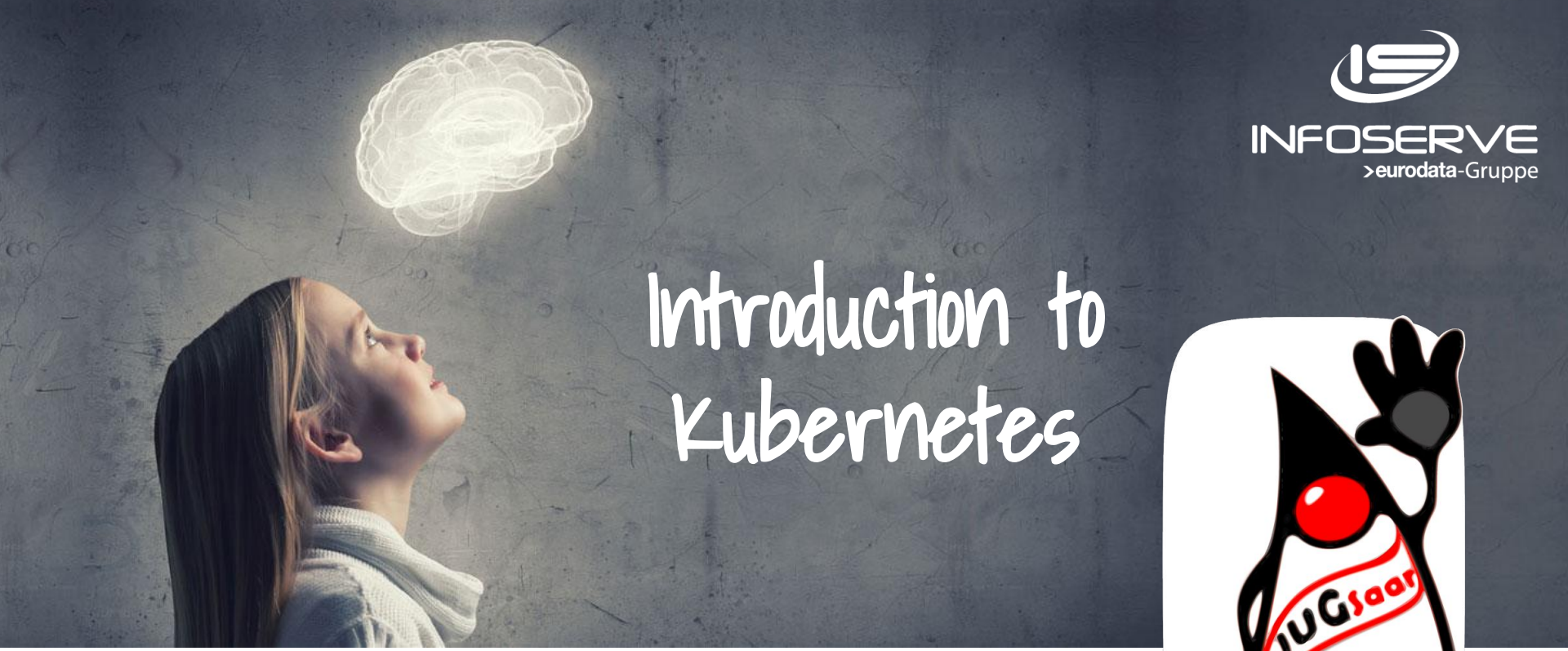


INFOSERVE
>eurodata-Gruppe

Introduction to Kubernetes

JUG Saar Meetup 37 | 23.08.2018

THOMAS DARIMONT | Software Architect | eurodata AG
DR. PHILIPP WALTER | Leiter IT | INFOSERVE GmbH

Agenda

JUG Saar Meetup 37 | 23.08.2018

INFOSERVE
>eurodata-Gruppe

18:00	Begrüßung und Einführung in Kubernetes
18:30	Demo 1: lokaler Kubernetes-Cluster mit Vagrant Demo 2: Cloud Native Spring Boot App Demo 3: Rolling Upgrades
19:15	Pizza!
19:45	Demo 4: HTTP Session Clustering mit Hazelcast Demo 5: Keycloak Cluster als Helm-Chart Demo 6: Kubernetes-Cluster auf VMs selbst einrichten
20:15	Frage- und Feedbackrunde
20:30	Ende der Veranstaltung

