Chapter Conclusion

We'll conclude this chapter with a quick summary of what we've learned.

WE'LL COVER THE FOLLOWING

Kafka offers an interesting **approach for the asynchronous communication between microservices**.

Benefits

- Kafka can **store records permanently**, which facilitates the use of event sourcing in some scenarios. In addition, approaches like Avro are available for solving problems like schema evolution.
- The **overhead for the consumers is low**. They have to remember only the position in each partition.
- With partitions, Kafka has a concept for parallel processing and, with consumer groups, it has a concept to guarantee the order of records for consumers. In this way, Kafka can **guarantee delivery** to a consumer and at the same time **distribute the work among several consumers**.
- Kafka can **guarantee the delivery of messages**. The consumer commits the records it has successfully processed. If an error occurs, it does not commit the record and tries to process it again.

Therefore, it is worth taking a look at this technology, especially for microservices, even if other asynchronous communication mechanisms are also useful.

Challenges

Challenges

However, Kafka also provides some challenges.

- Kafka is an additional infrastructure component. It must be operated. Especially with messaging solutions, **the configuration is often not easy**.
- If Kafka is used as event storage, the records must contain all the data that all clients need. This is often **not easy to implement** because of the bounded context.

In the next chapter, we'll discuss asynchronous communication with Atom and REST.