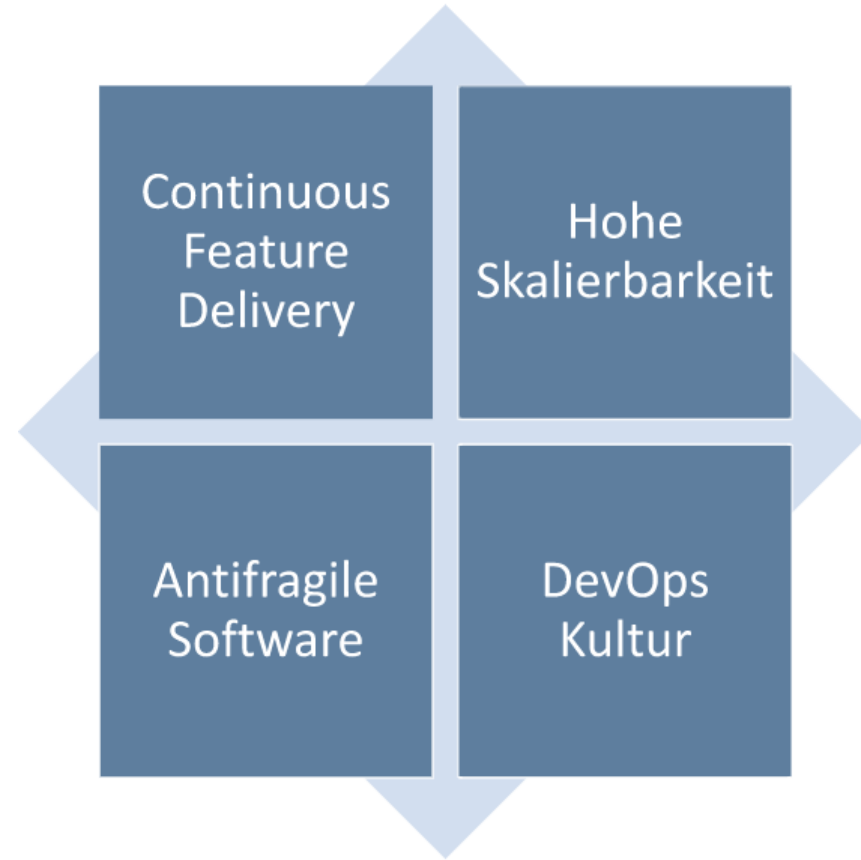


Kapitel: Continuous Delivery



Vorlesung
**CLOUD
COMPUTING**

Treiber für Cloud-native Anwendungen



Continuous Delivery - Definition

ContinuousDelivery



Martin Fowler

30 May 2013

Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.

martinfowler.com

Continuous delivery

From Wikipedia, the free encyclopedia

Continuous delivery (CD) is a [software engineering](#) approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.^[1] It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

Abgrenzung zu Continuous X

Continuous Integration (CI)

- Alle Änderungen werden sofort in den aktuellen Entwicklungsstand integriert und getestet.
- Dadurch wird kontinuierlich getestet, ob eine Änderung inkompatibel mit anderen Änderungen ist.

Continuous Delivery (CD)

- Der Code *kann* zu jeder Zeit deployed werden.
- Er muss aber nicht immer deployed werden.
- D.h. der Code muss (möglichst) zu jedem Zeitpunkt bauen, getestet und ge-debugged sein.

Continuous Deployment

- Jede stabile Änderung wird in Produktion deployed.
- Ein Teil der Qualitätstests finden dadurch in Produktion statt.
 - → Die Möglichkeit mit Fehlern umzugehen muss vorhanden sein (z.B. Canary Release, siehe später)

