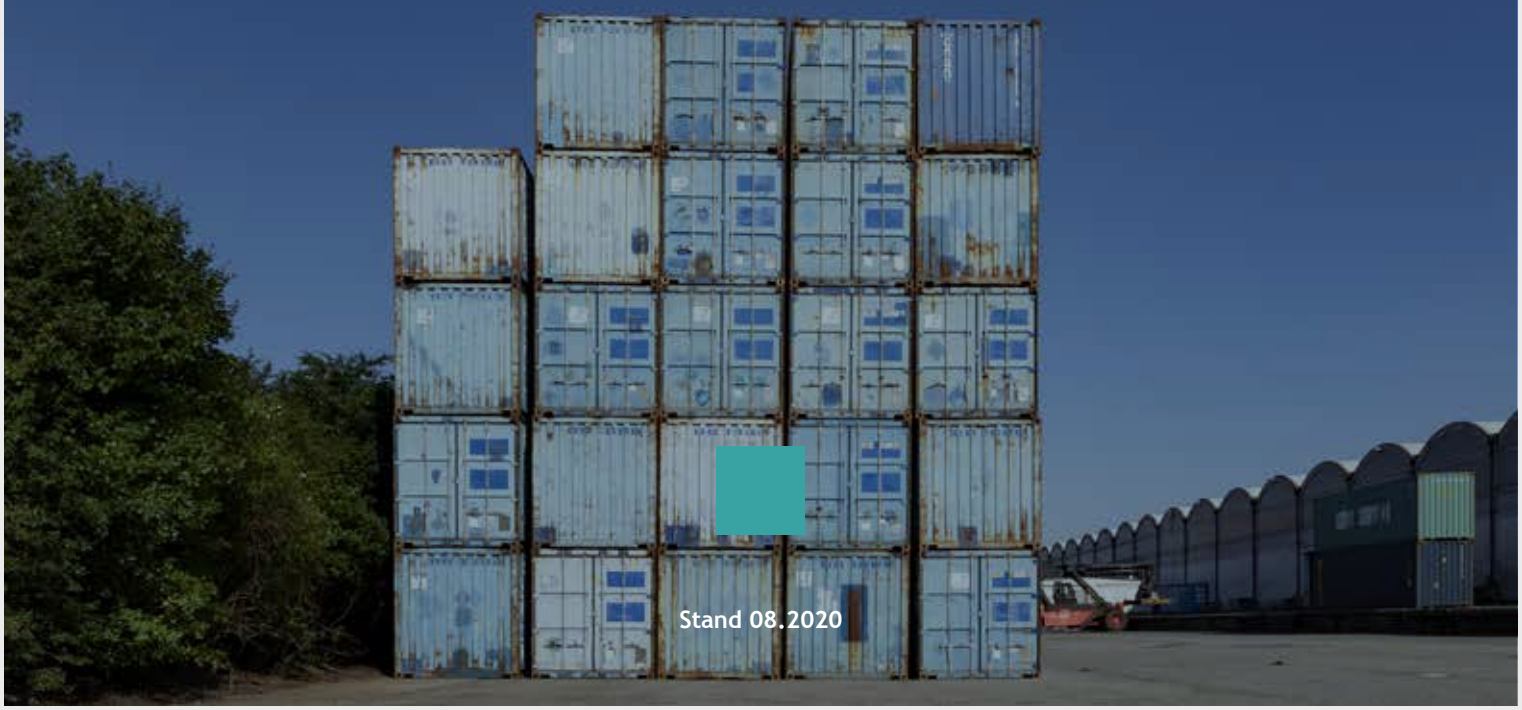
CON-TAINER VIRTUALI-SIERUNG

WORUM GEHT ES DABEI EIGENTLICH?



Stand 08.2020

VIRTUALI-SIERUNG IST IM TREND

Virtualisierung ist bereits seit einiger Zeit ein Trend beim Betrieb von Anwendungen. Neben dem bereits seit längerem bekannten Prinzip der „virtuellen Maschine“ existiert mit der Container-Virtualisierung ein weiterer Ansatz, um Anwendungen ressourcenschonend, aber gleichzeitig bequemer für Entwickler und Administratoren entwickeln und betreiben zu können.