INFOSERVE
>eurodata-Gruppe

VORWORT

Einführung in Kubernetes

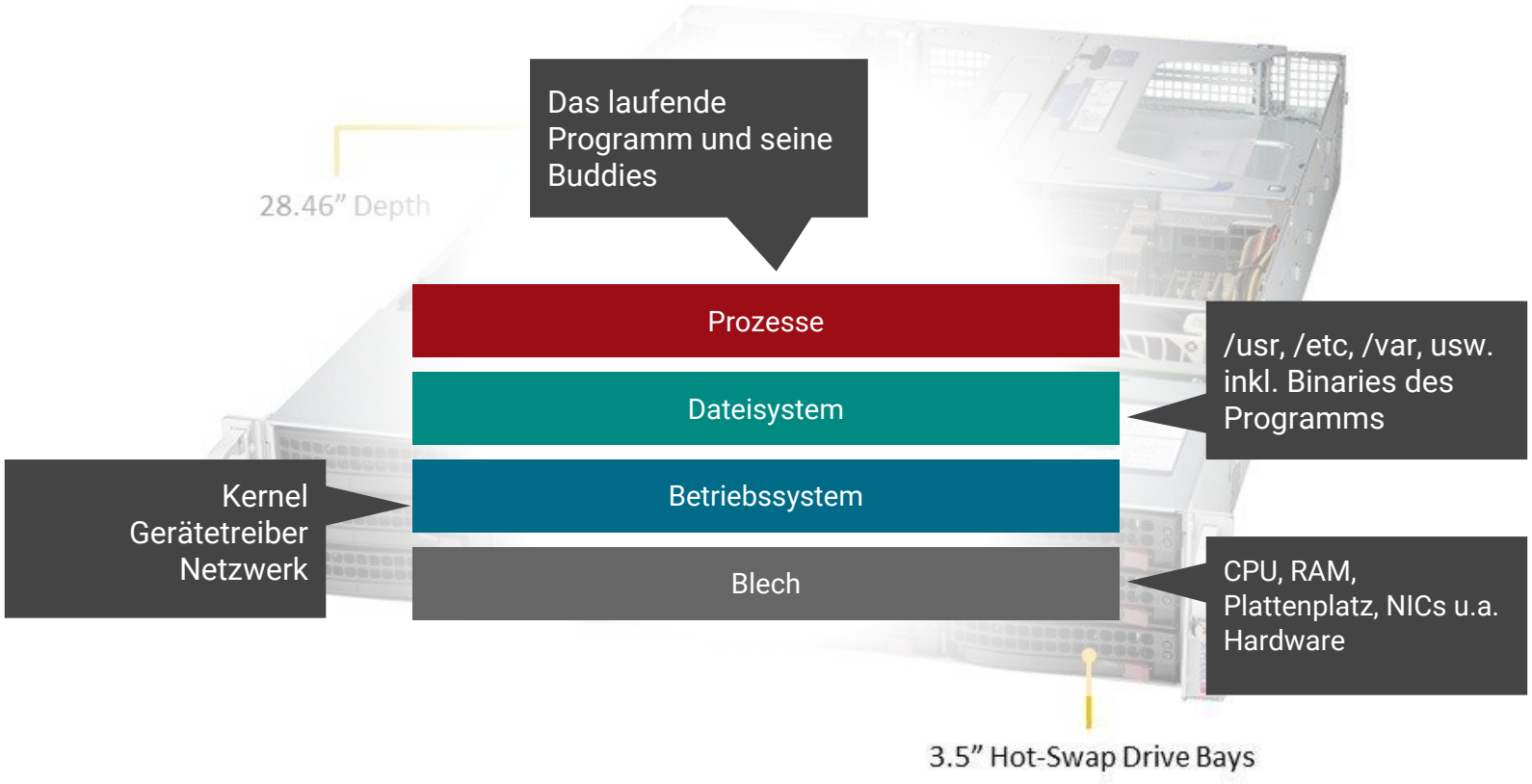
DR. PHILIPP WALTER

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs

Was braucht man, um ein Programm auszuführen?

INFOSERVE >eurodata-Gruppe

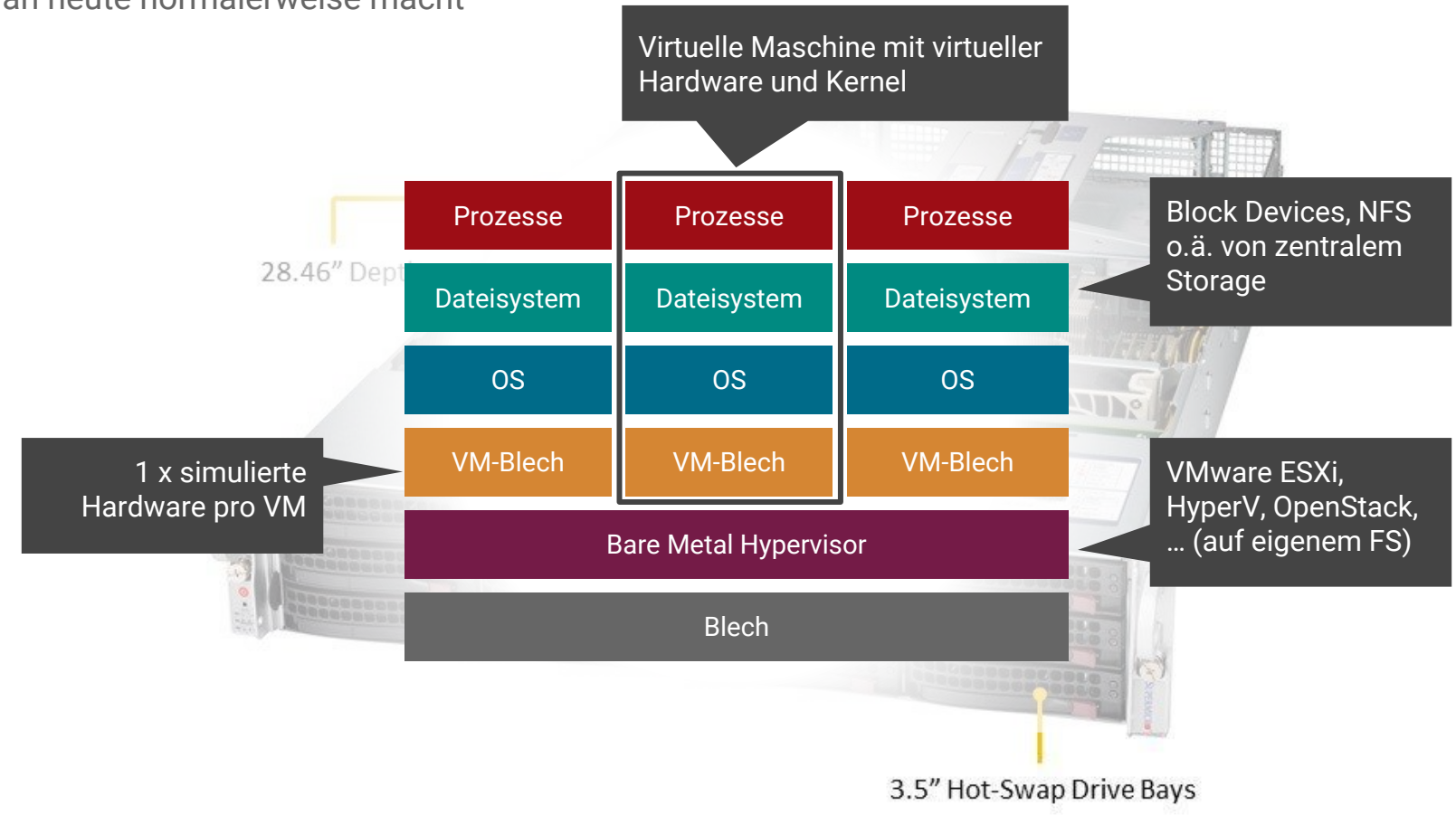
Die reine Lehre



Was braucht man, um ein Programm auszuführen?

