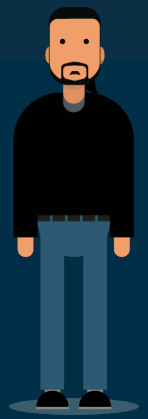
25.11.2020

APPLICATION MIGRATION AND MODERNIZATION ON APPUiO SWISS CONTAINER PLATFORM / OPENSHIFT TECHLAB

Christoph Raaflaub

chrira

@chrira

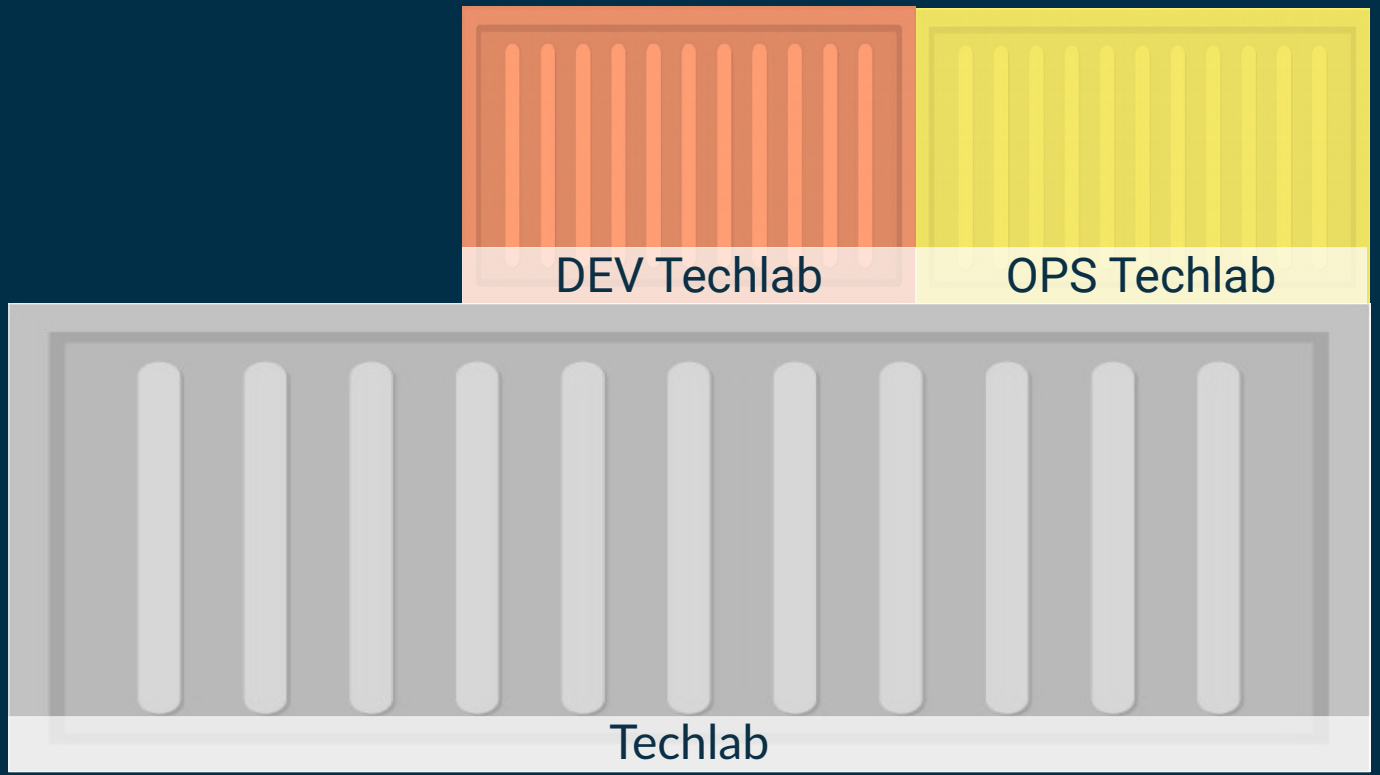


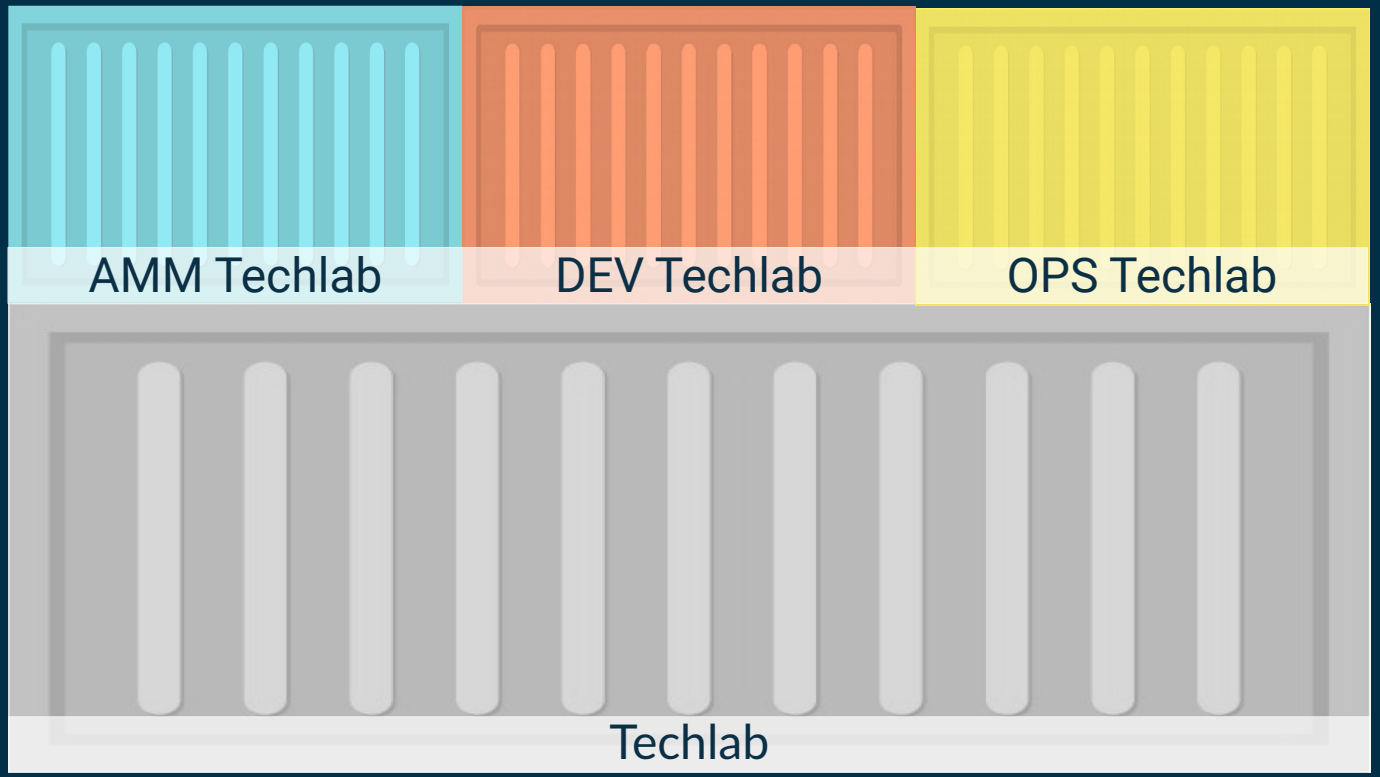
Thomas Philipona

phil-pona

@tphilipona


VSHN The DevOps Company
PUZZLE ITC


DEV Techlab
OPS Techlab
Techlab


AMM Techlab
DEV Techlab
OPS Techlab
Techlab

TECHLAB OBJECTIVES

Migration and Modernization Criteria

Learn Advanced OpenShift and Kubernetes Concepts

Architecture for Container Apps

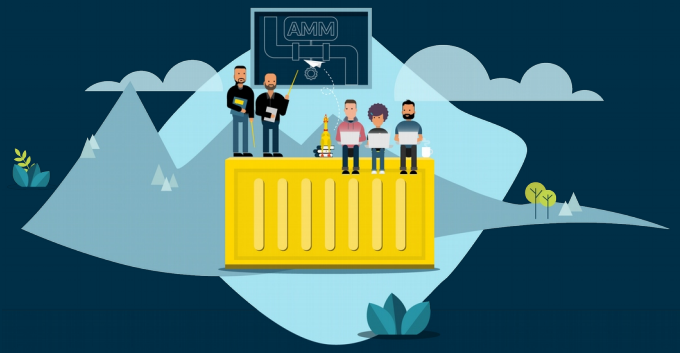
AGENDA

09:00 – 09:10	09:10 – 10:00	10:15 – 10:40	10:40 – 11:20	11:20 – 12:00
Intro Welcome, purpose of the workshop, agenda	**Talk** Introduction AMM LAB 1 Getting started	**Talk** Container Openshift Recap	**Lab** Containerisierung einer Applikation Best Practices	**Talk** Builden und Deployen von Applikationen **Lab** Lab 3 starten

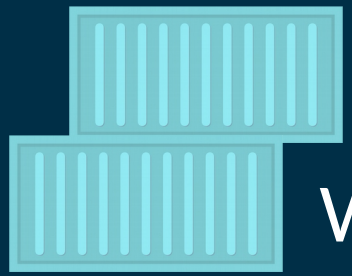
AGENDA

13:00	13.15 – 14:00	14:00 – 14:25	14:25 – 15:30	15:30	15:45 – 16:45	16:45
	Lab Event driven architecture	**Talk** CI/CD mit OpenShift	**Lab** CI/CD mit OpenShift		**Lab** Obser-vability	

APPLICATION MIGRATION AND MODERNIZATION



1



Wie bringt man «legacy» Workload auf OpenShift?

