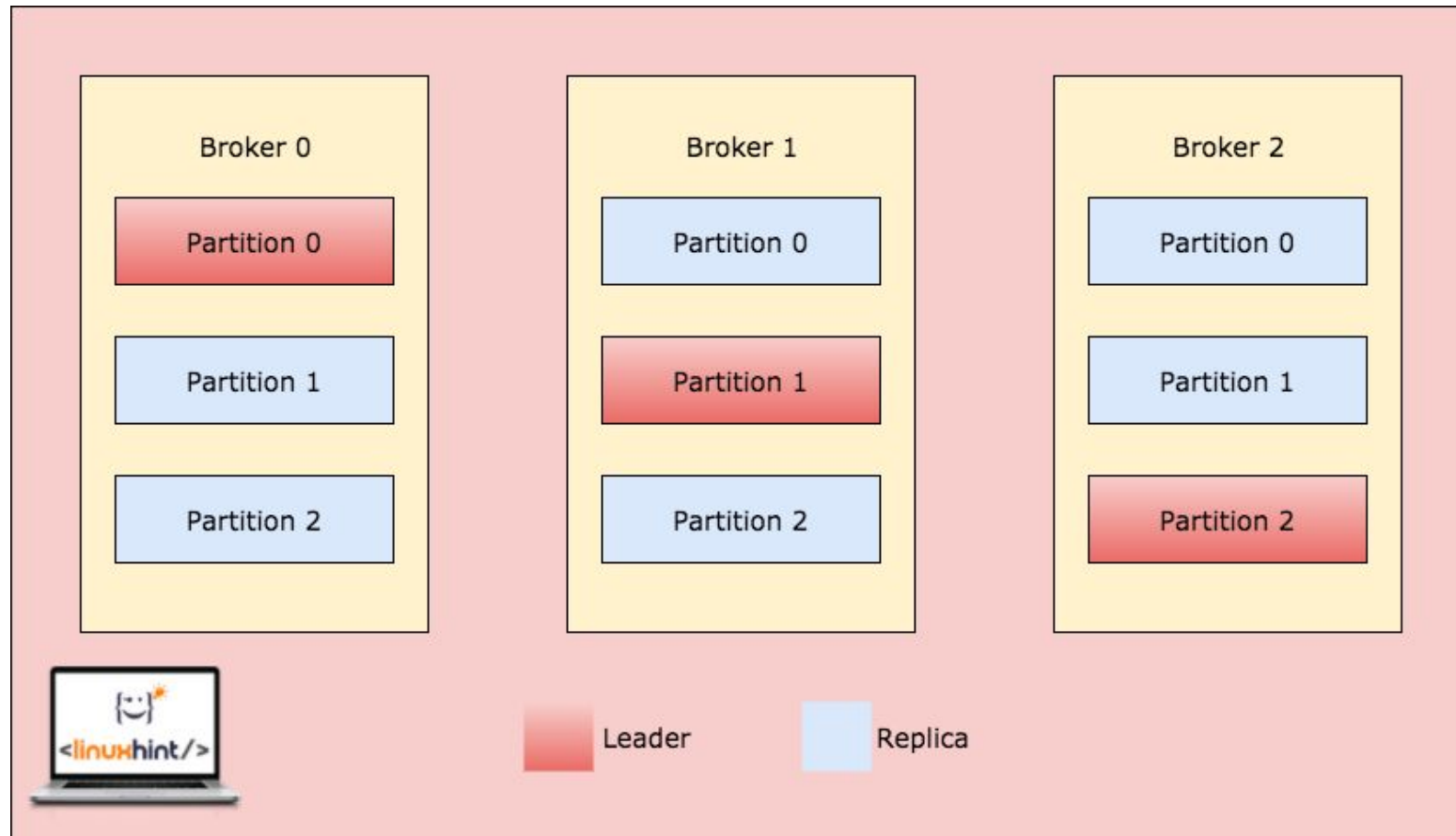
# Partitions in Apache Kafka





# Replication in Kafka

Fault tolerance



# Configuration terms

# Configurations Topic

- **retention.ms** => Amount of time logs will stay before they get deleted
- **cleanup.policy**=[delete|compact], either delete the messages from topic or compact them
- **partition**, scalability count
- **replication**, number of times a partition will be replicated

