

Many Roads Lead to Microservices

Eberhard Wolff
Fellow
innoQ
@ewolff



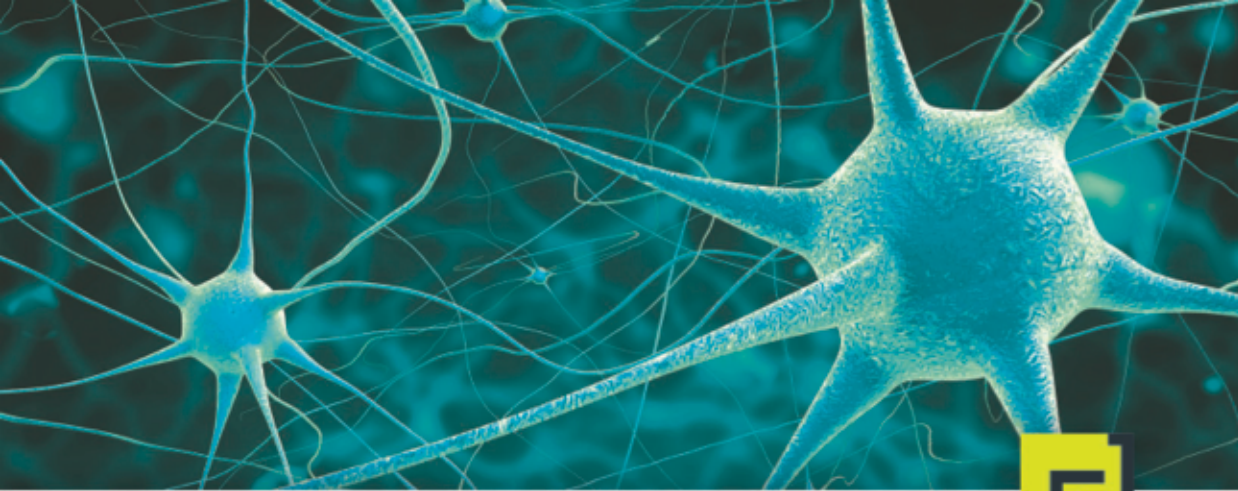


Eberhard Wolff

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

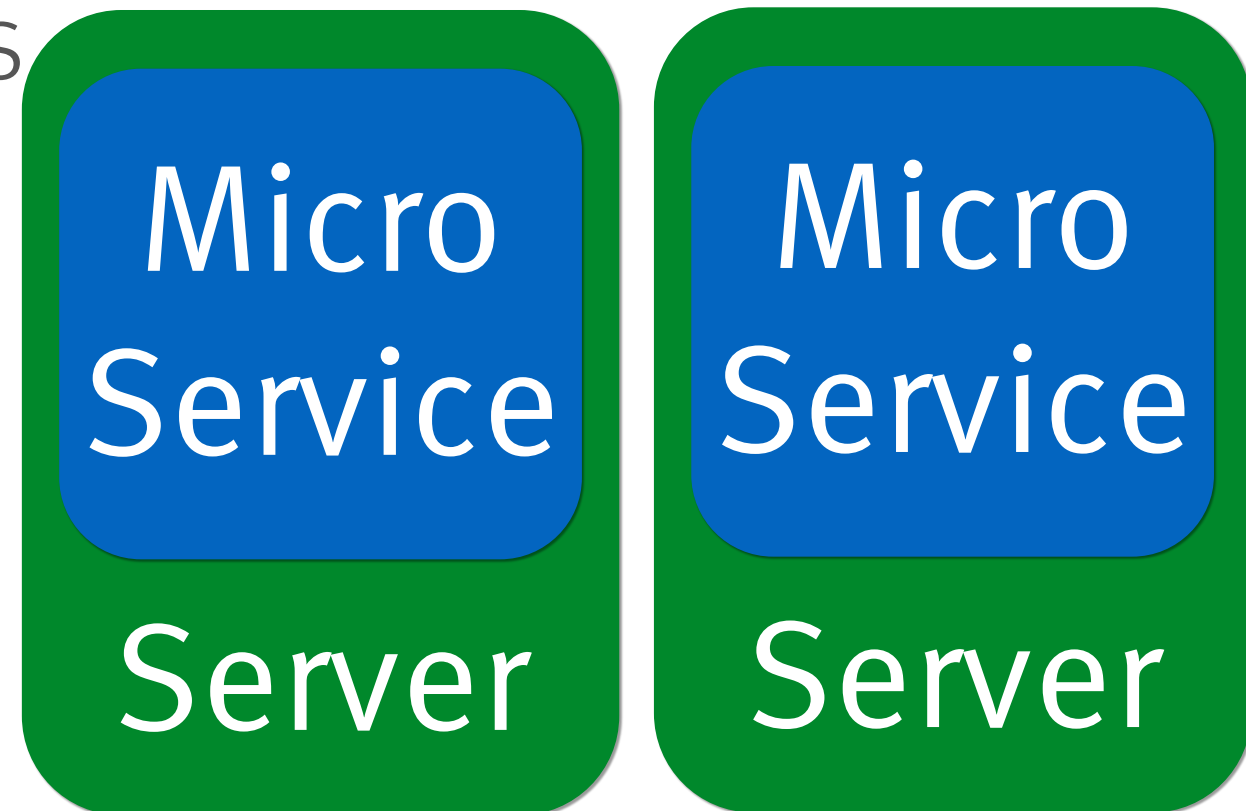
<http://microservices-book.com/primer.html>

