BGD2

Container und Docker – Einführung und Motivation

Andreas Scheibenpflug

Container

Definition of CONTAINER

: one that contains: such as
a : a receptacle (such as a box or jar) for holding goods
b : a portable compartment in which freight is placed (as on a train or ship) for convenience of movement

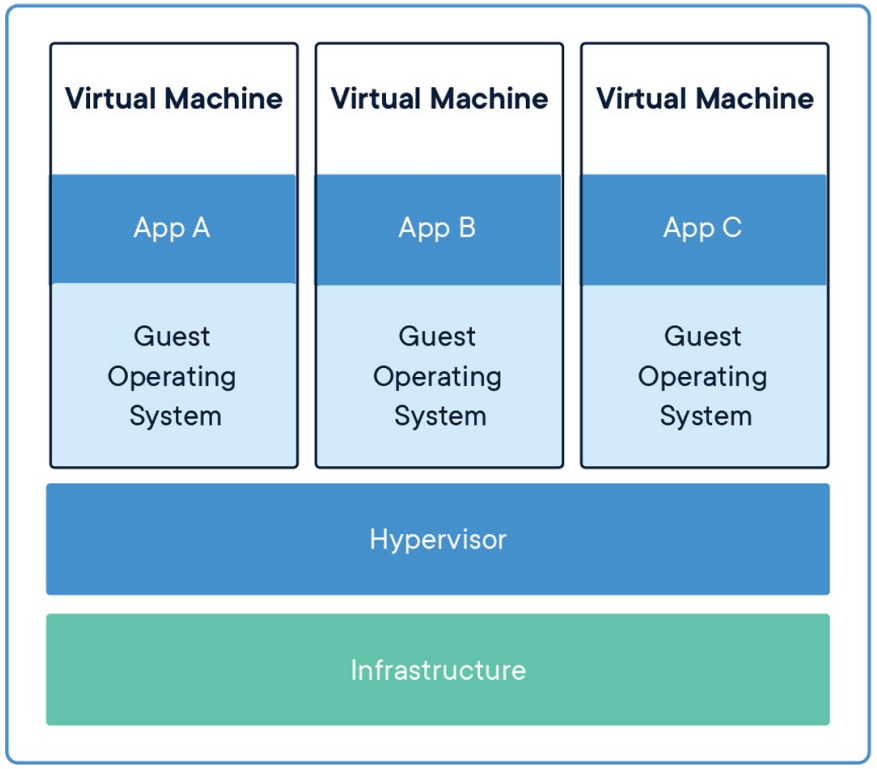
Container

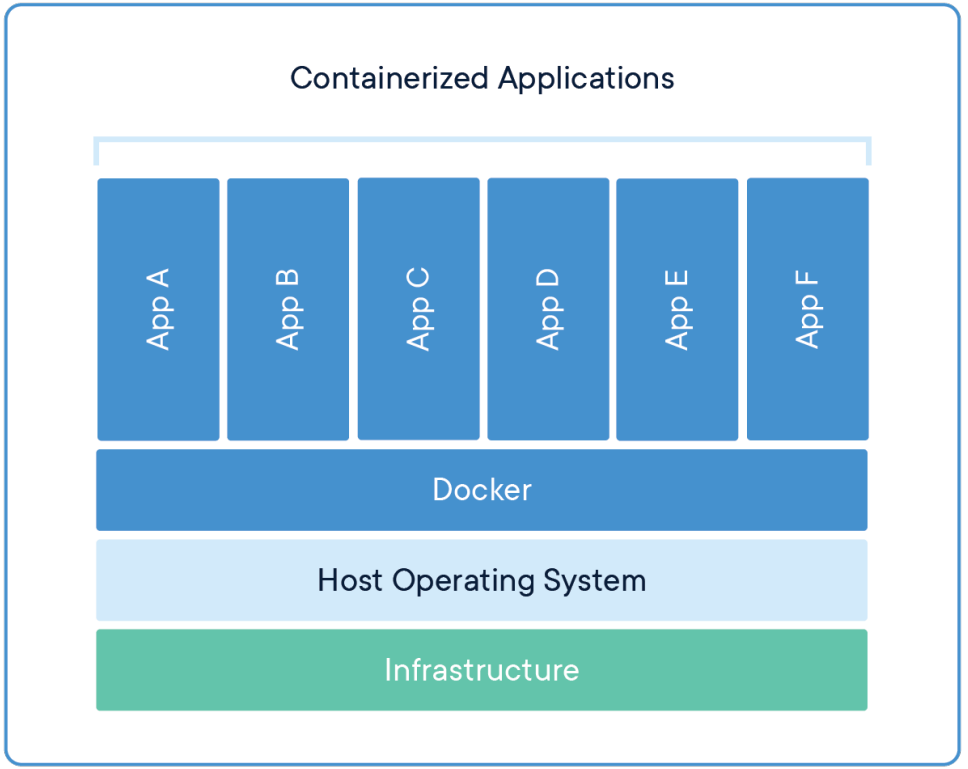
- Von einem Hostbetriebssystem verwaltete, abgegrenzte Umgebung, in der Prozesse ausgeführt werden
- Für Prozesse innerhalb des Containers stehen dezidierte Ressourcen (Dateisystem, Geräte, Dateien, Bibliotheken,...) zur Verfügung
- Prozesse in einem Container können nicht direkt auf Ressourcen des Hostbetriebssystems zugreifen (Ausnahmen bestätigen die Regel)

