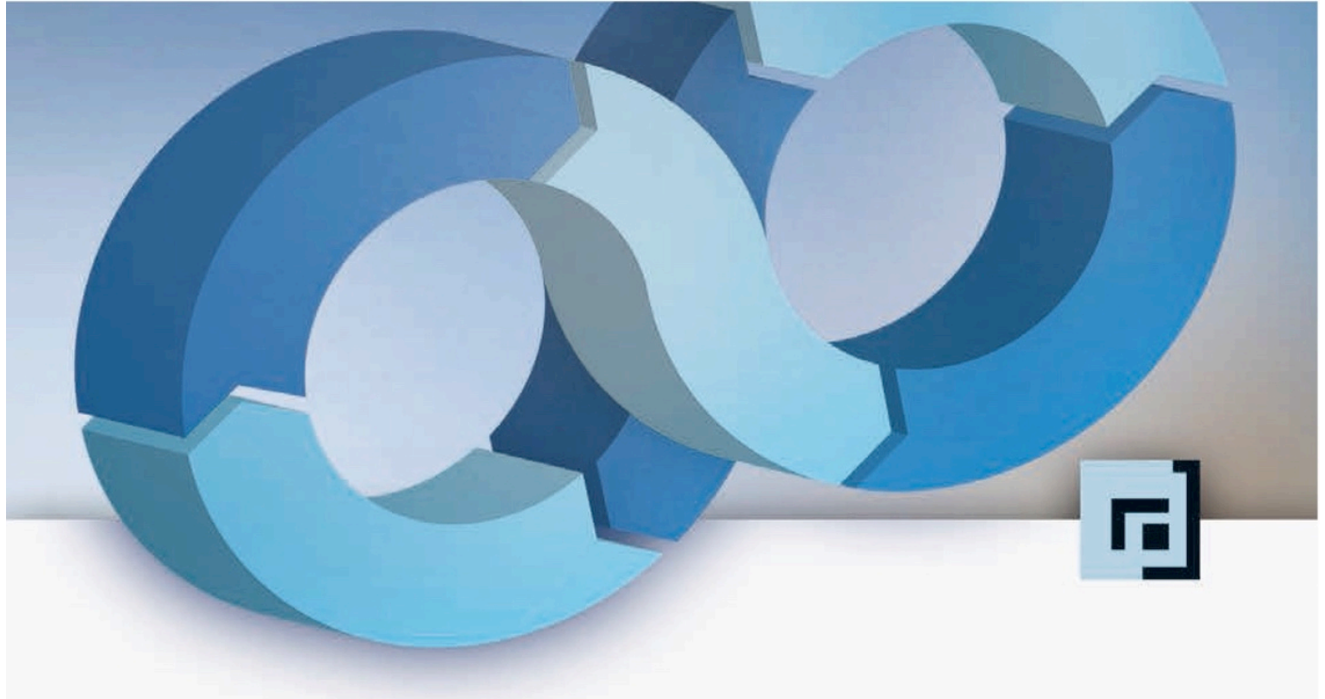
Many Roads Lead to Microservices

Eberhard Wolff
Fellow
innoQ
@ewolff



Eberhard Wolff

Continuous Delivery

Der pragmatische Einstieg

dpunkt.verlag

e4064ddeeee1e5e4f0e8-20141006091420-5129-1
eberhard.wolff@gmail.com

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!!!!

