



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	Florian Lubitz
Technische Informatik	Technische Informatik
85581	85900

Thomas Schöller	Marc Bitzer
Technische Informatik	Technische Informatik
87113	87117

Jonas Acker  
Technische Informatik  
85583

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Funktionalität von Containern</b>	<b>5</b>
<b>3</b>	<b>Containertechnologien</b>	<b>7</b>
3.1	chroot . . . . .	7
3.2	OpenVZ . . . . .	8
3.3	FreeBSD jails . . . . .	8
3.4	LXC . . . . .	8
3.5	LXD . . . . .	9
3.6	Solaris Container . . . . .	9
3.7	Windows Containers . . . . .	9
3.8	Docker . . . . .	9
3.9	Mesos . . . . .	9
3.10	rkt . . . . .	9
<b>4</b>	<b>Container und Softwareentwicklung</b>	<b>9</b>
<b>5</b>	<b>Cluster</b>	<b>11</b>
<b>6</b>	<b>Risiken der Containertechnologie</b>	<b>12</b>
<b>7</b>	<b>Fazit und Ausblick</b>	<b>12</b>
	<b>Abbildungsverzeichnis</b>	<b>13</b>
	<b>Tabellenverzeichnis</b>	<b>13</b>
	<b>Listings</b>	<b>13</b>
	<b>Abkürzungsverzeichnis</b>	<b>13</b>
	<b>Literaturverzeichnis</b>	<b>13</b>
<b>A</b>	<b>Anhang</b>	<b>I</b>
A.1	Begründung der ausgewählten Literatur . . . . .	I

# 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 Grafik 6 dar.

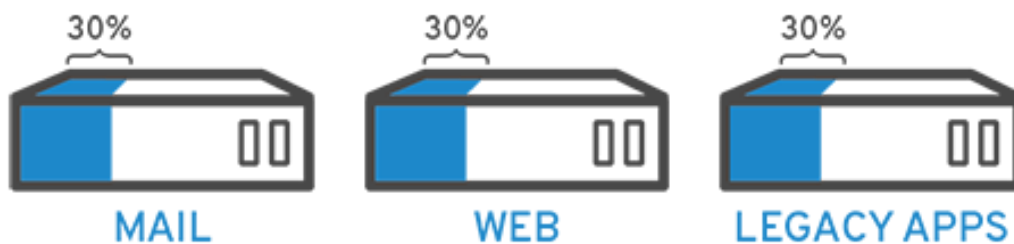


Abbildung 1: Serverauslastung ohne Virtualisierung <sup>1</sup>

<sup>1</sup>Quelle: <https://www.redhat.com/cms/managed-files/server-usage-500x131.png>

## 1 Einleitung

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. Grafik 2 zeigt die Auslastung der virtualisierten Server.

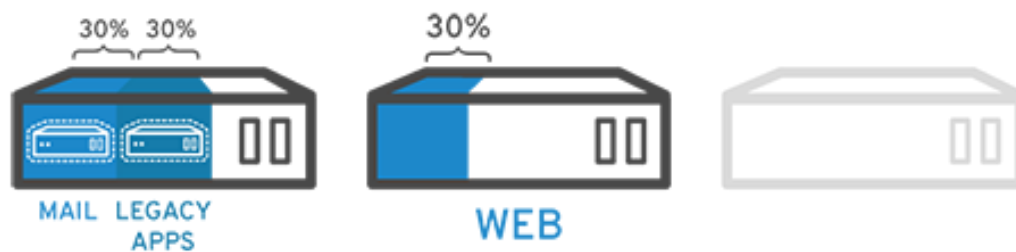


Abbildung 2: Serverauslastung mit Virtualisierung <sup>2</sup>

<sup>2</sup>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.



