Vorwort

Die weitverbreitete technische Evolution, die sich um uns herum vollzieht, konzentriert sich auf ein scheinbar einfaches Tool: den Container. Etwas vom Design her so Kleines und Leichtgewichtiges hat einen gewaltigen Einfluss auf die Softwareentwicklung in allen Branchen. Und das innerhalb kürzester Zeit.

Containerisierung ist allerdings nicht neu und war es auch 2013 nicht, als Docker zum ersten Mal vorgestellt wurde. Allerdings war die Containerisierung vor Docker bei den meisten Softwareentwicklern kaum auf dem Radar. Selbst die Low-Level-Konzepte hinter Containern wurden überwiegend nur von denen verstanden, die ein tiefes Verständnis vom Linux-Kernel hatten oder bei einigen der Tech-Giganten wie Sun oder Google arbeiteten. Windows-Entwickler oder Systemadministratoren wurden generell außen vor gelassen. Heutzutage ist es schwer, ein Gespräch über eine Software zu führen, ohne Docker zu erwähnen. Aber wie sind wir zu diesem Punkt gekommen, und wo geht die Reise noch hin?

Wir nutzen Docker nicht mehr, weil es neu ist, oder nur wegen der reinen Technologie. Es wird hauptsächlich verwendet, weil es den Entwicklungsprozess beschleunigt, die Ausgaben für Infrastruktur und Overhead verringert, das Onboarding neuer Entwickler erleichtert und sogar die Barriere zwischen den Entwicklungs- und Administratorenteams reduziert. Windows-Nutzer können dank der Arbeit von Microsoft, Docker und weiteren zahlreichen Open-Source-Software-(OSS-)Anbietern nun ebenfalls von den Vorteilen von Docker profitieren.

Trotz all seiner Vorteile ist Docker jedoch nicht immer ein einfaches Tool. Man muss es richtig verstanden haben, um Cloud-Native-Anwendungen bauen und administrieren zu können. Cloud-Native-Anwendungen sind hochverfügbare, skalierbare Anwendungen, die auf Managed-Cloud-Infrastrukturen laufen. Um diese Resilienz und Skalierbarkeit zu erreichen, muss man auf Container-Orchestrierungstechnologien wie Kubernetes setzen. Zusätzlich dazu sind Cloud-Native-Anwendungen in der Regel serviceorientiert oder folgen dem Microservice-Architektur-Ansatz.

Ich werde oft gefragt, ob Docker virtuelle Maschinen (VMs) ersetzt, ob Microservice-Architekturen eine Voraussetzung sind oder ob Unternehmen lieber alles über Docker vergessen und stattdessen auf Serverless Computing setzen sollten, was immer populärer wird. Die Antwort lautet immer: Nein! Tools im Cloud-Native-Ökosystem sind ein Zusatzmittel und nicht exklusiv. Docker und VMs ste-

hen nicht in Konkurrenz zueinander, sondern sollten gemeinsam genutzt werden, um den maximalen Nutzen herauszuziehen. Serverless Computing ist dann am sinnvollsten, wenn es mit Containern genutzt wird. Ich würde sogar behaupten, dass Serverless Computing überhaupt nicht so populär wäre, wenn es keine kurzlebigen und leichtgewichtigen Container gäbe. Microservices sind auch keine Voraussetzung für Container. Allerdings ziehen Sie mehr Vorteile aus Containern, wenn Ihre Architektur kleinere Services erlaubt.

Der Einsatz von Docker erlaubt den Entwicklern, ihre Zuständigkeiten in den Bereich der Administration auszudehnen und für das, was sie entwickelt haben, die Verantwortung zu übernehmen. Das kann die Zusammenarbeit über Abteilungsgrenzen hinweg fördern, da Details wie Abhängigkeiten auch in den Verantwortungsbereich des Entwicklungsteams fallen statt ausschließlich in den der Administratoren. Darüber hinaus sind Teams so in der Lage, bessere Artefakte zu generieren, die als Eckpunkte für die Dokumentation genutzt werden können: Ein *Dockerfile* und eine *docker-compose.yml* können zusammen die Rolle eines Leitfadens für die Inbetriebnahme des Projekts übernehmen. Neue Entwickler im Team oder Entwickler in Open-Source-Projekten können in wenigen Schritten produktiv arbeiten, wenn diese Dateien existieren. In der Vergangenheit war es oft eine mehrtägige Aufgabe, eine Entwicklungsumgebung aufzusetzen. Heutzutage können wir das mit einem einfachen, reproduzierbaren Workflow ersetzen: Docker installieren, das Repository klonen und `docker-compose up` laufen lassen.

