Hochschule
Albstadt-Sigmaringen

Albstadt-Sigmaringen University

Praktische Arbeit zur vorbereitenden Blockveranstaltung

Software-Container und Software-Development

Funktion von Software-Container und deren Einsatz in Entwicklung und Produktion

Autoren:

Maximilian Rieger
Technische Informatik
85581

Florian Lubitz
Technische Informatik
85900

Thomas Schöller
Technische Informatik
87113

Marc Bitzer
Technische Informatik
87117

Jonas Acker
Technische Informatik
85583

Inhaltsverzeichnis

1 Einleitung **1**

2 Funktionalität von Containern **4**

3 Containertechnologien **7**

4 Container in der Softwareentwicklung **11**

5 Cluster **13**

6 Risiken der Containertechnologie **14**

7 Fazit und Ausblick **15**

Abbildungsverzeichnis **16**

Tabellenverzeichnis **16**

Listings **16**

Abkürzungsverzeichnis **16**

Literaturverzeichnis **16**

A Anhang **I**

A.1 Mögliche Einsatzszenarien von Containern an der Hochschule Albstadt-Sigmaringen I

A.2 Begründung der ausgewählten Literatur II

1 Einleitung

Bis kurz vor der Jahrtausendwende führte die Virtualisierung von Servern ein Schattendasein und jeder Service wurde auf einem dedizierten Server zur Verfügung gestellt. Dabei war es keine Seltenheit, dass Server sehr gering ausgelastet waren, da der laufende Service nicht die gesamte Leistung der Hardware benötigte und der Ausfall eines nicht redundanten Servers einen Totalausfall eines Services bedeutete. Eine Beispielhafte dedizierte Serverkonstellation stellt Abbildung 1 dar.

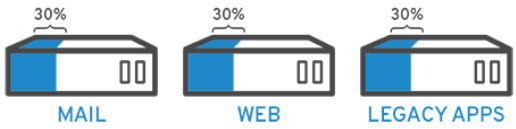

Abbildung 1: Serverauslastung ohne Virtualisierung [1]

Um diese und weitere Probleme zu lösen, gewann die Virtualisierung von Servern zum Anfang des neuen Jahrtausends immer mehr an Bedeutung und ist heutzutage ein fester Bestandteil vieler großer Unternehmen. Dabei werden auf einem physikalischen System mehrere Dienste zusammengefasst, die sonst nur einen Bruchteil der Leistung benötigen würden. Dadurch kommen noch andere Vorteile wie z.B. das Erstellen von Snapshots und das dynamische Verschieben der virtuellen Maschinen zum Tragen. Abbildung 2 zeigt die Auslastung der virtualisierten Server.

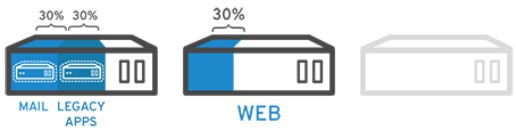

Abbildung 2: Serverauslastung mit Virtualisierung [2]

[1] Quelle: https://www.redhat.com/cms/managed-files/server-usage-500x131.png
[2] Quelle: https://www.redhat.com/cms/managed-files/server-usage-for-virtualization-500x131.png

Doch auch die Virtualisierung von Servern birgt noch Probleme, die einer Lösung bedürfen. So entsteht durch das Betriebssystem der virtuellen Maschinen ein deutlicher Overhead, da diese zur Laufzeit etliche Services benötigen. Außerdem beanspruchen die virtualisierten Betriebssysteme deutlich mehr Hardwareressourcen und die Startzeit ist relativ lang. Somit war die IT-Branche nicht in der Lage, wozu die Transportbranche längst in der Lage war: Güter in Container zu verpacken und diese Container aufgrund des standardisierten Formats auf den verschiedensten Verkehrswegen zu transportieren. Die Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V. veröffentlichte in einem Bericht (TUM) noch weitere Parallelen zwischen Software- und Transportcontainer:

Transport-Container	**Software-Container**
„Bessere Raumausnutzung“	Verzicht auf Betriebssystem im Container, Ressourcenallokation über cgroups, Overlay-FS
„Schutz gegen Beschädigungen und Diebstahl“	Virtuelle Netzwerke, wenige offene Ports, derzeit keine Signierung
„Effiziente Beladung, Transport, Entladung“	Sowohl einfache Shellkommandos wie auch komplexe Deploymenttols, zentrales Verzeichnis
„Beschleunigte Abfertigung (Sysadmin)“	Nachvollziehbarkeit von Änderungen durch Modifikationsskripte (Dockerfiles)
„Geschlossene Transportkette“	Gewisse Betriebssystemunabhängigkeit, Unterstützung durch Cloud-Provider
„Höhere Transportsicherheit“	Isolierung der Prozesse durch Linux Namespaces

Tabelle 1: Transport- und Software-Container

Um diese Lösung in die IT zu portieren, wurden auch für diese Problemstellung Container (in dem Fall für Software) entwickelt. Software-Container setzen wie die Schiffscontainer an dem Punkt Portabilität an. Es soll nicht für jeden Service ein zusätzliches Betriebssystem virtualisiert werden, sondern der Container soll nur das zusätzlich beinhalten, was er für den Service benötigt und trotzdem isoliert von den anderen Container auf der Hardware laufen. Außerdem soll es wie bei den virtuellen Maschinen möglich sein, dynamisch Ressourcen zuzuweisen.EDWARDS [2016]; REDHAT Die Abbildung 3 verdeutlicht nochmals den eingesparten Overhead bei Containern verglichen mit virtuellen Maschinen.

