High Scalable Streaming Microservices with Kafka Streams

Who am I?

Roberto Perez Alcolea

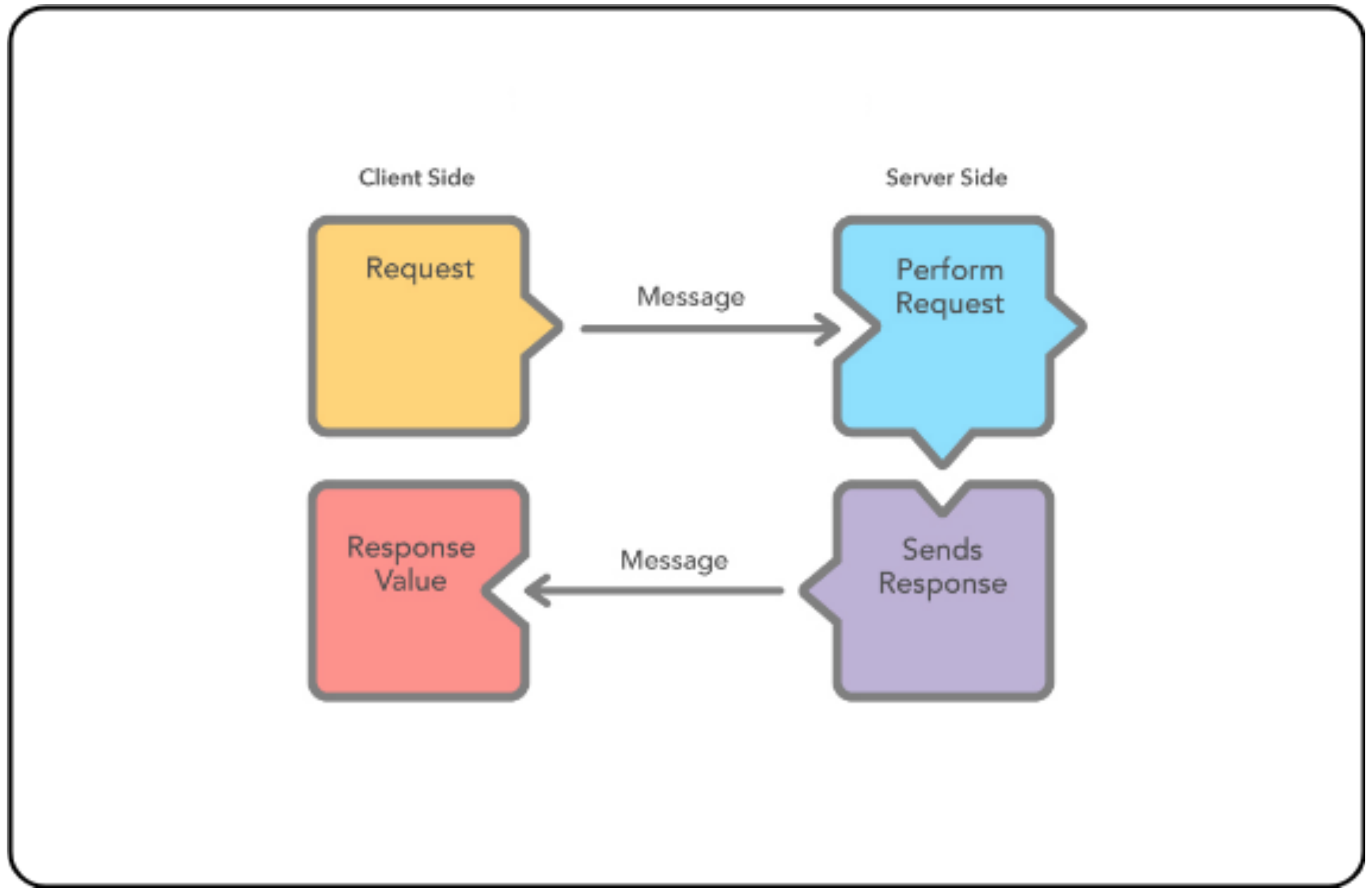
- Mexican
- Streaming Platform @ Target
- Groovy Enthusiast
- roberto@perezalcolea.info
- @rpalcolea

