

Microservices, DevOps, Continuous Delivery – More Than Three Buzzwords

Eberhard Wolff
Fellow
@ewolff



Microservices, DevOps, Continuous Delivery – How Do They Relate?

Microservices, DevOps, Continuous Delivery – How Do They Solve Problems?

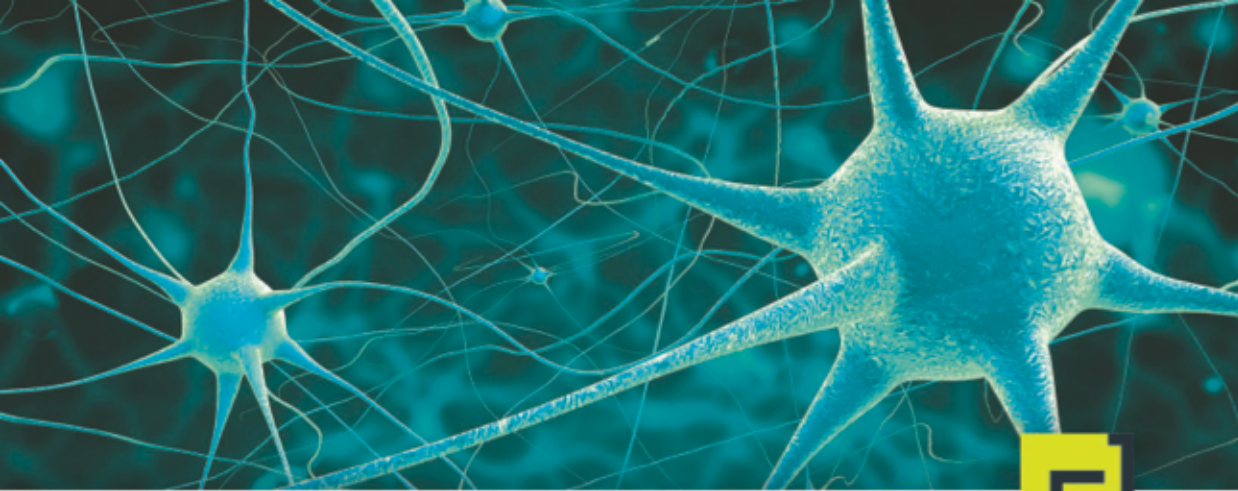


Eberhard Wolff

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/>

Microservices?

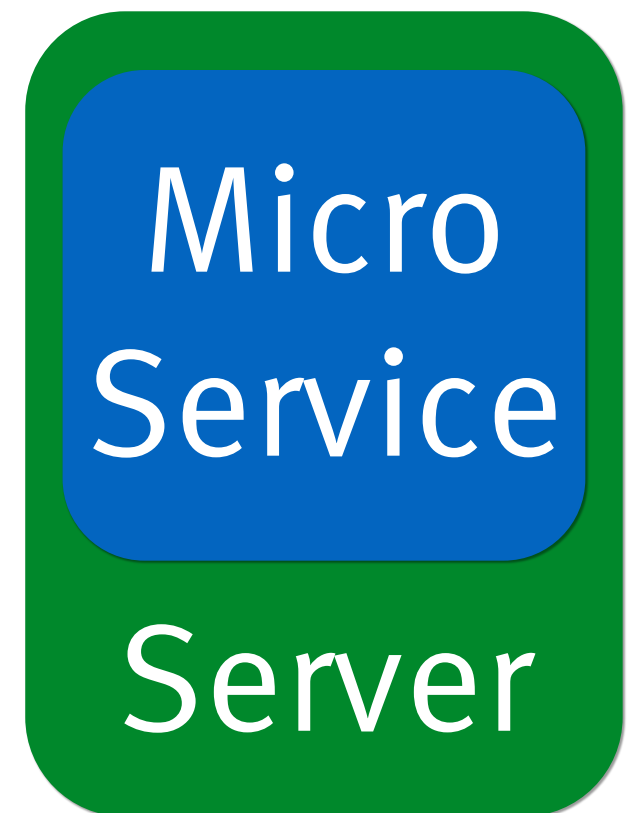
UNIX Philosophy

- › Write programs that do one thing and do it well
- › Write programs to work together
- › Write programs with a common interface