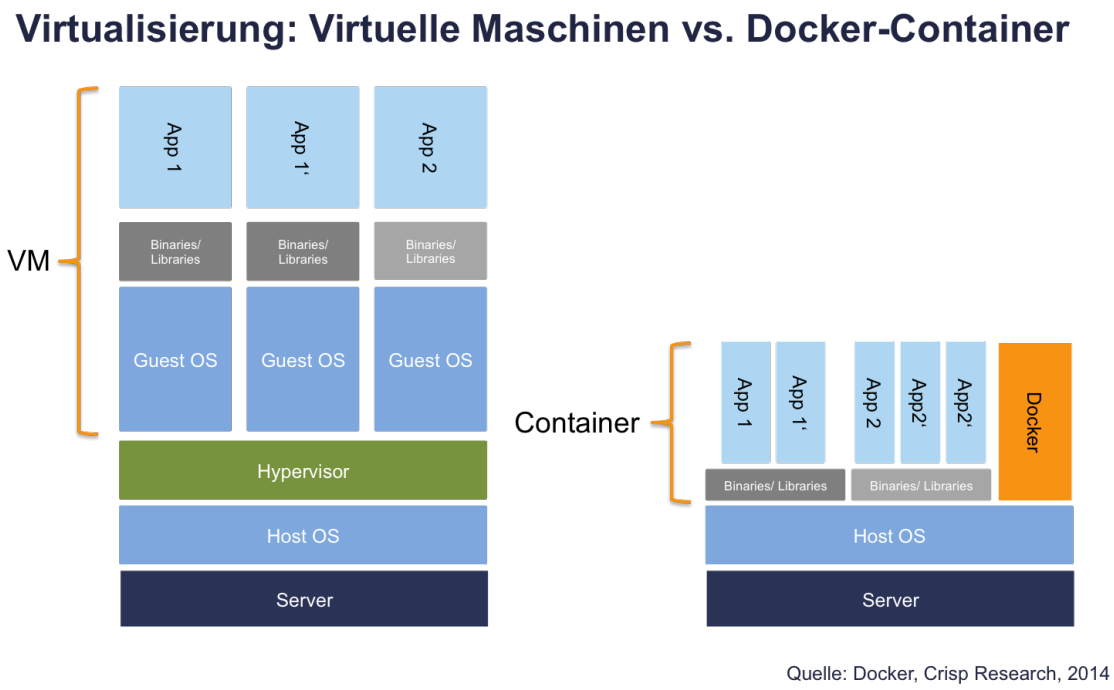

Abbildung 3: Vergleich Container und VM [3]

[3] Quelle: https://images.computerwoche.de/bdb/2668601/738x415_f5f5f5.jpg

2 Funktionalität von Containern

Container setzen direkt auf dem Kernel eines Linux-Betriebssystems auf. Um auf den Kernel durchgreifen zu können, verwenden Container standard-Linux-Techniken wie Cgroups und Namespaces oder selbst entwickelte Schnittstellen. Dadurch wird das Betriebssystem innerhalb des Containers, ohne einen Hypervisor und eine Kopie des Betriebssystems zwischen der Anwendung und der Hardware emulieren. Alles was die Anwendung zusätzlich benötigt wird mit in den Container "verpackt".ANDERSON [2015] Abbildung 4 verdeutlicht den Kernelzugriff.

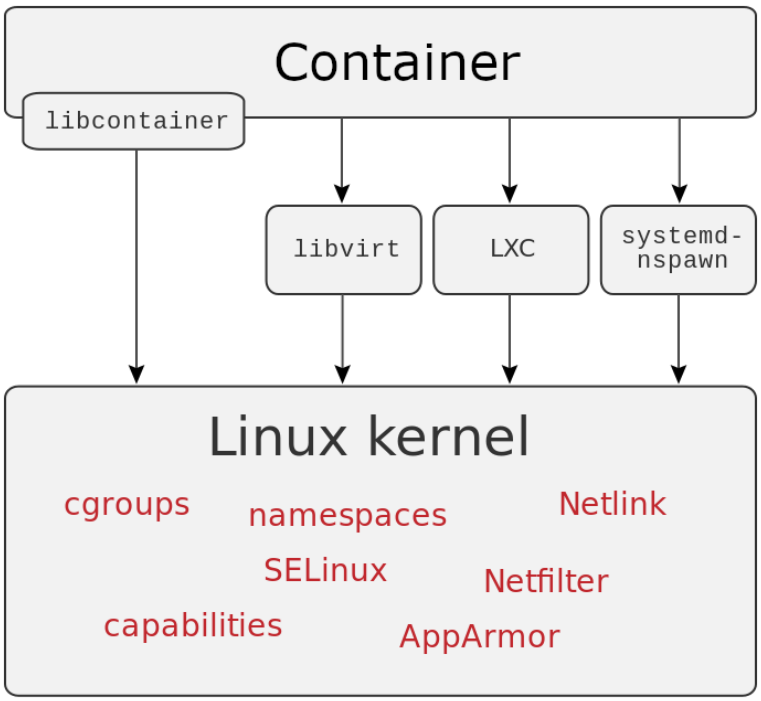

Abbildung 4: Schnittstelle vom Container zum Kernel [4]

[4] Quelle: https://www.datacenter-insider.de/container-technik-docker-co-a-480855/index2.html

Somit können CPU-Zyklen, Arbeitsspeicher, Blockspeicher und sonstige Schnittstellen über den Kernel angefordert und isoliert in dem jeweiligen Container zur Verfügung gestellt werden.Riskhan u. a. [2017]

Die Kommunikation mit Containern funktioniert mithilfe einer virtuellen Netzwerkschnittstelle für jeden Container. Außerhalb der Container können die Ports dann auf die Netzwerkkarte gemappt werden wobei auf dem Host dann natürlich jeder Port pro Netzwerkkarte nur einmal genutzt werden kann.Anderson [2015]

