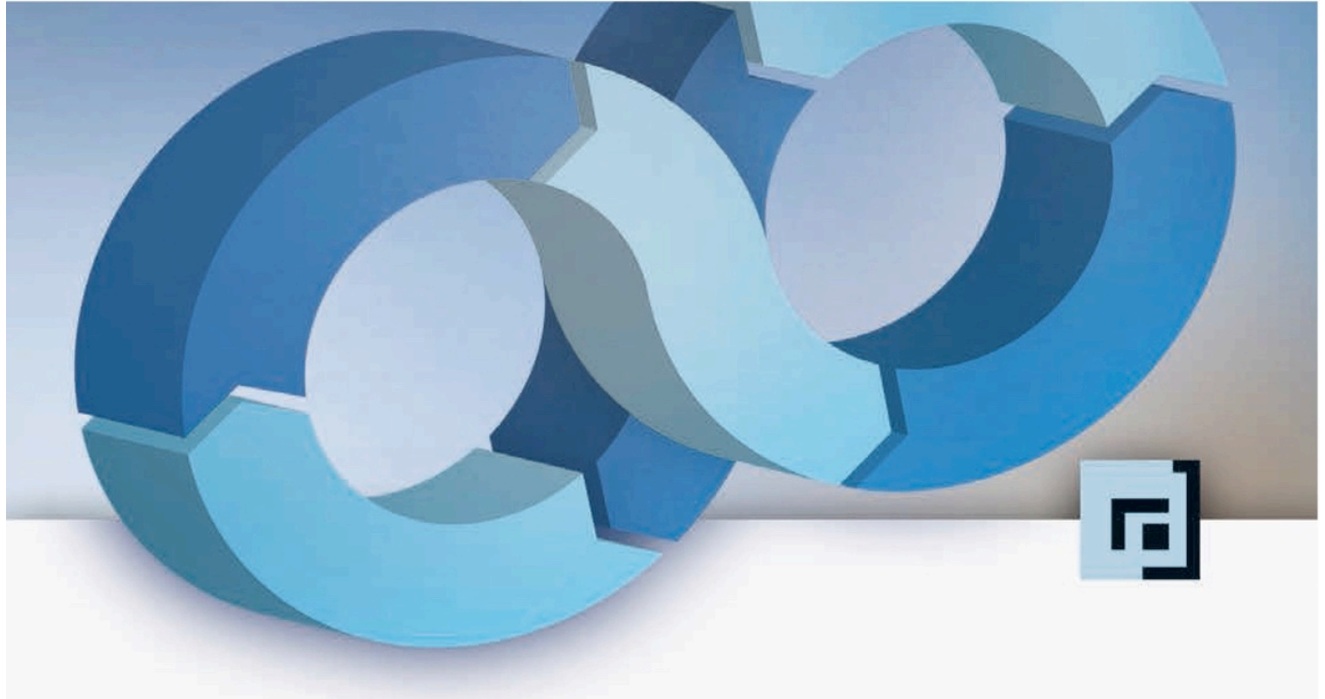
Self-contained Systems: A Different Approach to Microservices



Eberhard Wolff

Continuous Delivery

Der pragmatische Einstieg

dpunkt.verlag

Eberhard Wolff

Microservices

Grundlagen flexibler Softwarearchitekturen

dpunkt.verlag

http://microservices-buch.de/

Microservices

Flexible Software Architectures

Eberhard Wolff

http://microservices-book.com/

Eberhard Wolff

Microservices Primer

A Short Overview

FREE!!!!

