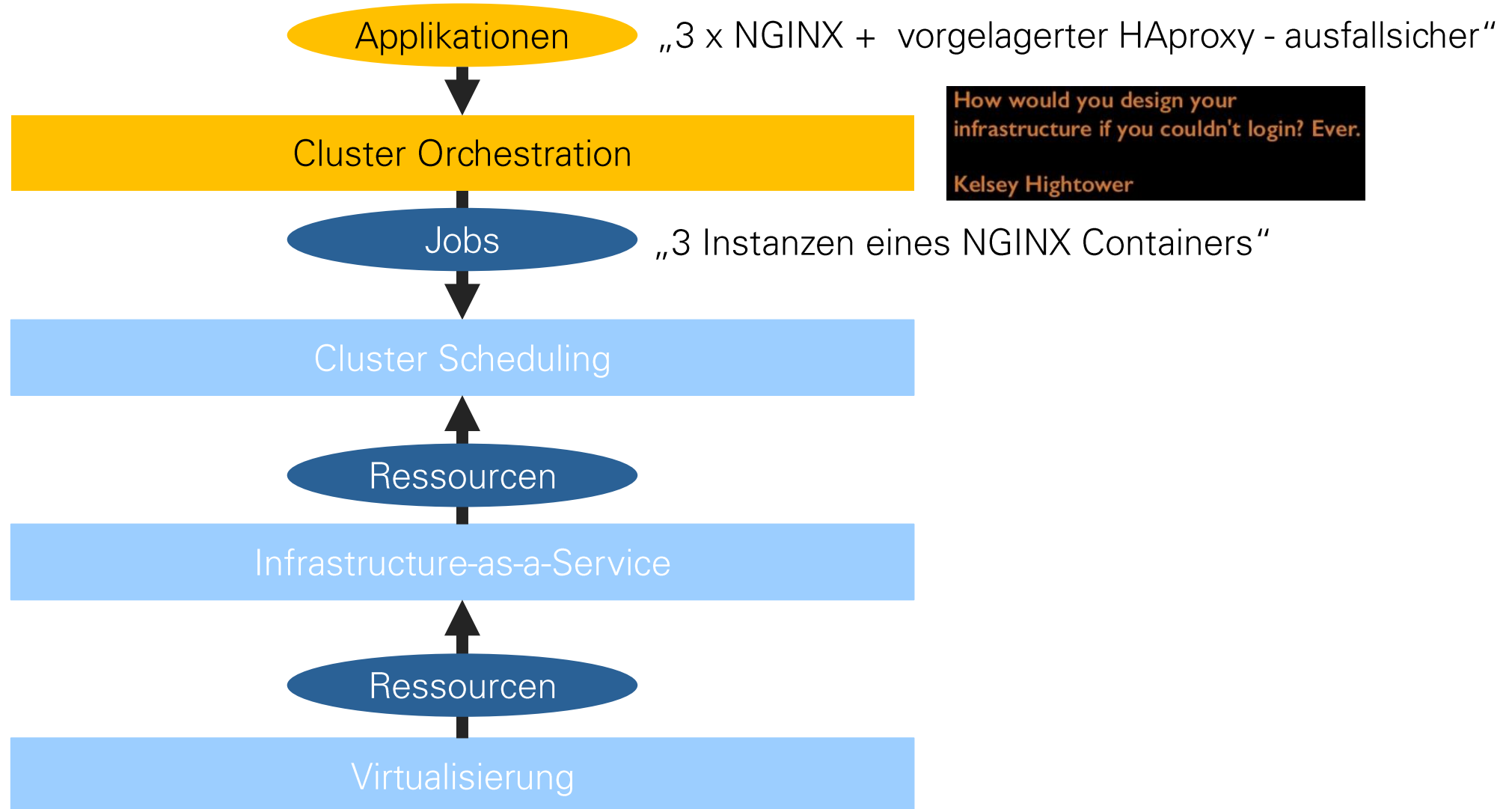


Kapitel 7: Cluster Orchestrierung



Vorlesung
CLOUD
COMPUTING

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.