Da Container nur als einzelnes Image abgelegt sind und kein Betriebssystem beinhalten, welches aktualisiert und gewartet werden müsste, beschränken sich die Installation und Deinstallation auf ein einfaches Kopieren oder Löschen des Containers. Aus einem Image können beliebig viele Container-Instanzen aufgerufen werden, da Schreibzugriffe nicht auf das Image zugreifen, sondern auf ein eigenes Dateisystem des Containers. Dieses Verhalten sorgt für eine sehr hohe Skalierbarkeit, da bei Bedarf einfach neue Instanzen der Anwendung gestartet werden können.LEUNG u. a. [2018] Durch diese dynamische Skalierung und da die Container mit einem Bruchteil einer Sekunde im Vergleich zu VMs oder dedizierten Servern sehr schnell gestartet und beendet werden können, haben sie eine deutlich kürzere durchschnittliche Lebensdauer. Die genaue prozentuale Verteilung der statistischen Ausführungszeiten von Containern (Dauer zwischen Containerstart und Containerende) kann der Abbildung 5 entnommen werden:

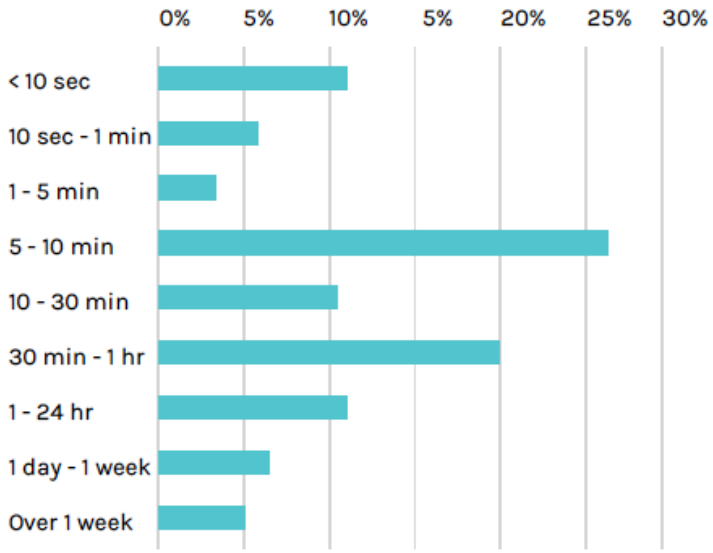

Abbildung 5: Lebensdauer eines Containers [5]

In den letzten 10 Jahren haben Container einen großen Wandel durchlebt, welcher in Abschnitt 3: Containertechnologien näher erläutert wird.

[5] Quelle: https://www.dailyhostnews.com/wp-content/uploads/2018/05/d3.png

3 Containertechnologien

In der Geschichte der Containertechnologie traten verschiedene Implementierungsformen auf. Hierbei waren die ersten Umsetzungen noch sehr einfach aufgebaut und wurden mit den Anforderungen an die Containerdienste immer komplexer. Im Folgenden findet sich eine Übersicht über die wichtigsten Technologien der Containerisierung.

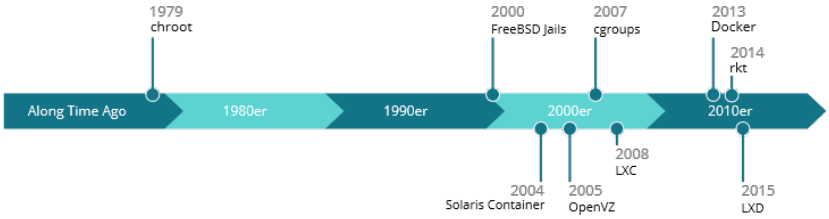

Abbildung 6: Containertechnologie im Laufe der Zeit

chroot

Chroot ist ein Befehl, der schon früh in Unix-Systemen eingebaut wurde. Er ermöglicht es einem Prozess, ein anderes Rootverzeichnis zu geben. Wird in einem Programm `chroot()` aufgerufen, wechselt es das Verzeichnis und kann nicht auf Dateien außerhalb der zugewiesenen Struktur zugreifen. Diese Abschottung eines Prozess war nie als Sicherheitsfeature vorgesehen und wird hauptsächlich zur Virtualisierung eingesetzt. Mit dem Befehl können einzelne Prozesse auf Dateiebene von anderen Anwendungen getrennt werden, weitere Sicherheitsmechanismen oder Isolierungen gibt es nicht.KAUR UND SINGH [2016]; SMITH [1996]; MANPAGES

OpenVZ

Abbildung 7: [6]

Im Jahr 2005 veröffentlichte die Firma SW-soft (später umbenannt zu Parallels) ihr Projekt OpenVZ unter der GNU GPL Lizenz. OpenVZ basierte auf der Idee der Container, ermöglicht es jedoch in jedem Container eine eigene Linux-Distribution auszuführen. Die durch die Containerumgebung abgegrenzten Betriebssysteme, teilen sich dabei einen Kernel. Dadurch ist der Overhead von OpenVZ deutlich geringer als bei der klassischen Vollvirtualisierung eines Betriebssystems. In den einzelnen Containern gibt es jeweils einen eigenen root-User und eine eigene Dateistruktur. Sie können unabhängig voneinander gestarten und gestoppt werden. Da sich die Betriebssysteme einen Kernel teilen, können auch die Gastsysteme nur Linux-Systeme sein. Da viele der Änderungen von OpenVZ den Kernel von Linux betreffen, werden regelmäßig Änderungen von OpenVZ-Patches in den Kernel von Linux übernommen.OpenVZ [b]; Ahmed u. a. [2008]; OpenVZ [a]