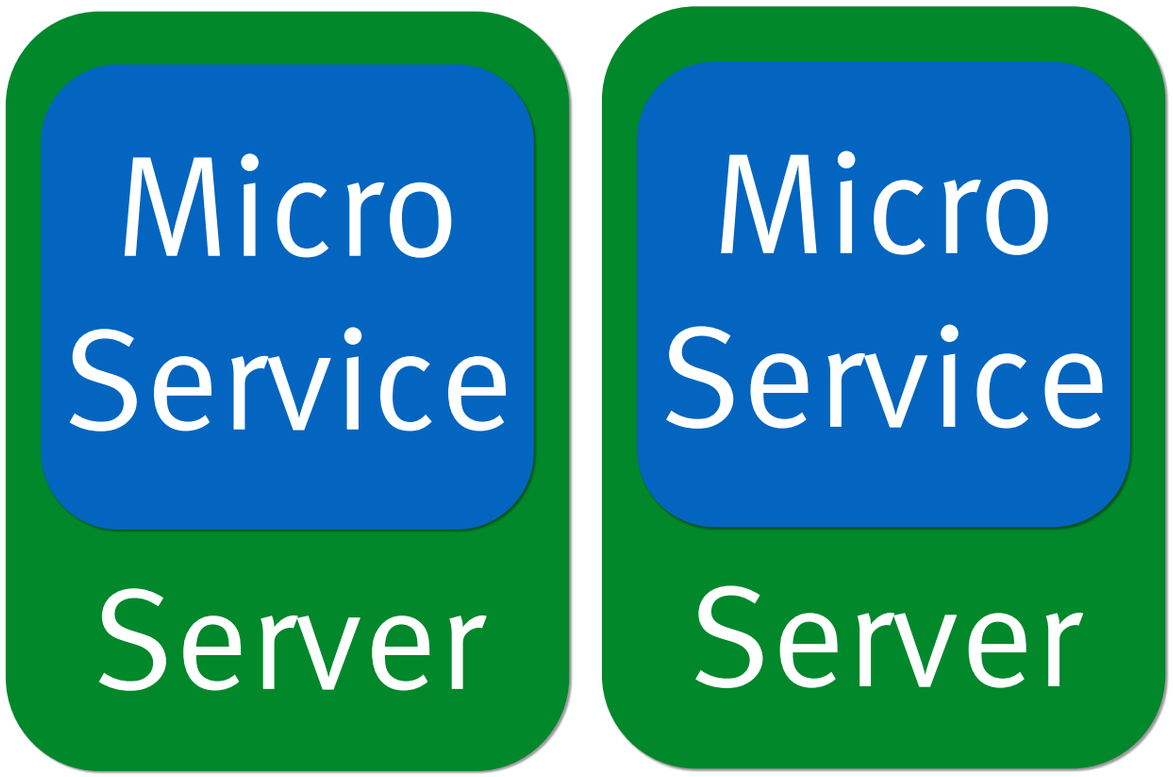
innoQ

http://microservices-book.com/primer.html

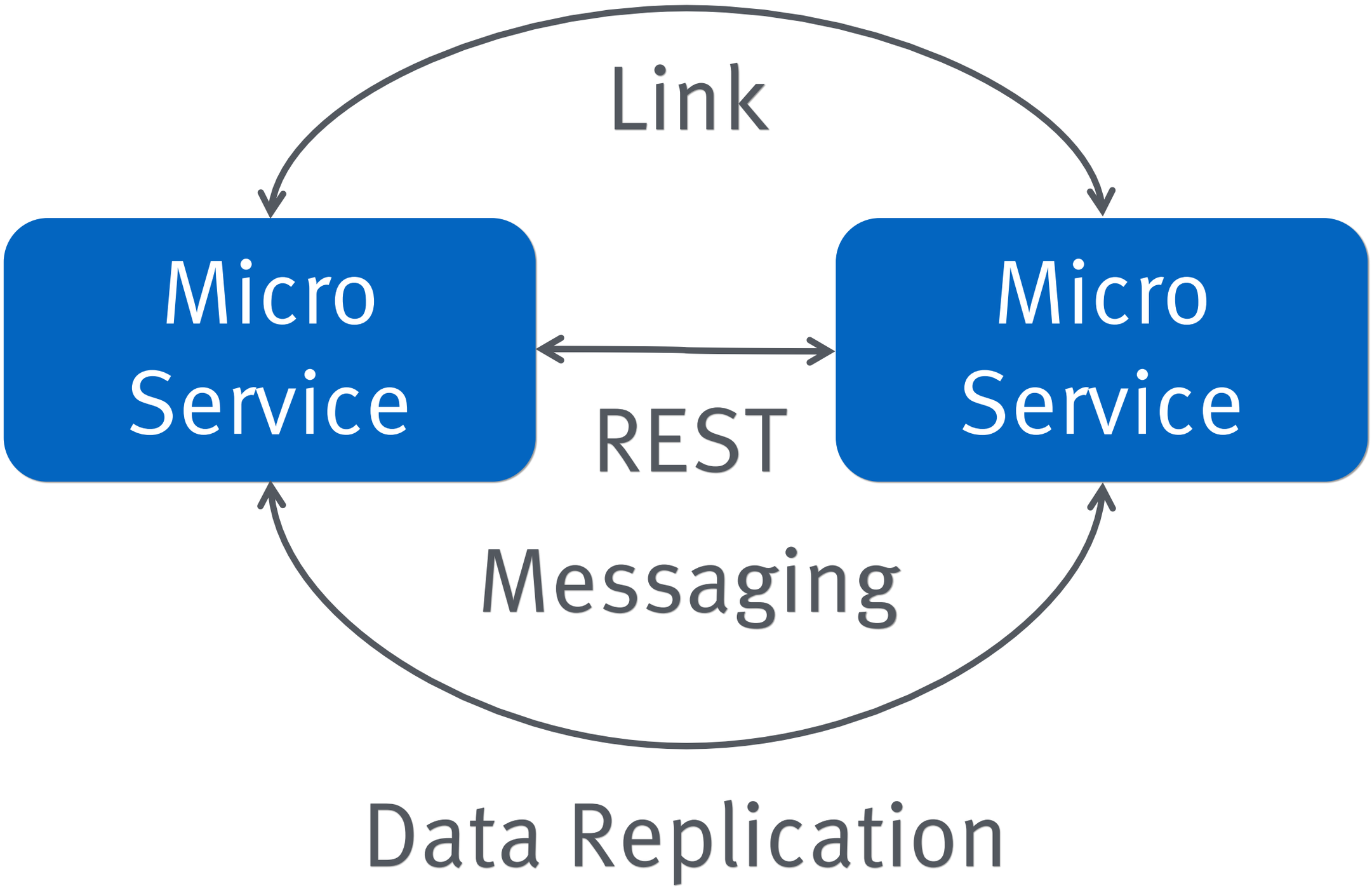
Microservice Definition

Microservices: Definition

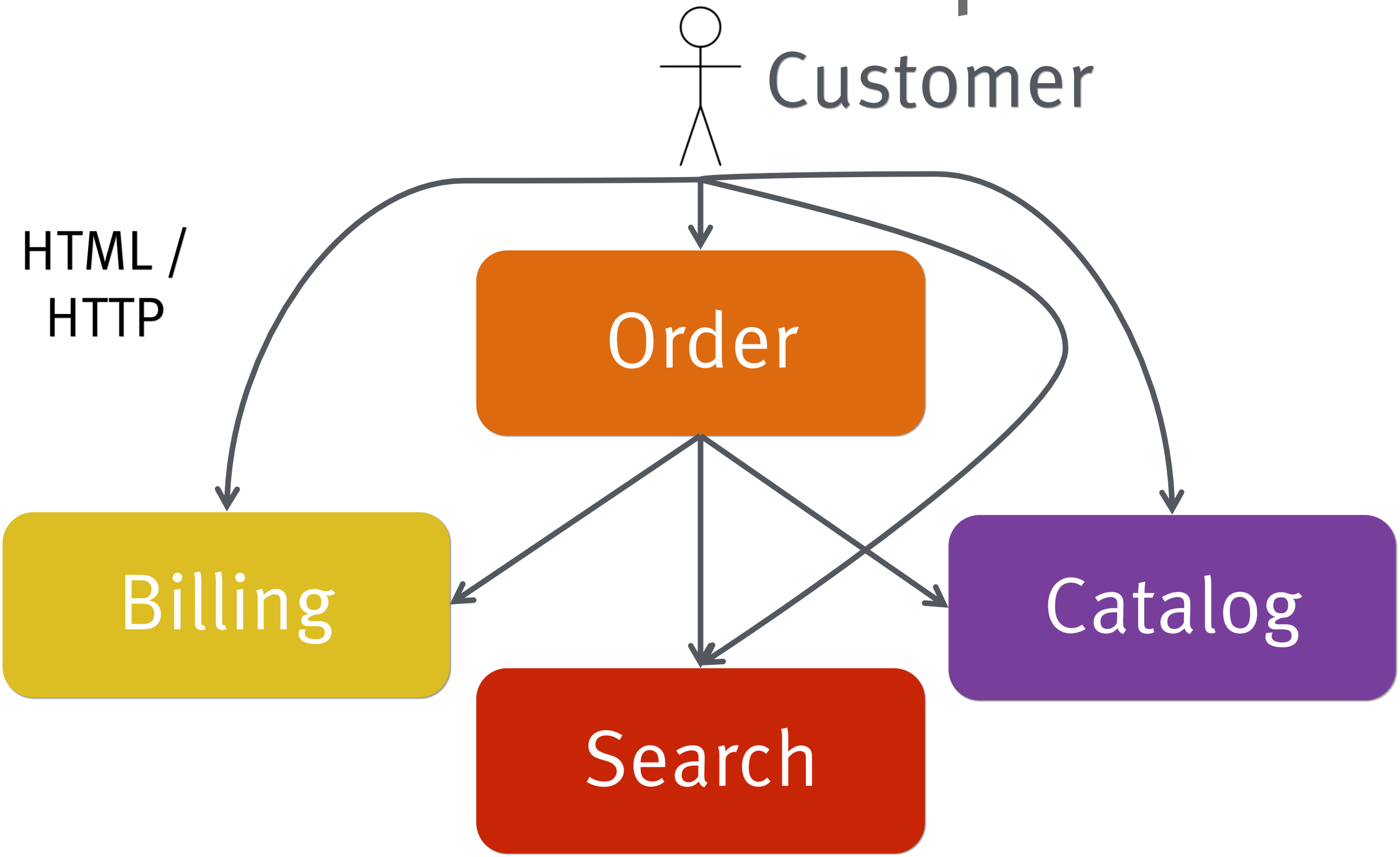
- Independent deployment units
- E.g. process, VMs, Docker containers
- Any technology
- Any infrastructure



Components Collaborate



Online Shop



Distributed System

Distributed System
Why??

Why Microservices?

Scaling Agile

Small teams develop and deploy independently

Handle Legacy efficient

Add services – not code

Sustainable development

Strong Modularization

Replaceable Services

Continuous Delivery

Small Services

Robustness

Failure limited to single Microservice

Independent Scaling

Free choice of technology

Why Microservices?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Deployment

Continuous Delivery

Units

Robustness

Independent Scaling

Technology

Free choice of technology

Single Developer

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Continuous Delivery

Deployment Units

Robustness

Independent Scaling

Technology

Free choice of technology

Replace Monolith

Scaling Agile

Handle Legacy efficient

Sustainable development

Continuous Delivery

Organization

Deployment Units

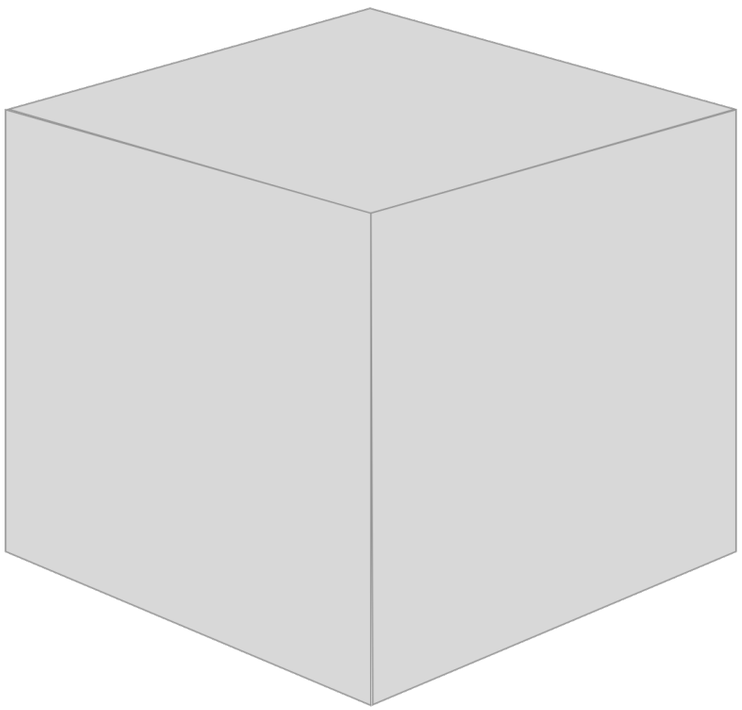
Robustness

Independent Scaling

Free choice of technology

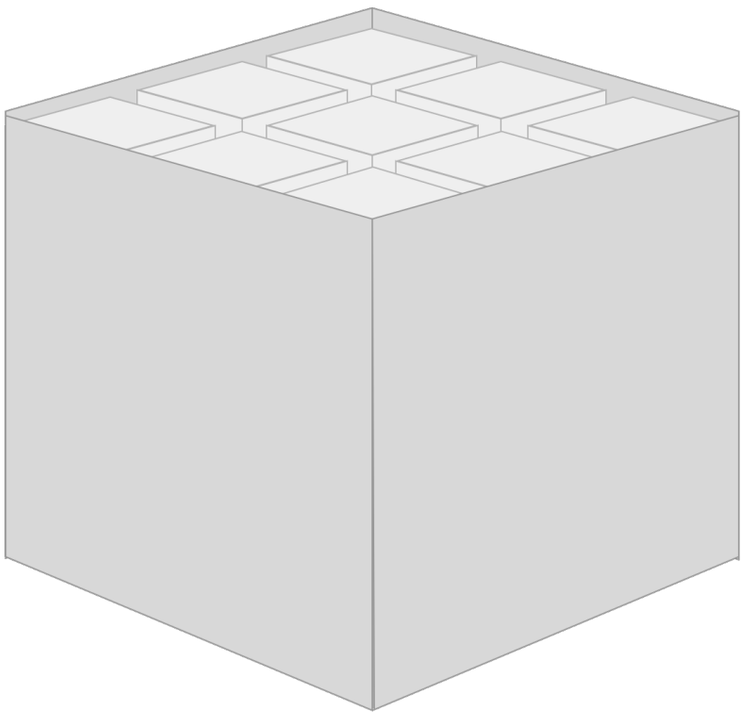
Technology

Self-Contained System (SCS)

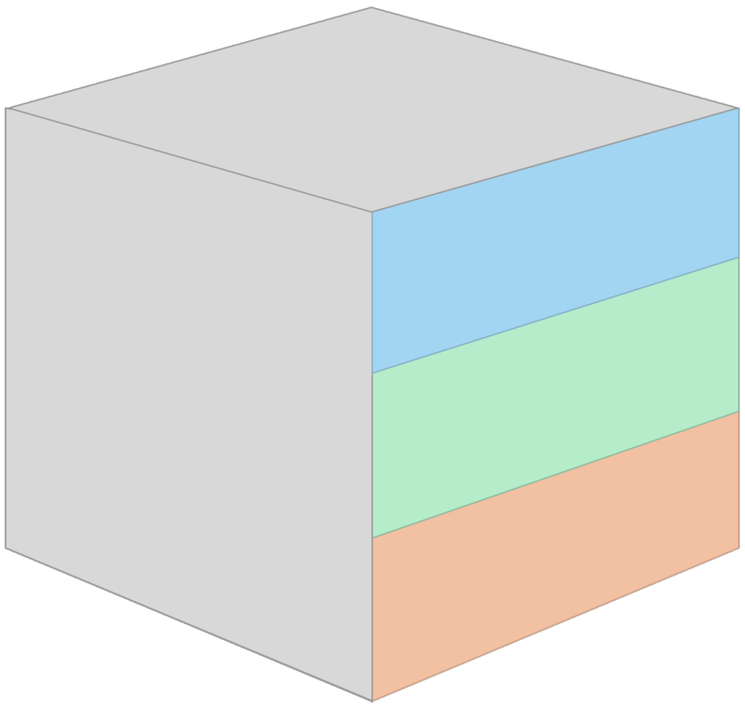


Deployment monolith

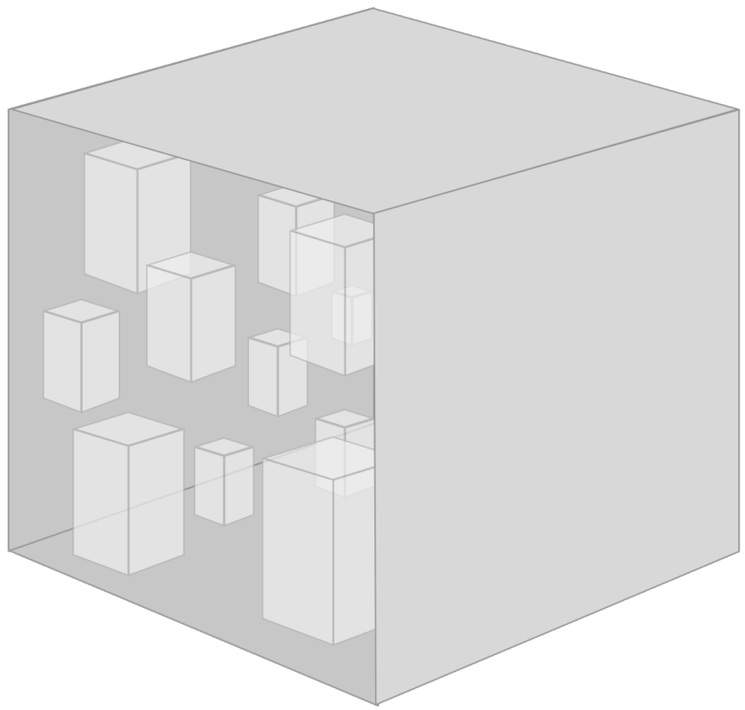
Graphics by Roman Stranghöhner, innoQ
http://scs-architecture.org