Stream Processing

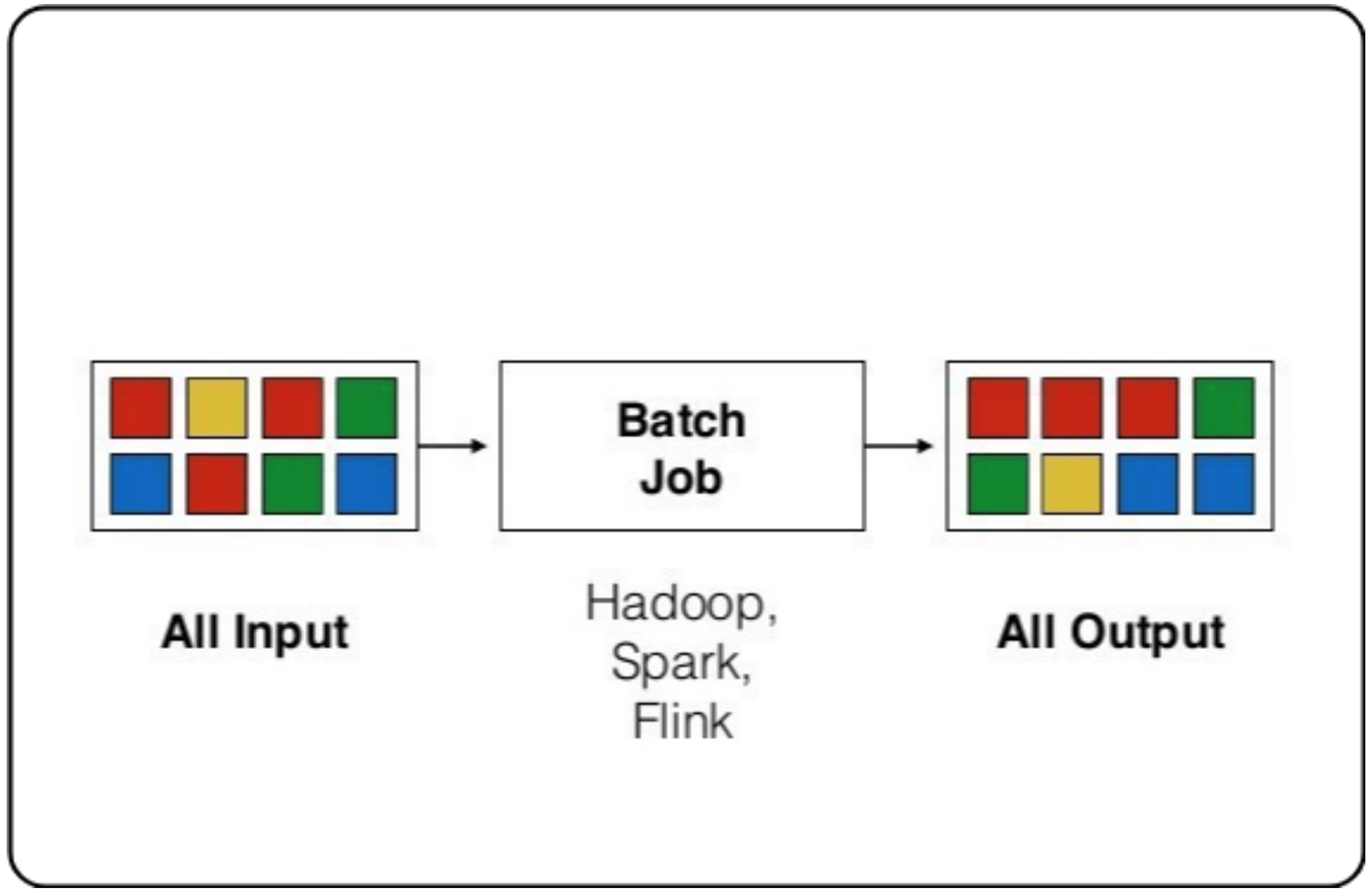
Paradigms getting input and producing outputs