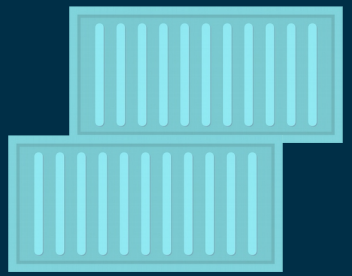
1 Software mit dem Tool "microctl" in Microservices aufteilen

2 Auf OpenShift deployen

3 sich ausruhen

Fragen?

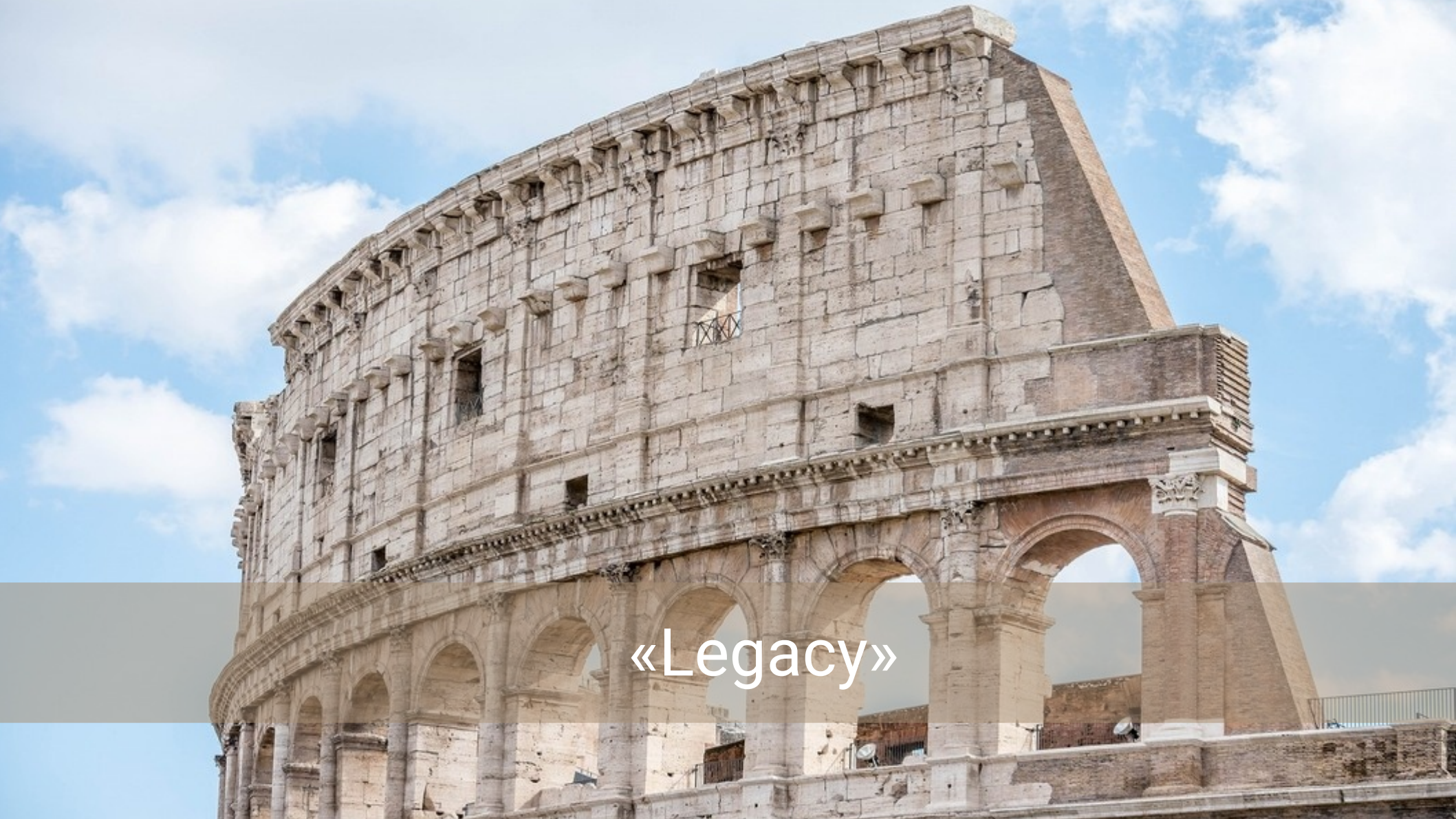
2



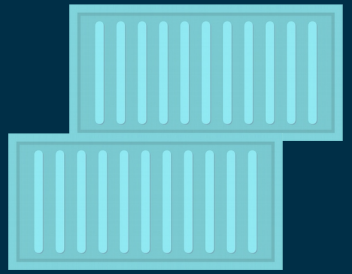
Ok… so einfach ist das nicht.