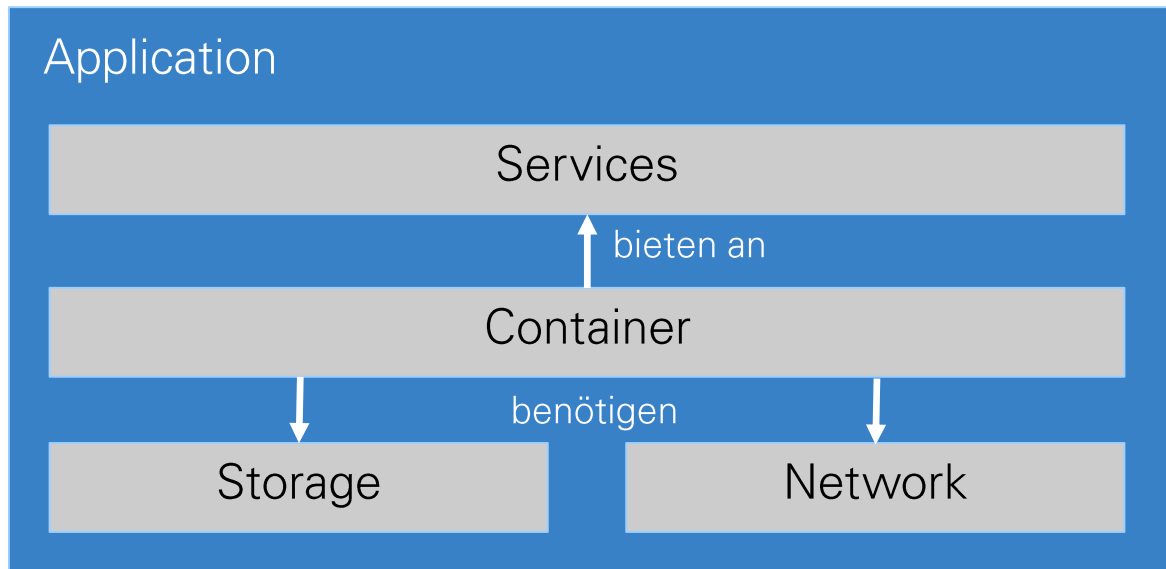
Cluster-Orchestrierer



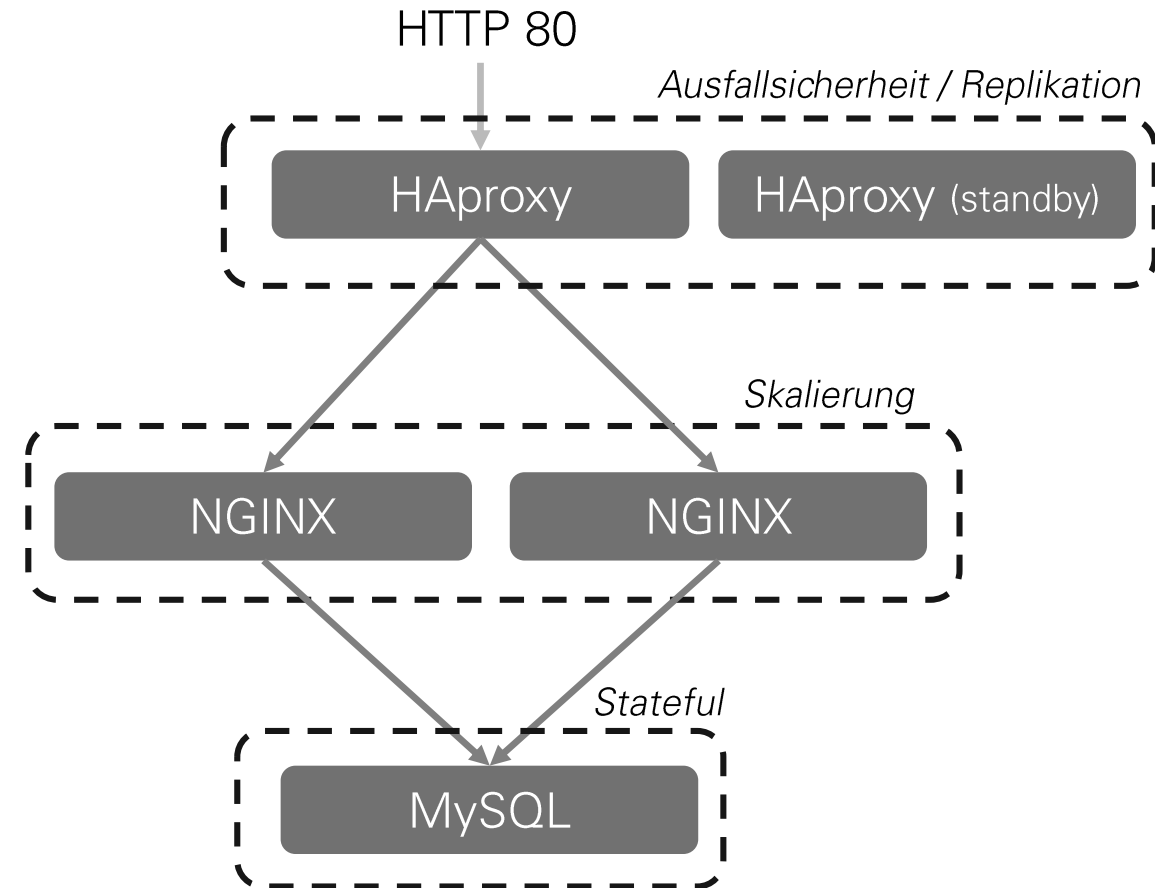
Steuerungsaktivitäten im Cluster:

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

Blaupause einer Anwendung (vereinfacht)



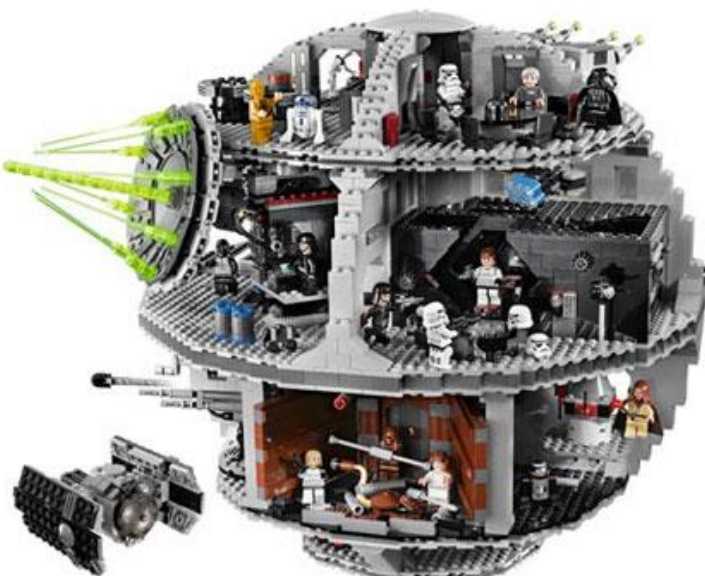
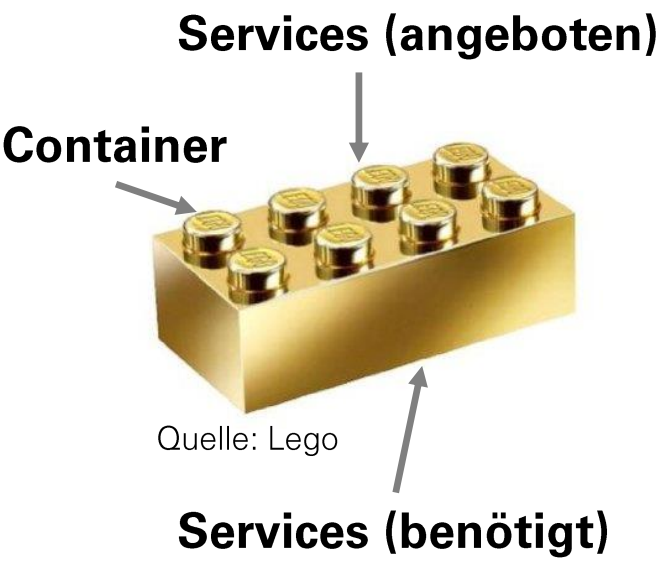
Metamodell



