OBJEKTspektrum

Container- und Cluster-Technologien für Microservices

supported by

FACH HOCHSCHULE LÜBECK
University of Applied Sciences

CoSA Center of Excellence

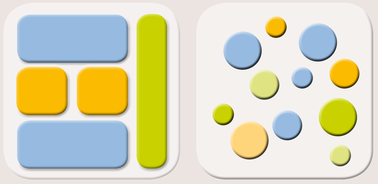
Inhaltliche Entwicklung: Prof. Dr. Nane Kratzke
Fachhochschule Lübeck, Mönkhofer Weg 239, 23562 Lübeck

Microservices

Definition:

„The microservice architectural style is an approach to develop a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are [...] independently deployable by fully automated deployment machinery. [...] Services [...] may be written in different programming languages and use different data storage technologies."

(M. Fowler, J. Lewis, Microservices, Blog Post on martinfowler.com)



- Der Microservice Architekturansatz befasst sich im Kern damit monolithische Systeme in kleinere (und damit handhabbare) und komponierbare Dienste zu zerlegen.
- Ein Microservice ist klein (ohne dass ‚klein' präzise definiert wäre) und löst idealerweise genau ein Problem auf wiederverwendbare Weise.
- Große (ehemals monolithische) Systeme werden aus diesen autonomen Bausteinen (wie Lego) zusammengesetzt.
- Häufig kommunizieren Microservices mittels REST-APIs (lose Kopplung).
- Zu Unternehmen die diesen Architekturstil erfolgreich anwenden gehören Amazon, Netflix, Soundcloud, eBay, Google, ...
- Es besteht totale Verantwortlichkeit in einem Team, keine DevOps Silos.

Wie bei jedem Architekturansatz

gibt es Vorteile

- Inhärente Modularisierung insbesondere durch Container
- Einfacheres Ausbringen (Deployment)
- Beherrschung technologischer Diversität

und Nachteile

- Steigende Komplexität verteilter Systeme (Entwicklung, Betrieb, Monitoring)
- Häufig Eventual statt Strong Consistency (CAP Theorem)

„The microservice approach has emerged from real-world use, taking our better understanding of systems and architecture to do SOA well. So [...] think of microservices as a specific approach for SOA in the same way that XP or Scrum are specific approaches for Agile Development."

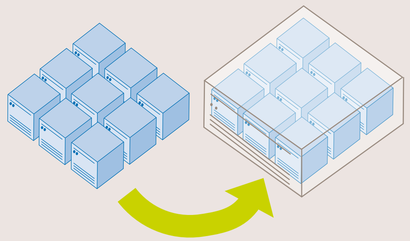
(Sam Newman, Building Microservices, O'Reilly, 2015)

Container Cluster machen die **Verteilung** und den **Betrieb** von komplexen Microservices beherrschbarer

Container Cluster

(Multi-Host Container Orchestration)

- Die Grundidee eines jeden Container Clusters ist es (tausende von) Rechnerknoten als eine logische Maschine zu verwalten.

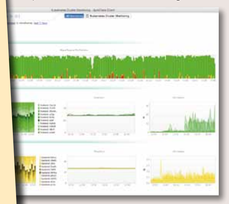


- Container werden auf diese logische Maschine aufgebracht.
- Verteilung, Load-Balancing, Rebalacing, Skalierung, etc. übernimmt der Container Cluster nach definierbaren Regeln und verbirgt so Komplexität.
- Container Cluster können über mehrere Public und Private IaaS Cloud Infrastrukturen ausgebracht werden und reduzieren damit Vendor Lock-In Problematiken.