INFOSERVE >eurodata-Gruppe

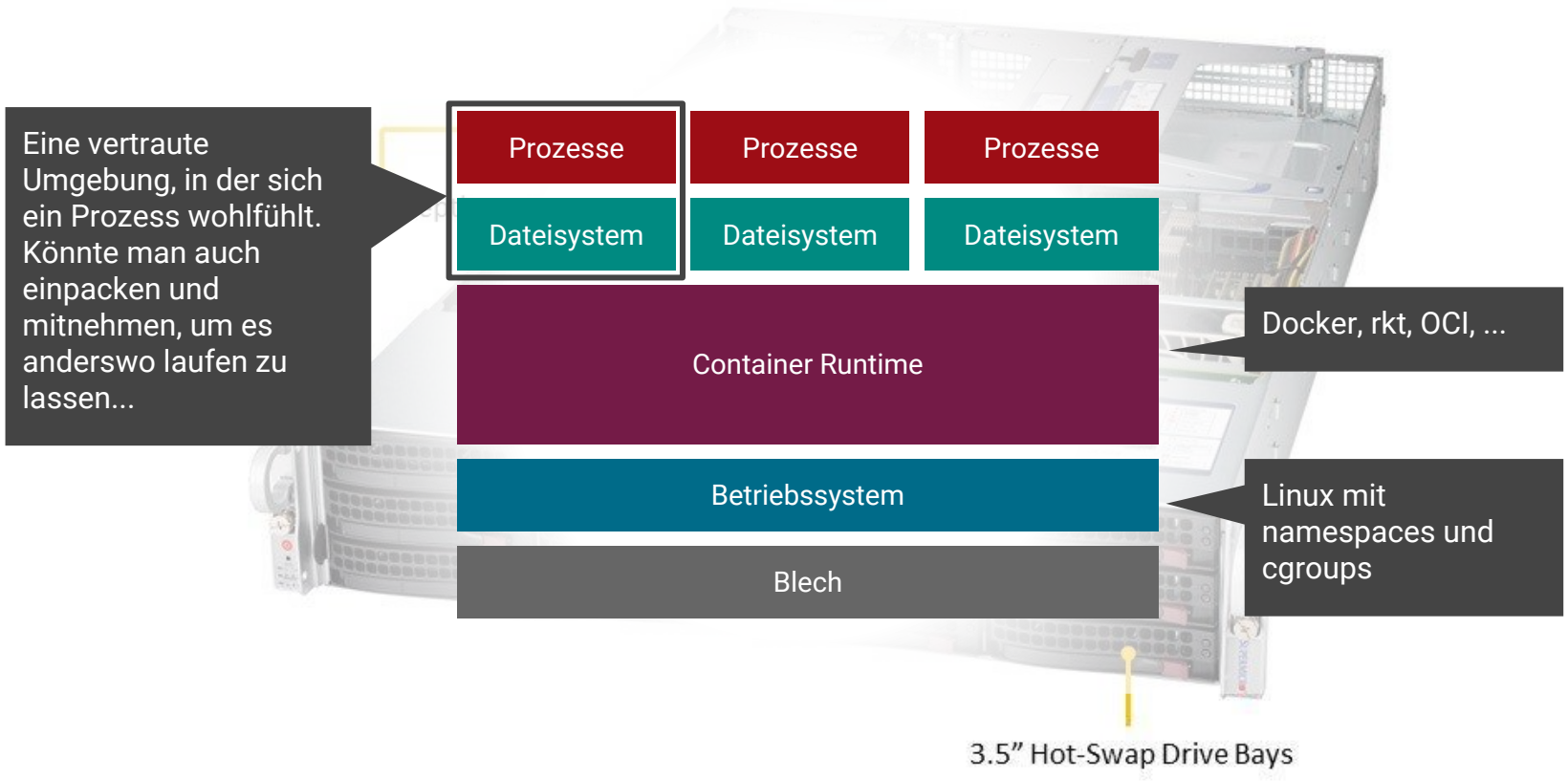
Was man heute normalerweise macht



Was braucht man, um ein Programm auszuführen?

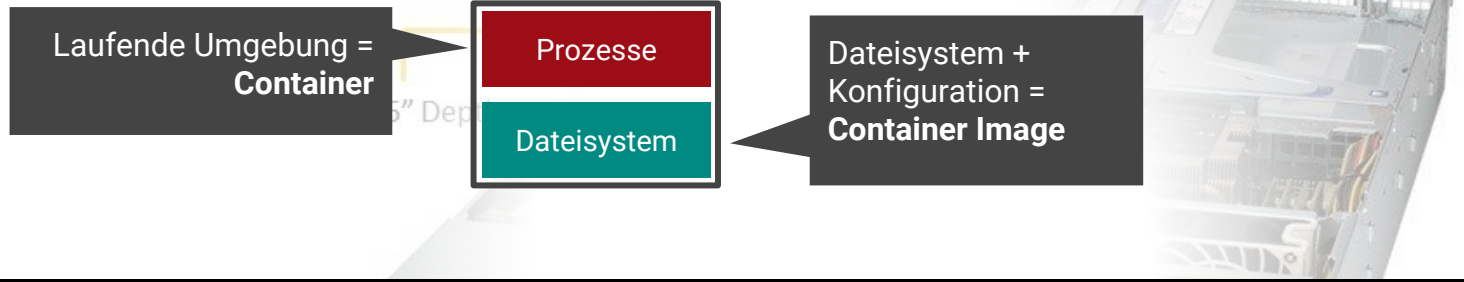
INFOSERVE >eurodata-Gruppe

Was man eigentlich bräuchte



Was braucht man, um ein Programm auszuführen?

Container transportieren Software samt ihrer Ausführungsumgebung




```
epic ~ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Status: Downloaded newer image for alpine:latest

epic ~ docker run -it --rm alpine
/ #
```

Container 101

Images suchen, herunterladen und starten

INFOSERVE
>eurodata-Gruppe

```
epic ~ docker search alpine
NAME                 DESCRIPTION                                     STARS    OFFICIAL  AUTOMATED
alpine               A minimal Docker image based on Alpine Linux... 4092     [OK]
mhart/alpine-node    Minimal Node.js built on Alpine Linux           373
anapsix/alpine-java  Oracle Java 8 (and 7) with GLIBC 2.23 over A…   341                [OK]
...

epic ~ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8593a0918d06d3ce30c6c93be117e430
Status: Downloaded newer image for alpine:latest

epic ~ docker run -it --rm alpine
/ #
```

Container 101

Environment-Variablen von außen setzen

```
epic ~ docker run -it --rm -e MEINE_VARIABLE=MEIN_WERT alpine

/ # env
HOSTNAME=1529dea4b71d
SHLVL=1
HOME=/root
MEINE_VARIABLE=MEIN_WERT
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/

/ #
```

Container 101

Host-Verzeichnisse und -Dateien als Container-Volumes mounten

```
epic ~ mkdir /tmp/demo
epic ~ echo huhu > /tmp/demo/hallo.txt

epic ~ docker run -it --rm -v /tmp/demo:/mnt alpine

/ # ls /mnt
hallo.txt
/ # cat /mnt/hallo.txt
huhu
```

Externe Shell

```
epic ~ echo Hallo Welt! > /tmp/demo/hallo.txt
```

```
/ # cat /mnt/hallo.txt
Hallo Welt!
/ #
```

Container 101

Netzwerkdienste im Container von außen zugänglich machen

INFOSERVE >eurodata-Gruppe

```
epic ~ docker run -it --rm -p 8080:80 alpine

/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
31: eth0@if32: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever

/ # nc -l -p 80
```

Externe Shell

```
epic ~ echo huhu | nc -c localhost 8080
```

```
huhu
^C
/ # wget https://www.infoserve.de
Connecting to www.infoserve.de (212.89.154.36:443)
index.html           100% |*******************************************************| 65886   0:00:00 ETA
/ #
```

Container 101

Registries, oder: wo kommen eigentlich die kleinen Container-Images her?

