

Docker Handbuch für Einsteiger

Der leichte Weg zum Docker-Experten

Hans-M. Hopp

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Informationen sind im Internet über <http://dnb.d-nb.de> abrufbar.

©2021 BMU Media GmbH
www.bmu-verlag.de
info@bmu-verlag.de

Lektorat: Dr. phil. Annette Schönberg-Al Meklef
Einbandgestaltung: Pro ebookcovers Angie
Druck und Bindung: WirMachenDruck

Taschenbuch-ISBN: 978-3-96645-067-6
Hardcover-ISBN: 978-3-96645-068-3
E-Book-ISBN: 978-3-96645-066-9

Dieses Werk ist urheberrechtlich geschützt.
Alle Rechte (Übersetzung, Nachdruck und Vervielfältigung) vorbehalten. Kein Teil des Werks darf ohne schriftliche Genehmigung des Verlags in irgendeiner Form – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert, verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit größter Sorgfalt erstellt, ungeachtet dessen können weder Verlag noch Autor, Herausgeber oder Übersetzer für mögliche Fehler und deren Folgen eine juristische Verantwortung oder irgendeine Haftung übernehmen. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären.

Docker Handbuch für Einsteiger

Inhaltsverzeichnis

1. Einleitung	11
1.1 Vorwort	11
1.2 Die Microservice Revolution	12
1.3 Das Ziel dieses Buches	14
1.4 Konventionen im Buch	16
1.5 Warum braucht man Docker?	18
1.6 Was muss ich mir unter Docker vorstellen?	18
1.7 Was ist Docker nicht?	19
1.8 Entwicklungsgeschichte	20
2. Docker-Begriffe	24
2.1 Was ist ein Container?	24
2.2 Was ist ein Container Image?	24
2.3 Das Dockerfile	25
2.3.1 Dockerfile-Elemente	25
2.4 Was ist die Docker Engine?	27
2.5 Wer ist der Container Host?	27
2.6 Was sind Container-Netzwerke?	27
2.7 Was ist die Container Registry?	28
2.8 Was ist der Docker Hub?	28
2.9 Was ist der Unterschied zwischen Containern und Virtuellen Maschinen?	29
3. Vorbereitung	31
3.1.1 Docker Desktop Installation	31
3.1.2 Docker Desktop für Windows Installieren	31
3.1.2.1 Systemvoraussetzungen	31
3.1.2.2 Download des Installationsprogramms	31
3.1.2.3 Installation von Docker	35
3.1.3 Andere Betriebssysteme	40
3.2 Erste Versuche mit Docker	40
3.2.1 Docker Desktop starten	40
3.2.2 Docker Container starten	43
3.2.3 Beispiel-Image ‚Hello-world‘	43
4. Docker-Grundlagen	47
4.1 Docker Hub nach Images durchsuchen	47
4.2 Die Version eines Docker Images bestimmen	51
4.3 Übungsaufgabe: Container für eine ältere Image-Version bauen	55

4.4	Häufig verwendete Docker Images	56
4.4.1	Couchbase	56
4.4.2	Arangodb	57
4.4.3	Apache http Server	57
4.4.4	CentOS	58
4.4.5	Elasticsearch	58
4.4.6	Fedora	58
4.4.7	Jenkins	59
4.4.8	Joomla.....	59
4.4.9	MariaDB.....	59
4.4.10	MongoDB	60
4.4.11	MySQL.....	60
4.4.12	Neo4J.....	61
4.4.13	Nginx	61
4.4.14	Node	62
4.4.15	PostgresSQL.....	62
4.4.16	Ruby	63
4.4.17	SonarQube.....	63
4.4.18	Tomcat	64
4.4.19	Ubuntu.....	64
4.4.20	WordPress	65
4.5	Ein „Hello Docker“ Image selbst gebaut	65
4.5.1	Ausführen und Test des „Ubuntu“ Images	66
4.5.2	Ein erstes einfaches abgeleitetes Image.....	67
4.5.3	Erweiterung unseres Images	69
4.5.4	Übungsaufgabe: Funktionalität des Images erweitern	71
4.6	Veröffentlichung des neuen Images in Docker Hub	74
4.7	Docker Container im „detached“-Modus starten und stoppen.....	76
4.7.1	Container „detached“ starten	76
4.7.2	Container stoppen	77
4.7.3	Container wieder entfernen.....	77
4.7.4	Container-Prozesse verwalten.....	79
4.7.4.1	Anzeige der Containerliste	79
4.7.4.2	Container „Killen“	80
4.7.4.3	Anzeigen der internen Container-Prozesse	81
4.8	Eine einfache Webseite mit NGINX Image.....	82
4.8.1	Ausführen und Test des ‚NGINX‘ Images	82
4.8.2	Unsere eigene Webseite mit NGINX	84
4.9	Eine etwas aufwendigere Webseite mit dem PHP Image	87
5.	Tools zur Arbeit mit Docker	94
5.1	Einfache Editoren	94
5.2	Visual Studio Code und Docker CLI.....	95
5.2.1	Visual Studio Remote WSL	95
5.2.2	Microsoft Docker Erweiterungen für VS Code.....	96
5.3	Visual Studio 2019 mit Docker Development Tools	97
5.3.1	Installation von Visual Studio für die Arbeit mit Docker.....	98