Modell

Analogie 1: Lego Star Wars

Cluster-Orchestrierer

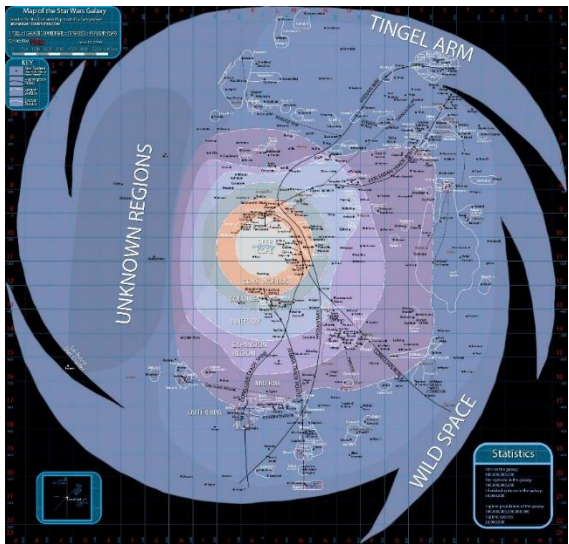


Quelle: Lego



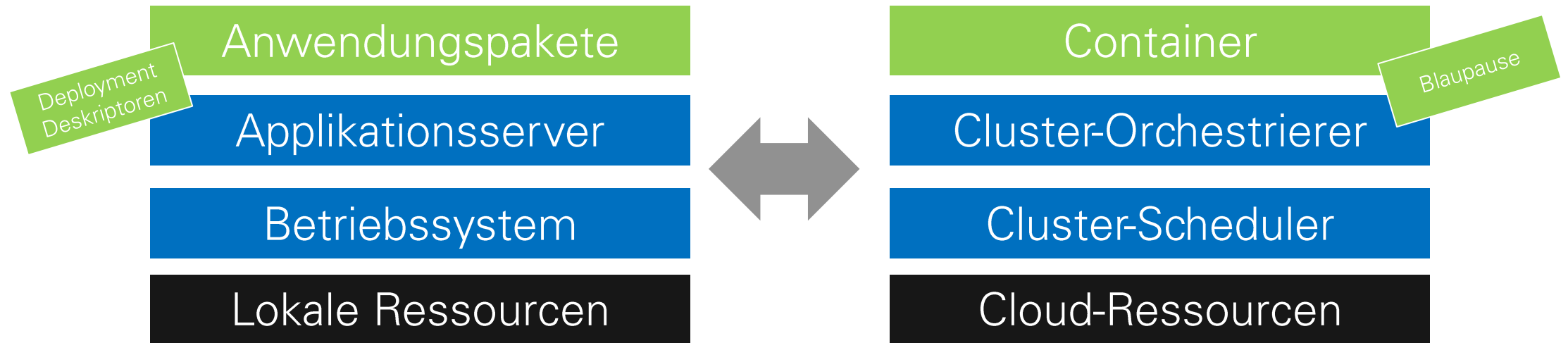
Blaupause

Cluster-Scheduler

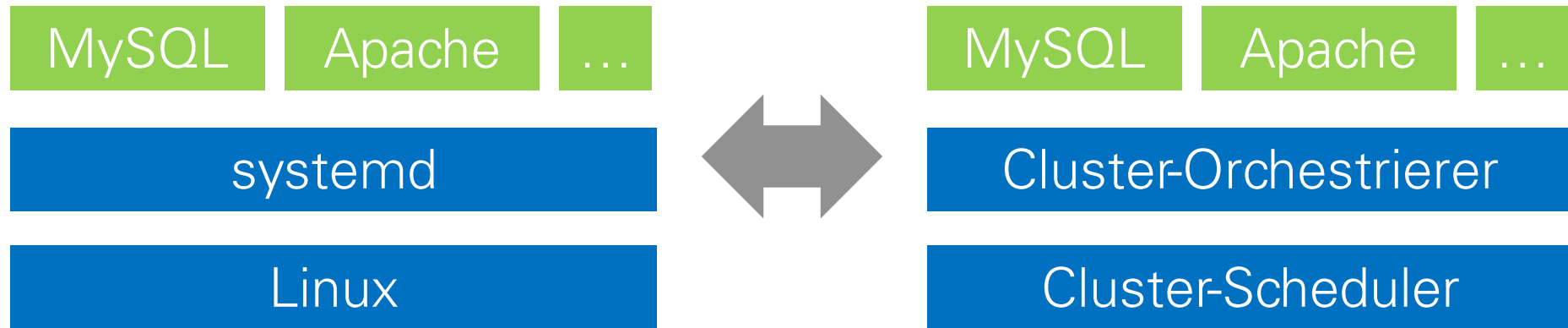


Quelle: wikipedia.de

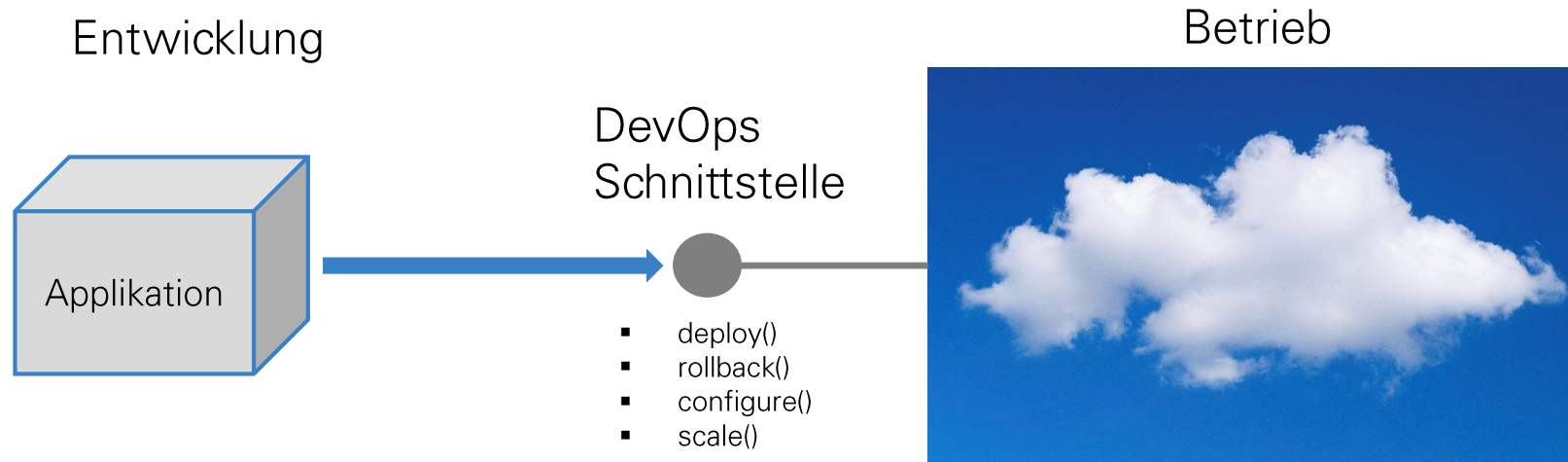
Analogie 2: Applikationsserver



