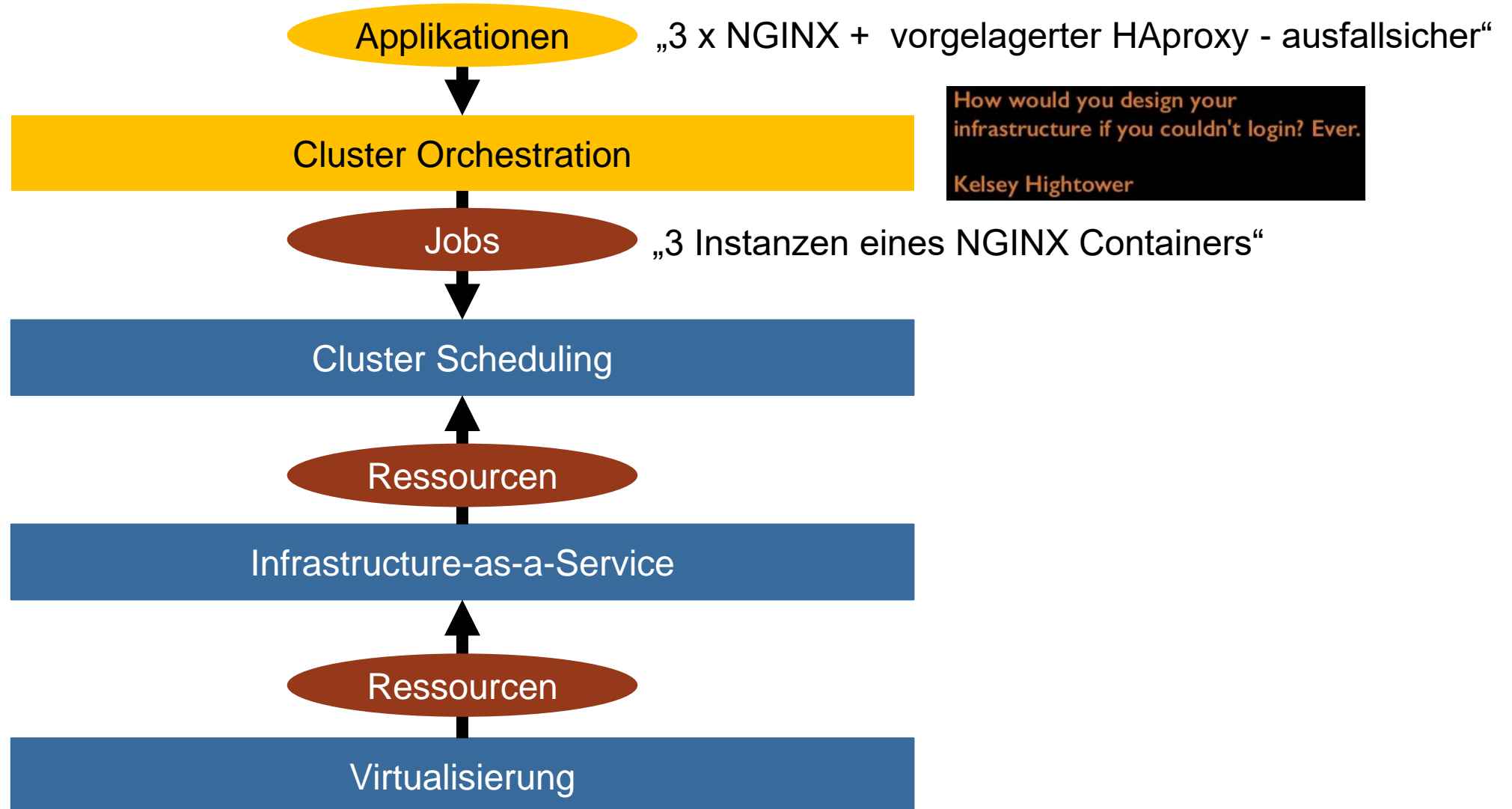


# Cloud Computing

## Kapitel 7: Orchestrierung

Dr. Josef Adersberger

# Das Big Picture: Wir sind nun auf Applikationsebene.



# Cluster-Orchestrierung

- Eine Anwendung, die in mehrere Betriebskomponenten (Container) aufgeteilt ist, auf mehreren Knoten laufen lassen.  
„Running Containers on Multiple Hosts“.  
DockerCon SF 2015: Orchestration for Sysadmins
- Führt Abstraktionen zur Ausführung von Anwendungen mit ihren Services in einem großen Cluster ein.
- Orchestrierung ist keine statische, einmalige Aktivität wie die Provisionierung sondern eine dynamische, kontinuierliche Aktivität.
- Orchestrierung hat den Anspruch, alle Standard-Betriebsprozeduren einer Anwendung zu automatisieren.

**Blaupause der Anwendung**, die den gewünschten Betriebszustand der Anwendung beschreibt: Betriebskomponenten (Container), deren Betriebsanforderungen sowie die angebotenen und benötigten Schnittstellen.