INFOSERVE >eurodata-Gruppe

```
epic ~ docker run -d -p 5000:5000 --name registry registry:2
f8ebdc69117d2f927ab57ad43db063c9af5baac76a3340497a179e413a5199c3

epic ~ docker ps
CONTAINER ID  IMAGE       COMMAND                  CREATED        STATUS        PORTS                    NAMES
f8ebdc69117d  registry:2  "/entrypoint.sh /etc..." 3 seconds ago  Up 2 seconds  0.0.0.0:5000->5000/tcp  registry

epic ~ docker pull alpine
Status: Image is up to date for alpine:latest

epic ~ docker image tag alpine localhost:5000/mein_alpine

epic ~ docker images
REPOSITORY                    TAG     IMAGE ID       CREATED        SIZE
alpine                        latest  11cd0b38bc3c   6 weeks ago    4.41MB
localhost:5000/mein_alpine    latest  11cd0b38bc3c   6 weeks ago    4.41MB

epic ~ docker push localhost:5000/mein_alpine
The push refers to repository [localhost:5000/mein_alpine]
73046094a9b8: Pushed
latest: digest: sha256:0873c923e00e0fd2ba78041bfb64a105e1ecb7678916d1f7776311e45bf5634b size: 528

epic ~ docker run -it --rm localhost:5000/mein_alpine
/ #
```

Kubernetes verwaltet Container auf Clustern

Docker : Kubernetes = 1 : N

	DOCKER	KUBERNETES
LAUFZEITUMGEBUNG	**Einzelner Host** mit Docker-Binaries	**Viele Hosts** mit Docker- und Kubernetes-Binaries
USE CASES	Software lokal laufen lassen Container-Images bauen und testen	Skalierbare und selbstheilende Produktionsumgebungen
API	docker pull, run, start, stop, ps, images, ...	kubectl + yaml-Specs (Details folgen…)
OBJEKTE	Container, Images, Volumes, Ports	Pods aus 1..n Containern, Services, Clusterknoten, Ingress, Deployments, ...
PARADIGMA	**Low-Level-Operationen**, die **manuell** auf einzelnen Containern ausgeführt werden	Definiere einen **Sollzustand**, den Kubernetes **automatisch** um jeden Preis zu halten versucht
SETUP	apt-get install docker	kubeadm, GKE, CoreOS, … (Details folgen)

Warum “Kubernetes”?

Ein bisschen Nerd muss sein: Google und die Borg

7 of 9, Borg

7 Seiten...

Interner Cluster Manager “Borg” bei Google (2015)

kubernetes

Large-scale cluster management at Google with Borg

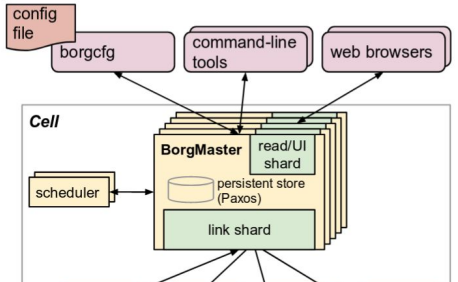
Abhishek Verma† Luis Pedrosa‡ Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes

Google Inc.

Abstract

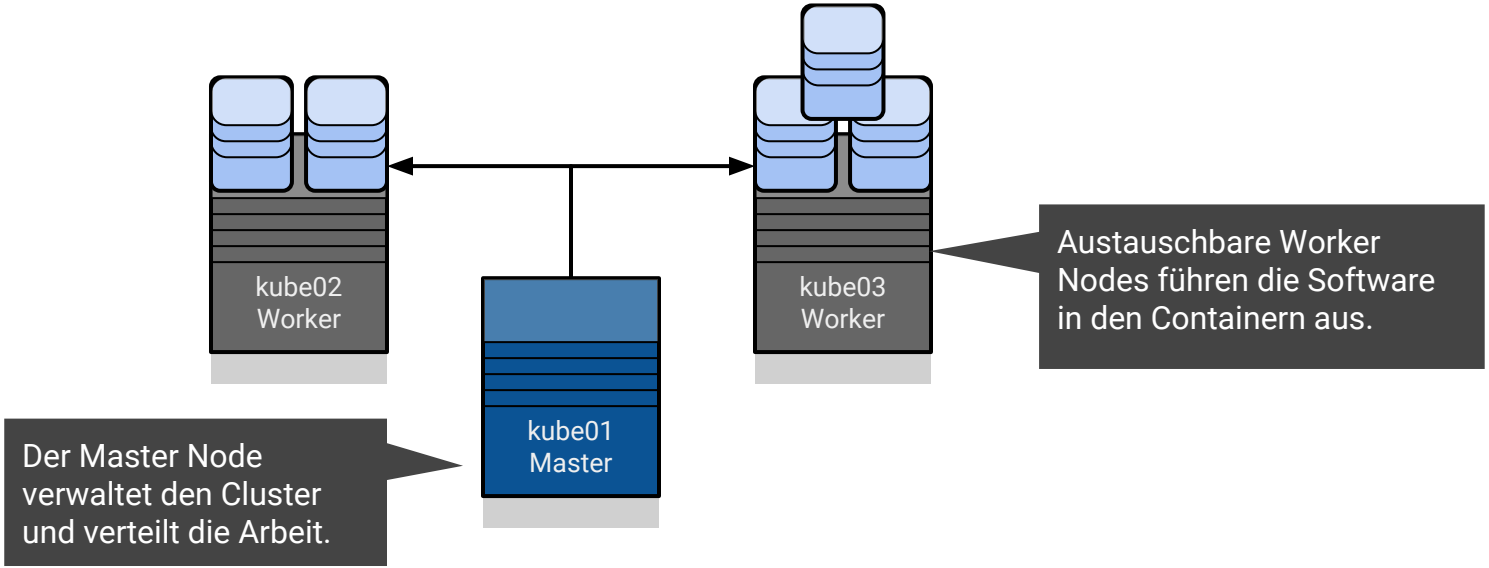
Google’s Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that min-