5.4	Eclipse und Docker.....	99
5.4.1	Installation von Doclipser.....	99
5.4.2	Editieren von Dockerfiles	100
5.4.3	Steuerung von Containern.....	100
5.5	Curl.....	100
5.5.1	Curl-Hilfe	103
5.5.2	Die wichtigsten curl-Parameter.....	103
6.	Docker-Architektur	106
6.1	Die Docker Engine	107
6.2	Docker Images und Registries	108
6.3	Docker Container.....	109
7.	Bewährte Praktiken bei der Arbeit mit Docker	111
7.1	Schreiben von Dockerfiles	111
7.1.1	Die Reihenfolge im Dockerfile	111
7.1.2	Gruppierung verwandter Build-Anweisungen	111
7.1.3	Halten Sie Ihre Images klein.....	112
7.1.4	Verbessern Sie die Wartbarkeit Ihrer Images	113
7.2	Entkoppeln Sie die Komponenten.....	113
7.3	Vergeben Sie Tags für Ihre Images.....	114
7.4	Verwenden Sie COPY anstelle von ADD	115
8.	Daten speichern in Docker	117
8.1	Docker Volumes.....	118
8.1.1	Docker Volume erzeugen.....	118
8.1.2	Docker Volume in Container einbinden.....	119
8.1.3	Docker Volume entfernen.....	123
8.2	Bind Mounts	125
8.2.1	Windows Host-Computer für „Bind Mount“ vorbereiten	125
8.2.2	„Bind Mount“ beim Start eines Containers angeben	127
9.	Log-Dateien	132
9.1	Container Logs anzeigen.....	132
9.2	Praktisches Beispiel zur Anzeige der Container Logs	134
9.3	Kontinuierliche Log-Ausgaben	137
9.4	Logging-Treiber konfigurieren.....	138
9.4.1	Konfiguration des Standard-Logging-Treibers	139
9.4.2	Konfiguration des Logging-Treibers für einen Container	143
9.5	Container Logs persistent auslagern	143

10. Netzwerke und Docker	147
10.1.1 None	148
10.1.2 Host	150
10.1.3 Bridge.....	152
10.1.4 Benutzerdefinierte bridge-Netzwerke.....	153
10.1.5 Overlay	155
10.1.6 Macvlan	156
10.1.7 Container mit Netzwerk verbinden	156
10.1.8 Container von einem Netzwerk entfernen	161
10.1.9 Übungsaufgabe: Arbeit mit Docker-Netzwerken.....	162
11. Erstellen eines WordPress-Blogs	169
11.1 Datenbank-Container starten.....	169
11.2 WordPress Container starten.....	173
11.3 Aufräumen der WordPress-Anwendung.....	178
12. Docker Compose	181
12.1 Was ist Docker Compose	181
12.2 Installation von Docker Compose.....	183
12.2.1 Installation unter Linux	183
12.3 Das YAML-Format	185
12.3.1 YAML-Elemente in Compose-Dateien.....	185
12.3.2 Sektionen in Docker Compose YAML-Dateien	187
12.3.2.1 Sektion Services	187
12.3.2.2 Networks	191
12.3.2.3 Volumes.....	191
12.4 Ein erstes Docker Compose YAML-Beispiel.....	192
12.5 Up and Down	193
12.6 Das NGINX-Beispiel erweitern.....	194
12.7 Übungsaufgabe: Docker Compose mit eigenem Image	196
12.8 Docker Compose mit zwei vernetzten Containern.....	199
12.8.1 Ein Container mit erweitertem Ubuntu Image.....	200
12.8.2 Erweiterten Ubuntu Container über Docker Compose ausführen	202
12.8.3 Einbinden eines NGINX Containers über Docker Compose	205
12.9 Umgebungsvariablen nutzen.....	210
12.9.1 Umgebungsvariable in einer Datei	210
12.9.2 Umgebungsvariablen in Compose	211
12.9.3 Umgebungsvariablen in Containern.....	212
12.9.4 Übungsaufgabe: Einsatz von Umgebungsvariablen.....	213
12.10 Services skalieren	216
12.11 Log-Dateien.....	218