FreeBSD Jails

Mit der Veröffentlicheung von FreeBSD 4.0 im Jahr 2000 war FreeBSD Jails das erste richtige System in der Containervirtualisierung. Die FreeBSD Jails basieren auf dem Konzept von chroot. Auch hier wird das root-Verzeichnis eines Prozess geändert. Zusätzlich verbessert Jails das Konzept um einige Aspekte, so erhält jede Jail einen eigenen Hostnamen und eine eigene IP-Adresse. Jede Jail hat auch ihre eigenen Benutzer, inklusive einem root-Benutzer. FreeBSD Durch diese Prozessisolation ergibt sich eine Art Containersystem. Da die Jails als eigener Prozess laufen, können sie unabhängig voneinander gestartet und gestoppt werden. Jails wird gerne für den Einsatz in Netzwerkaufgaben eingesetzt, da die Performance sehr gut ist. Jails besitzt allerdings kein so großes Ökosystem wie Bespielsweise Docker oder

[6] Quelle: https://upload.wikimedia.org/wikipedia/commons/b/bb/OpenVZ-logo.png?download

OpenVZ. Daher wird es in der Containervirtualisierung von diesen Gegenspielern verdrängt.

LXC

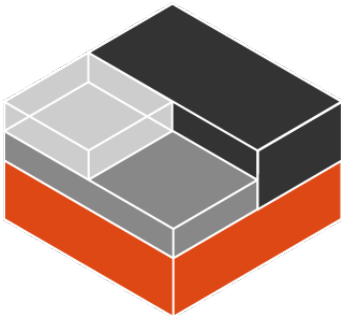

Abbildung 8: [7]

LXC ist seit der erstmaligen Veröffentlichung 2008 ein offizielles Kernelfeature und in den meisten Distributionen von Linux enthalten. Die Abkürzunhg LXC ist eine User Space-Schnittstelle für die erstellung von isolierten Umgebungen innerhalb eines Systems. Dies geschieht durch die Nutzung von Kernel namespace, Apparmor und SELinux-Profilen sowie chroots und cgroups. Diese Features standen schon vor LXC zur Verfügung, jedoch vereinigte sie LXC zu einer Schnittstelle für die Erzeugung von Containern. Zu Beginn der Entwicklung von LXC war die Isolation der Container nicht so gut, sondern glich eher einer Abwandlung der chroot-Funktion. Mit der Zeit wurde die Abschottung jedoch immer besser und die LXC-Container wurden zu richtigen virtualisierten Umgebungen. Dies geschah unter anderem dadurch, dass ab Version 1.0 die einzelnen Container als unpriviliegierte Benutzter ausgeführt werden können. Zuvor war dies nicht möglich und eine Abgrenzung der Container nur bedingt gegeben. LXC ist eine Technologie, die von vielen weiteren Projekten eingesetzt wird, unter anderen auch Proxmox oder Docker (bis Version 1.1)LXC; BERNSTEIN [2014]; BESERRA U. A. [2015]; RIZKI U. A. [2016]; UEHARA [2017]

LXD

Um die Verwendung von LXC zu vereinfachen wurde das Tool LXD entwickelt. Es besteht aus drei Elementen: Einem Deamon, der eine REST-API zur Verfü-

[7] Quelle: https://upload.wikimedia.org/wikipedia/commons/b/bb/OpenVZ-logo.png?download

gung stellt, einem Befehlszeilenclient sowie einem Open-Stack Nova Plugin. Die vom Deamon bereit gestellte Schnittstelle ermöglicht es, über das Netzwerk auf das Management der Container zuzugreifen. LXD ist somit eine Erweiterung, die eine Schnittstelle zu LXC-Containern schafft. Über das Nova Plugin können die einzelnen LXD-Maschinen als Rechenknoten verwendet werden. LXD

Solaris Container

Im Jahr 2004 veröffentlichte Oracle im Build 51 von Solaris 10 zum ersten Mal ein Feature mit dem Namen Solaris Containers. Solaris Container stellt eine Technologie dar, mit der auf x86 und SPARC-Systemen Betriebssystemlevelvirtualisierung durchgeführt werden kann. Später zusammengelegt zu Solaris Zones, bestanden die beiden Technologien Solaris Containers und Solaris Zones parallel zueinander. Dabei war Zones eine klassische Virtualisierungsplatform mit Hypervisor und Containers eine Containertechnologie, die analog zu chroot funktionierte. Mit der Zusammenlegung von Containers und Zones zum neuen Zones wurde daraus eine Containerumgebung, in der die Container sicher voneinander und dem Host getrennt sind und von einem Resourcenmanagement kontrolliert werden.ORACLE; DREWANZ UND GRIMMER

Docker

rkt

Abbildung 9: [8]

rkt (Ausprache wie "rocket") ist eine Containerengine, die sich als Alternative zu Docker etabliert und von CoreOS veröffentlicht wurde und weiterentwickelt wird. Das Projekt ist ein Open-Source-Projekt und unter der Apache License 2.0 veröffentlicht.COREOS

[8] Quelle: https://github.com/rkt/rkt/raw/master/logos/rkt-horizontal-color.png

[b] Unter rkt werden viele Grundgedanken von UNIX umgesetzt, so liegen alle Container als Dateien vor, die einfach verwaltet werden können. Auch legt rkt einen großen Wert auf Sicherheit und setzt dazu verschiedene Techniken ein, die inzwischen von den meisten Konkurennten übernommen wurden. So kann rkt für jeden Container entscheiden, ob dieser auf Basis von KVM oder einer Virtual Machine isoliert wird und führt alle Prozesse, auch den Download von Images, als nicht priviligierter Benutzer aus. In rkt wird die kleinste Einheit ein "pod"genannt. Sie kann aus einem oder mehrerm Containeren bestehen, die sich Ressourcen teilen. So passt das Konzept von rkt direkt zu den Konzepten von Cluster-Managern. Auch besitzt rkt keinen zentralen Service, der alle Container überwacht, sondern arbeitet direkt mit dem Systemeigenen systemd zusammen, um die Container zu verwalten. Somit lässt sich rkt auch direkt mit Kubernetes verknüpfen, für das der Herausgeber von rkt, CoreOS, die kommerzielle Implementierung Tectonic entwickelt. rkt unterstützt auch die Konvertierung von Docker-Containern zu rkt-Pods.CoreOS [a]; Yanar