Einordnung und Überblick

Zur Einordnung der Container-Virtualisierung zunächst ein Überblick über diese verschiedenen Ansätze:

Ohne Virtualisierung sieht die klassische (Hardware- und Software-)Architektur wie folgt aus: Im Rechenzentrum wird ein physikalischer Server aufgebaut, also ein „Rechner zum Anfassen“. Auf diesem Server wird dann ein Betriebssystem eingespielt und in diesem laufen dann die Anwendungen.

Beim Konzept von virtuellen Maschinen laufen nun mehrere logische Maschinen auf einem physikalischen Server. Jede dieser virtuellen Maschinen hat ein eigenes Betriebssystem. Auf dem physikalischen Rechner laufen somit mehrere Betriebssysteme unabhängig voneinander, die von unterschiedlichen Anwendern und/oder unterschiedlichen Anwendungen genutzt werden. Für die Anwender, denen vom Rechenzentrumsbetreiber eine solche virtuelle Maschine zur Verfügung gestellt wird, ist nicht ersichtlich, dass sie sich die darunterliegende Hardware mit anderen Anwendern teilen müssen. Diese Teilung der Hardware-Ressourcen ist Aufgabe der Administratoren des Rechenzentrums. Ein solches Modell ist sowohl unternehmensintern möglich („Jede Abteilung bekommt eine eigene virtuelle Maschine, es existiert aber nur ein physikalischer Server“) als auch in Cloud-Rechenzentren, in denen man sich als Kunde solche virtuellen Maschinen mieten kann, ohne zu wissen, welche parallel laufenden virtuellen Maschinen zusätzlich noch auf derselben Hardware laufen. Die Hardware wird hier also für die einzelnen Maschinen emuliert, während für jeden Nutzer ein eigenes Betriebssystem zur Verfügung steht.

Eine Ebene höher erfolgt die Virtualisierung nun bei einer Container-Infrastruktur: hier teilen sich mehrere (Container-)Anwendungen nicht nur die zugrundeliegende Hardware, sondern auch ein gemeinsames Betriebssystem. Alle Container müssen sich den einzigen installierten Betriebssystem-Kernel teilen und zwar im „Read-Only-Modus“. Darüberhinaus gibt es für jeden Container schreibbare Bereiche des Betriebssystems, sowohl für die Daten als auch für die Prozesse. Der gesamte Ressourcenverbrauch ist durch diese Konstellation im Vergleich zu virtuellen Maschinen deutlich geringer.

Was muss nun in einem solchen Container enthalten sein, damit die beschriebene Konstellation der gemeinsamen Betriebssystem-Nutzung funktioniert? Eine Anwendung, die als Container installiert werden soll, kann man sich als Paket vorstellen, in dem neben dem Anwendungscode auch alle benötigten Abhängigkeiten enthalten sind, also z.B. Systembibliotheken und Einstellungen. Das Paket muss also alles enthalten, was die Anwendung zum Laufen benötigt. In den anderen zuvor beschriebenen Architekturvarianten können diese Pakete kleiner sein, da Entwickler und Administratoren dort ja eine eigene – ggf. virtuelle – Maschine zur Verfügung hatten.

Container

Welche Vorteile haben nun diese Pakete, deren Zusammenstellung ja zunächst einen Mehraufwand bedeutet? Im Laufe ihres „Lebens“ muss eine Anwendung häufig umziehen: zunächst aus der Entwicklungsumgebung (z.B. dem Laptop des Entwicklers) in eine Testumgebung, von dort später dann in eine produktive Umgebung, möglicherweise in eine Public Cloud. Bei jedem Umzug muss genau darauf geachtet werden, dass die Umgebung, in der die Anwendung laufen soll, identisch bleibt. Liegt eine von der Anwendung benötigte externe Bibliothek beispielsweise in der produktiven Umgebung in einer anderen Version vor als zuvor in der Testumgebung, so kann dies unerwünschte und vor allem zuvor in den Tests unentdeckte Auswirkungen bzgl. Funktionalität, Stabilität oder Performance haben. Wird die Anwendung dagegen in Form eines Container-Paketes bereitgestellt und als solches auch in den verschiedenen Umgebungen betrieben, so sind alle Abhängigkeiten wie z.B. die benötigten Bibliotheken bereits im Paket enthalten, d.h. zu dem beschriebenen Szenario der unterschiedlichen Laufzeitumgebungen kann es nicht kommen. Diese Konstellation vereinfacht für Entwickler auch das Bereitstellen von Updates, das nun ebenfalls nur aus einem Update des Paketes besteht. Die Containertechnologie stellt also eine stabile Ablaufumgebung für Anwendungen zur Verfügung, die die Arbeiten und Abstimmungen an der Schnittstelle zwischen Entwicklung und Betrieb einer Anwendung besser automatisierbar und damit auch weniger fehleranfällig macht.
Viele Technologien und Architekturansätze sind nur so gut – bzw. nur so bekannt – wie die Tools, mit denen sie verwendet werden.

