Introduction

In this lesson, we'll look at an overview of what to expect from this chapter!

We'll cover the following

- Microservices: definition
    - Advantages of this microservice definition
    - Deployment monolith
    - Size of a microservice
- Chapter walkthrough

Microservices

Microservices: definition

Unfortunately, there is no universally acknowledged definition for the term *microservice*. In the context of this course the following definition will be used:

> **Microservices** are independently deployable modules.

For example, an **e-commerce system** can be divided into modules for:

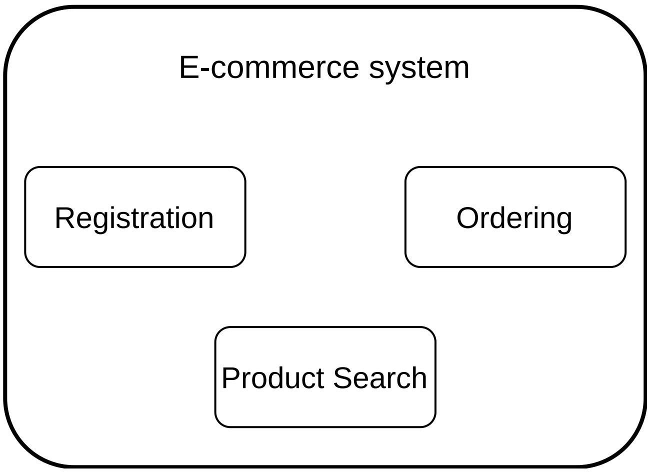
- ordering
- registration
- product search

Normally, all of these modules would be implemented together in one application. In this case, **a change in one of the modules** can only be brought into production by bringing a **new version of the entire application** with all its modules into production. However, when the modules are implemented as microservices, the ordering process cannot only be **changed independently of the other modules**, but it can even be brought into production independently.

This speeds up deployment and reduces the number of necessary tests since only a single module needs to be deployed. Due to this greater level of decoupling, a large project can turn into a number of smaller projects. Each project is in charge of an individual microservice.

To achieve this at the technical level, **every microservice has to be an independent process**. A better solution for decoupling microservices is to provide an independent virtual machine or Docker container for each microservice.

In that case, a deployment will replace the Docker container of an individual microservice with a new Docker container, which starts the new version and its direct requests. The other microservices will not be affected if such an approach is used.



An e-commerce system can be divided into modules as above

Advantages of this microservice definition

The definition of microservices as independently deployable modules has several advantages:

- It is very *compact*.
- It is very *general* and covers all kinds of systems which are commonly denoted as microservices.
- The definition is based on *modules* and is thus a well-understood concept. This allows us to adopt many ideas concerning modularization. This definition also highlights that microservices are part of a larger system and cannot function entirely on their own. Microservices have to be integrated with other microservices.
- The independent deployment is a feature that creates numerous advantages (https://www.educative.io/collection/page/10370001/6518081205567488/4998953437233152) and is therefore very important. Thus, the definition, in spite of its brevity, explains what the most *essential feature* of a microservice really is.

Deployment monolith

A system that is not made up of microservices can only be deployed in its entirety. Therefore, it is called a *deployment monolith*. Of course, a deployment monolith can be divided into modules. The term *deployment monolith* does not make a statement about the internal structure of the system.

Size of a microservice

The above definition of microservices does not say anything about the size of a microservice. Of course, the term *microservice* suggests that especially small services are meant. However, in practice, **microservices can vary hugely in size**. Some microservices keep an entire team busy, while others comprise only a few hundred lines of code. Thus, the size of microservices is ill-suited to be part of the definition.

Chapter walkthrough

This chapter introduces *microservices* and discusses:

- Advantages (https://www.educative.io/collection/page/10370001/6518081205567488/4998953437233152) and disadvantages (https://www.educative.io/collection/page/10370001/6518081205567488/4532272759832576) of microservices to enable the reader to evaluate the applicability and usefulness of this architecture for a specific project.
- The discussion of benefits explains which problems microservices can solve and how this architecture can be adapted for different scenarios.
- The discussion of disadvantages illustrates where technical challenges and risks lie and how these can be addressed.
- Recognizing advantages and disadvantages is critical for technology and architecture decisions since those have to be aimed at maximizing benefits and reducing disadvantages.

Q U I

Z

1 A microservice should not be any longer than a few hundred lines of code.

○ A) True

○ B) False

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the next lesson, let’s discuss the advantages of using microservices.

Next →

Advantages

☑ Mark as Completed

Report an Issue