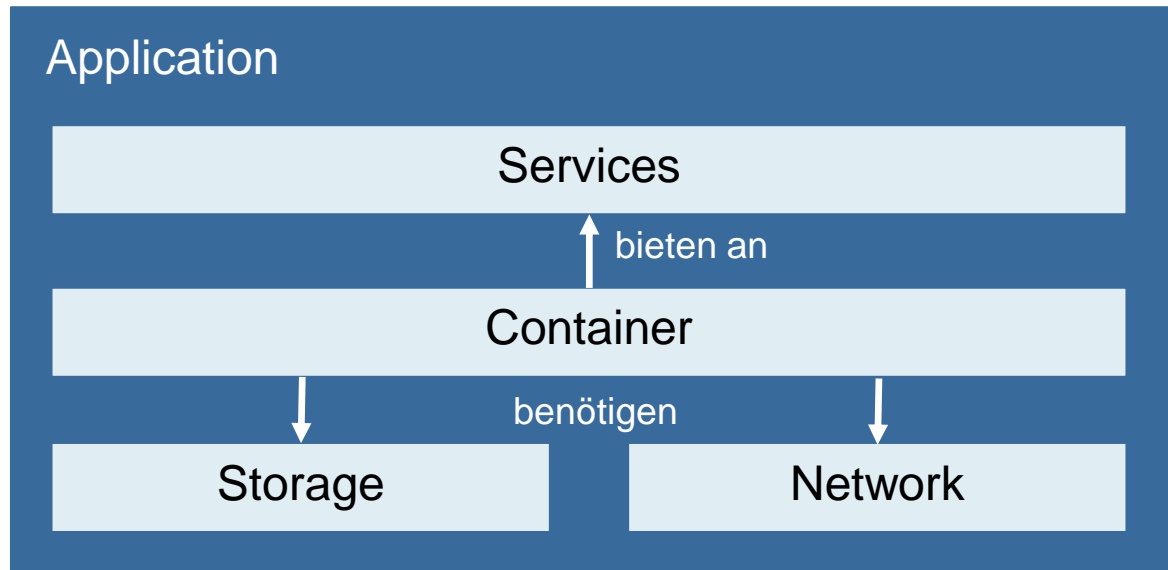
Cluster-Orchestrierer



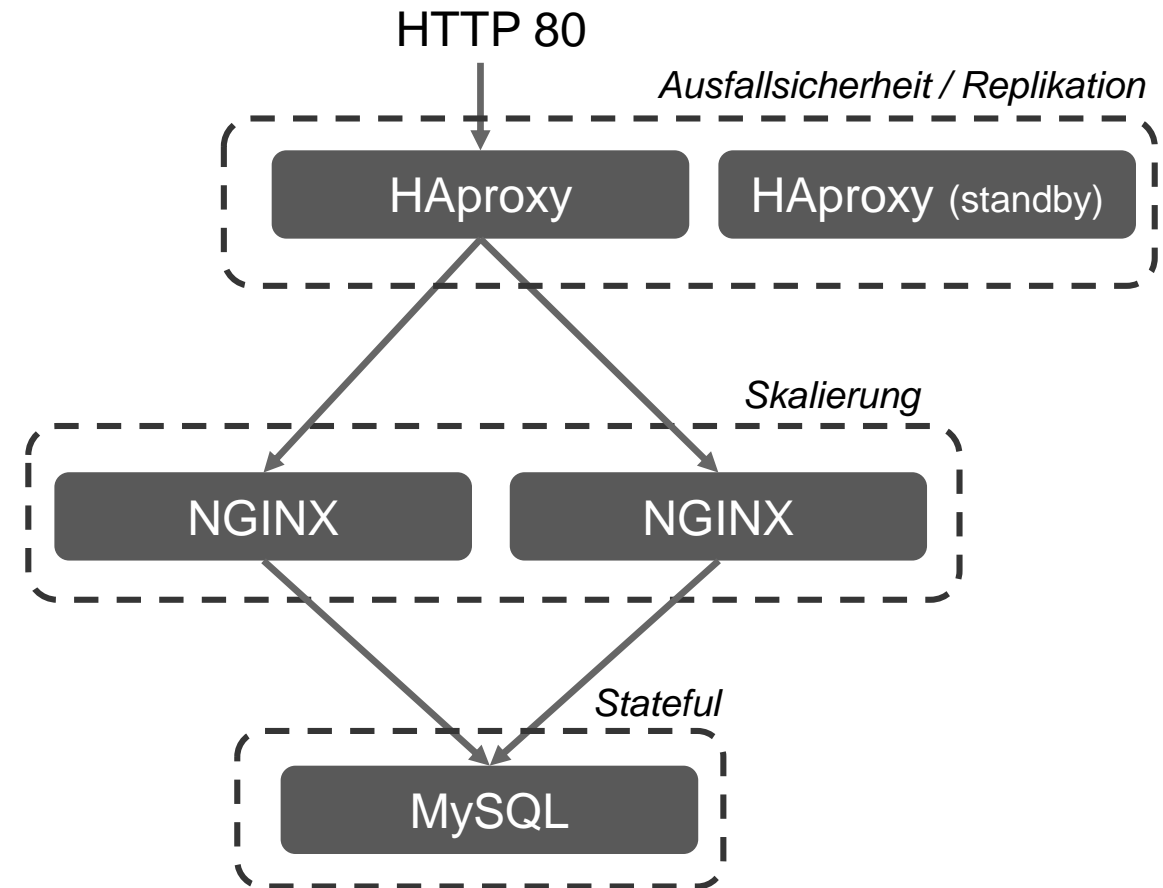
**Steuerungsaktivitäten im Cluster:**

- Start von Containern auf Knoten (→ Scheduler)
- Verknüpfung von Containern
- ...

# Blaupause einer Anwendung (vereinfacht)

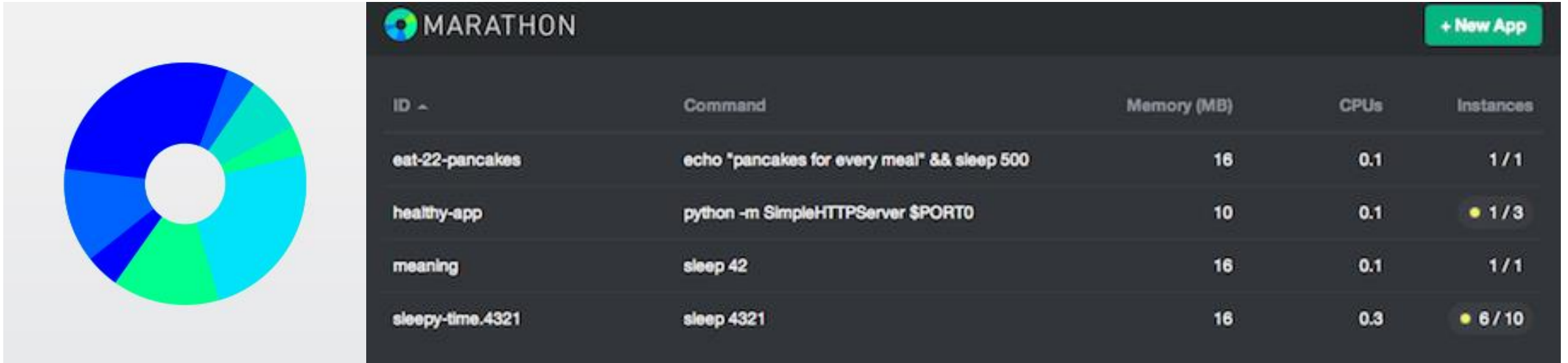


**Metamodell**

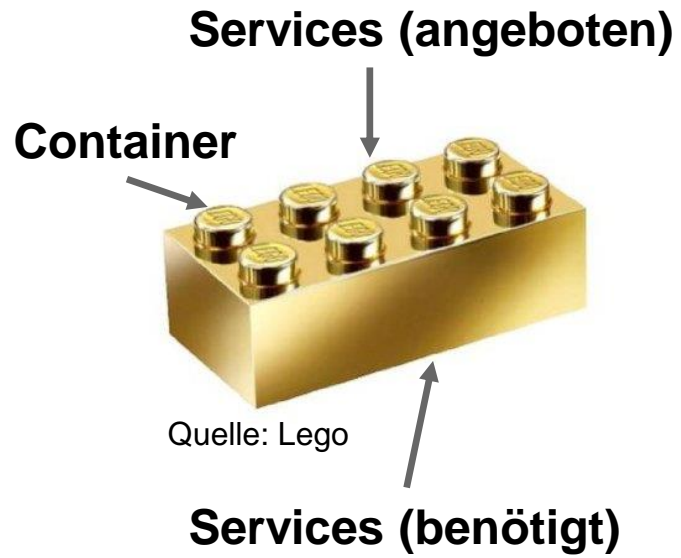


**Modell**

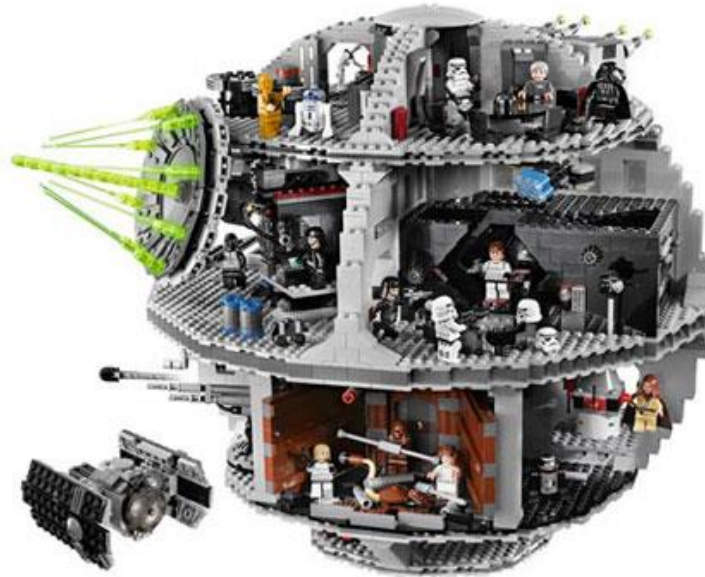
# Wir kennen bereits einen Cluster-Orchestrierer