Microservice Definition

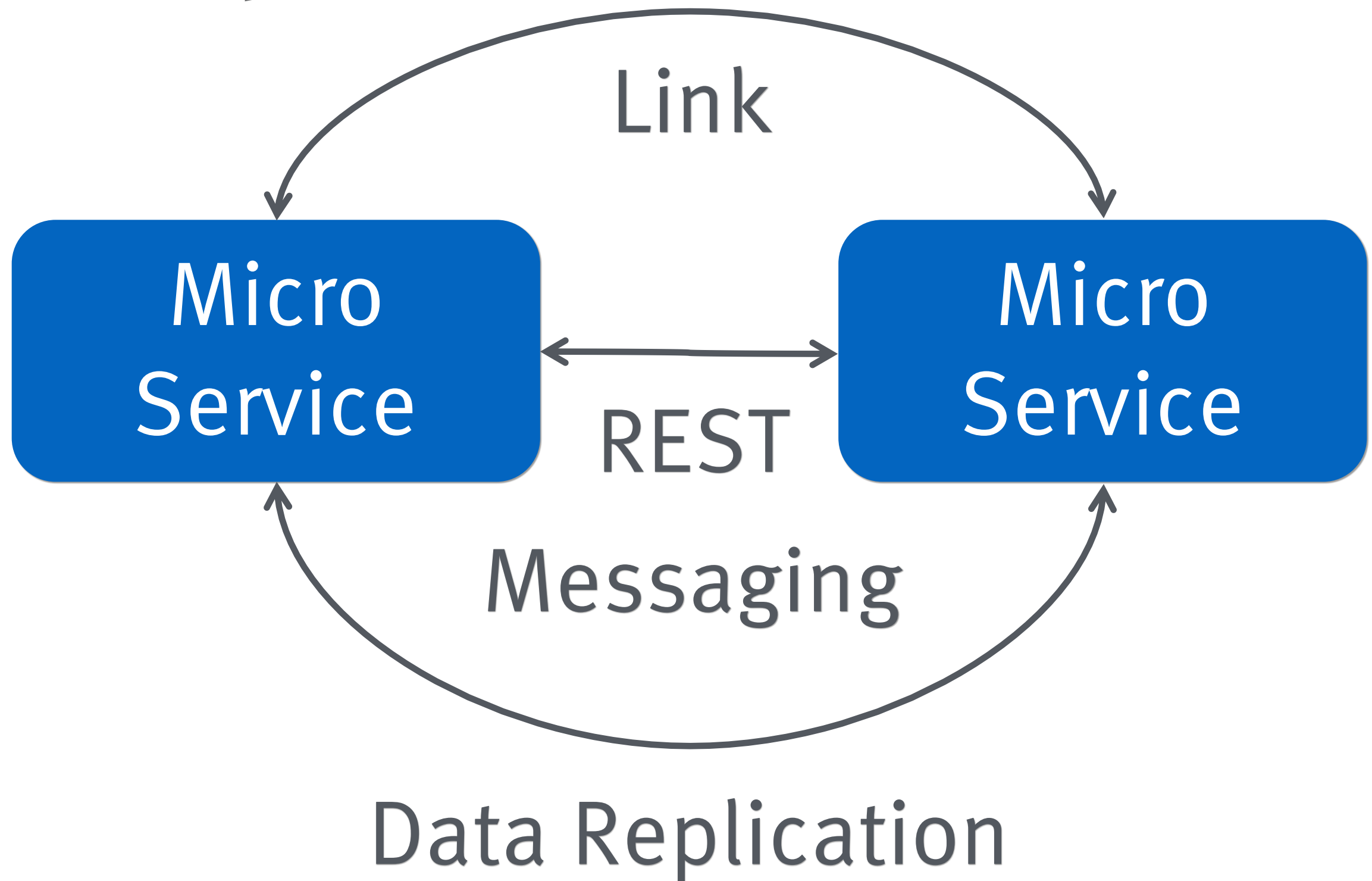


Microservices: Definition

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



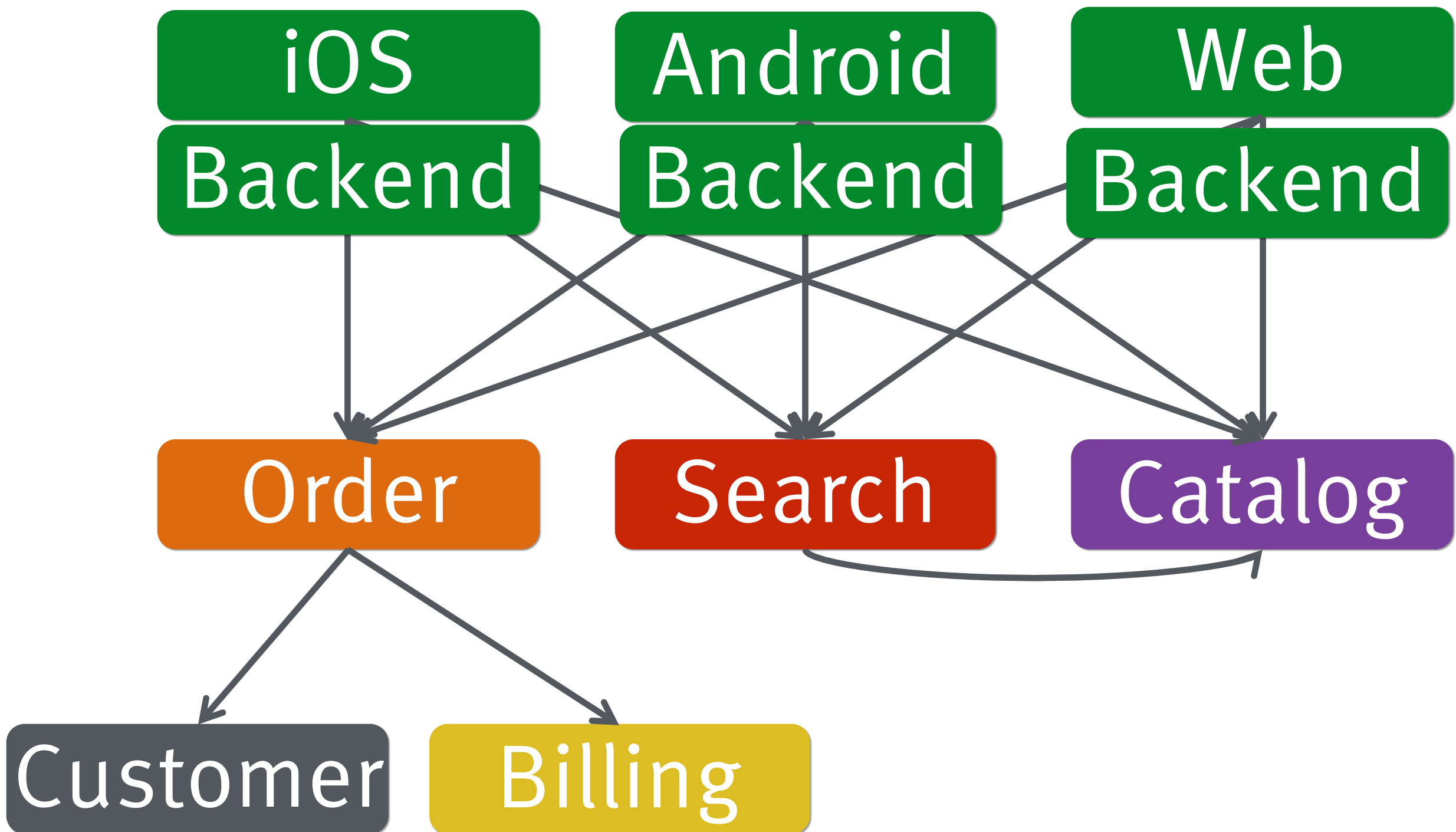
Components Collaborate



Microservices

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

Layered



Layered

- › Reusable Backend Services
- › Mobile client / Web App as frontend
- › Backend for frontend (BFF): Custom backend services
- › ...to implement frontend specific logic