13. Wordpress-Blog mit Docker Compose	221
14. Datenbank im Container	226
14.1 Beispiel MariaDB mit phpmyadmin.....	226
14.2 Abfrage der Datenbank über PHP	235
14.3 Übungsaufgabe: Die Telefon-App bearbeiten.....	242
15. Docker Swarm	246
15.1 Was ist Docker Swarm	246
15.2 Neue Begriffe für den Swarm Mode	247
15.3 Einen Single Node Swarm erstellen.....	250
15.3.1 Initialisierung des Docker Swarm Modes.....	251
15.3.2 Docker-Kommandos zur Node-Verwaltung.....	254
15.4 Docker Services.....	257
15.4.1 Einen Service erstellen	257
15.4.2 Eine Liste der Services ausgeben	259
15.4.3 Auflistung der Service Tasks.....	259
15.4.4 Einen Service entfernen	260
15.4.5 Weitere Parameter zum Erzeugen eines Service	261
15.4.6 Übungsaufgabe: Services mit Replikaten	262
15.4.7 Aktualisierung von Docker Services	264
15.4.8 Docker Services skalieren	265
15.4.9 Änderungen an Services rückgängig machen	266
15.4.10 Ausgabe von Service Logs	267
15.4.11 Ausgabe von detaillierten Service-Informationen	268
15.5 Multi Node Swarm.....	269
15.5.1 Virtuelle Nodes mit Docker Machine.....	270
15.5.2 Docker Machine unter Windows	271
15.5.2.1 Vorbereitung von Hyper-V	271
15.5.3 Docker Swarm mit Manager und Worker Nodes.....	275
15.5.3.1 Manager Node auf virtueller Maschine erstellen.....	275
15.5.3.2 Worker Node erstellen	279
15.5.3.3. Das Cluster untersuchen	282
15.5.3.4 Übungsaufgabe: den Swarm erweitern	283
15.5.3.5 Dem Swarm Services hinzufügen	284
15.5.4 Docker-Kommandos für Multi Node Swarms	286
15.6 Docker Configs - verteilte Konfigurationen	289
15.6.1 Docker-Konfiguration erstellen	290
15.6.2 Docker Configs einem Service übergeben	294
15.7 Secrets: sensitive Daten verstecken	298
15.7.1 Docker Secrets erstellen	299
15.7.2 Docker Secrets an einen Service übergeben	303
15.8 Einen Swarm auflösen	306

16. Docker Stack	309
16.1 Docker Stack in einer Single Node-Umgebung	309
16.1.1 Ein erster ganz einfacher Stack	310
16.1.2 Stack Service mit mehreren Replikaten	314
16.1.3 Configs mit Docker Stack verwalten	315
16.1.4 Secrets im Stack verwalten	319
17. Kubernetes	325
17.1 Das Zusammenspiel von Docker und Kubernetes	327
17.2 Docker Swarm und Kubernetes: eine Gegenüberstellung	327
17.3 Kubernetes-Grundlagen	328
17.3.1 Das Kubernetes-Cluster	329
17.3.1.1 Master	329
17.3.1.2 Node	331
17.3.2 Das Domain-Name-System eines Kubernetes-Clusters	333
17.3.3 Pods	333
17.3.4 Deployment	334
17.3.5 Kubernetes Services	334
17.4 Ein Kubernetes Single Node-Cluster zum Testen und Üben	336
17.4.1 Kubernetes für Docker Desktop aktivieren	337
17.4.2 Das Kubernetes-Kommando kubectl	340
17.4.3 Ein erstes einfaches Deployment	341
17.4.4 Die Deployment Manifest YAML-Datei	348
17.4.5 Ein einfaches Deployment deklarativ erstellen	350
17.4.5.1 Die YAML-Datei des Deployments	350
17.4.5.2 Ein Deployment mit create erstellen	353
17.4.6 Einen Service mit YAML erstellen	354
17.4.6.1 Die YAML-Datei	354
17.4.6.2 Einen Kubernetes Service mit create erstellen	356
17.4.7 Ein laufendes Deployment modifizieren	357
17.4.7.1 Die Anzahl der Pod-Replikate ändern	358
17.4.7.2 Anwendung mit Rolling Updates aktualisieren	359
17.4.7.3 Übungsaufgabe: Deployment ändern	363
17.4.7.4 Hier geht es weiter	365
17.5 Multi Node-Cluster mit Kubernetes	367
17.5.1 Hosted Kubernetes	368
17.5.2 Google Kubernetes Engine	369
17.5.2.1 Die Google Cloud Console	369
17.5.2.2 Erstellen eines Kubernetes-Clusters	370
17.5.2.3 Das neue Kubernetes-Cluster untersuchen	376
17.5.2.4 Mit dem Cluster verbinden	377
17.5.2.5 Manifest-Dateien für das Deployment anlegen	379
17.5.2.6 Das Deployment erzeugen	383
17.5.2.7 Das Deployment untersuchen	384
17.5.2.8 Löschen des Clusters	386
17.5.2.9 Übungsaufgabe: Die Applikation Telefon-App bereitstellen	386