Beispiele für Cluster Technologien:

- Core OS
- Kubernetes by Google
- Open Shift by Red Hat
- Docker Swarm
- Apache Mesos
- DCOS by Mesosphere
- EC2 Container Service by AWS

Dynatrace Monitor Plugin

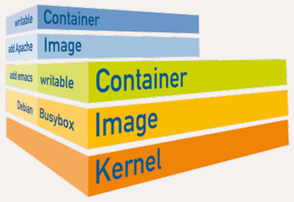


Container Technologien

Docker | HP-UX Containers | Rocket | LXC | Hyper-V Container | Solaris Zones | BSD Jails | +

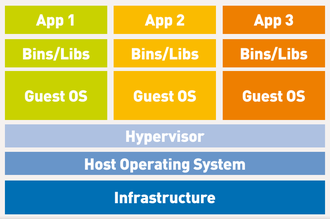
1 Container sind bewährt

- Die quelloffene Containertechnologie Docker basiert technologisch auf der Betriebssystemvirtualisierung von Linux.
- Betriebssystemvirtualisierung existiert seit Jahrzehnten und wird von vielen Betriebssystemen als Isolationsmöglichkeit für Prozesse angeboten.
- Docker nutzt Container um ausführbare und transferierbare Laufzeitumgebungen für Applikationen als Images bereitzustellen.



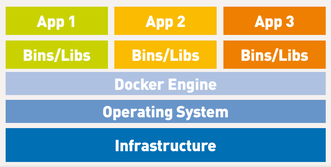
2 Container sind leichtgewichtig

Container sind im Vergleich zu virtuellen Maschinen ressourceneffizienter, allerdings müssen Gast- und Host-Betriebssystem gleich sein (Betriebssystemvirtualisierung).



Virtuelle Maschinen

- beinhalten die Anwendungen, erforderliche Binaries und ein komplettes Gastbetriebssystem. Dies kann schnell mehrere zehn GB an Daten bedeuten.



Container

- beinhalten nur die Anwendung und erforderliche Abhängigkeiten.
- teilen sich den Betriebssystemkern mit anderen Containern.
- laufen als isolierte Prozesse im Userspace des Betriebssystems.

3 Container sind kollaborativ

Container ermöglichen eine einfachere Verteilung von Anwendungen inkl. anwendungsbasierte Kollaboration

Mehr Freiheiten für Entwickler

- Entwickler sind weniger an „approved" Programmiersprachenstacks und Toolchains gebunden.
- Entwickler können problemangemessene Programmiersprachen und Tools für Anwendungen nutzen.

Eliminierung von Umgebungsinkonsistenzen

- Container beinhalten ihre Konfigurationsdateien und Abhängigkeiten.
- Die Anwendung wird daher auch auf allen Maschinen so laufen, wie sie lokal entwickelt, getestet und abgenommen wurde.
- Kein „It works on my machine" mehr.

Container wie Content teilen

- Docker Images können in einem Image-Repository bereitgestellt werden.
- Image Updates, Änderungen und Änderungshistorie können so mit einer Community oder innerhalb eines Unternehmens komfortabel geteilt werden.

4 Container sind dank Docker einfacher geworden

Ein Docker Image wird mittels einer einfachen Textdatei (Dockerfile) beschrieben:
Folgende Datei definiert einen einfachen statischen NGINX Webserver inkl. Inhalt.

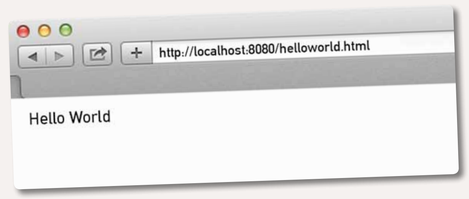
```
FROM nginx
RUN echo    „Hello World" > \
            /usr/share/nginx/html/helloworld.html
```

Ein Dockerfile kann in ein Image gewandelt werden

```
$> docker build –t helloworld .
```

und anschließend als Container ausgeführt werden.

```
$> docker run –p 8080:80 helloworld
```



sponsored by

dynatrace

Hewlett Packard Enterprise

ACANDO

New Relic.

OPITZ CONSULTING