WHITE PAPER

Einführung in das cloudnative Prinzip

Erste Überlegungen zur Definition und Entwicklung cloudnativer Funktionen

vmware®

Inhaltsverzeichnis

„Nach dem cloudnativen Prinzip werden Teams, Kulturen und Technologien strukturiert, um Komplexität mittels Automatisierung und Architektur zu bewältigen und die Geschwindigkeit zu erhöhen.“

JOE BEDA
PRINCIPAL ENGINEER
VMWARE

Definitionen

Es gibt keine prägnante Definition, die den Begriff „cloudnativ“ treffend umreißt. In der Tat sind weitere Begriffe und Vorstellungen im Umlauf, die sich teilweise ähneln. Grundsätzlich werden nach dem cloudnativen Prinzip jedoch Teams, Kulturen und Technologien strukturiert, um Komplexität mittels Automatisierung und Architektur zu bewältigen und die Geschwindigkeit zu erhöhen. Dieser operative Ansatz spiegelt sich im Sinnbild einer Gleichung wider, bei der beide Seiten (hier: Mitarbeiter und Infrastruktur) für eine Größenveränderung genutzt werden können.

Ein wichtiger Hinweis an dieser Stelle: Sie müssen sich nicht in die Cloud flüchten, um cloudnativ zu sein. Die hier beschriebenen Verfahren können je nach Bedarf schrittweise angewendet werden und dienen dem Ziel, den Übergang in die Cloud möglichst reibungslos zu gestalten.

Der eigentliche Mehrwert des cloudnativen Prinzips geht weit über die Ansammlung der damit verbundenen Technologien hinaus. Um die Entwicklungstrends in unserer Branche wirklich verstehen zu können, muss genau analysiert werden, inwiefern Unternehmen, Teams und Mitarbeitern zu größerem Erfolg verholfen werden kann.

Bis dato haben sich diese Verfahren insbesondere bei technologie- und zukunftsorientierten Unternehmen bewährt, die bereits umfangreiche Ressourcen in ihre Bestrebungen investiert haben. Denken Sie beispielsweise an Google, Netflix oder Facebook. Aber auch für kleinere, flexiblere Unternehmen ergibt sich hier ein Mehrwert. Andererseits wird dieses Konzept eher selten in Firmen eingesetzt, die nicht zu den Technologiepionieren zählen. Unter Berücksichtigung der gesamten IT-Branche befindet sich diese Entwicklung weiterhin in der Frühphase.

Da sich einige anfängliche Erfahrungen jedoch bereits bewährt haben, treten die folgenden Themenbereiche zunehmend in den Vordergrund:

- Effizientere und zufriedenere Teams – Mit cloudnativen Tools lassen sich umfangreiche Problemstellungen in kleinere Teilbereiche unterteilen, was zu spezialisierteren und agileren Teams führt.
- Weniger aufwendige Arbeitsformen – Die Umsetzung dieses Ziels erfolgt durch Automatisierung zahlreicher manueller Aufgaben, die betrieblichen Aufwand und Ausfallzeiten verursachen. Das setzt wiederum eine Infrastruktur voraus, die sich selbst reparieren und verwalten kann. Sie dürfen also durchaus höhere Erwartungen an die Systeme haben.
- Zuverlässigere Infrastruktur und Anwendungen – Durch Automatisierung lassen sich erwartete Änderungen besser handhaben, was bei unerwarteten Ereignissen und Ausfällen häufig zu einem besseren Umgang mit Fehlern führt. Wenn beispielsweise über einen einzigen Befehl oder Mausklick eine Anwendung für Entwicklungs-, Test- oder Produktionszwecke bereitgestellt werden kann, fällt eine automatisierte Bereitstellung in einem Disaster Recovery-Szenario deutlich leichter (ob automatisch oder manuell).
- Überprüfbar, sichtbar und debugfähig – Komplexe Anwendungen sind manchmal kaum nachvollziehbar. Die für cloudnative Anwendungen eingesetzten Tools machen das Innenleben einer Anwendung in der Regel zwangsläufig transparenter.
- Umfassende Sicherheit – Viele IT-Systeme haben heute eine harte, äußere Schale und einen weichen, zähflüssigen Kern. Moderne Systeme sollten sicher sein und standardmäßig zumindest die niedrigste Vertrauensstufe erfüllen. Dank des cloudnativen Ansatzes können Anwendungsentwickler bei der Erstellung sicherungsfähiger Anwendungen eine zentrale Rolle einnehmen.
- Effizientere Nutzung der Ressourcen – Durch automatisierte, cloudähnliche Verfahren zur Bereitstellung und Verwaltung von Anwendungen und Services ergibt sich die Chance einer algorithmischen Automatisierung. Ein Cluster-Orchestrierungstool kann etwa die Platzierung von Workloads in virtuellen Maschinen automatisieren, sodass Betriebsteams diese Platzierung nicht mehr anhand einer Tabelle vornehmen müssen.