4 Container in der Softwareentwicklung

Der Einsatz von Containern erleichtert die Entwicklung von Software in vielerlei Hinsicht. So müssen Entwickler ihre Applikationen für verschiedene Plattformen nicht grundlegend verschieden entwerfen und die Programmiersprache kann meist frei gewählt werden. Ob für Windows, Linux, MacOS, Cloud-Plattformen oder andere, der Fokus der Entwicklung kann deutlich stärker auf die Funktionalitäten der Applikation gerichtet werden, wenn die Eigenheiten der Ziel-Plattform in den Hintergrund rücken. Das macht den gesamten Entwicklungsprozess einfacher und somit effizienter. Mithilfe der Abstraktion durch Container vermeidet man Inkompatibilitätsprobleme auf den Host-Geräten und auch die Entwicklung sowie Softwaretests gestalten sich dadurch leichter, schneller und effizienter, denn alle für die Applikation wichtigen Daten, Tools und Systembibliotheken sind im Container vorhanden. Burns u. a. [2016]

Entwickler können davon ausgehen, dass ihre Applikation auf verschiedenen Systemen funktionieren wird und können sie immer unter konsistenten Bedingungen testen, egal wie später die Umgebung aussehen mag. Dies erhöht die Zuverlässigkeit enorm. Man ist nicht länger Abhängig von der Verfügbarkeit von identischen Entwicklungs- und Testsystemen und der Entwickler kann auf seinem eigenen Rechner auch schnelleres Feedback erhalten, wenn er den Container lokal ausführt und debuggt.

Auch ermöglichen Container die Verwendung von Microservices. Sonst als monolithische Applikation entworfene Software kann von Entwicklern unabhängig in mehreren Teilen erstellt werden, was die Agilität deutlich fördert. Außerdem ist das Software Deployment sehr simpel, da es nur gilt, ein Container Image zu erzeugen und zu verteilen. Dies kann auch über Container-Orchestration-Tools wie Kubernetes nach dem Prinzip von Continous Delivery automatisiert werden. Die Ausführung läuft auf jedem System dann jedes Mal gleich ab. IT AGILE

Dementsprechend benötigt man auch für Weiterentwicklung und Wartung der Applikationen weniger Zeit und Personal als wenn man für jedes System eigene Entwickler mit Fachkenntnissen bräuchte. Bei Veränderungen an der Hardware, kurzfristigem Wechsel, Neuanschaffungen aber auch bei Upgrades des Betriebssystems hat eine Firma keine größeren Schwierigkeiten durch Inkompatibilitäten zu befürchten. Somit ist sie auch freier in der Wahl ihrer Geräte.

Auch das sogenannte Monitoring, die laufende Überwachung der Systeme, über Schnittstellen (APIs) ist mit Containern kein Problem. Logs können von jeder Applikation erstellt, dann einfach gesammelt und in ein Management-System übertragen werden. Die Erkennung und Eingrenzung von Fehlerquellen beschleunigt sich dadurch, dass die Applikation im Container gekapselt ist und keine weiteren Programme oder Betriebssystemteile die Fehlersuche erschweren. Auch können die Container-Applikationen einfach mit ihrem vordefinierten Idealzustand neugestartet werden, sobald ein Problem erkannt wird. Diese Vereinfachung durch Abstraktion hilft dann nicht nur dem Entwickler, sondern trägt zur Zufriedenheit der Nutzer bei.

Besonders wenn es darum geht, neue Applikationen zu entwerfen, deren Zielplattformen noch nicht endgültig festgelegt sind, oder bei einem Umzug in die Cloud. Gerade bei Cloud-Diensten sind Container unter anderem wegen ihres geringeren Ressourcen-Umfangs beliebt. LEUNG u. a. [2018]

"Container eignen sich optimal für dienstbasierte Architekturen. Im Gegensatz zu monolithischen Architekturen, bei denen alle Teile einer Anwendung miteinander verknüpft sind [...], werden diese Komponenten bei einer dienstbasierten Architektur getrennt. Durch eine Trennung und Arbeitsteilung werden Ihre Dienste auch dann weiter ausgeführt, wenn andere fehlschlagen. Damit bleibt Ihre gesamte Anwendung zuverlässiger."Google

Tools, die sich speziell um das Ressourcen-Management kümmern, sind in vielen Containern mit inbegriffen, sodass beispielsweise der zur Verfügung stehende Speicher sinnvoll begrenzt werden kann, um Out-of-memory-Abstürzen vorzubeugen. Das schont die Server, auf denen die Applikationen laufen, und reduziert den Hardware-Bedarf und die Kosten, wenn weniger virtuelle Maschinen mit eigenem vollwertigen Betriebssystem aufgesetzt werden müssen. BURNS u. a. [2016]

5 Cluster

Wie in Abschnitt 1: Einleitung genannt, wurden in der Vergangenheit dedizierte Server für jeweils einen Prozess genutzt. Dies hatte den Nachteil einer geringen Serverauslastung sowie bei nicht reundanten Servern die Gefahr eines Totalausfalls eines Services. Applikationen ließen sich nicht ohne weiteres von einem Server auf einen anderen umziehen, da sie tief in das Hostsystem integriert waren.