Legacy Systeme

Historisch gewachsen | Integration | Oft Monolithisch


«Legacy»

3



Transformation Richtung OpenShift

Transformation einer Applikation

BEHALTEN

Im Moment nicht anfassen

Transformation einer Applikation

BEHALTEN

Im Moment nicht anfassen

RUHESTAND

Ausser Betrieb nehmen und Applikation ausschalten

Transformation einer Applikation

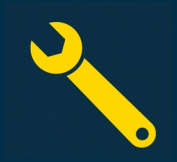
BEHALTEN

Im Moment nicht anfassen

RUHESTAND

Ausser Betrieb nehmen und Applikation ausschalten

PAKETIEREN



Applikation 1:1 nehmen und als Container paketieren

Transformation einer Applikation

BEHALTEN

Im Moment nicht anfassen

RUHESTAND

Ausser Betrieb nehmen und Applikation ausschalten

PAKETIEREN

Applikation 1:1 nehmen und als Container paketieren

REFACTORING

Grundlegendes Applikations-refactoring

Transformation einer Applikation

BEHALTEN

Im Moment nicht anfassen

RUHESTAND

Ausser Betrieb nehmen und Applikation ausschalten

PAKETIEREN

Applikation 1:1 nehmen und als Container paketieren

REFACTORING

Grundlegendes Applikations-refactoring

REPLACE

Applikation austauschen durch eine andere

Transformation einer Applikation

BEHALTEN

Im Moment nicht anfassen

RUHESTAND

Ausser Betrieb nehmen und Applikation ausschalten

PAKETIEREN

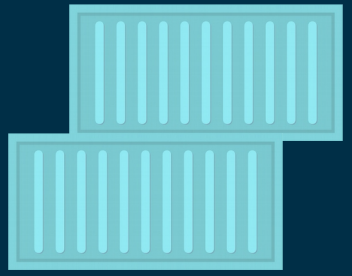
Applikation 1:1 nehmen und als Container paketieren

REFACTORING

Grundlegendes Applikations-refactoring

REPLACE

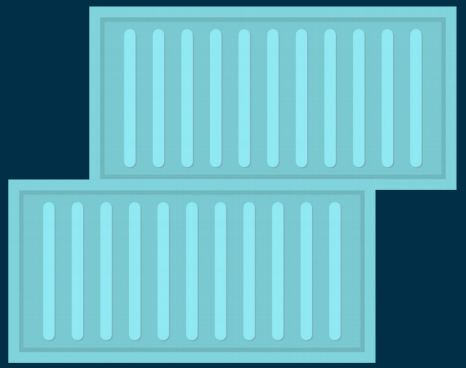
Applikation austauschen durch eine andere



Paketieren von bestehenden Applikationen

Welche Applikationen eignen sich besonders?

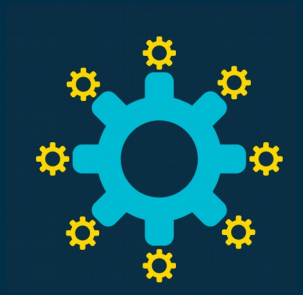
Welche Applikationen eignen sich besonders?



Container

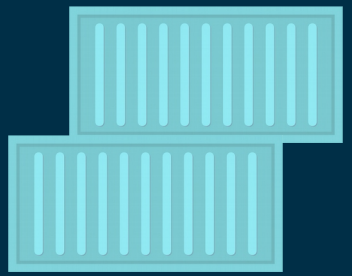
HTTP(S)

Bei welchen Applikationen ist Vorsicht geboten?



Ressourcenhungrig

Datenbanken



Refactoring von bestehenden Applikationen / Neue Applikationen

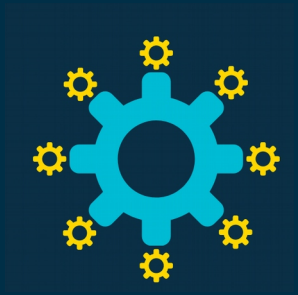
Was läuft gut auf Container Plattformen?

Cloud Native

Stateless

Was läuft gut auf Container Plattformen?

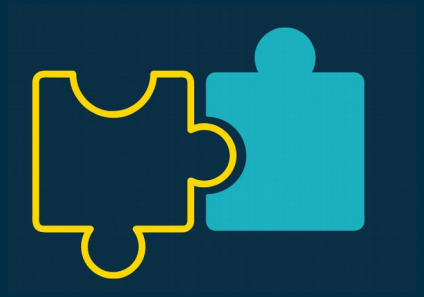
kurze Startzeiten



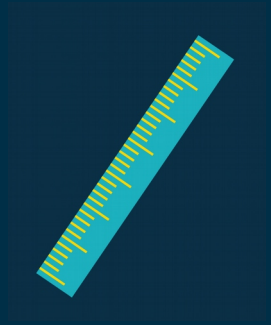
Microservices

Automatisierung

Was läuft gut auf Container Plattformen?



