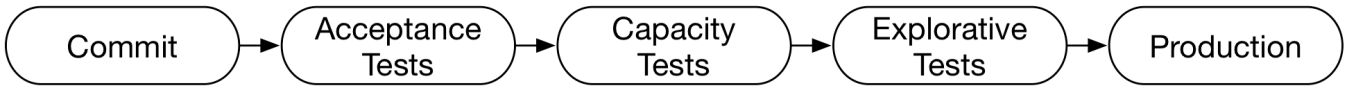
Advantage: Continuous Delivery

In this lesson, we'll focus on continuous delivery as an advantage of using microservices.

WE'LL COVER THE FOLLOWING

Continuous delivery

Continuous delivery is an approach where software is continuously brought into production with the help of a continuous delivery pipeline. The pipeline brings the software into production via different phases. Have a look at the following drawing:



Microservices continuous delivery pipeline

Let’s discuss each phase shown above.

Phases

- Typically, the **software compilation**, **unit tests**, and **static code analysis** are performed in the **commit phase**.
- In the **acceptance test** phase, **automated tests** assure the correctness of the software regarding domain logic.
- **Capacity tests** check the performance at the expected load.
- **Explorative** tests serve to perform not-yet-considered tests or to examine new functionalities. In this manner, explorative tests can analyze aspects

that are not yet covered by automated tests.
- In the end, the software is brought into **production**.

Microservices represent independently deployable modules. Therefore each microservice has its own continuous delivery pipeline.

This facilitates continuous delivery.

Microservices facilitate continuous delivery

- The continuous delivery pipeline is **significantly faster** because the deployment units are smaller. Consequently, deployment is faster.
- Continuous delivery pipelines contain many test stages. The software has to be deployed in each stage. Faster deployments speed up the tests and therefore the pipeline.
- The **tests are also faster** because they need to cover fewer functionalities. Only the features in the individual microservice have to be tested, whereas in the case of a deployment monolith, the entire functionality has to be tested due to possible regressions.
- **Building up a continuous delivery pipeline is easier** for microservices. Setting up an environment for a deployment monolith is complicated. Most of the time, powerful servers are required. In addition, third-party systems are frequently necessary for tests. A microservice requires less powerful hardware. Besides, not many third-party systems are needed in the test environments.
    - However, running all microservices together in one integration test can cancel out this advantage. An environment suitable for running all microservices would require powerful hardware as well as integration with all third-party systems.
- The **deployment of a microservice poses a smaller risk** than the deployment of a deployment monolith. In the case of a deployment monolith, the entire system is deployed anew, and in the case of a microservice, only one module. This causes fewer problems since less of the functionality is being changed.

In summary, **microservices facilitate continuous delivery**. Even their support of continuous delivery can be reason enough to migrate a deployment

monolith to microservices.

Deployment must be automated

However, note that microservice architectures can only work when the deployment is automated! Microservices substantially increase the number of deployable units compared to a deployment monolith. This is only feasible when the deployment processes are automated.

Independent deployment means that the continuous delivery pipelines have to be completely independent. Integration tests conflict with this independence. They introduce dependencies between the continuous delivery pipelines of the individual microservices. Therefore, **integration tests must be reduced** to the minimum. Depending on the type of communication, there are different approaches to achieve this for synchronous and asynchronous communication.

QUIZ

1 In what phase of continuous delivery would the performance of an application be checked against the expected load?

COMPLETED 0%

1 of 2

In the next lesson, we'll continue our discussion of the advantages of microservices.