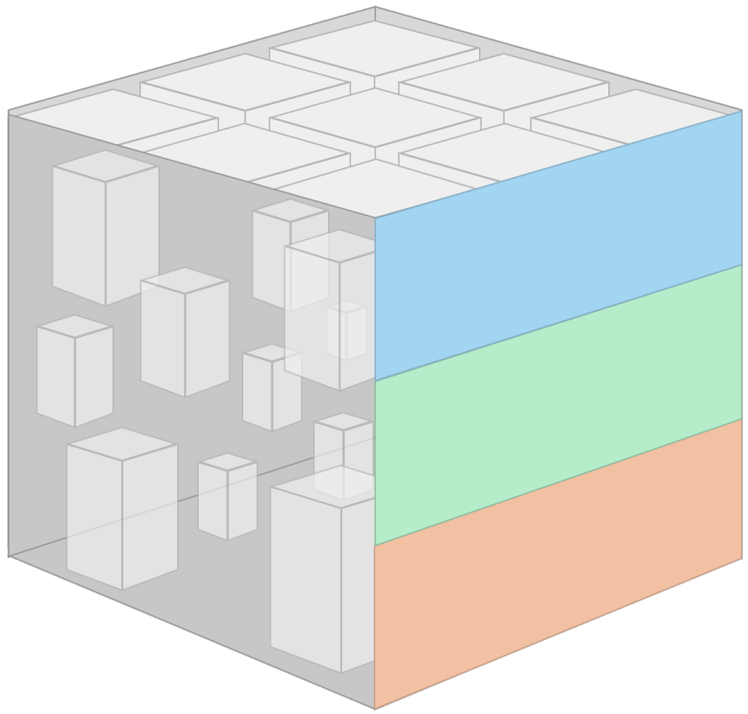
Various Domains



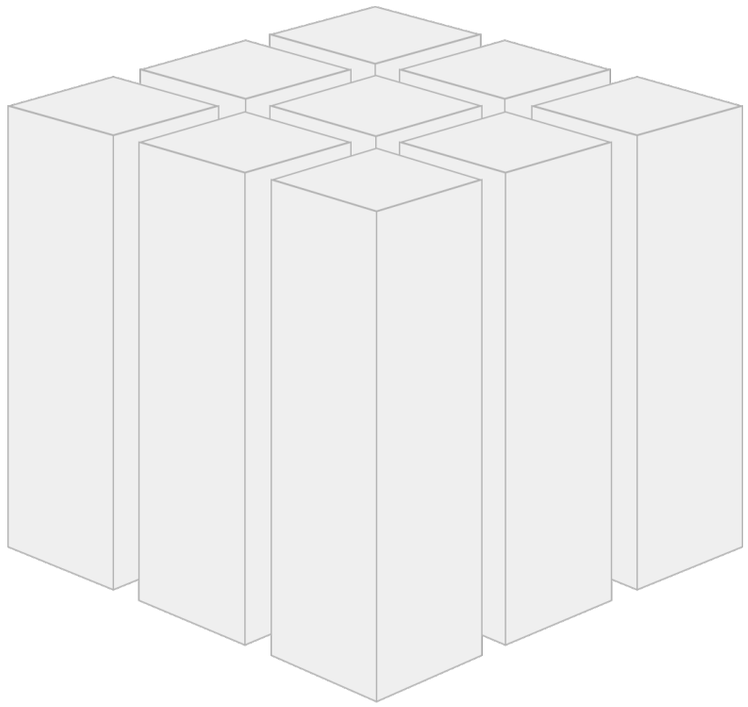
User interface
Business logic
Persistence



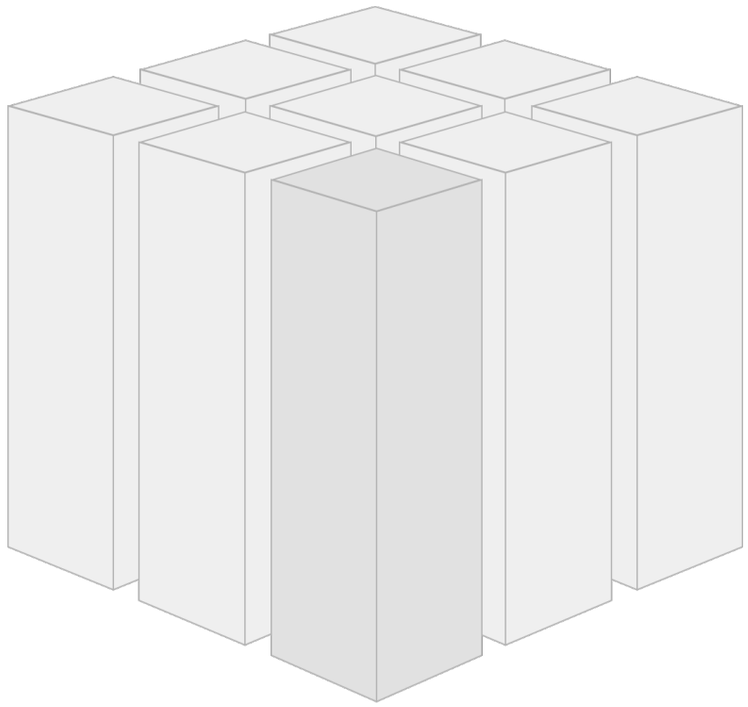
... **a lot** of modules, components, frameworks and libraries



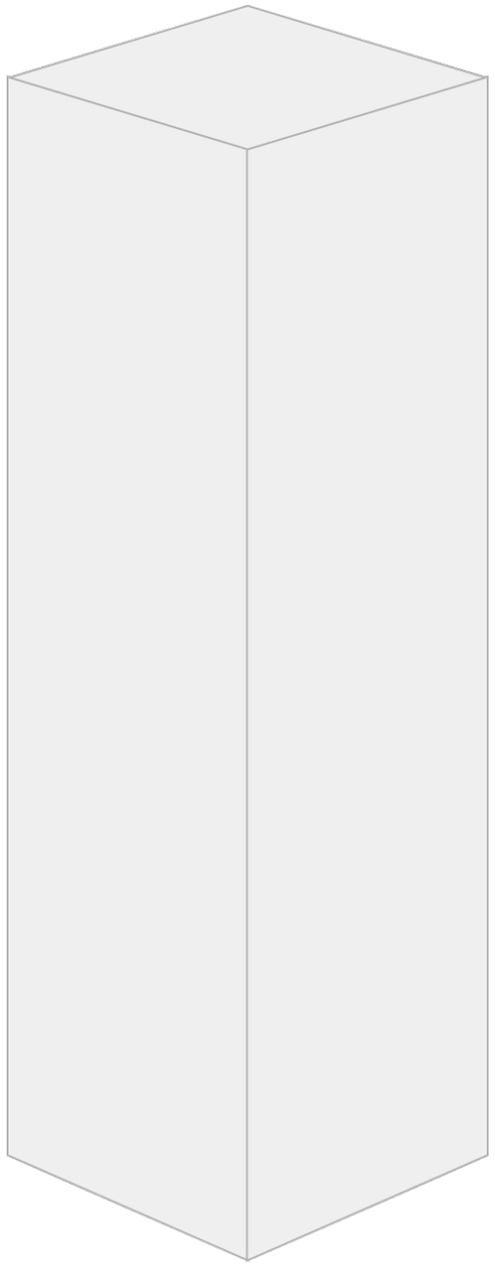
With all these layers in one place, a monolith tends to **grow**.