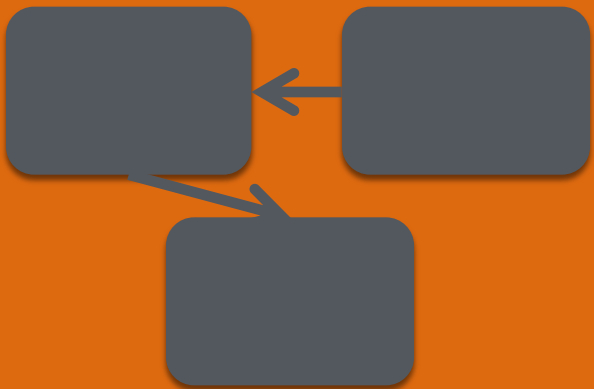
Layered: Issues

- › 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

Web

Order



Web

Billing

Web

Search

Web

Catalog

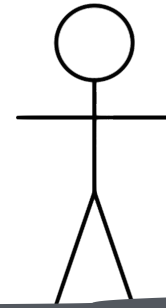
Self-contained Systems (SCS)

- › 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



Customer

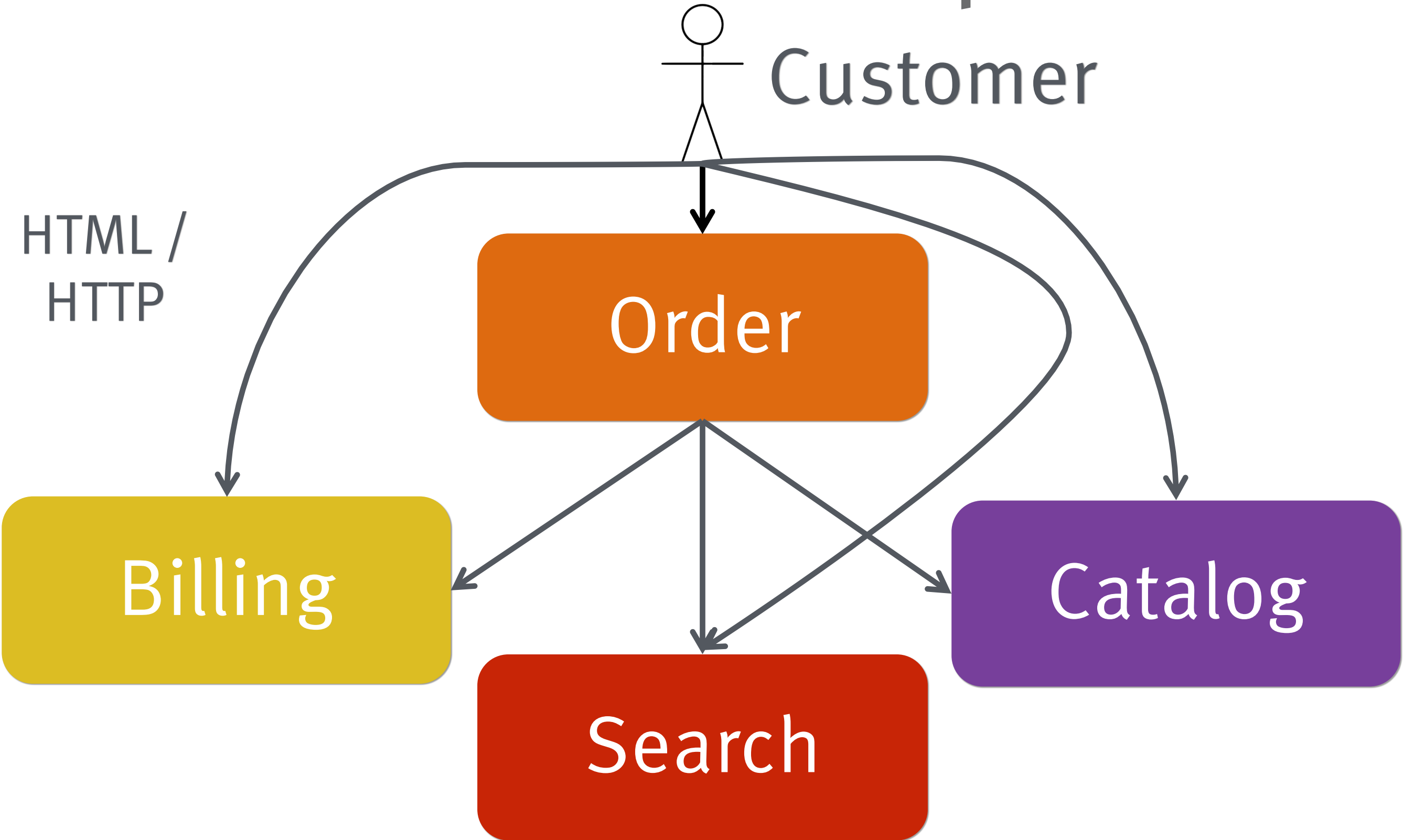
HTML /
HTTP

Order

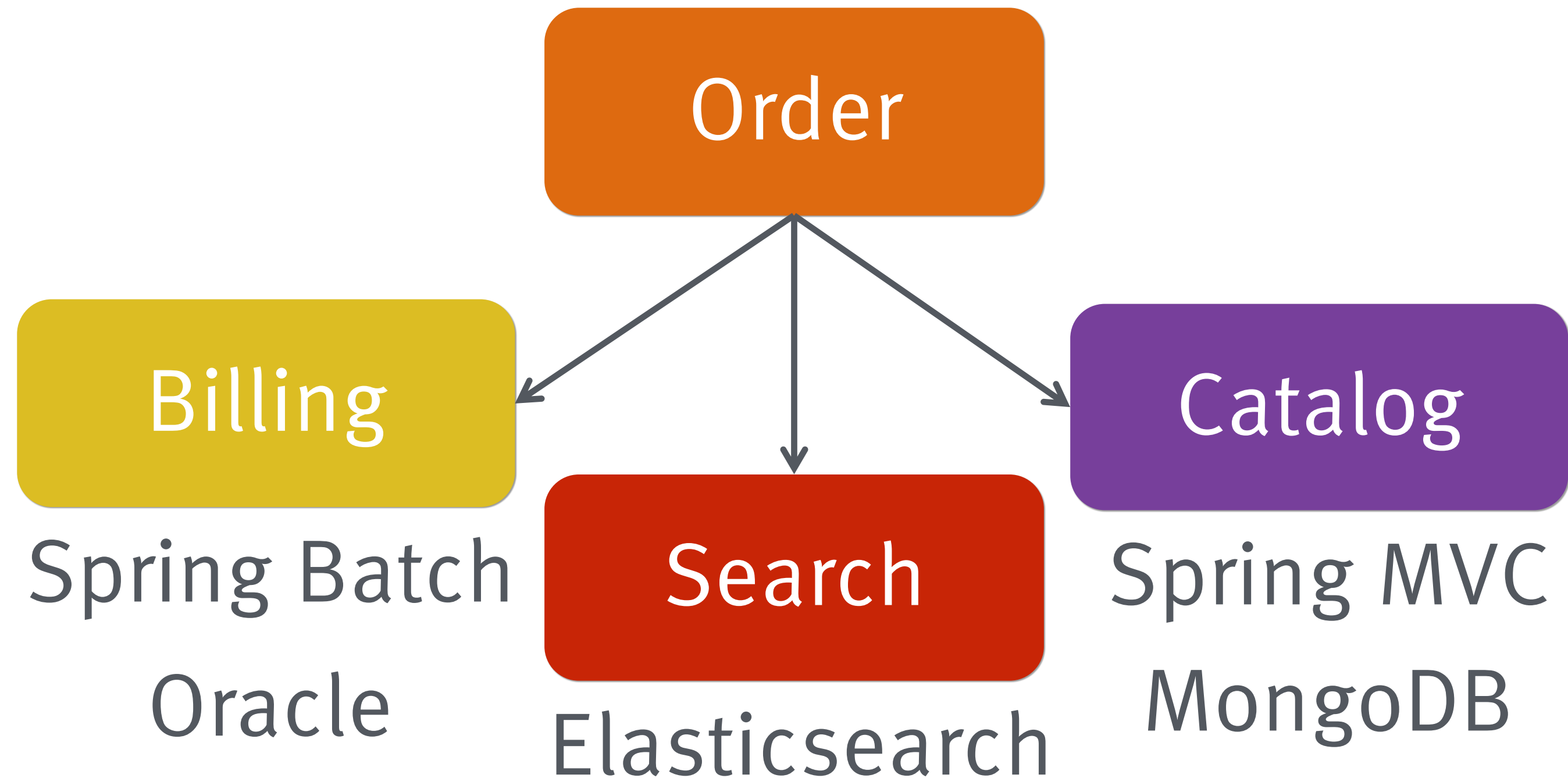
Billing

Search

Catalog



Online Shop



Distributed System

Distributed System

Why??

Why Microservices?

Scaling Agile

Small teams develop and deploy independently

Handle Legacy efficient
Sustainable development
Continuous Delivery

Add services – not code
Strong Modularization
Replaceable Services
Small Services

Robustness

Failure limited to single
Microservice

Independent Scaling

Free choice of technology

Why Microservices?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Continuous Delivery