18. Wie geht es weiter?	391
<hr/>	
19. Anhang	393
<hr/>	
19.1 MAC-OS Installation von Docker	393
19.1.1 Docker Desktop für MAC-OS installieren.....	393
19.1.1.1 Systemvoraussetzungen	393
19.1.1.2 Download des Installationsprogramms	393
19.1.1.3 Installation von Docker Desktop.....	397
19.1.1.4 Test der Installation	398
19.2 Linux-Installation von Docker Engine unter Ubuntu Linux.....	399
19.2.1 Betriebssystem-Anforderungen.....	399
19.2.2 Deinstallation von alten Versionen	399
19.2.3 Installation der Docker Engine Community Edition	400
19.3 Installation von Docker in einem Linux-Subsystem unter Windows	402
19.3.1 Aktivierung des Windows-Subsystems für Linux	402
19.3.2 Ubuntu-App installieren.....	405
19.3.3 Initialisierung der Ubuntu-App	407
19.3.4 Docker auf der Ubuntu-App installieren.....	407
19.4 Installation von docker-machine	411
19.4.1 Installation von docker-machine unter Windows 10.....	412
19.4.2 Installation von docker-machine unter Linux.....	414
19.4.3 Installation von docker-machine unter MAC-OS.....	414
19.5 Virtuellen Computer mit UBUNTU erstellen	414
19.6 Das Projekt „Play with Docker“	418
19.7 Das Projekt „Play with Kubernetes“	424
19.8 Ein Minikube-Cluster für Docker unter Ubuntu Linux anlegen.....	433
19.8.1 Installation von Minikube auf Ubuntu Linux	433
19.8.2 Minikube anwenden.....	435
19.8.3 Online Installationen von Minikube Terminals.....	436
19.9 Übersicht der Dockerfile-Anweisungen	437
19.10 Übersicht der Docker CLI-Kommandos	440
19.11 Format-Angaben für Docker-Kommandos.....	456
19.11.1 Abfrage der Werte von bestimmten Keys	457
<hr/>	
20. Glossar	460
<hr/>	
21. Index	469
<hr/>	

Kapitel 1

Einleitung

1.1 Vorwort

Herzlich willkommen in der Welt des Cloud-Computing mit Docker, Docker Swarm und Kubernetes. Es ist eine abenteuerliche Welt, die sich so rasant fortentwickelt, wie das bisher in der IT-Welt noch nie der Fall war. Experten, die sich in dieser Welt auskennen, sind daher in allen Branchen gefragt. Wenn sie also Ihre berufliche Zukunft verbessern wollen, dann sollten Sie sich im Umgang mit Docker und Kubernetes oder ähnlichen Technologien vertraut machen.

Es fließen im Umfeld von Docker so viele IT-Themen und Sachgebiete zusammen, dass man sich schon überwältigt fühlen könnte. Andererseits wird jeder in seinem Aufgabenbereich nur einen Teil der Möglichkeiten in der Praxis nutzen. Frontend-Entwickler werden zum Beispiel andere Schwerpunkte haben als Backend-Entwickler, DevOps-Entwickler oder Programmierer im Test-Umfeld.

Dieses Buch soll aber kein umfangreiches Nachschlagewerk sein, sondern es wird ein Überblick über Zusammenhänge von verschiedenen Themen angeboten, die in Verbindung mit Docker und Kubernetes von Bedeutung sind. Dabei wird bei diesen Themen aber nicht in die Tiefe gegangen. Es ist auch so, dass die Kapitel in diesem Buch aufeinander aufbauen. Es werden zu Beginn einfache und leicht verständliche Beispiele vorgestellt, die in nachfolgenden Kapiteln Schritt für Schritt erweitert werden.

Wie gesagt: Die Entwicklung im Umfeld des Cloud-Computing schreitet in einer solchen Geschwindigkeit voran, dass es nicht einfach ist, da Schritt zu halten.

Ständige Veränderungen sind heute die neue Normalität! Damit muss man sich in Zukunft als Entwickler abfinden.