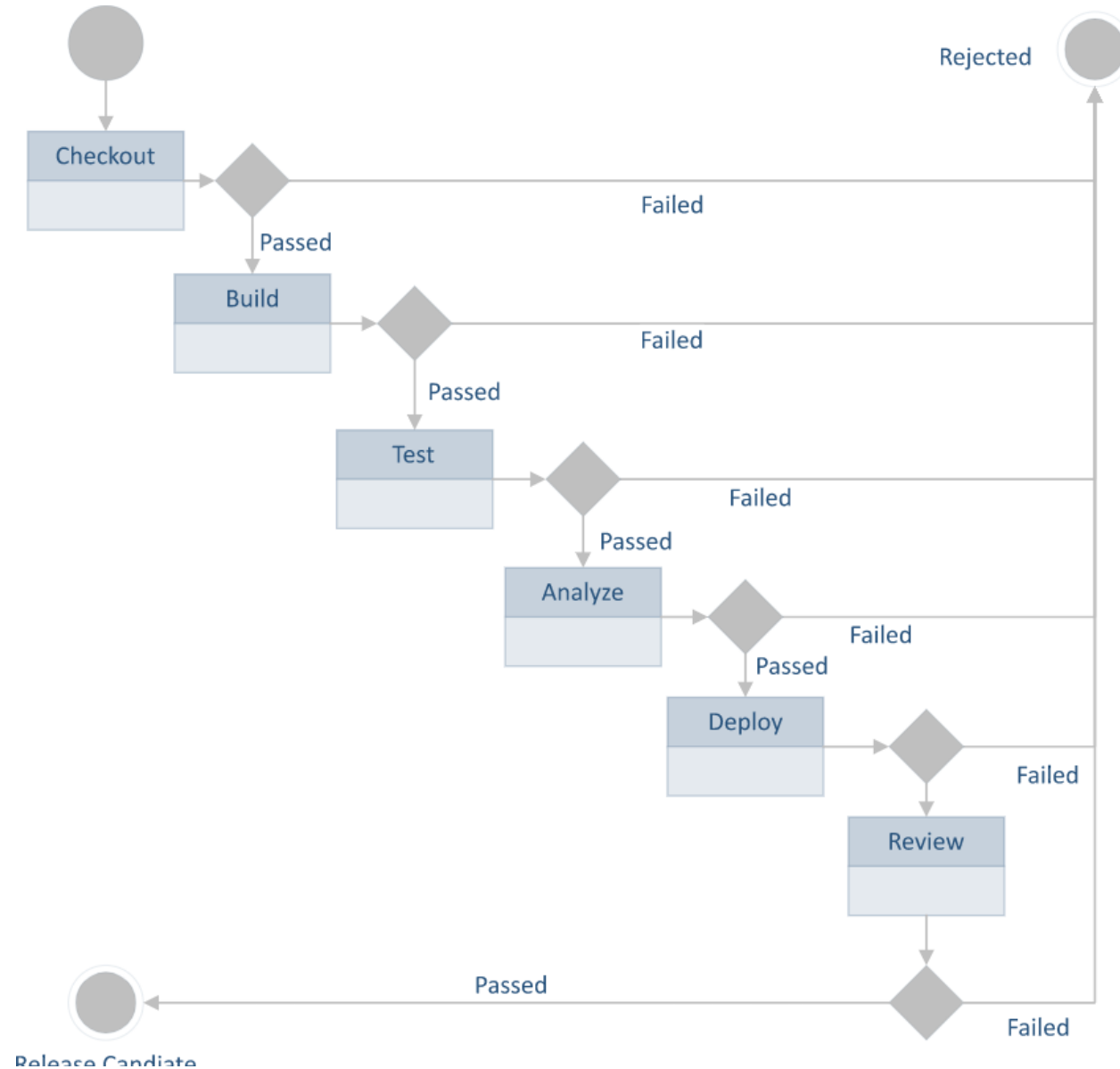
Kriterien für Continuous Deployment

“You’re doing continuous delivery when:

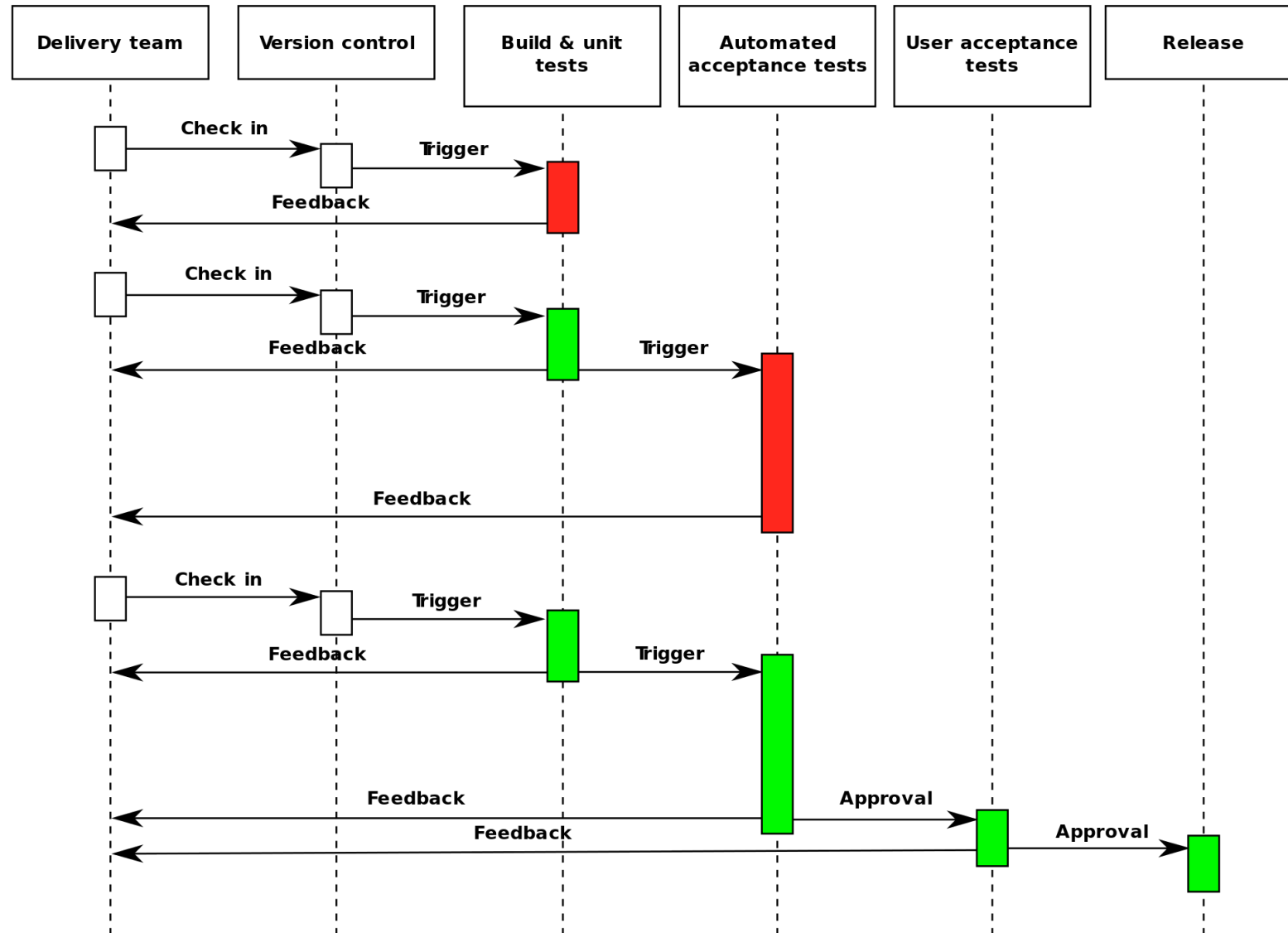
- Your software is deployable throughout its lifecycle
- Your team prioritizes keeping the software deployable over working on new features
- Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them
- You can perform push-button deployments of any version of the software to any environment on demand”

