Challenges

In this lesson, we'll look at possible challenges involved in a microservice architecture.

WE'LL COVER THE FOLLOWING

Increased operations effort

- The *operation* of a microservice system **requires more effort than running a deployment monolith**.
  - This is due to the fact that in a microservice system, many more deployable units exist that all have to be deployed and monitored.
  - This is feasible only when the operation is largely automated and the correct functioning of the microservices is guaranteed via appropriate monitoring.

Must Be Independently Deployable

- Microservices have to be **independently deployable**. For example, dividing them, into Docker containers is a prerequisite for this, but it is not enough on its own.
- Changes to interfaces must be implemented in such a way that an independent deployment of individual microservices is still possible.
  - For example, the microservice which implements the interface has

    - For example, the microservice which implements the interface has to offer the new and the old interface. Then this microservice can be deployed without requiring that the calling microservice be deployed at the same time.

Testing must be independent

- Also, **testing must be independent**. When all microservices have to be tested together, one microservice can block the test stage and prevent the deployment of the other microservices making testing much harder.
- Due to the split into microservices, there are more interfaces to test, and **testing has to be independent for both sides of the interface**.

Difficult to change multiple microservices

- Changes that affect **multiple microservices** are **more difficult to implement** than the changes that concern several modules of a deployment monolith.
    - In a microservice system, such changes require several deployments. These deployments must be coordinated.
    - In the case of a deployment monolith, only one deployment would be necessary.

Lost overview

In a microservice system, the *overview* of the microservices can get lost. However, experience teaches that in practice, a sound domain-based division can restrict changes to one or a few microservices. Therefore, the overview of the system is less important because the interaction between the microservices hardly influences development due to the high degree of independence.

Increased latency and failures

Microservices communicate through the *network*. Compared to local communication,

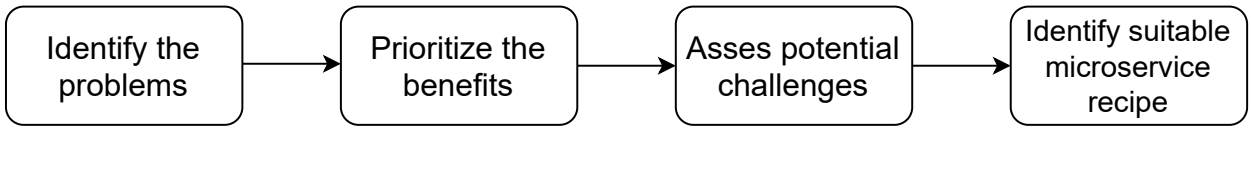
- The **latency is much higher**.
- It is also more likely that **communication will fail**.

A microservices system cannot rely on the availability of other microservices. This makes the systems more complex.

Weighing benefits and disadvantages

The most important rule is that microservices should only be used if they represent the simplest solution in a certain scenario.

The previously mentioned benefits should outweigh disadvantages resulting from the higher level of complexity for deployment and operation. Choosing a more complex solution is rarely a good idea.



Finding the right microservice recipe

Experiments

The following approach helps to find the right recipe to divide a system into microservices.

1. **Identify the problems** in your current system (for example, resilience, development agility, too slow deployment, and so on).
2. For the projects that you've worked on, **prioritize the benefits** of using microservices.
3. Weigh **which of the challenges** in this project could **pose a risk**.
4. Look at the possible technical and architectural solutions in the following chapters to determine the most sensible solutions for their requirements.

For the concrete division into microservices and for technical decisions, additional concepts are necessary. So, let's discuss the question of how best to divide a system into microservices in the next chapter.

QUIZ

QUIZ

1

Suppose you’re designing an application where one microservice gets data from another. If this data fetching fails, the functionality of the app will be compromised. What would be the best course of action in this situation?

COMPLETED 0%

1 of 3

In the next lesson, we’ll look at a quick summary of what we have learned in this chapter.