Das hat auch Konsequenzen für dieses Buch. Die eine oder andere Information in diesem Buch ist wahrscheinlich schon dann wieder überholt, wenn es frisch aufgelegt und im Handel ist. Das gilt vor allem für Screenshots von Internetseiten und auch von Dialog- oder Programmfenstern.

Bis zuletzt habe ich bei der Arbeit an diesem Buch Screenshots ersetzt und Beschreibungen angepasst. Die Änderungen sind im Normalfall nicht so gravierend, dass man den Beschreibungen nicht mehr folgen kann. Meist ist nur der Aufbau von Fenstern oder Webseiten anders, es gibt neue Steuerelemente oder Steuerelemente verschwinden. Gelegentlich ändert sich auch die Reihenfolge von Aktionen, die dort durchgeführt werden sollen.

1.2 Die Microservice Revolution

Microservice-Architekturen in Kombination mit der Virtualisierung von Systemen durch Container sind in den letzten Jahren wie eine Flutwelle über die Softwareentwicklung und auch über die Entwickler-teams hereingebrochen. Es haben sich ganz neue Technologien und Tools etabliert, ganz neue Möglichkeiten sind sichtbar geworden. Die Sub-Systeme werden unabhängiger voneinander und sogar unabhängiger von den Betriebssystemen und von der Hardware, auf denen sie ausgeführt werden.

Den entscheidenden letzten Schub erhielt die Idee der Microservices durch die Veröffentlichung von Docker, was zur Folge hatte, dass die vereinfachte Nutzung auf Container basierten, virtuellen Systemen einer noch größeren Gemeinschaft von Entwicklern zugänglich gemacht wurde.

Die Grundidee der Microservice Architektur ist nicht neu. Seit vielen Jahren hat sich in der Softwareentwicklung die Erkenntnis durchge-

setzt, dass es besser ist, ein System in viele kleine Komponenten aufzuteilen, von denen jede genau eine Aufgabe perfekt erledigt (do one thing and do it well), als ein großes monolithisches System zu erstellen, das alles kann (die berühmte eierlegende Wollmilchsau).

Damit wird die Wiederverwendbarkeit von Softwarekomponenten drastisch erhöht. Gleichzeitig ist es leichter, solche Systeme zu debuggen, zu warten und zu erweitern.

Auch Microservices folgen diesem Architekturmuster, bei dem komplexe Softwaresysteme aus Komponenten zusammengesetzt werden, die voneinander unabhängig, also entkoppelt, sind.

Neu bei den Microservices ist die hohe Flexibilität. Für die Ausführung jedes Services wird jeweils ein eigener Prozess zur Verfügung gestellt. Die Kommunikation zwischen diesen Prozessen erfolgt über sehr schlanke Schnittstellen, die auch noch unabhängig von der verwendeten Programmiersprache sind.

Entwickler werden bei der Entwicklung von Microservices verstärkt dazu bewegt, die Systeme in kleinere Komponenten aufzubrechen und diese voneinander zu entkoppeln.

Wird ein System beim Entwurf in viele Microservices aufgebrochen, dann ist es möglich, dass diese dezentral und unabhängig voneinander von verschiedenen Teams entwickelt und verteilt werden. Auch die Skalierung der beteiligten Services ist unabhängig vom Gesamtsystem.

Durch die Einführung der Container-Technologie mit virtuellen Maschinen und virtuellen Servern hat man die Voraussetzungen geschaffen, dass Microservices überall im Web und unabhängig von den verfügbaren Plattformen ausgeführt werden können. Es ist so auch möglich, die Systeme gegen Ausfall anderer Services abzusichern.

Noch ein Vorteil ist, dass „Continuous Delivery“ durch die Aufteilung in kleinere Services einfacher wird.

Als Nachteil muss hier allerdings eine erhöhte Komplexität beim Testen der Software, beim Verteilen der Software, beim Logging und beim Monitoring in Kauf genommen werden. Die Fehlertoleranz dieser Systeme sinkt meist und die Last im Netzwerk steigt.

Insgesamt kann man aber sagen, dass mit der Einführung von Microservice-Architekturen die damit entwickelten Applikationen schneller entwickelt werden können. Sie sind leichter zu warten, robuster und von insgesamt höherer Qualität.

1.3 Das Ziel dieses Buches

Dieses Buch wurde für verschiedene Lesergruppen konzipiert.

Da sind in erster Linie die Entwickler, die mit Docker flexible Webservices oder Applikationen entwickeln möchten. Dabei soll auf der einen Seite dem Docker-Neuling ein verständlicher und leicht nachvollziehbarer Einstieg in die Dockerwelt geboten werden. Es werden zunächst die Begriffe aus dem Docker-Umfeld erklärt. Es folgt eine Schritt-für-Schritt-Anleitung zur Installation von Docker unter Windows. Die Installation unter MacOS und Linux wird im Anhang beschrieben. Dann gibt es eine Übersicht über die Funktionen und Einsatzmöglichkeiten von Docker. Die praktische Handhabung wird mit einfachen Beispielen demonstriert. Zusätzliche Aufgaben im Anschluss an ein Thema helfen dabei, das neu Erlernte zu vertiefen.