- Request/Response
- Batch
- Stream processing

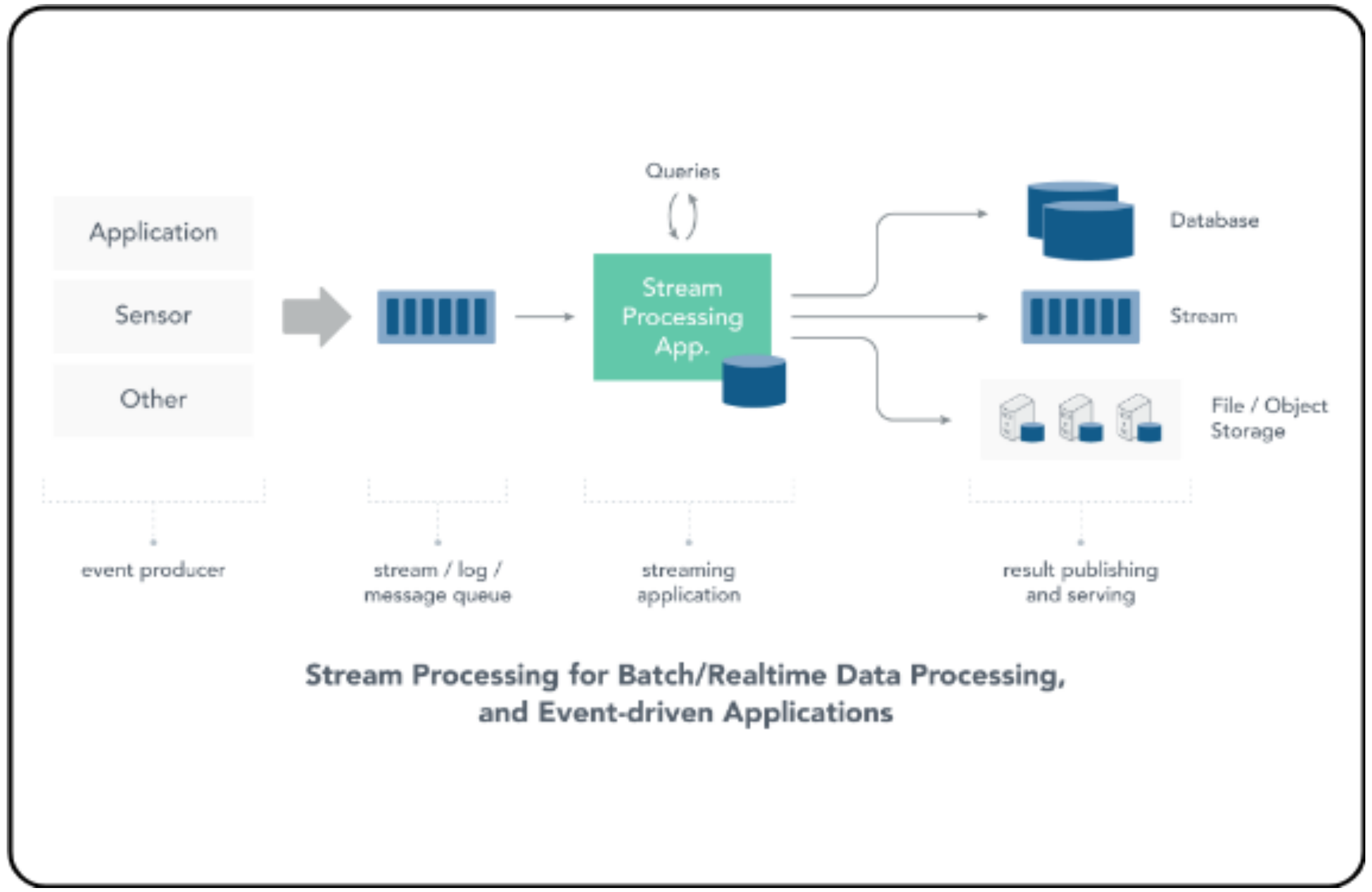
Request/Response



Batch



Stream Processing


Queries
Application
Sensor
Other
Stream Processing App.
Database
Stream
File / Object Storage
event producer
stream / log / message queue
streaming application
result publishing and serving
Stream Processing for Batch/Realtime Data Processing, and Event-driven Applications

Stream Processing

- Applications react to events instantly
- Can handle data volumes that are much larger than other data processing systems
- Naturally and easily models the continuous and timely nature of most data
- Decentralizes and decouples the infrastructure
- Asynchronous

Stateful Stream Processing

- Computation maintains contextual state
- State is used to store information derived from the previously-seen events
- Requires a stream processor that supports state management

Stream Processing - Hard?

- Partitioning & scalability
- Fault tolerance
- Time
- Re-processing

Stream Processing - Use cases

- Network monitoring
- Intelligence and surveillance
- Risk management
- E-commerce
- Fraud detection
- Smart order routing

Stream Processing - Semantics

Systems fail! Depending on the action the producer takes to handle such a failure, you can get different semantics:

- At least once
- At most once
- Exactly once

Apache Kafka

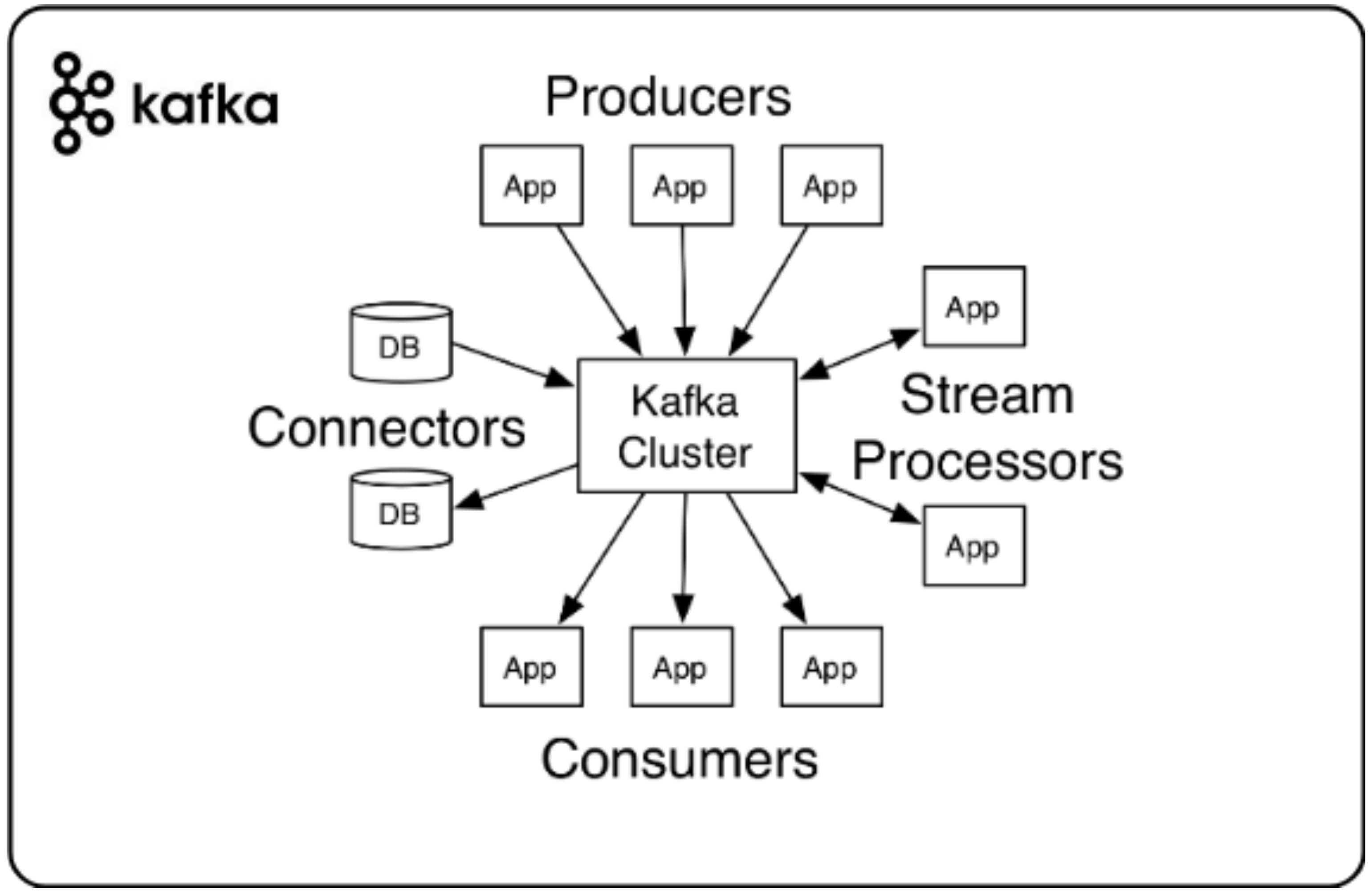
What is Apache Kafka?

- Distributed streaming platform
- Based on an abstraction of a distributed commit log
- Created and open sourced by LinkedIn
- Provides low-latency, high-throughput, fault-tolerant publish and subscribe pipelines and is able to process streams of events

Apache Kafka - Concepts

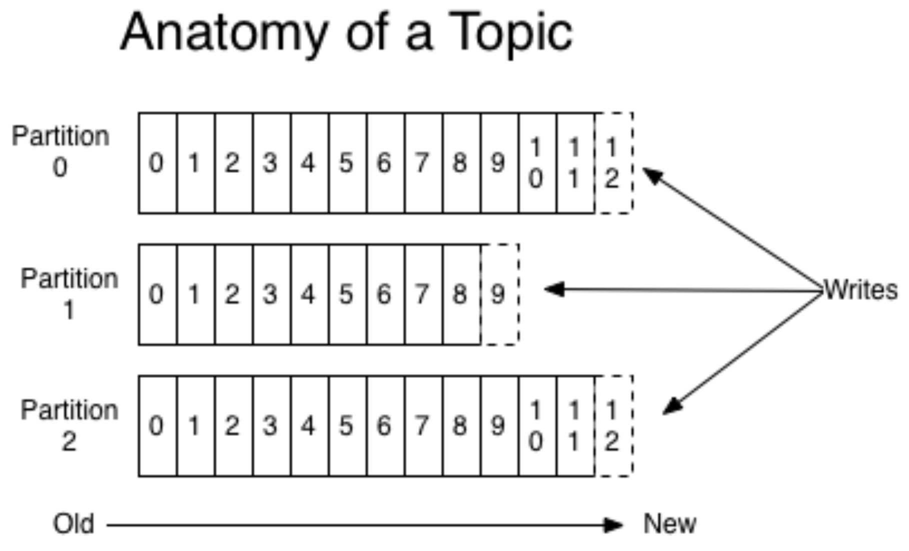
- Run as a cluster on one or more servers that can span multiple datacenters
- The Kafka cluster stores streams of records in categories called topics
- Each record consists of a key, a value, and a timestamp

