THENEWSTACK

Lagebericht zum Kubernetes-Ökosystem

ZWEITE AUSGABE

The New Stack

Lagebericht zum Kubernetes-Ökosystem, Zweite Ausgabe

Alex Williams, Gründer und Verleger

E-Book-Team:

Gabriel H. Dinh, Produktionsleiter

Janakiram MSV (Janakiram & Associates), Autor

Jonathan Gold (Container Solutions), Technischer Redakteur

Judy Williams, Redakteurin

Lawrence Hecht, Beitragender Forschungsanalyst

Libby Clark, Redaktions- und Marketingleiterin

Richard MacManus, E-Book-Redakteur

Sebastien Goasguen (TriggerMesh), Technischer Redakteur

Unterstützungsteam:

B. Cameron Gain, EU-Korrespondent

Benjamin Ball, Leiter Vertriebs- und Kundenmanagement

Colleen Coll, Leiterin digitale Medien und Marketing

Eddie Rogers, Produzent für digitale Medien

Jennifer Riggins, UK-Korrespondentin

Joab Jackson, Leitender Redakteur

Michelle Maher, Redaktionsassistentin

20210205

THENEWSTACK

Inhaltsverzeichnis

Sponsoren und Partner

Wir bedanken uns für die Unterstützung unserer E-Book-Sponsoren:

DataStax ist das Unternehmen hinter der gleichnamigen, massiv skalierbaren, hochverfügbaren, cloudnativen NoSQL-Datenplattform, die auf Apache Cassandra basiert. Mit DataStax können Benutzer und Unternehmen ihre Daten in jeder Cloud nutzen, überall auf der Welt und bei jedem Anbieter – und das jederzeit, ohne Unterbrechungen.

Dynatrace ist der führende Anbieter im Bereich Software Intelligence. Der speziell für die Enterprise-Cloud entwickelte vollständige Stack ist die einzige KI-gestützte, umfassend automatisierte Intelligence-Plattform, die tiefgreifende Einblicke in dynamische, webumfassende Hybrid-Cloud-Ökosysteme liefert. Aus diesem Grund vertrauen die weltweit führenden Marken auf Dynatrace, um ihren Benutzern eine perfekte Anwendungserfahrung zu bieten.

Die Softwarebereitstellungsplattform Harness automatisiert den gesamten CI/CD-Prozess und ermöglicht den Ingenieuren so das Entwickeln, Testen, Bereitstellen und Prüfen ihrer Software mit der Sicherheit und Geschwindigkeit der Enterprise-Klasse. Harness nutzt maschinelles Lernen, um Sie zu schützen, wenn einzelne Umgebungen ausfallen. Die Softwarebereitstellung war noch nie so einfach.

Hewlett Packard Enterprise

HPE Ezmeral bringt Projekte zur digitalen Transformation durch die Verlagerung von Zeit und Ressourcen vom IT-Betrieb zur Entwicklung von Innovationen voran. Modernisieren und schützen Sie Ihre Apps, vereinfachen Sie Ihren Betrieb und nutzen Sie Daten, um Erkenntnisse in effektive Maßnahmen umzuwandeln.

THENEWSTACK

KubeCon | CloudNativeCon

Auf den Konferenzen KubeCon + CloudNativeCon treffen sich Nutzer und Technologen zum Wissensaustausch über cloudnatives Computing und dessen Weiterentwicklung. Die anbieterneutralen Veranstaltungen sind ein Treffpunkt für Fachgebietsexperten und wichtige Akteure hinter großen Projekten wie Kubernetes, Prometheus, Envoy, CoreDNS, containerd und anderen. KubeCon + CloudNativeCon wird von der Cloud Native Computing Foundation (CNCF) organisiert.

PRISMA™
BY PALO ALTO NETWORKS

Granulare Zugriffskontrollen und umfassender Schutz für Daten, Apps, Hosts, Container und serverlose Technologien: Prisma, eine der umfassendsten Cloud-Sicherheitssuites der Branche, bietet Transparenz und konsistente Sicherheit. Die perfekte Plattform für Ihren Erfolg.

Und unsere Partner:

Container Solutions

Container Solutions ist ein auf die cloudnative Transformation spezialisierter Anbieter von Professional Services. Wir betrachten das Gesamtbild von Technologie, Unternehmenskultur und Strategie und sorgen für eine reibungslose Transformation. Container Solutions hat Büros in Amsterdam, London, Berlin, Montreal, Prag, Warschau und Zürich.

TRIGGERMESH
CLOUD NATIVE INTEGRATION PLATFORM

TriggerMesh bietet eine cloudnative Plattform für die Echtzeitintegration von Services zur Automatisierung von Workflows und Beschleunigung des Informationsflusses im Unternehmen. Mit TriggerMesh können Entwickler in Unternehmen ereignisgesteuerte Anwendungen erstellen und die Services von verschiedenen Cloud-Anbietern und On-Premises-Systemen nutzen.