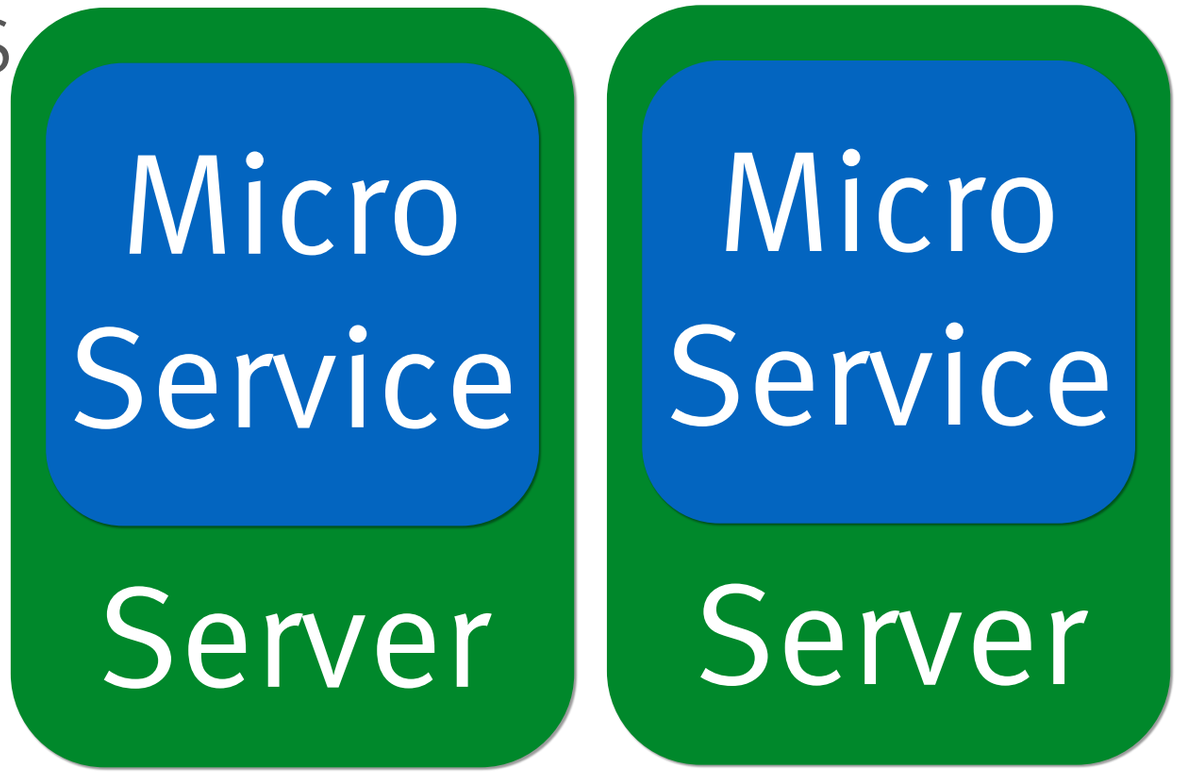
innoQ

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

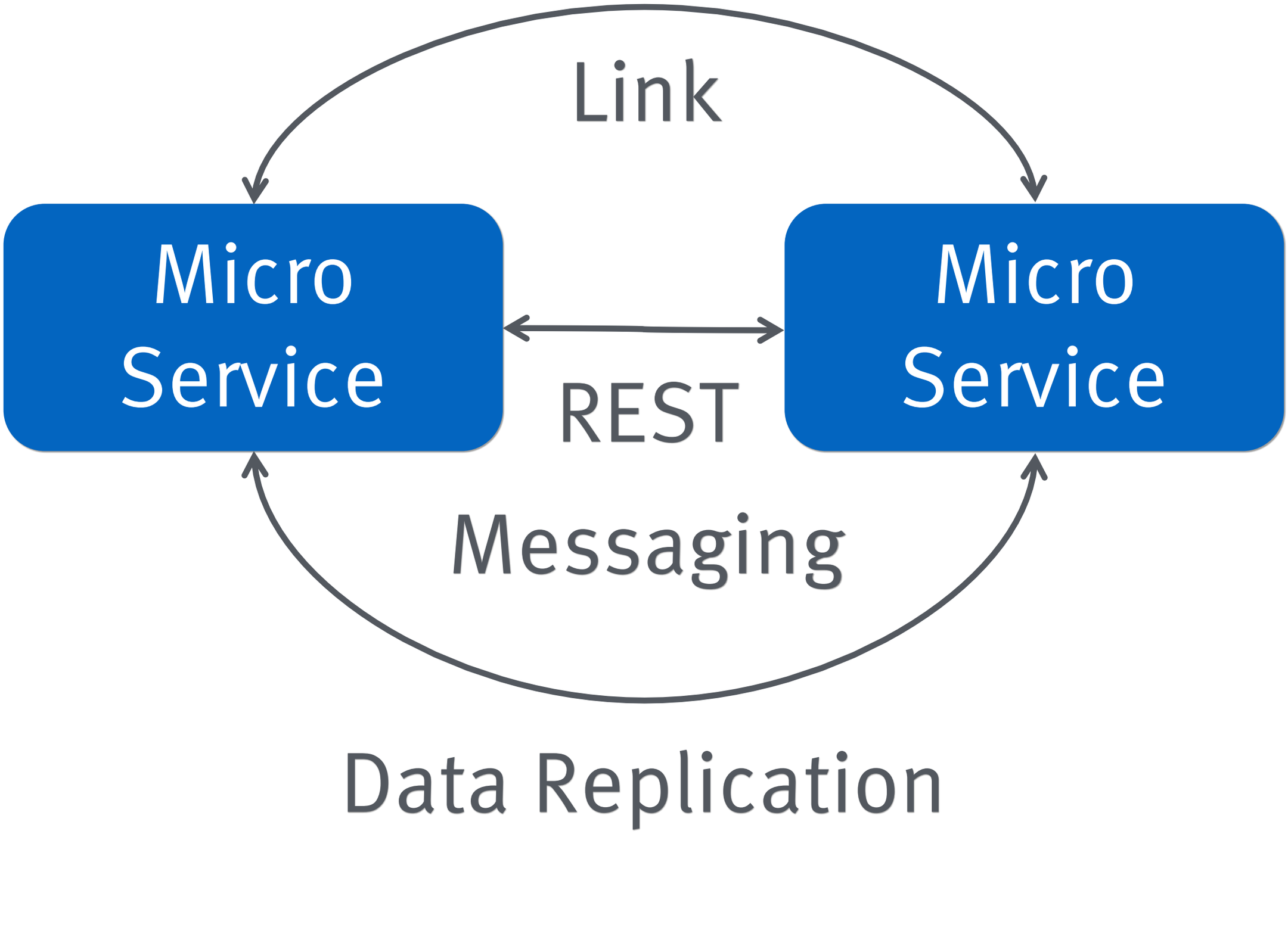
Microservice Definition

Microservices: Definition

- Small
- Independent deployment units
- Separate VM / process
- Any technology
- Any infrastructure