Definition

Microservice

- › Independent deployment unit
- › Separate data handling & storage
- › Should include UI
- › Order process, Billing, Catalog...
- › Process
- › VM
- › Docker container



Deployment Monolith

- › Might be well-structured inside.
- › But: Can only be deployed as a whole



Continuous Delivery: Build Pipeline

Commit
Stage

Automated
Acceptance
Testing

Deploy

Automated
Capacity
Testing

Deploy

Manual
Explorative
Testing

Deploy

Release

Deploy

Automated

Reproducible

Fast

Continuous Delivery: Build Pipeline



Commit
Stage

Automated
Acceptance
Testing

Deploy

Automated
Capacity
Testing

Deploy

Manual
Explorative
Testing

Deploy

Release

Deploy

Billing

Continuous Delivery
Pipeline too complex
or slow?



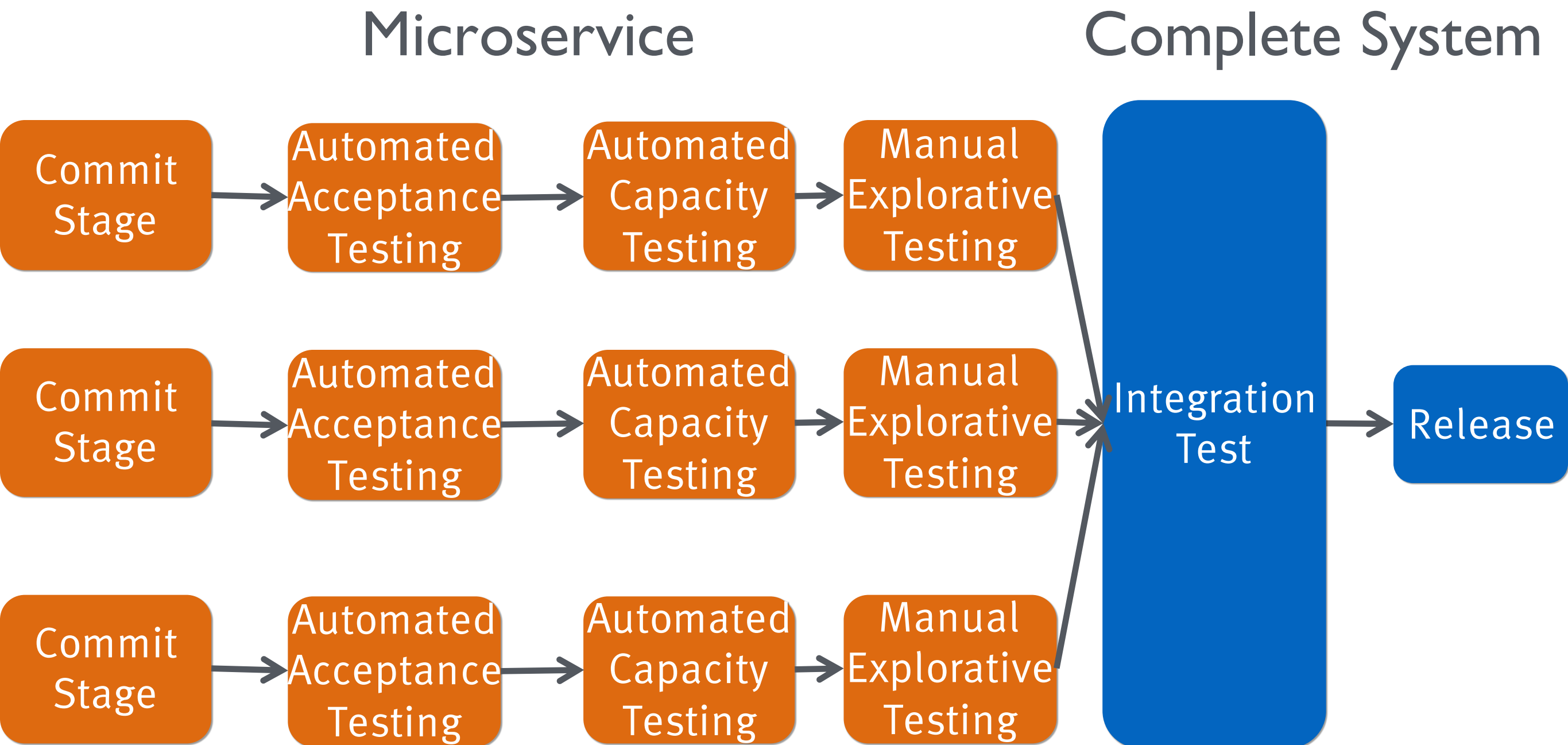
Continuous Delivery
Pipeline too complex
or slow?