# Configuration Consumer

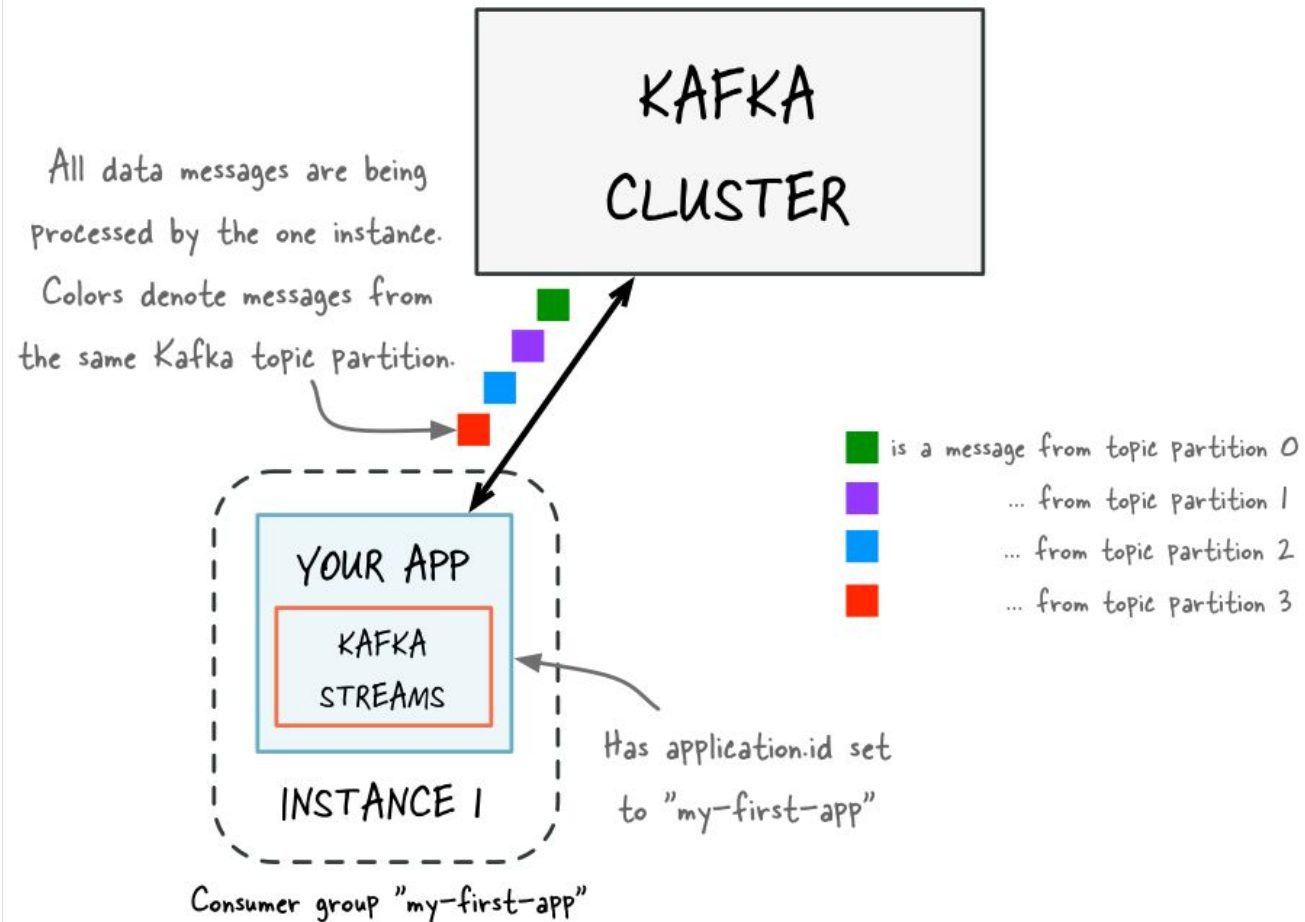
- **offset** => What has already been read by the consumer
- **consumer.group.id** => Identifier for the consumer group
- **auto.offset.reset**=[earliest|latest], when consumer connects for first time to a topic (offset does not exists for this consumer.group.id), where to start reading from. From first (earliest) message or last (latest) message

# Configuration Producer

- **acks:** [0 | 1 | all]
  - 0 => Does not wait for leader or replica broker to write the message to disk
  - 1 => Waits for leader broker to write the message to disk
  - all => Waits for leader and all replica to write the message to disk