Abbildung 3: Container<sup>3</sup>

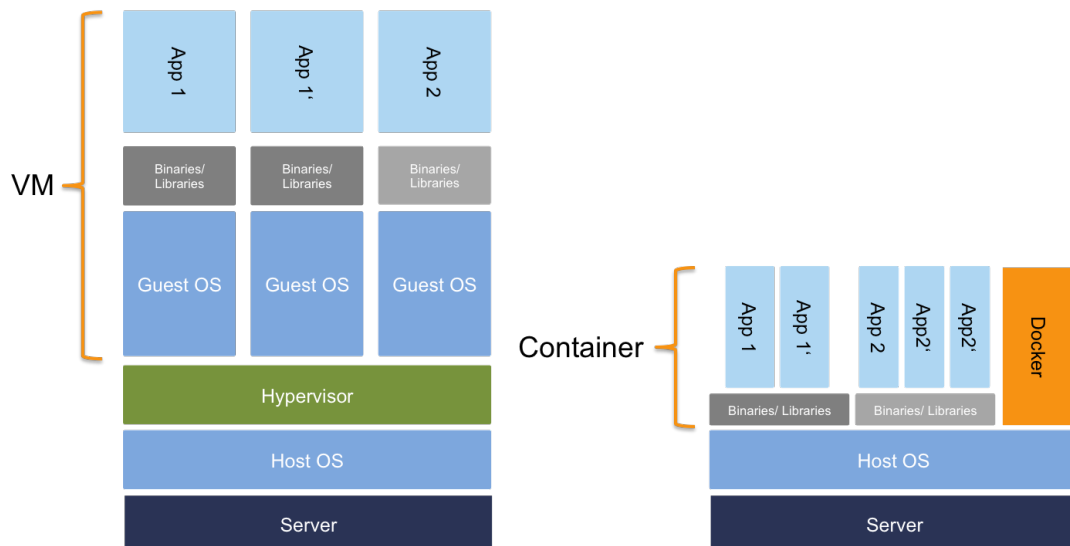
Um diese Lösung in die IT zu portieren, wurden auch für diese Problemstellung Container (in dem Fall für Software) entwickelt. Software-Container setzen wie die Schiffscontainer an dem Punkt Portabilität an. Es soll nicht für jeden Service ein zusätzliches Betriebssystem virtualisiert werden, sondern 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

---

<sup>3</sup>Quelle: [http://tricon-terminal.de/uploads/pics/\\_MG\\_8745\\_06.jpg](http://tricon-terminal.de/uploads/pics/_MG_8745_06.jpg)

Die Grafik 4 verdeutlicht nochmals den eingesparten Overhead bei Containern verglichen mit virtuellen Maschinen.

### Virtualisierung: Virtuelle Maschinen vs. Docker-Container



Quelle: Docker, Crisp Research, 2014

Abbildung 4: Vergleich Container und VM <sup>4</sup>

<sup>3</sup> Quelle: [https://images.computerwoche.de/bdb/2668601/738x415\\_f5f5f5.jpg](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 vollständige Virtualisierung, emuliert. Grafik 6 verdeutlicht den Kernelzugriff.