Docker und Kubernetes

Dies trifft auch auf die Container-Virtualisierung zu, die häufig sogar auf einer Ebene mit den Begriffen Docker und Kubernetes genannt wird. Man kann Container-Virtualisierung zwar auch ohne Docker und Kubernetes durchführen, trotzdem werden diese beiden Tools inzwischen so häufig eingesetzt, dass sich ein Blick darauf lohnt, wenn man sich Gedanken über die Einführung von Container-Virtualisierung macht.
Docker ist eine freie Software, die erstmals im Jahre 2013 veröffentlicht wurde. Technisch gesehen basiert sie auf Linux-Technologien, ist allerdings auch in anderen Umgebungen, insbesondere auch unter Windows einsetzbar. Zentrale Begriffe in der Docker-Welt sind das Docker-Image sowie die Docker-Runtime:

Wie oben bereits beschrieben, teilen sich bei der Container-Virtualisierung mehrere in Containern verpackte Anwendungen ein Betriebssystem. In der Docker-Terminologie tun sie dies, indem sie in der Docker-Runtime ausgeführt werden. Diese Docker-Runtime ist eine Laufzeitumgebung für die Container. Zur Verfügung gestellt werden die Applikationen dabei in einem Docker-Image, das man sich als eine Art Template für die Erstellung der Container vorstellen kann, die wiederum dann in der Docker-Runtime laufen.

Ist eine in einem Container ausgelieferte Software ausführlich getestet worden, wird sie „in einem Container verpackt“ an den Betrieb über-

geben. Dieser kann nun das Container-Paket installieren. Im laufenden Betrieb können nun diverse Administrationsaufgaben anstehen, z.B. eine Überwachung der Performance und – insbesondere bei steigenden Nutzerzahlen aufgrund des Erfolges der Anwendung – der Bedarf nach einer Skalierung nach dem Motto „Wir brauchen mehr Server und mehr Rechenleistung, um die wachsenden Zugriffszahlen bedienen zu können“. Auch Updates der Software müssen eingespielt werden, in der Regel mit der Erwartung, dass dies ohne Herunterfahren der Anwendung, also tatsächlich im laufenden Betrieb erfolgt.

Hier kommt nun mit Kubernetes eine Open-Source-Software ins Spiel, die im Jahr 2015 von Google entwickelt, inzwischen aber von einer breiten Community weiterentwickelt wird und sich zu einer Art Standard zur Orchestrierung von Containern entwickelt hat. Unter den Begriff Orchestrierung fallen dabei eine Vielzahl von (Administrations-)Tätigkeiten, mit denen man das Zusammenspiel einzelner Services bzw. Anwendungsbestandteile einrichten, überwachen und verbessern kann.

Container-Virtualisierung ist ein zwar noch vergleichsweise neuer, trotzdem aber bereits weit verbreiteter Ansatz, um sowohl Administratoren als auch Entwicklern die Arbeit bei der Erstellung und Verwaltung von Applikationen zu erleichtern. Mit Docker und Kubernetes stehen darüberhinaus Technologien zur Verfügung, mit denen das Konzept der Container-Virtualisierung in der Praxis sowohl testweise ausprobiert als auch produktiv verwendet werden kann.

ÜBER DEN AUTOR

Rudolf Jansen ist Diplom-Informatiker aus Aachen und arbeitet als freiberuflicher Softwareentwickler und Autor. Er schreibt über alle IT-Themen und unterstützt seine Kunden bei Projekteinsätzen als Business Analyst und Requirements Engineer.