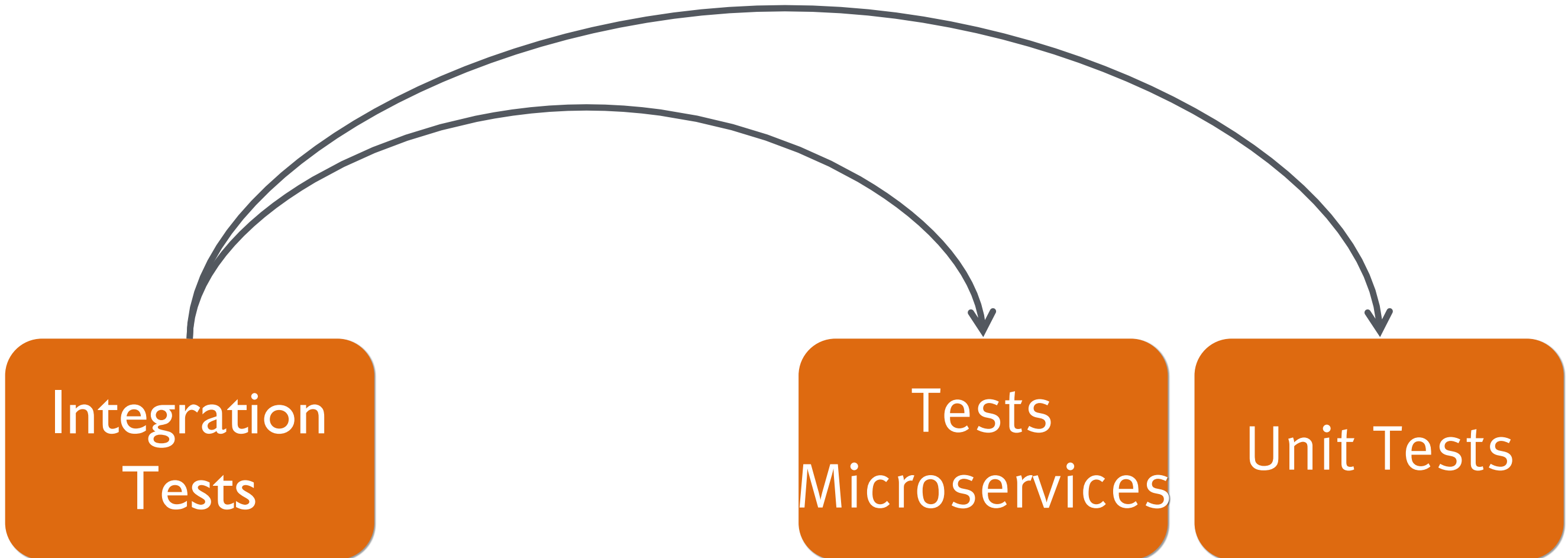
Consider changing the
architecture!

Microservices: Challenges

- › 50 or 100 Microservices
- › Deployment?
- › Monitoring?
- › Log Analysis?
- › Need Automation
- › Need common foundation for operations
- › Small deployment units are not enough

Continuous Delivery Pipeline





Integration
Tests

Tests
Microservices

Unit Tests

Microservices can
only be deployed
independently if tests
are independent!

Microservices can
only be deployed
independently if your
Continuous Delivery
Pipeline works.



Microservices + Continuous Delivery!



Why all the hassle?