Analogie 3: Betriebssystem



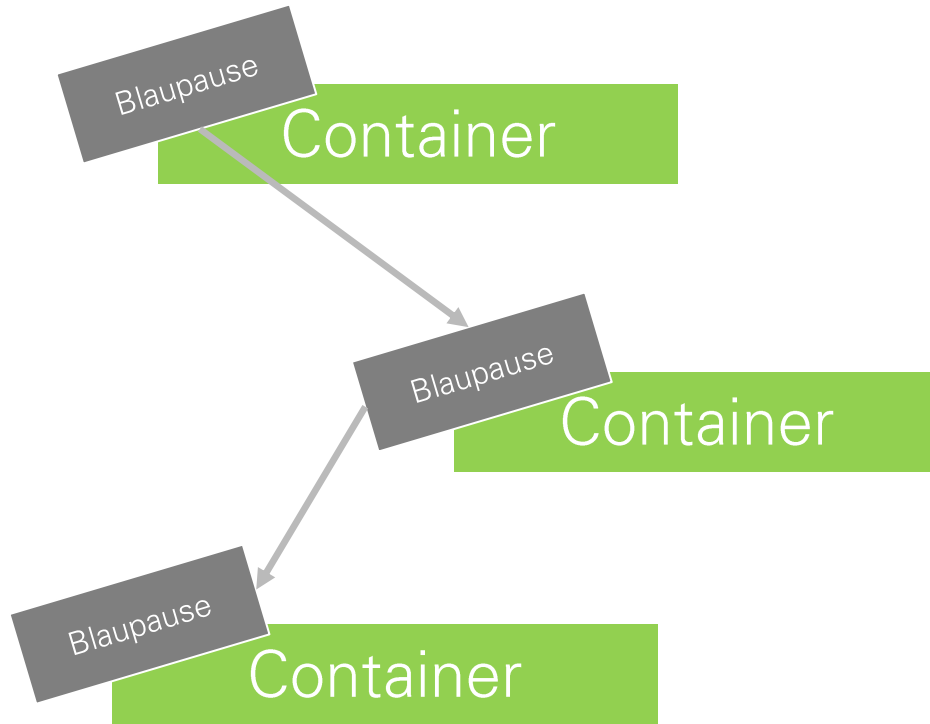
Ein Cluster-Orchestrierer bietet eine Schnittstelle zwischen Betrieb und Entwicklung für ein Cluster an.



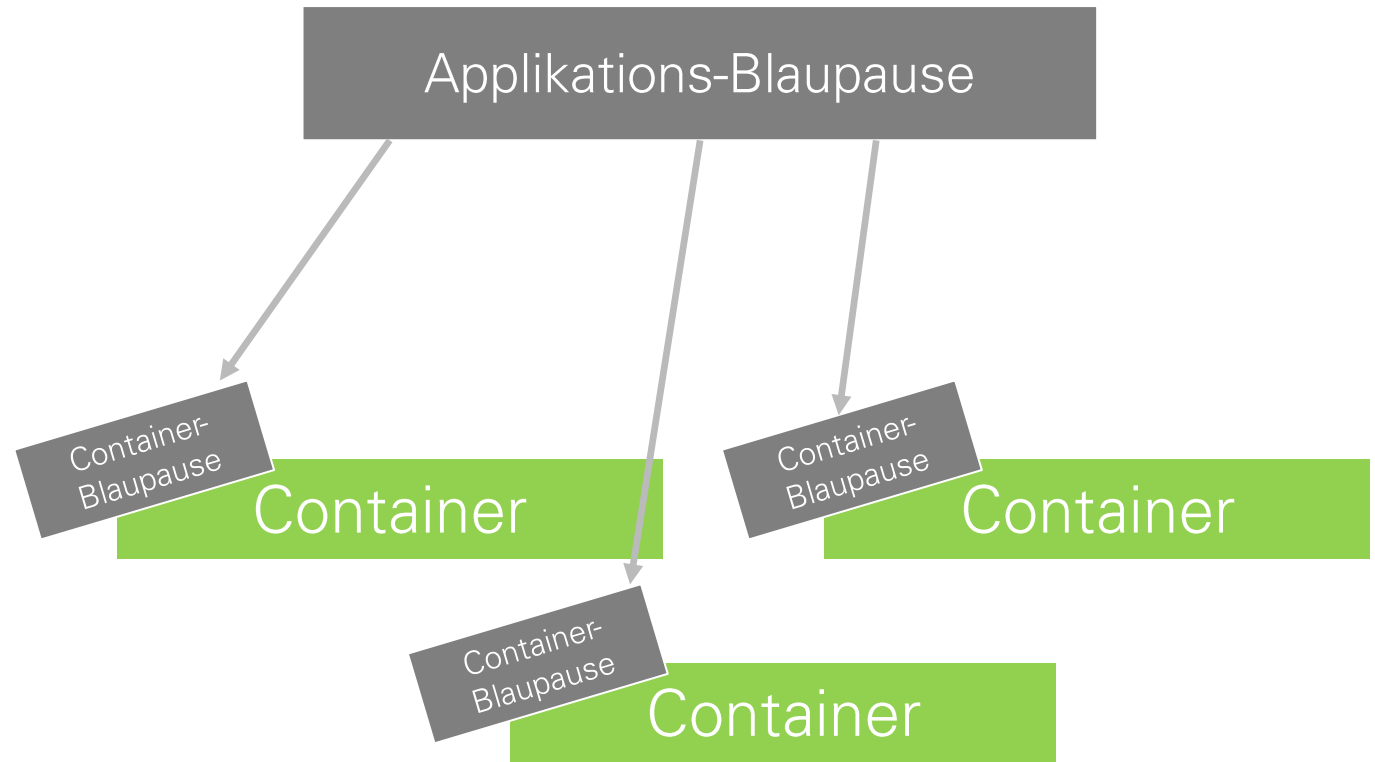
Ein Cluster-Orchestrierer automatisiert vielerlei Betriebsaufgaben für Anwendung auf einem Cluster.