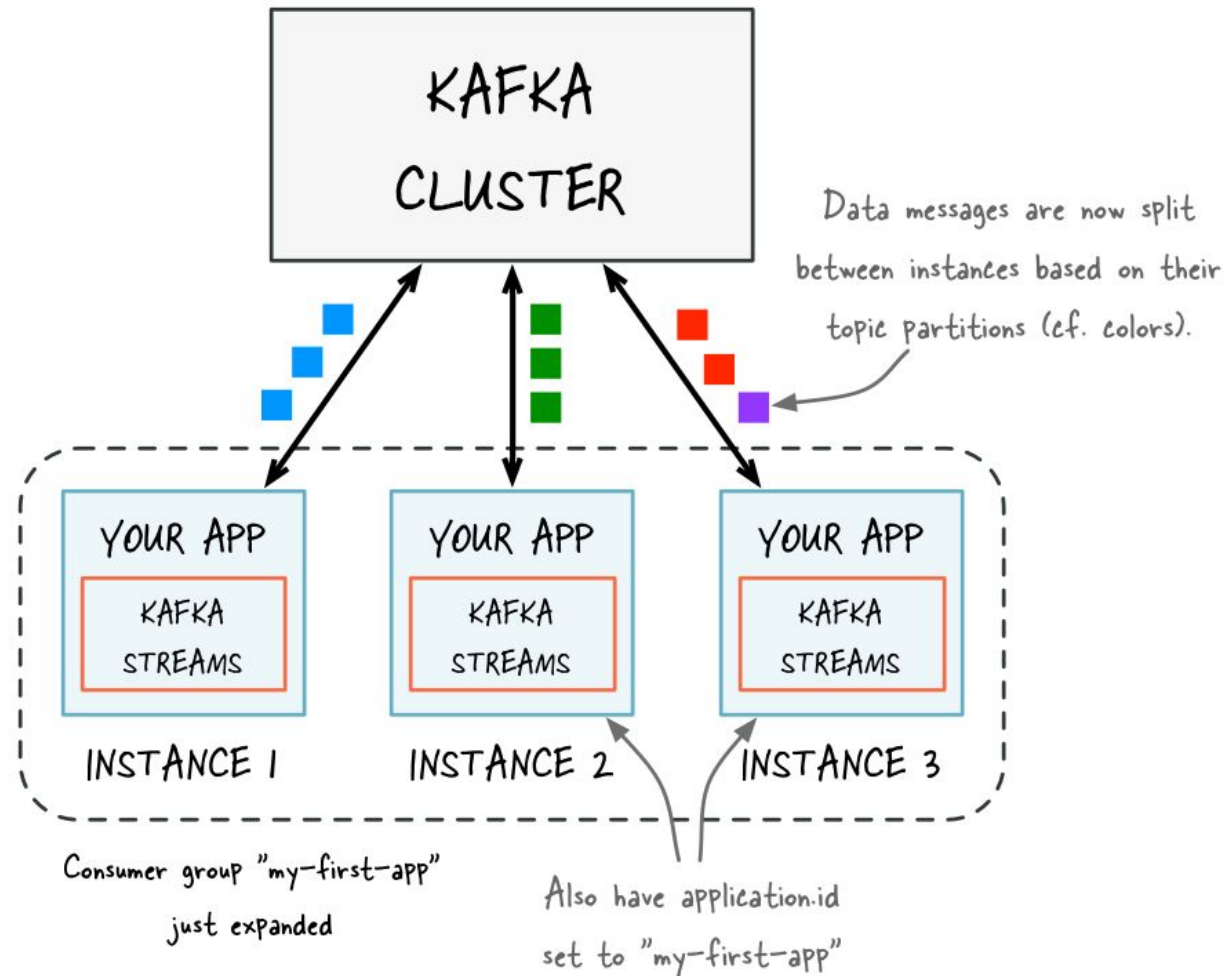
# Kafka Scaling

# Kafka Scaling

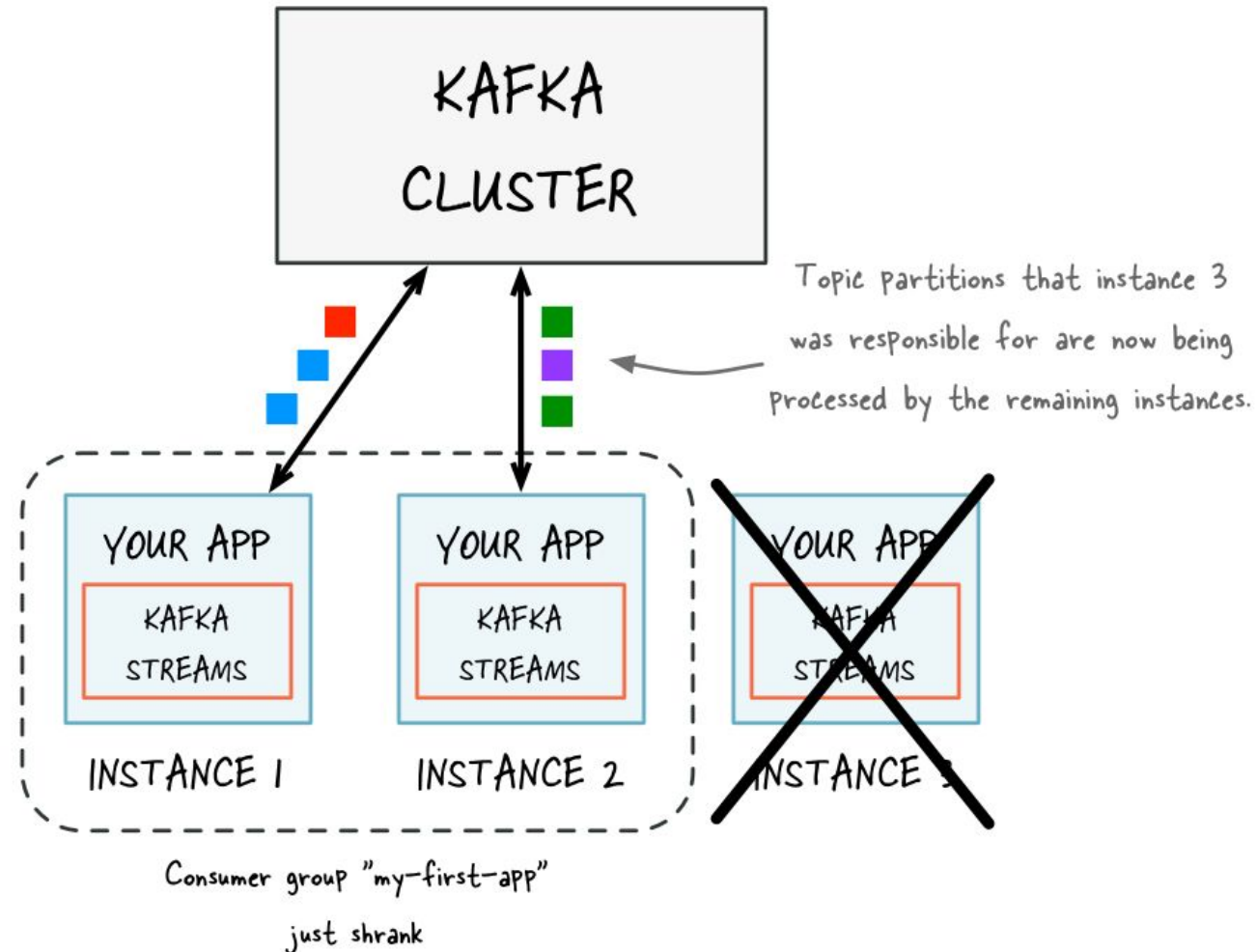




# Kafka Scaling

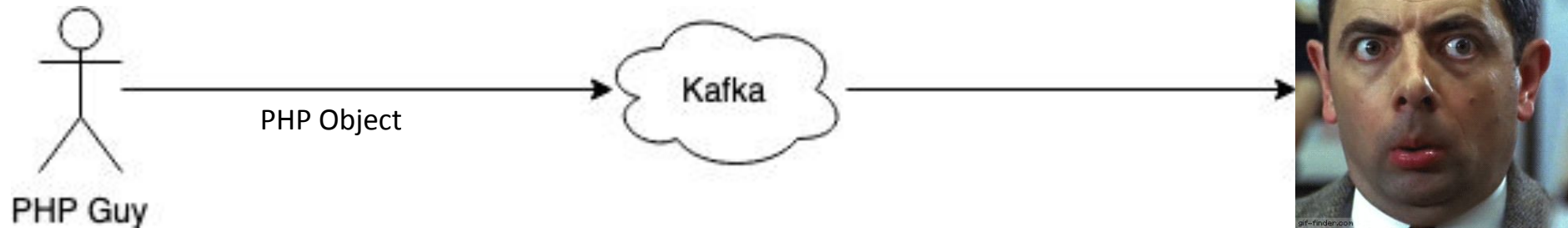


# Kafka Scaling



# Avro & Schema Registry

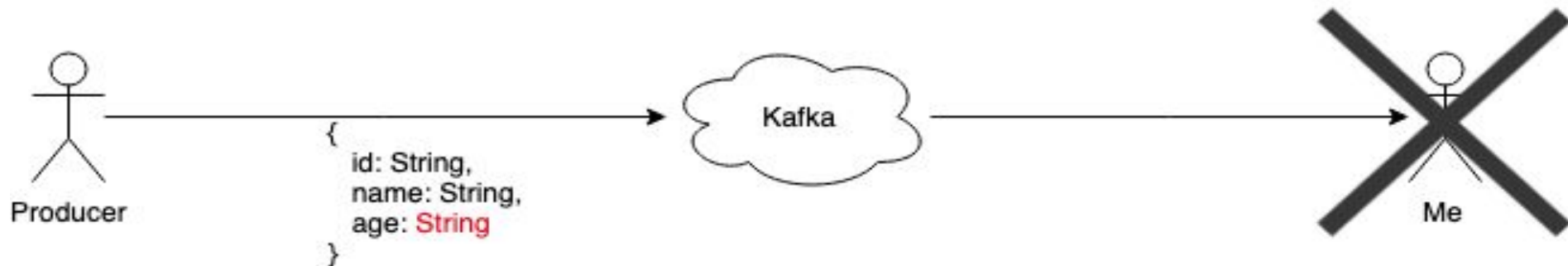
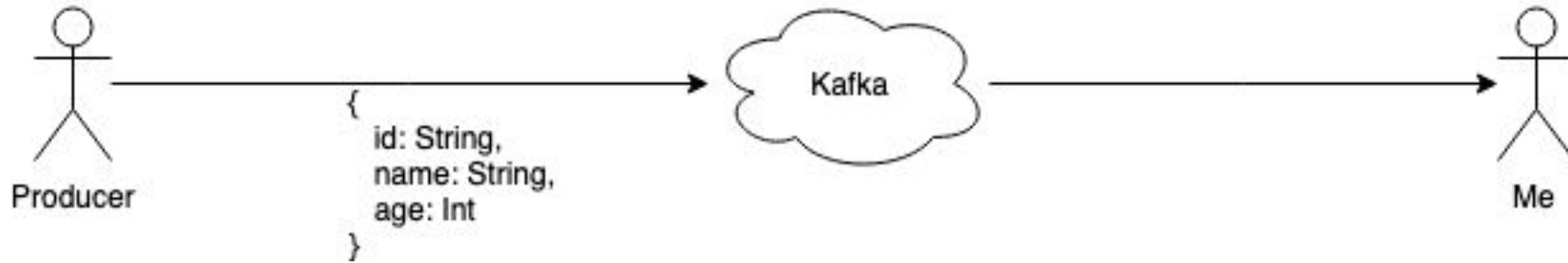
# Why is schema needed?



# What is Avro

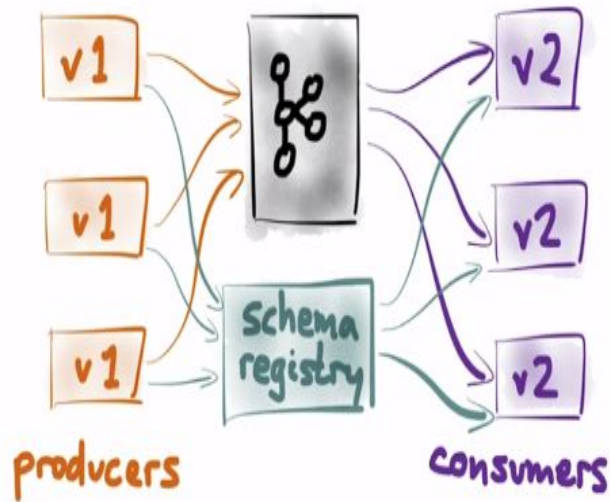
- **Avro** is a data serialization system
- **Schema stored separate from Record** (i.e. need schema to read record) (unlike *ProtoBuffers* or *JSON*)
- Records stored using binary encoding or JSON
- Avro advantages:
  - Smaller filesize (vs JSON)
  - Schema evolution
  - Avro clients provide automatic validation against schema