Conway's Law

Architecture

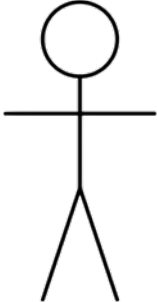
copies

communication structures

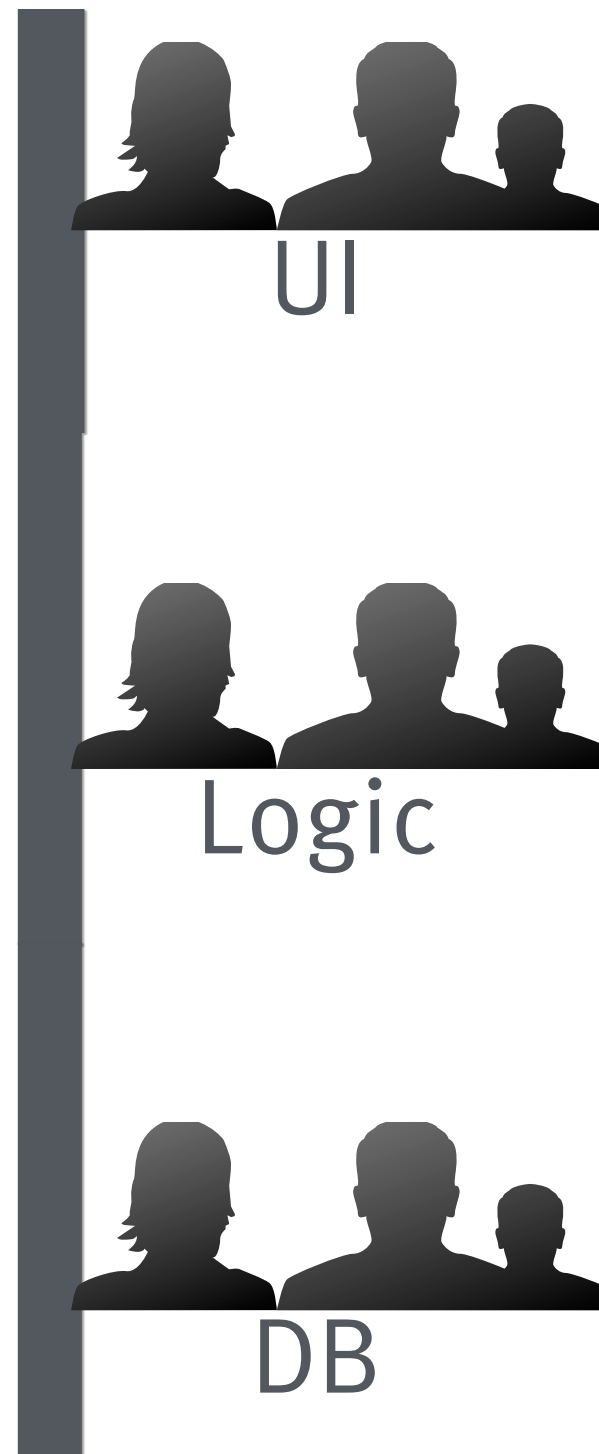
of the organization

Conway's Law as a Limit

- › Organization drives architecture
- › Teams of experts
- › i.e. UI, logic & database team
- › Three technical artifacts



Change
Order
Process!



