# Evaluation verschiedener Container-Technologien

Der Trend zur Cloud ist unumkehrbar. Bereits heute setzen Firmen wie Microsoft verstärkt auf das Cloudgeschäft [1]. Die treibende Technologie hinter diesem Trend sind Container. Im folgenden wird diese Technologie genauer betrachtet. Dabei soll die Frage beantwortet werden, wie Docker die populärste Technologie wurde, welche aktuellen Probleme bestehen und wie versucht wird, diese zu lösen.

**Container: Was ist das?**

Container dienen der Isolation von Prozessen. Dabei sind sie ressourcensparender und deutlich schneller als Virtuelle Machinen. Diese Einsparungen werden erreicht, indem nicht ein gesamtes Betriebssystem virtualisiert, sondern lediglich eine Isolation einzelner Systemkomponenten vorgenommen wird.

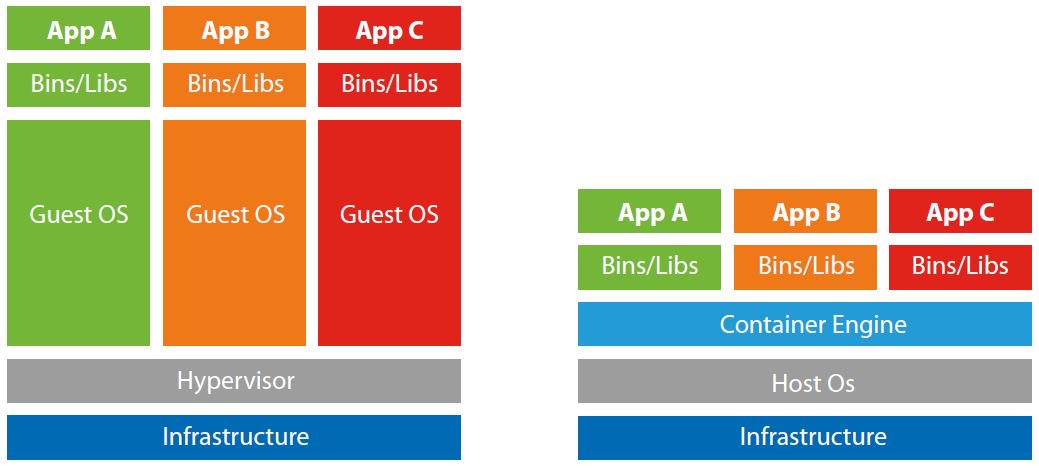
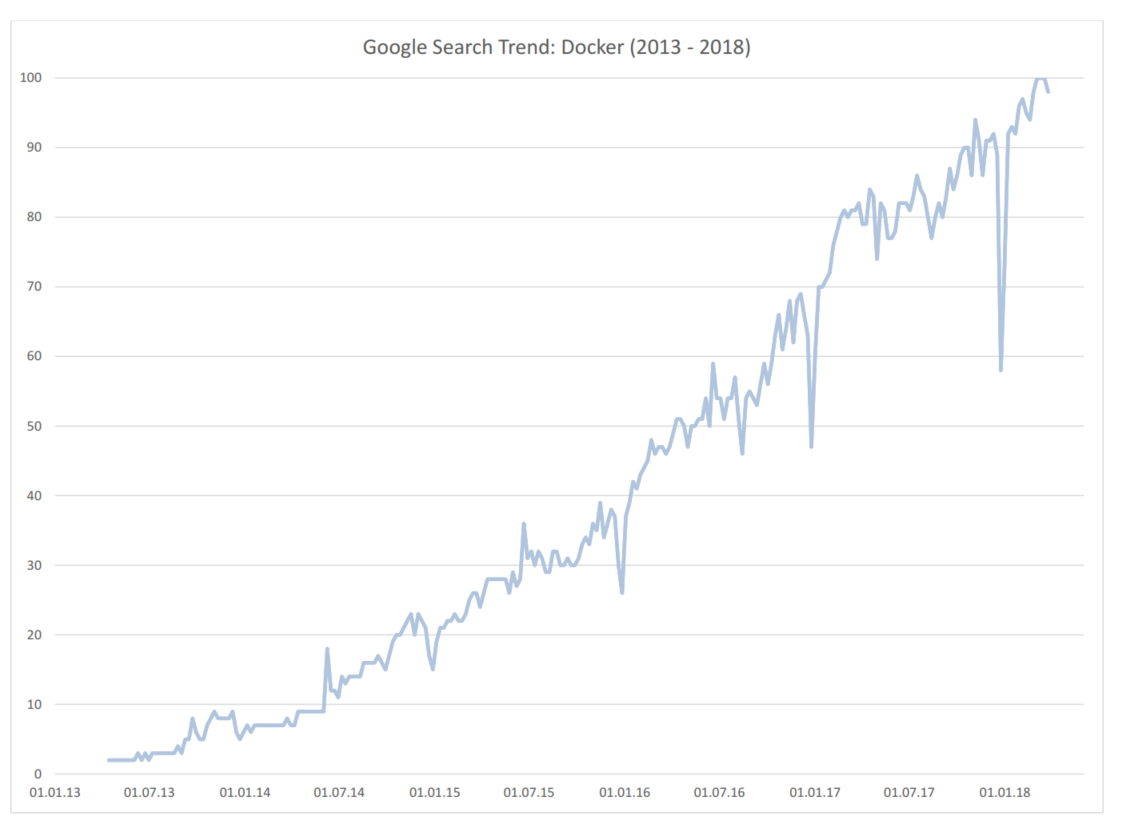