- 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

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

Plain Docker

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

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

1-Level-Orchestrierung
(Container-Graph)

weba:

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

webb:

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

haproxy:

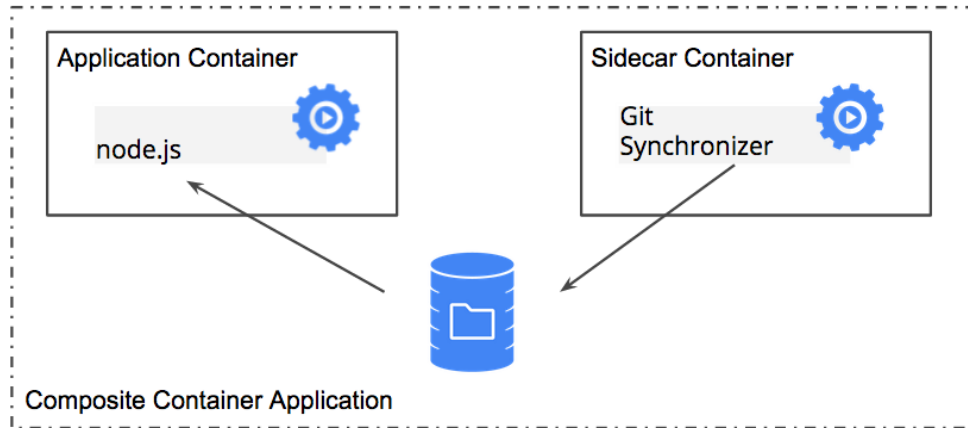
```
image: qaware/haproxy
links:
  - weba
  - webb
ports:
  - „80:80“
expose:
  - 80
```

FROM ubuntu
ENTRYPOINT nginx
EXPOSE 80

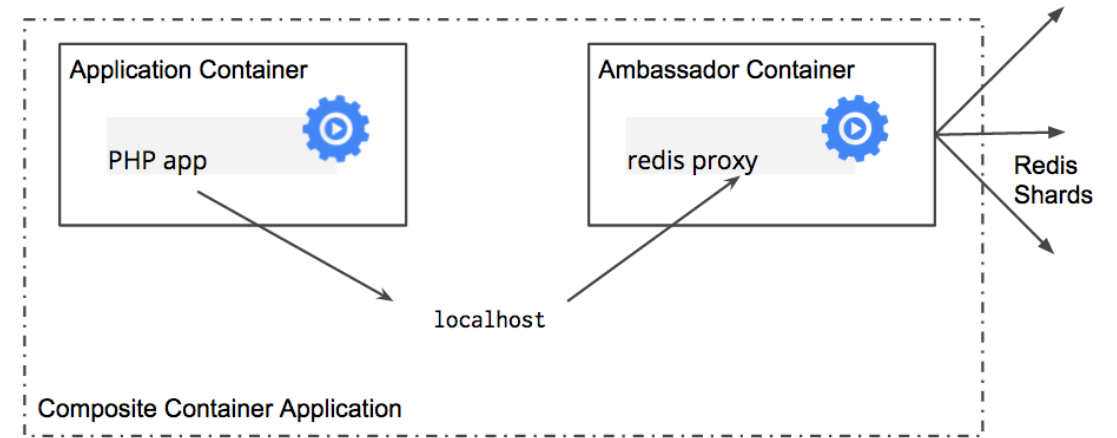
FROM ubuntu
ENTRYPOINT haproxy
EXPOSE 80

