



Fakultät Informatik

Softwaretechnik und Medieninformatik

Bachelorthesis

Evaluierung verschiedener Container Technologien

Corvin Schapöhler

751301

Semester 2018

Firma: NovaTec GmbH

Betreuer: Dipl.-Ing. Matthias Haeussler

Erstprüfer: Prof. Dr.-Ing. Dipl.-Inform. Kai Warendorf

Zweitprüfer: Prof. Dr. Dipl.-Inform. Dominik Schoop

Ehrenwörtliche Erklärung

Hiermit versichere ich, Corvin Schapöhler, dass ich die vorliegende Bachelorarbeit mit dem Titel Evaluierung verschiedener Container Technologien selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht in gleicher oder anderer Form als Prüfungsleistung vorgelegt worden.

Stuttgart, 21. März 2018

Ort, Datum

Corvin Schapöhler

Kurzfassung

Stichwörter: *Container, Docker, Cloud Native, OCI, Linux, rkt, Evaluation*

Abstract

Keywords: *Container, Docker, Cloud Native, OCI, Linux, rkt, Evaluation*

Inhaltsverzeichnis

Ehrenwörtliche Erklärung	I
Kurzfassung	II
Abstract	II
1 Einleitung	1
1.1 Motivation	1
1.2 Aufbau der Arbeit	1
2 Grundlagen	3
2.1 Standards	4
2.1.1 App Container	4
2.1.2 Open Container Initiative	4
2.1.3 Cloud Native Computing Foundation	4
2.2 Funktionsweise	5
2.2.1 Change Root	5
2.2.2 Control Groups	5
2.2.3 Namespaces	6
2.3 Eigene Implementierung	6
Abbildungsverzeichnis	A
Tabellenverzeichnis	B
Listings	C
Akronyme	D
Glossar	E
Literatur	F

1 Einleitung

1.1 Motivation

Die Welt wird immer stärker vernetzt. Durch den Drang, Anwendungen für viele Nutzer zugänglich zu machen besteht der Bedarf an Cloud-Diensten wie Amazon Web Services (AWS). Eine dabei immer wieder auftretende Schwierigkeit ist es, die Skalierbarkeit des Services zu gewährleisten. Selbst wenn viele Nutzer gleichzeitig auf einen Service zugreifen, darf dieser nicht unter der Last zusammenbrechen.

Bis vor einigen Jahren wurde diese Skalierbarkeit durch Virtuelle Maschinen (VMs) gewährleistet. Doch neben großem Konfigurationsaufwand haben VMs auch einen großen Footprint und sind für viele Anwendungen zu ineffizient. Eine Lösung für dieses Problem stellen Container.

Diese Arbeit beschäftigt sich mit dem Thema Containering und zeigt auf, wie diese den Entwicklungszyklus für Entwickler und DevOps erleichtern.

1.2 Aufbau der Arbeit

Zu Beginn dieser Arbeit werden die Grundlagen der Containertechnologie erklärt. Dabei wird darauf eingegangen, wie Container eine vollständige Isolation des Kernels schaffen. Um die technischen Grundlagen zu verstehen, wird eine eigene Abstraktion eines Container-Prozesses geschaffen.

Dabei wird ein Prozess auf einem Linux Hostsystem vollständig isoliert und die Kapselung dieses gezeigt. Es wird erkenntlich, dass die Isolation einzelner Prozesse auf Linux durch Kernelfeatures ermöglicht wird.

Folgend werden Standards für Container-Technologien aufgezählt. Diese sind in den letzten Jahren durch den Boom der Technologie unerlässlich geworden. Dabei wird vor allem der Open Container Initiative (OCI) Standard näher beleuchtet, der sich durch die Vielzahl der kooperierenden Firmen durchsetzt.

Im Folgenden wird ein Blick auf die Geschichte der Technologie geworfen und anhand dieser erklärt, wie Docker die am weitesten verbreitete Technologie wurde. Zudem werden alternative Softwarelösungen zu Docker vorgestellt und miteinander verglichen. Dabei soll ein Fokus auf die Aspekte Konfiguration, Sicherheit und Orchestrierung der Container gelegt werden.

Zum Abschluss der Arbeit wird ein Blick in die Zukunft gewagt und Container im Zusammenhang mit Serverless Technologien gebracht.

2 Grundlagen

Container werden häufig als leichtgewichtige VMs beschrieben. Dies ist allerdings nicht ganz richtig. Wie in Abbildung 1 zu erkennen, virtualisieren Container kein vollständiges Operating System (OS), sondern lediglich das benötigte Dateisystem. Dabei wird der Kernel des Hosts nicht virtualisiert, sondern lediglich mitverwendet. Dies macht Container deutlich leichtgewichtiger als VMs, isoliert allerdings weniger umfangreich als diese.