Kubernetes 101

Laufzeitumgebung: Cluster statt Computer




```
epic ~ kubectl get nodes -o wide
NAME    STATUS ROLES  AGE VERSION  EXTERNAL-IP  OS-IMAGE             KERNEL-VERSION             CONTAINER-RUNTIME
kube01 Ready  master 3d  v1.11.2  <none>       CentOS Linux 7 (Core) 3.10.0-862.11.6.el7.x86_64 docker://1.13.1
kube02 Ready  <none> 3d  v1.11.2  <none>       CentOS Linux 7 (Core) 3.10.0-862.11.6.el7.x86_64 docker://1.13.1
kube03 Ready  <none> 3d  v1.11.2  <none>       CentOS Linux 7 (Core) 3.10.0-862.11.6.el7.x86_64 docker://1.13.1
```

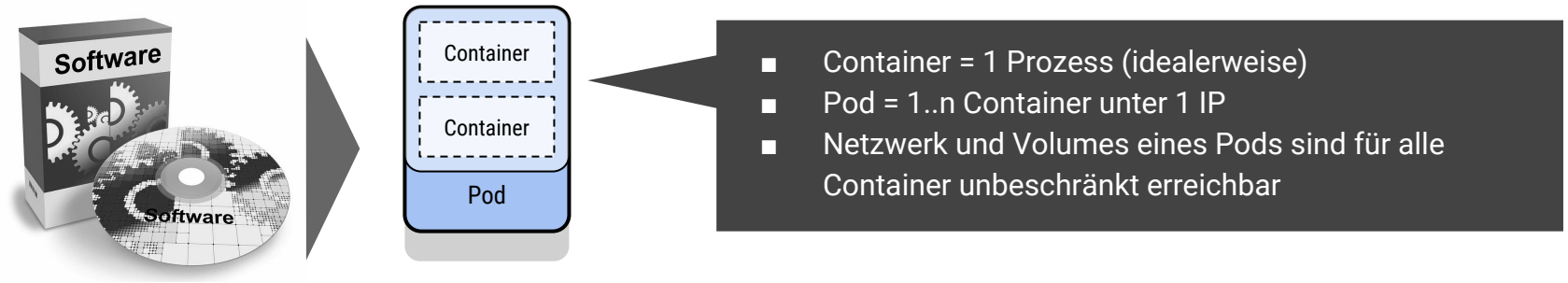
Kubernetes 101

Kubernetes versucht, den Sollzustand zu erhalten, indem Prozesse am Laufen gehalten werden

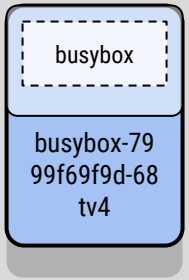
```
epic ~ kubectl run -i -t busybox --image=busybox:1.28
If you don't see a command prompt, try pressing enter.
/ # exit
Session ended, resume using 'kubectl attach busybox-7999f69f9d-68tv4 -c busybox -i -t' command when
the pod is running

epic ~ kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
busybox-7999f69f9d-68tv4   1/1       Running   1          6m

epic ~ kubectl attach busybox-7999f69f9d-68tv4 -c busybox -i -t
If you don't see a command prompt, try pressing enter.
/ # exit
Session ended, resume using 'kubectl attach busybox-7999f69f9d-68tv4 -c busybox -i -t' command when
the pod is running

epic ~ kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
busybox-7999f69f9d-68tv4   1/1       Running   2          7m
```

Kubernetes 101

Pods bestehen aus Containern und sind der elementare Baustein von Kubernetes



- Container = 1 Prozess (idealerweise)
- Pod = 1..n Container unter 1 IP
- Netzwerk und Volumes eines Pods sind für alle Container unbeschränkt erreichbar




```
epic ~ kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
busybox-7999f69f9d-68tv4   1/1       Running   2          7m

epic ~ kubectl attach busybox-7999f69f9d-68tv4 -c busybox -i -t
If you don't see a command prompt, try pressing enter.
/ #
```

Kubernetes 101

Replica Sets halten Pods am Laufen und werden als Deployments verwaltet

INFOSERVE >eurodata-Gruppe

```
epic ~ kubectl describe pod busybox-7999f69f9d-68tv4
...
Status:             Running
IP:                 10.116.0.3
Controlled By:      ReplicaSet/busybox-7999f69f9d
...

epic ~ kubectl describe ReplicaSet/busybox-7999f69f9d
...
Controlled By:  Deployment/busybox
Replicas:       1 current / 1 desired
Pods Status:    1 Running / 0 Waiting / 0 Succeeded / 0 Failed
...

epic ~ kubectl describe Deployment/busybox
...
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
RollingUpdateStrategy:  1 max unavailable, 1 max surge
...
```

Kubernetes 101

Deployments bündeln Pods und Replica Sets

INFOSERVE >eurodata-Gruppe

```
epic ~ kubectl get all
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/busybox   1         1         1            1           1h

NAME                   DESIRED   CURRENT   READY     AGE
rs/busybox-7999f69f9d   1         1         1         1h

NAME                         READY     STATUS    RESTARTS   AGE
po/busybox-7999f69f9d-68tv4   1/1       Running   3          1h

NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   3d

epic ~ kubectl delete deployment busybox
deployment "busybox" deleted

epic ~ kubectl get all
NAME                         READY     STATUS        RESTARTS   AGE
po/busybox-7999f69f9d-68tv4   1/1       Terminating   3          1h

NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   3d
```

Deployments

Deployments können als YAML oder JSON beschrieben werden. Hurra, Infrastructure as Code!