Integration



Skalierung

Konfiguration

Best Practices
12Factor App

Methode um Software-as-a-Service Apps zu bauen

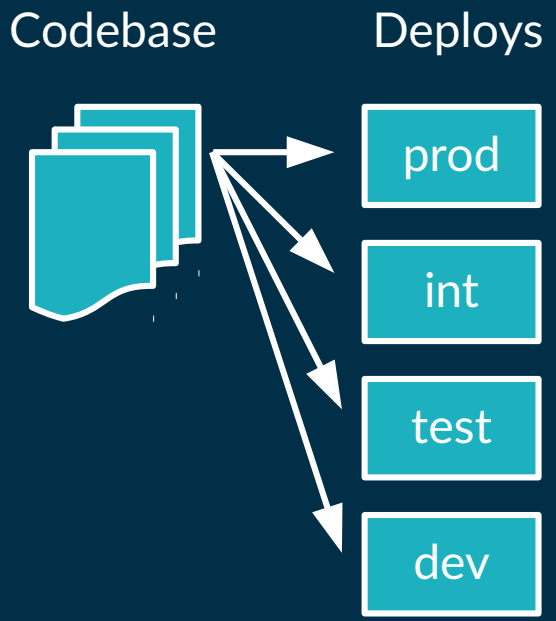
- maximale Portierbarkeit
- das Deployment auf Cloud-Plattformen
- Continuous Deployment ermöglichen
- Skalierbarkeit

12Factor App
1. Codebase

Codebase ist in einem **einzigen** Repo.

Mehrere Codebases → Verteiltes System

Mehrere Apps teilen sich denselben Code → Libraries als Dependency



12Factor App
2. Dependencies

Eine Zwölf-Faktor-App **verlässt sich nie auf die Existenz von systemweiten Paketen** (curl, ImageMagick, ...).

Sie deklariert alle Abhängigkeiten vollständig und korrekt über eine **Abhängigkeitsdeklaration** (Gemfile, pom.xml, package.json, ...)

Zur Laufzeit wird ein Werkzeug zur **Isolation** von Abhängigkeiten benutzt.

Gemfile.lock

```
GEM
  remote: https://rubygems.org/
  specs:
    actionmailer (4.2.5.1)
      actionpack (= 4.2.5.1)
      actionview (= 4.2.5.1)
      activejob (= 4.2.5.1)
      mail (~> 2.5, >= 2.5.4)
      rails-dom-testing (~> 1.0, >= 1.0.5)
    actionpack (4.2.5.1)
```