Danksagung

Janakiram MSV ist der Verfasser dieses E-Books, Principal Analyst bei Janakiram & Associates und Lehrbeauftragter am International Institute of Information Technology in Hyderabad. Außerdem ist er ein Google Qualified Cloud Developer, Amazon Certified Solution Architect, Amazon Certified Developer, Amazon Certified SysOps Administrator und Microsoft Certified Azure Professional. Janakiram MSV engagiert sich als Botschafter für die Cloud Native Computing Foundation und war einer der ersten Certified Kubernetes Administrators und Certified Kubernetes Application Developers. Stationen seiner Karriere waren u. a. Microsoft, AWS, Gigaom Research und Alcatel-Lucent.

Lawrence Hecht, der zu diesem E-Book beitragende Forschungsanalyst, verfasst seit 17 Jahren Forschungsberichte über IT-Märkte. Zuvor leitete er für 451 Research und TheInfoPro Umfragen zur „Stimme des Kunden“ zu B2B-Märkten für Unternehmens-IT wie Cloud-Computing, Datenanalyse und Informationssicherheit. Im Jahr 1999 gründete er das Internet Public Policy Network (IPPN), ein Netzwerk von Fachexperten, das Auftragsforschung, Whitepaper und Beratung über öffentlich-politische Themen mit Technikbezug bereitstellte. Zudem unterstützt Hecht Autoren beim Finden von Daten in primärer und sekundärer Forschung, aus denen sich praxisrelevante Erkenntnisse ableiten lassen. Zurzeit nutzt er diese Fähigkeiten, um Inhalte für Vordenker und zur Überzeugung der Fachwelt zu erstellen. Hecht erwarb einen Bachelor of Arts an der Rutgers University und einen Master of Public Policy an der Georgetown University.

Einleitung

Vor gerade einmal fünf Jahren, im Juni 2015, hatten viele Schwierigkeiten mit dem Namen und der Idee hinter Kubernetes, einem Projekt, das sich in der ersten Jahreshälfte aus Google entwickelt hatte. „Kuber..., Kuber... was?“

Der Name Kubernetes ist von dem altgriechischen Begriff für das Ruder eines Schiffes abgeleitet. Diese neue Technologiearchitektur sollte zu einem wichtigen Aspekt eines Umbruchs werden, der sich bereits einige Jahre zuvor mit der Entstehung von Unternehmen in der Größenordnung des Internets abgezeichnet hatte. Docker war immer noch beliebt, aber die Community begann sich mit Ideen rund um die Containerorchestrierung zu beschäftigen. Im Juni 2015 wurde die Gründung einer neuen Organisation, der Cloud Native Computing Foundation (CNCF), bekannt gegeben, mit der Kubernetes ein neues Zuhause bekam.

Horizontal skalierbare Architekturen waren entstanden. Docker war das erste Open-Source-Megaprojekt. Kubernetes wurde zu einem richtungsweisenden Projekt. Über die nächsten Jahre entwickelte es sich zu einer Bewegung, in der verteiltes Computing, Open-Source-Communitys und neue Anwendungsarchitekturen sich gegenseitig bereicherten. Das Ergebnis war eines der bedeutendsten Open-Source-Projekte der letzten 20 Jahre. Manche sahen darin sogar die Cloud-Version von Linux.

Um die Bewegung zu dokumentieren, veröffentlichte The New Stack im Jahr 2017 das erste E-Book über die Kubernetes-Community und ihr cloudnatives Ökosystem.

Seit 2018 hat sich die Community so stark verändert und so rasch weiterentwickelt, dass die erste Ausgabe inzwischen stellenweise veraltet ist. Eine Aktualisierung war notwendig und musste aus einem anderen Blickwinkel erfolgen, um der heutigen Kubernetes-Community und künftigen Entwicklungen gerecht zu werden.

Einer Untersuchung aus dem Jahr 2019 zufolge nutzen inzwischen nahezu 80 Prozent der vom CNCF befragten Unternehmen Kubernetes in der Produktion.

Die Verbreitung von Kubernetes und cloudnativen Technologien spiegelt sich in der Beliebtheit

der KubeCon + CloudNativeCon wider, deren Besucherzahlen in der North America Show von etwa 1.000 Personen im Jahr 2016 auf 4.000 Teilnehmer im Jahr 2017 anstiegen. Im Herbst 2019 überstieg die Teilnehmerzahl der KubeCon in San Diego die Marke von 10.000. 2020 wird die KubeCon als virtuelle Konferenz stattfinden, an der möglicherweise noch weitaus mehr Personen teilnehmen werden.