In der Praxis

Wie in jedem Bereich, der von Innovation geprägt ist, sind auch in der cloudnativen Welt viele Veränderungen zu verzeichnen. Es ist nicht immer leicht zu erkennen, wie Ideen, die sich in der vorherigen Phase abgezeichnet haben, am besten umzusetzen sind. Darüber hinaus ist jedes bedeutende Projekt zu wichtig und zu groß, um es komplett neu zu schreiben. Stattdessen möchte ich Sie ermutigen, bei neuen Projekten oder neuen Teilbereichen eines bestehenden Projekts mit diesen neuen Strukturen zu experimentieren. Nehmen Sie sich einfach die entsprechende Zeit, neue Techniken und Erkenntnisse anzuwenden, wenn ältere Teile eines Systems verbessert werden. Suchen Sie nach Möglichkeiten, neue Funktionen oder Systeme separat als Mikroservices abzubilden.

„Ein SRE definiert sich dadurch, was um 10 UHR am nächsten Morgen passiert.“

JOE BEDA
PRINCIPAL ENGINEER
VMWARE

Es gibt hier keine festen Regeln. Jedes Unternehmen ist anders und die Methoden der Softwareentwicklung müssen so dimensioniert sein, dass sie für das verfügbare Team und Projekt passen. Denn: Die Landkarte entspricht nicht dem tatsächlichen Gelände. Manche Projekte sind für Experimente geeignet, wohingegen andere so wichtig sind, dass sie nur sehr vorsichtig angegangen werden dürfen. Es gibt jedoch auch Situationen, in denen Projekte genau in der Mitte zu verorten sind. Das heißt, die Techniken haben sich bewährt, müssen aber vor dem endgültigen Einsatz in kritischen Systemen formalisiert und in großem Maßstab getestet werden.

Das cloudnative Prinzip definiert sich durch bessere Tools und Systeme. Ohne diese Tools wird jeder neue Service in der Produktion hohe Betriebskosten verursachen. Dies ist eine Sache für sich, die überwacht, verfolgt, bereitgestellt usw. werden muss. Dieser Overhead ist einer der Hauptgründe dafür, dass die Dimensionierung von Mikroservices in geeigneter Weise erfolgen sollte. Den Vorteilen hinsichtlich der Entwicklungsgeschwindigkeit müssen daher die Mehrkosten für den zusätzlichen Produktionsbetrieb gegenübergestellt werden. In ähnlicher Weise sind mit der durchaus spannenden Einführung von neuen Technologien und Sprachen auch Kosten und Risiken verbunden, die es sorgfältig abzuwägen gilt.

Automatisierung ist der Schlüssel zu reduzierten Betriebskosten, die bei der Entwicklung und Bereitstellung neuer Services anfallen. Systeme wie Kubernetes, Container, Continuous Integration und Continuous Delivery (CI/CD) sowie deren Überwachung haben dasselbe übergeordnete Ziel. Sie sollen Anwendungsentwicklungs- und Operations-Teams zu größerer Effizienz verhelfen, damit diese schneller arbeiten und zuverlässigere Produkte entwickeln können.

Die jüngste Generation von Tools und Systemen ist besser dafür ausgelegt, das cloudnative Versprechen einzulösen. Im Vergleich zu älteren, herkömmlichen Konfigurationsmanagementtools können neue Tools das Problem so unterteilen, dass sich Teilbereiche problemlos verschiedenen Teams zuweisen lassen. Neuere Tools räumen den einzelnen Entwicklungs- und Operations-Teams mehr Besitzrechte ein und machen sie dank Self-Service-IT produktiver.