# Analogie 1: Lego Star Wars



**Cluster-Orchestrierer**

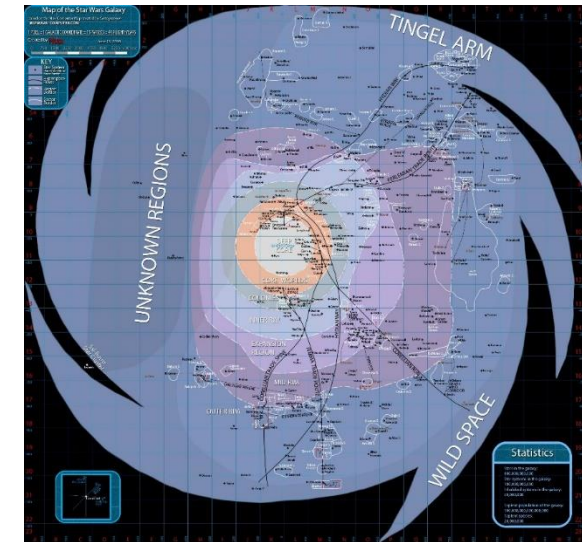


Quelle: Lego



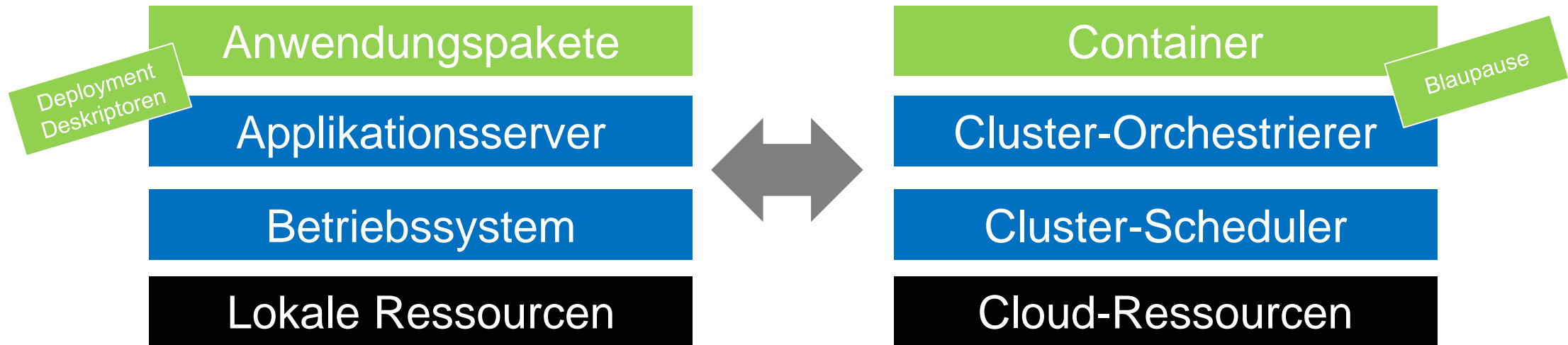
**Blaupause**

**Cluster-Scheduler**

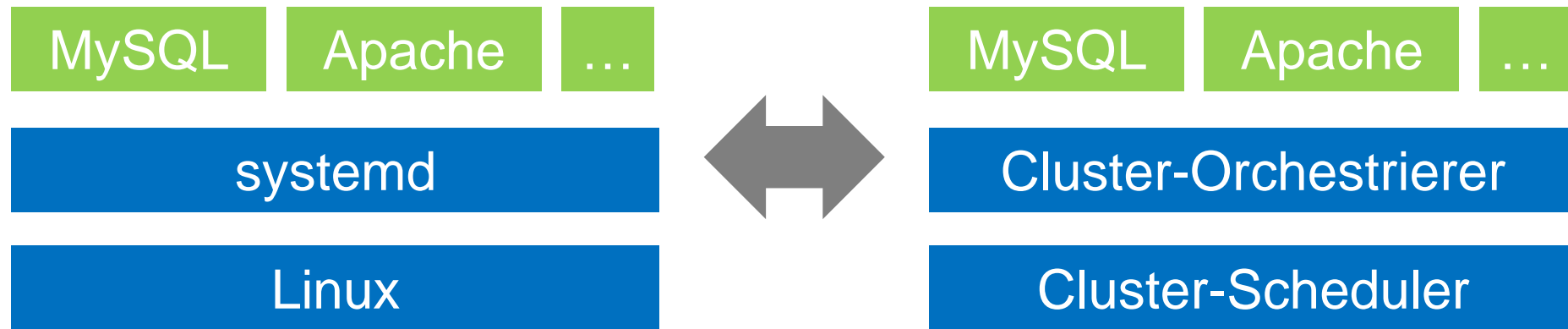


Quelle: wikipedia.de

## Analogie 2: Applikationsserver

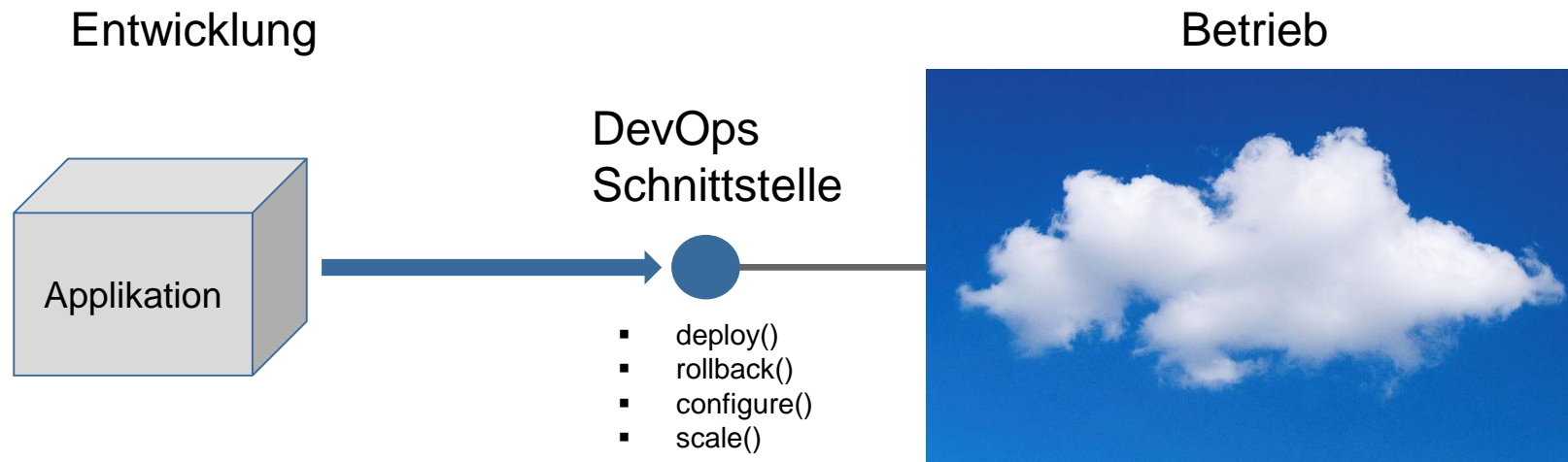


