Hochschule Esslingen
University of Applied Sciences

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 erkläre ich, dass die vorliegende Arbeit von mir selbstständig und, ohne Hilfe Dritter und ausschließlich unter Verwendung der angegebenen Quellen angefertigt wurde. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen sind, habe ich als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form, auch nicht in Teilen, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Stuttgart, 13. 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

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 zeitgleich auf einen Service zugreifen, darf dieser nicht unter der Last zusammenbrechen.

Bis vor einigen Jahren wurde diese Skalierbarkeit durch Virtuelle Machinen (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 Grundalgen 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 instanziieren.

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, sodass 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. Die runtime-spec wird mit der Beispielimplementierung der OCI runC implementiert. Dabei ruft runC Operating System (OS) Funktionen über die Binary libcontainer auf, die mittlerweile in die OCI übergegangen ist. Docker und Cloud Foundry (CF) nutzen runC als Container-Laufzeitumgebung.

Das zweite Projekt der OCI ist die image-spec. Dieses versucht einen Standard für Images zu definieren. Als Grundlage des Standards dient das Image Format, welches von Docker, Rocket (rkt) und anderen Laufzeitumgebungen genutzt wird.

Image Std und Docker Images verstehen

2.1.2 Cloud Native Computing Foundation

2.2 Funktionsweise

2.3 Eigene Implementierung

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 Continous Integration (CI) Tools wie Jenkins um eine automatisierte Bereitstellung zu gewährleitsen. 1

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

Image Container Images spezifizieren, wie ein Container ausgeführt werden soll. 3

Akronyme

AWS Amazon Web Services. 1

CF Cloud Foundry. 3

CI Continous Integration. iv

OCI Open Container Initiative. 1, 3

OS Operating System. 3

rkt Rocket. 3

VM Virtuelle Machine. 1

Literatur

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

Todo list