Zur Zeit der Veröffentlichung zwingt uns COVID-19, zu Hause zu bleiben, und beschränkt die Open-Source-Teilnahme auf das Virtuelle. Aber das Wachstum von Kubernetes hält an – und so auch die Konsolidierung.

Die Marktkapitalisierung der CNCF-Landschaft beläuft sich derzeit auf 17,8 Billionen US-Dollar bei 570 Mitgliedsunternehmen. Der Betrag ist eine Gesamtsumme der Unternehmen, der anzeigt, wie hoch die Investitionen in der Community insgesamt sind. Außerdem gab es einige bemerkenswerte Übernahmen. Im Januar 2018 wurde CoreOS, das Softwareunternehmen, das die Containerlaufzeitumgebung Tectonic entwickelte, für 250 Millionen US-Dollar an Red Hat verkauft. Ende 2019 kaufte VMware das von Joe Beda und Craig McLuckie gegründete Unternehmen Heptio. Beda und McLuckie hatten gemeinsam mit Brendan Burns das Team bei Google geleitet, von dem Kubernetes ursprünglich entwickelt wurde. Burns hatte Google seither ebenfalls verlassen und war zu Microsoft gewechselt. Berichten zufolge belief sich der Kaufpreis für Heptio auf 450 Millionen US-Dollar. Im Mai 2019 erwarb Palo Alto Networks für 410 Millionen US-Dollar Twistlock, ein Unternehmen, das sich mit Containersicherheit befasst. Im gleichen Monat wurde Bitnami von VMware übernommen. Anfang 2020 gab VMware den Kauf von Pivotal bekannt, das die Kommerzialisierung von Cloud Foundry, der Open-Source-Platform-as-a-Service, angeführt hatte. Im Juli 2020, kurz vor der Veröffentlichung dieses E-Books, wurde Rancher Labs von SUSE übernommen, Berichten zufolge für 600 Millionen US-Dollar.

Das Wachstum der Community und die Reifung von Microservices als ein Entwicklungs-architekturmodell haben gemeinsam zu einem umfassenderen Wissen über Technologien für die Containerorchestrierung und verteiltes Computing im Allgemeinen geführt.

Dadurch ist klarer geworden, dass Kubernetes einen Ressourcenpool für Rechenoperationen, Speicher und Netzwerk als Basis benötigt, der von einer On-Premises-Infrastruktur oder einem

Cloud-Service bereitgestellt werden kann. Dennoch ist es hilfreich, Anfängern oder Nutzern mit einer gewissen Erfahrung die zugrunde liegende Kernarchitektur zu erläutern.

Diesen Grundlagen von Kubernetes widmet sich das vorliegende E-Book von The New Stack. Wir erläutern Konzepte wie Steuerungsebenen, Workerknoten, Serviceinventarisierung, Speicher und Netzwerke, bevor wir untersuchen, wie Kubernetes Kapazitäten für Computing auf der Größenordnung des Internets bereitstellt, wie es genutzt wird und welche Rolle es als universeller Bestandteil der Kerninfrastruktur von Unternehmen spielt.

Im zweiten Kapitel dieses E-Books wenden wir uns Containermanagementplattformen zu. Händler und Cloud-Services bieten vielfältige Plattformen an, denen allen eines gemeinsam ist: Kubernetes. Dieses E-Book erläutert, wie Kubernetes als die zugrunde liegende Architektur für Plattformen dient, die für Unternehmensrechenzentren, Cloud-Services und einen Hybridansatz sowie das Edge geeignet sind.

Außerdem wird im zweiten Kapitel dieses E-Books die Rolle cloudnativer Technologien in Containerplattformen betrachtet. Es wird gezeigt, dass cloudnative Technologien im Grunde containerbasierte Technologien sind, die als Grundlage eine containerisierte Infrastruktur, insbesondere Kubernetes, benötigen.

Microservices nutzen Komponenten, die über mehrere Container verteilt ausgeführt werden. Aber was steht für das Anwendungsmanagement zur Verfügung? Dies ist eine der großen Fragen, auf die Entwickler noch immer eine Antwort suchen. Anwendungen werden aus Komponenten zusammengesetzt. Dies sollte so geschehen, dass die Anwendung nicht nur wie gewünscht entsteht, sondern sich auch während ihres gesamten Lebenszyklus als Service einfach verwalten lässt.

Der Autor dieses E-Books, Janakiram MSV, vertritt die Meinung, dass das Container-as-a-Service-(CaaS-)Modell inzwischen die Cloud-Infrastruktur definiert. Allerdings ist es komplex und in der Verwaltung anspruchsvoll. Man kann sich eine Kubernetes-Architektur in der Produktion wie einen Bienenstock vorstellen. Vordergründig herrscht Chaos, aber aufgrund der technischen Umsetzung werden Microservices in Containern ausgeführt, die wiederum von Kubernetes orchestriert werden. Diejenigen, die das verstehen, können Entwicklung und

Management beliebig skalieren. Sie sind auf der Erfolgsspur. Aber in Wirklichkeit stehen wir fast alle immer noch ganz am Anfang.