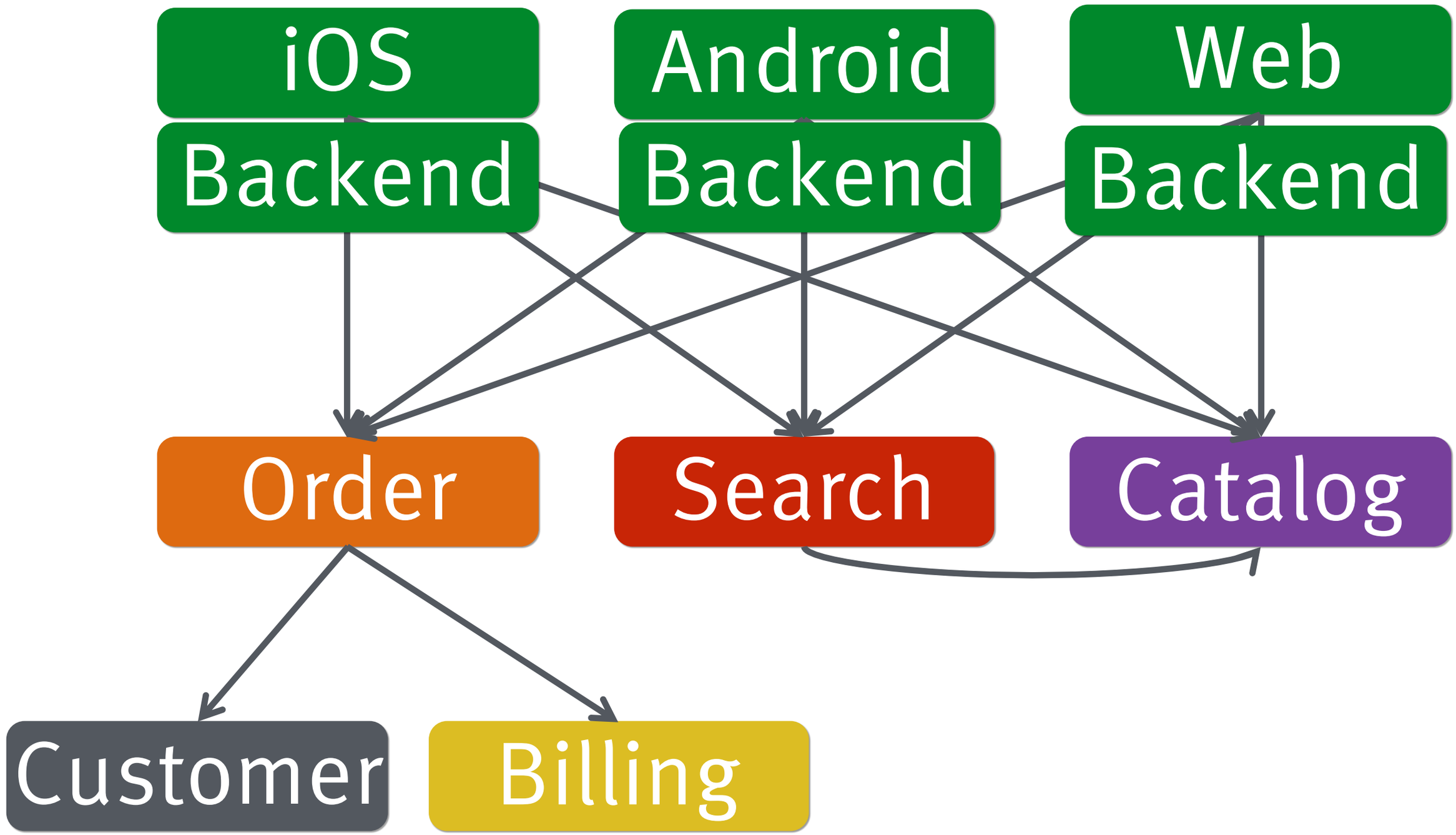
Components Collaborate



Microservices

- Component Model
- Component...
  - Individual deployment unit
  - GUI+Logic?

Layered


iOS
Backend
Android
Backend
Web
Backend
Order
Search
Catalog
Customer
Billing

Layered

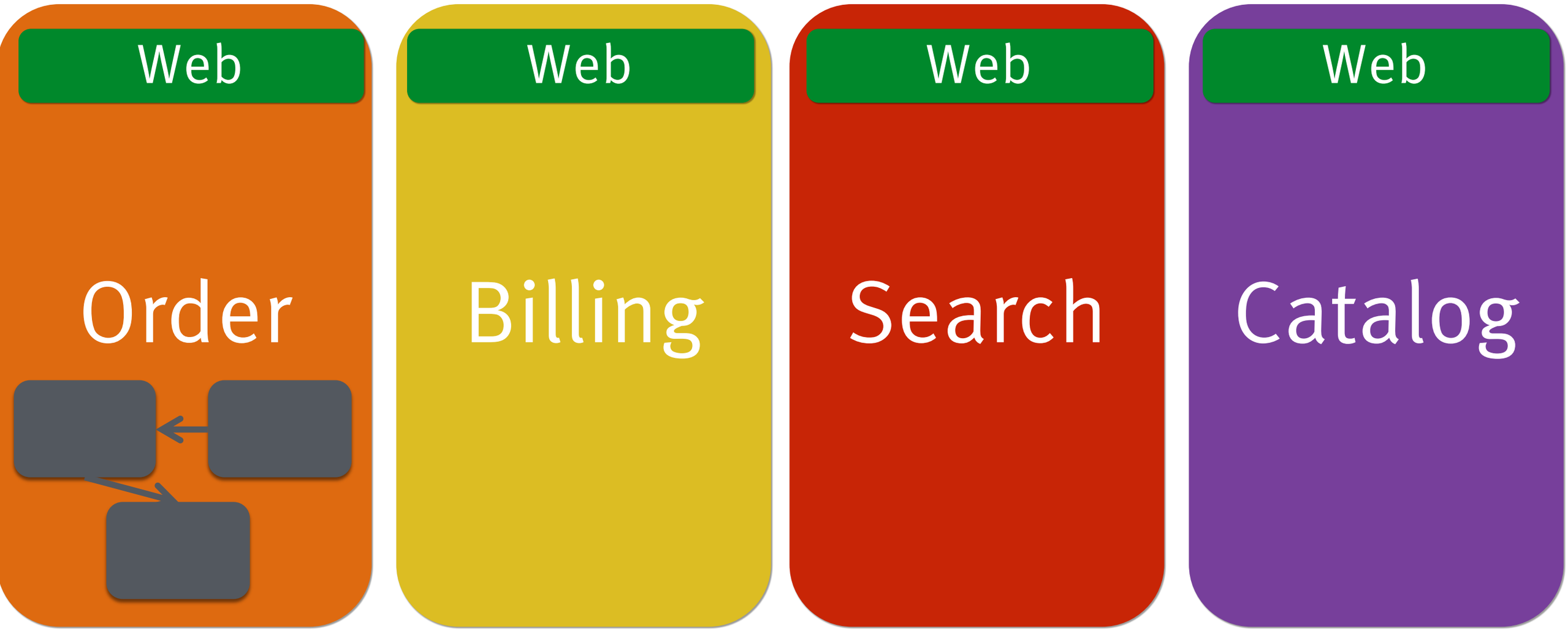
- Reusable Backend Services
- Mobile client / Web App as frontend

- Backend for frontend (BFF): Custom backend services
- ...to implement frontend specific logic

Layered: Issues

- All BFF support the same processes
- BFF contain most relevant logic
- Change to a business process means changing many services
- Lots of communication

Self-contained Systems



Self-contained Systems (SCS)