E Commerce
Shop

3 sprints

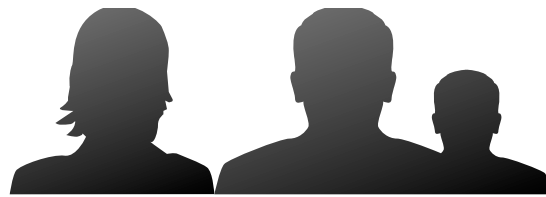
DB
Team Sprint

Logic
Team Sprint

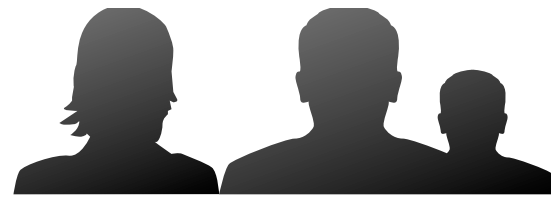
GUI
Team Sprint

time

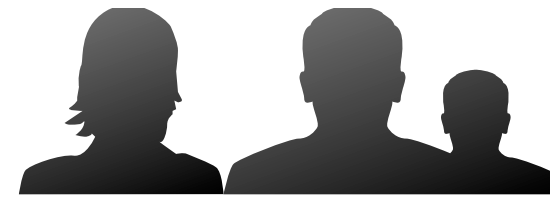




Order



Billing



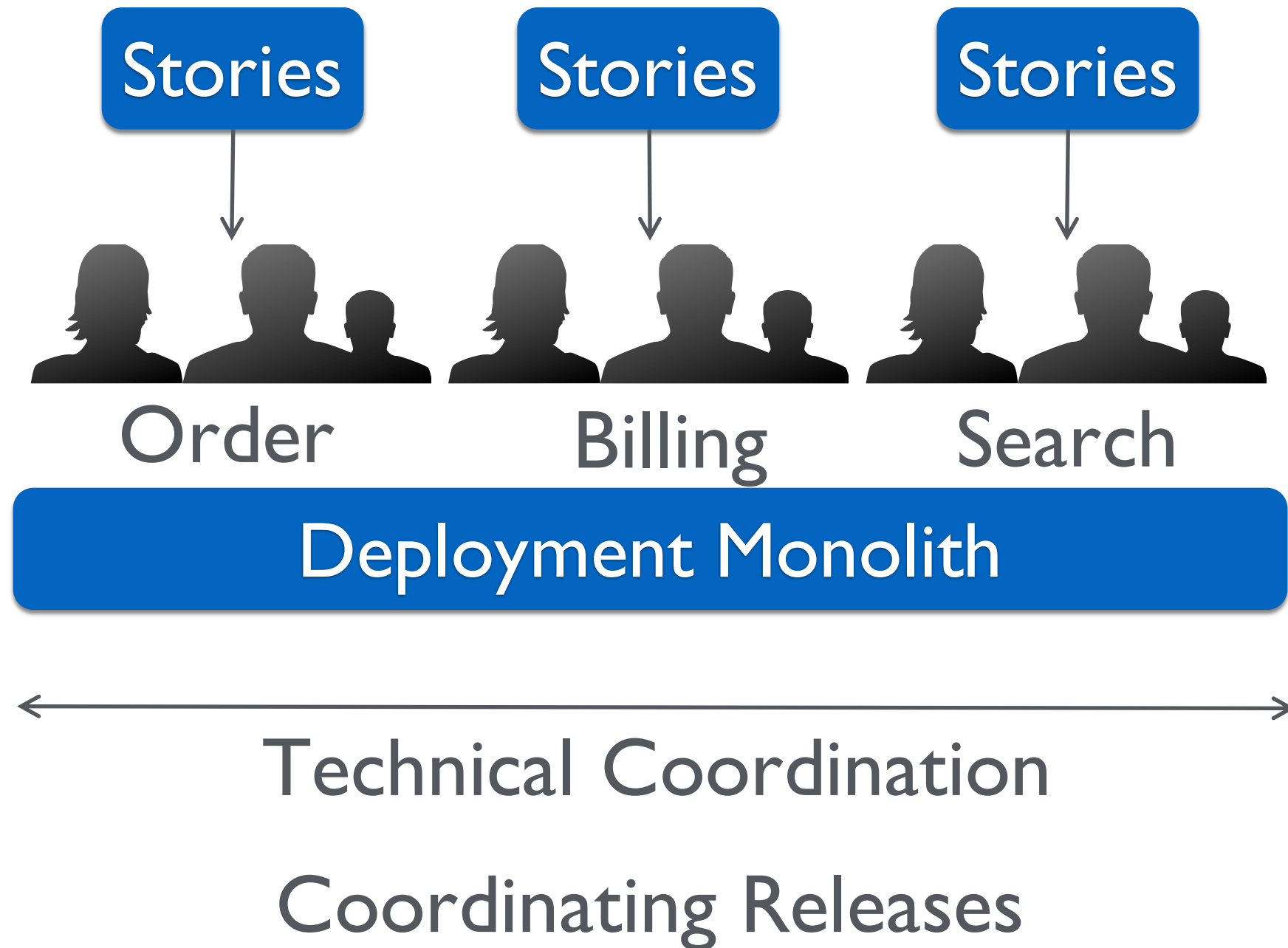
Search

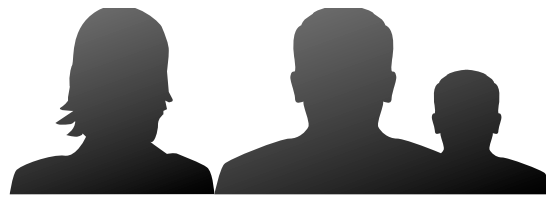


Let architecture drive the organization

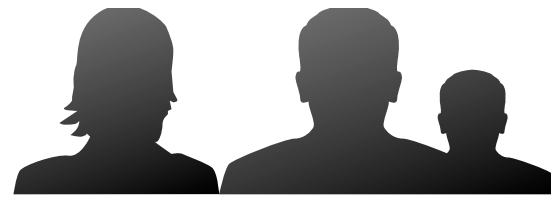
Team for each business feature

Deployment Monolith + Conway's Law

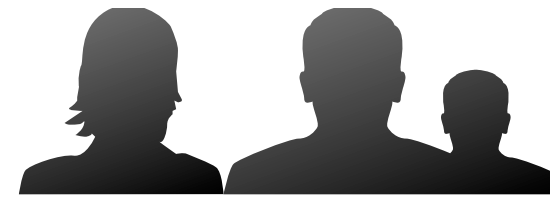




Order



Billing



Search



Synergy Microservices / Conway's Law

One or many Microservices per Team

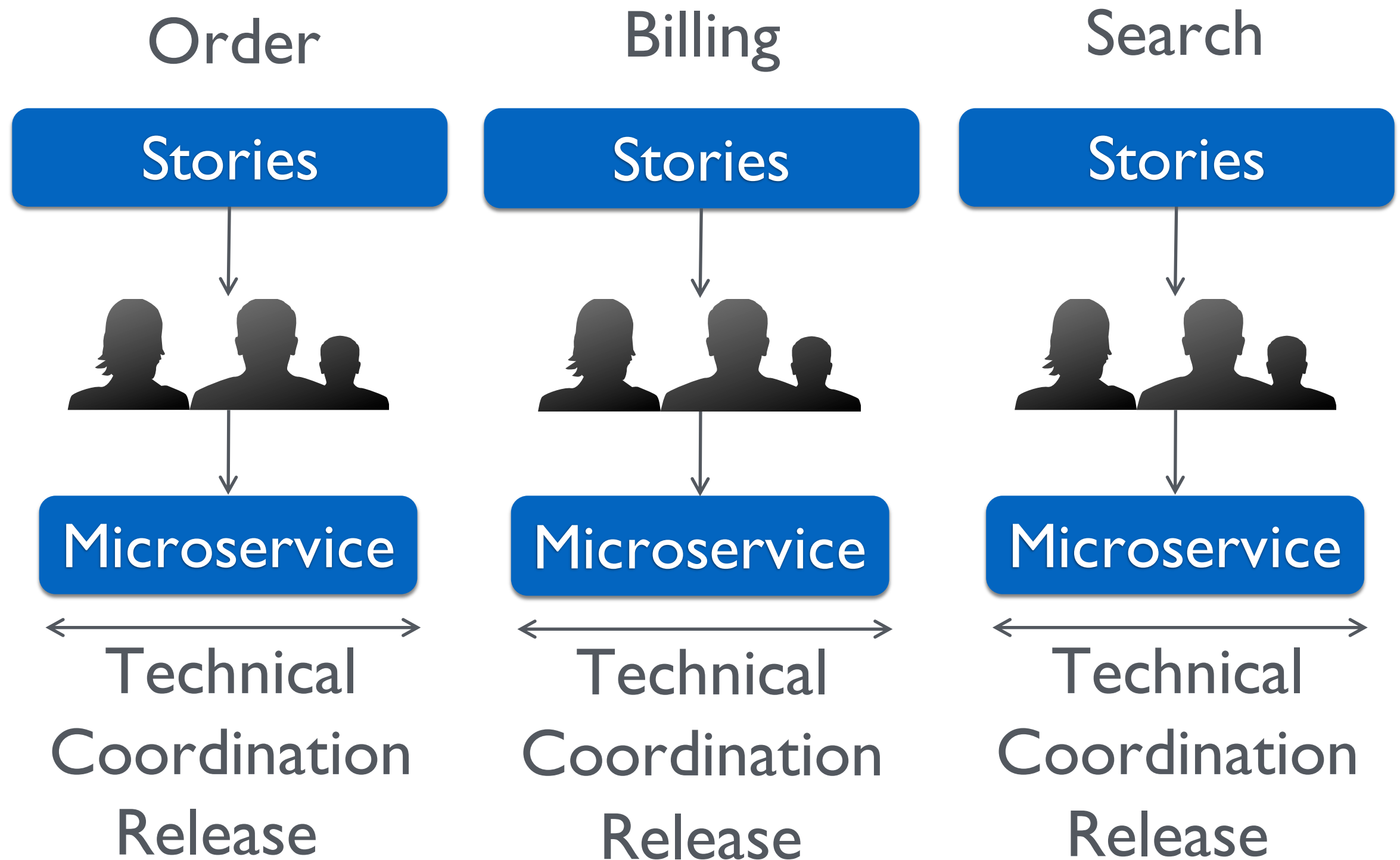
Technology stack per Microservice

Team can deploy without integration

Changes can be deployed independently & quickly

Strong & enforced modularization

Microservices



Microservices: Challenges

› Deployment?

› Monitoring?

› Log Analysis?

Ops

Micro- and Macro- Architecture

Macro-Architecture

- › Global decision
- › Influence the whole system
- › i.e. all Microservices