nach M. Fowler / Continuous Delivery working group at ThoughtWorks

Die Continuous Delivery Pipeline

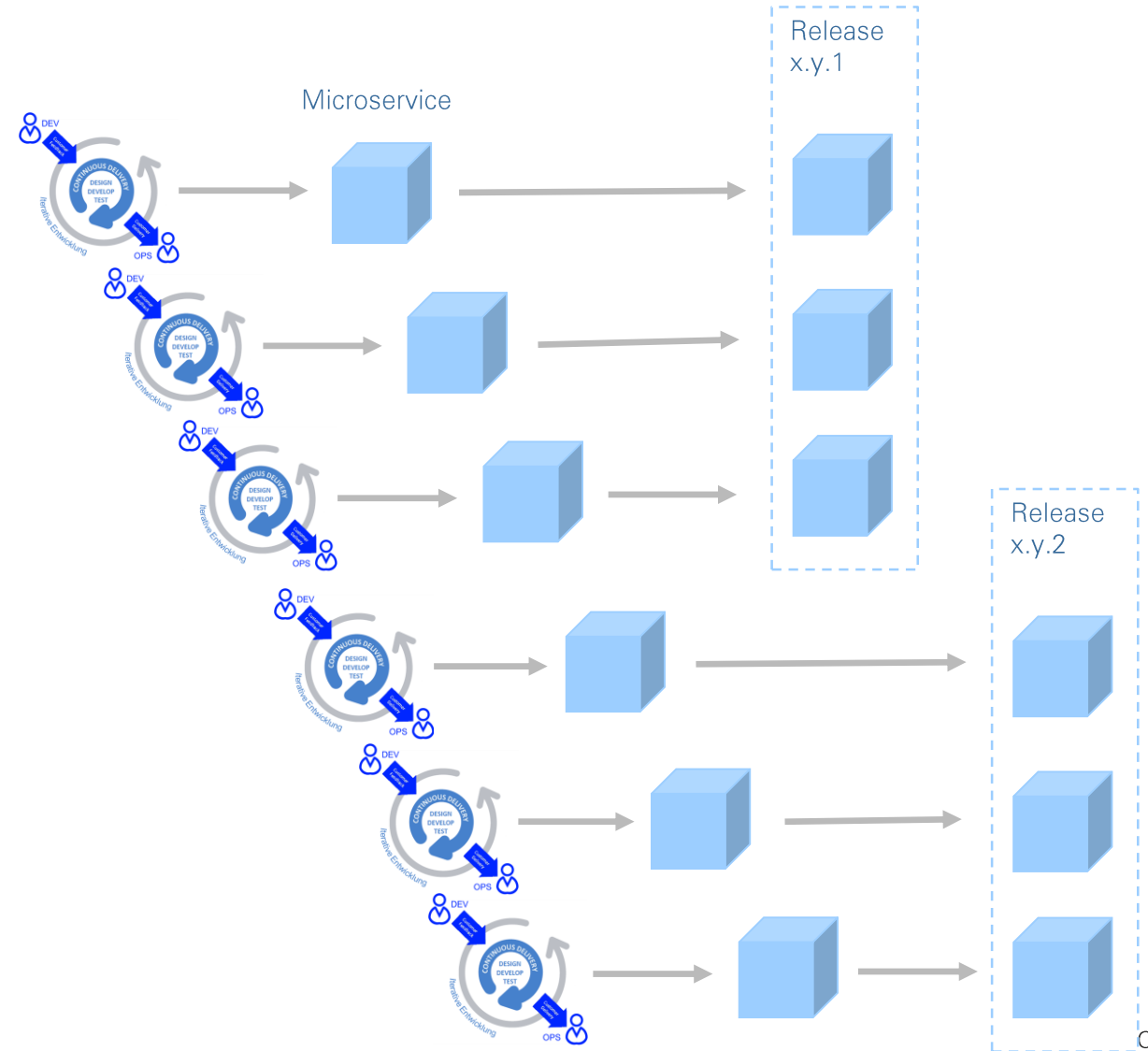
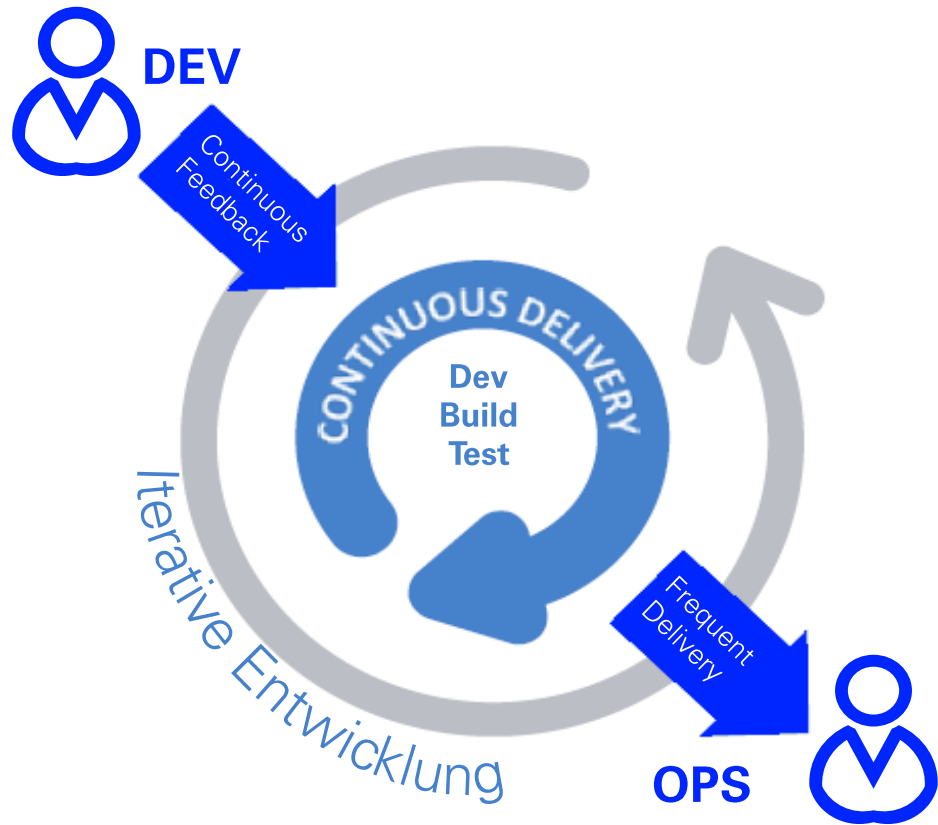


Wichtig ist ein schnelles Feedback an das Entwicklerteam, damit Fehler zügig behoben werden