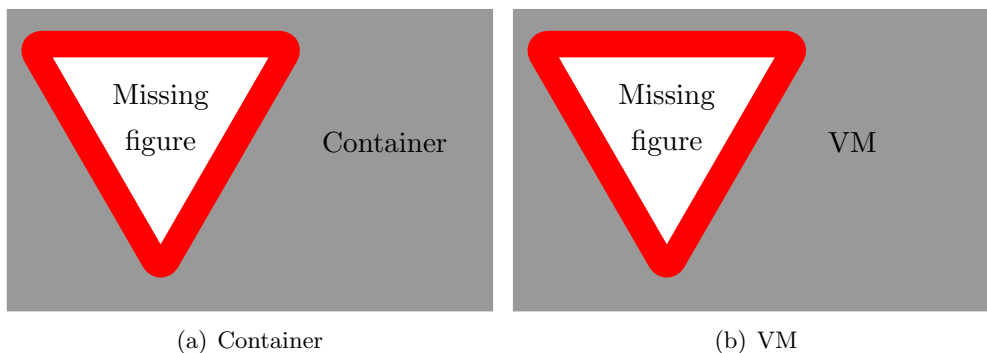


Abbildung 1: Virtualisierung durch VMs und Container

Dieses Kapitel behandelt alle benötigten Grundlagen, die für die Isolation eines Prozesses benötigt werden. Es werden vorhandene Standards wie die OCI und benötigte Systemcalls wie Change Root (chroot) näher erläutert. Zudem wird beschrieben, wie man die Isolation, die Container bieten, durch Systemmittel des Linuxkernels selber erreichen kann.

2.1 Standards

2.1.1 App Container

2.1.2 Open Container Initiative

Die OCI ist eine Initiative, die seit 2015 unter der Linux Foundation agiert. Das Ziel der OCI ist es, einen offenen Standard für Container zu schaffen, so dass die Wahl der Container-Laufzeitumgebung nicht mehr zu Inkompatibilität führt. Dabei liegt der Fokus auf eine einfache, schlanke Implementierung (Open Container Initiative, 2018).

Die OCI arbeitet aktuell an zwei Spezifikationen. Die runtime-spec standardisiert die Laufzeitumgebung von Containern. Dabei wird festgelegt, welche Konfiguration, Prinzipien und Schnittstellen Laufzeitumgebungen stellen müssen. Um die Umsetzung der runtime-spec zu fördern, stellt die OCI eine beispielhafte Implementierung durch runC.

Das zweite Projekt der OCI ist die image-spec. Dieses versucht einen Standard für Images zu definieren. Dabei plant die OCI nicht, vorhandene Image-Formate zu ersetzen, sondern auf diesen aufzubauen und sie zu erweitern (Open Container Initiative, 2018).

2.1.3 Cloud Native Computing Foundation

Die Cloud Native Computing Foundation (CNCF) beschäftigt sich im Gegensatz zur OCI nicht nur mit Containern, sondern der kompletten Cloud Native Landscape. Projekte wie Kubernetes (K8) und Prometheus werden durch die CNCF weiterentwickelt und publiziert (CNCF, 2017). Da der Cloud-Native Entwicklungsprozess von Containern getragen wird, spielen Technologien wie containerd und rkt eine entscheidende Rolle für die CNCF.

AppC einordnen
appc →
oci und cnf
für Projekte
die nicht in
oci passen

CoreOS Blog
- Wo sitzt die
CNCF zwi-
schen appC
und OCI

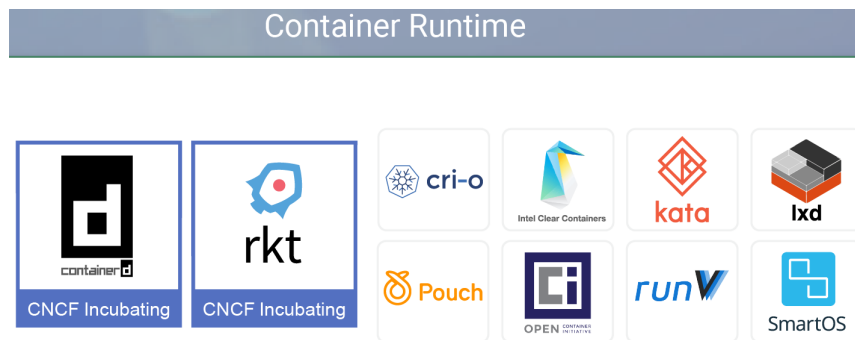


Abbildung 2: CNCF Container Landscape (CNCF, 2018)

2.2 Funktionsweise

Container isolieren einzelne Prozesse durch verschiedene Kernel Technologien, die im Folgenden erklärt werden sollen.