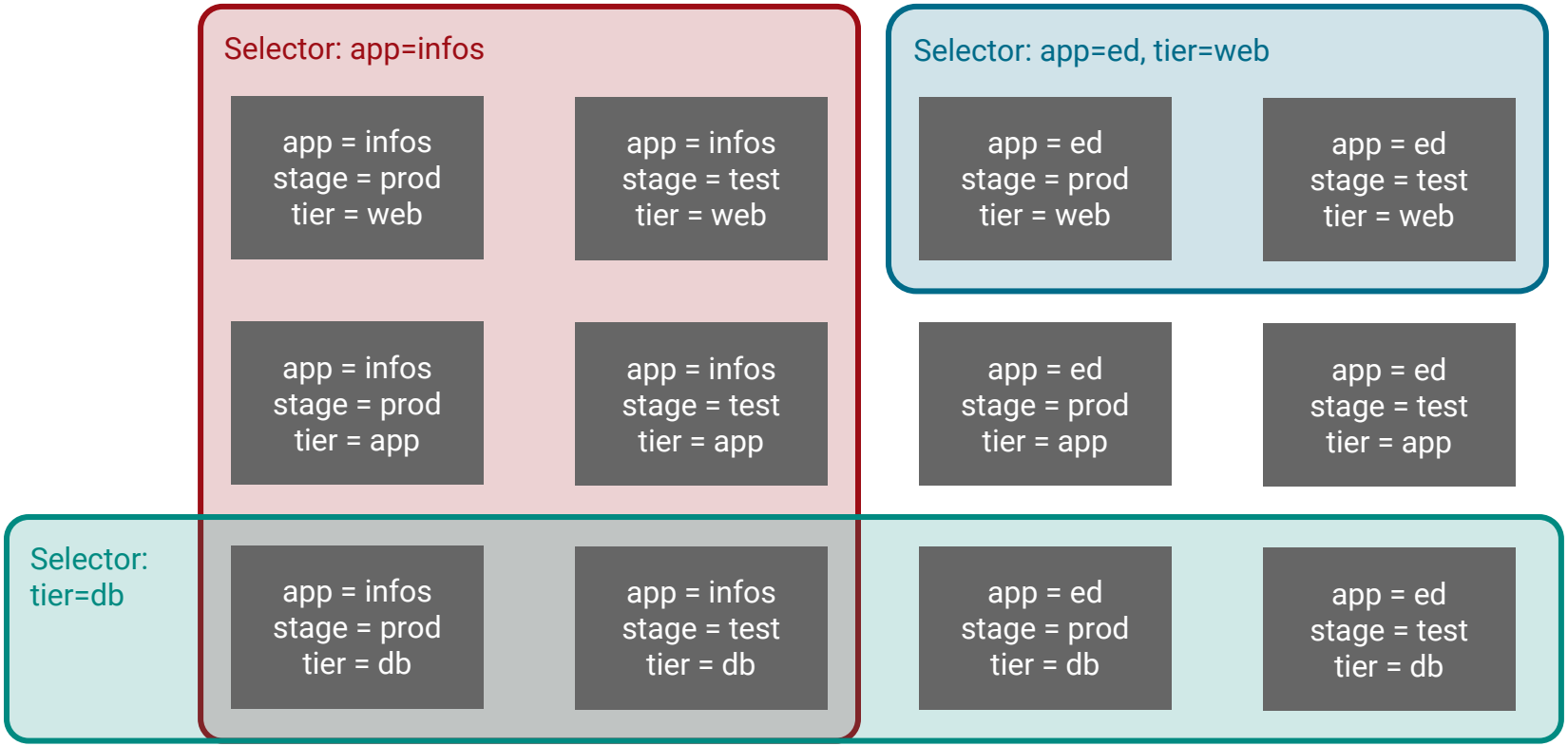
echo.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo
spec:
  replicas: 2
  selector:
    matchLabels:
      pod: echo
  template:
    metadata:
      labels:
        pod: echo
    spec:
      containers:
      - name: echo
        image: jmalloc/echo-server:latest
        ports:
        - containerPort: 8080
```

```
epic ~ kubectl apply -f echo.yaml
deployment "echo" created

epic ~ kubectl get all
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/echo      2         2         2            2           6m

NAME                    DESIRED   CURRENT   READY     AGE
rs/echo-8594ff85cb      2         2         2         6m

NAME                          READY     STATUS    RESTARTS   AGE
po/echo-8594ff85cb-vg72l      1/1       Running   0          6m
po/echo-8594ff85cb-xbl9t      1/1       Running   0          6m

NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP    5d

epic ~ kubectl scale deployment echo --replicas 3
deployment "echo" scaled

epic ~ kubectl get all
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/echo      3         3         3            3           7m
...
```

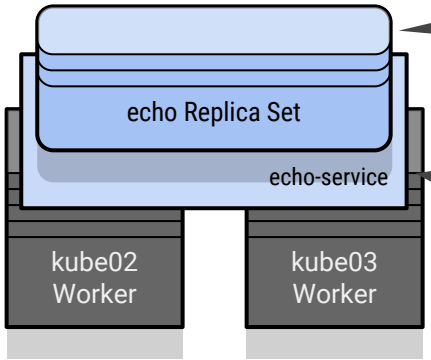
Exkurs: Labels und Selektoren

Man kann alles beschriften und damit auswählen



Services

Services ermöglichen Layer-4-Load-Balancing über Pods



echo-server läuft im Container auf Port 8080

Die Pods sind loadbalanced im ganzen Cluster über eine Cluster-IP auf Port 80 erreichbar. Jede neue TCP-Verbindung wird an ein zufälliges Backend durchgestellt.

echo-service.yaml

```
kind: Service
apiVersion: v1
metadata:
  name: echo-service
spec:
  selector:
    pod: echo
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: NodePort
```

```
epic ~ kubectl apply -f echo-service.yaml
service "echo-service" created

epic ~ kubectl get svc
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
echo-service   NodePort    10.104.0.183    <none>        80:31763/TCP   6m
kubernetes     ClusterIP   10.96.0.1       <none>        443/TCP        5d
```

Services

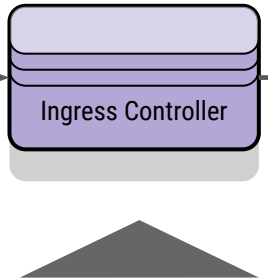
Die Cluster-IP eines Service kann überall im Cluster auch über seinen DNS-Namen aufgelöst werden

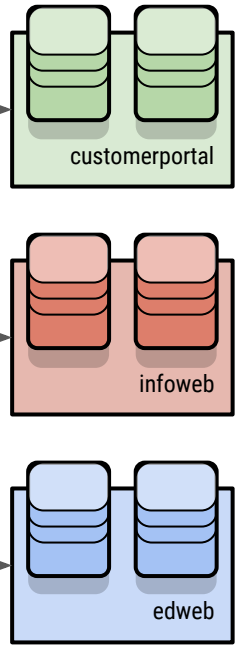
```
epic ~ kubectl run -i -t busybox --image=busybox:1.28
If you don't see a command prompt, try pressing enter.

/ # nslookup echo-service
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      echo-service
Address 1: 10.104.0.183 echo-service.default.svc.cluster.local

/ # watch -n 1 wget -qO - echo-service | grep served
Request served by echo-8594ff85cb-vg72l
Request served by echo-8594ff85cb-xbl9t
Request served by echo-8594ff85cb-xbl9t
Request served by echo-8594ff85cb-vg72l
Request served by echo-8594ff85cb-vg72l
...