DevOps

Es dürfte am sinnvollsten sein, sich DevOps als einen kulturellen Wandel vorzustellen, durch den Entwickler schrittweise mehr Verantwortung dafür übernehmen, wie ihre Anwendungen in einer Produktionsumgebung ausgeführt werden. Darüber hinaus sind Mitglieder des Operations-Teams auf diesem Themengebiet sensibilisiert und wissen genau, wie die Anwendung funktioniert. So können sie aktiv dazu beitragen, die Anwendung gemeinsam zuverlässiger zu machen. Gegenseitiges Verständnis und größere Empathie zwischen diesen Teams ist der Schlüssel zum Erfolg.

Aber das ist noch nicht alles. Bei einer genauen Überprüfung der Anwendungsentwicklung und der Struktur des Operations-Teams wird deutlich, dass diese Beziehung weiter verbessert und vertieft werden kann. Google beschäftigt beispielsweise keine herkömmlichen Operations-Teams mehr. Stattdessen hat Google einen neuen Ingenieurtypus definiert, den sogenannten Site Reliability Engineer (SRE). Hierbei handelt es sich um hervorragend ausgebildete Techniker derselben Vergütungsstufe wie andere Techniker, die nicht nur einen Pager bei sich führen, sondern mit ihren speziellen Befugnissen eine entscheidende Rolle bei der Anwendungsverteilung einnehmen sollen, damit diese dank Automatisierung noch zuverlässiger werden.

Wenn der Pager nachts um 2 Uhr klingelt, macht jeder, der sich jetzt zurückmeldet, exakt dasselbe: Alle versuchen herauszufinden, was vor sich geht, damit sie schnell wieder ins Bett gehen können. Ein SRE definiert sich dadurch, was um 10 Uhr am nächsten Morgen passiert. Wollen sich die operativen Mitarbeiter nur beschweren oder arbeiten sie mit dem Entwicklungsteam so zusammen, dass eine Pager-Nachricht wie die aktuell vorliegende nie mehr verschickt werden muss? Die Anreize für den SRE und die Entwicklungsteams sind aufeinander abgestimmt, um das Produkt möglichst zuverlässig zu machen. Dieser Ansatz kann neben einer schonungslosen, nachträglichen Analyse zu erfolgreichen Projekten führen, die keine technischen Unzulänglichkeiten aufweisen.

SREs gehören bei Google zu den Mitarbeitern, die allerhöchste Wertschätzung erfahren. In der Tat werden Produkte oftmals ohne Rücksprache mit SREs in der Erwartung eingeführt, dass das Entwicklungsteam ihr Produkt in der Produktionsumgebung ausführen wird. Damit SREs in den Prozess eingebunden werden können, muss das Entwicklungsteam dem SRE-Team in vielen Fällen zunächst nachweisen, dass die Produktentwicklung abgeschlossen ist. Es wird davon ausgegangen, dass das Entwicklungsteam seine Hausarbeiten erledigt und etwa Überwachungs- und Benachrichtigungsabläufe, Leitfäden für Benachrichtigungen und Release-Prozesse eingerichtet hat. Das Entwicklungsteam sollte nachweisen können, dass die Pager-Nachrichten auf ein Minimum zurückgegangen und die meisten Probleme automatisiert sind.

Während das Operations-Team immer intensiver und anwendungsspezifischer eingebunden wird, macht es immer weniger Sinn, dass ein einzelnes Team alle Rechte für den gesamten Operations-Stack besitzt. Diese Einsicht führt zur Idee der Operations Specialization. In gewisser Weise ist dies der Gegenentwurf zu DevOps. Aber lassen Sie uns ganz unten anfangen:

- Hardwarebetrieb – Dieser Bereich ist bereits klar abtrennbar. In der Tat ist es einfach, Cloud-basierte Infrastructure as a Service (IaaS) als Ops as a Service auf Hardwarebasis zu betrachten.
- OS-Betrieb – Es muss jemanden geben, der sicherstellt, dass virtuelle Maschinen hochgefahren werden und der Kernel ordnungsgemäß ausgeführt wird. In der Trennung zwischen diesem Bereich und dem Management von Anwendungsabhängigkeiten spiegelt sich auch der Trend zu Distributionen mit minimalem OS wider, die auf das Container-Hosting ausgerichtet sind (wie etwa Project Photon OS™, CoreOS, Red Hat Project Atomic, Ubuntu Snappy, RancherOS und Google Container-Optimized OS).
- Cluster-Betrieb – In einer containerbasierten Welt wird der Computing-Cluster zu einer logischen Infrastrukturplattform. Das Cluster-System (Kubernetes) stellt zahlreiche Primitive zur Verfügung, mit deren Hilfe viele herkömmliche Betriebsaufgaben im Self-Service-Modus laufen können.
- Anwendungsbetrieb – Jede Anwendung kann jetzt bei gegebenenfalls ein eigenes Anwendungsteam haben. Wie oben erwähnt, kann und sollte das Entwicklerteam diese Rolle im Bedarfsfall einnehmen. Dieses Operations-Team setzt sich erwartungsgemäß gründlicher mit der Anwendung auseinander, aber sie müssen auf den anderen Ebenen keine Experten sein. So steht bei Google AdWords das Front-End-SRE-Team stärker mit dem Front-End-Entwicklungsteam im Austausch als mit dem Cluster-SRE-Team (das Borg-SRE-Team). Diese Abstimmung von Anreizen kann zu besseren Ergebnissen führen.

Je nach Bedarfslage des Unternehmens wird es wahrscheinlich Platz für andere spezialisierte SRE-Teams geben. Storage-Services könnten beispielsweise als separater Service mit eigenen SREs vom Gesamtsystem getrennt werden. Oder es ist möglicherweise ein Team dafür verantwortlich, ein Basis-Container-Image zu entwickeln und zu validieren, das grundsätzlich alle Teams nutzen sollten.

Container und Cluster

Container sind derzeit in aller Munde. An dieser Stelle ist es hilfreich, sich vor Augen zu führen, warum Container für so viel Begeisterung sorgen. Aus meiner Sicht gibt es hierfür drei Gründe:

1. Paketierung und Portabilität
2. Effizienz
3. Sicherheit

Im Folgenden werden diese Funktionen genauer erklärt.

Erstens bieten Container einen Paketierungsmechanismus. Damit kann ein System aufgebaut werden, das von der Bereitstellung solcher Systeme getrennt ist. Darüber hinaus lassen sich bereits entwickelte Artefakte/Images leichter umgebungsübergreifend (Entwicklung, Test, Staging, Produktion) portieren als herkömmliche Konzepte wie Images virtueller Maschinen (VM). Bereitstellungen werden allmählich immer stärker atomisiert. Herkömmliche Konfigurationsmanagementsysteme (Puppet, Chef, Salt, Ansible) können Systeme in einem nicht abgeschlossenen Konfigurationszustand belassen, der nur schwer zu debuggen ist. Es ist ebenso leicht möglich, dass eine andere Version auf Maschinen läuft als beabsichtigt, ohne dies zu bemerken.

Zweitens sind Container grundsätzlich schlanker als vollständige Systeme, was zu einer erhöhten Ressourcenauslastung führt. Das war der maßgebliche Anlass für Google, mit cgroups eine elementare, den Containern zugrunde liegende Kernel-Technologie einzuführen. Dank des gemeinsam genutzten Kernels und der nun deutlich flexibleren Mehrfachvergabe kann die Nutzung von Computing-Ressourcen durch Container leichter optimiert werden. Im Laufe der Zeit sind durchdachtere Methoden zu erwarten, um ein ausgewogeneres Gleichgewicht zwischen mehreren Containern innerhalb eines einzigen Hosts herzustellen und dabei Konflikte zwischen „Noisy Neighbors“ zu vermeiden.

Viele Anwender nehmen Container letztlich als Sicherheitsgrenze wahr. Obwohl Container sicherer sein können als einfache Unix-Prozesse, ist Vorsicht geboten, bevor sie wirklich als harte Sicherheitsgrenze angesehen werden können. Die Sicherheitsvorkehrungen von Namespaces unter Linux reichen möglicherweise für eine flexible Mandantenfähigkeit aus (hier sind Workloads nicht voll vertrauenswürdig), aber keinesfalls für eine vollständige Mandantenfähigkeit (mit aktiv antagonistischen Workloads).