pom.xml

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.6.3</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.6.3</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.6.3</version>
</dependency>
```

12Factor App
3. Konfiguration

Strikte Trennung der Konfiguration vom Code.

Die Konfiguration ändert sich deutlich von Deploy zu Deploy, ganz im Gegensatz zum Code.

- Datenbank Verbindungsinformationen
- Credentials zu externen Diensten
- Hostnamen und IP Adressen

12Factor App
6. Prozesse

Zwölf-Faktor-Apps sind **zustandslos** und **shared nothing.**

Alle Daten werden in unterstützenden Diensten gespeichert, normalerweise einer Datenbank.

Der RAM oder das Dateisystem des Prozesses kann als **kurzfristiger** Cache für die Dauer **einer Transaktion** verwendet werden.

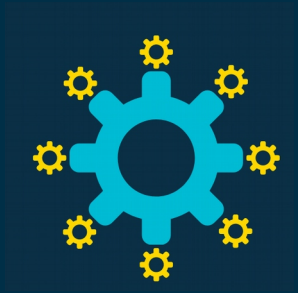
12Factor App
9. Disposability (Einweggebrauch)

Robuster mit **schnellem** Start und problemlosen Stopp

Die Prozesse einer Zwölf-Faktor-App **können weggeworfen werden**, sie können also schnell gestartet und gestoppt werden.

Prozesse **fahren ohne Schwierigkeiten** und mit **möglichst geringen Nebeneffekten** herunter.

Microservices Architektur



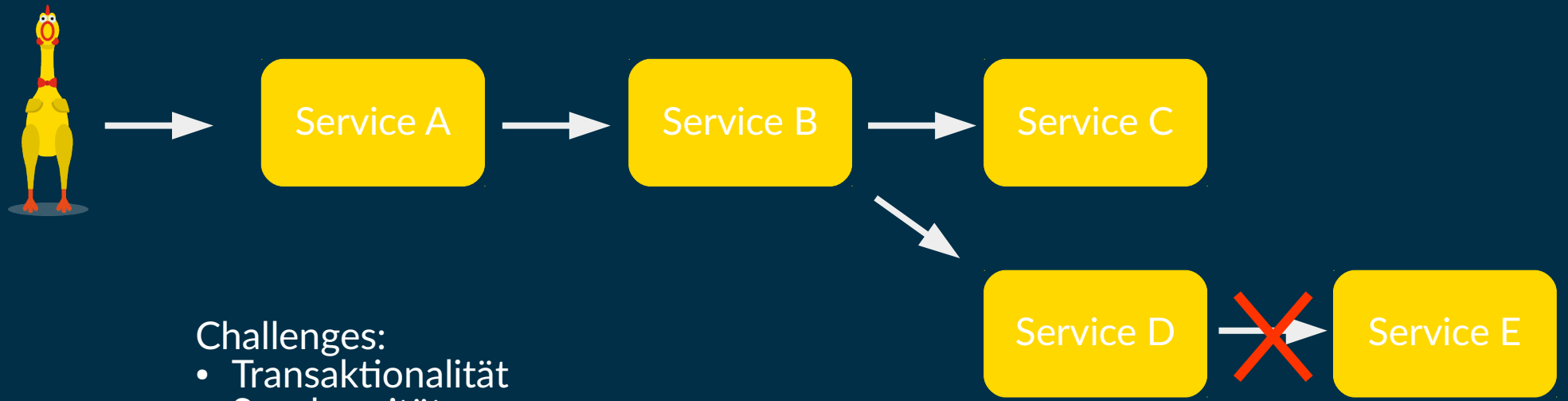
Loosely Coupled

- Deployment
- Communication
- Organisation

Small Services

Scale

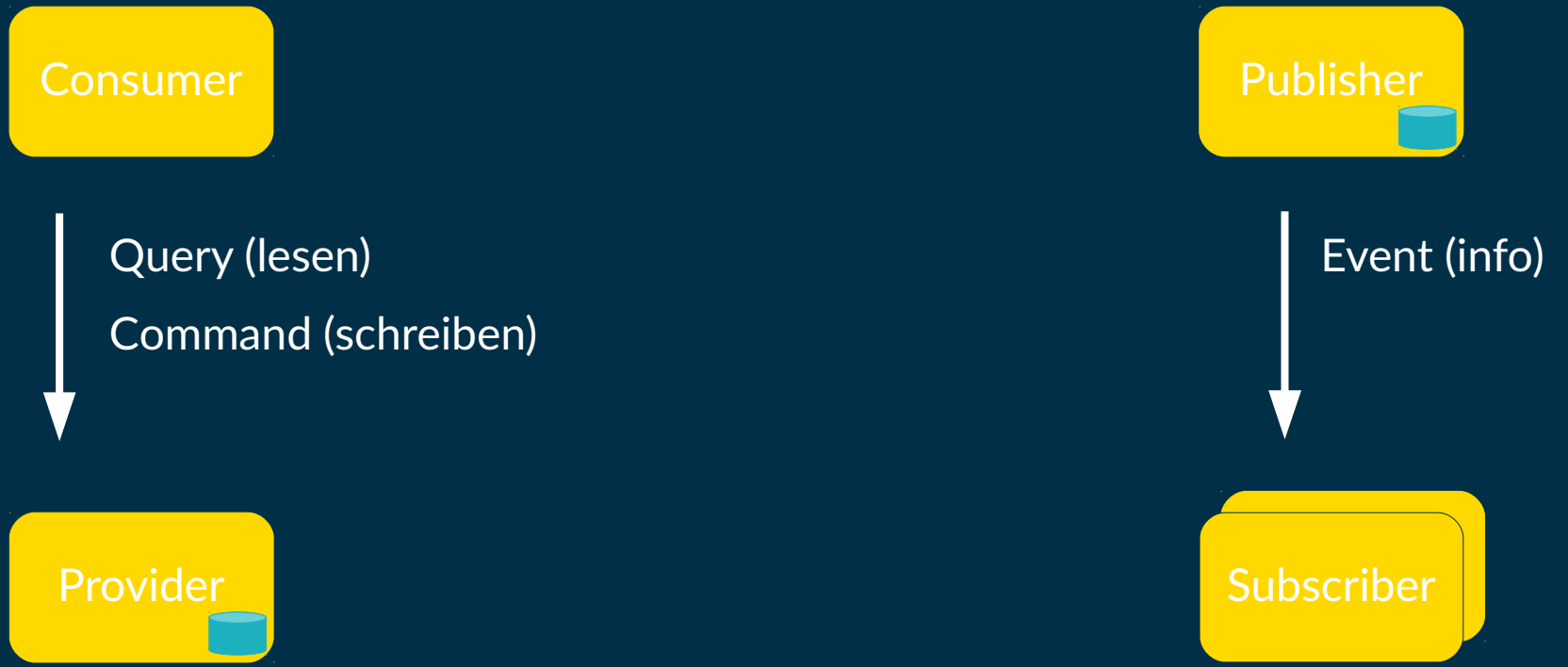
Microservices Architektur



Challenges:

- Transaktionalität
- Synchronität
- Kaskadierende Fehler
- Aufschaukelnde Timeouts
- ...

Event Driven Architektur



Event Driven Architektur

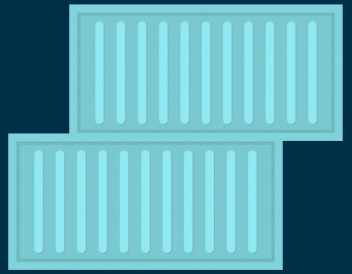
Publisher informiert über seinen Zustand. Subcriber können darauf reagieren, kein Befehl.

Asynchrone Kommunikation

Paradigmenwechsel

Orchestration vs. Choreography

4



Praxiserfahrungen und Bestpractices

keep everything you need to build,