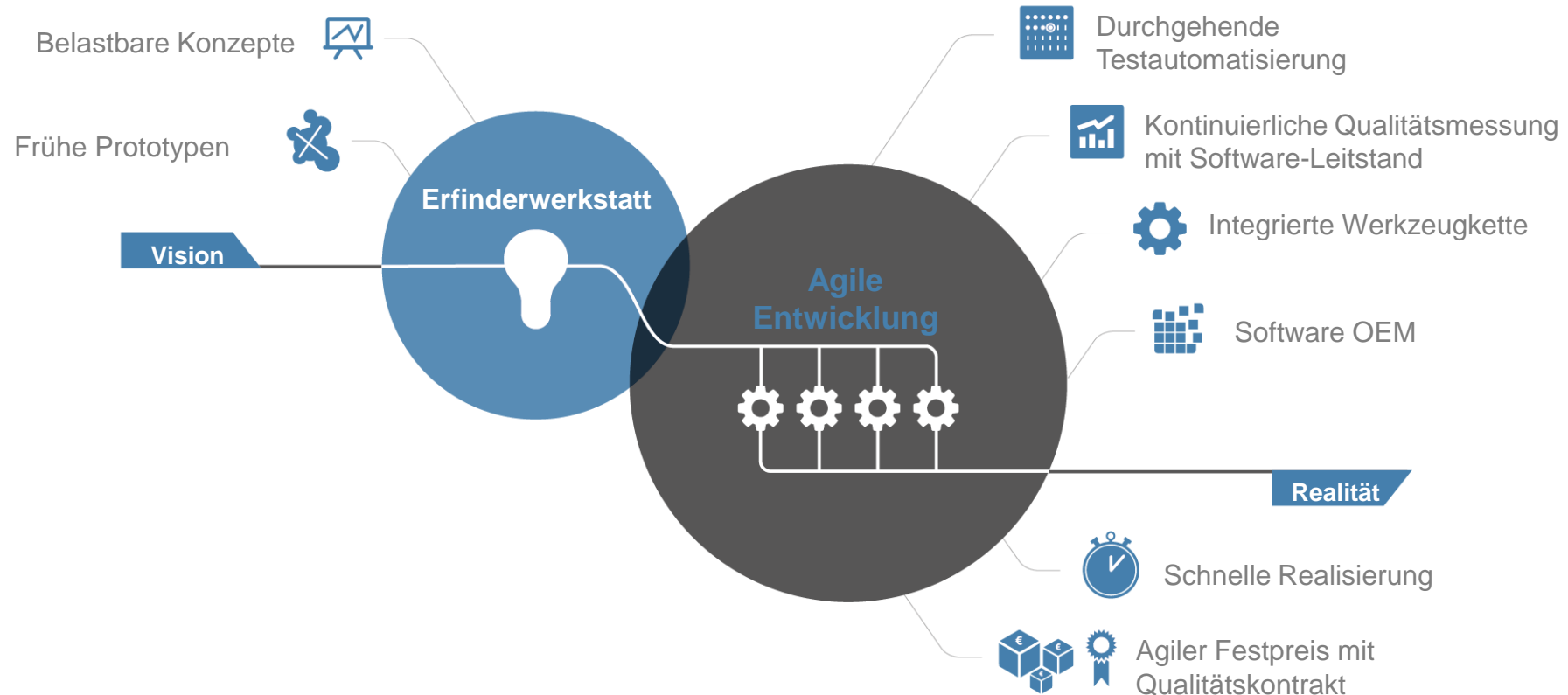


„Continuous delivery process diagramm“
by Grégoire Détrez
Creative Commons 4.0

Continuous Delivery: Build und Release Modell in Microservicearchitekturen

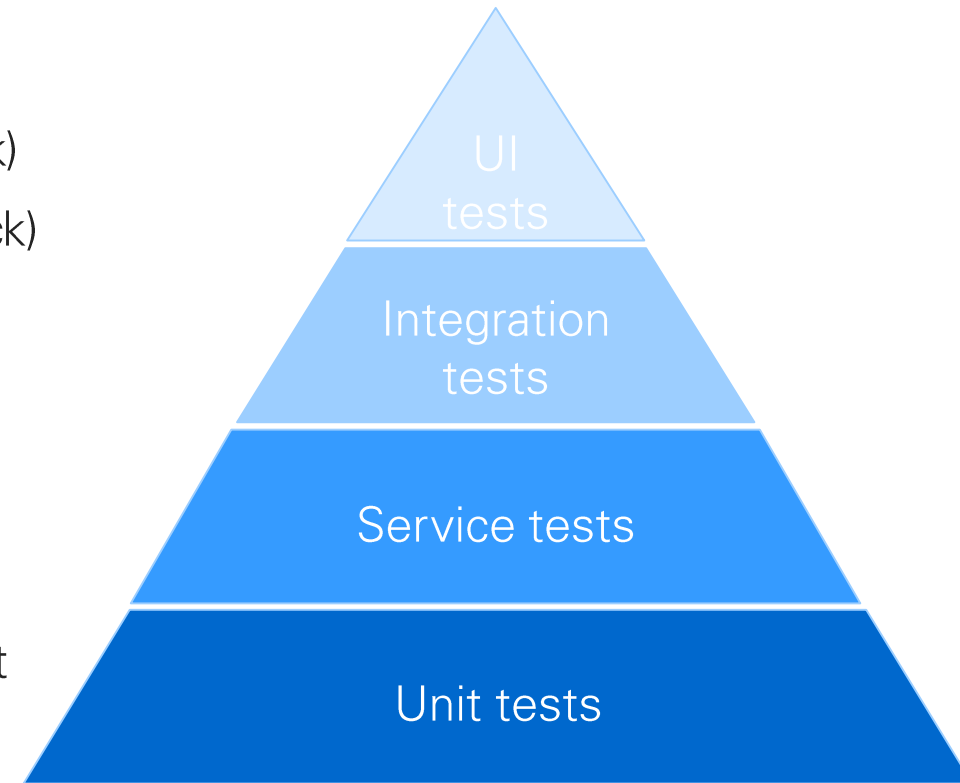


Eine integrierte Werkzeugkette ist ein erheblicher Produktivitätsfaktor



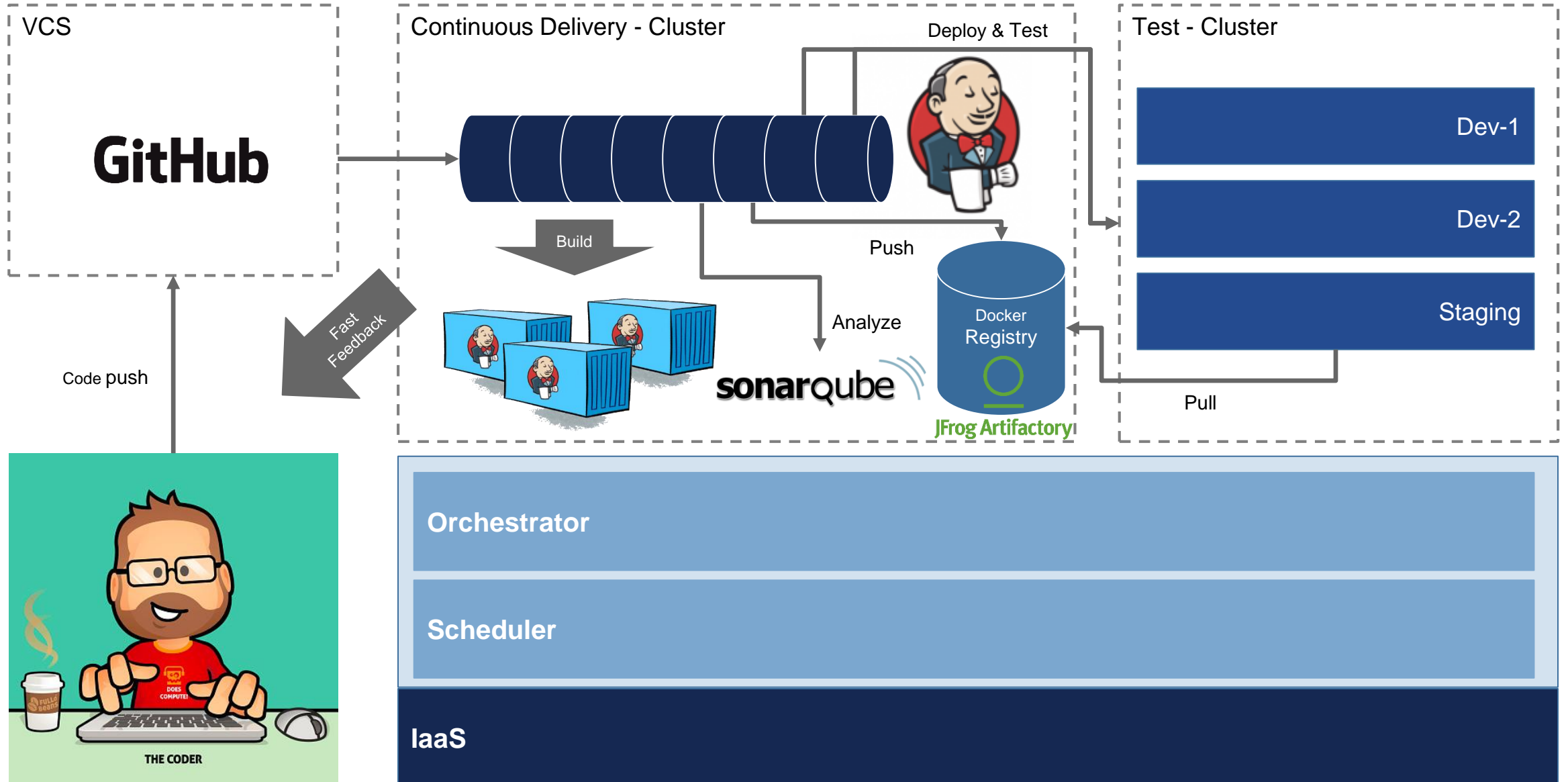
Beispiel-Aufbau einer Test-Pyramide: Sofortige Feedback bei Fehlern

- Unit Tests: Die klassischen Unit Tests (z.B. JUnit, Mockito)
- Service Tests: Tests eines einzelnen Microservices, inkl. der REST-Controller und Client-Calls (z.B. JUnit, Spring MVC Tests, Wiremock)
 - Mocks der anderen Microservices notwendig (z.B. mit Wiremock)
- Integration Tests:
 - Testet die Integration mehrerer Services und deren Interaktion (z.B. JUnit, Spring MVC Tests)
 - Performance Tests: Testet, ob es signifikante Performance-Änderungen gibt (z.B. Gatling)
- UI-Tests: Testet die UI-Funktionalität und deren Zusammenspiel mit dem Backend (z.B. Selenium, Protractor)



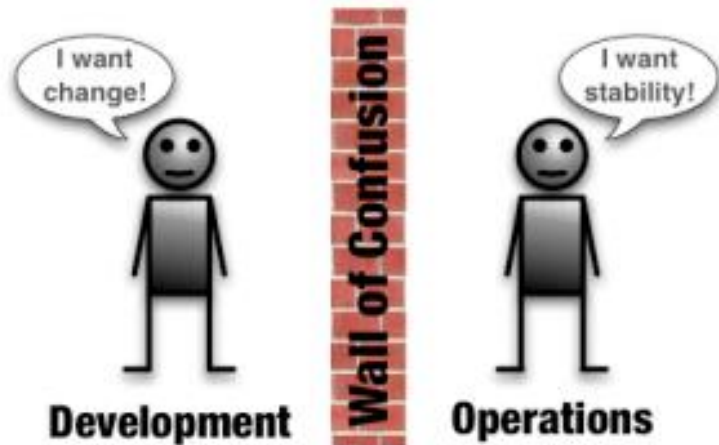
Alle Tests sollten so oft wie möglich ausgeführt werden. Idealerweise bei jedem Commit!

Beispiel einer Continuous Delivery Pipeline



Was ist eigentlich DevOps?

DevOps ist die **verbesserte Integration** von **Entwicklung und Betrieb** durch mehr **Kooperation und Automation** mit dem Ziel, Änderungen schneller in Produktion zu bringen und die MTTR dort gering zu halten. DevOps ist somit eine Kultur.