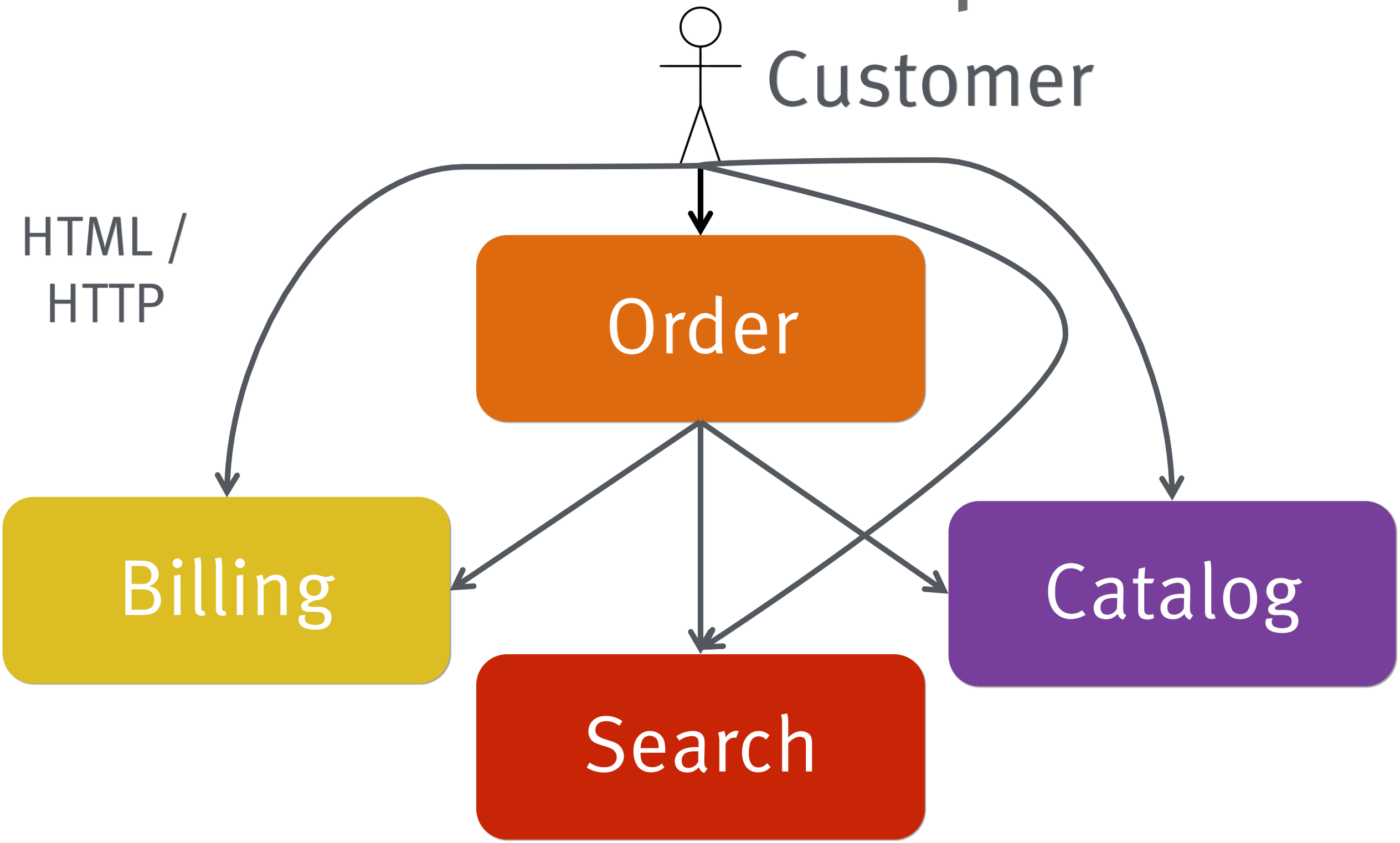
- SCS: Autonomous web application
- Optional service API (e.g. for mobile clients)
- Includes data & logic
- Might contain several microservices
- No shared UI
- No shared business code
- E.g. Otto, Kaufhof …

SCS: Benefits

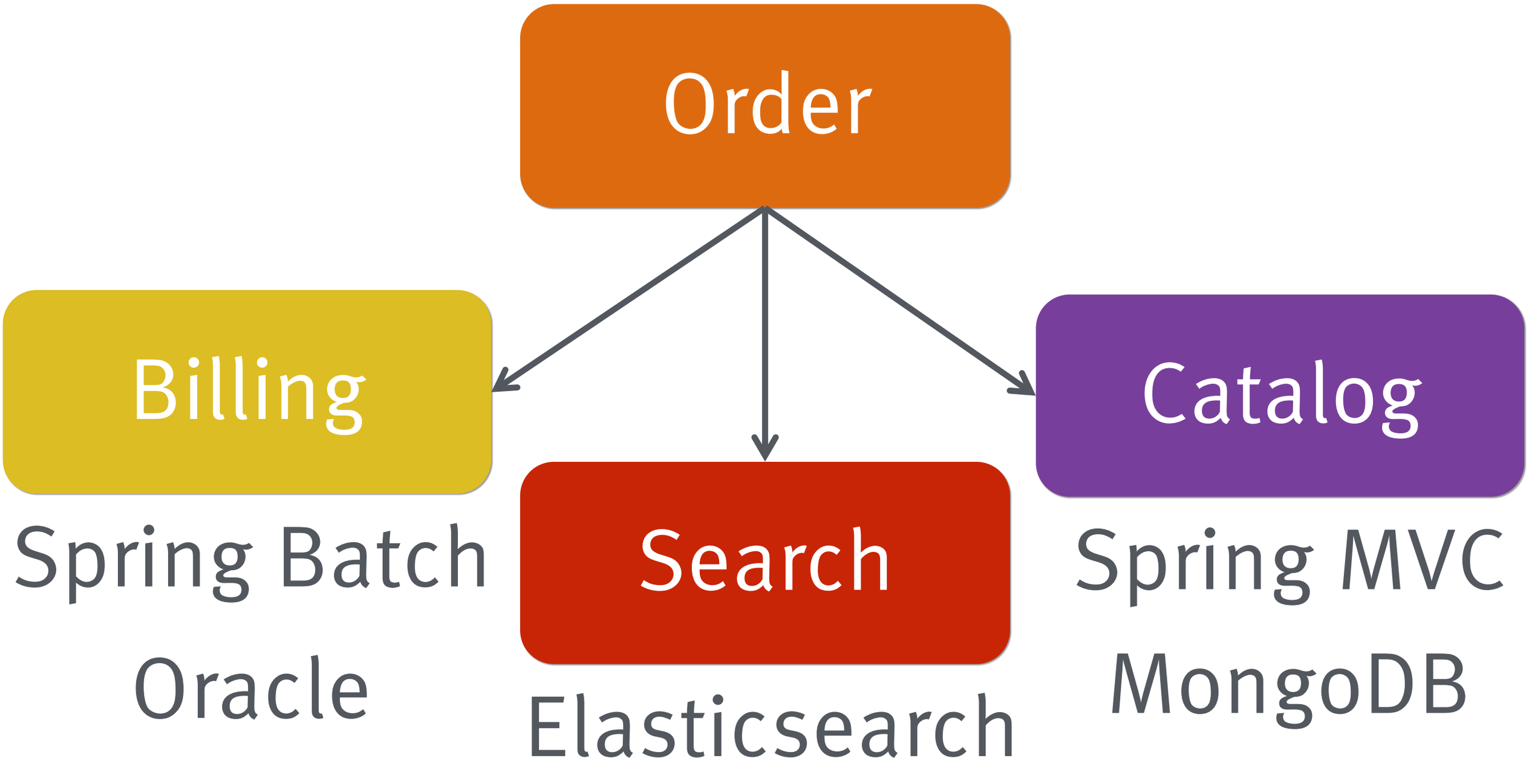
- Business logic for one domain in one SCS
- Change usually local to one SCS
- Less communication between SCS