Wir geben Ihnen eine Übersicht über die vorhandenen Images im Docker Hub. Dabei stellen wir einige nützliche Images, die häufig zum Einsatz kommen, genauer vor.

Ein eigenes Kapitel ist der Übersicht über verfügbare nützliche Tools für die Arbeit mit Docker gewidmet. Die Funktionalität dieser Werkzeuge wird kurz beschrieben und es wird auch gezeigt, wie sie die Arbeit mit Docker unterstützen bzw. vereinfachen.

Um die Funktionsweise von Docker besser zu verstehen, werden im Kapitel über die Docker-Architektur die verschiedenen Docker-Komponenten genauer beschrieben. Das Verständnis für Struktur und die Abhängigkeiten der Komponenten wird vertieft und durch Grafiken veranschaulicht.

Falls Sie als Entwickler schon Erfahrung mit Docker gesammelt haben, wird der Teil des Buches mit den fortgeschrittenen Techniken der interessanter für Sie sein. Dort werden Ihre Docker-Kenntnisse vertieft und weitere Tools vorgestellt. Dazu gehören Themen wie die Orchestrierung von Docker Containern mit Docker Compose, der Einsatz von Docker Swarm und Docker Stack.

Als umfangreicheres praktisches Beispiel entwickeln wir mit Ihnen schrittweise eine WordPress-Blog-Anwendung. Dabei werden die benötigten Funktionalitäten als Microservices entworfen, auf mehrere Container aufgeteilt und mit Docker Compose verwaltet. Später stelle ich vor, wie man solche Anwendungen mit Docker Swarm verteilt.

Zahlreiche Tipps und Tricks aus der Praxisarbeit mit Docker dürfen in diesem Buch natürlich auch nicht fehlen.

Schließlich stelle ich Ihnen in diesem Buch noch vor, wie bei Docker-Anwendungen die Bereitstellung, Skalierung und Verwaltung mithilfe von Kubernetes automatisiert werden kann.

Neben Informationen für Web-Entwickler enthält dieses Buch auch Informationen für Softwarearchitekten, Projektleiter und andere Entscheidungsträger. Diese sind bei Entscheidungen zur Auswahl aus möglichen Technologien und verfügbaren Tools recht nützlich.

Nicht zuletzt möchte ich hier noch die Studierenden verschiedener Fachrichtungen erwähnen, die mit dem Einsatz von Containertechnologie neue und innovative Anwendungen im Web entwickeln möchten.

1.4 Konventionen im Buch

Texte, die in der Kommandozeile einer Shell (z.B. PowerShell oder einem Linux TTY) eingegeben werden, sind in Schreibmaschinenschrift (Courier) gesetzt. Das gleiche gilt für Texte die als Programm-Quellcode oder als Skript-Anweisungen eingegeben werden. Auch der Inhalt von Dockerfiles und Docker-Compose-Dateien ist in Courier formatiert.

Shell-Kommandos für die Windows PowerShell werden mit dem führenden Zeichen `>` angegeben. Bei anderen Shells wird `$` oder `#` so verwendet, dass es jeweils zu den Screenshots der vorgegebenen Beispiele passt.

```
1 > docker image ls
2 $ sudo apt-get update
3 # cat hello.txt
```

Dieses Zeichen wird nicht mit eingetippt. Es steht als Platzhalter für das System Prompt.

Um umfangreichere Kommandos übersichtlicher zu gestalten, werden diese im Buch mehrzeilig, als mit Zeilenumbruch, dargestellt. Bei der mehrzeiligen Darstellung von Shell Kommandos wurden hier Back-ticks (```) zum Maskieren des Zeilenendes verwendet.

Beispiel:

```
1 > docker run -i -t `
2 --name=voltest `
3 --mount source=test-vol,target=/test_data `
4 ubuntu /bin/bash
```

Dies ist die Variante für die Eingabe in einer PowerShell. Bei anderen Shell-Applikationen, wie zum Beispiel der Ubuntu Shell, ist das Zeichen zum Maskieren von Sonderzeichen wie einem Zeilenende in der Regel der Backslash (`\`).

Falls Sie also nicht mit der PowerShell arbeiten, dann ersetzen sie das Backtick-Zeichen aus den entsprechenden Beispielen durch einen Backslash.