Ähnlich wie das Cloud-Native-Ökosystem ein Hilfsmittel ist, sind auch viele Docker-eigene Tools praktische Hilfsmittel, und das Beherrschen der Grundlagen wird Ihnen helfen, erfolgreicher zu sein. In diesem Buch sollten Sie Kapitel 4 ganz besondere Aufmerksamkeit widmen. Dieses Kapitel beschäftigt sich mit den Container-Images inklusive der wichtigsten Datei im Projekt: dem Dockerfile. Diese Datei ist, neben den anderen Images, die Sie möglicherweise direkt von einer Image-Registry herunterladen, die Basis für Ihre Anwendung. Jeder weitere Layer, wie Container-Orchestrierung, basiert darauf, dass Sie Ihre Anwendung mittels eines Dockerfiles in einem Image paketiert haben. Sie lernen, dass jeder Anwendungscode, unabhängig von Alter, Framework, Sprache oder Architektur, in ein Container-Image paketiert werden kann.

In diesem Buch teilen Sean und Karl ihr umfangreiches kollektives Wissen, um Ihnen das breite theoretische und taktische Verständnis von Docker und dem dazugehörigen Ökosystem zu vermitteln mit dem Ziel, Ihnen zu einem erfolgreichen Start in die Containerisierung zu verhelfen. Während Docker die Entwicklungsprozesse vereinfachen und optimieren kann, ist es aber auch ein mächtiges Tool mit zahlreichen Komplexitätsschichten. Sean und Karl haben dieses Buch sorgfältig zusammengestellt, um sich auf das Wesentliche zu konzentrieren und Ihnen dabei zu helfen, schnell produktiv zu werden und trotzdem die wichtigen Grundlagen zu verstehen.

Saugen Sie das Wissen und die Erfahrung, die auf diesen Seiten geteilt werden, auf und behalten Sie dieses Buch als Nachschlagewerk. Sie werden es nicht bereuen.

-- Laura Frank Tacho

Docker Captain und Director of Engineering, CloudBees

Twitter: @rhein_wein

Einleitung

Dieses Buch richtet sich sowohl an System Engineers als auch an Entwickler. Es weist Ihnen den Weg zu einer funktionierenden Docker-Umgebung und einer vernünftigen Produktivumgebung. Auf dem Weg werden wir erfahren, wie man Docker-Anwendungen baut, testet, deployt und debuggt – und das sowohl in der Entwicklung als auch in der Produktion. Wir behandeln zudem ein paar wichtige Orchestrierungstools aus dem Docker-Ökosystem. Hilfestellungen zu Security und Best Practices für Ihre Containerumgebung runden das Kapitel ab.

Wer sollte dieses Buch lesen?

Das Buch richtet sich an Leser, die nach Lösungen für die verschiedenen mit dem komplexen Workflow bei Entwicklung und Deployment von Anwendungen einhergehenden Problemen suchen. Wenn Sie an Docker, Linux-Containern, DevOps und umfangreichen skalierbaren Softwareinfrastrukturen interessiert sind, ist dieses Buch genau das Richtige für Sie.

Warum überhaupt dieses Buch lesen?

Heutzutage sind jede Menge Foren, Projektbeschreibungen und Artikel zum Thema Docker im Internet verfügbar. Warum also sollten Sie Ihre kostbare Zeit mit dem Lesen dieses Buchs verbringen?

Nun, auch wenn tatsächlich schon viele Informationen bereitstehen, ist Docker doch eine neue Technologie, die sich rasant weiterentwickelt. Während wir die erste Auflage dieses Buchs geschrieben haben, hat Docker, Inc. allein fünf neue Versionen veröffentlicht und sein hauseigenes Ökosystem um eine Reihe bedeutender Tools erweitert. In den drei Jahren zwischen der ersten und zweiten Auflage dieses Buchs hat sich die Docker-Landschaft stark verändert. Docker wurde viel stabiler, und mittlerweile gibt es eine Auswahl an guten Tools für nahezu jeden Aspekt aus dem DevOps-Workflow. Zu verstehen, was mit Docker alles möglich ist und wie es zu Ihrem Workflow passt und darin eingebunden werden kann, ist keine triviale Aufgabe. So haben wir an Aufbau und Betrieb von produktiven Docker-Umgebungen für mehrere Unternehmen über vier Jahre gearbeitet.