Motivation

Einige Themen im Umfeld Container und Docker

Virtuelle Maschinen vs. Container





https://www.docker.com

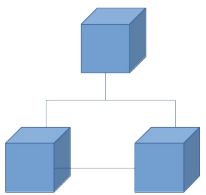
A little bit of history

in software architecture

Monolith

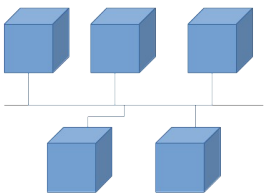
TCP/IP

Komponenten/RPC



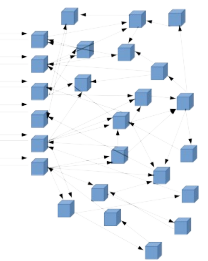
COM/CORBA/RMI/EJB

(EAI) / SOA



XML/SOAP/VM

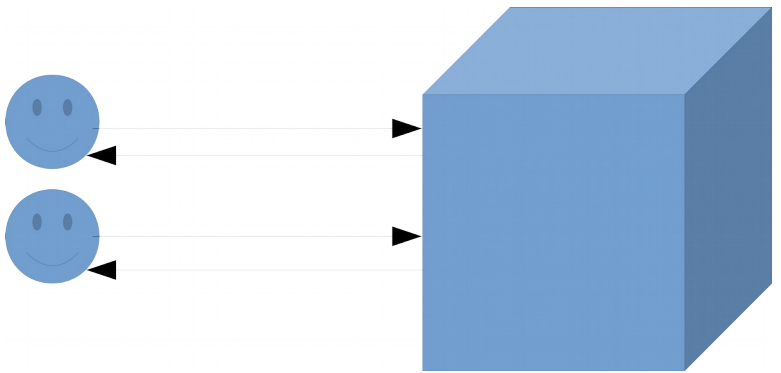
Microservices



REST/MQ/Container

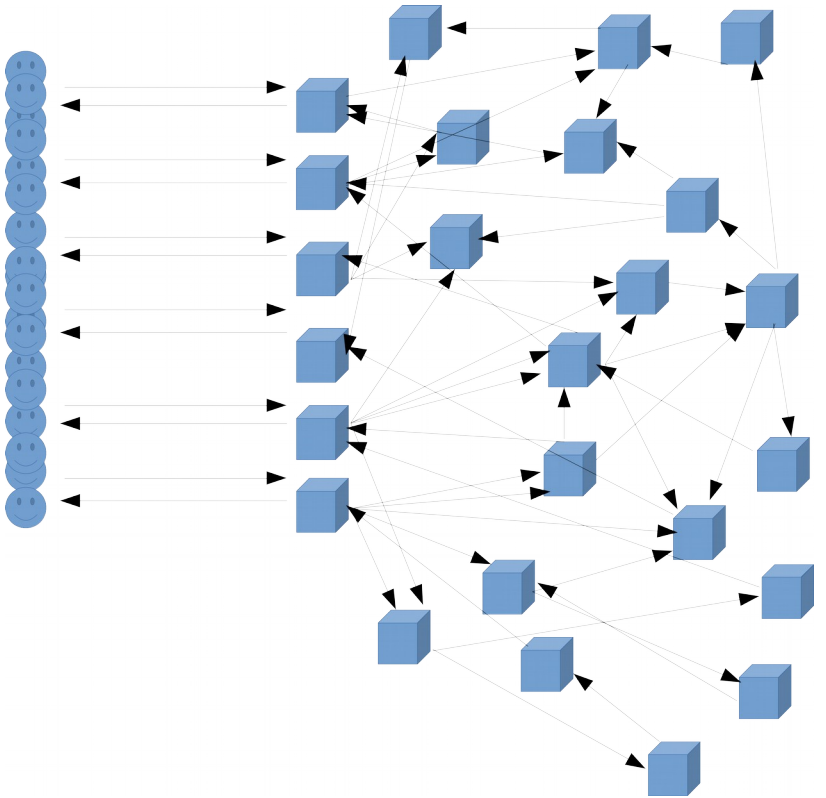
Früher war alles besser

- Wenige Benutzer
- Ein Server
- Ein Programm

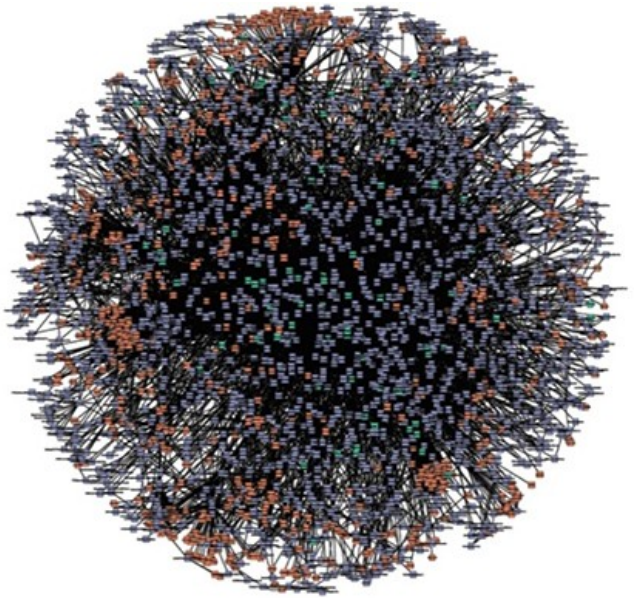


State of the Union

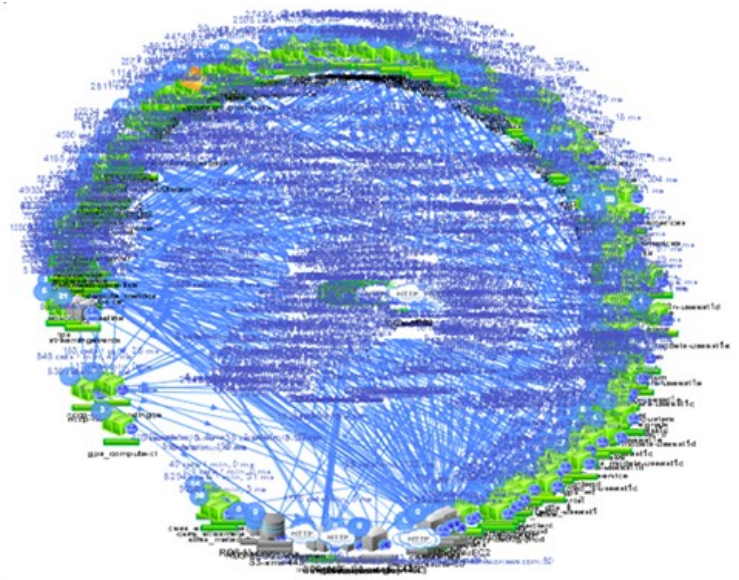
- Millionen Benutzer
- Sehr, sehr viele
  - Services
  - Computer
  - Deployments