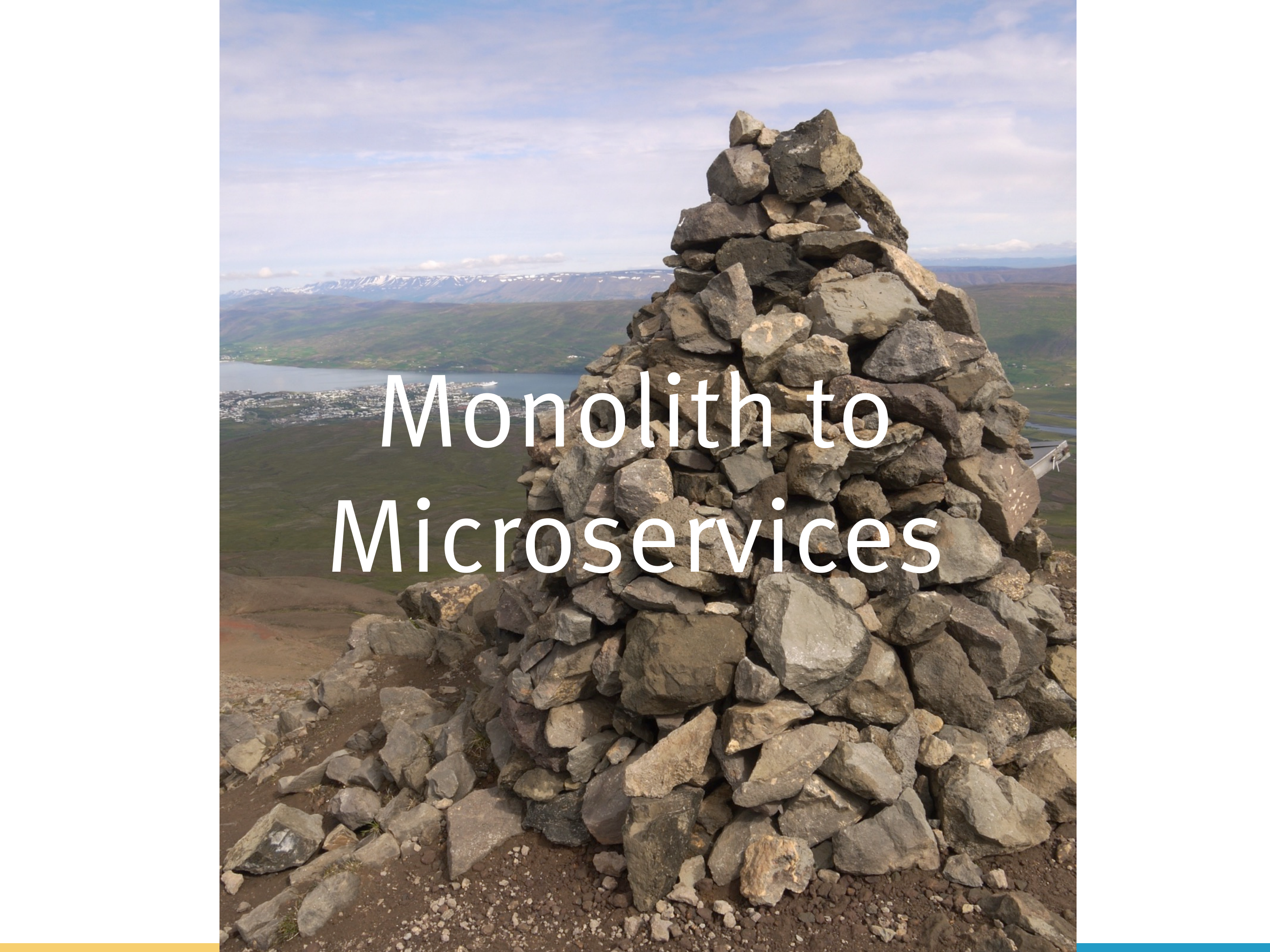
Deployment
Units

Robustness

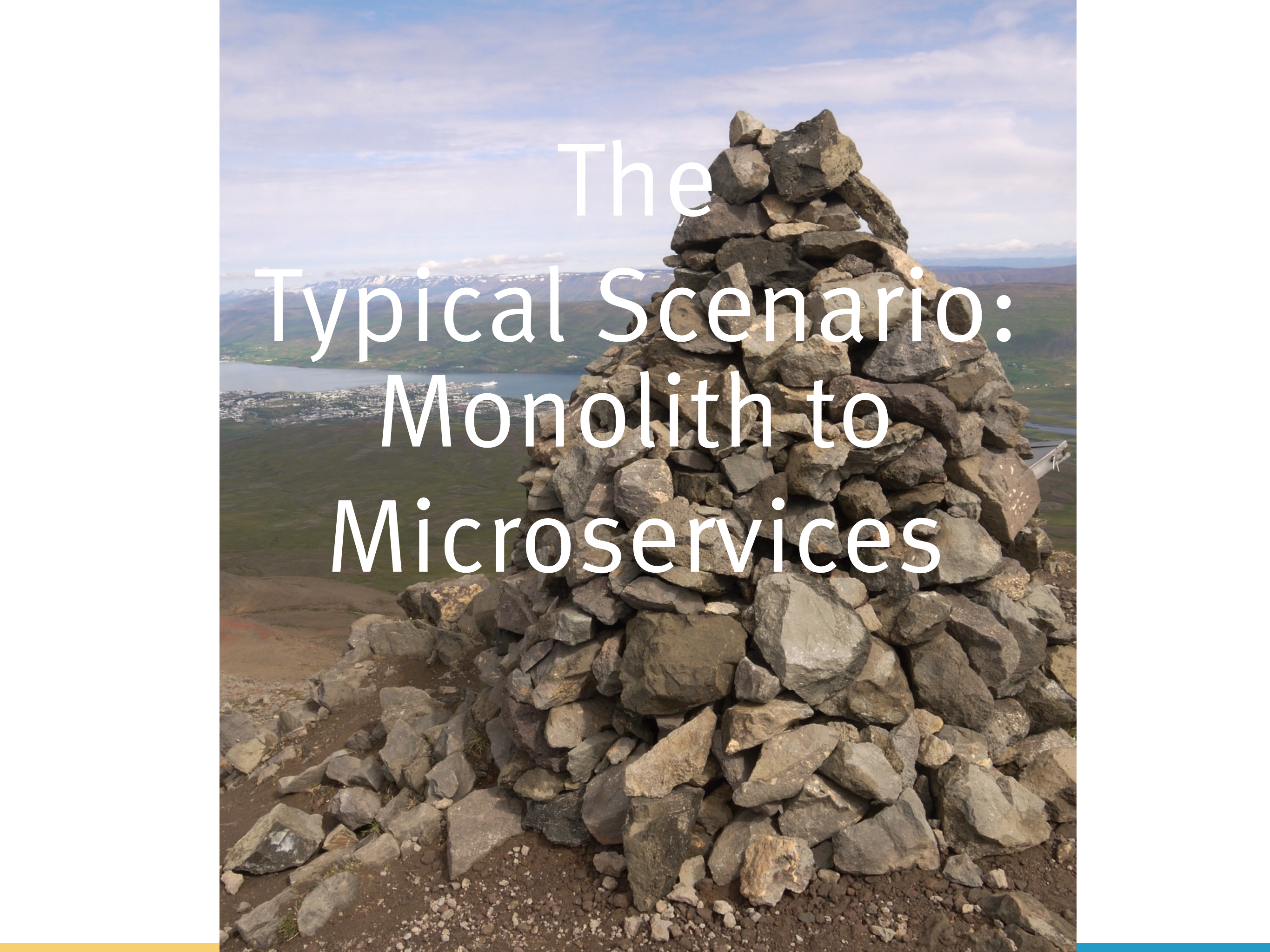
Independent Scaling

Technology

Free choice of technology

A tall, conical stone cairn made of dark, irregularly shaped rocks stands prominently on a rocky, brownish-grey mountain peak. The cairn is the central focus of the image. In the background, a wide valley stretches out, featuring a small town or village with white buildings and a blue body of water. Beyond the valley, a range of mountains is visible, with the higher peaks covered in snow under a blue sky with scattered white clouds. The overall scene conveys a sense of a remote, high-altitude location.

Monolith to Microservices

A tall, conical stone cairn made of dark, irregularly shaped rocks stands prominently on a mountain peak. The background reveals a vast landscape with a winding river, a small town nestled in a valley, and distant mountains under a blue sky with scattered clouds. The text "The Typical Scenario: Monolith to Microservices" is overlaid in white, sans-serif font across the center of the image.