Cluster Manager verbinden mehrere Maschinen zu einer Einheit. Während Lösungen wie Apache Mesos eine Abstraktion der Hardware vornehmen, basieren Kubernetes und Docker Swarm auf der Container-Architektur. Diese Cluster Manager übernehmen die Verwaltung der Container sowie ihre Zuordnung zu den jeweiligen Maschinen.

Clustering sorgt für eine verbesserte Redundanz. Außerdem lässt sich so eine bessere Ressourcen-Allokation vornehmen.

Thema aktueller Forschungsarbeiten ist die Verbesserung des Sheduling, um die Ressourcennutzung zu optimieren.Liu u. a. [2018]

6 Risiken der Containertechnologie

Die Containertechnologie erobert in den letzten Jahren mehr und mehr die Rechenzentren. Doch welche Risiken verbergen sich dahinter und wie kann man sich schützen?

Durch die hohe Anzahl an Container pro Server ist das Risiko bei einer Sicherheitslücke deutlich höher, da sich diese dann in beispielsweise 80 Containern, anstatt in vier virtuellen Maschinen oder einem dedizierten Server ausnutzen lässt.LANLine

Um sich den Aufwand für die Konfiguration der Images zu sparen (diese kann sehr aufwendig sein), verwenden viele Administratoren vorgefertigte Container aus einem Respository. Dabei muss dem Ersteller vertraut werden, dass das Image keinen Schadcode oder Hintertüren enthält, da der Aufwand für eine genaue Prüfung des Container-Inhalts sehr aufwendig wäre. Im Juni 2018 hatte die Sicherheitsfirma Kromtech berichtet, dass über das Repository Docker Hub mehrere Images über ein Jahr lang verfügbar waren, die Schadcode zum Minen von Kryptowährungen enthielten. Diese wurden insgesamt fünf Millionen mal installiert, bevor die Betreiber von Docker Hub reagierten und diese entfernten. Kromtech Die betroffenen Administratoren hätten das Risiko minimieren können, indem sie nur über das offizielle Docker Repository die Container bezogen hätten. Dort werden Images vor ihrer Veröffentlichung geprüft. Docker Ein Angreifer müsste zur Verteilung eines infizierten Images den Schadcode verstecken, sodass er bei der Prüfung nicht sichtbar wird. Dies stellt eine wesentlich höhere Hürde dar.

Werden Applikationen in Containern richtig verpackt, so sind die einzigen Abhängigkeiten nach außen hin die Systemaufrufe des Betriebssystems. Dies verbessert die Portabilität der Anwendungen ungemein, allerdings sind auch Systemaufrufe wie z.B. Socket-Schnittstellen sowie hardwarespezifische Systemaufrufe nicht auf allen Systemen einheitlich, wodurch die Portabilität eingeschränkt wird. Die Open Container Initiative der Linux Foundation arbeitet neben einem Standard für Container Formate auch an einem Standard für Container Runtimes. Dieser könnte helfen, die Schnittstelle zwischen Container und Betriebssystem besser festzulegen.

Container können nicht gegen Einflüsse schützen, die nicht vom Betriebssystem verwaltet werden. Hierzu sind virtuelle Maschinen als zusätzliche Sicherheitsschicht notwendig.BURNS u. a. [2016]

Nicht zuletzt haben die Sicherheitslücken Meltdown (Lipp u. a. [2018]) und Spectre (Kocher u. a. [2018]) gezeigt, dass über Sicherheitslücken in Prozessoren containerübergreifende Angriffe auf Applikationen möglich sind. Hiergegen schützten virtuelle Maschinen allerdings ebenfalls nicht.

7 Fazit und Ausblick

Abbildungsverzeichnis

1 Serverauslastung ohne Virtualisierung 1
2 Serverauslastung mit Virtualisierung 1
3 Vergleich Container und VM 3
4 Schnittstelle vom Container zum Kernel 4
5 Lebensdauer eines Containers 6
6 Containertechnologie im Laufe der Zeit 7
7 Logo OpenVZ 8
8 Logo LXC 9
9 Logo LXC 10

Tabellenverzeichnis

1 Transport- und Software-Container 2

Listings

Abkürzungsverzeichnis

Literaturverzeichnis

it agile
AGILE it: *Container im Agilen Entwicklungsprozess*.

https://www.it-agile.de/fileadmin/docs/Whitepaper_ContainerImAgilenEntwicklungsprozess_it-agile.pdf, Abruf: 25.07.2018

Ahmed u. a. 2008

AHMED, M. ; ZAHDA, S. ; ABBAS, M.: Server consolidation using OpenVZ: Performance evaluation. In: *2008 11th International Conference on Computer and Information Technology*, 2008, S. 341–346

Anderson 2015

ANDERSON, Charles: Docker. In: *IEEE Software* 32 (2015), Nr. 3, 102 - c3. http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3degs%26AN%3d102288020%26site%3dehost-live. – ISSN 07407459

Bernstein 2014

BERNSTEIN, D.: Containers and Cloud: From LXC to Docker to Kubernetes. In: *IEEE Cloud Computing* 1 (2014), Sept, Nr. 3, S. 81–84. http://dx.doi.org/10.1109/MCC.2014.51. – DOI 10.1109/MCC.2014.51. – ISSN 2325–6095

Beserra u. a. 2015

BESERRA, D. ; MORENO, E. D. ; ENDO, P. T. ; BARRETO, J. ; SADOK, D. ; FERNANDES, S.: Performance Analysis of LXC for HPC Environments. In: *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, 2015, S. 358–363

BURNS u. a. 2016

BURNS, BRENDAN ; GRANT, BRIAN ; OPPENHEIMER, DAVID ; BREWER, ERIC ; WILKES, JOHN: Borg, Omega, and Kubernetes. In: *Communications of the ACM* 59 (2016), Nr. 5, 50 - 57. http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3degs%26AN%3d115178361%26site%3dehost-live. – ISSN 00010782