Die üblichen Verdächtigen



amazon.com



NETFLIX

https://www.appcentrica.com/the-rise-of-microservices/

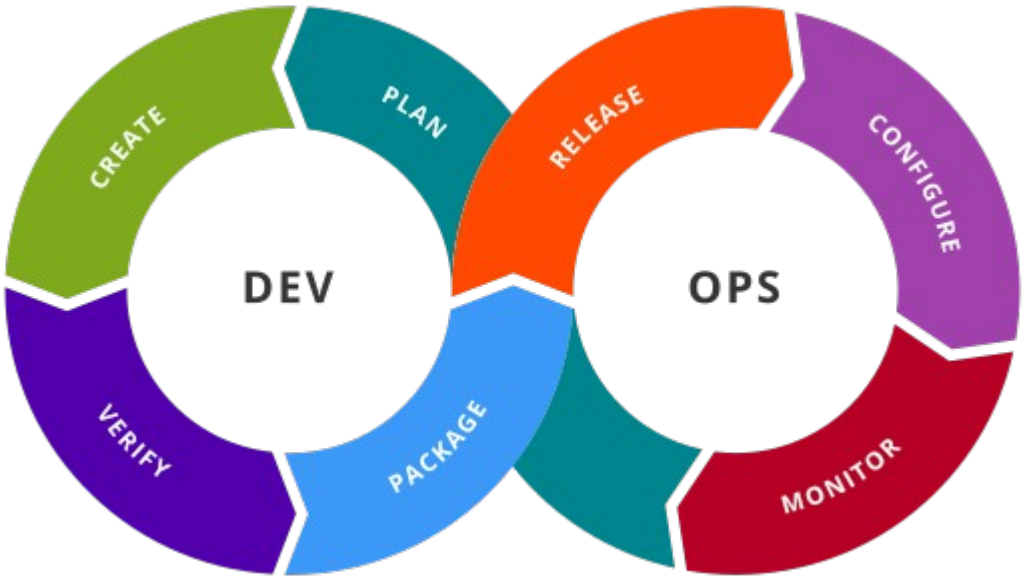
Zusammenfassung

- Softwaresysteme werden komplexer und sind lose gekoppelt → Microservices
- Komplexität der Konfiguration / Deployments steigt
- Hohe Anzahl an Benutzer einer Software → Skalierung
- Komplexität hat einen Einfluss sowohl auf die Architektur als auch auf den Softwareentwicklungsprozess

DevOps?

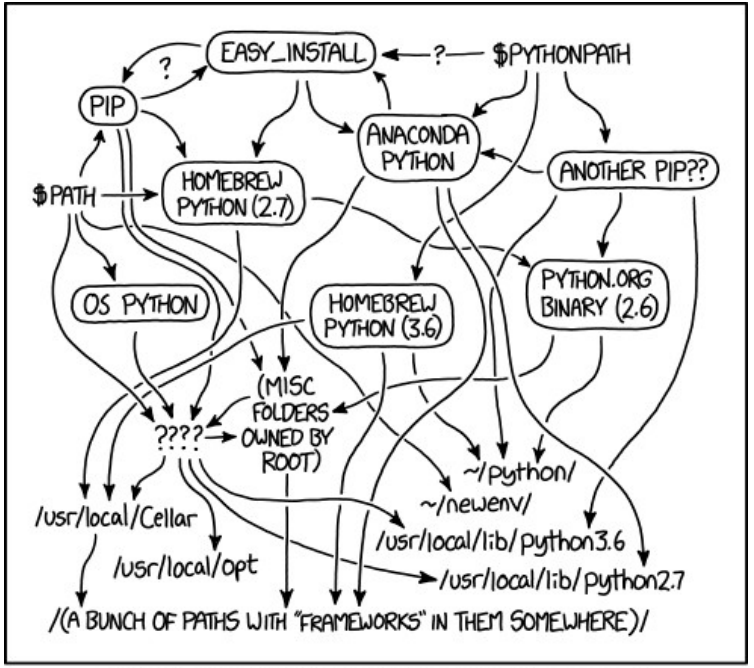
DevOps

- Deployment von komplexen Softwaresystemen
  - Production
  - Test
- Release early, release often
- Automate everything
  - Build
  - Test
  - Package
  - Release
  - Configure
  - Monitor



