Praxiswissen Docker

Grundlagen und Best Practices
für das Deployen von Software
mit Containern

DAS VORWORT

Vorwort

Container sind allgegenwärtig. Von der lokalen Entwicklung über Continuous Integration bis hin zum Managen produktiver Workloads im großen Maßstab sind überall Container im Spiel. Wie ist es dazu gekommen? Wohin wird die Reise noch gehen? Und was müssen Sie als Leser oder Leserin über diese Revolution wissen, die unsere Branche umgewälzt hat?

Viele ältere Technologien haben es versprochen: »Write once, run everywhere.« Aber nicht alle Runtimes haben diese Möglichkeit geboten, und selbst die, bei denen es möglich war, erforderten deren Verfügbarkeit (und die aller zusätzlichen Abhängigkeiten), damit eine Anwendung laufen konnte. Container versprechen nun: »Build once, run anywhere.« Sie erlauben es Ihnen, Ihre Anwendungen, die dafür erforderlichen Runtimes, Konfigurationsdateien und alle weiteren Dateiabhängigkeiten in einem Artefakt zu verpacken. Solange Sie eine Container-Runtime auf der Zielmaschine haben, funktioniert Ihre Anwendung einfach. Das erlaubt Ihrer Infrastruktur, tatsächlich agnostisch gegenüber den Anwendungen zu sein. »Also auf meinem Rechner läuft es« – das ist Vergangenheit!

Container bieten ein Standard-Application-Programming-Interface (eine API) zum Verwalten des Lebenszyklus eines Containers und der im Container verpackten Anwendungen. Diese API stellt eine homogene Schnittstelle zu einer ansonsten heterogenen Deployment-Landschaft bereit und befreit Operations-Teams davon, sämtliche Details und Besonderheiten beim Deployen und Ausführen von Anwendungen kennen zu müssen. Damit können sie sich darauf konzentrieren, was sie am besten können – Infrastruktur managen, Sicherheit und Compliance sicherstellen und den Laden am Laufen zu halten.

Diese Schnittstelle ist auch die Basis für einen ganzen Berg an Innovationen. Container-Orchestrierer wie Kubernetes oder Nomad nutzen diese Kontrollebene, um für mehr Abstraktion zu sorgen und es einfacher zu machen, containerisierte Workflows im großen Maßstab zu managen. Service-Mesh-Technologien wie Istio arbeiten Hand in Hand mit Orchestrierern und entkoppeln Aspekte wie Service Discovery und die Sicherheit vom Anwendungsstack.

All die Vorteile einer Standardschnittstelle machen sich auch weiter oben im Stack bemerkbar und erleichtern den Alltag in der Entwicklung. Ein einzelner Befehl kann eine ganze Entwicklungsumgebung erschaffen. Mit *Continuous Integration* (CI) lassen sich problemlos Container hochfahren, die Datenbanken, Queues oder sonstige Abhängigkeiten Ihrer Anwendung beheimaten, sodass mit dieser Integrations-, Smoke- oder End-to-End-Tests ausgeführt werden können und Ihre Arbeit kontrolliert werden kann. Und schließlich erlaubt die Portabilität der Container den Entwicklungsteams, für ihre Arbeit auch im Produktivumfeld Verantwortung zu übernehmen und damit viele Facetten von DevOps umzusetzen.

In einer Welt, in der Runtimes regelmäßig neue Major-Versionen erhalten, Teams sowie Organisationen polyglott und DevOps-Praktiken wie Blue-Green- oder Canary-Releases die Norm sind und in der die Größe von Umgebungen beispiellos ist, ist die Technologie, mit der Teams auf der ganzen Welt ihre Anwendungen bauen und deployen, die der Container. Container sind nicht mehr länger neu oder ungewöhnlich – sie sind die Regel, wenn Organisationen Anwendungen verpacken und deployen.

Aber die Arbeit mit Containern ist nicht einfach. Ich habe nun nahezu zehn Jahre lang Container genutzt, viel Zeit damit verbracht, Menschen auf der ganzen Welt davon zu erzählen, und kann guten Gewissens sagen: Dieses Thema besitzt viele Facetten.

Sean und Karl haben jahrelange Erfahrung in einen sehr gut lesbaren, aber trotzdem umfassenden Leitfaden für den Einsatz von Containern mit Docker umgesetzt. Was Sie für den Anfang wissen müssen, um Docker produktiv einzusetzen, finden Sie sämtlich in diesem Buch – von der Installation über das Verwenden und Bauen von Images, die Arbeit mit Containern, das Untersuchen von Builds und der Runtime bis hin zum produktiven Einsatz von Containern ist hier alles beschrieben.

Sean und Karl scheuen darüber hinaus auch nicht davor zurück, auf kleinste Details einzugehen. Sie zeigen beispielsweise, wie einfache Linux-Primitive wie `cgroups` und Namensräume dazu beitragen, die Magie von Containern möglich zu machen. Zudem wächst das Ökosystem von Docker immer weiter, und in diesem Buch finden Sie auch dieses beschrieben.

Im Vorwort der zweiten Auflage von *Praxiswissen Docker* hat Laura Tacho eine scharfsinnige Beobachtung gemacht: Cloud-native Technologien wie VMs und Container konkurrieren nicht miteinander, stattdessen ergänzen sie sich sogar. Diese Aussage ist heutzutage noch viel wahrer – das Aufkommen von Technologien wie Kata Containers (*https://katacontainers.io/*), die schlanke virtuelle Maschinen zum Ausführen von Containern nutzen und damit das Beste aus beiden Welten bieten (die Isoliertheit von VMs mit der Portabilität von Containern), sind ein Beweis für Lauras Kommentar.

Container sind allgegenwärtig. Auch eine lange Reise beginnt mit einem ersten Schritt – und tatsächlich ist die Reise, Container zu verstehen, eine lange. Wenn dieses Buch Ihr erster Schritt ist, haben Sie die richtige Wahl getroffen. Zwei sehr erfahrene Reiseleiter zeigen Ihnen den Weg, und auch wenn mir klar ist, dass Sie es nicht brauchen, wünsche ich Ihnen viel Glück.

Viel Spaß mit Containern.

– Raji Gandhi
Begründer von DefMacro Software, LLC, und Autor von
Head First Software Architecture, Head First Git und JavaScript Next
@looselytyped (https://twitter.com/looselytyped)
Columbus, Ohio
April 2023