Apache Kafka - Core APIs



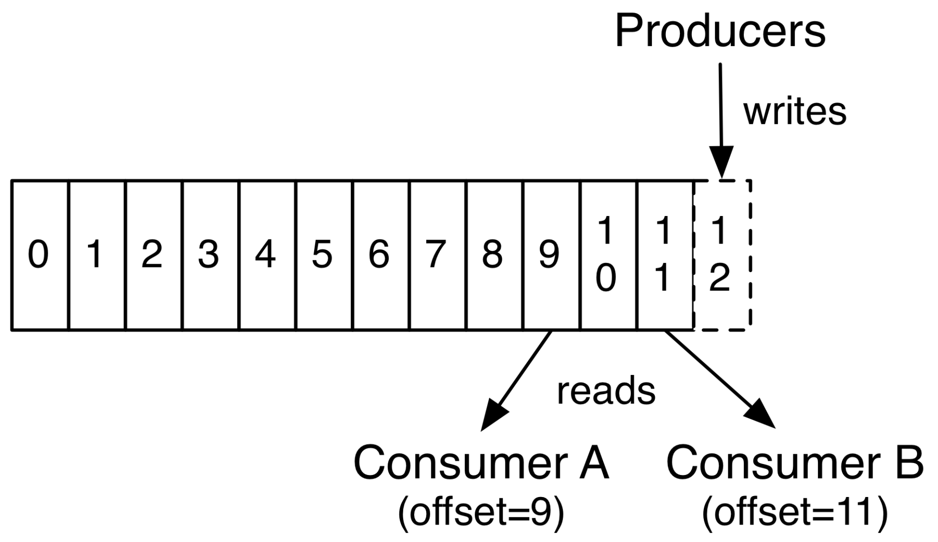
Apache Kafka - Topics and logs

- Category or feed name to which records are published
- Multi-subscriber
- Kafka cluster maintains a partitioned log for each topics



Apache Kafka - Topics and logs

- Partition is an ordered, immutable sequence of records
- Records have offsets
- Records are persisted with a retention period

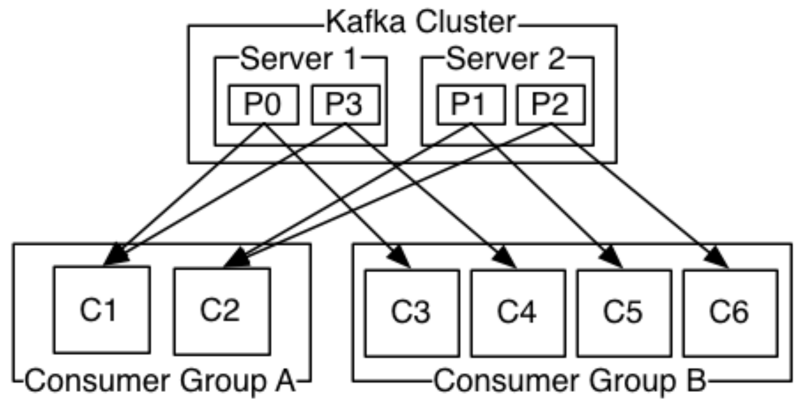


Apache Kafka - Producers

- Publish data to the topics of their choice
- Responsible for choosing which record to assign to which partition within the topic

Apache Kafka - Consumers

- Has a consumer group
- Records are load balanced over the consumer instances of the same group
- Partitions are divided across consumer instances