deploy, test, & release in version control

Build and Run Robust Applications

Fehlertoleranz

Health checks

Logging

Exceptionhandling (Fehler frühzeitig erkennen)

Hohe Test Coverage

If It Hurts, Do It More Frequently, and Bring the Pain Forward

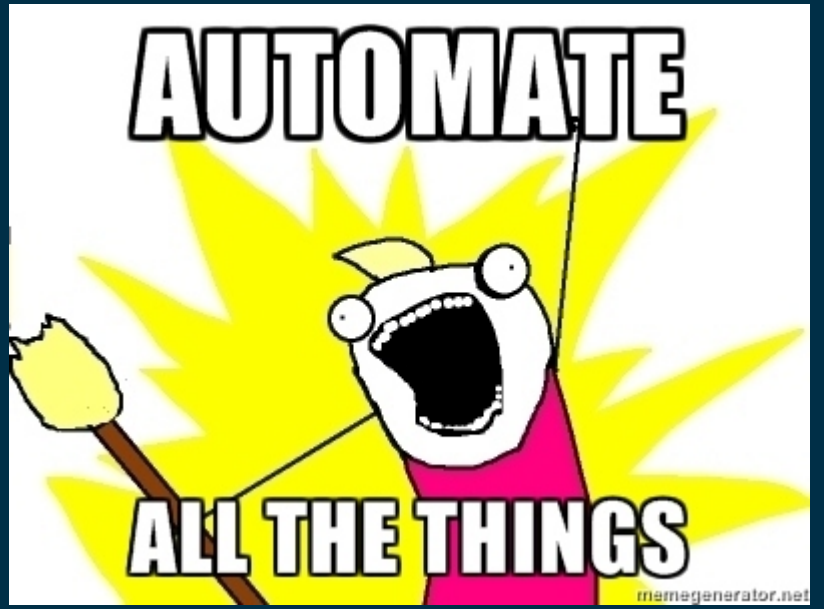
Source: Jez Humble, David Farley, Continuous Delivery, Pearson Education, Inc. 2011

Automatisierung

Robuste Workflows bis in Produktion

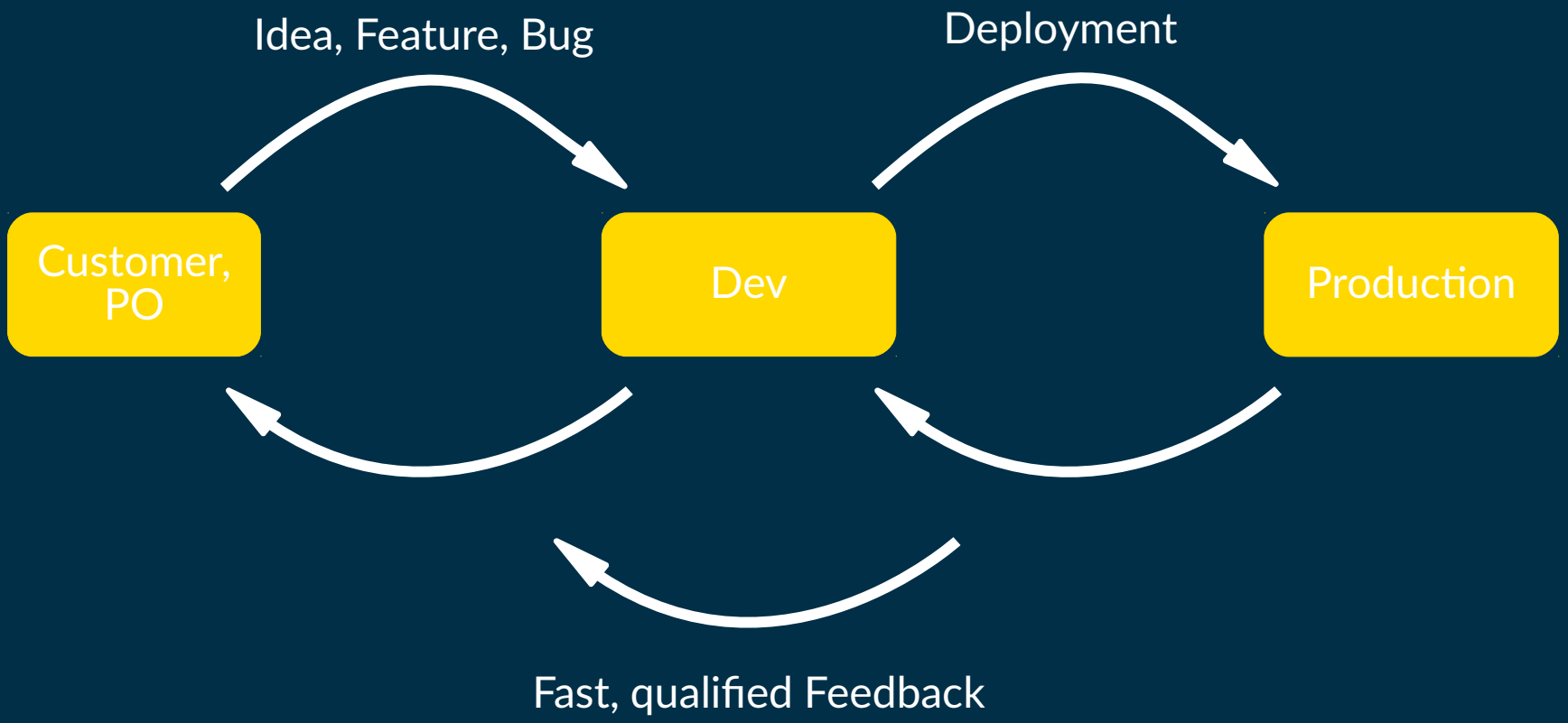
So viel wie möglich automatisieren

- Build
- Test
- Release
- Deployment



Checklisten eignen sich als Anhaltspunkt für Automatisierung

Fast Feedback is Key



Monitoring - Healthchecks

Sicht in die Plattform

Fast Feedback

Einfach in die Infrastruktur integrierbar

Vorsicht!

Lifecycle

Integraler Bestandteil des Dev Prozesses

Applikationskomponenten

Base Images

Resource Management

Used vs. Requested vs. Limit

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape l.cncf.io has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally

cncf.io/enduser

WHAT IS CLOUD NATIVE?

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

l.cncf.io

v20200501

1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing
- Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using GitOps paradigms such as continuous and progressive delivery and MLops

argo — CNCF Incubating

3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: cncf.io/ck
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application

kubernetes — CNCF Graduated
HELM — CNCF Graduated

4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

Prometheus — CNCF Graduated
fluentd — CNCF Graduated
JAEGER — CNCF Graduated
OPENTRACING — CNCF Incubating

5. SERVICE PROXY, DISCOVERY, & MESH

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