# Analogie 3: Betriebssystem





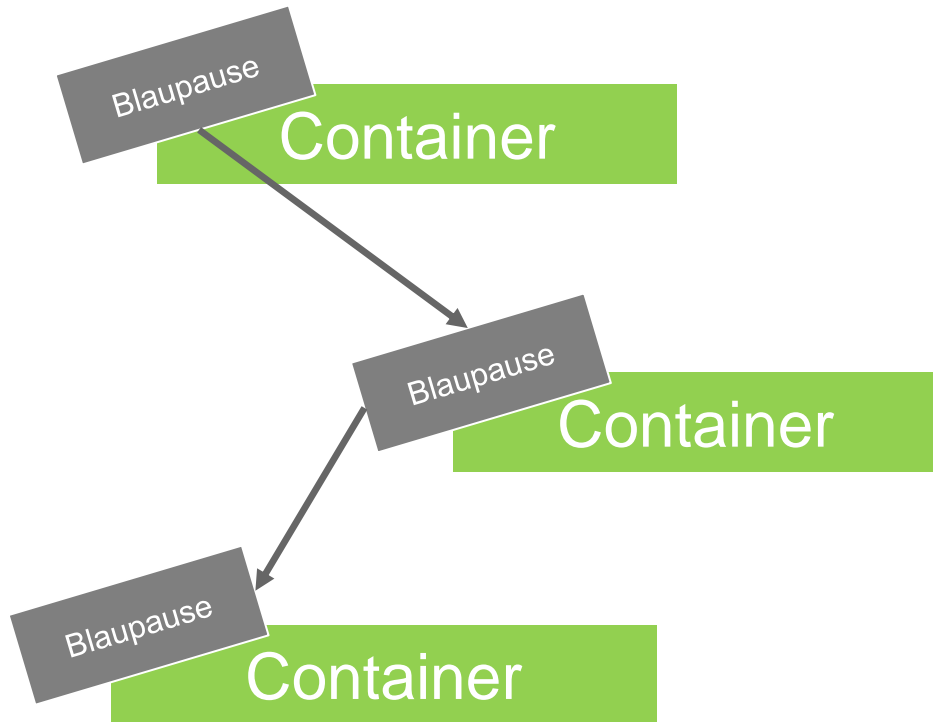
# Ein Cluster-Orchestrierer bietet eine Schnittstelle zwischen Betrieb und Entwicklung für ein Cluster an.



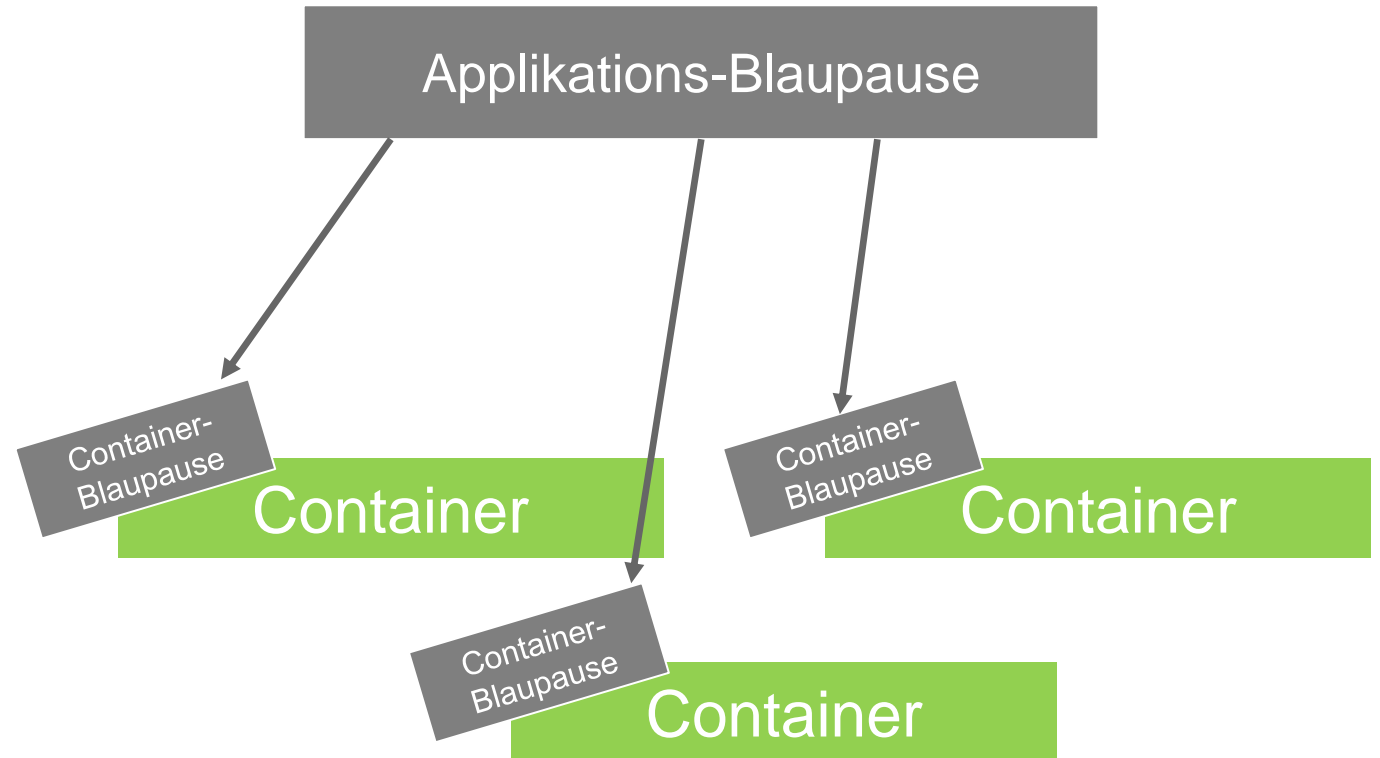
# Ein Cluster-Orchestrierer automatisiert vielerlei Betriebsaufgaben für Anwendung auf einem Cluster.

