



Fakultät Informatik

Softwaretechnik und Medieninformatik

Ausarbeitung zum Thema

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, 19. März 2018

Ort, Datum

Corvin Schapöhler

Kurzfassung

Container, Docker, Cloud Native, OCI, Linux, rkt, Evaluation

Abstract

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	3
2.1.1 Open Container Initiative	3
2.1.2 Cloud Native Computing Foundation	4
2.2 Funktionsweise	4
2.2.1 Change Root	4
2.2.2 Control Groups	4
2.3 Eigene Implementierung	4
Abbildungsverzeichnis	i
Tabellenverzeichnis	ii
Listings	iii
Glossar	iv
Akronyme	v

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

Dieses Kapitel behandelt alle Grundlagen, die für Linux Container benötigt werden. Dabei liegt der Schwerpunkt auf den bestehenden Standards, die Funktionsweise hinter Containern und der Vorgehensweise, um eigene isolierte Prozesse zu instanziiieren.

Um ein besseres Verständnis für die Funktionsweise und benötigte Technologien zu geben, wird zudem behandelt, wie man eigene Prozesse, im Beispiel eine Eureka Instanz, in einem Linux System vollständig unabhängig und voneinander isoliert ausführen kann und so die Separation erhält, die Container versprechen.

2.1 Standards

2.1.1 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 (OCI, 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 (OCI, 2018).

2.1.2 Cloud Native Computing Foundation

CNCF be-
schreiben
(containerd,
K8, ...)

2.2 Funktionsweise

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

2.2.1 Change Root

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 durch Container.

2.2.2 Control Groups

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

Tabellenverzeichnis

Listings

Glossar

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. 1

Eureka Eureka ist eine von Netflix OSS entwickelte Software zur Service Discovery. Eureka wird im Rahmen der Spring Cloud Services entwickelt und veröffentlicht. 3

Image Ein Container Image ist eine Datei, die spezifiziert, wie ein Container von einer Laufzeitumgebung ausgeführt werden soll. 3

Akronyme

AWS Amazon Web Services. 1

chroot Change Root. 4

CI Continuous Integration. iv

OCI Open Container Initiative. 1, 3

VM Virtuelle Machine. 1

Literatur

MCGRATH, Roland, 2017. *chroot(1) - Linux Manual Page*. Auch verfügbar unter: <http://man7.org/linux/man-pages/man1/chroot.1.html>.

OCI, 2018. *Open Container Initiative*. Auch verfügbar unter: <https://www.opencontainers.org/>.

Todo list

CNCF beschreiben (containerd, K8, ...)	4
RunC Erklärung und ausführliche Beschreibung (libcontainer, runC, containerd ?, ...)	4