/ #
```

Ingress Controller

Layer-7-Load-Balancing mit URL Mappings auf verschiedene Services

Der “Ingress Controller” ist der Container mit der Reverse-Proxy-Software, der die Ingress-Konfiguration umsetzt.





“Ingress”-Konfiguration ist die URL-Mapping-Tabelle. Kann von mehreren Deployments unabhängig gefüllt werden.

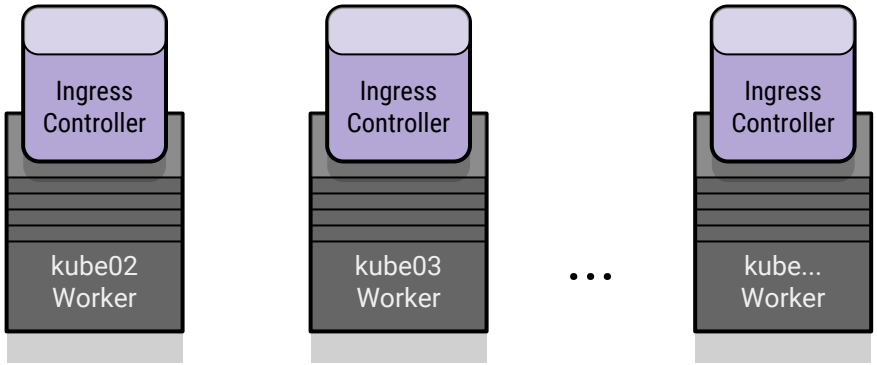
Host	URL	Backend
infos.de	/myinfoserve/*	customerportal:80
infos.de	/*	infoweb:80
eurodata.de	/*	edweb:80

Ingress Controller

Ingress Controller auf eigenem Cluster (kein GKE) deployen

traefik-ingress-controller.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
---
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
    ...
```



https://docs.traefik.io/user-guide/kubernetes/

```
epic ~ kubectl get all --namespace=kube-system -l k8s-app=traefik-ingress-lb
NAME                                DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds/traefik-ingress-controller-v1    2         2         2         2            2           <none>          8m

NAME                                    READY     STATUS    RESTARTS   AGE
po/traefik-ingress-controller-v1-7664s  1/1       Running   0          8m
po/traefik-ingress-controller-v1-pk7bq  1/1       Running   0          8m

NAME                      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)                   AGE   SELECTOR
traefik-ingress-service   ClusterIP   10.99.243.201  <none>        80/TCP,443/TCP,8080/TCP   8m    k8s-app=traefik-ingress-lb
```

Ingress-Ressourcen

Ingress-Ressourcen machen Services unter einer URL und einem Pfad zugänglich

echo-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: echo-ingress
spec:
  rules:
  - http:
      paths:
      - path: /echo
        backend:
          serviceName: echo-service
          servicePort: 80
```

```
epic ~ kubectl create -f echo-ingress.yaml
ingress "echo-ingress" created

epic ~ curl kube02/echo
Request served by echo-8594ff85cb-xmbb7

HTTP/1.1 GET /echo

Host: kube02
X-Forwarded-Port: 80
X-Forwarded-Server: traefik-ingress-controller-qvqbj
X-Real-Ip: 10.124.0.0
User-Agent: curl/7.61.0
X-Forwarded-For: 10.124.0.0
X-Forwarded-Host: kube02
Accept-Encoding: gzip
Accept: */*
X-Forwarded-Proto: http