- Scheduling von Containern mit applikationsspezifischen Constraints (z.B. Deployment- und Start-Reihenfolgen, Gruppierung, ...)
- Aufbau von notwendigen Netzwerk-Verbindungen zwischen Containern.
- Bereitstellung von persistenten Speichern für zustandsbehaftete Container.
- (Auto-) Skalierung von Containern.
- Re-Scheduling von Containern im Fehlerfall (Auto-Healing) oder zur Performance-Optimierung.
- Container-Logistik: Verwaltung und Bereitstellung von Containern. Package-Management: Verwaltung und Bereitstellung von Applikationen.
- Bereitstellung von Administrationsschnittstellen (Remote-API, Kommandozeile).
- Management von Services: Service Discovery, Naming, Load Balancing.
- Automatismen für Rollout-Workflows wie z.B. Canary Rollout.
- Monitoring und Diagnose von Containern und Services.

# 1-Level- vs. 2-Level-Orchestrierung



**1-Level-Orchestrierung**  
(Container-Graph)



**2-Level-Orchestrierung**  
(Container-Repository mit zentraler Bauanleitung)

# 1-Level- vs. 2-Level-Orchestrierung

## Plain Docker

```
FROM ubuntu
ENTRYPOINT nginx
EXPOSE 80
```

```
docker run -d --link
nginx:nginx
```

**1-Level-Orchestrierung**  
(Container-Graph)

## <https://docs.docker.com/compose/compose-file> Docker Compose

weba:

```
image: qaware/nginx
expose:
  - 80
```

webb:

```
image: qaware/nginx
expose:
  - 80
```

haproxy:

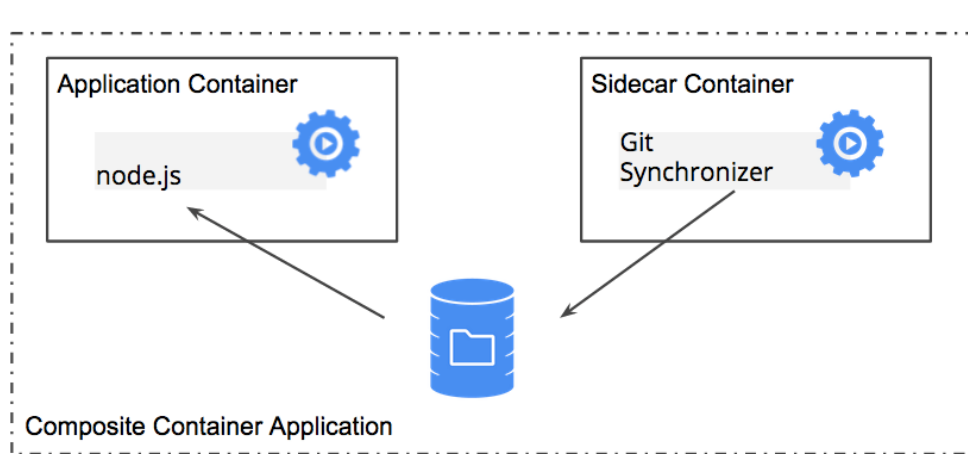
```
image: qaware/haproxy
links:
  - weba
  - webb
ports:
  - „80:80“
expose:
  - 80
```

FROM ubuntu  
ENTRYPOINT nginx  
EXPOSE 80

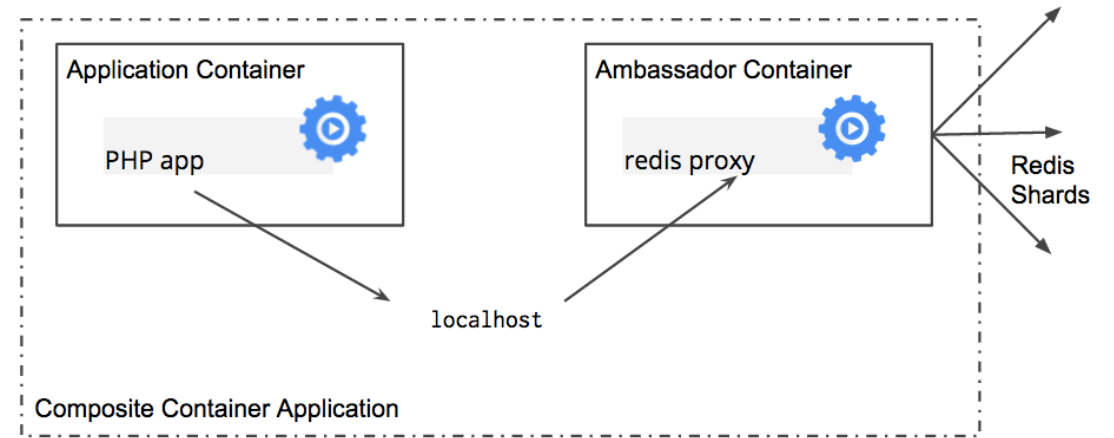
FROM ubuntu  
ENTRYPOINT haproxy  
EXPOSE 80

**2-Level-Orchestrierung**  
(Container-Repository mit zentraler Bauanleitung)

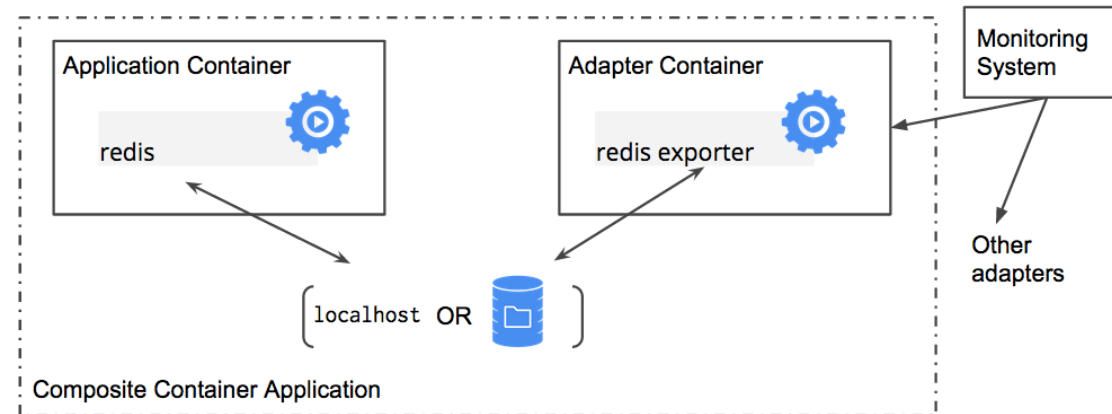
# Orchestrierungsmuster



Sidecar Container



Ambassador Container



Adapter Container

# Cluster Orchestrierer

- Kubernetes
- Apache Marathon & Chronos
- Docker Compose

# Kubernetes



kubernetes by Google

Manage a cluster of Linux containers as a single system to accelerate Dev and simplify Ops.