Dependency Hell

- Container enthalten genau jene Abhängigkeiten einer Anwendung, die auch benötigt werden



https://tech.instacart.com/freezing-pythons-dependency-hell-in-2018-f1076d625241

Eigenschaften Container 1/2

- Gekapselt
  - Ein Container stellt eine in sich geschlossene, vollständige und konfigurierte Laufzeitumgebung für eine Anwendung zur Verfügung
- Portabel
  - Ein Container kann auf jedem beliebigen Hostbetriebssystem ausgeführt werden, solange darauf die Container Engine läuft
- Sicher
  - Ein Container kann weder dem Hostbetriebssystem noch anderen darauf laufenden Containern Schaden zufügen

Eigenschaften Container 2/2

- Effizient
  - Im Vergleich zu virtuellen Maschinen wird kein Hypervisor benötigt
- Skalierbar
  - Clonen von Containern und Verwaltung mittels Orchestration
- Konfiguration
  - Einfache Verwaltung der Konfiguration von Umgebungen (Infrastructure as Code)

Container Implementierungen (1/2)

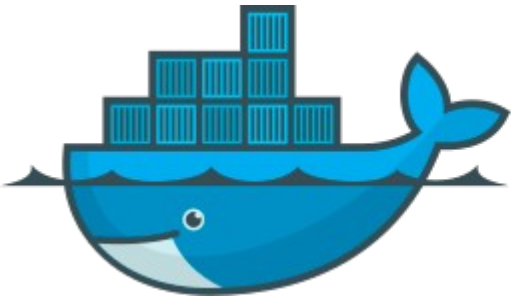
- 1979 Version 7 Unix: chroot („change root“)
  - Ändert für Kindprozesse „/“ im Dateisystem
  - `chroot [OPTION] NEWROOT [COMMAND [ARG]...]`
- 2000 FreeBSD 4.0: jail
  - Isolation von Prozessen
  - Fokus auf Sicherheit
- 2001/04 Solaris Zones
  - Komplette Container Lösung
  - Zuteilung von Ressourcen zu Zones

Container Implementierungen (2/2)

- 2001-2005 Virtuozzo
  - OpenVZ wurde unter die GPL Lizenz gestellt
  - Erlaubt Zuteilung/Einschränkung von Ressourcen (z.B. CPU) zu Containern
- 2008 Linux Containers LXC
  - Baut auf existierenden Linux Kernel Technologien auf (z.B. cgroups, namespaces)
  - Erste Versionen von Docker verwendeten LXC
- 2013 Docker
  - Verbesserungen in der Usability von Containern, vor allem durch Images
- 2016 Windows Server 2016
  - Windows Containers

Docker

- Set von Werkzeugen um
  - Container zu erstellen (Images)
  - zu konfigurieren (Dockerfile)
  - und auszuführen (Docker Engine)
- Docker Hub: Plattform zur Verteilung von Images



Recap - Container

- Ein Container hat
  - Eigene Prozesse
  - Eigene Benutzer
  - Eigenes Netzwerk (und andere Geräte)
  - Es kann eigene Software installiert werden
  - Es können Anwendungen und Services ausgeführt werden

Recap - Container

- Container können nicht (im Gegensatz zu VMs)
  - eigene Kernel verwenden (es wird der Kernel des Hostbetriebssystems verwendet)
  - ein anderes Betriebssystem ausführen
  - fremde Hardware emulieren (Ausnahmen ausgenommen)
  - Kernel Module laden
- Init ist nicht der erste Prozess der gestartet wird
  - Anders gesagt: Es wird kein Betriebssystem gebootet

Recap - Container

- Daher sind Container und Prozesse „im Container“
  - Ganz „normale“ Prozesse
  - deren Rechte eingeschränkt werden
  - und vom Hostbetriebssystem direkt verwaltet (im Gegensatz zu VMs)
- Linux Kernel stellt Funktionalität bereit, um Prozesse in deren Rechten und Sichtbarkeiten einzuschränken