Beispiel:

```
1 > docker run -i -t \  
2 --name=voltest \  
3 --mount source=test-vol,target=/test_data \  
4 ubuntu /bin/bash
```

1

Dateinamen oder andere Namen, die im System vergeben werden können (z.B. Namen von Datenbanken oder Tabellen), sind ebenfalls in der Schriftart Courier gehalten und werden zwischen einfache Anführungszeichen gesetzt.

Beispiel:

```
1 'docker-compose.yaml'
```

Textverweise auf Elemente von Benutzeroberflächen wie Fenster oder Webseiten, wie zum Beispiel Menübefehle, Schaltflächen und Steuerelemente, sind als KAPITÄLCHEN dargestellt. Die Angabe

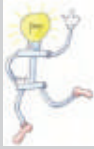
DATEI | NEU | PROJEKT

bedeutet, dass sie aus dem Hauptmenü den Menüpunkt DATEI auswählen, aus dem Drop-Down-Menü den Befehl NEU und dort aus dem Untermenü die Auswahl PROJEKT.

Die Angabe [HILFE] weist auf eine Schaltfläche mit dem Label [Hilfe] hin.

Optionen in Dialogfeldern sind *Kursiv* formatiert.

Namen von Fenstern, wie zum Beispiel Dialogfenster, werden als Kapitälchen zwischen „ANFÜHRUNGSZEICHEN“ gesetzt.



Eine vollständige Beschreibung der Kommandos für die Docker CLI sowie die ausführliche Beschreibung der Dockerfile-Anweisungen finden Sie im Anhang dieses Buches.

1.5 Warum braucht man Docker?

Wenn man eine Anwendung entwickelt, dann möchte man diese so vielen Anwendern wie möglich zugänglich machen. Aber diese Anwender nutzen verschiedene Ausführungsumgebungen, das heißt verschiedene Hardware mit unterschiedlichen Betriebssystemen, die dann auch noch verschiedene Versionen haben können.

Dann sollen in der Regel auf diesen Systemen auch noch mehrere Anwendungen lauffähig sein. Die nutzen dann meistens Laufzeitbibliotheken, Tools oder Datenbanken, die es auch wieder in mehreren Versionen gibt und die natürlich untereinander, mit den installierten Anwendungen und mit der aktuellen Betriebssystemversion kompatibel sein müssen.

Diese Situation hat in der Vergangenheit regelmäßig zu Problemen geführt, die nur mit viel Aufwand gelöst werden konnten. Administratoren haben in diesem Zusammenhang schon von der ‚Konfigurationshöhle‘ oder dem ‚DLL Versions-Alptraum‘ gesprochen.

Um die Verteilung von Anwendungen zu vereinfachen, sind als Lösung für diese Probleme sogenannte Container-Plattformen entwickelt worden. Eine davon ist Docker, das im Moment wahrscheinlich beliebteste und am weitesten verbreitete Container-System.

1.6 Was muss ich mir unter Docker vorstellen?

Docker ist ein Tool, welches die Entwicklung, Verteilung und Ausführung von Anwendungen durch die Nutzung von Containertechnologie vereinfacht.

Eine Applikation kann dabei mit allen Bestandteilen, die sie braucht, zusammengepackt werden. Dazu gehören zum Beispiel Bibliotheken, Datenbanken, Treiber oder auch Konfigurationsdateien.

Docker ist also eigentlich nichts weiter als eine Ansammlung von Produkten. Diese kommen aus dem Umfeld von , Plattform-as-a-Service‘ (PaaS).

Durch den Einsatz virtueller Betriebssysteme, die auf den Ziel-Plattformen laufen, wird es dann möglich, Serveranwendungen als komplette Pakete, sogenannte ,Container‘, auszuliefern.

Docker Container sind dadurch systemunabhängig. Ob Anwendungen unter Windows, Linux oder MacOS ausgeführt werden, spielt keine Rolle mehr, wenn diese in Docker Container verpackt sind und durch ein virtuelles System ausgeführt werden.

Server-Applikationen können in Docker Containern fertig installiert und mit diesen verteilt werden. Aufwendige und zeitraubende Setup- oder Install-Aktivitäten, wie wir sie bisher kennen, bleiben den Administratoren damit erspart.

Ein weiterer Vorteil dabei ist, dass vorhandene Server-Applikationen, die bisher direkt aus einem Betriebssystem heraus ausgeführt wurden, ohne Weiteres in Docker Container überführt werden können. Dort laufen diese, ohne dass der Quellcode geändert oder angepasst werden muss.

1.7 Was ist Docker nicht?

Nur um Missverständnissen vorzubeugen: Docker Container sind nicht für die Ausführung von typischen Client-Anwendungen gedacht (z.B. Word oder Power Point). Das können virtuelle Maschinen wie zum Beispiel ,Virtual Box‘ oder VMware besser.