Josef Adersberger @adersberger · Jul 21

Google spares no effort to launch  
#kubernetes @ #OSCON

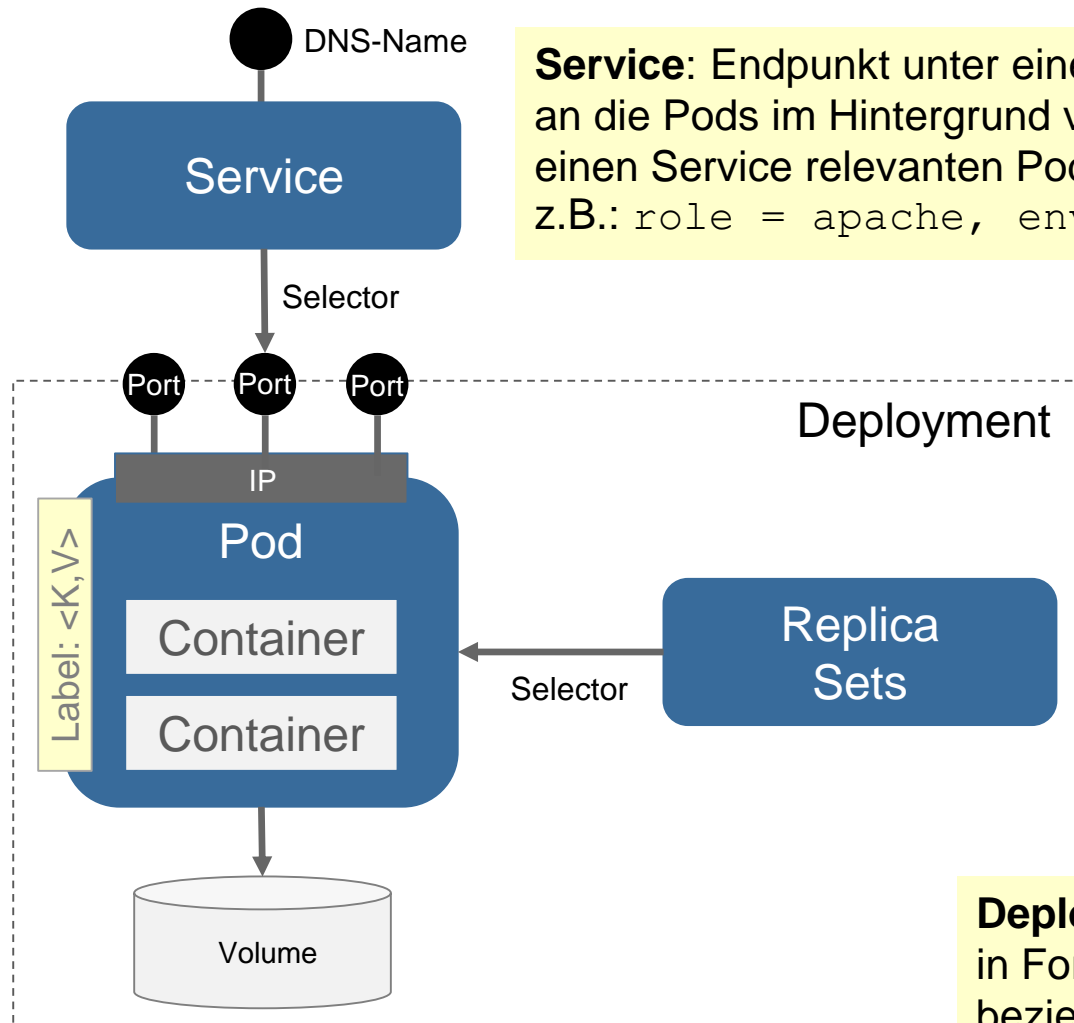


# Kubernetes

- Cluster-Orchestrierer auf Basis von Docker-Containern, der eine Reihe an Kern-Abstraktionen für den Betrieb von Anwendungen in einem großen Cluster einführt. Die Blaupause wird über YAML-Dateien definiert.
- Open-Source-Projekt, das von Google initiiert wurde. Google will damit die jahrelange Erfahrung im Betrieb großer Cluster der Öffentlichkeit zugänglich machen und damit auch Synergien mit dem eigenen Cloud-Geschäft heben.
- Seit Juli 2015 in der Version 1.0 verfügbar und damit produktionsreif. Skaliert aktuell nachweislich auf  $10^2$  großen Clustern.
- Aktuell bereits bei einigen Firmen im Einsatz wie z.B. Google im Rahmen der Google Container Engine, Wikipedia, ebay. Beiträge an der Codebasis aus vielen Firmen neben Google – u.A. Mesosphere, Microsoft, Pivotal, RedHat.
- Soll den Standard im Bereich Cluster-Orchestration setzen. Dafür wurde auch eigens die Cloud Native Computing Foundation gegründet (<https://cncf.io>).



# Der Kern-Abstraktionen von Kubernetes.



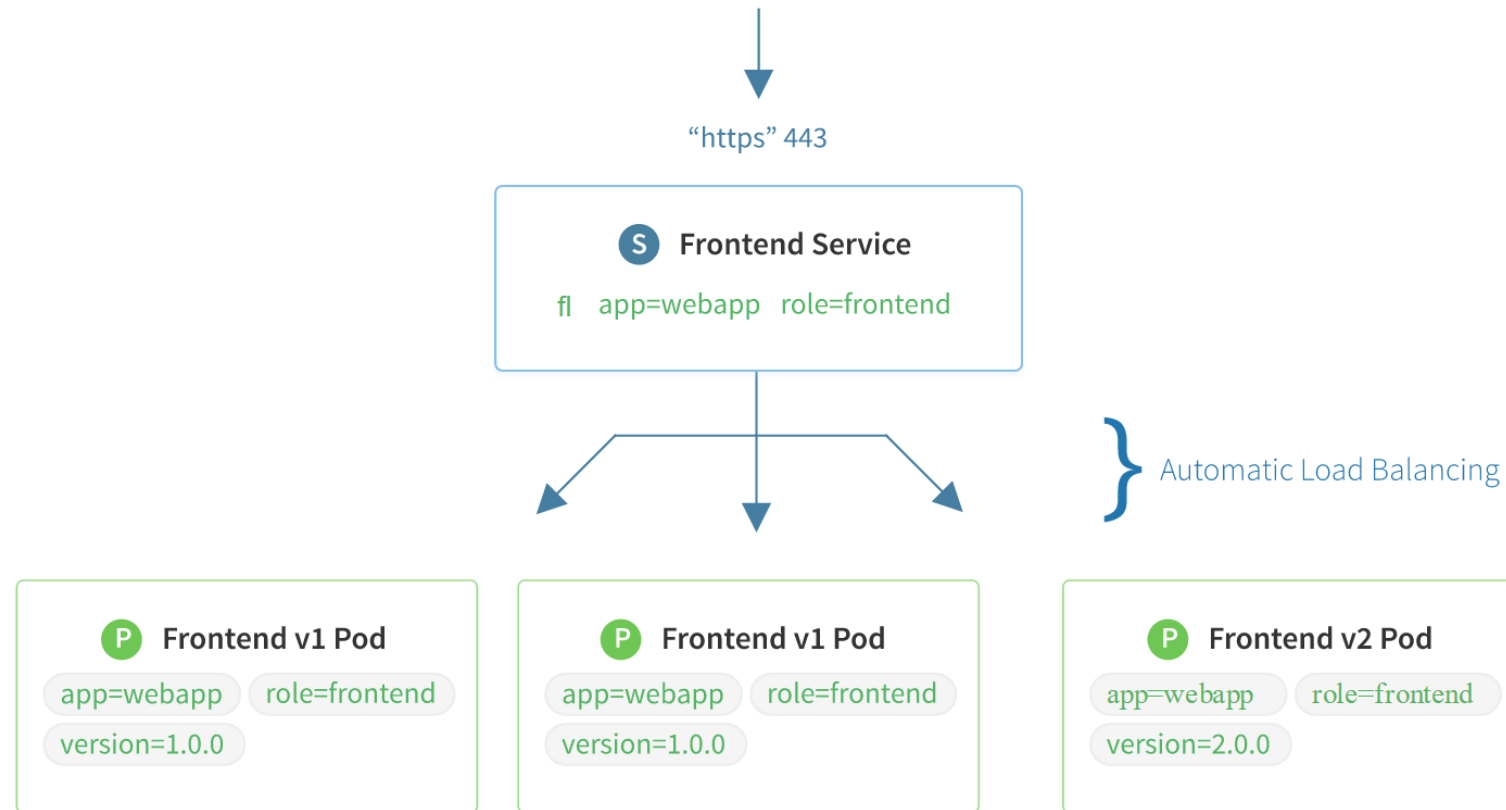
**Service:** Endpunkt unter einem definierten DNS-Namen, der Aufrufe an die Pods im Hintergrund verteilt (Load Balancing, Failover). Die für einen Service relevanten Pods werden über ihre Labels selektiert, z.B.: `role = apache, env != test, tier in (web, app)`