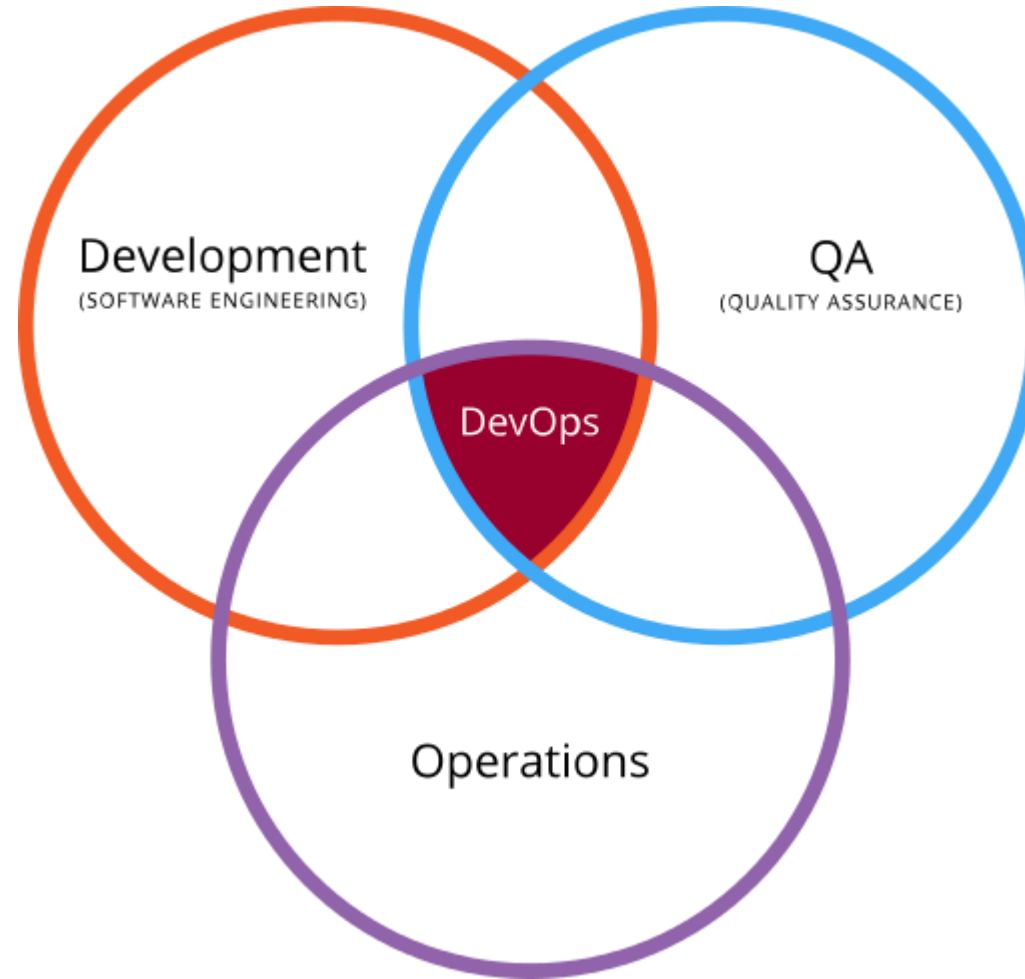


MVP + Feature-Strom

Pro Feature:

- Minimaler manueller Post-Commit-Anteil bis PROD
- Diagnostizierbarkeit des Erfolgs eines Features
- Möglichkeit Feature zu deaktivieren / zurückzurollen

DevOps verbindet DEVelopment, OPerations und Quality Assurance



„Venn diagram showing DevOps“ by Rajiv.Pant/Wylve; Creative Commons 3.0



Continuous Delivery Bausteine



Everything as C<>de

Beispiel: Bootstrapping der CD-Plattform mit DC/OS



Marathon-Deployments

Services:

- Marathon Event Subscriber
- OpsBot



Marathon-Deployments

Platform:

- Jenkins
- SonarQube (mit Datenbank)
- NGINX Reverse Proxy
- Artifactory



Terraform

Infrastructure:

- CoreOS VMs
- DC/OS Nodes (Master, Private, Public)
- NFS

Alles was die CD-Umgebung mit Leben befüllt kommt aus dem VCS.

■ Build-as-Code

- Maven, Gradle, ...
- Beschreibt wie die Anwendung gebaut wird

■ Test-as-Code

- Unit-, Component-, Integration-, API-, UI-, Performance-Tests
- Beschreibt wie das Projekt getestet wird

■ Infrastructure-as-Code

- Docker, Terraform, Vagrant, Ansible, Marathon-Deployments
- Beschreibt wie die Laufzeitumgebungen aufgebaut wird

■ Pipeline-as-Code

- Build-Pipeline per Jenkinsfile
- Buildklammer: Beschreibt alle Schritte bis zur Lauffähigen Installation



Self service & Blueprints

Damit Entwickler schnell arbeitsfähig sind, sind Generatoren und Blueprints wichtig (1/2).

ChatBots sind z.B. eine Lösung zur intuitiven Steuerung von Generatoren.

- Direkte Integration in HipChat / Slack / Mattermost
- Aufträge an den OpsBot werden einfach per Message gestellt
- Feedback von CI/CD Ereignissen und Aufträgen kommen als Antwort zurück



The screenshot displays a chat log with the following messages:

- svc.qabuild · Apr-6 15:51**
 - @TobiasPlacht** Success! Your Project is available on GitHub: <https://github.com/foo-org/hello-world>
 - Creating Github Webhook
 - Creating Jenkins Job
 - @TobiasPlacht** Success! Your Jenkins Job is available at: 147.75.100.119/job/hello-world
 - @TobiasPlacht** To start the Jenkins Job use buildNow hello-world
- qabuild-jenkins · Apr-6 15:51**
 - Deployment with ID 043527e6-9c2e-4135-b9df-c658437c22d3 succeeded at 13:50:54
 - Application: /zwitscher-eureka
- qabuild-jenkins · Apr-6 15:51**
 - Deployment with ID 043527e6-9c2e-4135-b9df-c658437c22d3 completed the following step at 13:50:54
 - Completed Step: **ScaleApplication**
- qabuild-jenkins · Apr-6 15:52**
 - Started Build for: Job 'hello-world/master [1]' (<http://147.75.100.133:8001/job/hello-world/job/master/1/>)
- qabuild-jenkins · Apr-6 15:52**
 - FAILED: Job 'hello-world/master [1]' (<http://147.75.100.133:8001/job/hello-world/job/master/1/>)
- qabuild-jenkins · Apr-6 15:52**
 - Started Build for: Job 'hello-world/new-feature-branch [1]' (<http://147.75.100.133:8001/job/hello-world/job/new-feature-branch/1/>)
- qabuild-jenkins · Apr-6 15:53**
 - SUCCESS: Job 'hello-world/new-feature-branch [1]' (<http://147.75.100.133:8001/job/hello-world/job/new-feature-branch/1/>)

Damit Entwickler schnell arbeitsfähig sind, sind Generatoren und Blueprints wichtig (2/2).