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

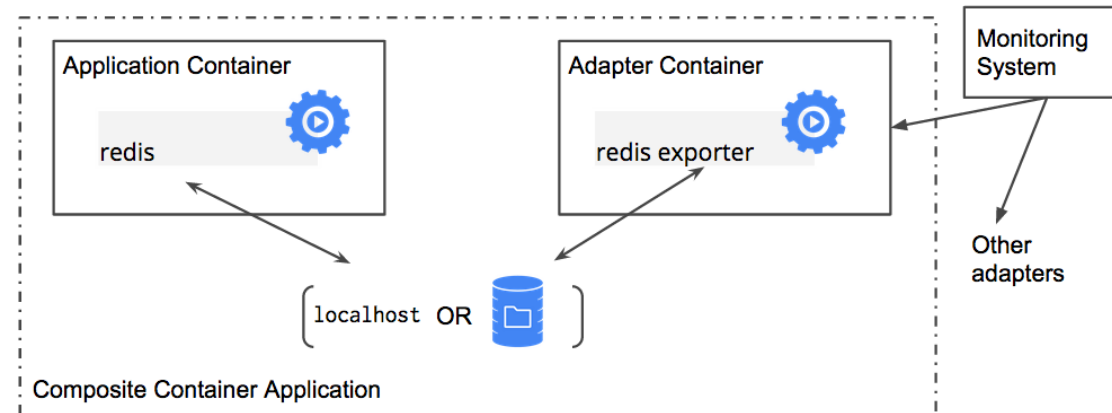
Orchestrierungsmuster



Sidecar Container



Ambassador Container



Adapter Container

Cluster Orchestrierer

- Kubernetes
- Apache Marathon & Chronos
- Docker Compose & Swarm

Kubernetes

Josef Adersberger @adersberger · Jul 21

Google spares no effort to launch
#kubernetes @ #OSCON



kubernetes by Google

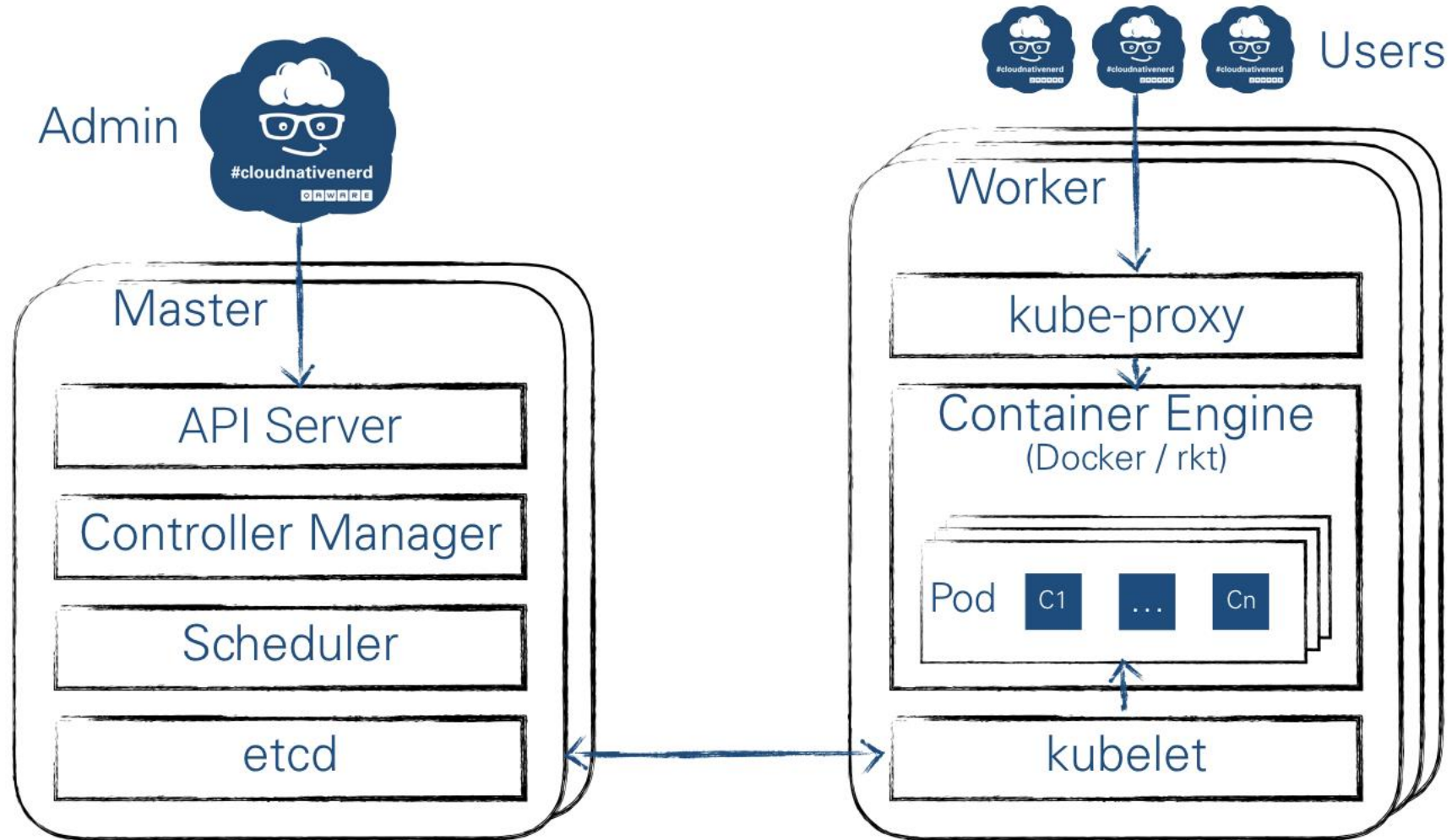
Manage a cluster of Linux containers as a single
system to accelerate Dev and simplify Ops.



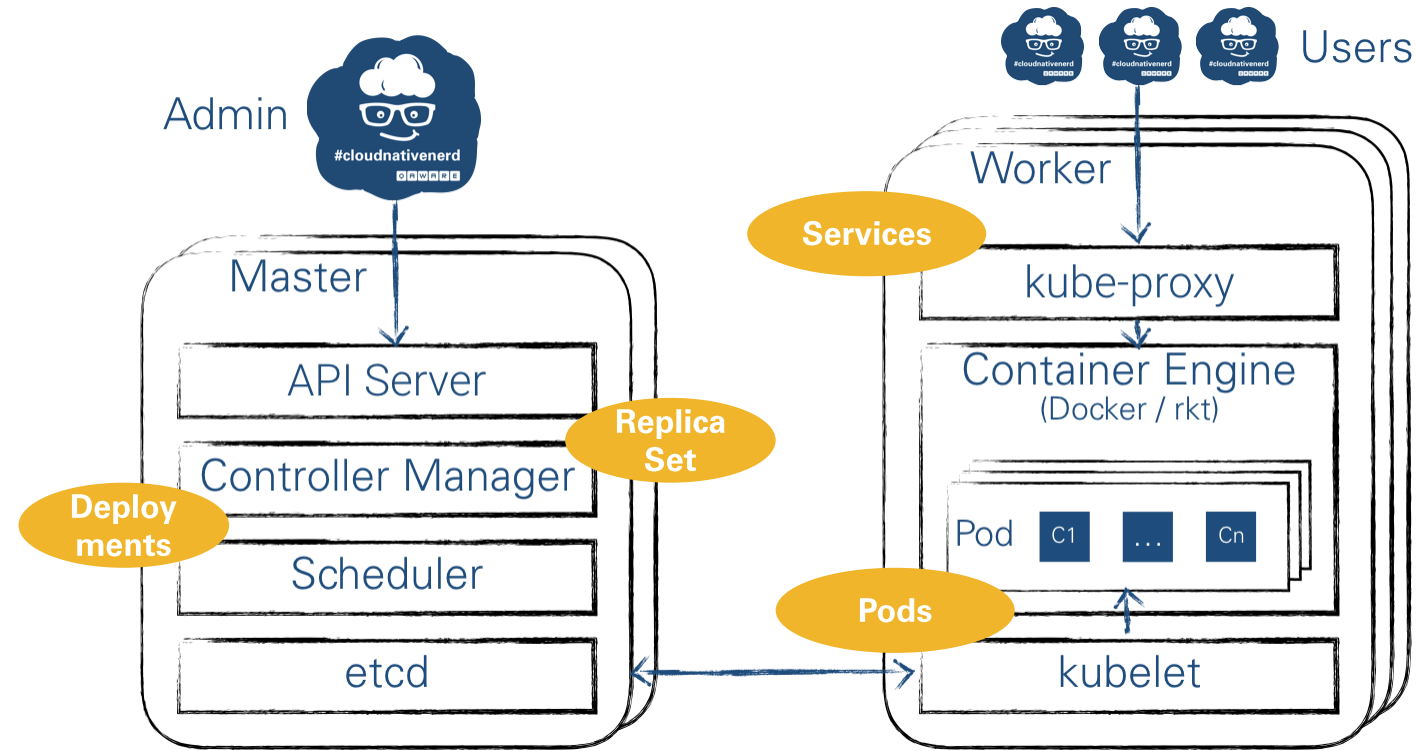
Kubernetes

- 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 1000 Nodes 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.
- Setzt den Standard im Bereich Cluster-Orchestrierung. Dafür wurde auch eigens die Cloud Native Computing Foundation gegründet (<https://cncf.io>).

Architektur von Kubernetes.