2.2.1 Change Root

Chroot ist ein Unix Systemaufruf, der es erlaubt einen Prozess in einem anderen Wurzelverzeichnis auszuführen (McGrath, 2017). Daraus folgt, dass der Prozess in einer eigenen Verzeichnisstruktur arbeitet und keine Änderungen am Dateisystem des Hosts machen kann. Chroot erlaubt somit die Isolierung des Dateisystems, die Container nutzen.

2.2.2 Control Groups

control groups (cgroups) dienen dazu, Systemressourcen für einzelne Prozesse zu limitieren.

beschreibe
cgroups,
siehe (Heo,
2015)

2.2.3 Namespaces

2.3 Eigene Implementierung

Um die technischen Grundlagen zu vertiefen wurde eine eigene Container-
Runtime entwickelt, die einen in Python geschriebenen Prozess isoliert und
vom Hostsystem abschottet.

RunC Er-
klärung und
ausführliche
Beschreibung
(libcontainer,
runC, contain-
erd ?, ...

Abbildungsverzeichnis

1	Virtualisierung durch VMs und Container	3
2	CNCF Container Landscape (CNCF, 2018)	5

Tabellenverzeichnis

Listings

Akronyme

AWS Amazon Web Services.

cgroup control group.

chroot Change Root.

CI Continuous Integration.

CNCF Cloud Native Computing Foundation.

K8 Kubernetes.

OCI Open Container Initiative.

OS Operating System.

VM Virtuelle Machine.

Glossar

Cloud Native .

Cloud Native
environment
erklären

DevOps DevOps (von *Development* und *Operations*) dienen der einfacheren Auslieferung von Software an Entwickler wie an den Kunden. Dabei verwenden sie Continuous Integration (CI) Tools wie Jenkins um eine automatisierte Bereitstellung zu gewährleisten.

Image Ein Container Image ist eine Datei, die spezifiziert, wie ein Container von der Laufzeitumgebung ausgeführt werden soll.

Literatur

- CHIANG, Eric, 2017. *Containers from Scratch*. Auch verfügbar unter: <https://ericchiang.github.io/post/containers-from-scratch/>.
- CNCF, 2017. *Cloud Native Computing Foundation*. Auch verfügbar unter: <https://www.cncf.io/>.
- CNCF, 2018. *CNCF Cloud Native Interactive Landscape*. Auch verfügbar unter: <https://landscape.cncf.io/>.
- CREQUY, Simone Gotti; Luca Bruno; Iago López Galeiras; Derek Gonyeo; Alban, 2018. *App Container*. Auch verfügbar unter: <https://github.com/appc>.
- DOCKER INC, 2018. *Docker security*. Auch verfügbar unter: <https://docs.docker.com/engine/security/security/>.
- HARRINGTON, Brian "Readbeard", 2015. *Building minimal Containers: Getting Weird with Containers*. Auch verfügbar unter: https://github.com/brianredbeard/minimal_containers.
- HEO, Tejun, 2015. *Control Group v2*. Auch verfügbar unter: <https://www.kernel.org/doc/Documentation/cgroup-v2.txt>.
- MATTHIAS, Karl; KANE, Sean P., 2015. *Docker: Up & Running: Shipping Reliable Containers in Production*. O'Reilly Media. ISBN 978-1-491-91757-2.
- MCGRATH, Roland, 2017. *chroot(1) - Linux Manual Page*. Auch verfügbar unter: <http://man7.org/linux/man-pages/man1/chroot.1.html>.
- OPEN CONTAINER INITIATIVE, 2018. *Open Container Initiative*. Auch verfügbar unter: <https://www.opencontainers.org/>.
- PETAZZONI, Jérôme, 2013. *Create Lightweight Containers with Buildroot*. Auch verfügbar unter: <https://blog.docker.com/2013/06/create-light-weight-docker-containers-buildroot/>.

- POLVI, Alex, 2015. *Making Sense of Container Standards and Foundations: OCI, CNCF, appc and rkt*. Auch verfügbar unter: <https://coreos.com/blog/making-sense-of-standards.html>.
- ROUMELIOTIS, Rachel, 2016. The Open Container Essentials Video Collection. In: *The Open Container Essentials Video Collection. O'Reilly Open-Source Conference*. ISBN 9781491968253. Auch verfügbar unter: <https://www.safaribooksonline.com/library/view/the-open-container/9781491968260/>.

Todo list

Figure: Container	3
Figure: VM	3
AppC einordnen appc → oci und cnf für Projekte die nicht in oci passen	4
CoreOS Blog - Wo sitzt die CNCF zwischen appC und OCI	4
beschreibe cgroups, siehe (Heo, 2015)	5
RunC Erklärung und ausführliche Beschreibung (libcontainer, runC, containerd ?,	6
Cloud Native environment erklären	E