„Cluster unterstützen uns bei lästigen Aufgaben im Operations-Bereich.“

JOE BEDA
PRINCIPAL ENGINEER
VMWARE

„Die Anwendungsarchitektur sollte auf praktische und organische Weise wachsen können.“

JOE BEDA
PRINCIPAL ENGINEER
VMWARE

In etlichen Bereichen sind derzeit Bestrebungen festzustellen, die die Grenze zwischen Containern und VMs immer mehr verschwinden lassen. Erste Ergebnisse zu Systemen wie Unikernels sind interessant. Aber der aktuelle Stand zeigt auch, dass sie erst nach Jahren die nötige Reife für den Einsatz in der Produktion haben werden.

Obwohl die oben genannten Ziele mithilfe von Containern leicht erreicht werden können, sind sie nicht zwingend erforderlich. Netflix setzt beispielsweise seit jeher auf einen sehr modernen Stack, in dem VM-Images auf ähnliche Weise paketiert und genutzt werden, wie Container bei anderen Anbietern.

Der ursprüngliche Entwicklungsimpuls galt vor allem Containern und zielte darauf ab, die Software über einen einzigen Knoten zuverlässiger und vorhersehbarer zu verwalten. Im nächsten Schritt konzentriert sich diese Entwicklung auf Cluster (oftmals auch Orchestrierungstools genannt). Durch die Anbindung mehrerer Knoten an automatisierte Systeme entsteht ein neuer Self-Service-Satz logischer Infrastruktur für Entwicklungs- und Operations-Teams.

Mithilfe eines Container-Clusters wird Computern die Aufgabe überlassen, herauszufinden, welcher Workload an welche virtuelle Maschine weitergereicht werden soll. Bei einem Hardwareausfall mitten in der Nacht können Cluster zudem Probleme automatisch im Hintergrund beheben – und es muss niemand per Pager-Nachricht geweckt werden.

In einer allerersten Aktion aktivieren Cluster die Funktion Operations Specialization (wie oben beschrieben), mit der sich der Anwendungsbetrieb erfolgreich als eigenständige Disziplin ausführen lässt. Dank der klar definierten Cluster-Schnittstelle können sich Anwendungsteams auf die Lösung von Problemen konzentrieren, die sich unmittelbar auf die Anwendung auswirken.

Der zweite Vorteil von Clustern ist, dass sie zusätzliche Services starten und verwalten können. Auf diese Weise entstehen neue Architekturen, mit deren Hilfe Entwicklungsteams Geschwindigkeitsreserven erschließen können. Dies geschieht mittels Mikroservices, auf die im nächsten Abschnitt näher eingegangen wird.

Mikroservices

Mikroservice ist ein neuer Begriff für ein altes Konzept, das bereits seit geraumer Zeit existiert. Grundsätzlich geht es hier um ein Verfahren, das eine große Anwendung in kleinere Teilstücke aufteilt, die unabhängig voneinander entwickelt und verwaltet werden können. Nachfolgend sind die wichtigsten Aspekte aufgeführt:

- Stabile und eindeutige Schnittstellen – Das enge Verkoppeln von Services muss vermieden werden. Dokumentierte und versionierte Schnittstellen tragen unterstützend dazu bei, dieses Vorhaben zu realisieren und ein gewisses Maß an Freiheit zu gewährleisten, die Konsumenten und Entwicklern dieser Services gleichermaßen vorbehalten ist.
- Unabhängiges Bereitstellen und Verwalten – Ein einzelner Mikroservice sollte aktualisiert werden können, ohne ihn mit allen anderen Services synchronisieren zu müssen. Es ist ferner wünschenswert, ein einfaches Rollback auf eine frühere Version des Mikroservice ausführen zu können. Das heißt, dass die bereitgestellten Binärdateien auf- und abwärtskompatibel sein müssen, was die API und jedes Datenschema betrifft. Darüber hinaus können so die Kooperations- und Kommunikationsmechanismen zwischen den entsprechenden Betriebs- und Entwicklerteams getestet werden.
- Integrierte Stabilität – Mikroservices sollten so aufgebaut und getestet werden, dass sie unabhängig voneinander stabil laufen. Code, der auf einen Service zugreift, sollte auch weiterhin funktionieren und zu sinnvollen Ergebnissen führen, wenn der genutzte Service ausfällt oder fehlerhaft ist. Jeder Service sollte bei unvorhersehbaren Lasten und fehlerhaften Eingaben ein ähnliches Abwehrverhalten aufweisen.

