lessons learned so far

Tiago Palhota
@ FirstDerivatives

Session 1

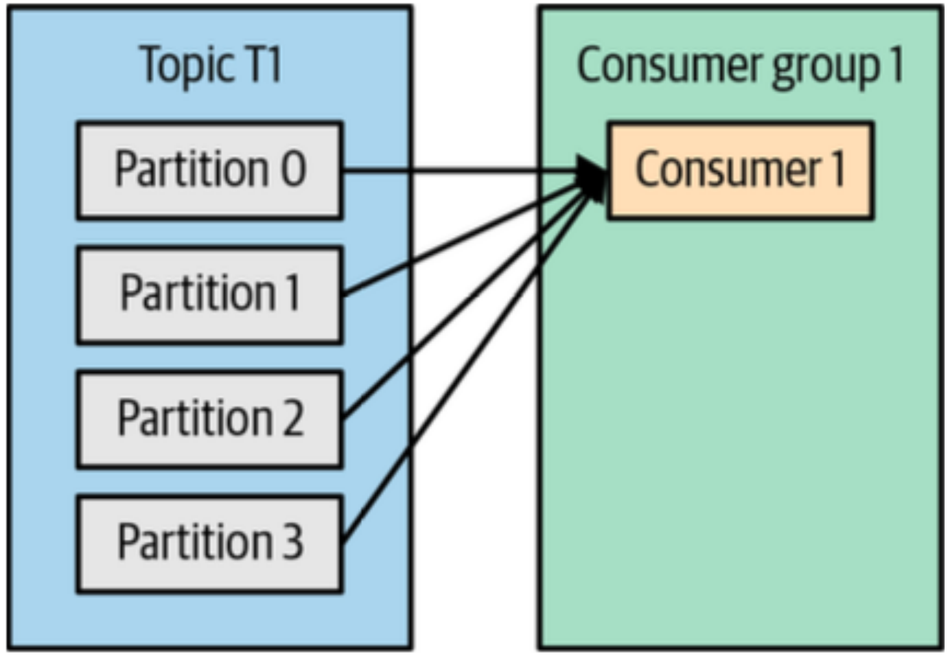
- Kafka - Introduction
- Kafka Elements
- CLI commands
- KAFKA API - Producers vs Consumers

Agenda

- Consumer groups
- Consumers - Assign and Seek
- Producers - Synchronous vs Asynchronous
- Twitter API
- Elastic Search
- Connectors
- Kafka Streams

Consumer groups

- Way of parallel reading
  - Message consumption is load balanced across all consumers in a group
- Consumers usually do intensive operations that add latency (+ consumers)
- Who handles rebalances?
  - Group Leader
- Who handles group membership?
  - Group Coordinator



Single Consumer

Consumer Assign and SEEK

DEMO TIME!

