Reasons for Migrating

In this lesson, we'll discuss a few common reasons for migrating to a microservices architecture.

WE'LL COVER THE FOLLOWING

When migrating to microservices, it is **important to know the objectives for taking this step**. Depending on the reasons that led to this decision, the procedure for implementing them may vary.

Microservices offer a fresh start

Especially when replacing legacy systems, microservices have several advantages. Let’s discuss them.

The code of the legacy system no longer needs to be used in the new microservices because the microservices are implemented separately from the legacy system. The **microservices offer an unencumbered restart**.

The code of a legacy system is often no longer maintainable, and the technologies are frequently outdated. Therefore, reusing the old code would hinder the development of a clean new system.

Microservices thus solve the most important challenges when dealing with legacy systems, because otherwise, a restart is difficult.

Migration to microservices has the potential to solve the problem with the legacy system. After migration to microservices, further migrations can be limited to one or a few microservices. A migration of the entire system will probably not be necessary again.

A typical **reason for a system migration is outdated technology**. In a microservices system, such a migration can take place step by step – that is, microservice by microservice.

Another migration reason is an **unmaintainable system**. In this case, each microservice can be replaced individually.

The reasons are already known

The reasons for a migration are in the end identical to the reasons for using microservices. These have already been discussed in detail in the last chapter and may include:

- Increased security
- Robustness
- Independent scaling of individual microservices

A typical reason: speed of development

A typical reason for introducing microservices is the lack of speed in developing with a deployment monolith. When many developers are working on a deployment monolith, they need to **closely coordinate their work**. This costs time and therefore **slows down development**.

But even with a small team, a deployment monolith can be problematic because the **deployment is quite huge**. The size makes continuous delivery difficult to implement, and each release requires a lot of testing.

QUIZ

1 A team of two, working on a popular monolithic website with a very small codebase that started 15 years ago want to migrate to a microservices system. What would the primary reason for the migration be?

In the next lesson, we'll study typical migration strategies.