Events with Kafka

In this lesson, we'll discuss events with Kafka.

WE'LL COVER THE FOLLOWING

Introduction

Systems that communicate via Kafka can easily exchange events (see also Events).

- Records can be saved permanently, and a consumer can read out the history and rebuild its state. The consumer does not have to store the data locally but can rely on Kafka.
- However, this means that all relevant information must be stored in the record. Events discussed the benefits and disadvantages of this approach.
- If an event becomes irrelevant due to a new event, the data can be deleted by Kafka's log compaction.
- Via consumer groups, Kafka can ensure that a single consumer handles each record. This simplifies matters, for example, when an invoice is to be written. In this case, only one consumer should write an invoice. It would be an error if several consumers were to create several invoices at the same time.

Sending events

The producer can send the events at different times.

After action has taken place

The simplest option is to send the event **after the actual action has taken place**.

The producer **first processes an order** before informing the other microservices about the order with an event. In this case, though, the producer could possibly change the data in the database and not send the event if, for example, it fails prior to sending the event.

Before data is changed

However, the producer can also send the events **before the data is actually changed**. When a new order arrives, the producer sends the event before modifying the data in the local database.

This doesn't make much sense though as events are information about an event that has already happened.

Finally, **an error may occur during the action**. If this happens, the event has already been sent, although the action never actually took place.

Sending events before the actual action also has the disadvantage that the **actual action is delayed**. First, an event is sent, which takes some time, and only after the event has been sent can the action be performed. The action is delayed by the time it takes to send the event. This can lead to a performance problem.

Send in a batch

It is also possible to collect the events in a local database and to **send them in a batch**.

In this case, writing the changed data and generating the data for the event in the database can take place in one transaction. The transaction can ensure that either the data is changed, and an event is created in the database, or neither takes place.

This solution also achieves **higher throughput** because batches can be used in Kafka to send several events to the database table at once.

However, the **latency is higher**; a change can be found in Kafka only after the next batch has been written.

QUIZ

1

Suppose you are working on an app that involves extremely time sensitive events. Which of the discussed approaches would you use in that case?

COMPLETED 0%

1 of 2

In the next lesson, we'll look at a live example of Kafka!