Producers

How a message is sent

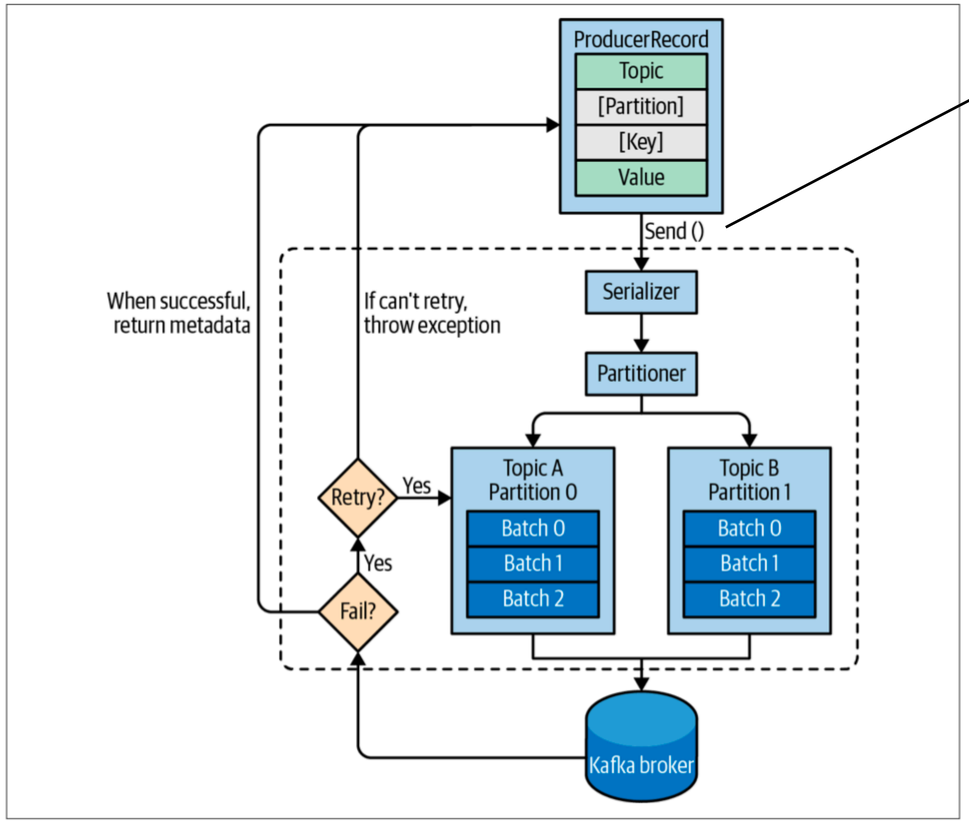


Figure 3-1. High-level overview of Kafka producer components

Returns a Future Object with recordMetadata

```
ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Precision Products",
        "France"); ❶
try {
    producer.send(record); ❷
} catch (Exception e) {
    e.printStackTrace(); ❸
}
```

APACHE kafka®

Producer

Sending messages synchronously

- Allows Sender to catch exceptions when:
  - Kafka replies back with an error
  - exhausted number of retries
- Brokers can take different times to get back
- Sending thread is blocked until broker replies back

```
ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Precision Products", "France");
try {
    producer.send(record).get(); ❶
} catch (Exception e) {
    e.printStackTrace(); ❷
}
```

Producer

Sending messages asynchronously

- Do we need the broker reply? Maybe not
- What about errors? Maybe log them for later analysis?
- Solution? Callbacks!

```
private class DemoProducerCallback implements Callback { ❶
    @Override
    public void onCompletion(RecordMetadata recordMetadata, Exception e) {
        if (e != null) {
            e.printStackTrace(); ❷
        }
    }
}

ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Biomedical Materials", "USA"); ❸
producer.send(record, new DemoProducerCallback()); ❹
```

Twitter API

Fetching data from Twitter

- Hosebird Java HTTP client for consuming Twitter Streaming API
  https://github.com/twitter/hbc

DEMO TIME!

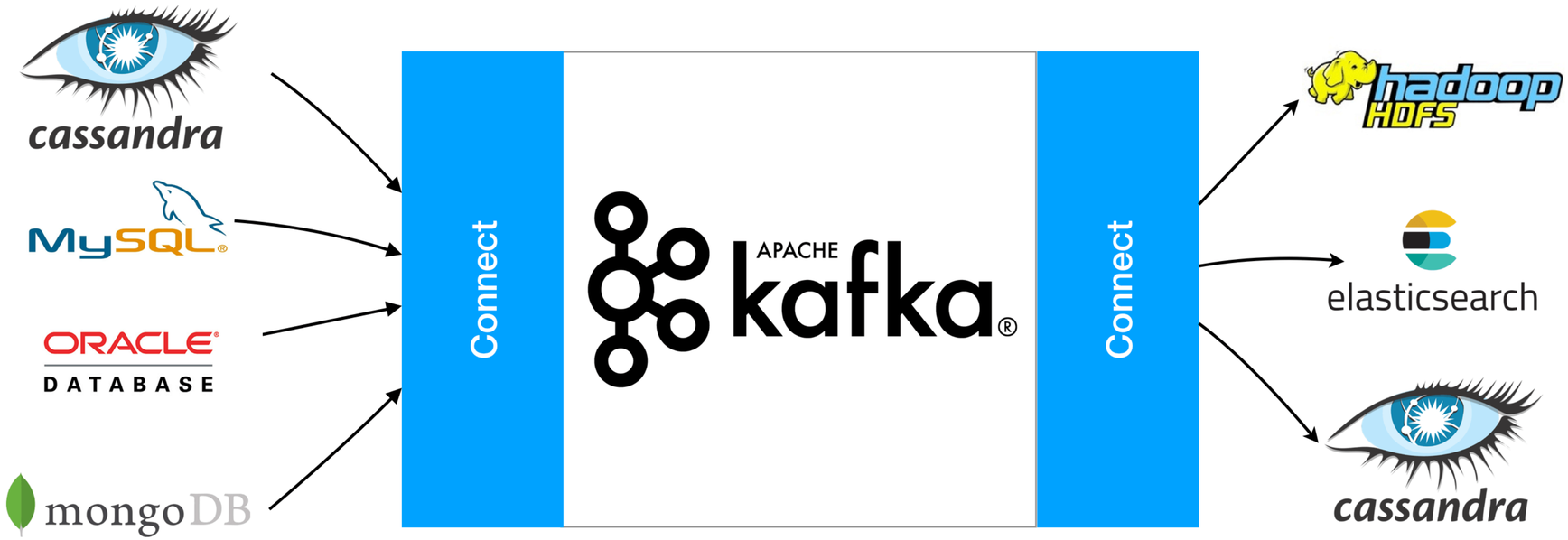
Elastic Search

Distributed search and analytical engine

DEMO TIME!

KAFKA Connectors

Save your time!



APACHE kafka

KAFKA Connectors

DEMO TIME!

KAFKA Streams

Filtering Tweets

DEMO TIME!

Questions?

Thank you!