# Why Schema Compatibility?

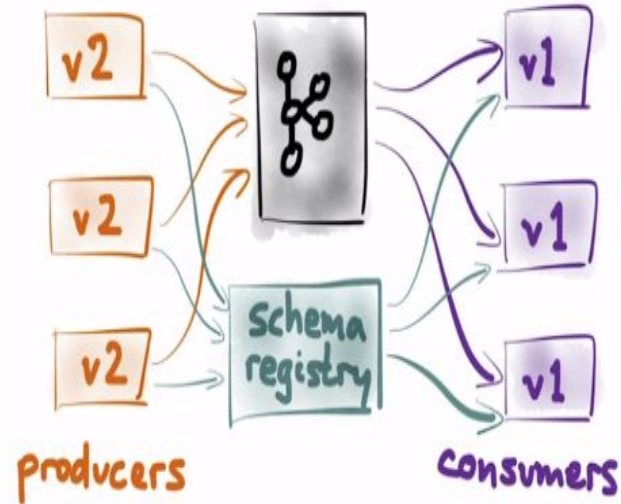


# Avro schema evolution

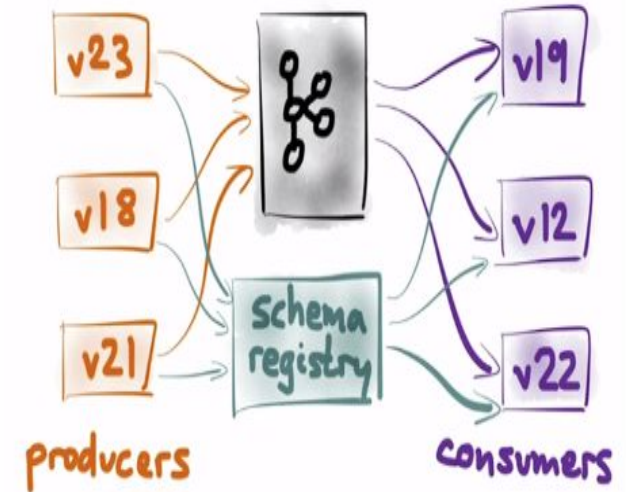
Backward compatibility



Forward compatibility

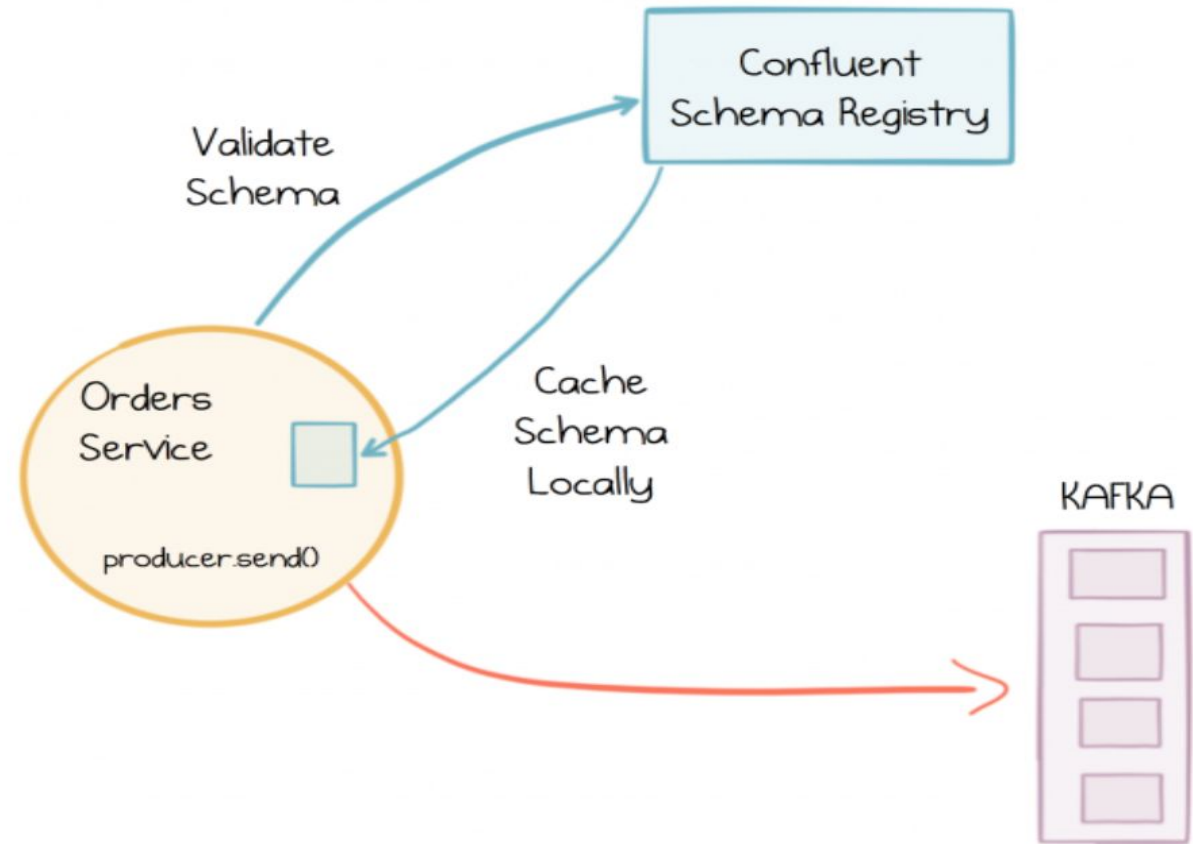


Mixed versions



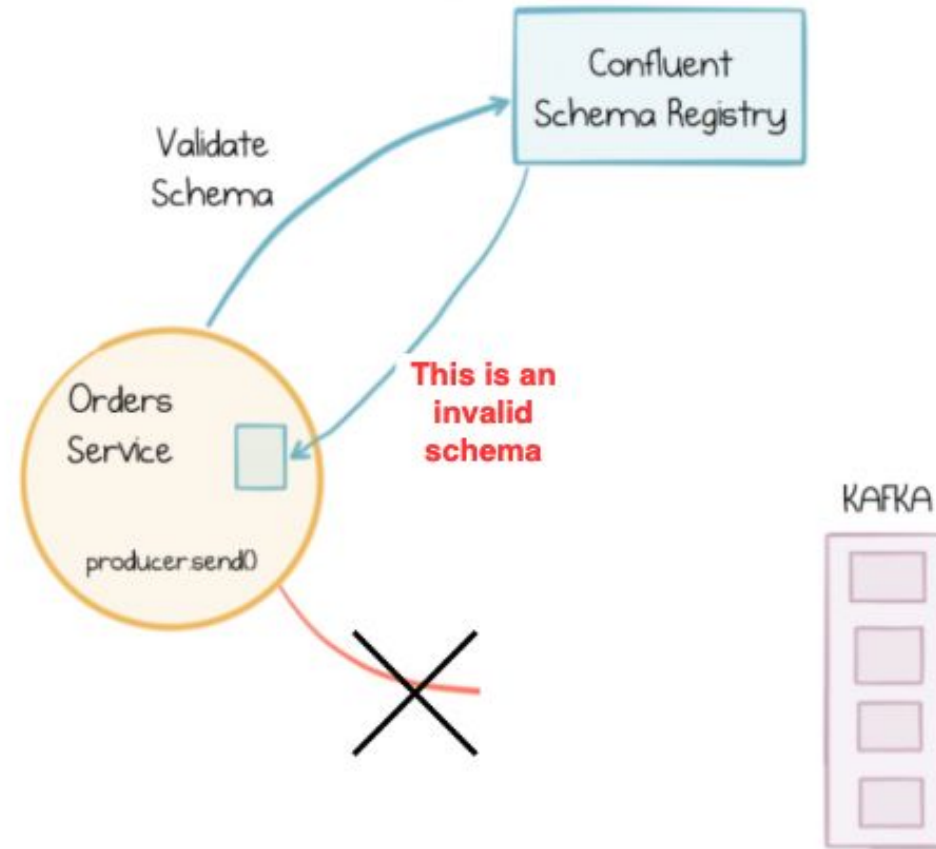
# Schema Registry

The Confluent Schema Registry provides both runtime validation of schema compatibility, as well as a caching feature for Avro schemas, so they don't need to be included in the message payload.