In Docker Containern werden also ausschließlich Server-Anwendungen ausgeführt, die über Schnittstellen mit Protokollen wie HTTP kommunizieren.

Man kann sich Docker ein wenig wie eine Virtuelle Maschine vorstellen (z.B. Virtual Box), aber Docker ist viel leichtgewichtiger als eine Virtuelle Maschine.

1.8 Entwicklungsgeschichte

Docker ist jetzt 7 Jahre alt und in der Zwischenzeit ist recht viel passiert. Es folgt an dieser Stelle eine Übersicht über die wichtigsten Meilensteine in der Entwicklungsgeschichte von Docker.

- ▶ Die Firma ‚Docker Inc.‘ wurde im Sommer 2010 von Solomon Hykes und Sebastian Pahl gegründet. Hykes startete damals das Docker-Projekt in Frankreich innerhalb von ‚dotCloud‘, einer ‚PaaS‘ (Platform as a Service) Firma.
- ▶ Docker wurde erstmals 2013 während der Python Conference (PyCon) in Santa Clara vorgestellt.
- ▶ Im März 2013 wurde Docker als Open-Source-Software freigegeben (released).
- ▶ Zunächst lief Docker noch auf LXC, einem virtuellen Linux Container.
- ▶ Ein Jahr später hat man LXC durch eigene Docker-Komponenten ersetzt. Diese wurden in der Programmiersprache GO entwickelt. GO oder auch Golang ist eine Programmiersprache, die von Robert Griesemer bei Google entworfen wurde. GO ähnelt sehr der Programmiersprache C, bietet aber Speicher-Sicherheit, Garbage Collection und Typ-Sicherheit.
- ▶ Im September 2013 gaben Docker und Red Hat ihre Zusammenarbeit im Umfeld von ‚Fedora Linux‘, ‚Red Hat Enterprise Linux‘ und ‚Open Shift Container‘ bekannt.

- ▶ Danach, im Oktober 2013, kündigt dotCloud an, dass es sich in Docker umbenennt.
- ▶ Amazon gibt im November 2014 den Einsatz von Docker Container Services im Umfeld der ‚Amazon Elastic Compute Cloud‘ (EC2) bekannt. EC2 ist der zentrale Bestandteil der Amazon Web Services (AWS).
- ▶ Microsoft integriert im Oktober 2014 die Docker Engine in ‚Windows Server‘.
- ▶ Docker gibt im November 2014 seine Partnerschaft mit Stratoscale bekannt.
- ▶ Im Dezember 2014 folgt Partnerschaft von Docker mit IBM. Dadurch wird die Voraussetzung für eine bessere Integration der IBM Cloud mit Docker geschaffen.
- ▶ Docker, in Zusammenarbeit mit verschiedenen anderen Organisationen, erklären im Juni 2015, dass sie an einem neuen Standard für Software Container arbeiten, der unabhängig von Herstellern und Betriebssystemen sein soll.
- ▶ Im Herbst 2016 ist Docker zum ersten Mal im ‚Native Mode‘ (d.h. ohne zusätzliche externe Software Layer) unter ausgewählten Windows Versionen verfügbar.
- ▶ Microsoft gibt im Mai 2019 die Version 2 von WSL, dem Windows Subsystem für Linux, heraus. Es handelt sich dabei um eine Kompatibilitätsschicht zur Ausführung von LINUX-Applikationen unter Windows 10. Docker Inc. beginnt danach mit der Entwicklung einer Docker-Version für Windows, die auf der Basis von WSL 2 läuft.
- ▶ Im Oktober 2019 wird bekannt, dass Docker finanzielle Probleme hat. Trotz zahlreicher Erfolge hat Docker es wohl nicht geschafft, wirtschaftlich erfolgreich zu sein.
- ▶ Die bisher neueste Meldung kommt Mitte November 2019 – Docker verkauft die Docker Enterprise Sparte an den Cloud-Dienstleister Mirantis. Docker Inc. erklärt dazu, dass man sich wieder mehr auf Docker Hub und Docker Desktop konzentrieren will.

Mit dem Verkauf wurde auch der CEO von Docker ausgetauscht. Den Posten übernimmt der bisherige CPO Scott Johnston von Rob Bearden.

Alle Programmcodes aus diesem Buch sind als PDF zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:

<https://bmu-verlag.de/books/docker/>



Außerdem erhalten Sie die eBook Ausgabe zum Buch im PDF Format kostenlos auf unserer Website:



<https://bmu-verlag.de/books/docker/>

Downloadcode: siehe Kapitel 19