**Pod:** Gruppe an Containern, die auf dem selben Knoten laufen und sich eine Netzwerk-Schnittstelle inklusive einer dedizierten IP, persistente Volumes und Umgebungsvariablen teilen. Ein Pod ist die atomare Scheduling-Einheit in K8s. Ein Pod kann über sog. *Labels* markiert werden, das sind frei definierbare Schlüssel-Wert-Paare.

**Replica Sets / Replication Controller:** stellen sicher, dass eine spezifizierte Anzahl an Instanzen pro Pod ständig läuft. Ist für Reaktionen im Fehlerfall (Re-Scheduling), Skalierung und Rollouts (Canary Rollouts, Rollout Tracks, ..) zuständig.

**Deployment:** Klammer um einen gewünschten Zielzustand im Cluster in Form eines Pods mit dazugehörigem ReplicaSet. Ein Deployment bezieht sich nicht auf Services, da diese in der K8s-Philosophie einen von Deployments unabhängigen Lebenszyklus haben.

# Ein Beispiel für das Zusammenspiel zwischen Services und Pods.



## Quellen

- Services: <https://coreos.com/kubernetes/docs/latest/services.html>
- Pods: <https://coreos.com/kubernetes/docs/latest/pods.html>

# Eine Blaupause wird in Kubernetes per YAML oder JSON beschrieben.

## NGINX Pod

```
apiVersion: v1
kind: Pod
metadata:
  name: www
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - mountPath: /srv/www
      name: www-data
      readOnly: true
  - name: git-monitor
    image: kubernetes/git-monitor
    env:
    - name: GIT_REPO
      value: http://github.com/some/repo.git
    volumeMounts:
    - mountPath: /data
      name: www-data
  volumes:
  - name: www-data
    emptyDir: {}
```