- I think this should be the goal
- http://scs-architecture.org

Online Shop



Online Shop


Order
Billing
Spring Batch
Oracle
Search
Elasticsearch
Catalog
Spring MVC
MongoDB

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

Monolith to Microservices

The Typical Scenario: Monolith to Microservices

Why Migrate?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Deployment

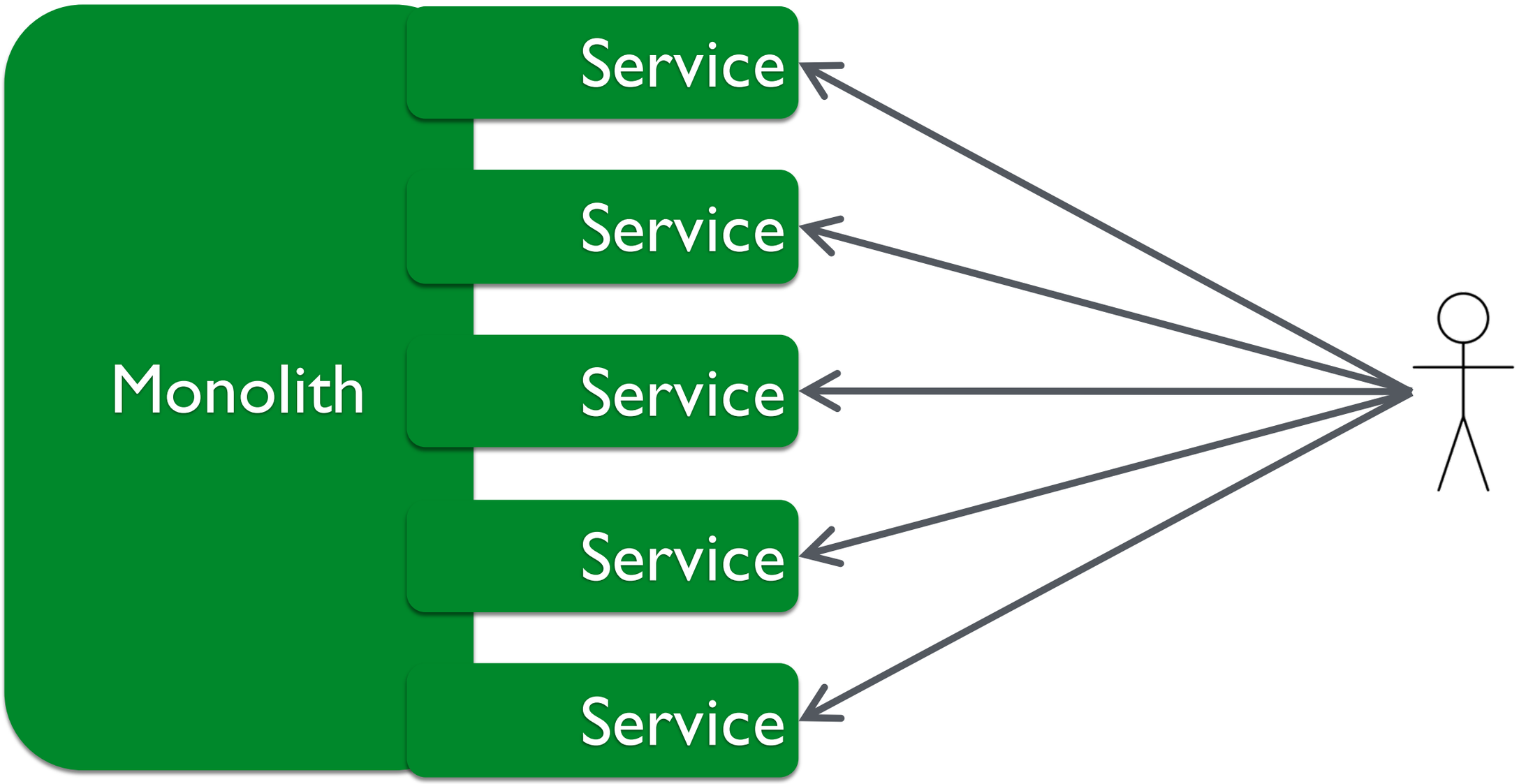
Continuous Delivery

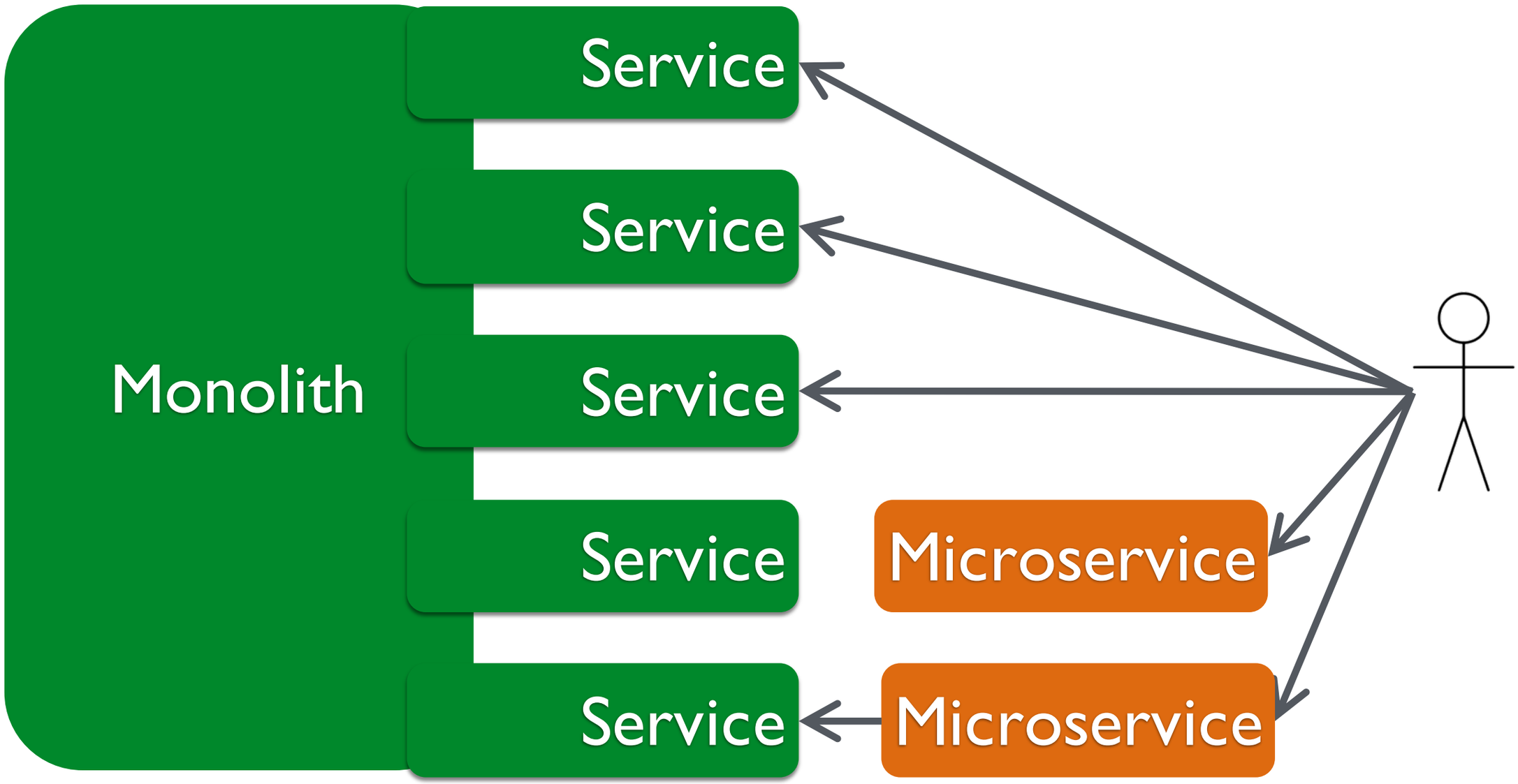
Robustness

Independent Scaling

Technology

Free choice of technology


Monolith
Service
Service
Service
Service
Service


Monolith
Service
Service