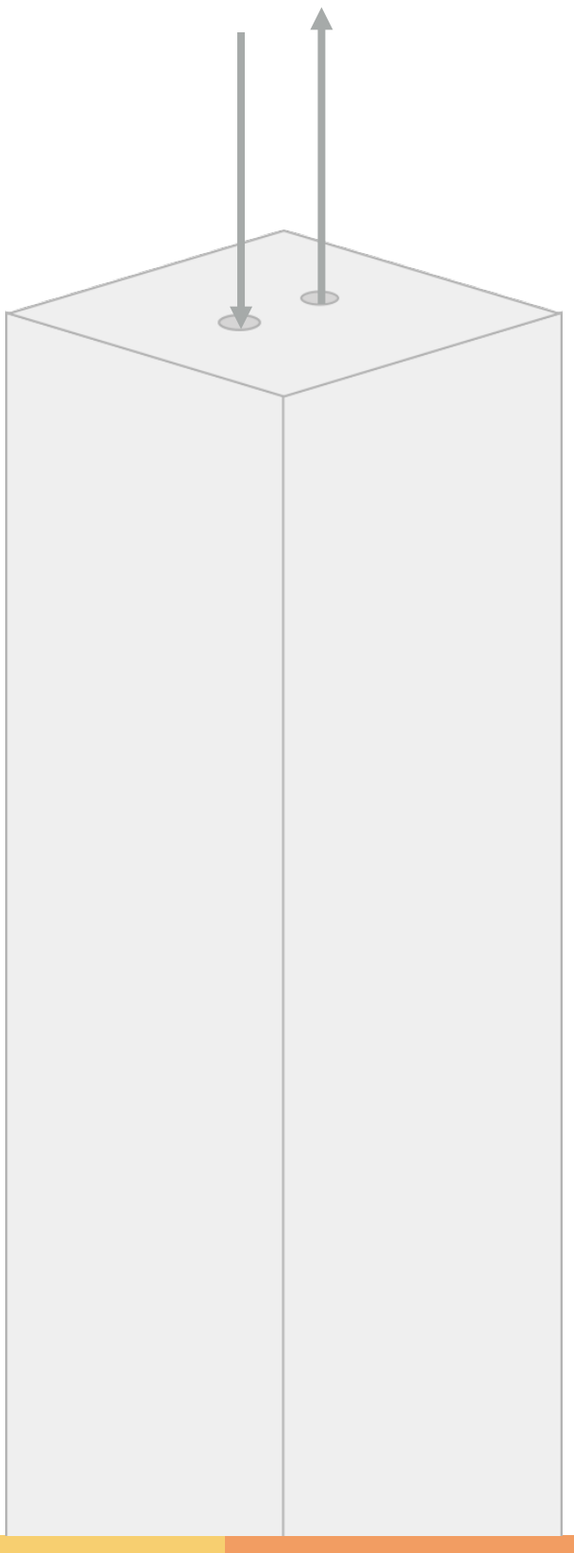
Cut Deployment monolith along domains …



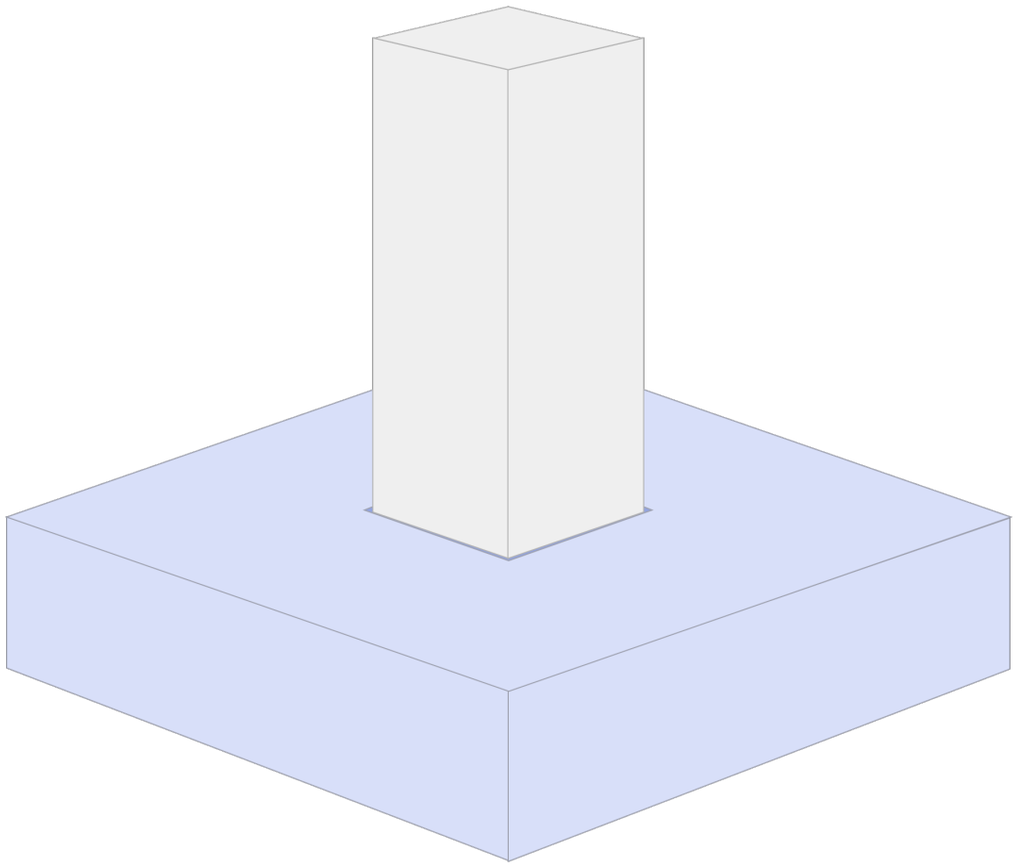
... wrap domain in **separate** web application ...



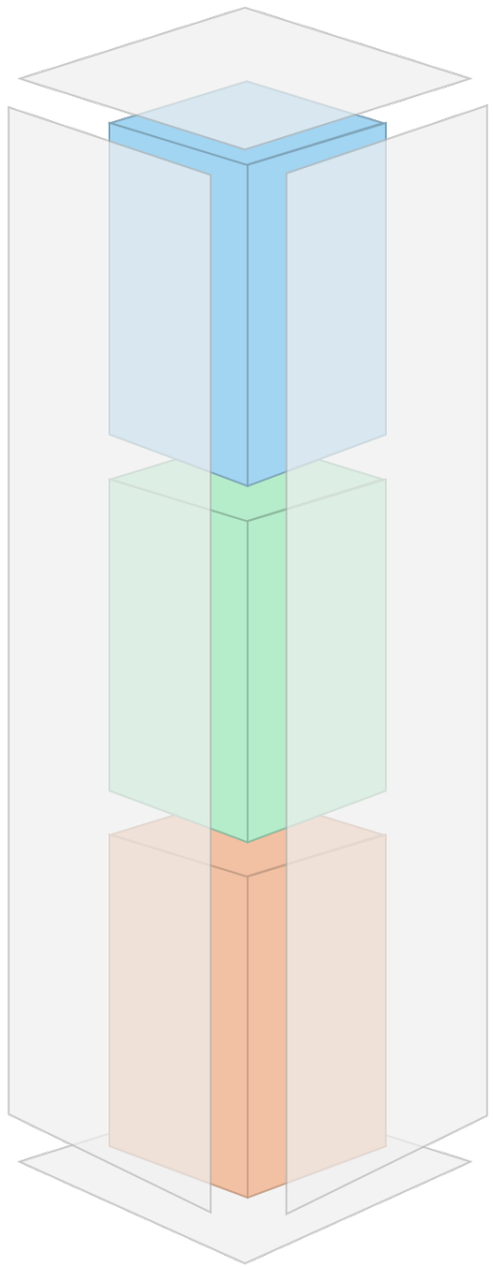
Self-contained System (SCS) – individually deployable



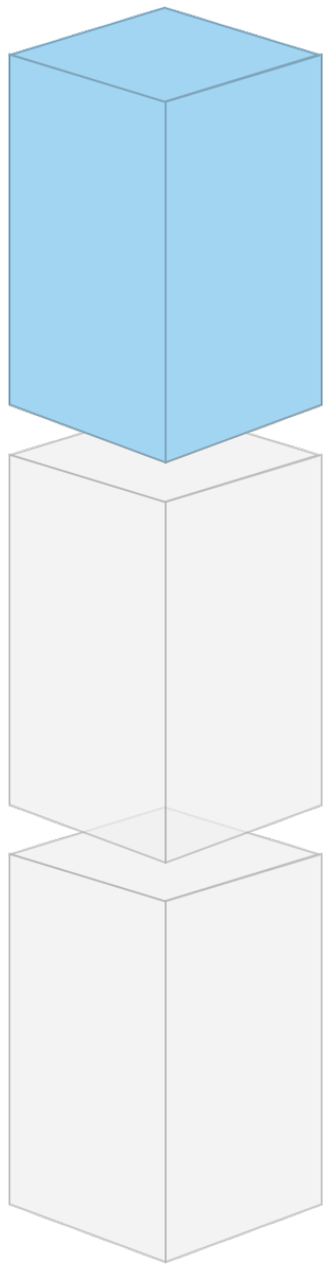
Decentralized unit communicating with other systems via RESTful HTTP or lightweight messaging.



SCS can be individually developed for **different** platforms.

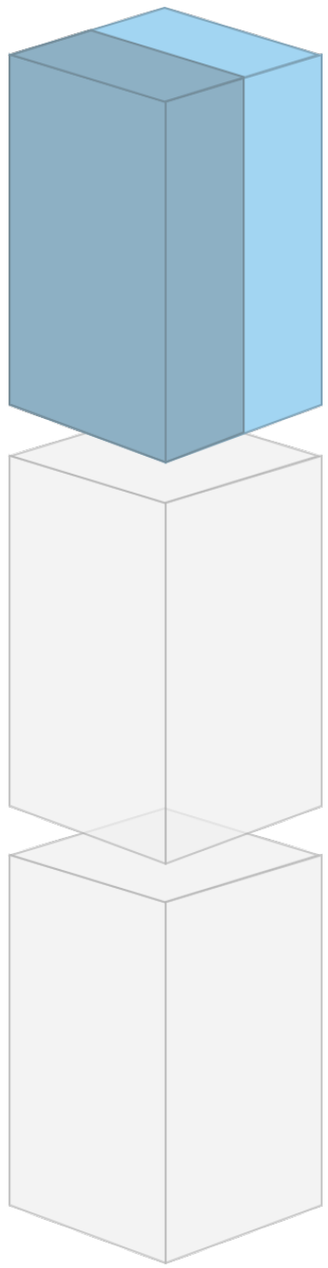


An SCS contains its own user interface, specific business logic and separate data storage

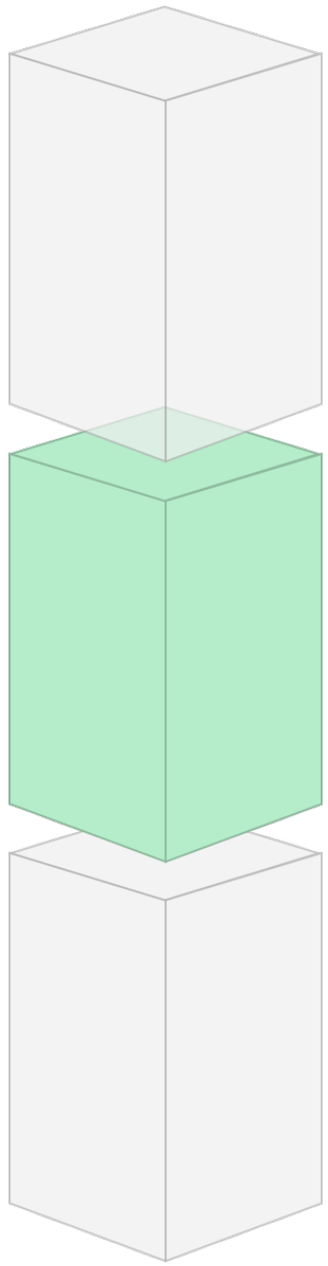


Web user interface composed according to **ROCA principles**.

