# Vorwort

Zu Beginn der 2000er-Jahre stellte Virtualisierungssoftware den Alltag vieler Entwickler auf den Kopf: Plötzlich war es möglich, auf einem Rechner Linux *und* Windows auszuführen, unkompliziert Programme in verschiedenen Umgebungen bzw. Web-Apps in alten Versionen von Webbrowsern auszuprobieren, verschiedene Software-Stacks in virtuellen Maschinen parallel zu installieren und zu testen und vieles mehr.

Natürlich spielen virtuelle Maschinen für Entwickler weiter eine große Rolle; außerdem ist die Cloud in ihrer jetzigen Form ohne Virtualisierung gar nicht denkbar. Dennoch hat vor einigen Jahren ein Umbruch weg von virtuellen Maschinen hin zu Containern begonnen – und dieser Umbruch scheint sich mehr und mehr zu beschleunigen.

Container ermöglichen es, bestimmte Softwarekomponenten (Webserver, Programmiersprachen, Datenbanken) ohne den Overhead einer virtuellen Maschine auszuführen. Warum ein ganzes Betriebssystem (meist Linux) in eine virtuelle Maschine installieren, wenn es doch nur um *eine* ganz spezifische Funktion geht?

Selten trifft das Paradigma »weniger ist mehr« so gut zu wie auf die Container-Technologie. Das »weniger« drückt sich in unzähligen Vorteilen aus: Container sind viel schneller aufgesetzt als virtuelle Maschinen, lassen sich leichter auf verschiedenen Entwicklungssystemen replizieren, beanspruchen weniger Ressourcen und bieten wesentlich bessere Möglichkeiten zur Skalierung und Lastverteilung. Container sind insofern nicht nur ein Segen für Entwicklerteams, sondern bieten auch vollkommen neue Möglichkeiten im Deployment, also im produktiven Betrieb der entwickelten Lösung.

## Docker

Docker ist *die* Container-Software schlechthin. Zwar gibt es mittlerweile durchaus interessante Alternativen, aber Docker hat den Container-Markt als solchen geschaffen. In einer Umfrage der Website *Stack Overflow* unter Tausenden von Entwicklern war Docker nach Linux und Windows die wichtigste Plattform. Bei der Fragestellung »Most Wanted Platform« war Docker sogar die Nummer 1. Docker ist also das Werkzeug, das die meisten Entwickler neu erlernen wollen. Und bei den Entwicklern, die Docker bereits anwenden, erreichte Docker Platz 2 bei der Fragestellung »Most Loved Platform«. (Ein paar Prozent mehr Zustimmung hatte in diesem Fall Linux.)

*https://insights.stackoverflow.com/survey/2020*

Allerdings geriet die Firma *Docker Inc.* 2019 in finanzielle Schwierigkeiten. Im November 2019 kaufte der Cloud-Anbieter Mirantis die Enterprise-Sparte von Docker und machte aus *Docker Enterprise* das neue Produkt *Mirantis Kubernetes Engine*. Die verbleibende Firma kümmert sich seither stärker um die Bedürfnisse von »gewöhnlichen« Entwicklern und Entwicklerteams, die nicht Teil riesiger Unternehmen sind.

Müssen Sie sich Sorgen machen, dass Docker aufgrund wirtschaftlicher Probleme einfach verschwinden könnte? Glücklicherweise nicht! Nahezu alle Komponenten von Docker sind als Open-Source-Software verfügbar. Einige Schnittstellen von Docker wurden zudem durch die *Open Container Initiative* (OCI) standardisiert und erleichtern es anderen Anbietern, mit Docker kompatible Produkte anzubieten. Deswegen berücksichtigen wir in diesem Buch auch das für Linux-Anwender interessanteste Konkurrenzprodukt *Podman*.

Docker hat also einen Standard gesetzt, der aller Voraussicht nach Bestand haben wird. Wie agil Docker nach wie vor auftritt, war zuletzt bei der Einführung der neuen Apple-CPUs auf ARM-Basis zu sehen. Innerhalb weniger Monate hat Docker sämtliche Tools auf ARM portiert. Kein anderes Container-System kann da mithalten.

### Wozu dieses Buch?

In diesem Buch geben wir Ihnen eine Einführung in den Umgang mit Docker und einen Überblick über die wichtigsten Bausteine (Images), aus denen Sie eigene Container-Welten zusammensetzen können. Wir zeigen Ihnen anhand mehrerer großer Beispiele, wie Sie Docker in der Praxis einsetzen, und gehen ausführlich auf das Deployment in der Cloud ein.