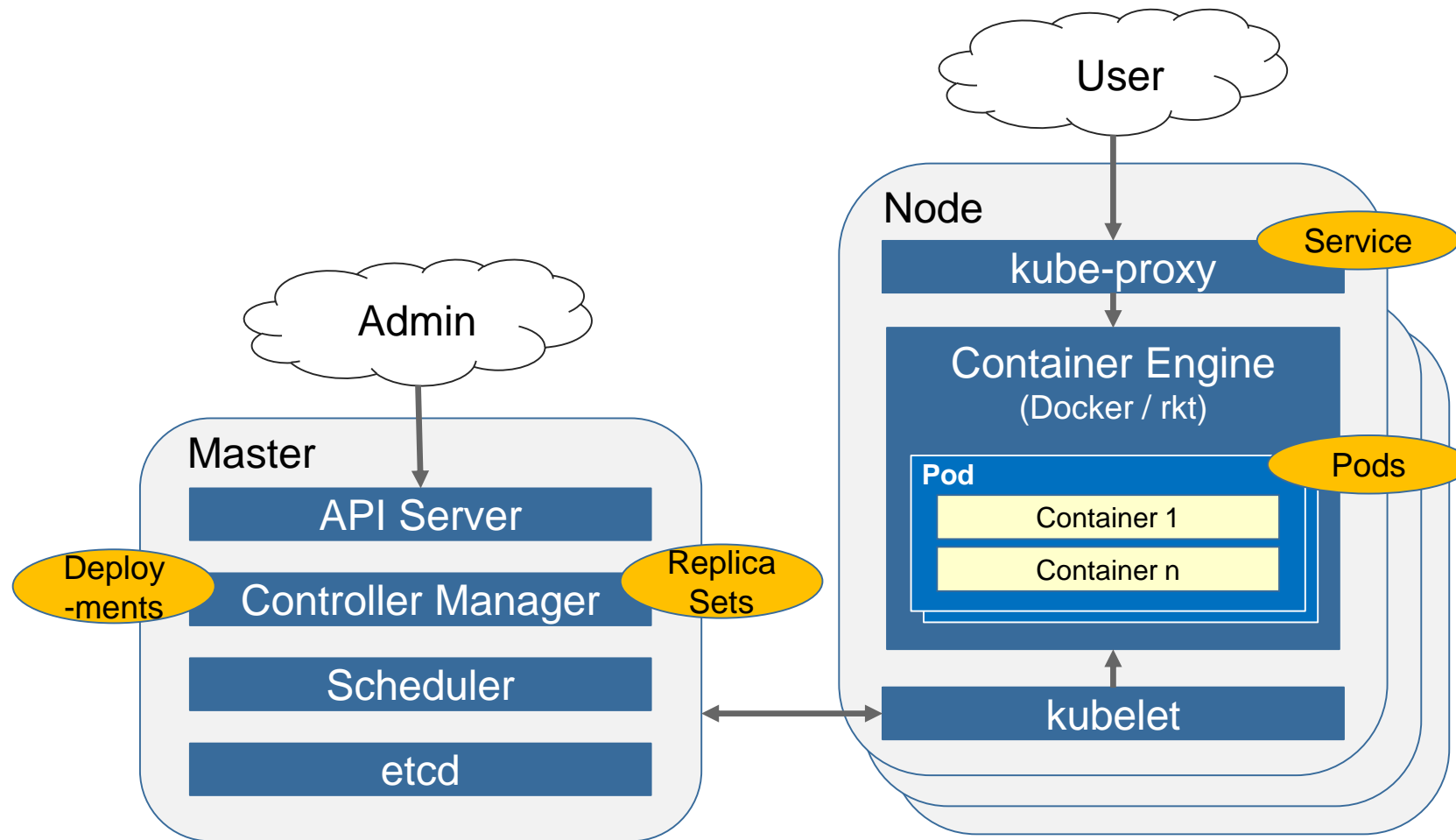
## NGINX Service

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  ports:
  - port: 8000 # the port that this service should serve on
    # the container on each pod to connect to, can be a name
    # (e.g. 'www') or a number (e.g. 80)
    targetPort: 80
    protocol: TCP
  # just like the selector in the replication controller,
  # but this time it identifies the set of pods to load balance
  # traffic to.
  selector:
    app: nginx
```

## NGINX Replication Controller (optional)

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-controller
spec:
  replicas: 2
  # selector identifies the set of Pods that this
  # replication controller is responsible for managing
  selector:
    app: nginx
  # podTemplate defines the 'cookie cutter' used for creating
  # new pods when necessary
  template:
    metadata:
      labels:
        # Important: these labels need to match the selector above
        # The api server enforces this constraint.
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

# Architektur von Kubernetes



- **API Server:** Stellt die REST API von Kubernetes zur Verfügung (Admin-Schnittstelle)
- **Controller Manager:** Verwaltet die *Replica Sets / Replication Controller* (stellt Anzahl Instanzen sicher) und *Node Controller* (prüfen Maschine & Pods)
- **Scheduler:** Cluster-Scheduler.
- **etcd:** Stellt einen zentralen Konfigurationsspeicher zur Verfügung.
- **Kubelet:** Führt *Pods* aus.
- **Container Engine:** Betriebssystem-Virtualisierung.
- **kube-proxy:** Stellt einen Service nach Außen zur Verfügung.

# Neben einem Cluster-Scheduler setzt Kubernetes auch noch auf Netzwerk- und Storage-Virtualisierungen auf.

## ■ Netzwerk-Virtualisierung (Overlay Network)

- OpenVSwitch

- Flannel

- Weave

- Calico

## ■ Storage-Virtualisierung (Persistent Volume), insbesondere zur Behandlung von zustandsbehafteten Containern.

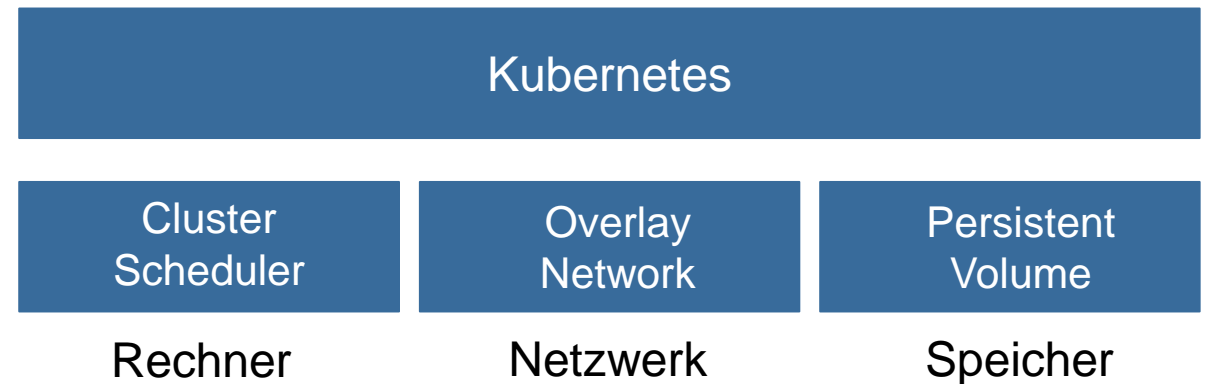
- GCE / AWS Block Store

- NFS

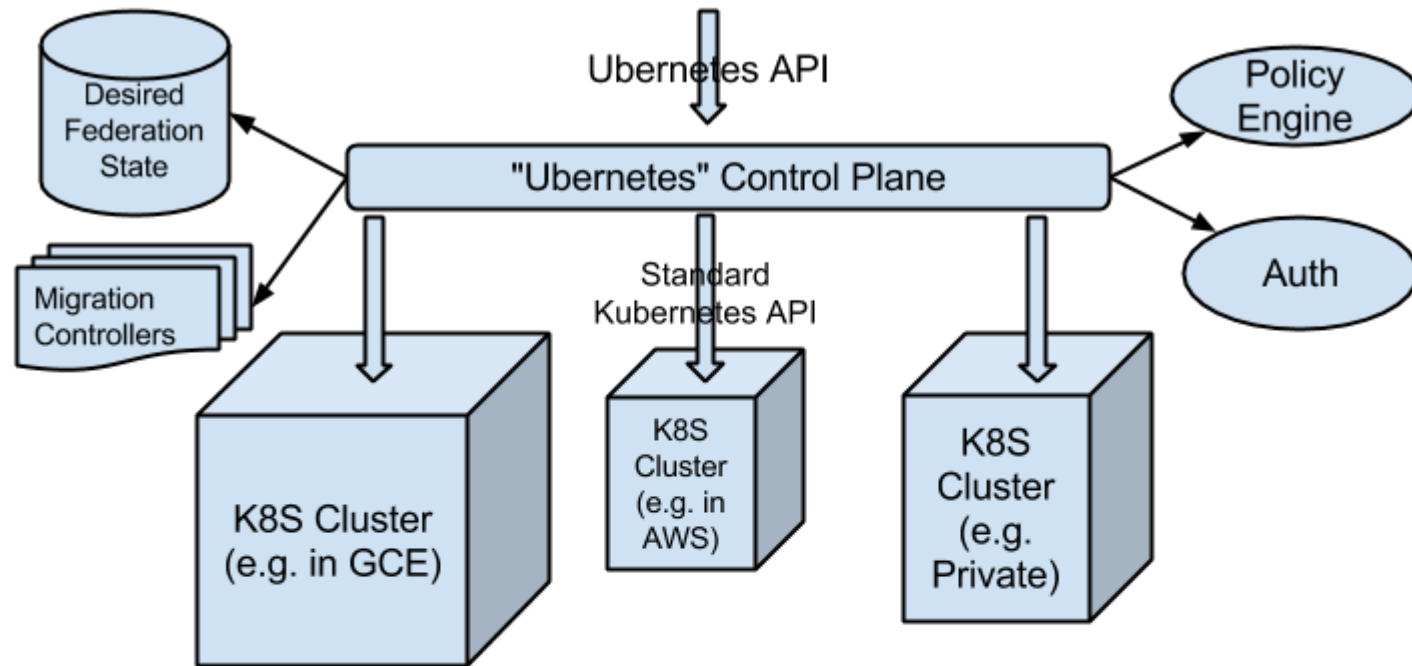
- iSCSI

- Ceph



- GlusterFS



# Ubernetes: Kubernetes für die hybride Cloud.



# Helm: Verwaltung von Applikationspaketen für Kubernetes.



## The package manager for Kubernetes

Helm is the best way to find, share, and use software built for Kubernetes.

### Search

Search for available charts.

```
$ helm search redis
```

redis-cluster (redis-cluster 0.0.5) - Highly available Redis cluster with multiple sentinels and standbys.

redis-standalone (redis-standalone 0.0.1) - Standalone Redis Master

### Install

Deploy the chart to your kubernetes cluster!

```
$ helm install redis-cluster
```

```
---> Running `kubectl create -f` ...  
services/redis-sentinel  
pods/redis-master  
replicationcontrollers/redis  
replicationcontrollers/redis-sentinel  
---> Done
```

# Quellen zu Kubernetes

- <https://dzone.com/refcardz/kubernetes-essentials>
- <https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes>
- <http://kubernetes.io>
- <https://www.udacity.com/course/scalable-microservices-with-kubernetes--ud615>