Wir haben Docker nur wenige Monate nach seinem Release in einer Produktivumgebung implementiert und möchten in den nachfolgenden Kapiteln einige Erkenntnisse mit Ihnen teilen, die wir in den Jahren 2014 und 2015 im Rahmen der Weiterentwicklung unserer Plattform gewonnen haben. Ziel soll es hierbei sein, Sie von unseren Erfahrungen profitieren zu lassen, sodass Sie den Stolpersteinen, denen wir begegnet sind, soweit möglich aus dem Weg gehen können. Natürlich ist auch die Onlinedokumentation des Docker-Projekts zwar durchaus nützlich, wir möchten Ihnen hier jedoch ein etwas umfassenderes Gesamtbild vermitteln und Ihnen einige der Verfahrensweisen vorstellen, die sich bestens bewährt haben.

Nach der Lektüre dieses Buchs sollten Sie über hinreichende Kenntnisse verfügen, um zu verstehen, was Docker eigentlich leistet, warum es von Bedeutung ist, wie Sie es zum Laufen bekommen, wie Sie Ihre Anwendungen bereitstellen können und was erforderlich ist, um es in einer Produktivumgebung einzusetzen. Lassen Sie sich von diesem Buch auf eine kurze, aufschlussreiche Reise in das Universum einer interessanten Technologie mitnehmen, die einige sehr praktische Anwendungen bietet.

Aufbau des Buchs

Hier ein Überblick über den Inhalt des Buchs:

- Kapitel 1 und Kapitel 2 bieten Ihnen eine Einführung in Docker und erläutern, was genau Docker eigentlich ist und wie Sie es verwenden können.
- Kapitel 3 führt Sie schrittweise durch die Installation von Docker.
- Die Kapitel 4 bis Kapitel 6 sind dem Docker-Client, Images und Containern gewidmet und untersuchen deren Aufgaben und Funktionsweisen.
- Kapitel 7 zeigt auf, wie Sie Images und Container debuggen können.
- Kapitel 8 führt in Docker Compose ein. Sie erfahren, wie signifikante Vereinfachungen in der Softwareentwicklung von komplexen containerbasierten Services mit Docker Compose möglich sind.
- Kapitel 9 behandelt Themen, die wichtig sind, um einen reibungslosen Übergang in die Produktivumgebung zu gewährleisten.
- Kapitel 10 demonstriert das Deployment von Containern in Public und Private Clouds in größerem Maßstab.
- In Kapitel 11 geht es um fortgeschrittene Themen, die einige Erfahrung mit Docker voraussetzen und von Bedeutung sind, wenn Sie anfangen, Docker in Ihrer Produktivumgebung einzusetzen.
- Kapitel 12 untersucht einige der grundlegenden Konzepte, die sich beim Design der nächsten Generation internetweit verfügbarer Produktivsoftware herausgebildet haben.

- Und das Kapitel 13 schnürt die maßgeblichen Inhalte dieses Buchs schließlich zu einem ansehnlichen, mit einer hübschen Schleife dekorierten Paket zusammen: Es enthält eine Zusammenfassung der verfügbaren Tools, die Ihnen dabei helfen sollen, das Deployment und die Skalierung von Softwarediensten zu verbessern.

Natürlich sind wir uns darüber im Klaren, dass kaum jemand technische Fachbücher von vorne bis hinten durchliest und Einleitungen nur allzu leicht übersprungen werden. Wenn Sie es allerdings schon mal bis hierher geschafft haben, finden Sie nachstehend noch einige Hinweise dazu, wie Sie bei der Lektüre des Buchs vorgehen sollten:

- Wenn Linux-Container Neuland für Sie sind, sollten Sie das Buch von Anfang an lesen. Die ersten beiden Kapitel erörtern die Grundlagen von Docker und Linux-Containern und beschreiben, was sie leisten, wie sie funktionieren und warum Sie all dem Beachtung schenken sollten.
- Falls Sie sofort loslegen und Docker auf Ihrem Rechner installieren und ausführen wollen, sollten Sie direkt zu Kapitel 3 und Kapitel 4 springen. Hier erfahren Sie, wie Docker installiert wird, wie Images erstellt oder heruntergeladen werden, wie Sie Container starten können und vieles mehr.
- Sind Sie mit den Grundlagen von Docker vertraut, benötigen aber dennoch Hilfe, um es in der Entwicklung zu nutzen, sollten Sie die Kapitel 5 bis Kapitel 8 lesen. Diese behandeln sehr viele Themen zum täglichen Einsatz von Docker mit Docker Compose, das Ihnen den Alltag erleichtert.
- Wenn Sie Docker bereits zur Entwicklung verwenden, aber Hilfe benötigen, um eine Produktivumgebung einzurichten, sollten Sie die Lektüre ab Kapitel 9 in Betracht ziehen, die sich mit dem Deployment und dem Debugging von Containern sowie weiteren fortgeschrittenen Themen befassen.
- Sie sind Software- oder Plattformarchitekt? Dann dürfte Sie Kapitel 12 interessieren, denn hier werden aktuell gängige Erwägungen zum Design containerisierter Anwendungen und horizontal skalierbarer Services betrachtet.