Mikroservices sind unter Umständen schwierig zu dimensionieren. Daher sollten Services vermieden werden, die zu kleinteilig sind (Pico-Services). Stattdessen ist es vorzuziehen, Services über ihre natürliche Grenzen (Sprachen, asynchrone Warteschleifen, Skalierungsanforderungen) hinweg zu splitten und angemessene Teamgrößen festzulegen (z.B. zwei Pizza-Teams).

Statt mit 20 Services sollten Sie mit zwei bis drei Services starten und sie splitten, sobald die Komplexität in diesem Bereich zunimmt. Oftmals wird die Architektur einer Anwendung solange nicht wirklich gut verstanden, bis die Entwicklung bereits weit fortgeschritten ist. Diese Beobachtung bestätigt auch die Einsicht, dass Anwendungen nur selten „fertig“ sind. Vielmehr befinden Sie sich dauerhaft in der Entwicklung.

„Sobald ein potenziell anfälliges Image vorhanden ist, wird ein technisches Problem zu einem Prozess- oder Workflow-Problem.“

JOE BEDA
PRINCIPAL ENGINEER
VMWARE

MIKROSEGMENTIERUNG FÜR CONTAINER

Durch Mikrosegmentierung wird ein Rechenzentrum inklusive aller dazugehörigen Workloads mittels Netzwerkvirtualisierung in logische Segmente unterteilt, die jeweils einen einzelnen Workload enthalten. Dadurch können Sicherheitskontrollen auf die einzelnen Segmente angewendet werden, um zu verhindern, dass Angreifer auch auf andere Segmente oder Workloads zugreifen.

Sind Mikroservices ein neues Konzept? Nicht unbedingt. Es ist eigentlich ein weiterer Typus der Software-Komponentisierung. Code wurde schon immer in Bibliotheken aufgeteilt. Hier wird lediglich der Linker verschoben, denn das Erstellungszeitkonzept ist nun ein Laufzeitkonzept. Vor einigen Jahren war im Bereich SOA eine sehr ähnliche Entwicklung zu beobachten, allerdings ohne XML. Aus einer anderen Perspektive betrachtet war die Datenbank fast schon immer ein Mikroservice. In diesem Kontext wurde sie häufig so implementiert und bereitgestellt, dass vorgenannte Aspekte erfüllt wurden.

Einschränkungen können die Produktivität erhöhen. Auch wenn es verlockend erscheint, jedem Team die Auswahl unterschiedlicher Sprachen und Frameworks für jeden Mikroservice zu gestatten, sollten Sie stattdessen eine Standardisierung auf wenige Sprachen und Frameworks in Betracht ziehen. Diese Vorgehensweise wird den Wissenstransfer und die Mobilität innerhalb des Unternehmens verbessern. Bleiben Sie jedoch aufgeschlossen und weichen Sie im Bedarfsfall von dieser Richtlinie ab. Diese Offenheit ist ein wesentlicher Vorteil dieser Umgebung gegenüber der eher vertikal integrierten und strukturierten Platform as a Service (PaaS). Anders ausgedrückt: Die Frage der Einschränkung ist eher richtlinienorientiert als funktional zu beantworten.

Obwohl die meisten in Mikroservices ein Verfahren zur Implementierung einer umfangreichen Anwendung sehen, gibt es weitere Services, die Teil des Servicespektrums sind:

1. Service als Implementierungsdetail – Wie oben bereits beschrieben, ist diese Vorgehensweise sinnvoll, um ein großes Anwendungsteam in kleinere Teams zu unterteilen, die von der Entwicklung bis zum Betrieb alle Bereiche abdecken.
2. Gemeinsames Artefakt, eigene Instanz – In diesem Szenario wird der Entwicklungsprozess über viele Service-Instanzen hinweg gemeinsam genutzt. So kann es ein einzelnes Entwicklungsteam und viele Betriebsteams geben oder auch ein einheitliches Betriebsteam, das über dedizierte Instanzen hinweg zusammenarbeitet. Viele Datenbanken fallen in diese Kategorie, denn hier lassen viele Teams eigene Instanzen einer einzigen MySQL-Binärdatei ausführen.
3. Gemeinsam genutzte Instanz – In diesem Fall stellt ein einziges Team einen gemeinsam genutzten Service für viele Anwendungen und Teams innerhalb eines Unternehmens bereit. Der Service kann Daten und Aktionen pro Anwender (mandantenfähig) partitionieren oder einen einzigen, einfachen Service bieten, der umfassend genutzt wird (HTML-basierte Benutzeroberfläche für eine gemeinsame Markenleiste, Modelle für maschinelles Lernen usw.).
4. Big-S-Service – Die meisten Unternehmen werden einen solchen Service nicht entwickeln, aber möglicherweise nutzen wollen. Hierbei handelt es sich um einen typischen vollständig mandantenfähigen Service, der entwickelt wurde, um eine große Anzahl unterschiedlicher Kunden abzudecken. Diese Art von Service erfordert ein Maß an Buchhaltung und Sicherheitshärtung, das innerhalb eines Unternehmens oftmals nicht notwendig ist. SendGrid oder Twilio würden beispielsweise in diese Kategorie fallen.