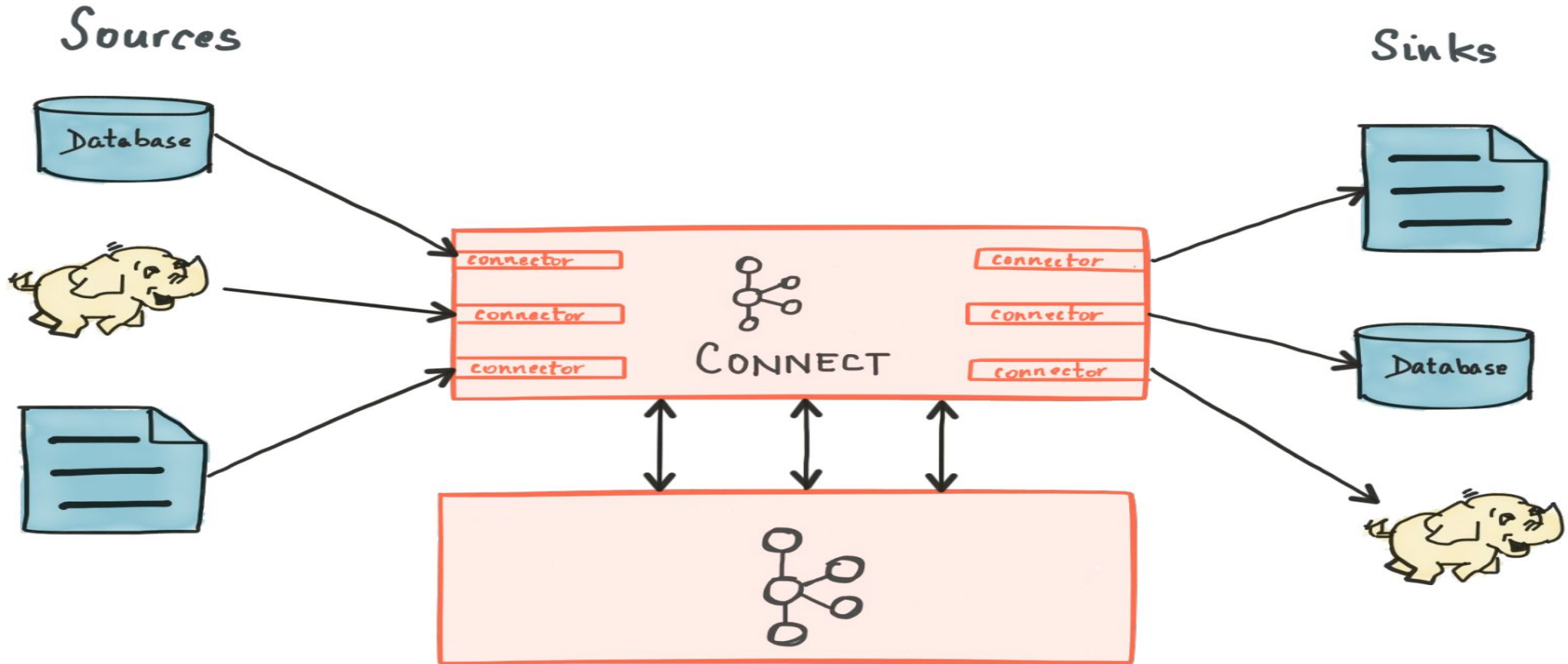


# Schema Registry



# Kafka Connect

# Sources and Sinks - Connectors



# Available - Kafka Connect

<https://www.confluent.io/hub/>

Alternatives:

- Alpakka
- Custom Kafka Connect
- ...

```
curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d '{
```

```
  "name": "google_history_trip_connector",
  "config": {
    "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
    "tasks.max": "1",
    "topics": "google_history_trip",
    "connection.url": "jdbc:mysql://localhost/kafka_offload",
    "auto.create": "true",
    "connection.user": "root",
    "connection.password": "*****",
    "key.converter": "io.confluent.connect.avro.AvroConverter",
    "value.converter": "io.confluent.connect.avro.AvroConverter",
    "value.converter.schema.registry.url": "http://localhost:8081",
    "key.converter.schema.registry.url": "http://localhost:8081",
    "key.converter.schemas.enable": "true",
    "value.converter.schemas.enable": "true"
```

# Kafka Stream

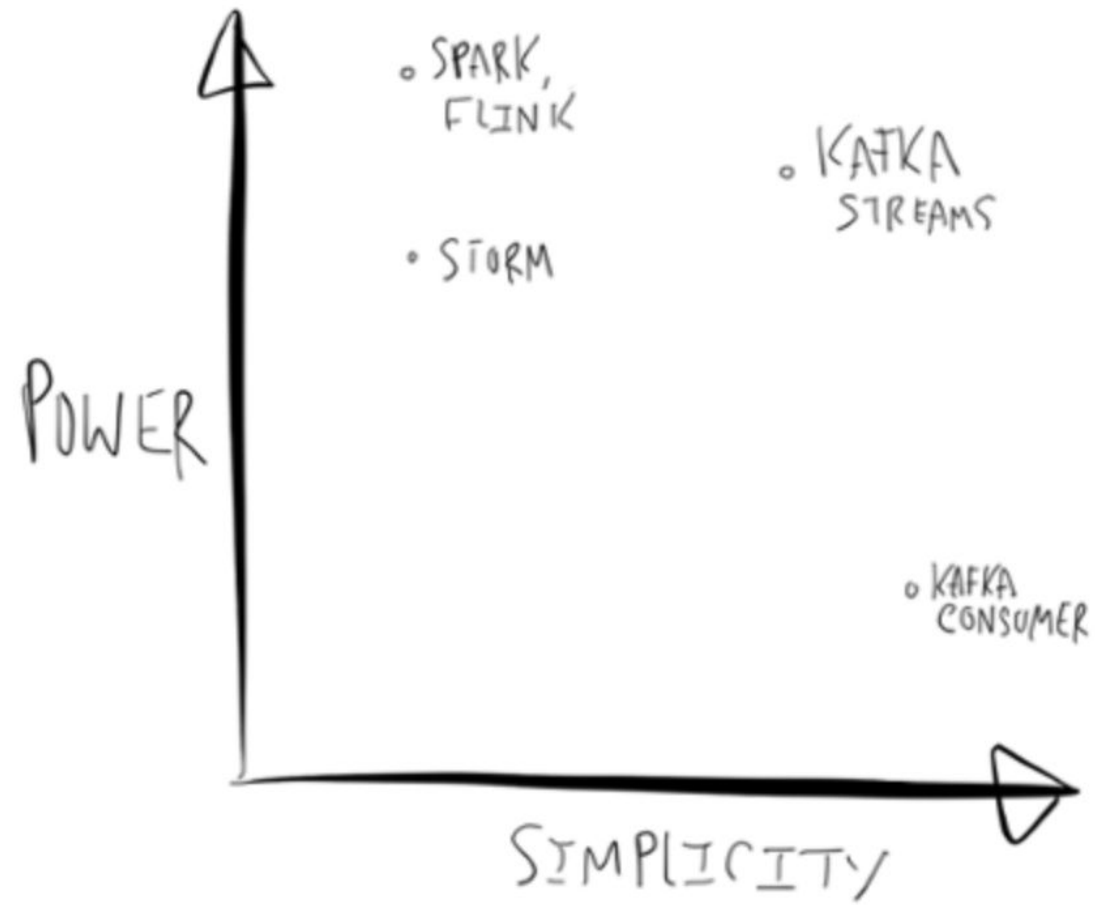
# Kafka stream

- Client library for building stream application
- Data from Kafka to Kafka
- Stream application
  - Fault tolerant
  - Scalable
- Event processing with milliseconds latency
- Provides a convenient DSL