- › Less Marco-Architecture – less coordination

Micro-Architecture

- › Local decisions
- › Per Microservice

Macro- and Micro-Architecture for Ops

- › Macro

- › Define Log Analysis, Monitoring, Deployment tools
- › Defined globally

- › Micro

- › Concrete deployment, monitoring, logging
- › Defined per team / Microservice

Microservices Teams
need many technical
skills.

Microservices Teams
need Ops.

DevOps

DevOps / Microservices

- › Teams consist primarily of Devs
- › Devs interested in Docker, ELK ...
- › ... lots of experience in Ops
- › More Ops effort

Microservices = huge
opportunity for Ops!

How Much DevOps is Needed?

- › Should all teams do full ops for their services?
- › IMHO optional
- › But: Organizational barriers might cause additional technical complexity

Continuous Delivery: Build Pipeline

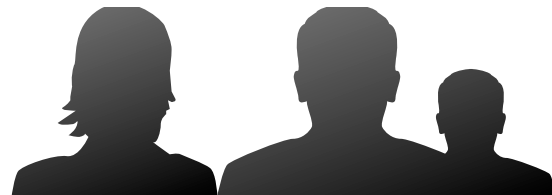
Commit
Stage

Automated
Acceptance
Testing

Automated
Capacity
Testing

Manual
Explorative
Testing

Release



Dev



Ops

Common Technologies
Log, Monitoring, Deployment

DevOps =
Collaboration not
Organization

This changes software
development
fundamentally.

Maintainability

Maintainability: Classical Approach

- › Clear architecture
- › Good code quality
- › Many tests

Dev

Maintainability: New Approach

Fast und Reliable

```
graph LR; A[Commit Stage] --> B[Automated Acceptance Testing]; B --> C[Automated Capacity Testing]; C --> D[Manual Explorative Testing]; D --> E[Release]; E --> F[Fast Feedback]; F --> A;
```

The diagram illustrates a new approach to maintainability. It features a horizontal sequence of five orange rounded rectangular boxes representing stages: 'Commit Stage', 'Automated Acceptance Testing', 'Automated Capacity Testing', 'Manual Explorative Testing', and 'Release'. Above these boxes is a large blue arrow pointing to the right, labeled 'Fast und Reliable'. Below the boxes is a large blue arrow pointing to the left, labeled 'Fast Feedback'. Five small grey arrows point downwards from each of the five boxes to the 'Fast Feedback' arrow, indicating that feedback is collected at every stage of the process.