The Typical Scenario: Monolith to Microservices

Why Migrate?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Deployment

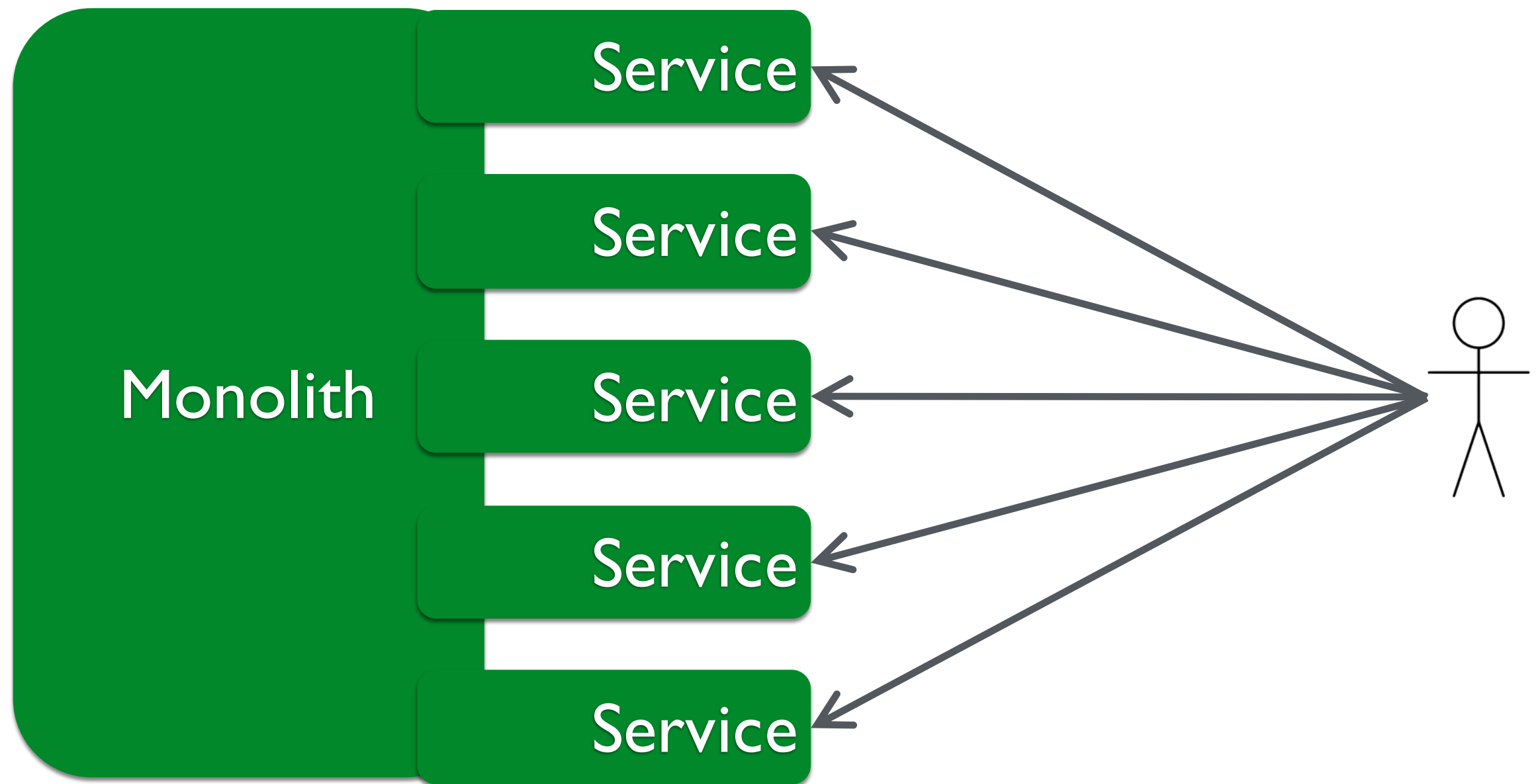
Continuous Delivery

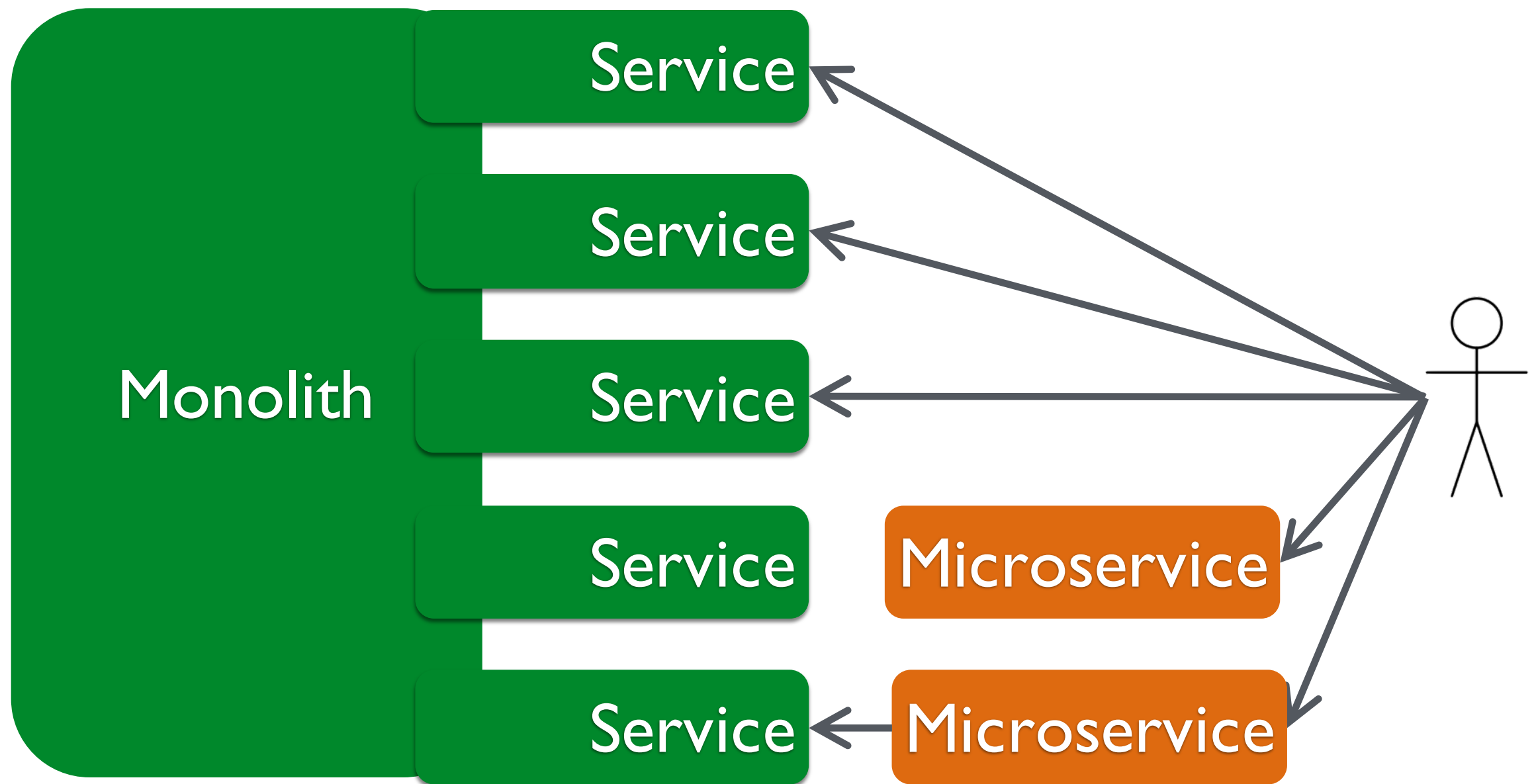
Robustness

Independent Scaling

Technology

Free choice of technology



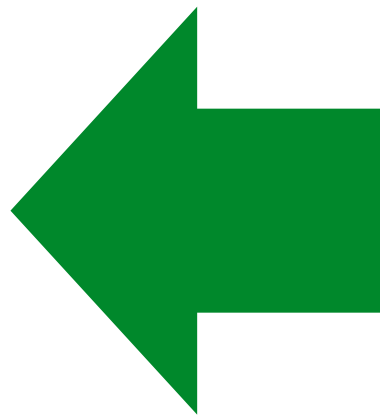


Order

Billing

Search

Catalog



ECommerce
System

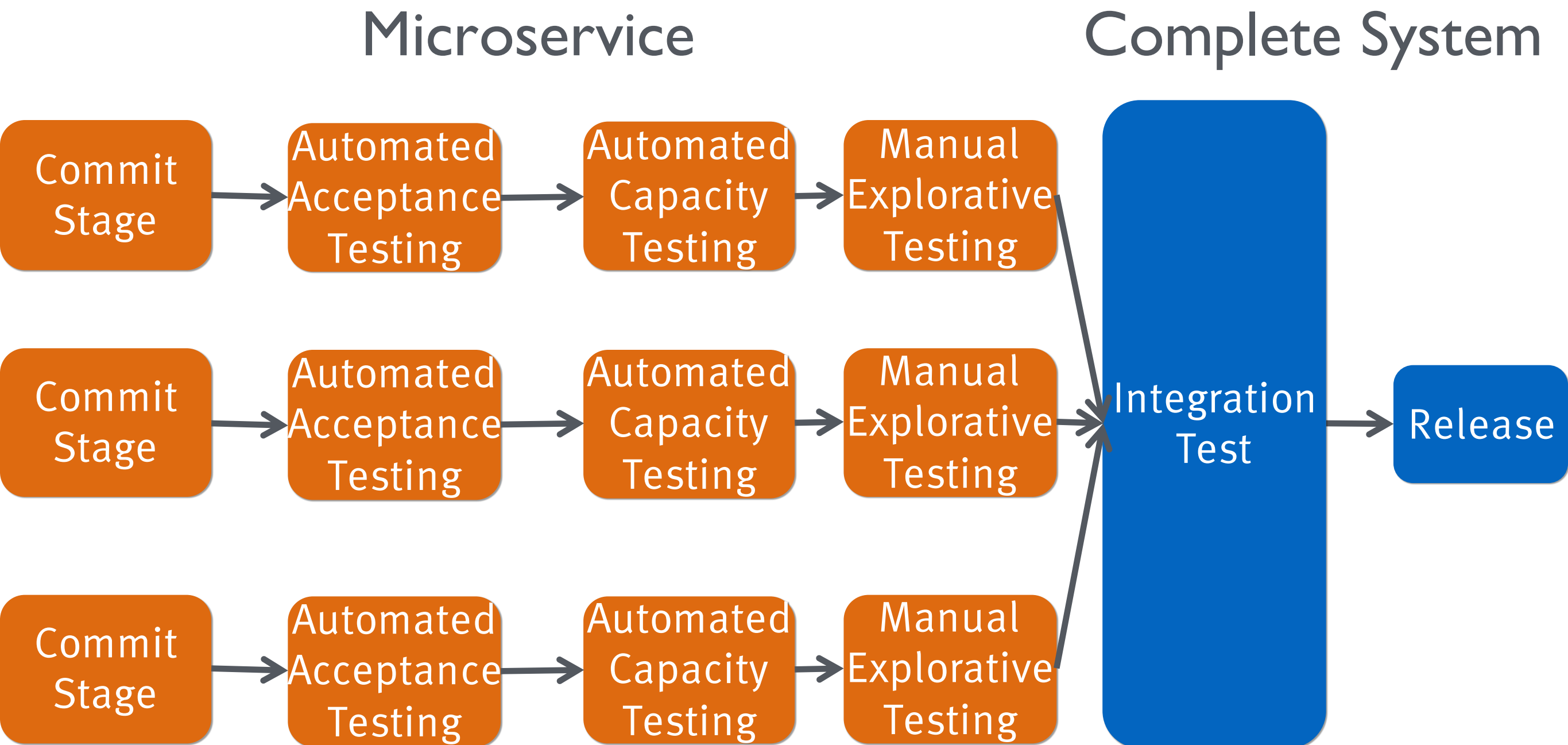
Approach

- › 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

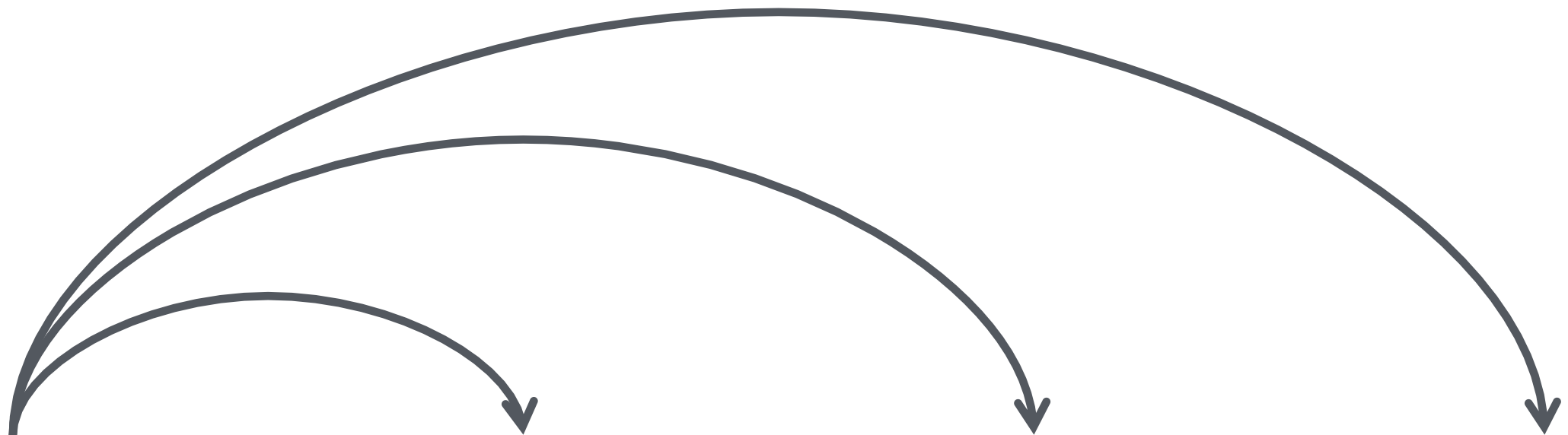


Tests