Wir haben das Buch in drei Teile gegliedert:

- **Teil I** stellt Docker vor. Sie lernen anhand vieler Beispiele, wie Sie die Kommandos `docker` und `docker-compose` sinnvoll einsetzen und wie die Syntax der Dateien `Dockerfile` und `docker-compose.yml` aussieht.
- **Teil II** präsentiert wichtige Images, die als Basis für eigene Projekte dienen können. Dazu zählen unter anderem:
  - Alpine Linux
  - die Webserver Apache und Nginx (inklusive Proxy-Setup mit Traefik sowie Let's-Encrypt-Konfiguration)
  - die Datenbankserver MySQL/MariaDB, MongoDB, PostgreSQL und Redis
  - die Programmiersprachen JavaScript (Node.js), Java, PHP, Ruby und Python
  - die Webapplikationen WordPress, Joomla und Nextcloud





- **Teil III** zeigt den Einsatz von Docker in der Praxis. Wir zeigen Ihnen sowohl, wie Sie moderne Webapplikationen mit Docker besonders effizient entwickeln, als auch, wie Sie vorhandene Projekte mit all ihren Altlasten in besser wartbare Docker-Projekte umwandeln.

  Zwei Kapitel zur Nutzung von GitLab mit Docker und zu *Continuous Integration* (CI) und *Continuous Delivery* (CD) demonstrieren Ihnen neue Paradigmen und Hilfsmittel für das Entwickeln von Software im Team.

  Auch das Deployment kommt nicht zu kurz: Mit Docker *Swarm* und *Kubernetes* bringen Sie Ihre Docker-Projekte in die Cloud und profitieren von den dort gegebenen Möglichkeiten zur Skalierung. Eine Sammlung von Tipps stellt sicher, dass dabei die Sicherheit nicht zu kurz kommt.

  Schließlich stellen wir Ihnen im Anhang das von Red Hat entwickelte Programm *Podman* kurz vor. Es übernimmt viele Ideen und die Syntax von Docker, beschreitet aber bei der Implementierung neue Wege. Podman ist allerdings nur in der Linux-Welt relevant.

### Neu in der dritten Auflage

Für diese Auflage haben wir das Buch vollständig aktualisiert und die Einführungskapitel übersichtlicher strukturiert. Wichtige inhaltliche Neuerungen sind:

- **Rootless Docker:** Docker ohne `root`-Rechte verwenden
- **CPU-Architekturen:** Docker auf Apple-Computern mit ARM-Prozessoren
- **Pull-Limit:** Docker-Hub-Limits beim Image-Zugriff umgehen
- **Container automatisch starten:** `restart`-Option und systemd
- **neue GUI-Tools:** Docker Desktop, VSCode, Portainer
- **Traefik:** ein Proxy-Server speziell für Container-Anwendungen

### Schöne neue Docker-Welt

Wer Docker einmal ausprobiert und kennengelernt hat, will nie wieder auf seine Funktionen verzichten. Lassen Sie sich von uns in eine neue Welt führen!

Bernd Öggl (*https://komplett.cc*)
Michael Kofler (*https://kofler.info*)

### Materialien zum Buch

Auf der Webseite zu diesem Buch stehen folgende Materialien für Sie zum Download bereit:

- alle Projektdateien
- alle Codebeispiele

Gehen Sie auf *https://www.rheinwerk-verlag.de/5393*. Klicken Sie auf den Reiter MATERIALIEN ZUM BUCH. Dort sehen Sie eine Liste der herunterladbaren Dateien samt einer Kurzbeschreibung. Klicken Sie auf den Button HERUNTERLADEN, um den Download zu starten.

### GitHub

Wir verwenden Git zur Verwaltung unserer Beispieldateien. Deshalb finden Sie (eventuell bereits aktualisierte oder korrigierte) Beispieldateien auch auf GitHub:

*https://github.com/docbuc*

Das Repository *https://github.com/docbuc/dockerbuch* enthält eine Übersicht der einzelnen Kapitel mit den Links zu den entsprechenden Repositories mit Git-Sub-Modulen. Das bedeutet, dass Sie mit den folgenden Befehlen alle Beispiele von GitHub auf Ihren Computer laden können:

```
git clone https://github.com/docbuc/dockerbuch.git
cd dockerbuch
git submodule update --init --recursive
```

Sollten Sie Verbesserungsvorschläge haben, freuen wir uns natürlich über Pull-Requests von Ihnen.

### Testplattformen und Linux-Distributionen

Unsere Haupttestplattform für dieses Buch war Ubuntu Linux. Parallel dazu haben wir viele Beispiele auch unter Windows und macOS sowie unter Debian, Fedora und Oracle Linux ausprobiert.

Noch eine Anmerkung zu Red Hat: Wenn in diesem Buch von RHEL die Rede ist, dann ist damit nicht nur Red Hat Enterprise Linux in der Version 8 gemeint; vielmehr gelten die Informationen auch für alle damit kompatiblen Distributionen. Das gilt insbesondere für AlmaLinux, CentOS Stream, Rocky Linux und Oracle Linux.