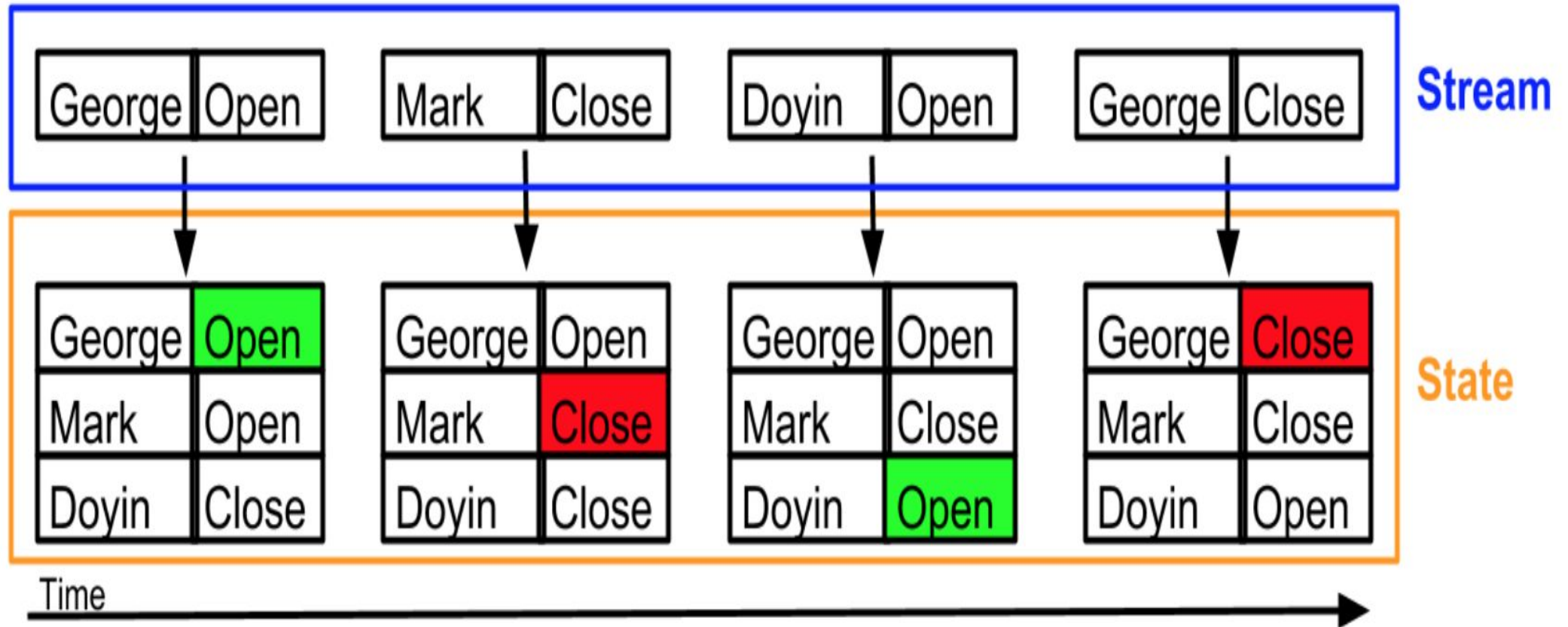
# Kafka stream

- Kafka stream in short
  - Millisecond delay
  - Balance the processing load as new instances of your app are added or existing ones crash
  - Maintain local state for tables
  - Recover from failures