Monolith

Integration
Tests

Tests
Microservices

Unit Tests



Eventually the
Monolith will be
unimportant

No need for full
migration

A landscape photograph showing a vast, green field in the foreground, dotted with small white and yellow flowers. A narrow, muddy path or streambed winds through the grass. In the background, a large, dark mountain with patches of snow or ice on its slopes rises against a pale, overcast sky. The text "Green Field" is overlaid in the center of the image.

Green Field

A landscape photograph showing a vast green field in the foreground, with a dirt path leading towards a mountain in the background. The mountain has patches of snow and is partially covered in smoke or mist. The sky is overcast with soft, pinkish light.

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 Advantage s

Why Migrate?

Scaling Agile

Organization

Handle Legacy efficient

Sustainable development

Deployment

Continuous Delivery

Robustness

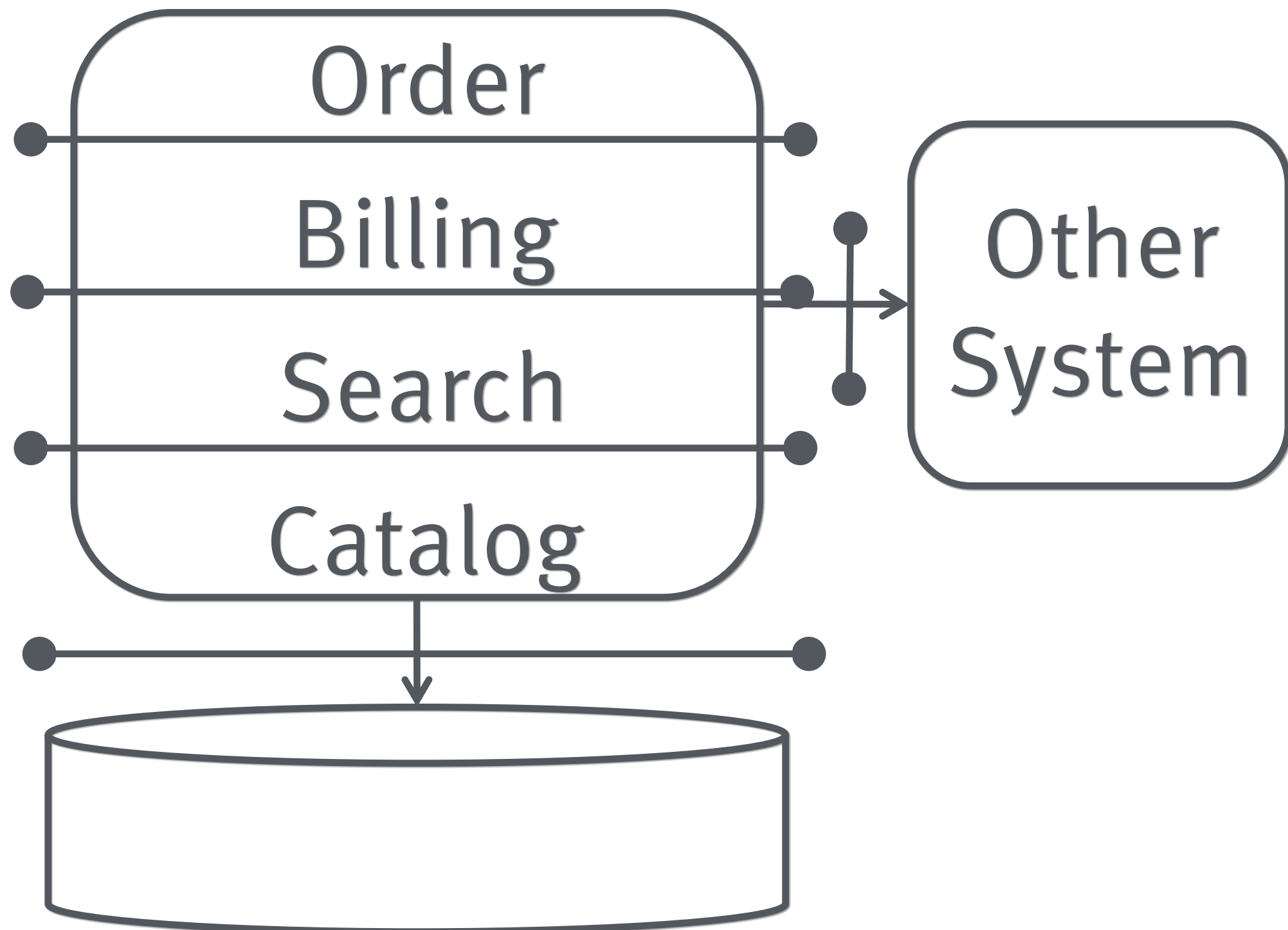
Independent Scaling

Technology

Free choice of 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>