CoreOS a

COREOS: *Homepage rkt*. https://coreos.com/rkt/, Abruf: 25.07.2018

CoreOS b

COREOS: *Projektrepository rkt*. https://github.com/rkt/rkt, Abruf: 25.07.2018

Docker

DOCKER: *Official repositories on Docker Hub*. https://docs.docker.com/docker-hub/official_repos/, Abruf: 25.07.2018

Drewanz und Grimmer

DREWANZ, Detlef ; GRIMMER, Lenz: *The Role of Oracle Solaris Zones and Linux Containers in a Virtualization Strategy*. http://www.oracle.com/technetwork/articles/servers-storage-admin/zones-containers-virtualization-1880908.html, Abruf: 25.07.2018

Edwards 2016

EDWARDS, Chris: Containers Push Toward the Mayfly Server. In: *Communications of the ACM* 59 (2016), Nr. 12, 24 - 26. http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3degs%26AN%3d120050683%26site%3dehost-live. – ISSN 00010782

FreeBSD

FREEBSD: *FreeBSD Handbuch*. FreeBSD. – 277 – 293 S. https://download.freebsd.org/ftp/doc/de/books/handbook/book.pdf, Abruf: 25.07.2018

Google

GOOGLE: *Container bei Google*. https://cloud.google.com/containers/, Abruf: 25.07.2018

Kaur und Singh 2016

KAUR, N. ; SINGH, M.: Improved file system security through restrictive access. In: *2016 International Conference on Inventive Computation Technologies (ICICT)* Bd. 3, 2016, S. 1–5

Kocher u. a. 2018

KOCHER, Paul ; GENKIN, Daniel ; GRUSS, Daniel ; HAAS, Werner ; HAMBURG,

Mike ; LIPP, Moritz ; MANGARD, Stefan ; PRESCHER, Thomas ; SCHWARZ, Michael ; YAROM, Yuval: Spectre Attacks: Exploiting Speculative Execution. In: *CoRR* abs/1801.01203 (2018). http://arxiv.org/abs/1801.01203

Kromtech

KROMTECH: *Cryptojacking invades cloud. How modern containerization trend is exploited by attackers*. https://kromtech.com/blog/security-center/cryptojacking-invades-cloud-how-modern-containerization-trend-is-exploited-by-a Abruf: 25.07.2018

LANLine

LANLINE: *Container sicher nutzen*. https://www.lanline.de/container-sicher-nutzen/, Abruf: 24.07.2018

LEUNG u. a. 2018

LEUNG, ANDREW ; SPYKER, ANDREW ; BOZARTH, TIM: Titus: Introducing Containers to the Netflix Cloud. In: *Communications of the ACM* 61 (2018), Nr. 2, 38 - 45. http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3degs%26AN%3d127712851%26site%3dehost-live. – ISSN 00010782

Lipp u. a. 2018

LIPP, Moritz ; SCHWARZ, Michael ; GRUSS, Daniel ; PRESCHER, Thomas ; HAAS, Werner ; MANGARD, Stefan ; KOCHER, Paul ; GENKIN, Daniel ; YAROM, Yuval ; HAMBURG, Mike: Meltdown. In: *CoRR* abs/1801.01207 (2018). http://arxiv.org/abs/1801.01207

Liu u. a. 2018

LIU, Bo ; LI, Pengfei ; LIN, Weiwei ; SHU, Na ; LI, Yin ; CHANG, Victor: A new container scheduling algorithm based on multi-objective optimization. In: *Soft Computing* (2018), Jul. http://dx.doi.org/10.1007/s00500-018-3403-7. – DOI 10.1007/s00500–018–3403–7. – ISSN 1433–7479

LXC

LXC, Offizelle H.: *Linux Containers - LXC*. https://linuxcontainers.org/lxc/, Abruf: 25.07.2018

LXD

LXD, Offizelle H.: *Linux Containers - LXD*. https://linuxcontainers.org/lxd/, Abruf: 25.07.2018

Manpages

MANPAGES, Linux: *chroot - Wurzelverzeichnis wechseln*. https://manpages.debian.org/stretch/manpages-de-dev/chroot.2.de.html, Abruf: 24.07.2018

OpenVZ a

OPENVZ, Homepage: *History*. https://wiki.openvz.org/History, Abruf: 24.07.2018

OpenVZ b

OPENVZ, Homepage: *News*. https://wiki.openvz.org/News, Abruf: 24.07.2018

Oracle

ORACLE: *Oracle Solaris Zones Introduction*. https://docs.oracle.com/cd/E36784_01/html/E36848/zones.intro-1.html#scrolltoc, Abruf: 25.07.2018

redhat

REDHAT: *Was ist Virtualisierung?* https://www.redhat.com/de/topics/virtualization/what-is-virtualization, Abruf: 24.07.2018

Riskhan u. a. 2017

RISKHAN, Basheer ; KE, Zhou ; MUHAMMAD, Raza: Energy Management of the System: An Empirical Investigation of Virtualization Approaches in Static and Dynamic Modes. In: *Information Technology Journal* 16 (2017), Nr. 1, 1 - 10. http://www.redi-bw.de/db/ebsco.php/search.ebscohost.com/login.aspx%3fdirect%3dtrue%26db%3degs%26AN%3d120592540%26site%3dehost-live. – ISSN 18125638

Rizki u. a. 2016

RIZKI, R. ; RAKHMATSYAH, A. ; NUGROHO, M. A.: Performance analysis of container-based hadoop cluster: OpenVZ and LXC. In: *2016 4th International Conference on Information and Communication Technology (ICoICT)*, 2016, S. 1–4