epic ~ curl kube03/echo
Request served by echo-8594ff85cb-mj65h

HTTP/1.1 GET /echo
...
```

INFOSERVE
>eurodata-Gruppe

DEMO 1

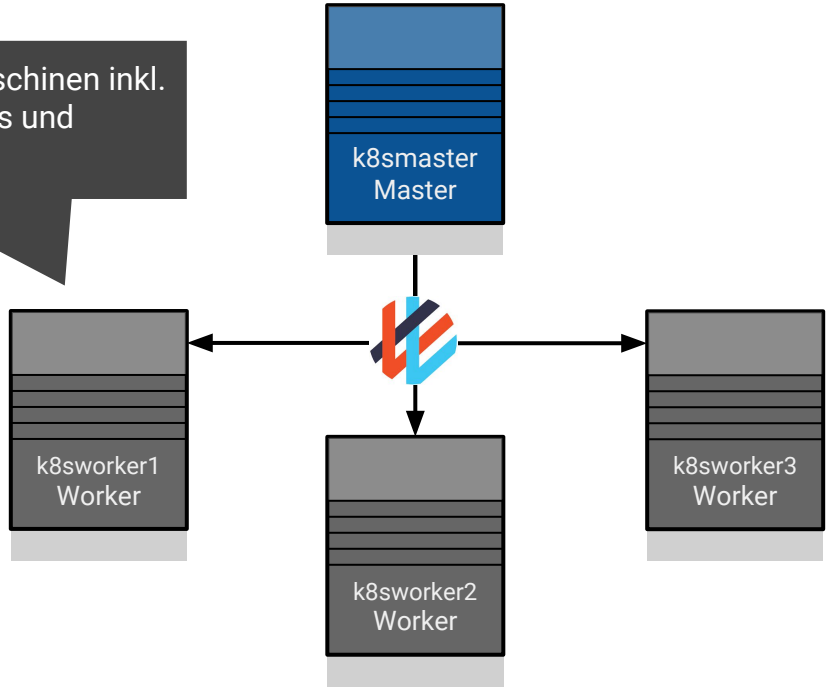
Lokaler Kubernetes-Cluster mit Vagrant

THOMAS DARIMONT

https://github.com/thomasdarimont/vagrant-kubernetes-lab/tree/poc/kubernetes-next

Lokaler Kubernetes-Cluster mit Vagrant

Vagrant richtet lokale virtuelle Maschinen inkl. Netzwerkkonfiguration, Kubernetes und WeaveNet als CNI-Plugin ein.



INFOSERVE
>eurodata-Gruppe

DEMO 2

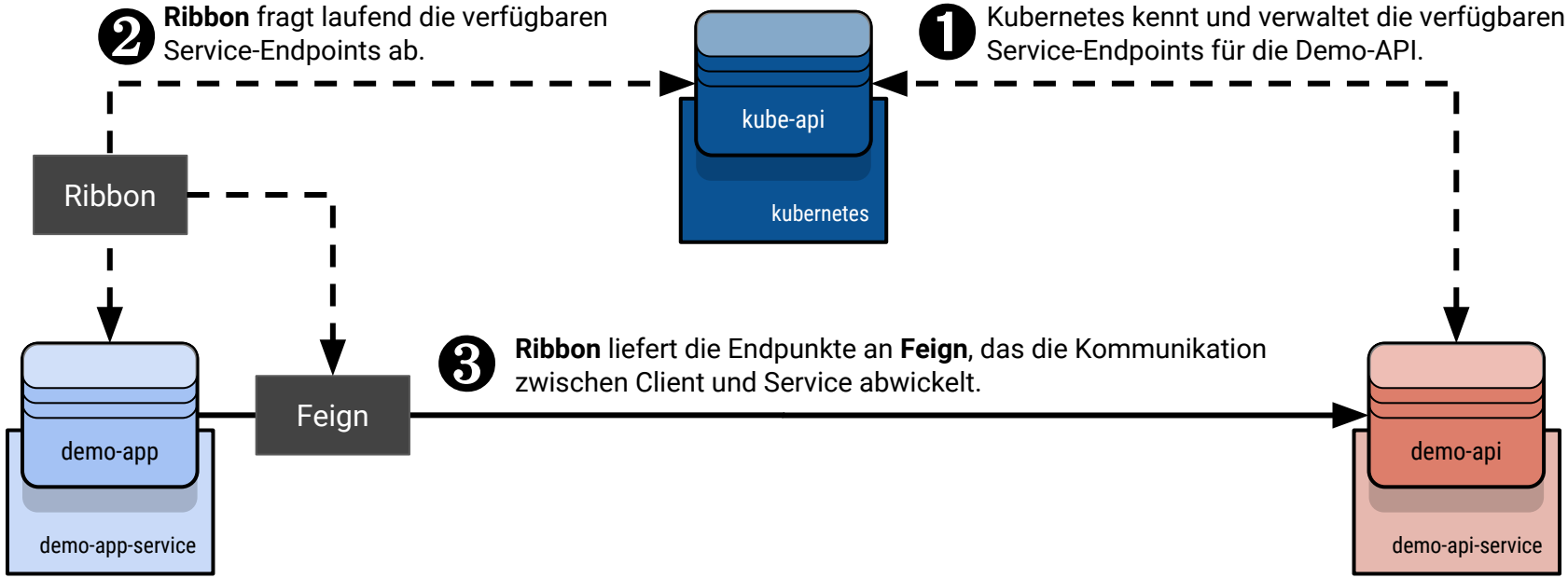
Cloud-Native Spring Boot App

THOMAS DARIMONT

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs/tree/master/code/spring-boot-k8s-demo

Cloud Native Spring Boot App

inkl. REST-Call mit Feign und Client-side Load Balancing mit Ribbon



INFOSERVE
>eurodata-Gruppe

DEMO 3

Rolling Upgrade

THOMAS DARIMONT

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs/tree/master/code/rolling-upgrade-demo

INFOSERVE
>eurodata-Gruppe

DEMO 4

HTTP Session Clustering mit Hazelcast

THOMAS DARIMONT

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs/tree/master/code/spring-boot-k8s-hazelcast

INFOSERVE
>eurodata-Gruppe

DEMO 5

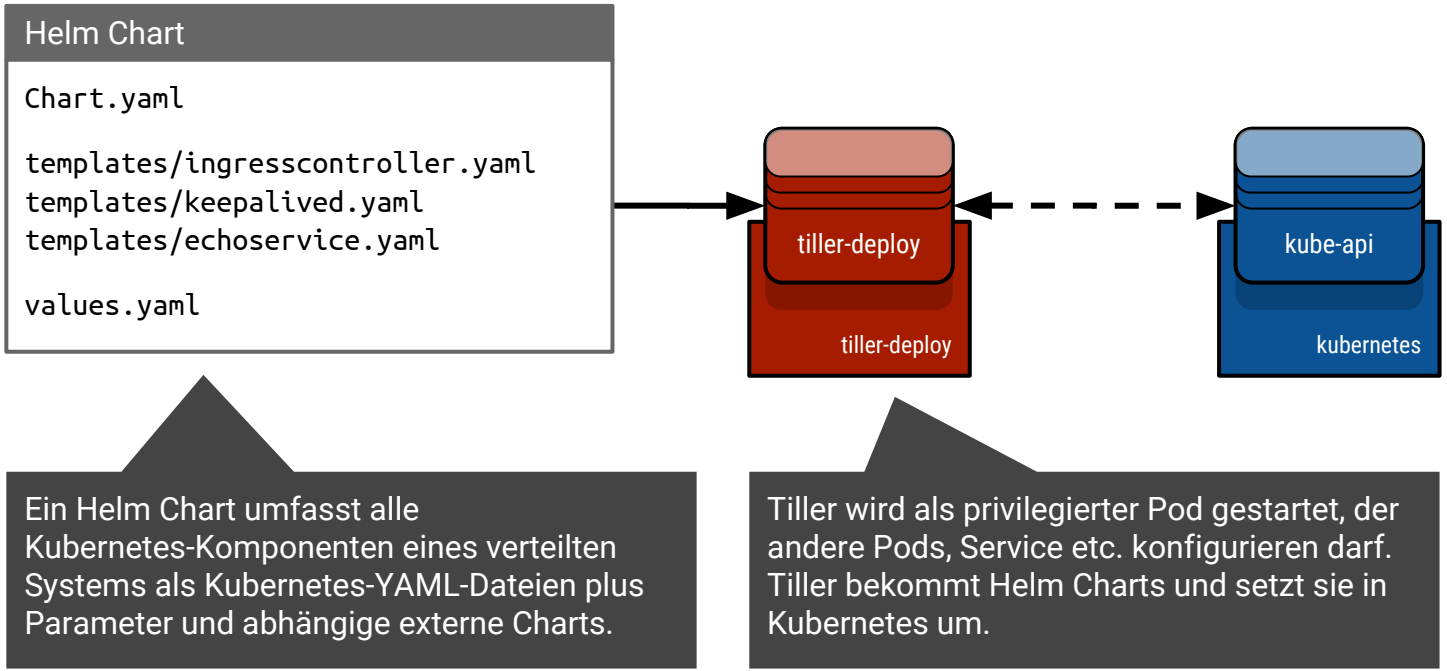
Keycloak Cluster als Helm Chart

THOMAS DARIMONT

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs/tree/master/code/keycloak-helm

Helm

Parametrisierbare Infrastructure as Code | “Maven für Kubernetes” | http://helm.sh



INFOSERVE
>eurodata-Gruppe

DEMO 6

Kubernetes-Cluster auf VMware-Basis

DR. PHILIPP WALTER

https://github.com/jugsaar/jugsaar-37-kubernetes-for-devs