Aufgaben der Kubernetes Bausteine.



- **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.

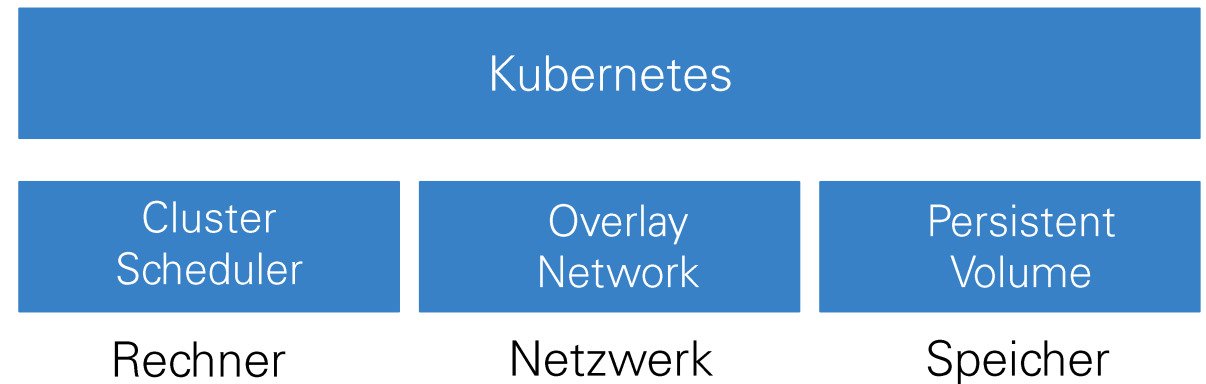
Neben einem Cluster-Scheduler setzt Kubernetes auch noch auf Netzwerk- und Storage-Virtualisierungen auf.

Netzwerk-Virtualisierung (Overlay Network)

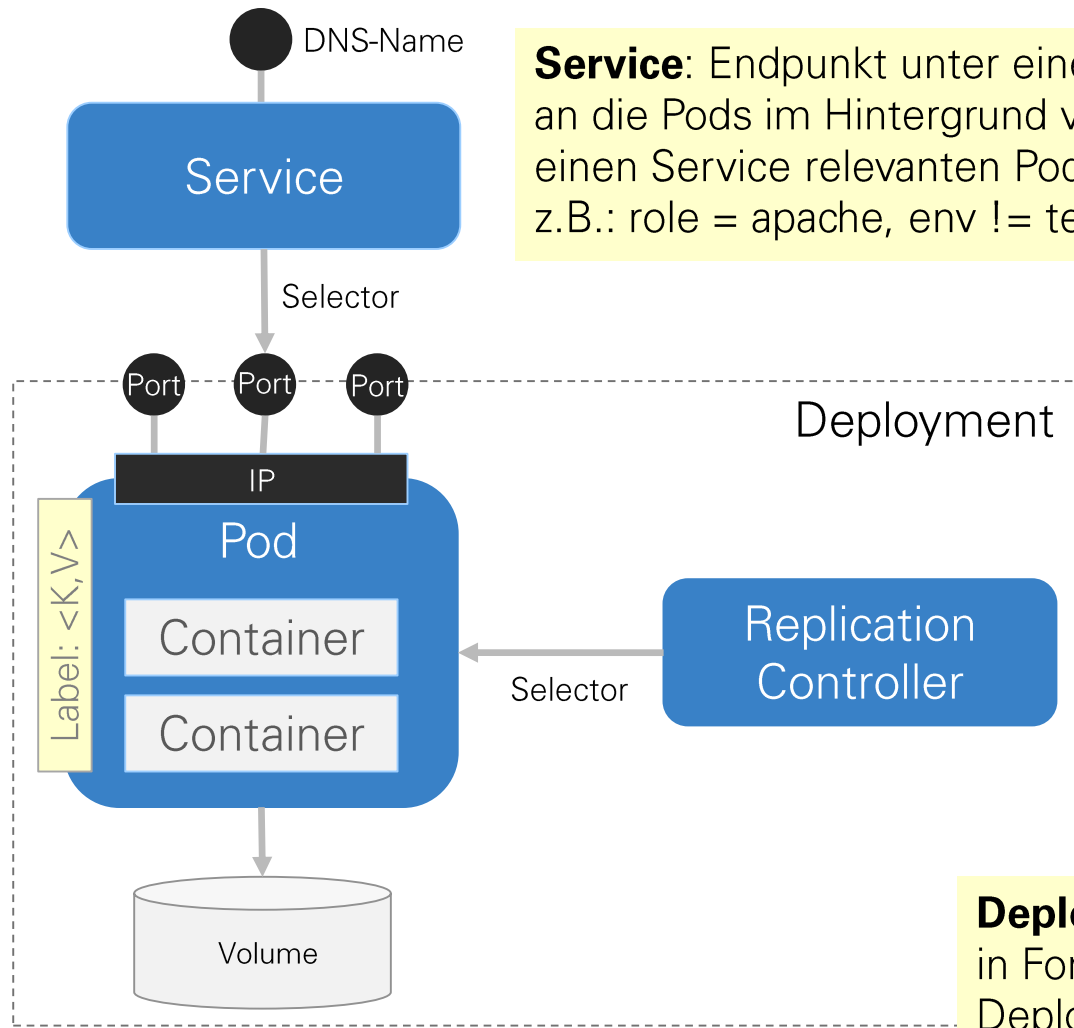
1. OpenVSwitch
2. Flannel
3. Weave
4. Calico

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

1. GCE / AWS Block Store
2. NFS
3. iSCSI
4. Ceph
5. GlusterFS



Der Kern-Abstraktionen von Kubernetes.