Konventionen dieses Buchs

In diesem Buch gelten folgende typografische Konventionen:

- Neue Begriffe, Dateinamen und Dateinamenserweiterungen sind *kursiv* gedruckt.
- URLs und E-Mail-Adressen sind im `Hyperlink-Format` dargestellt.
- Für Programm-Listings oder im Fließtext vorkommende Variablen- oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter wird eine `nicht-proportionale Schrift` verwendet.

- Texte, die vom Benutzer durch eigene Eingaben oder aus dem Kontext ersichtliche Werte ersetzt werden sollen, sind in KAPITÄLCHEN gedruckt.
- Vorschläge, Tipps, Hinweise und Warnungen sind in gesonderten Kästen angegeben.

Danksagungen

Wir möchten den vielen Menschen danken, die jede Auflage dieses Buchs überhaupt erst möglich gemacht haben:

- Nic Benders, Bjorn Freeman-Benson und Dana Lawson (New Relic), die unsere Bemühungen stets unterstützten und uns die nötige Zeit für die erste Auflage verschafften.
- Roland Tritsch und Nitro Software für die Unterstützung von Karl bei der Arbeit an der zweiten Auflage.
- Laurel Ruma (O'Reilly), die uns vorschlug, ein Buch über Docker zu schreiben, und Mike Loukides, der alles Notwendige arrangierte.
- Besonderer Dank gilt unserem Lektor Brian Anderson, der uns klargemacht hat, worauf wir uns einlassen, und uns bei jedem Schritt zur Seite stand.
- Nikki McDonald und Virginia Wilson, die uns durch den Prozess der zweiten Auflage führten.
- Eine neue Leserschaft an eine neue Technologie heranzuführen, benötigt besonderes Talent. Wir sind sehr dankbar, dass Lars Herrmann und Laura Frank Tacho sich die Zeit genommen haben, jeweils ein Vorwort zu schreiben.
- Den Lesern unseres Buchentwurfs, die gewährleisteten, dass wir beim Schreiben nicht vom richtigen Weg abkamen: Ksenia Burlachenko, die eine erste Bewertung und eine vollständige technische Rezension lieferte, sowie Andrew T. Baker, Sébastien Goasguen, Henri Gomez, Chelsey Frank und Rachid Zarouali.
- Besondere Erwähnung verdienen Alice Goldfuss und Tom Offermann, die uns detailliertes und durchweg nützliches Feedback gaben.
- Gillian McGarvey und Melanie Yarbrough für das Redigieren des Manuskripts, damit es so aussieht, als hätten wir in der Schule bei der Rechtschreibung und Zeichensetzung aufgepasst. 517 fehlende Kommata, und die Zählung läuft weiter ...
- Wendy Catalano und Ellen Troutman, die dafür gesorgt hat, dass alle Leser im Stichwortverzeichnis sinnvolle Einträge vorfinden.
- Unseren Kollegen bei New Relic, die uns beim Einsatz von Docker begleiteten und viele der Erfahrungen sammelten, von denen wir hier berichten.

- Grains of Wrath Brewery, World Cup Coffee, McMenamins Ringlers Pub, Old Town Pizza, A Beer at a Time!, Taylor‘s Three Rock pub und Weitere, die uns freundlicherweise ihre Tische und Strom zur Verfügung stellten, auch wenn unsere Speisen und Getränke schon längst verzehrt waren.
- Unseren Familien für ihre Unterstützung und dafür, dass sie uns die nötige Ruhe gewährten, wenn wir sie brauchten.
- Und schließlich allen, die uns ermutigten, uns Ratschläge gaben oder uns in anderer Weise irgendwie beim Schreiben dieses Buchs unterstützt haben.