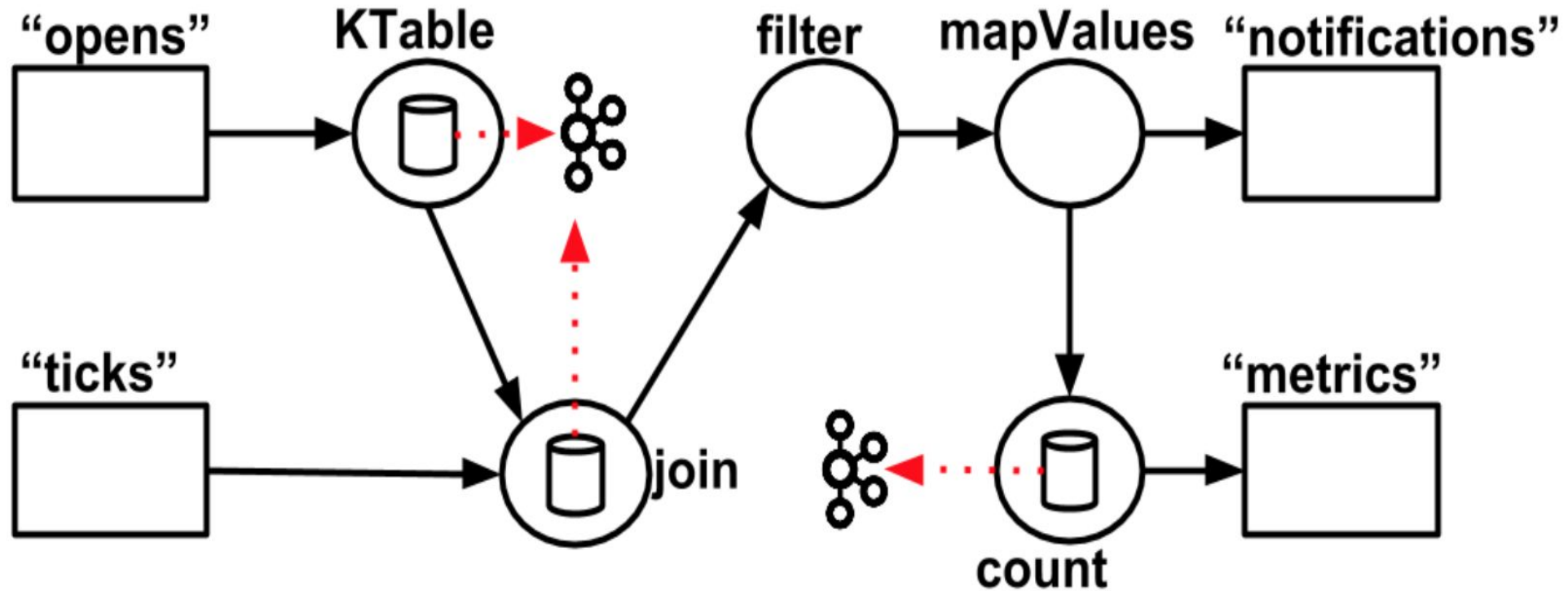




# Kafka streams vs State



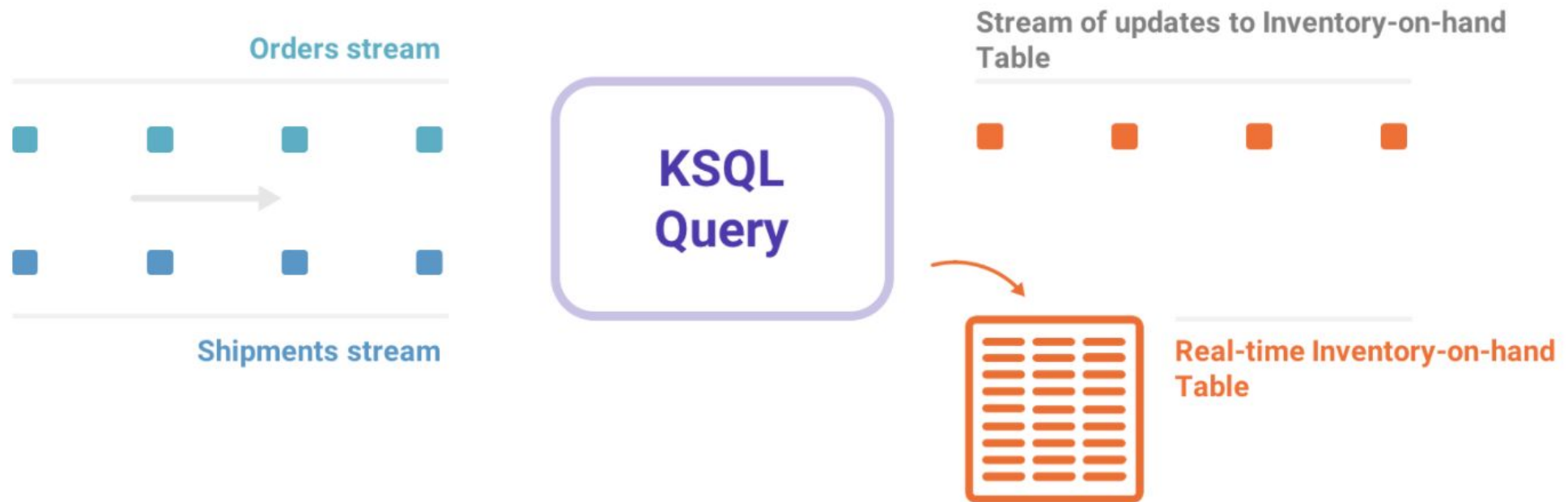
# Kafka streams topology



# Kafka stream features

- Aggregates: count, groupby
- Stateful processing (Stored internally in Kafka topic)
- Joins
  - KStream with KStream
  - KStream with KTable
  - KTable with KTable
- Windows
  - Time based
  - Session based

KSQL



**What questions to ask?**

# What questions to ask?

- **Replication factor?**
  - $\geq 3$ , for Kafka Brokers ( $\geq 3$ )
- **Retention time?**
  - $\leq 1$  month, think wisely if you would like to have more
- **Partition size?**
  - $\geq 5$  &  $\leq 20$  (normal cases), depends on your size of incoming data, consumer throughput
  - Kafka can handle high partition size, but has extra cost
- **Is key needed?**
  - Yes, same key to same partition
  - Prevent skew - if key is random enough
  - Scalability
  - Worst case: null, never a constant

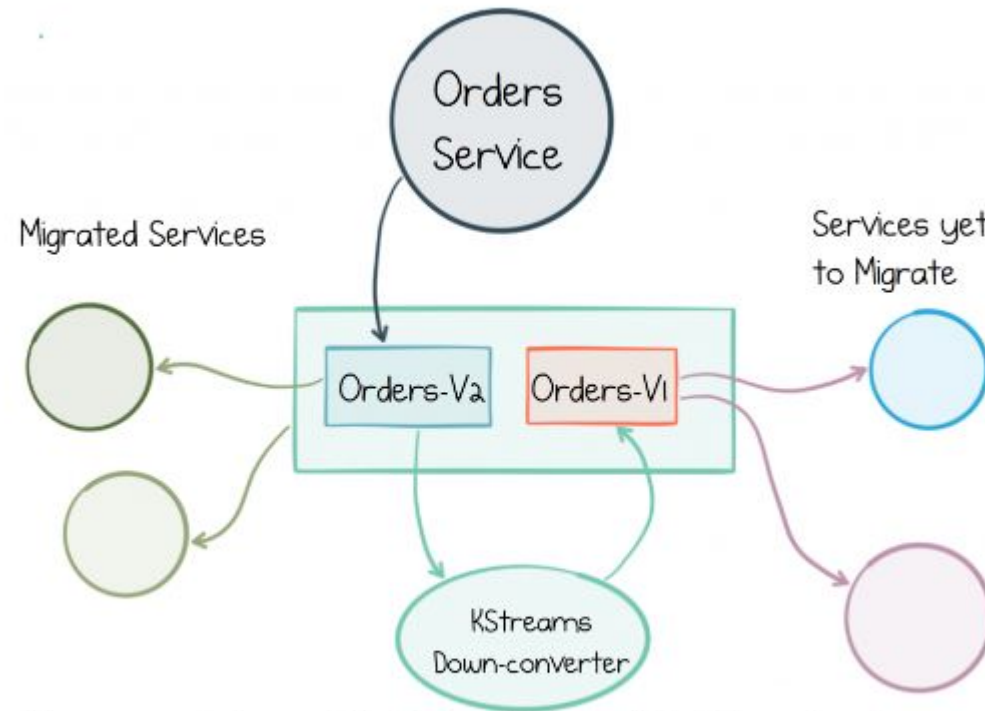


# What questions to ask?

- **Topic name structure?**
  - tr-<TEAM\_NAME>-<CONTENT>-<EXTENSION>
- **Consumer group-id?**
  - tr-<TEAM\_NAME>-<CONTENT>-<ACTION>-<EXTENSION>

Questions?

# What if Schema cannot be compatible?



The same data coexists in two topics, with different schemas, so there is a window during which services can upgrade.