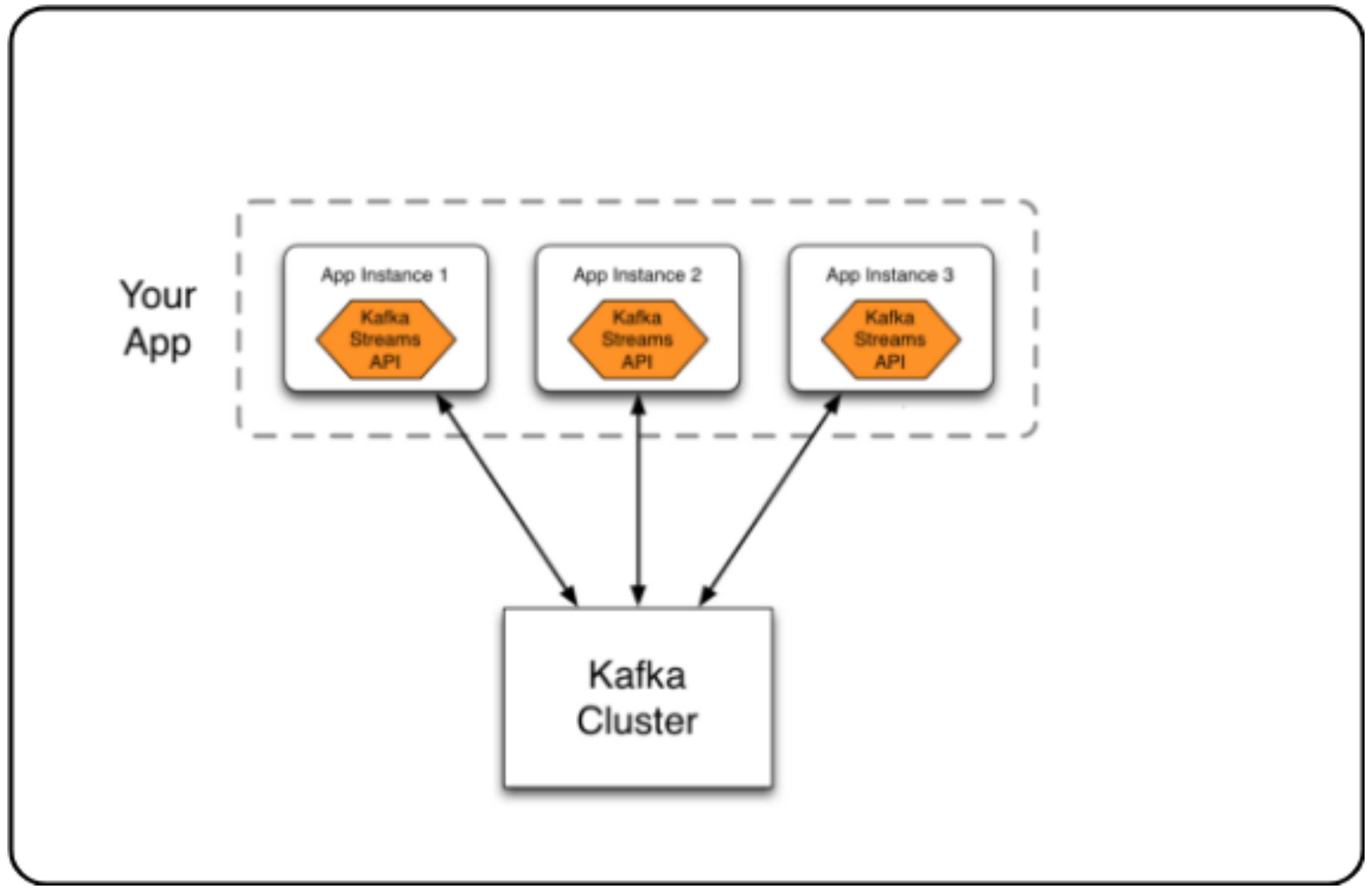


Kafka Streams

What is Kafka Streams?

- Built upon important concepts for stream processing
- Java Library
- Highly scalable, elastic and fault tolerant
- Exactly Once capabilities

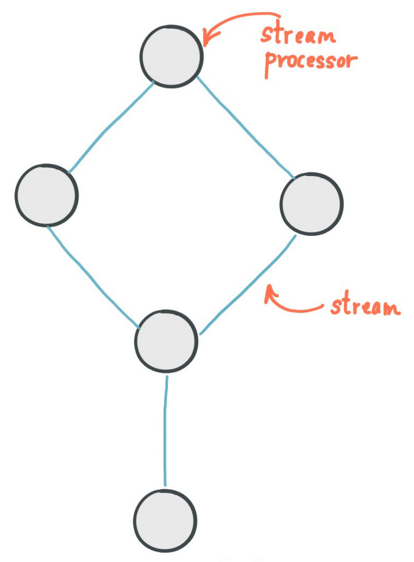
What is Kafka Streams?