Blueprints & Templates:

- Die Build-Pipelines der (Micro-)Services in einem Projekt sind sich oft sehr ähnlich.
- Blueprints und Templates geben einen Rahmen vor und schaffen implizit Konventionen.
- Beispiele:
 - Durch die Verwendung des Jenkins Pipeline Multibranch Plugins wird für jeden Branch automatisch eine eigene Pipeline angelegt.
 - Identische Anbindung / Integration von Plattform-Komponenten (z.B. SonarQube, Artifactory)

```
stages {  
  
    stage('Send Build started Notification') {  
        steps {  
            slackSend (color: '#FFFF00', message: "STARTED: Job '${env.JOB_NAME}' [${env.BUILD_NUMBER}]")  
        }  
    }  
  
    stage('Build project') {  
        steps {  
            sh './gradlew clean build --info --no-daemon'  
        }  
    }  
  
    stage('Unit Test Reporting') {  
        steps {  
            junit allowEmptyResults: true, testResults: '**/build/test-results/*.xml'  
        }  
    }  
}
```


Continuous Delivery Pipeline für Cloud-native Anwendungen





Diagnosibility

Nicht nur Cloud Native Anwendungen müssen diagnostizierbar sein, sondern auch die CD-Umgebung. (1/2)



Beispiel Log-File Auswertung mit ELK/EFK:

- ELK:
 - Elasticsearch als DB
 - Kibana für Dashboards und Auswertungen
 - Logstash zum einsammeln der verteilten Logdaten
- Auch hier sollte die Plattform (Cloud, Jenkins, ...) und die Applikationsumgebungen integriert werden.
- Für Docker Container, die auf stdout loggen, können Log-Driver (z.B. Fluentd) konfiguriert werden.
 - = EFK (Elasticsearch, Fluentd, Kibana)
- Log-Files in den Containern können per Logstash & Filebeat integriert werden.

Nicht nur unsere Cloud Native Anwendungen müssen diagnostizierbar sein, sondern auch die CD-Umgebung. (2/2)



Beispiel Monitoring mit Prometheus:

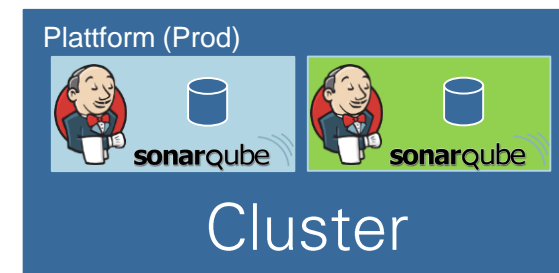
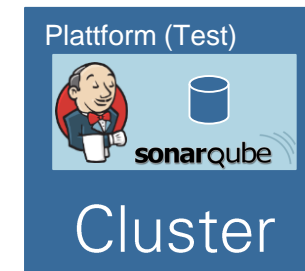
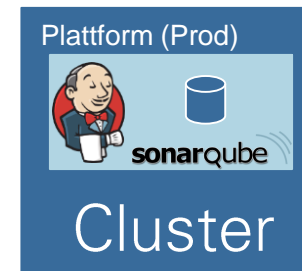
- Prometheus kann für das Monitoring der CD-Plattform sowie für das Monitoring der Applikationsumgebungen benutzt werden.
- Prometheus kann Marathon als Service Discovery nutzen.
- Client Libraries zur Instrumentierung sind für alle wichtigen Programmiersprachen vorhanden.
- Dashboards können einfach mit Grafana angelegt werden.
- Alert-Manager kann Störungen per E-Mail, Pager-Duty, HipChat, Slack ... melden.



CD für CD

Continuous Delivery für Continuous Delivery

- Auch Änderungen und Erweiterungen der CD Plattform müssen getestet werden.
- Durch den „Everything-as-Code“ Ansatz ist das aber sehr einfach:
 - Komplette Klone der Testumgebung (z.B. für Infrastruktur-Tests) können in unter einer Stunde instanziiert werden.
 - Build-Plattform kann für Tests (z.B. bei Jenkins-Update) können als weitere Instanz angelegt werden.