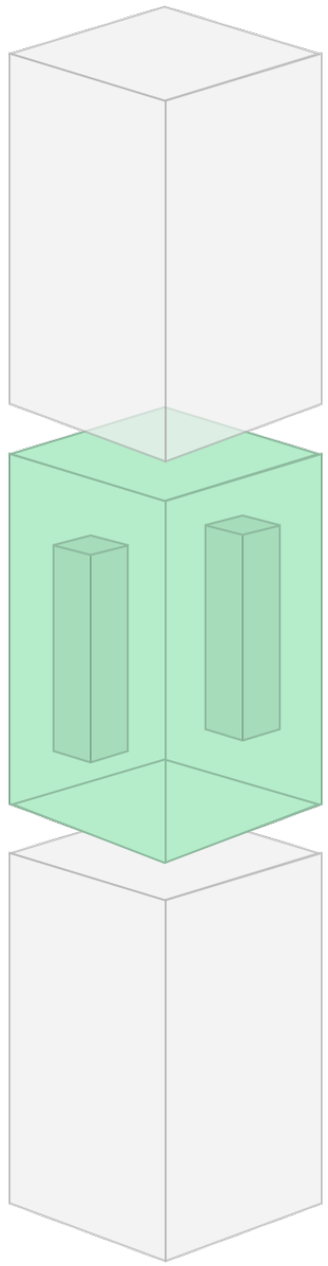
http://roca-style.org



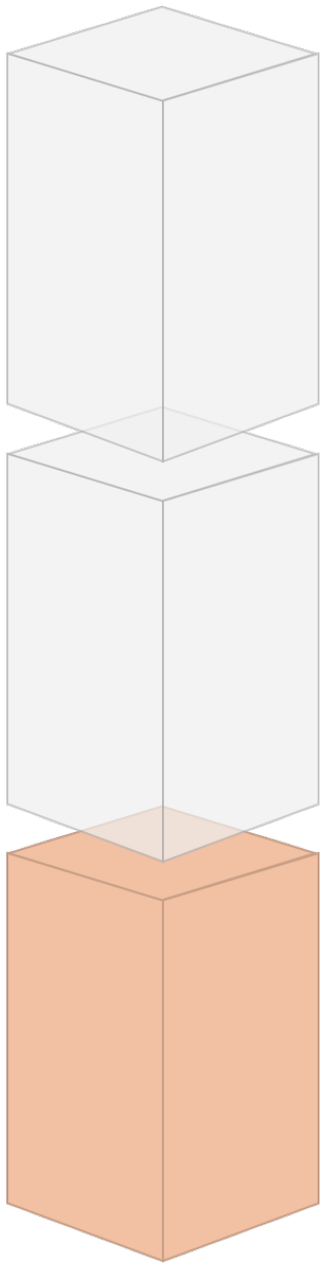
optional API e.g. for mobile



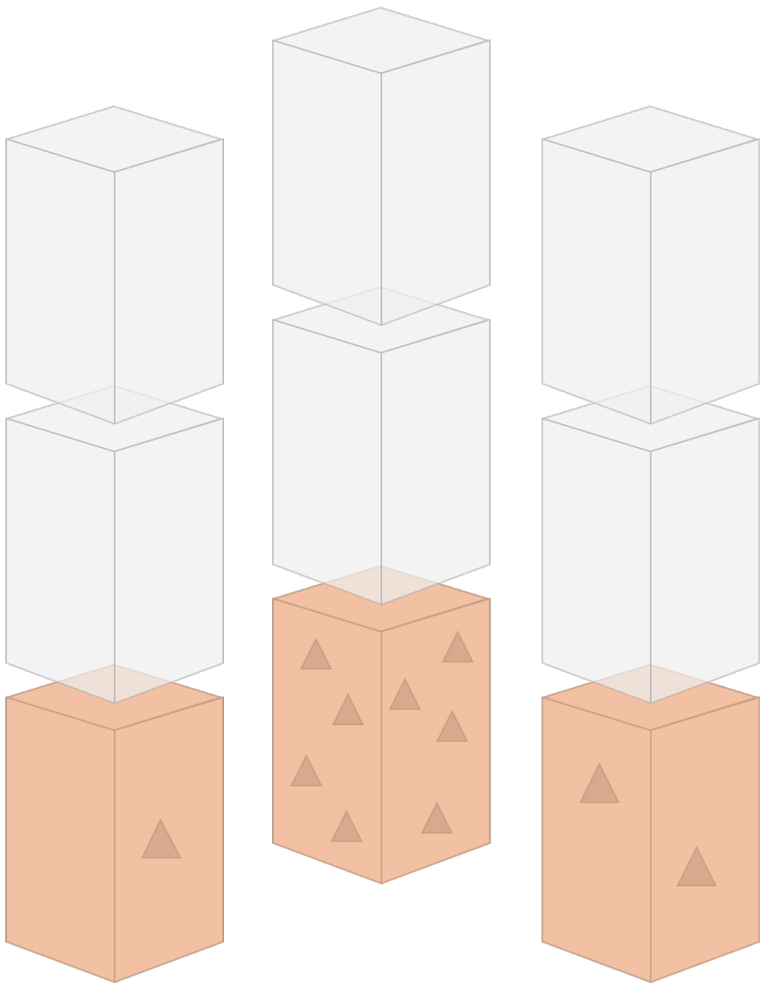
Logic only shared over a **well defined** interface.



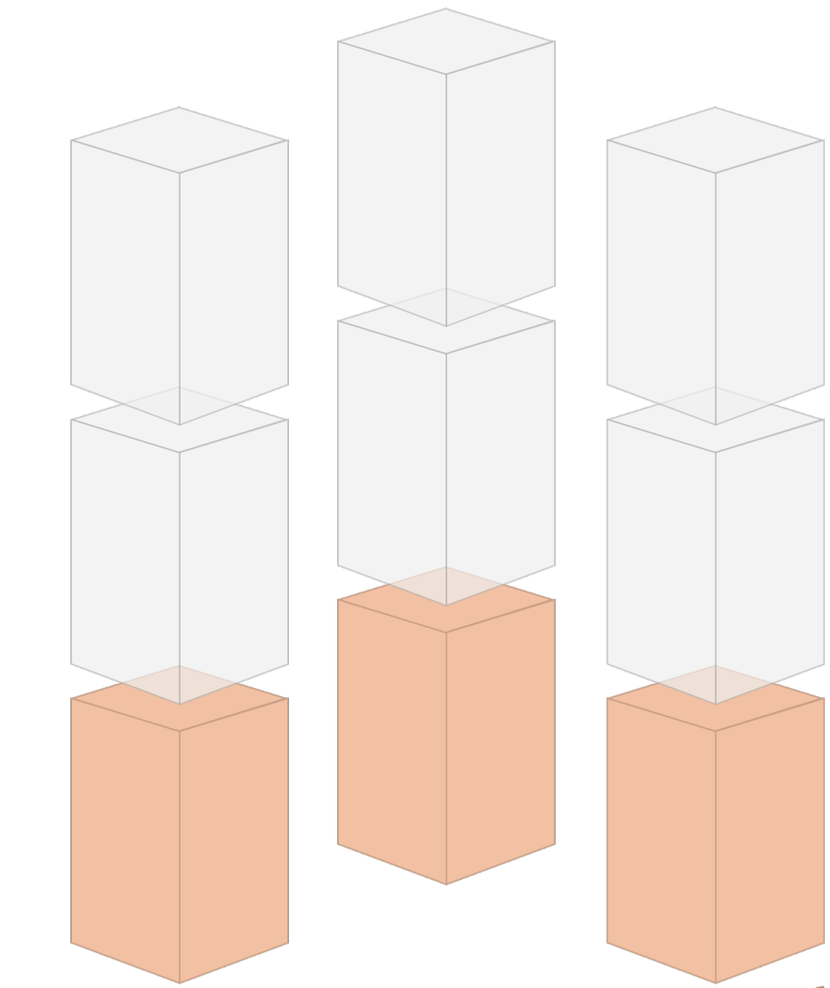
Business logic can consist of microservices



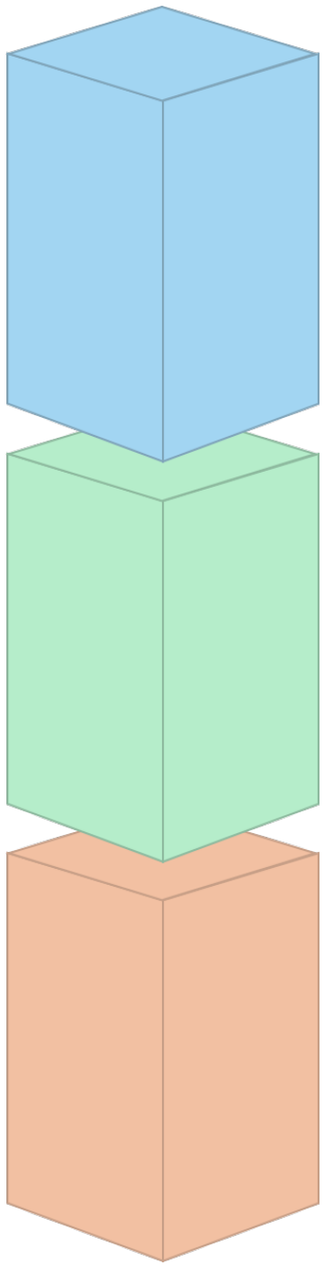
Every SCS brings its **own data storage** with ist own (potentially redundant) data



Redundancies: tolerable as long as sovereignty of data by owning system is not undermined.

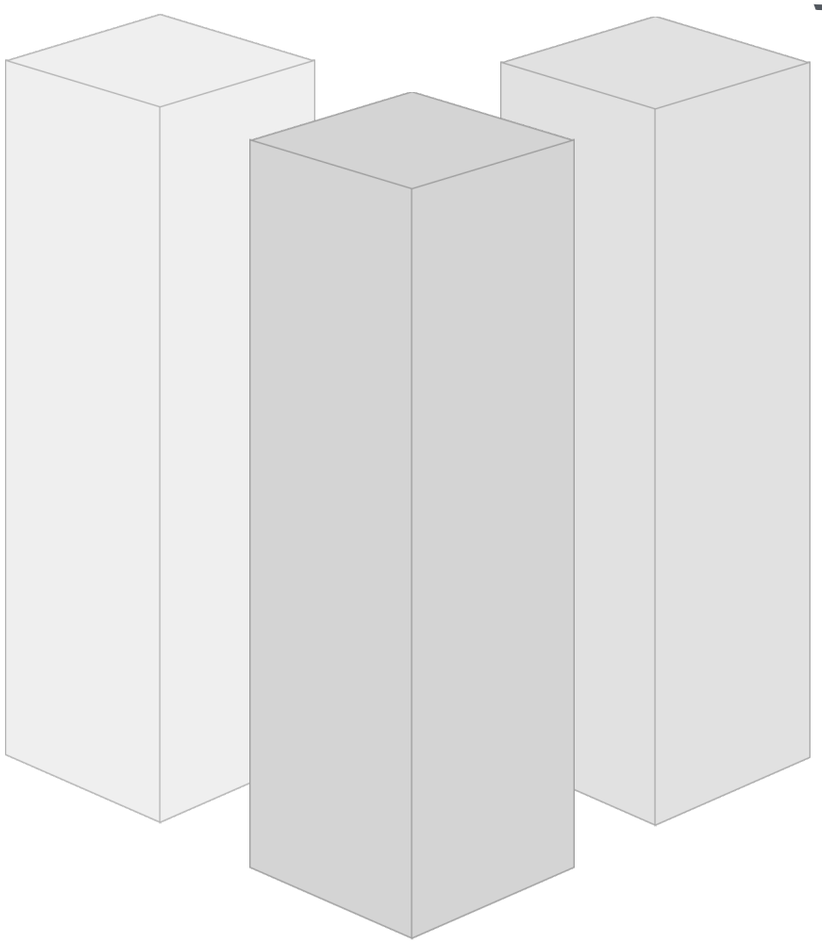


Enables **polyglot** persistence



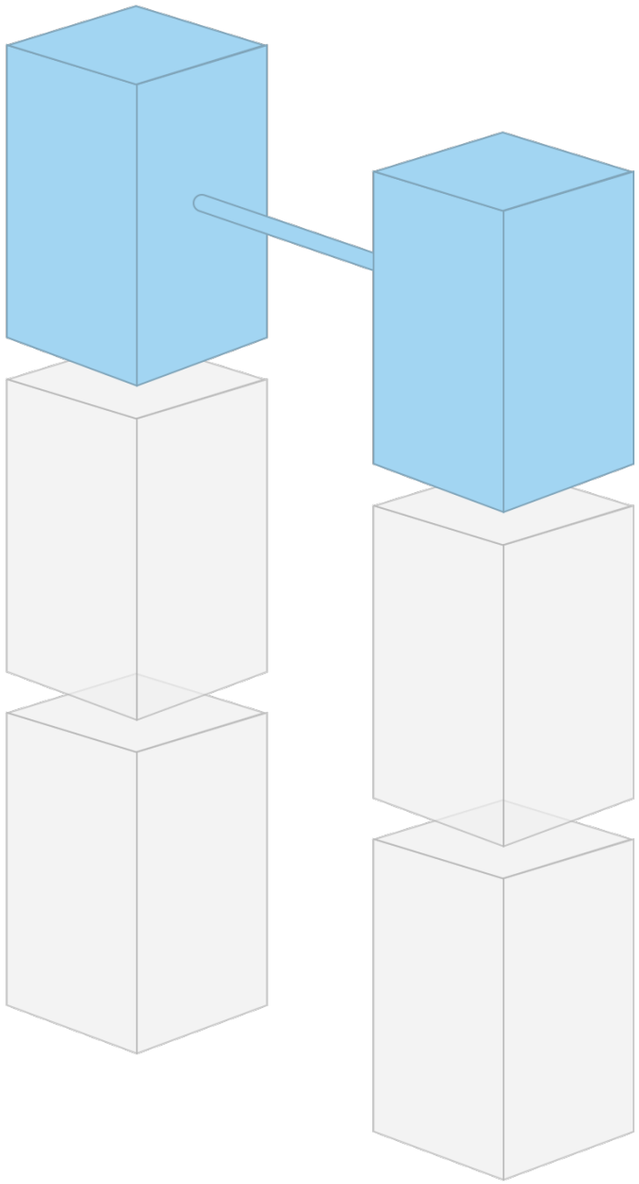
Technical decisions can be made independently from other systems (programming language, frameworks, tooling, platform)