Concepts

Processor topology

- Defines the computational logic of the data processing that needs to be performed by a stream processing application
- low-level Processor API or Kafka Streams DSL



PROCESSOR TOPOLOGY

KStream

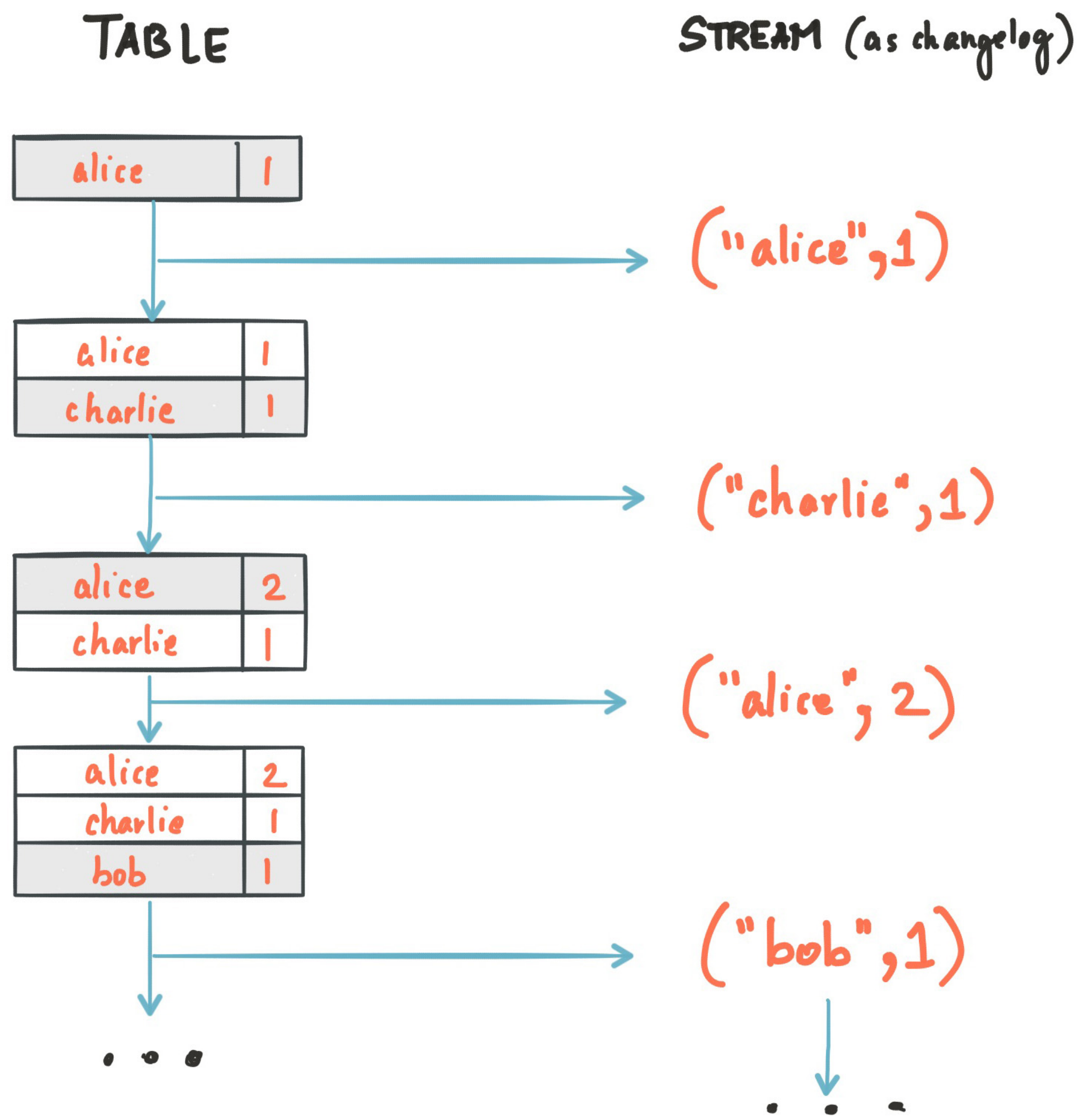
- Record stream abstraction
- Read from/written to external topic or product from other KStream
- append-only

```
("alice", 1) --> ("alice", 3)
```

KTable

- Changelog stream abstraction (snapshot of the latest value for each key in a stream)
- Each data record represents an update
- Produced from other tables or stream join/aggregation
- Read from external topic