Während sich Services von einem Implementierungsdetail zu einer gemeinsamen Infrastruktur innerhalb eines Unternehmens entwickeln, verwandelt sich das Servicenetzwerk von einem anwendungsbasierten Konzept zu einem Gebilde, das das gesamte Unternehmen umfasst. Durch diese Formen von Abhängigkeiten ergeben sich Chancen und Risiken zugleich.

Sicherheit

Die Sicherheitsfrage ist weiterhin ein großes Thema in der cloudnativen Welt. Alte Verfahren lassen sich nicht sauber anwenden, weshalb das cloudnative Prinzip zunächst als Rückschritt erscheinen mag. Aber diese neue Welt bietet auch Chancen.

Sicherheit von Container-Images

Es gibt recht viele Tools, die Anwender beim Audit ihrer Container-Images unterstützen und sicherstellen, dass diese vollständig gepatcht sind. Ich habe keine wirklich fundierte Meinung zu den verfügbaren Optionen.

Das eigentliche Problem ist nämlich folgendes: Wie verfahren Sie, wenn Sie ein potenziell anfälliges Container-Image finden? Das ist die Frage, für die der Markt bislang keine ausreichende Antwort in Form guter Lösungen gefunden hat. Sie möchten möglicherweise zuerst herausfinden, welche Gruppen innerhalb Ihres Unternehmens betroffen sind, wo in Ihrer Container-Image-Struktur das Problem zu beheben ist und wie eine neue gepatchte Version am besten getestet und verteilt werden kann.

CI/CD ist ein wichtiger Bestandteil dieses Puzzles, weil es automatisierte und schnelle Freigabeprozesse für die neuen Images gewährleistet. Darüber hinaus können Sie dank der

WEITERE INFORMATIONEN ZU CLOUDNATIVER TECHNOLOGIE VON VMWARE

Weitere Informationen dazu, wie Sie mithilfe von VMware cloudnative Anwendungen erstellen, ausführen und verwalten, finden Sie unter

https://cloud.VMware.com/de

LEBENSLAUF DES AUTORS

Joe Beda ist Principal Engineer bei VMware. Zudem war er Mitgründer von Heptio und dort als CTO tätig. Während seiner Tätigkeit bei Google trat er als Mitinitiator der Google Compute Engine in Erscheinung und rief das allererste Kubernetes-Projekt ins Leben. Er ist ein überzeugter Verfechter von Open Source-Software und verfasst regelmäßig Fachbeiträge für diese Community.

Integration in Orchestrierungssysteme Anwender ausfindig machen, die potenziell anfällige Images nutzen. Darüber hinaus lässt sich verifizieren, ob eine neue korrigierte Version tatsächlich in der Produktionsumgebung ausgeführt wird. Abschließend ist eine Richtlinie in Ihrem Bereitstellungssystem hilfreich, um zu verhindern, dass neue Container mit einem bekanntermaßen fehlerhaften Image gestartet werden (in Kubernetes wird diese Richtlinie als Zulassung bezeichnet).

Mikroservice- und Netzwerksicherheit

Aber selbst wenn sämtliche ausgeführten Komponenten in Ihrem Cluster gepatcht sind, ist immer noch nicht sichergestellt, dass alle Aktivitäten in Ihrem Netzwerk auch vertrauenswürdig sind.