Team 2 Team 3

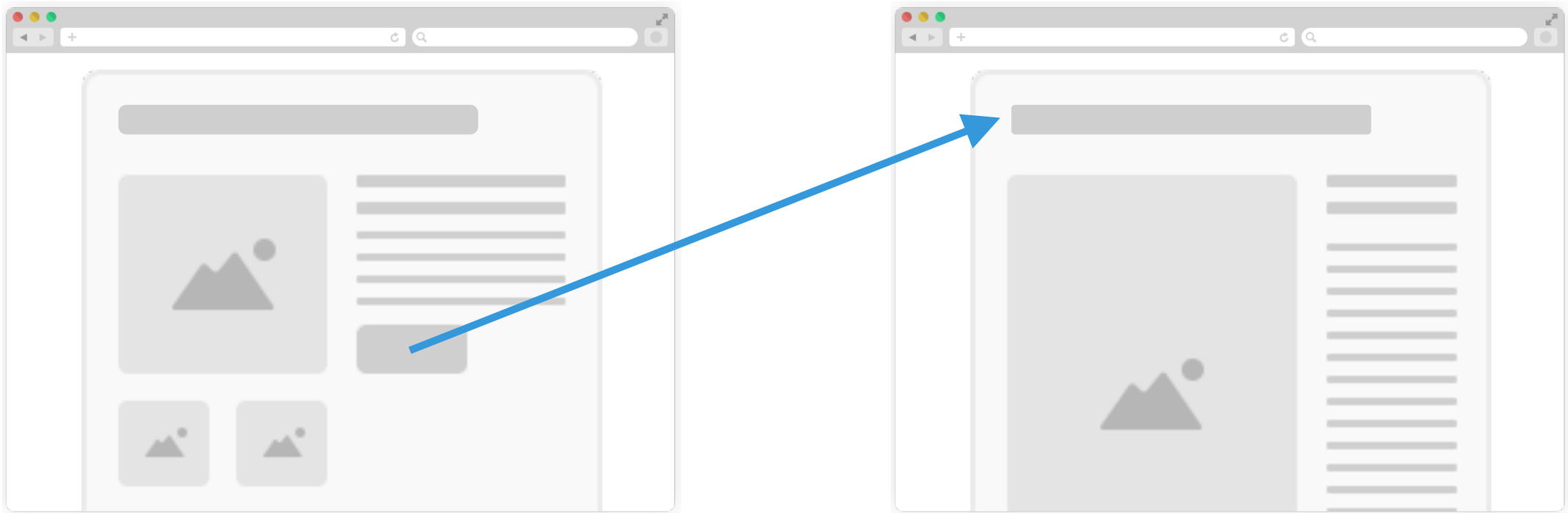


Team 1

Domain scope enables development, operation and maintenance of SCS by a **single team**.



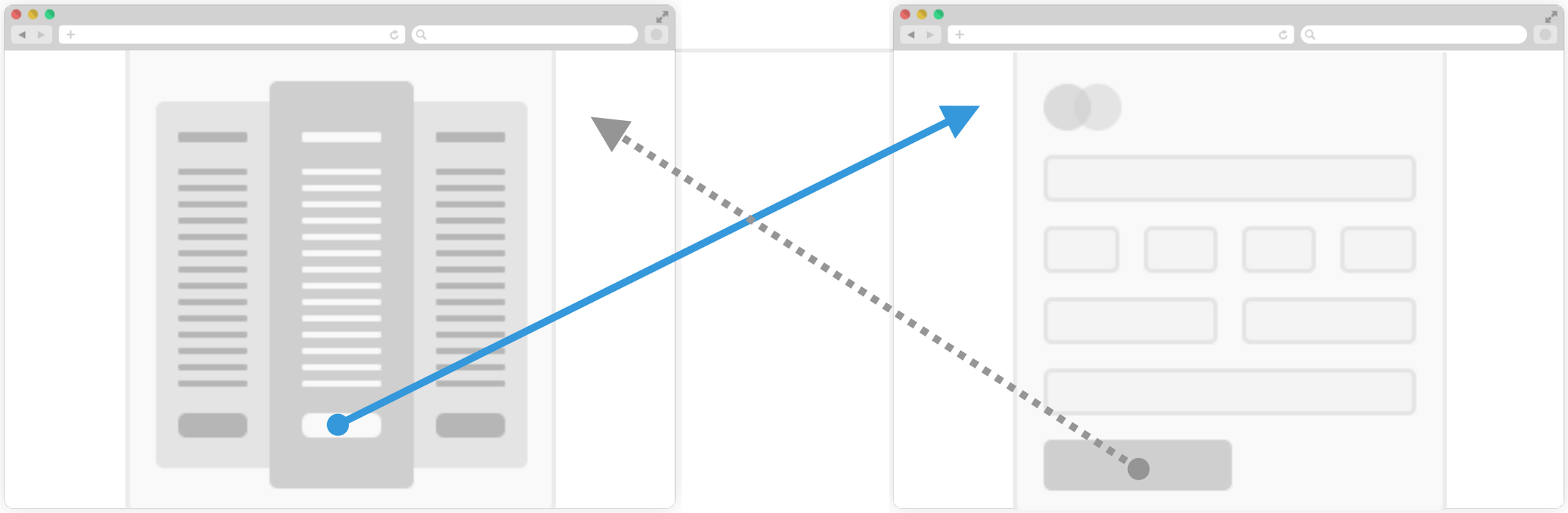
Self-contained Systems should be integrated in the **web interface**



System 1 System 2

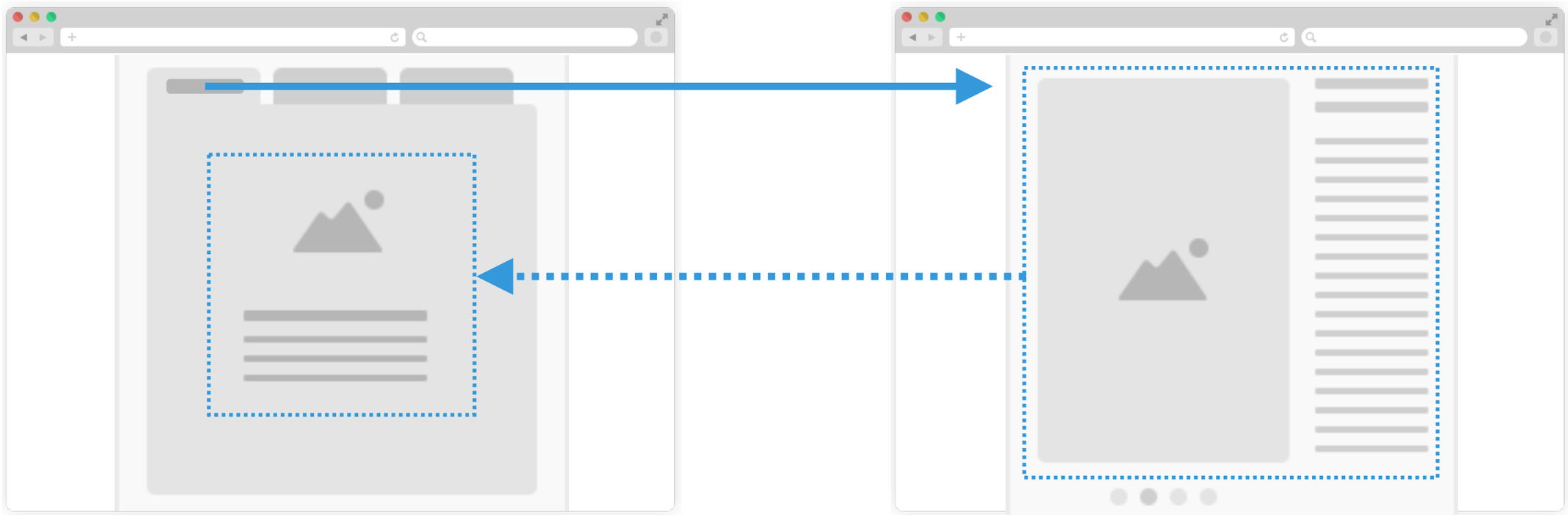
Hyperlinks to navigate between systems.

