Variations & Experiments

In this lesson, we'll look at some variations to the approach that we've learned in this chapter and some experiments you can do based on them.

Recipe variations

The example sends the data for the event along in the records. There are alternatives to this (see Example):

- The **entire dataset is always sent along**, in this example that means the complete order.
- The records could contain only an ID of the dataset for the order. As a result, the **recipient can retrieve just the information about the dataset it really needs**.
- An individual topic exists for each client. All the **records have their own data structure** adapted to the client.

Other MOMs

An alternative to Kafka would be **another MOM**. This might be a good idea if the team has experience with a different MOM. Kafka differs from other MOMs in the long-term storing of records.

However, this is relevant only for event sourcing. And even then, every microservice can save the events itself. Therefore, storage in the MOM is not absolutely necessary. It can even be difficult because the question of the data model arises.

Atom

Likewise, asynchronous communication with Atom (see chapter 8) can be implemented.

In a microservices system, however, there should only be one solution for asynchronous communication so that the effort for building and maintaining the system does not become too great.

Therefore, using Atom and Kafka or any other MOM at the same time should be avoided.

Frontend integration

Kafka can be combined with frontend integration (see chapter 3).

Synchronous mechanisms

These approaches act at different levels so that a combined use does not represent a problem. A combination with synchronous mechanisms (see chapter 9) makes less sense because the microservices should communicate either synchronously or asynchronously.

Still, such a combination might be sensible in situations where synchronous communication is necessary.

Experiments

Try the following experiments in the coding environment below!

Supplement the system with an **additional microservice**.

- As an example, a microservice can be used that credits the customer with

- As an example, a microservice can be used that credits the customer with a bonus depending on the value of the order or counts the orders.

- Of course, you can copy and modify one of the existing microservices.

- Implement a Kafka consumer for the topic `order` of the Kafka server `kafka`. The consumer should credit the customer with a bonus when ordering or count the messages.

- In addition, the microservice should also present an HTML page with the information (customer bonuses or number of messages).

- Place the microservice into a Docker image and reference it in the `docker-compose.yml`. There you can also specify the name of the Docker container.

- In `docker-compose.yml`, create a link from the container `apache` to the container with the new service, and from the container with the new service to the container `kafka`.

- The microservice must be accessible from the homepage. To do this, you have to create a load balancer for the new Docker container in the file `000-default.conf` in the Docker container `apache`. Use the name of the Docker container, and then add a link to the new load balancer to the file `index.html`.

- It is possible to start additional instances of the shipping or invoicing microservice. This can be done with `docker-compose up -d --scale shipping=2` or `docker-compose up -d --scale invoicing=2`. `docker logs mskafka_invoicing_2` can be used to look at the logs. In the logs the microservice also indicates which Kafka partitions it processes.

- Kafka can also transform data with Kafka streams. Explore this technology by searching for information about it on the web!

- Currently, the example application uses JSON. Implement a serialization with Avro. A possible starting point for this can be https://www.codenotfound.com/2017/03/spring-kafka-apache-avro-example.html.

- Log compaction is a possibility to delete superfluous records from a topic. The Kafka documentation explains this feature. To activate log compaction, it has to be switched on when the topic is generated. See also

https://hub.docker.com/r/wurstmeister/kafka/. Change the example in such a way that log compaction is activated.

QUIZ

1 What are the reasons for using a MOM *other* than Kafka?

COMPLETED 0%

1 of 2

We’ll look at a quick chapter conclusion in the next lesson!