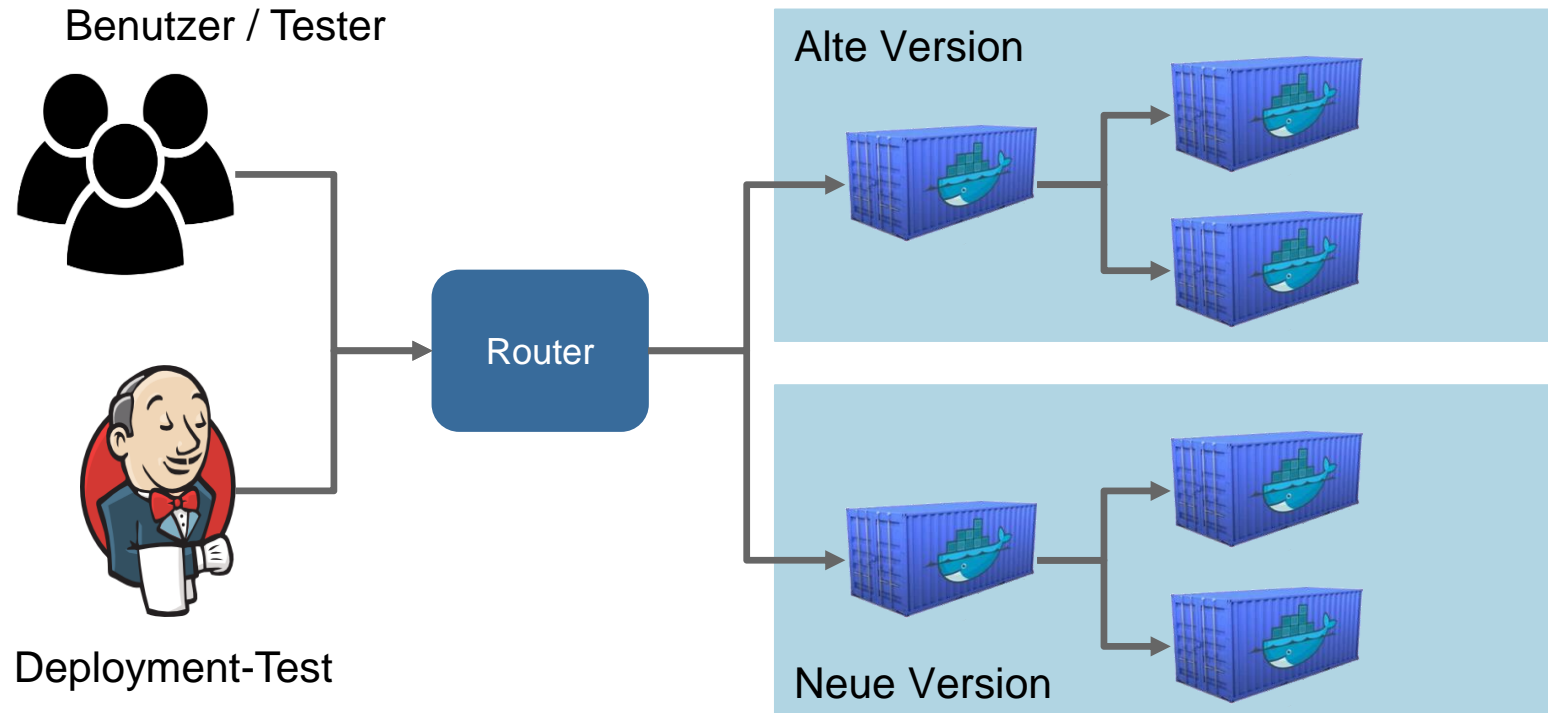




Deployment

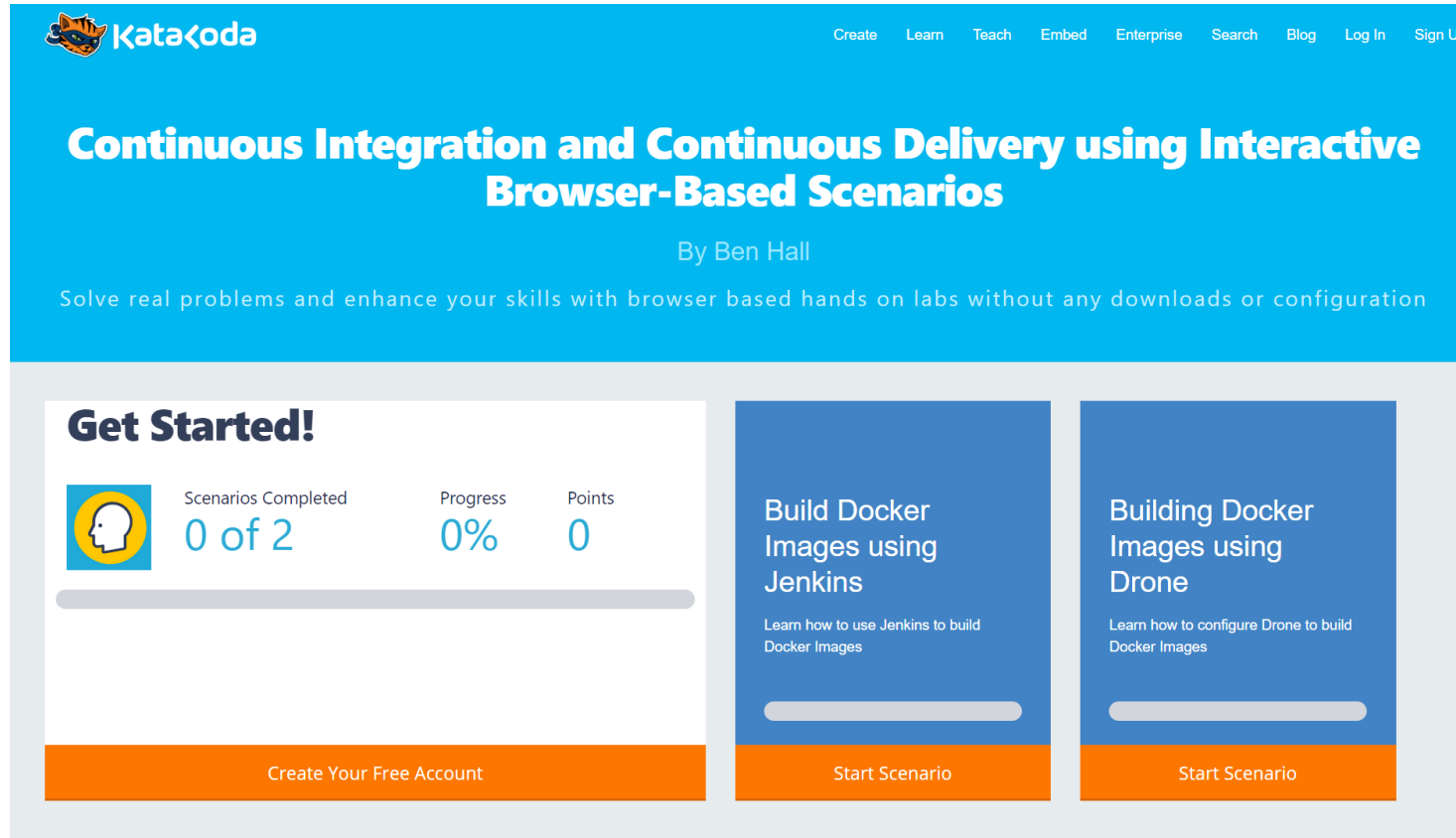
Wie kommen die Microservices in die Testumgebung?

- Canary-Release mit Vamp (Very Awesome Microservices Platform)
- Grundidee:
 - Neue Deployments werden nur von Testern / einem kleinen Teil der Nutzer benutzt.
 - Der Großteil der Benutzer wird erst auf die neue Version geleitet, falls sich diese als Stabil erwiesen hat.



1. Die neue Version wird neben der alten Version deployed. Ein Router steuert, wer welche Version benutzt
2. Nur der Post-Deployment Test aus der Pipeline heraus wird auf die neue Version geleitet.
3. Erst danach wird die erste Teilgruppe der Benutzer / Tester auf umgeleitet.
4. Wenn keine Fehler auftreten, werden alle Benutzer umgeleitet
5. Die alte Version wird offline genommen

Übung: CI/CD



Katacoda Create Learn Teach Embed Enterprise Search Blog Log In Sign Up

Continuous Integration and Continuous Delivery using Interactive Browser-Based Scenarios

By Ben Hall

Solve real problems and enhance your skills with browser based hands on labs without any downloads or configuration

Get Started!

Scenarios Completed	Progress	Points
0 of 2	0%	0

Create Your Free Account

Build Docker Images using Jenkins

Learn how to use Jenkins to build Docker Images

Start Scenario

Building Docker Images using Drone

Learn how to configure Drone to build Docker Images

Start Scenario

<https://katacoda.com/courses/cicd>