Figure 1 Container und VM Virtualisierungsstack (1)

Durch diese Methode sind Container hoch skalierbar und können binnen Bruchteile von Sekunden gestartet oder zerstört werden. Diese Skalierbarkeit und Isolation sind die tragenden Faktoren der heutigen Cloud-Architekturen.

**Docker: Der Platzhirsch unter den Containern**

Die grundlegenden Konzepte zur Isolation wurde bereits in den frühen 80er mit der Einführung des Unix Systemaufrufs chroot implementiert. Der erste große Durchbruch kam allerdings erst 2008 mit LXC [2]. Dieses implementiert eine vollständige Isolation des Linux-Kernels. In den folgenden Jahren wurden die unterliegenden Konzepte erweitert und vereinfacht. Die Popularität, die Container heute für sich haben kam mit dem Release der Container-Plattform Docker im Jahr 2013 [2]. Docker setze zu Beginn auf LXC auf, wechselte bald aber zu einer eigenen Implementierung. Zudem bietet Docker mit dem Docker Hub eine SaaS-Plattform, die die Widerverwendung von Containern ermöglicht. 

**Container-Runtimes: Aktuelle Konkurrenten zu Docker**

Neben Docker hat sich ein großes Markt an Alternativen Container-Runtimes gebildet. So bietet CoreOS mit rkt eine Alternative, die auf Sicherheit und Widerverwendbarkeit einzelner Container setzt. Zudem setzt rkt darauf, eine der ersten Container-Runtimes zu sein, die den AppContainer Standard implementiert.

Eine weitere Alternative bietet Canonical mit LXD, einer Weiterführung der Container-Runtime LXC. LXD setzt, recht ähnlich zu ersten Versionen von Docker auf eine RESTful API zur Steuerung von LXC. Im Gegensatz zu rkt oder Docker ist LXD dafür gedacht, komplette Linux-Distributionen zu isolieren.

Durch das rapide Wachstum am Interesse für Container wurden Standards für diese gefordert. Neben AppContainer wurde 2015 unter der Obhut der Linux Foundation die Open Container Initiative gegründet [3]. Diese definiert Standards für Container-Images und Runtimes, die durch die Implementierung runc genutzt wird. Mittlerweile nutzen Docker und viele weitere Tools die Spezifikation und bauen ihre Angebote auf runc um.

Sources:   
[1]: [https://www.microsoft.com/investor/reports/ar17/index.html#](https://www.microsoft.com/investor/reports/ar17/index.html)  
[2]: <https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016>  
[3]: https://www.opencontainers.org/

Images:  
(1): Virtualisierung Container und Virtuelle Machinen: <http://www.serverspace.co.uk/hubfs/Blog_Images/container-vs-vm.jpg?t=1524959203460>

(2): <https://trends.google.de/trends/explore?date=today%205-y&q=Docker>.