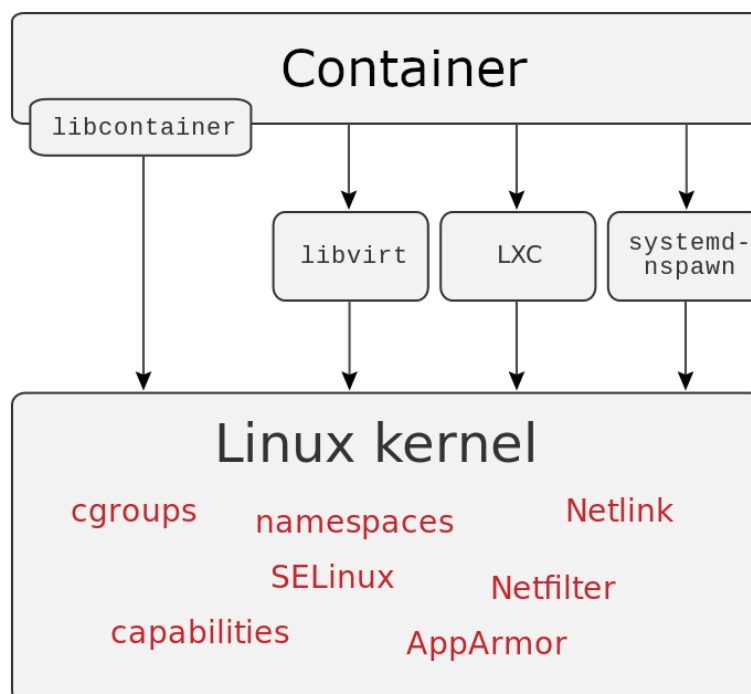


Abbildung 5: Schnittstelle vom Container zum Kernel <sup>5</sup>

Somit können CPU-Zyklen, Arbeitsspeicher, Blockspeicher und sonstige Schnittstellen über den Kernel angefordert und isoliert in dem jeweiligen Container zur Verfügung gestellt werden. [DATACENTERINSIDER](https://www.datacenter-insider.de/container-technik-docker-co-a-480855/index2.html)

<sup>4</sup> Quelle: <https://www.datacenter-insider.de/container-technik-docker-co-a-480855/index2.html>

## 2 Funktionalität von Containern

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.<sup>5</sup> Durch diese dynamische Skalierung haben Container eine kurze durchschnittliche Lebensdauer. Die genaue prozentuale Verteilung der statistischen Ausführungszeiten von Containern (Dauer zwischen Containerstart und Containerende) kann der Grafik 6 entnommen werden:

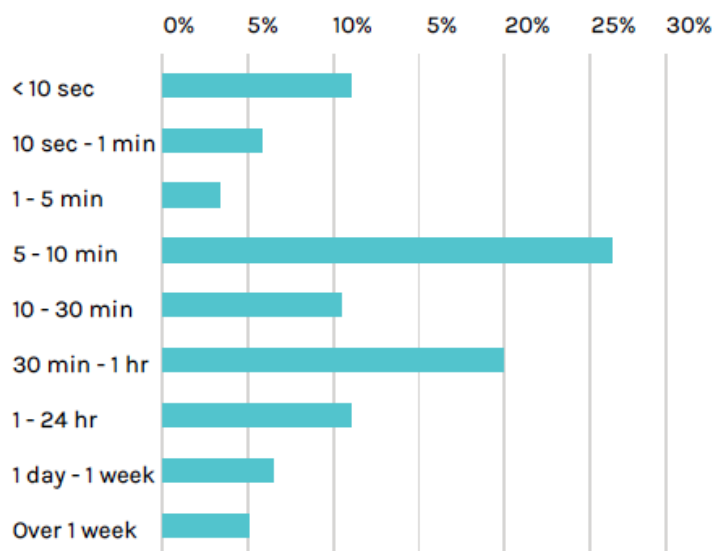


Abbildung 6: Lebensdauer eines Containers<sup>6</sup>

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

<sup>5</sup> 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.

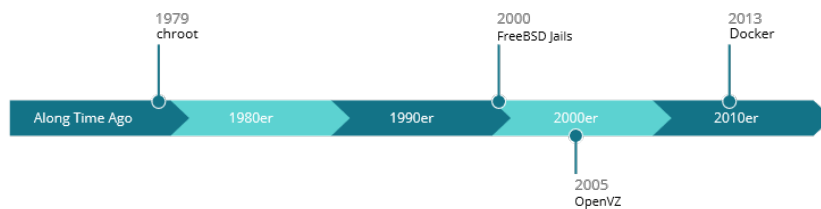


Abbildung 7: Containertechnologie im Laufe der Zeit

#### 3.1 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](#)

## 3.2 OpenVZ

Im Jahr 2005 veröffentlichte die Firma SWsoft (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 gestartet und gestoppt werden. Da sich die Betriebssysteme einen Kernel teilen, können auch die Gastssysteme 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 \[a\]](#)