envoy — CNCF Graduated
CoreDNS — CNCF Graduated
LINKERD — CNCF Incubating

6. NETWORKING, POLICY, & SECURITY

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general-purpose policy engine with uses ranging from authorization and admission control to data filtering. Falco is an anomaly detection engine for cloud native.

CNI — CNCF Incubating
Open Policy Agent — CNCF Incubating
falco — CNCF Incubating

7. DISTRIBUTED DATABASE & STORAGE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding. Rook is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. Serving as the "brain" of Kubernetes, etcd provides a reliable way to store data across a cluster of machines. TiKV is a high performant distributed transactional key-value store written in Rust.

Vitess — CNCF Graduated
ROOK — CNCF Incubating
etcd — CNCF Incubating
TiKV — CNCF Incubating

8. STREAMING & MESSAGING

When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues. CloudEvents is a specification for describing event data in common ways.

gRPC — CNCF Incubating
NATS — CNCF Incubating
cloudevents — CNCF Incubating

9. CONTAINER REGISTRY & RUNTIME

Harbor is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.

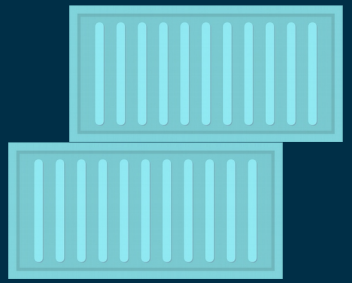
containerd — CNCF Graduated
HARBOR — CNCF Incubating
cri-o — CNCF Incubating

10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

TUF — CNCF Graduated
Notary — CNCF Incubating

5



Fazit

Es kommt drauf an...

Fokus auf Applikationen

Als ersten Schritt OpenShift als Runtime Plattform anschauen.

Container / OpenShift Knowhow ist der Schlüssel zum Erfolg.

Nicht zu viel auf einmal machen.

Fazit

Applikation für Applikation, Step by Step

Java Applikationen eignen sich sehr gut

Das Modern von heute ist das Legacy von morgen.


DANKE

Techlab – Hands-on 1

Lab resources:

- siehe Chat