Service
Service
Service
Microservice
Microservice

Order

Billing

Search

Catalog

ECommerce System

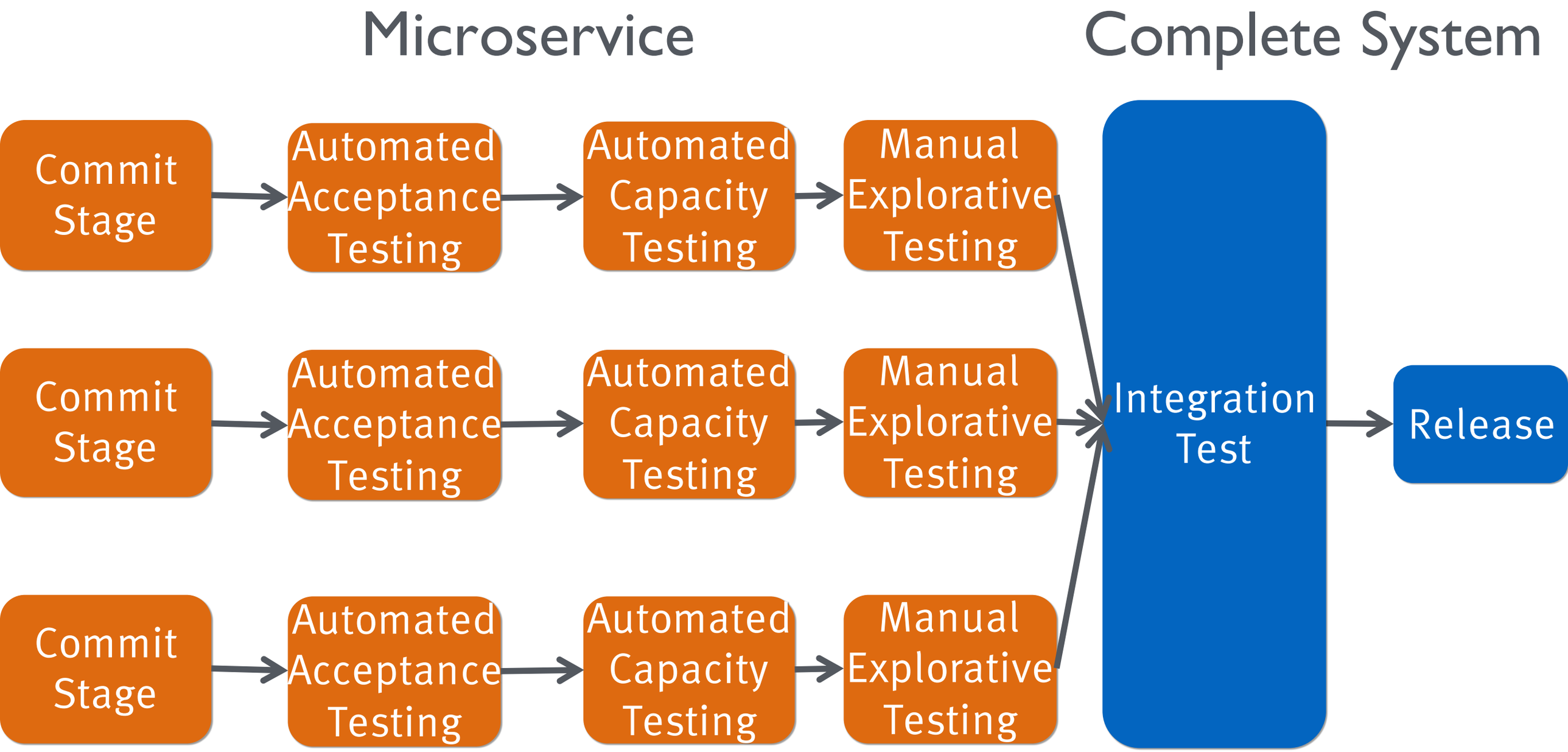
Approach

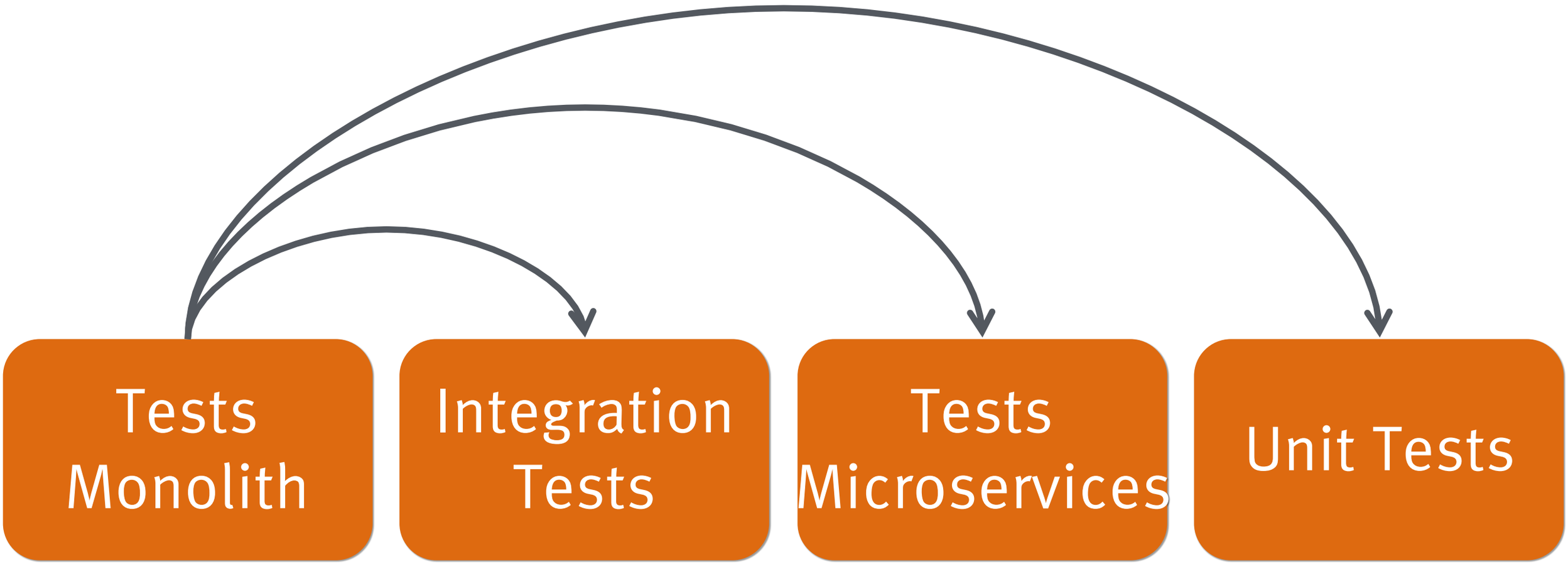
- Add Microservice to Monolith
- Microservice might implement only specific use cases
- Separate by domain

Challenges

- Operations: A lot more deployment units
- Separate tests, too!

Continuous Delivery Pipeline




Tests
Monolith
Integration
Tests
Tests
Microservices
Unit Tests

Eventually the Monolith will be unimportant

No need for full migration

Green Field

Green Field
The Exception

Why Start?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Deployment

Continuous Delivery

Robustness

Independent Scaling

Technology

Free choice of technology

Start With a Monolith

- Easy to refactor – even large scale
- Break it apart
- We know how to do it!
- http://martinfowler.com/bliki/MonolithFirst.html

Seriously?

- Breaking Monoliths apart is hard
- Structure might deteriorate
- Standardized Ops not established
- http://martinfowler.com/articles/dont-start-monolith.html (Stefan Tilkov)

Start BIG

- Rough structure should be clear
- Project not very large
- E.g. Component by team
- Create more Microservices as you go

Technical Advantages

Why Migrate?

