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.

WirmachenDruck.de
Sie sparen, wir drucken!

Docker Handbuch für Einsteiger

Inhaltsverzeichnis

16. Docker Stack 309

17. Kubernetes 325

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 Entwicklerteams 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 zugängig 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 „Continous 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 interessantere 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 benötigen 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.

```
> docker image ls
$ sudo apt-get update
# 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 Backticks (`) zum Maskieren des Zeilenendes verwendet.

Beispiel:

```
> docker run -i -t '
--name=voltest '
--mount source=test-vol,target=/test_data '
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:


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

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:

```
'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 ‚Konfigurationshölle' 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 ‚Vitual 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

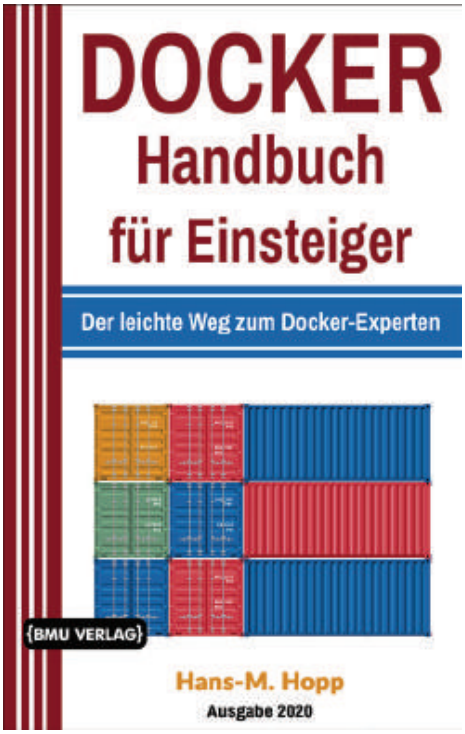
Docker ist jetzt 7 Jahre alt und in der Zwischenzeit ist recht viel passiert. Es folgt an dieser Stelle eine Übersicht über die wichtigsten Meilensteine in der Entwicklungsgeschichte von Docker.

- 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 Conferece (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 Grisemeier 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 Enterprice 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 Cloude' (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