KTable and KStream

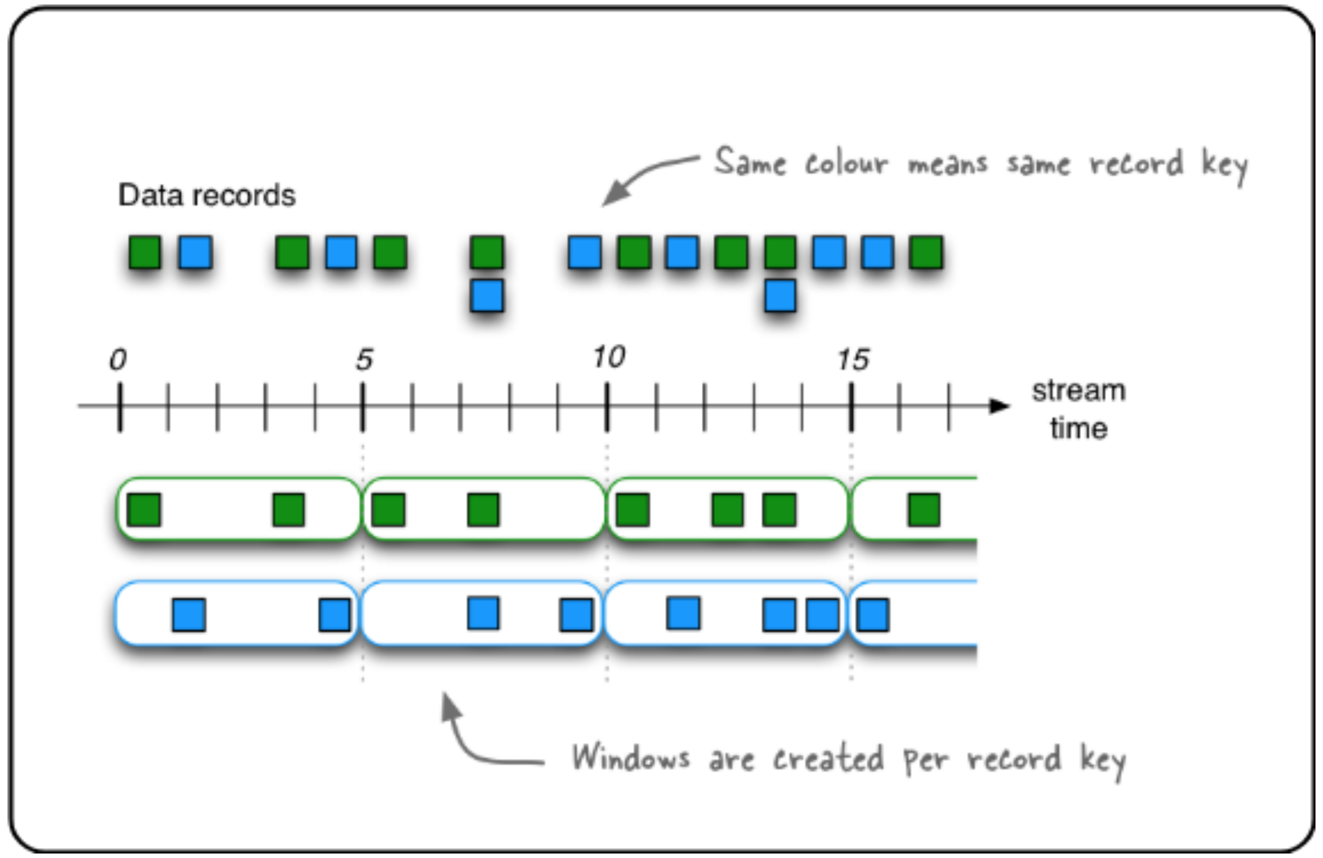


State Store

- Key-value store for stateful operations
- RocksDB or in-memory hash map
- Fault tolerant
- Interactive Queries

Time Windows

- Control how to group records that have the same key for stateful operations


Data records
Same colour means same record key
0
5
10
15
stream time
Windows are created per record key

Important Considerations

- Internal topics
- Need of disk when using RocksDB
- Proper partitioning

Example

```
static void main(final String[] args) throws Exception {
       Properties streamsConfiguration = KafkaStreamsConfig.getConfig("order-filter-example")
       final StreamsBuilder builder = new StreamsBuilder()

       KStream<String, String> ordersStream = builder.stream("orders")
       KStream<String, String> ordersPerBook = ordersStream.filter({
           key, value -> objectMapper.readValue(value, Order).quantity > 5
       })
       ordersPerBook.to("filtered-orders")

       final KafkaStreams streams = new KafkaStreams(builder.build(), streamsConfiguration)
       streams.cleanUp()
       streams.start()
}
```

Demo

Useful links

https://docs.confluent.io/current/streams/index.html
https://kafka.apache.org/documentation/
https://github.com/rpalcolea/gr8confus-2018-presentations
https://github.com/rpalcolea/gr8confus-2018-kafka-streams-demos

Thank

You