Herkömmliche netzwerkbasierte Sicherheitstools funktionieren in einer kurzlebigen Containerwelt mit dynamischer Zeitplanung nicht sonderlich gut. Kurzlebige Container werden möglicherweise gar nicht erst von herkömmlichen Überprüfungstools gescannt, weil ihre Laufzeit so gering ist. Denn zum Zeitpunkt der Berichterstellung könnte die Laufzeit des betreffenden Containers schon längst wieder abgelaufen sein.

Bei dynamischen Orchestrierungstools haben IP-Adressen keine langfristige Bedeutung und können automatisch wiederverwendet werden. Die Lösung liegt darin, Netzwerkanalysetools so in das Orchestrierungstool zu integrieren, dass neben reinen IP-Adressen auch logische Namen (und andere Metadaten) verwendet werden können. Dieses Vorgehen dürfte Benachrichtigungen leichter umsetzbar machen.

Viele Networking-Technologien nutzen Kapselung, um eine IP-Adresse pro Container zu implementieren. Das kann bei netzwerkbezogenen Ablaufverfolgungs- und Prüfungstools zu Problemen führen. Diese müssen angepasst werden, falls derartige Networking-Systeme in Produktionsumgebungen bereitgestellt werden. Glücklicherweise wurden viele dieser Aspekte in VXLANs und VLANS standardisiert oder es gibt keine Kapselung, sodass sie über viele dieser Systeme hinweg genutzt werden können.

Jedoch sind aus meiner Sicht die größten Probleme im Umfeld der Mikroservices auszumachen. Wenn viele Services in der Produktionsumgebung laufen, muss sichergestellt sein, dass nur autorisierte Clients bestimmte Services aufrufen dürfen. Darüber hinaus müssen Clients im Falle einer Wiederverwendung von IP-Adressen erkennen können, dass sie den richtigen Service aufrufen. Es gibt zwei (sich gegenseitig nicht ausschließende) Ansätze, um dieses Problem anzugehen.

Beim ersten Ansatz können die flexibleren Networking-Systeme Firewall-Regeln auf Hostebene (außerhalb aller Container) implementieren und so detaillierte Zugriffsrichtlinien für Container aktivieren, um weitere Container aufzurufen. Ich bezeichne diesen Ansatz als netzwerkbezogene Mikrosegmentierung. Die Herausforderung liegt hier darin, solche Richtlinien unter Berücksichtigung der dynamischen Zeitplanung zu konfigurieren. Auch wenn sich diese Entwicklung noch in der Anfangsphase befindet, gibt es bereits mehrere Unternehmen, die an einer einfacheren Lösung arbeiten. Unterstützung im Netzwerk, Koordination mit dem Orchestrierungstool und umfangreichere Anwendungsdefinitionen stehen dabei im Vordergrund. Dennoch gibt es einen großen Haken: Mikrosegmentierung ist umso ineffektiver, je häufiger ein bestimmter Service verwendet wird. Wenn ein Service von unzähligen Anwendern aufgerufen wird, greifen einfache zugangsbedingte Autorisierungsmodelle nicht mehr.

Im zweiten Ansatz spielen Anwendungen bei der Implementierung von Authentifizierungs- und Verschlüsselungsverfahren im Rechenzentrum eine wichtigere Rolle. Diese Vorgehensweise funktioniert, weil Services viele Clients einbinden und somit für eine flexible Mandantenfähigkeit innerhalb eines großen Unternehmens geeignet sind. Das setzt voraus, dass in der Produktionsumgebung ein System zur Identifizierung von Services vorhanden ist. Vor diesem Hintergrund habe ich ein Nebenprojekt namens SPIFFE (Secure Production Identity Framework For Everyone) gestartet. Diese Ideen haben sich bei Unternehmen wie Google nachweislich bewährt.

Sicherheit ist ein sehr umfassendes Thema – und ich bin mir sicher, dass hier keineswegs alle Bedrohungen und Überlegungen angeführt werden. Daher muss dieses Thema weiterhin diskutiert werden.

Hier konnte nur eine erste, kurze Einführung in das cloudnative Prinzip stattfinden. Wenn Sie daran interessiert sind, diesen Austausch fortzusetzen, können Sie sich gerne jederzeit an uns wenden.

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com Zweigniederlassung Deutschland
Willy-Brandt-Platz 2 81829 München Tel.: +49 89 370 617 000 Fax: +49 89 370 617 333 www.vmware.com/de