## 3.3 FreeBSD jails

## 3.4 LXC

LXC ist seit der erstmaligen Veröffentlichung 2008 ein offizielles Kernelfeature und in den meisten Distributionen von Linux enthalten. Die Abkürzung 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 unprivilegierte Benutzer 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 \[2017\]](#)

### **3.5 LXD**

Um die Verwendung von LXC zu vereinfachen wurde das Tool LXD entwickelt. Es besteht aus drei Elementen: Einem Daemon, der eine REST-API zur Verfügung stellt, einem Befehlszeilenclient sowie einem Open-Stack Nova Plugin. Die vom Daemon 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](#)

### **3.6 Solaris Container**

### **3.7 Docker**

### **3.8 Mesos**

### **3.9 rkt**

## **4 Container und 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. Ob für Windows, Linux,

#### 4 Container und Softwareentwicklung

---

MacOS, Cloud-Plattformen oder andere, der Fokus der Entwicklung kann deutlich stärker auf die 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. Entwickler können auf verschiedenen Systemen arbeiten und unter den selben Bedingungen testen, was die Zuverlässigkeit erhöht. Auch ermöglichen Container die Verwendung von Microservices. Dadurch kann sonst als monolithische Applikation entworfene Software von Entwicklern unabhängig in mehreren Teilen erstellt werden. Außerdem ist das Software Deployment sehr simpel, da es nur gilt, ein Container Image zu erzeugen und zu verteilen. Die Ausführung läuft auf jedem System dann jedes Mal gleich ab.

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 neugestartet werden, sobald ein Problem erkannt wird. Diese Vereinfachung durch Abstraktion hilft dann nicht nur dem Entwickler, sondern trägt zur Zufriedenheit der Nutzer bei.

Seit dem Bekanntwerden von Docker im Jahr 2013 haben viele bedeutende Firmen wie Google, IBM und Netflix die Virtualisierung mit Containern ein-

geführt und für sich genutzt. Besonders wenn es darum geht, neue Applikationen zu entwerfen, deren Zielplattformen noch nicht endgültig festgelegt sind, oder bei einem Umzug in die Cloud sind Container ideal. Insbesondere bei Cloud-Diensten sind sie aufgrund ihres geringeren Ressourcen-Umfangs beliebt. 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.

## 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 redundanten 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 Scheduling, um die Ressourcennutzung zu optimieren.?

## 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. Da die grundlegende Konfiguration sehr aufwändig ist, verwenden viele Unternehmen vorgefertigte Container aus einem Repository. Dabei kann man sich dann nicht sicher sein, was der Container genau beinhaltet und ob eine schadhafte Absicht dahinter steckt. [LANLINE](#)

## 7 Fazit und Ausblick

## Abbildungsverzeichnis

1	Serverauslastung ohne Virtualisierung . . . . .	1
2	Serverauslastung mit Virtualisierung . . . . .	2
3	Container . . . . .	3
4	Vergleich Container und VM . . . . .	4
5	Schnittstelle vom Container zum Kernel . . . . .	5
6	Lebensdauer eines Containers . . . . .	6
7	Containertechnologie im Laufe der Zeit . . . . .	7

## Tabellenverzeichnis

## Listings

## Abkürzungsverzeichnis

## Literaturverzeichnis

### 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

**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

**DataCenterInsider**

DATACENTERINSIDER: *Container-Technik: Docker und Co.* <https://www.datacenter-insider.de/container-technik-docker-co-a-480855/>, Abruf: 24.07.2018

**DevInsider**

DEVINSIDER: *Was sind Docker-Container?* <https://www.dev-insider.de/was-sind-docker-container-a-597762/>, Abruf: 24.07.2018

**Edwards 2016**

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

**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

**LANLine**

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



**LXC**

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

**LXD**

LXD, Offizielle 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

**redhat**

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

**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

**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

## A Anhang

### A.1 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 vergangenen 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 Literatursauswahl umfasst außerdem Dokumentationen der gängigsten 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 offiziellen Webseiten der verschiedenen Hersteller und Projekten werden von den Entwicklern oder Firmen offizielle 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.