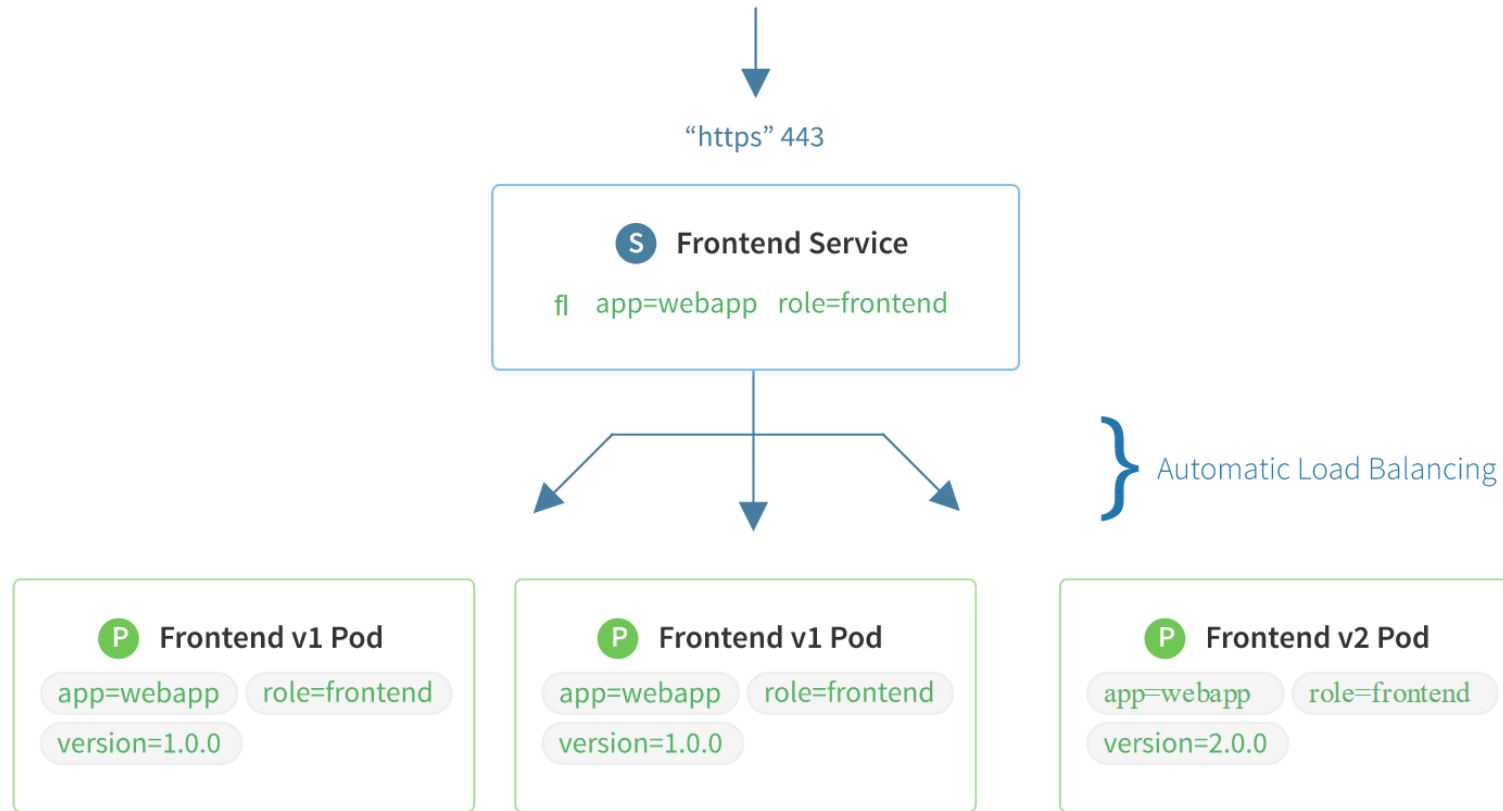
Service: Endpunkt unter einem definierten DNS-Namen, der Aufrufe an die Pods im Hintergrund verteilt (Load Balancing, Failover). Die für einen Service relevanten Pods werden über ihre Labels selektiert, z.B.: `role = apache, env != test, tier in (web, app)`

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.

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 Replication Controller. Ein Deployment bezieht sich nicht auf Services, da diese in der K8s-Philosophie einen von Pods unabhängigen Lebenszyklus haben.

Ein Beispiel für das Zusammenspiel zwischen Services und Pods über Labels.



Quellen

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

Deployment Definition

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: zwitscher-service
spec:
  replicas: 3
  template:
    metadata:
      labels:
        zwitscher: service
    spec:
      containers:
        - name: zwitscher-service
          image: "hitchhikersguide/zwitscher-service:1.0.1"
          ports:
            - containerPort: 8080
          env:
            - name: CONSUL_HOST
              value: zwitscher-consul
```

Resource Constraints

resources:

Define resources to help K8S scheduler

CPU is specified in units of cores

Memory is specified in units of bytes

required resources for a Pod to be started

requests:

memory: "128M"

cpu: "0.25"

the Pod will be restarted if limits are exceeded

limits:

memory: "192M"

cpu: "0.5"

Liveness und Readiness Probes

```
# container will receive requests if probe succeeds
```

```
readinessProbe:
```

```
  httpGet:
```

```
    path: /admin/info
```

```
    port: 8080
```

```
  initialDelaySeconds: 30
```

```
  timeoutSeconds: 5
```

```
# container will be killed if probe fails
```

```
livenessProbe:
```

```
  httpGet:
```

```
    path: /admin/health
```

```
    port: 8080
```

```
  initialDelaySeconds: 90
```

```
  timeoutSeconds: 10
```


Service Definition

```
apiVersion: v1
kind: Service
metadata:
  name: zwitscher-service
  labels:
    zwitscher: service
spec:
  # use NodePort here to be able to access the port on each node
  # use LoadBalancer for external load-balanced IP if supported
  type: NodePort
  ports:
    - port: 8080
  selector:
    zwitscher: service
```

Helm: Verwaltung von Applikationspaketen für Kubernetes.



spark-k8s-plain

- namespace-spark-cluster.yaml
- spark-master-controller.yaml
- spark-master-service.yaml
- spark-ui-proxy-controller.yaml
- spark-ui-proxy-service.yaml
- spark-worker-controller.yaml
- zeppelin-controller.yaml
- zeppelin-service.yaml

- `kubectl, kubectl, kubectl, ...`
- Konfiguration?
- Endpunkte?





spark-helm

- templates
 - _helpers.tpl
 - NOTES.txt
 - spark-master-deployment.yaml
 - spark-sql-test.yaml
 - spark-worker-deployment.yaml
 - spark-zeppelin-deployment.yaml
 - spark-zeppelin-ingress.yaml
- .helmignore
- Chart.yaml
- README.md
- values.yaml

- Chart suchen auf <https://hub.kubeapps.com>
- Doku dort lesen (README.md)
- Konfigurationsparameter lesen:
`helm inspect stable/spark`
- Chart starten mit überschriebener Konfiguration:
`helm install --name my-release
-f values.yaml stable/spark`

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
```

Quelle: <https://github.com/helm/helm>