Vorwort

»Irgendwann hat alles wieder Konjunktur« ist eine häufig zitierte Redewendung, die nicht nur auf die Modewelt und die Politik, sondern auch auf Technologien zutrifft. Vielmehr gilt dieses Statement ebenso für Linux-Container – und ich würde es sogar noch ein wenig weiter fassen: »Irgendwann hat alles wieder Konjunktur – und ist trotzdem abermals spannend.«

Container werden zwar schon seit vielen Jahren in diversen Linux-Distributionen zur Verfügung gestellt, sie wurden allerdings nur selten benutzt, weil ihre Realisierung in einer nutzbringenden Form eine recht hohe Komplexität erfordert. Traditionell hatte man bei der Erstellung von Linux-Containern ein festes, auf einen bestimmten Zweck ausgerichtetes Ziel vor Augen, was die Implementierung zusätzlicher Anforderungen wie Skalierbarkeit und Portabilität weiter erschwerte oder sogar unmöglich machte.

Doch dann kam Docker und leistete der Erschließung des Nutzwertes von Linux-Containern phänomenalen Vorschub, indem es ein standardisiertes Paketformat und einfache Verwendbarkeit vereinte. Vormals hochkomplexe und unverständliche Vorgänge wurden für Entwickler und Administratoren zu leicht einsetzbaren Ressourcen. In gewisser Weise hat Docker eine Renaissance für Linux-Container eingeläutet, indem es eine Welle des Interesses auslöste und neue Möglichkeiten eröffnete, die zu einer raschen Akzeptanz dieser Technologie führten. Docker hilft den Entwicklerteams, die mit Linux-Containern einhergehenden Vorteile zu erkennen wie die Portabilität von Anwendungen, vereinfachte Integration und optimierte Entwicklung, die bisher durch zu viel Komplexität verborgen waren.

Durch Docker haben sich Linux-Container zu einer wahrhaft umwälzenden Technologie gemausert, die das Zeug hat, die IT-Landschaft sowie die zugehörigen Ökosysteme und Märkte nachhaltig zu verändern. Infolge dieses Aufstiegs ist inzwischen eine Innovationswelle zu beobachten, die erkennen lässt, dass Linux-Container das Potenzial besitzen, das Deployment von Anwendungen für verschiedene Umgebungen und Plattformen durch den Einsatz eines breiten Spektrums technischer Fachkenntnisse drastisch zu verändern.

Innovation bedeutet nicht notwendigerweise die Einführung einer vollständig neuen, weltbewegenden Technologie. Der Erfolg von Docker beruht darauf, dass es – wie auch viele seiner Vorgänger – auf den Schultern von Giganten steht. Es baut auf der jahrelangen Fortentwicklung von Linux und den damit einhergehen-

den technischen Innovationen auf, die seine problemlose Verwendung gestatten. Die von Docker instrumentalisierten Linux-Fähigkeiten sind inzwischen so ausgereift, dass sie auch in anderen Betriebssystemen nachgebildet werden können – und damit ist es mittlerweile auch außerhalb von Linux einsetzbar.

Docker fördert und erleichtert ein radikales Umdenken aufseiten der Technologieexperten und gestattet eine neue Sichtweise darauf, welche Aspekte bei der Entwicklung und dem Deployment von Anwendungen sowie der Verwaltung der Infrastruktur als »Wetteinsätze« zu betrachten sind und welche technologische oder verfahrenstechnische Lösungen erfordern. In der Anfangsphase der Übernahme einer umwälzenden Technologie richtet sich der Blick typischerweise auf die unmittelbar anstehenden Probleme, wobei oft eine zu starke Vereinfachung erfolgt und relevante Aspekte übersehen werden. Aber Docker und Linux-Container besitzen ein weitaus größeres Potenzial, als einfach nur die Entwicklung von Anwendungen neu zu regeln – vielmehr wird mit ihrer Hilfe die Art der Anwendungen selbst neu definiert.

Der offenkundigste Effekt von Docker und dem damit einhergehenden einfacheren Gebrauch von Linux-Containern ist die Möglichkeit, die organisatorische Aufteilung von Geschäftsbetrieb, Anwendungsentwicklung und IT-Infrastrukturteam neu zu definieren. Docker stellt in gewisser Hinsicht eine handfeste Methode zur Einführung von »DevOps« dar: einem Zusammenschluss der oftmals in Konkurrenz stehenden Teams für die Anwendungsentwicklung und den IT-Betrieb (zumindest kommt es zu einer Art Waffenstillstand). Die Containerisierung modernisiert IT-Umgebungen, erlaubt auf organisatorischer Ebene die »richtige« Zuordnung der technischen Aufgabenstellungen und reduziert Übergabevorgänge sowie den damit einhergehenden Koordinierungsaufwand.

In seiner Eigenschaft als Paketformat für Anwendungen einerseits *und* als einheitliche Schnittstelle sowie methodische Vorgehensweise andererseits ermöglicht es Docker, dass die Zuständigkeit für das Container-Image inklusive aller Abhängigkeiten allein beim Entwicklerteam liegt, während das Administratorenteam weiterhin für die Infrastruktur verantwortlich ist. Nach der Einrichtung einer standardisierten Container-Infrastruktur kann sich die IT-Organisation voll und ganz auf die Verwaltung und das Deployment von Anwendungen, die Einhaltung der Sicherheitsrichtlinien, erforderliche Automatisierungen, Funktionalitäten und Kostenprofile konzentrieren – und all das, ohne die Fähigkeit aufzugeben, das Entwicklerteam für die Auswirkungen auf Sicherheit und Kosten verantwortlich zu machen, die der von ihnen verfasste und im Container enthaltene Code hat.