Redirection

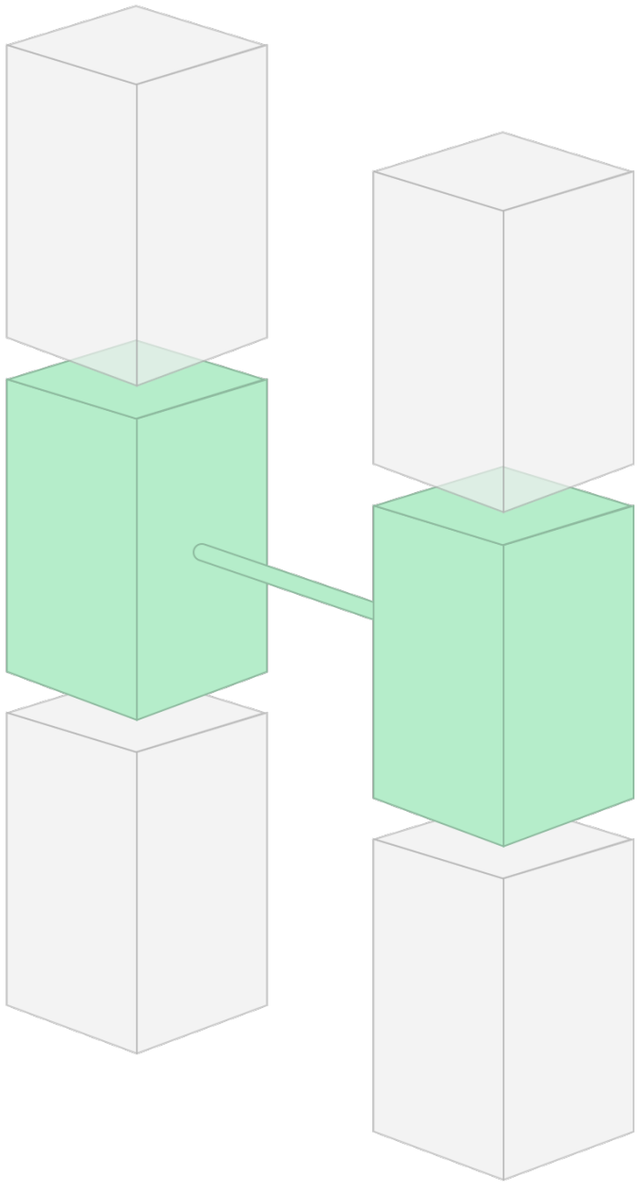


System 1 System 2

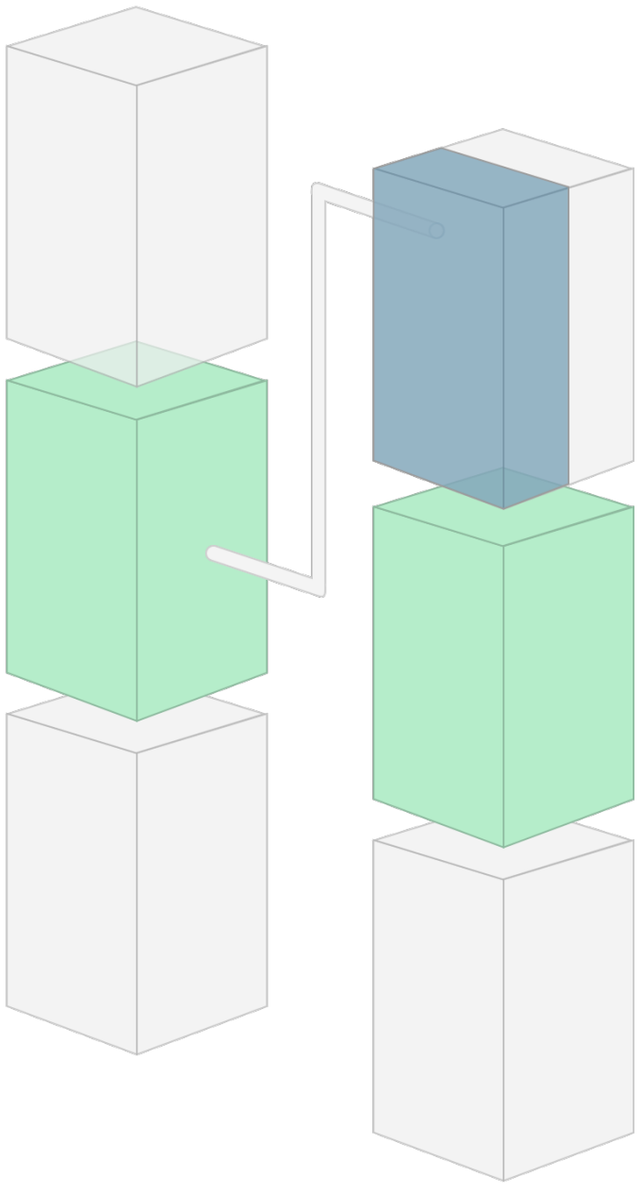
- Use of callback URIs
- As seen e.g. in OAuth flows



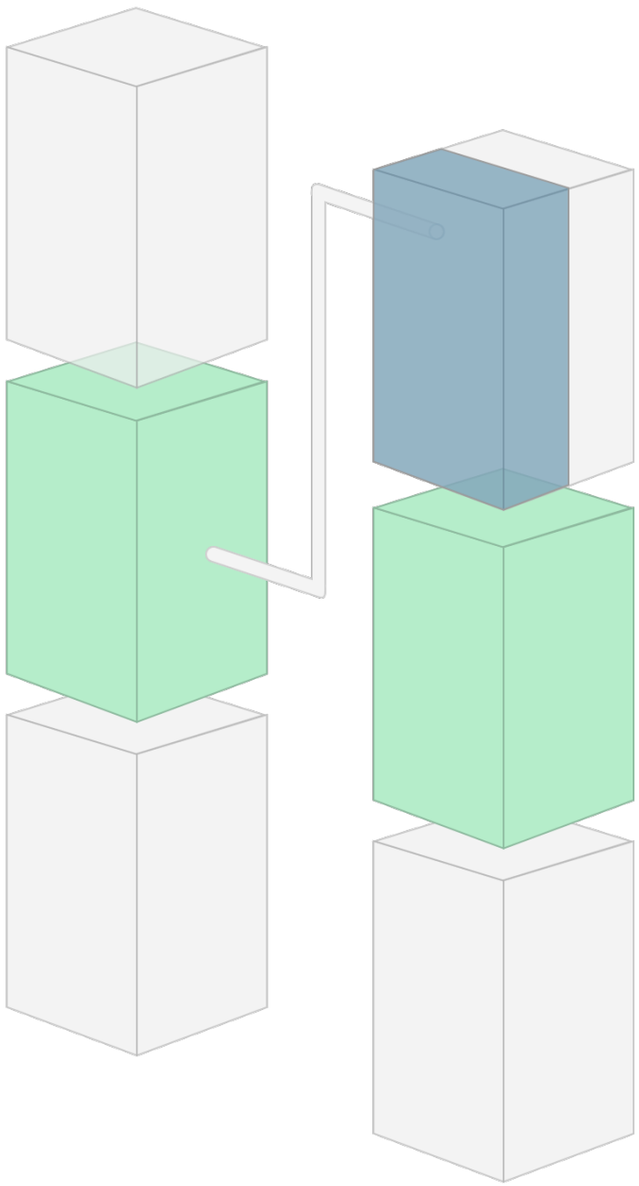
Dynamic inclusion of content served by another application



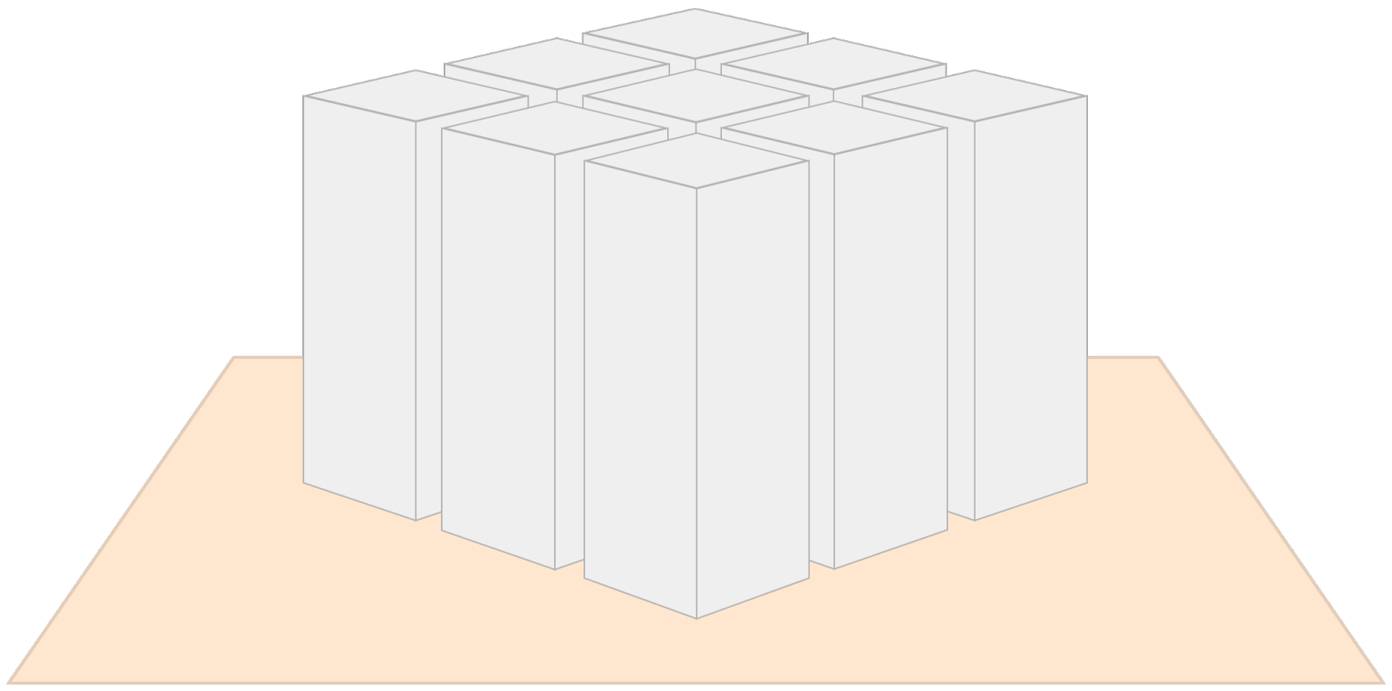
Synchronous remote calls inside the business logic should be **avoided**.



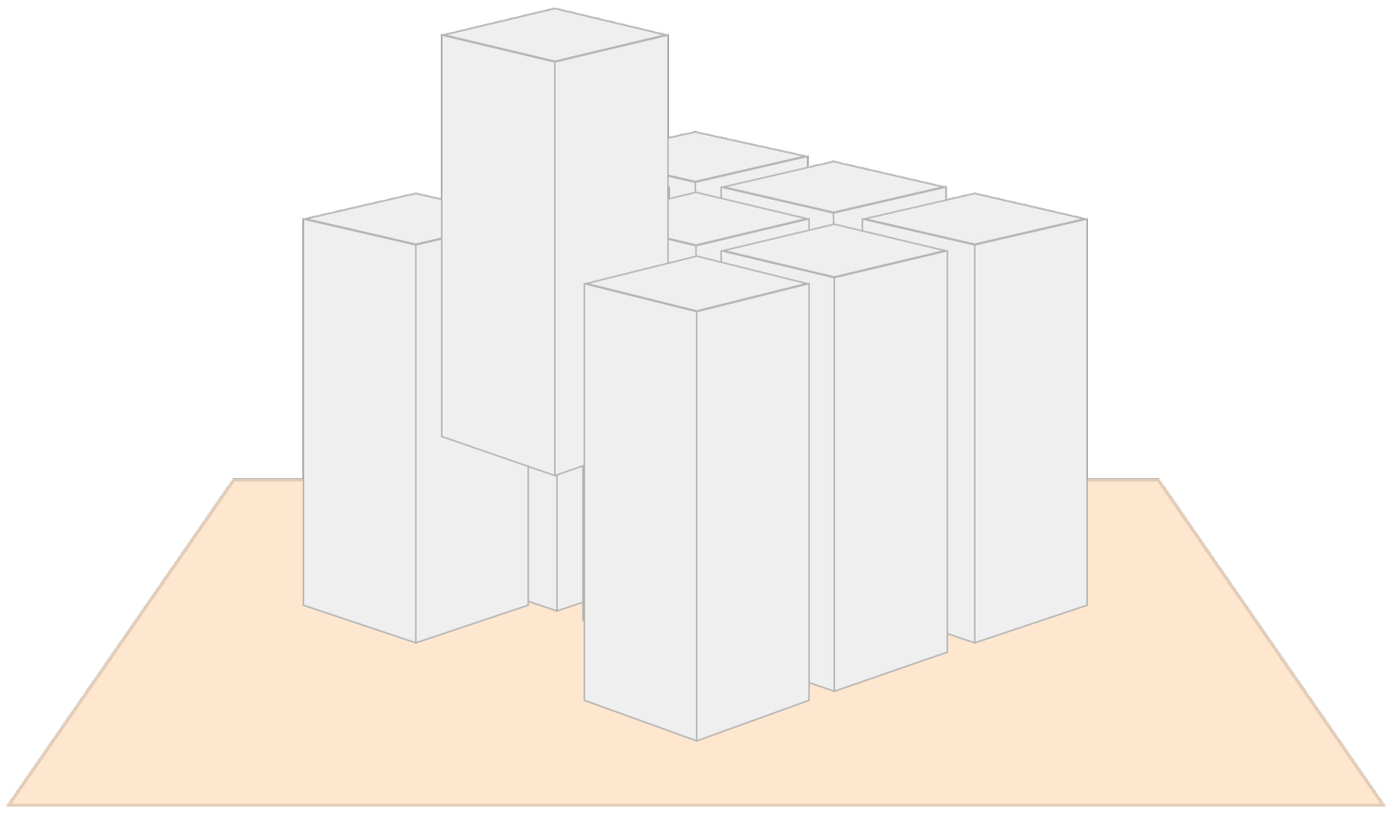
Asynchronous Remote calls reduce dependencies and prevent error cascades.



