Introduction

In this lesson, we'll look at a quick introduction to self-contained systems.

WE'LL COVER THE FOLLOWING

Definition

A **self-contained system (SCS)** is a type of microservice that **specifies elements of a macro architecture**. SCSs do not represent a complete macro architecture. The area of operation, for instance, is completely missing.

The idea behind SCS is to provide microservices that are self-contained, providing everything needed for the implementation of a part of the domain logic.

This means an SCS includes **logic, data, and a UI**. That also means if a change impacts all technical layers, it can still be contained in one SCS, making it easier to perform the change and put it into production.

So, an SCS for a microservice **payment** would store all information relevant to payment – that is, it would implement a **bounded context**. But it would **also implement the UI** – web pages to show the payment history or make a payment. **Data** about customers or ordered items would need to be replicated from other SCSs.

Reasons for the term self-contained systems

There is no uniform definition for microservices. **Self-contained systems, however, are precisely defined**. You can read the definition of SCSs on the

http://scs-architecture.org website. The content of the site is provided under a creative commons license, so that the material on the site may be reused by anyone if the source is named, and the materials are distributed under the same license terms.

The content of the website is available as source code at https://github.com/innoq/SCS so that everyone can make changes. The website contains links to articles about experiences with SCSs from different projects and companies.

> **Self-contained systems** are best practices that have proven their usefulness in various projects.

While microservices do not provide many rules about how systems should be built, SCSs have precise rules based on proven patterns. Thus, SCSs give a point of reference as to how microservices architecture can look.

Although SCSs are a collection of best practices, the SCS approach is not the best architecture for every situation. Therefore, it is important to understand the reasons for the SCS rules. By doing so, teams can choose variations of this approach or even completely different approaches that might be better adapted to the respective project.

Chapter walkthrough

This chapter covers the following:

- First, it describes the reasons for using SCSs.
- SCSs determine different macro architecture decisions. This chapter explains the reasons for these macro architecture decisions and discusses the advantages to each decision.
- SCSs are a variation of microservices. This chapter addresses the differences between the terms “SCS” and “microservice.”
- This chapter then discusses the benefits of the SCS approach.
- Finally, challenges connected to the development of an SCS system are described and potential solutions are discussed.

described and potential solutions are discussed.

SCSs include a UI and focus on UI integration. Therefore, SCSs are a good motivation for the UI integration techniques that the next chapters explain.

QUIZ

1

Suppose a microservice contains the data and logic but the UI is implemented from another microservice. Is this microservice an SCS?

COMPLETED 0%

1 of 2

In the next lesson, we'll discuss a more thorough definition of self-contained systems.