Zudem bringt Docker Vorteile hinsichtlich der Skalierbarkeit und der Geschwindigkeit, weil der Speicherbedarf von Anwendungen, die als Docker-Container vorliegen, auf das absolute Minimum sinkt – oftmals werden nur ein paar Dutzend oder wenige Hundert Megabyte benötigt. Vergleichen Sie das einmal mit den Images herkömmlicher virtueller Maschinen, die typischerweise mehrere Giga-

byte Speicherplatz belegen. Berücksichtigt man darüber hinaus auch noch die Geschwindigkeit, dann wird deutlich, dass es sich nicht nur um eine innovative, sondern um eine wahrhaft revolutionäre Technologie handelt.

Das Starten eines Containers bemisst sich in Millisekunden – ein ziemlicher Unterschied zu den Minuten, die die User von virtuellen Maschinen gewohnt sind. Das Deployment von Container-Images erfolgt schneller, wenn weniger Daten über Netzwerke oder andere Speicherstrukturen transportiert werden müssen, daher können moderne, flexible Anwendungen, deren Zustand sich häufig ändert und die Ressourcen dynamisch zuweisen, erheblich effizienter erstellt werden und Ressourcenanforderungen können in Echtzeit erfolgen.

Die vielleicht größte Innovation und bedeutendste Auswirkung, die Docker und Linux-Container mit sich bringen, ist die grundlegend andersartige Nutzung von Anwendungen. Die altbekannten monolithischen Anwendungs-Stacks können in Dutzende oder sogar Hunderte winziger zielgerichteter Dienste aufgeteilt werden, die so miteinander verwoben sind, dass sie dieselben Aufgaben wie herkömmliche Anwendungen erfüllen. Der Vorteil dieser Verfahrensweise ist, dass sich die betreffenden Bestandteile sehr viel effizienter umschreiben, wiederverwenden und verwalten lassen als monolithische Anwendungen. Auf diese Weise entsteht eine Anwendung, die vollständig aus Microservices zusammengesetzt ist.

Container sind im Bereich der Anwendungsentwicklung richtungsweisend, es ist jedoch von entscheidender Bedeutung, in Anbetracht der Neuerungen auch die vorhandenen Gegebenheiten nicht außer Acht zu lassen. Docker und Linux-Container bringen ihre eigenen Schwierigkeiten mit sich. Verwaltung, Sicherheit und Zertifizierung sind drei offensichtliche Herausforderungen, mit denen sich Unternehmen bei der Einführung von Containern konfrontiert sehen – und diese Aspekte unterscheiden sich gar nicht so sehr von denen herkömmlicher Anwendungen. Natürlich müssen die Container auf einem sicheren Host betrieben werden, wichtiger ist jedoch, dass die Sicherheit eines Containers von seinem Inhalt abhängt: Enthält er Schwachstellen, Malware oder ist er anfällig für bekannte Exploits? Eine digitale Signatur einer containerisierten Anwendung durch eine vertrauenswürdige Zertifizierungsstelle trägt viel dazu bei, diese Fragen zu beantworten.

Beim Einsatz von Docker und containerisierten Anwendungen ist eine vernünftige Verwaltung unverzichtbar. Das Potenzial für eine übermäßige Vermehrung von Containern ist exponentiell größer als bei virtuellen Maschinen, und schon allein die Verwaltung all dieser Container stellt eine Herausforderung dar. Ebenso wichtig wie die Gewährleistung der Sicherheit ist allerdings auch die Handhabung ihrer Inhalte: Wie werden Updates und Rollbacks durchgeführt? Was ist mit der Orchestrierung? Wie soll die »übermäßige« Vermehrung von Containern definiert werden? Sollen Container zurückgezogen oder archiviert werden, wenn sie sich auf einem alten Server in einer Art Schwebezustand befinden? Auch diese Fragen

müssen geklärt werden, bevor ein Unternehmen containerisierte Anwendungen nutzt, die das Etikett *betriebsnotwendig* tragen.

Sieht man von diesen Herausforderungen ab, stellen Linux-Container hinsichtlich der Erstellung, Nutzung und Handhabung von Anwendungen im Unternehmensumfeld einen fundamentalen Umbruch dar. Herkömmliche monolithische Anwendungen wird es auch weiterhin geben (immerhin haben viele alte Technologien bis heute überdauert), aber Container besitzen großes Potenzial, das althergebrachte Modell zu modernisieren und gestatten durch containerisierte Anwendungen sowohl in Rechenzentren als auch in der Hybrid Cloud enorme Flexibilität, Portabilität und Effizienz.

Sean und Karl arbeiten seit Jahren mit Containern (und mit Docker) und haben in diesem Buch alles Wissenswerte über Docker und die *Renaissance des Containers* für die IT-Welt zusammengetragen. Die Autoren geben einen soliden Überblick über die Funktionsweise von Docker in der Praxis und demonstrieren, wie Entwickler und andere IT-Experten Docker und Linux-Container so einsetzen können, dass es für sie und ihre Unternehmen größtmöglichen Sinn ergibt.

Lars Herrmann, General Manager
for Enterprise Linux, Enterprise
Virtualization and Container Strategy,
Red Hat