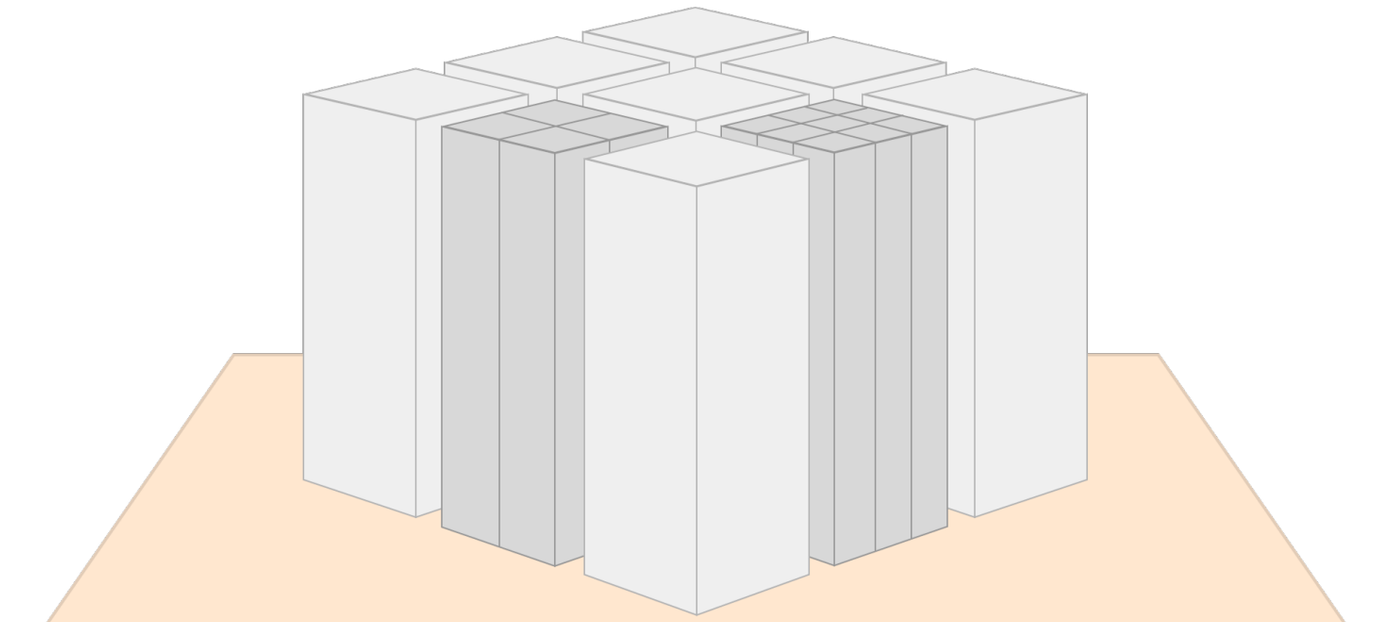
Data model's consistency guarantees are relaxed.



An integrated
system of systems
like this has many
benefits.

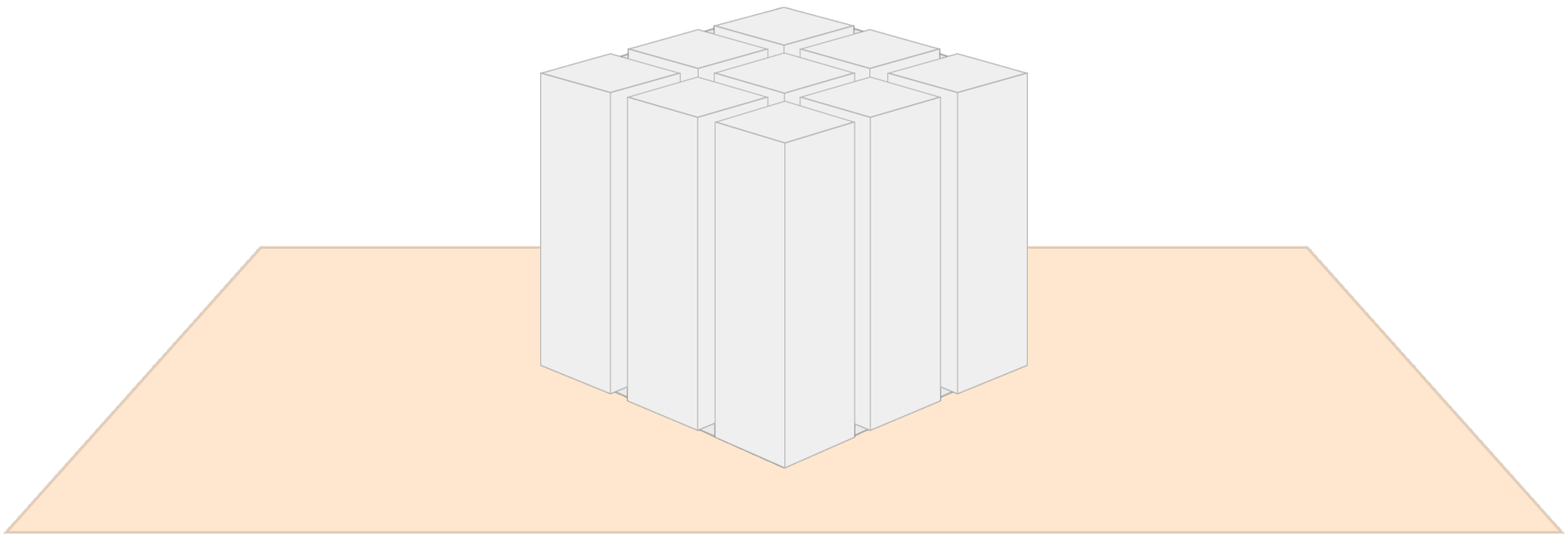


Resilience is improved through loosely coupled, replaceable systems.

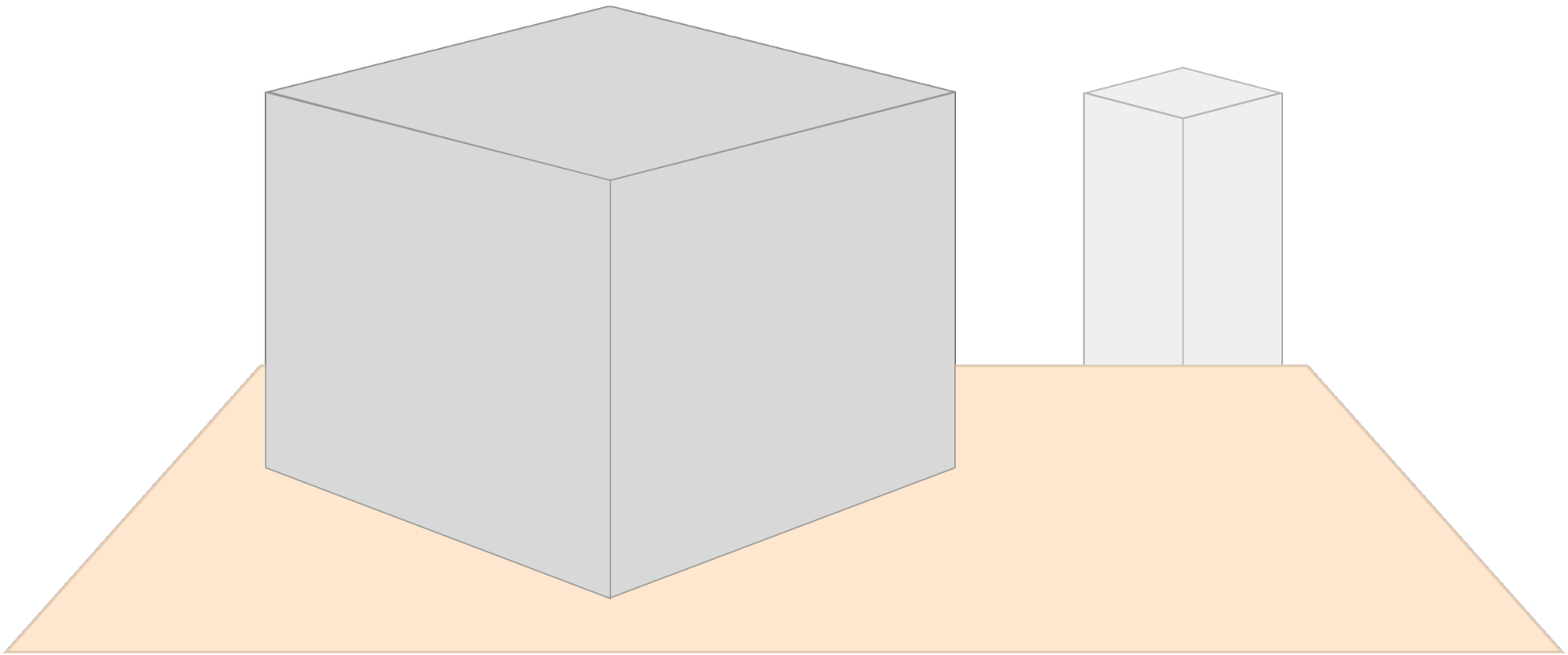


SCSs can be
individually
scaled to serve
varying demands.

Version 2



No risky **big bang** to migrate an outdated, monolithic system into a system of systems.



Migration in small, manageable steps
which minimize risk of failure and lead to
an **evolutionary modernization**
of big and complex systems.

SCS 1---* Microservice

Conclusion

- SCS: autonomouos web application
- Might consist of Microservices
- Focus on UI Integration

- Almost completet independence
- Coarse-grained architecture approach

Conclusion

- Self-contained systems are Microservices …
- that are not “micro”…
- and don’t have to be “services”
- Many are doing it already!

- http://scs-architecture.org
- Creative commons
- Source on Github
- Slidedeck