Commit
Stage

Automated
Acceptance
Testing

Automated
Capacity
Testing

Manual
Explorative
Testing

Release

Fast Feedback

Maintainability: New Approach

- › Continuous Delivery simplifies roll out
- › ...and test them
- › ...and monitor them

DevOps

- › Microservices limit size and risk of changes

Scalability

Scalability:

Classical Approach

- › Implement technical constraints
- › E.g. statelessness

Dev

- › Use appropriate technologies

Performance Risk

- › Hard to really predict performance
- › Load test simulate user on a different environment

Scalability:

Alternative Approach

- › Don't do anything stupid in the architecture
- › Identify bottleneck
- › Eliminate bottleneck
- › Common technique

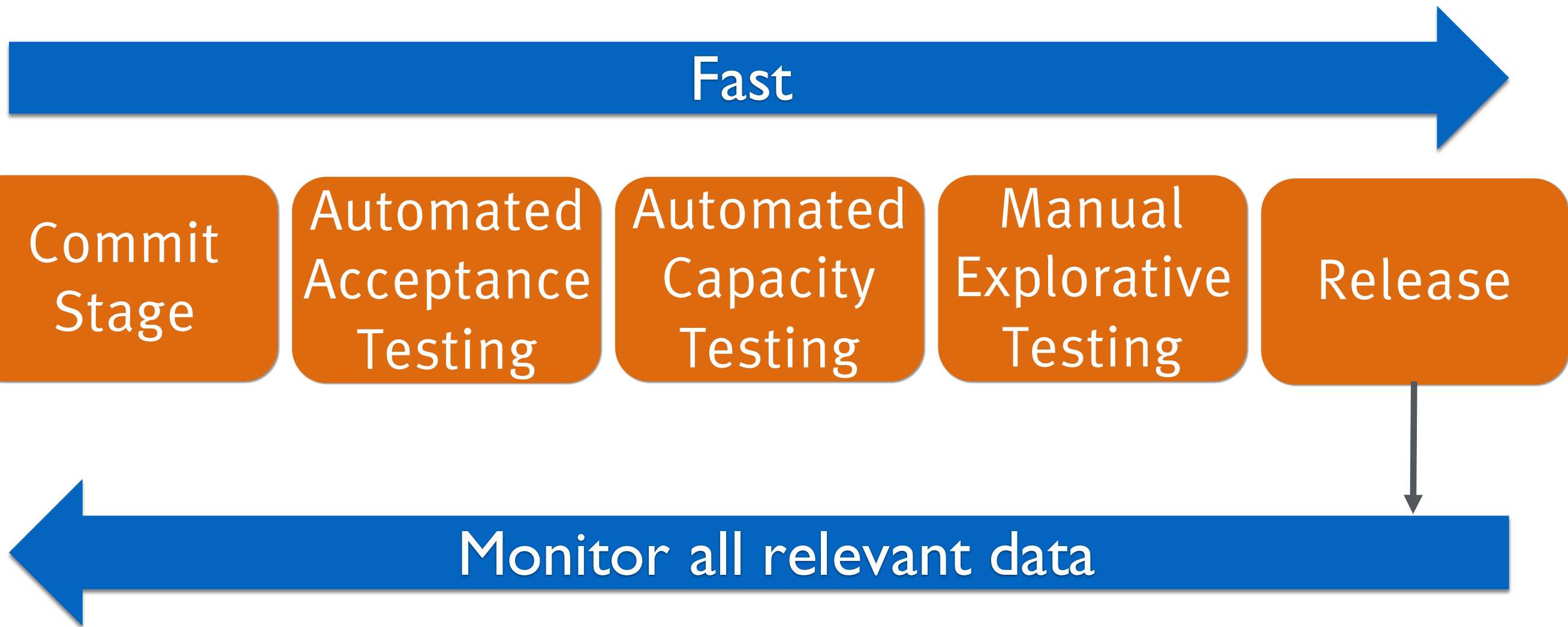
Enable Alternative Approach

- › Monitoring to identify bottleneck

DevOps

- › Fast deployment to eliminate bottleneck

Maintainability: New Approach



Conclusion

Conclusion

- › Synergy:
Microservices and Continuous Delivery
- › Microservices and Continuous Delivery support and require DevOps
- › DevOps = Collaboration
- › Enable alternative approaches e.g. to maintainability and scalability

Thank You!
@ewolff