Smith 1996

SMITH, R. E.: Mandatory protection for Internet server software. In: *Proceedings 12th Annual Computer Security Applications Conference*, 1996. – ISSN 1063–9527, S. 178–184

TUM

TUM: *Grundlagen Container-Virtualisierung*. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwjR_Lq79LncAhVQjqQKHbGqCVsQFgg6MAE&url=https%3A%2F%2Fwww.tmf-ev.de%2FDesktopModules%2FBring2mind%2FDMX%2FDownload.aspx%3FMethod%3Dattachment%26Command%3DCore_Download%26EntryId%3D28917%26PortalId%3D0&usg=AOvVaw26Jn9693iPmp8xCk4w9e9g, Abruf: 25.07.2018

Uehara 2017

UEHARA, M.: Performance Evaluations of LXC Based Educational Cloud in a Bare Metal Server. In: *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2017, S. 415–420

Yanar

YANAR, Erkan: Im Steigflug. , 07, 94. https://www.heise.de/ix/heft/Im-Steigflug-3754458.html, Abruf: 25.07.2018

A Anhang

A.1 Mögliche Einsatzszenarien von Containern an der Hochschule Albstadt-Sigmaringen

Im folgenden soll betrachtet werden, welche Einsatzmöglichkeiten sich für Container in der IT-Infrastruktur der Hochschule Albstadt-Sigmaringen anbieten und welche Vorteile dies mit sich bringen würde.

Prädestiniert für den Einsatz sind die Server-Dienste der Hochschule. Hier wäre denkbar den in Abschnitt 5: Cluster vorgestellten Ansatz eines Serververbunds in Verbindung mit einem Cluster Manager wie Docker Swarm oder Kubernetes zu verwenden. Die einzelnen Applikationen wie z.B. das E-Learning System Ilias, die CentOS Instanzen, die Bibliotheksdienste und die Website könnten dann in mehreren Container-Instanzen laufen. Der Cluster-Manager würde dynamisch die Last auf die verschiedenen Maschinen verteilen und könnte Instanzen mit Programmfehlern erkennen sowie schnell neustarten.

Außerdem wäre es möglich, Lastspitzen abzufangen, indem Serverkapazitäten anderer Bildungseinrichtungen genutzt werden, um dort bei Bedarf Containerinstanzen zu starten.

Ein konkreter Anwendungsfall für die Hochschue Albstadt-Sigmaringen ist die Containerisierung der Oracle Datenbank. Die Datenbank wird im Rahmen mehreren Vorlesungen und Praktikas von Professoren und Studenten genutzt. Dabei ist der Zugriff auf die Datenbank, außerhalb von Vorlesungen oder Praktika, sehr gering. Während Praktika und Vorlesungen wird die Datenbank rege genutzt, wodurch die Performance spürbar leidet. Diesen starken Unterschied der benötigten Leistung könnte perfekt durch Containerisierung der Datenbank ausgeglichen werden, da je nach Nachfrage innerhalb von Sekunden Container hoch- oder runter-gefahren werden können. Diese Vorgehensweise spart nicht nur Strom, sondern garantiert auch Professoren und Studierenden eine gute Performance der Datenbank. Oracle bietet im Docker Store ein vorgefertigtes Docker Image an, was die Nutzung von Oracle auf

Docker sehr vereinfacht. Eine detaillierte Anleitung für die Containerisierung von Oracle findet sich im Internet. Um die Container dynamisch an die Last anzupassen, könnte ein in Abschnitt 5: Cluster vorgestelltes Cluster-Manager verwendet werden.

A.2 Begründung der ausgewählten Literatur

Zur Verfügung stand lediglich sehr aktuelle Literatur, da die Containervirtualisierung erst seit Erscheinen von Docker im Jahr 2013 in der IT-Branche an Bedeutung gewonnen hat. Daher sind auch viele der hier betrachteten Werkzeuge erst in den vergangen Jahren entwickelt worden.

In Anbetracht des kurzen Zeitraumes, der den Autoren zur Verfügung stand, konnte keine Fernleihe durchgeführt werden. Eine Vorbestellung der Literatur war daher ebenfalls nicht möglich. Am ersten Tag der Bearbeitung des Artikels stand außerdem die Bibliothek aufgrund des Betriebsausflugs nicht zur Verfügung, weshalb auch nicht auf die physischen Medien zurückgegriffen werden konnte. Deshalb hat sich die Bücher- bzw. Artikelauswahl auf die über die Hochschule verfügbaren digitalen Medien beschränkt.

Die Literaturauswahl umfasst außerdem Dokumentationen der gängisten Software zum Thema Container-Technologie. Diese wurde zum Verständnis des Aufbaus und der Nutzung des jeweiligen Werkzeugs genutzt. Die Dokumentationen sind online bzw. zusammen mit dem jeweiligen Source Code verfügbar und werden von den Entwicklern zur Verfügung gestellt. Daher handelt sich bei den Dokumentationen um eine verlässliche Quelle über das jeweilige Werkzeug.

Auf den offizellen Webseiten der verschiedenen Hersteller und Projekten werden von den Entwicklern oder Firmen offizelle Informationen publiziert oder auch oben genannte Dokumentationen veröffentlicht. Der Inhalt der Webseiten kann als verlässliche Quelle angesehen werden, da hier der Ersteller des Produkts direkt veröffentlicht.

Blog Einträge dienten den Autoren als Ideengeber für einen Teil des Inhalts der vorliegenden Arbeit. Da diese am Puls der Zeit sind, zeigen sie aktuelle Trends und populäre Software zum Thema Container-Technologie auf. Ein Blog wird nicht überprüft und stellt daher selbstverständlich keine zuverlässige Quelle dar. Zur weiteren Recherche wurden aufgrund dessen wissenschaftlich verlässliche Quellen verwendet.