Scaling Agile

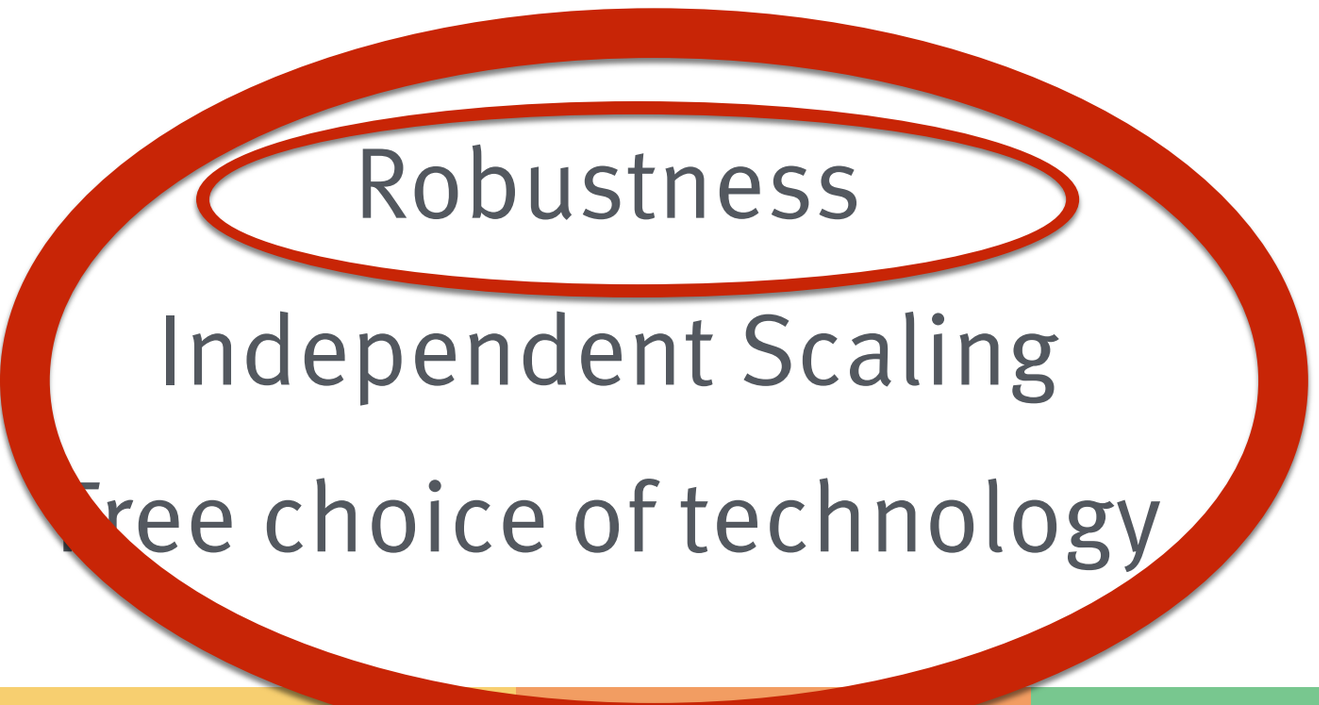
Organization

Handle Legacy efficient

Sustainable development

Deployment

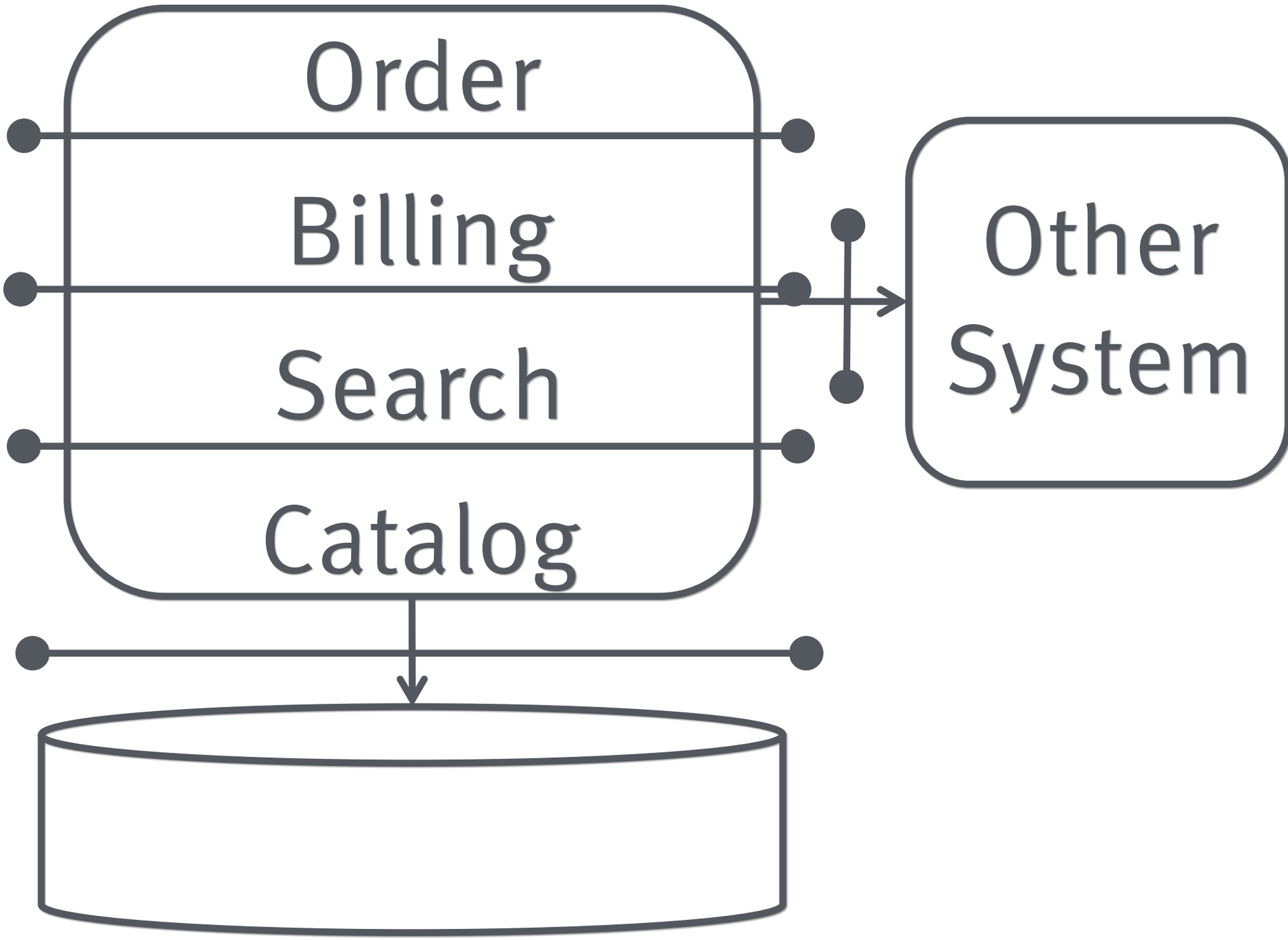
Continuous Delivery



Technology

ECommerce System

Bulkhead



Bulkhead

- Separate thread pools
- Separate connection pools

- Might add Timeouts
- Might add Circuit Breaker

Technical Advantages

- Separation by domain?
- Might include interface to e.g. database / other systems
- Different from breaking apart the Monolith

Conclusion

Conclusion: Three Scenarios

- Breaking apart monolith
- Green field
- Realizing technical advantages

Meta Conclusion

- More than one reason for Microservices
- More than one way to adopt Microservices
- Probably more than a hype

Thank You!

@ewolff

https://github.com/ewolff/microservice

http://microservices-buch.de/

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