Und das ist tatsächlich die Quintessenz der Herausforderung, vor der die Kubernetes-Community am Übergang von hoffnungsvollen Anfängen zu seiner laufenden Entwicklung als eine Plattform steht, die nicht nur für große Unternehmen relevant ist, sondern auch für mittlere und kleine.

Die Bereitstellungskultur ist noch immer lückenhaft. Entwickler benötigen eine Möglichkeit zur einfachen Verpackung von Anwendungen und es muss geklärt werden, wer für die Konfiguration der Services zuständig ist. Eine weitere Herausforderung besteht darin, dass bei Richtlinien- und Sicherheitsfragen noch immer die herkömmliche Praxis vorherrscht.

Die Aussichten für Kubernetes und Teile des Ökosystems, die noch in den Kinderschuhen stecken, sind vielversprechend. Daten sind in Kubernetes im Großen und Ganzen immer noch ein weißer Fleck. Während die Bedeutung von Kubernetes als Steuerungsebene inzwischen unangefochten ist, bleiben Fragen zur Datenebene offen. Service-Mesh-Fähigkeiten haben das Management des Datenverkehrs, die Sicherheit und Beobachtbarkeit zwar verbessert, doch ist die Einarbeitung noch immer sehr anspruchsvoll.

Am spannendsten finden wir allerdings die Konzepte im Zusammenhang mit der Beobachtbarkeit. Monitoring eignete sich vor allem für unternehmenseigene On-Premises-Architekturen. Es wurde für die Reaktion konzipiert, aber das genügt nicht mehr. Die heutige Realität verlangt von uns, zu beobachten, was vor sich geht, vorauszuschauen und proaktiv Lösungen zu finden.

Vielen Dank, dass Sie dieses E-Book heruntergeladen haben. Wenn Sie Kontakt zu uns aufnehmen möchten, sind Sie jederzeit herzlich dazu eingeladen.

Alex Williams

Gründer und Verleger

The New Stack

@alexwilliams

alex@thenewstack.io

KAPITEL 01

Kubernetes, das Betriebssystem für die Cloud

In den letzten zehn Jahren hat der IT-Infrastrukturmarkt einen kompletten Umbruch durchlebt. Zuerst revolutionierte Infrastructure-as-a-Service (IaaS) die Bereitstellung und Nutzung von Rechen-, Speicher- und Netzwerkressourcen in Unternehmen. Dann tauchten in der zweiten Hälfte der 2010er-Jahre immer mehr Container, Containermanagementplattformen und cloudbasierte Container-as-a-Service-(CaaS-)Angebote auf.

Cloudbasierte Infrastrukturdienste wie Amazon EC2, Azure Virtual Machines und Google Compute Engine boten Infrastrukturen an, die sich als Selfservicemodelle oder programmgesteuert bedarfsgerecht bereitstellen ließen. Mit diesen Diensten konnten Unternehmen klein anfangen und schnell hochskalieren.

Im Jahr 2013 stellte Docker, Inc. Docker vor, eine schlanke Plattform auf Betriebssystemebene, die auf Linux-Containern basiert. Docker nutzte die Tatsache aus, dass in Linux mehrere voneinander isolierte Anwendungen in ein und demselben Betriebssystem ausgeführt werden können.

Der etablierte Ansatz war bis dahin, mithilfe von Virtualisierung mehrere Betriebssysteme im selben übergeordneten Betriebssystem auszuführen. Dabei waren Hypervisoren, also Softwarekomponenten, die die Virtualisierung von Rechenressourcen verwalten, für die Partitionierung der Ressourcen sowie die strikte Isolierung der virtuellen Maschinen zuständig. Linux-Container ermöglichen eine Virtualisierung auf Betriebssystemebene, sodass mehrere

voneinander isolierte Linux-Prozesse (Container) auf einem Host mit nur einem Linux-Kernel ausgeführt werden können. Der Linux-Kernel nutzt cgroups für das Ressourcenmanagement sowie Linux-Namespaces zum Isolieren der Anwendungen und priorisiert dabei Ressourcen wie CPU, Speicher, Block I/O und Netzwerk – ganz ohne virtuelle Maschinen.

Im Gegensatz zu virtuellen Maschinen benötigen Container daher keinen Hypervisor. Stattdessen greifen sie gemeinsam auf die zugrunde liegenden Betriebssystemdienste auf Kernelebene zu. Linux-Container sind kleiner, können so schnell wie Prozesse gestartet werden, lassen sich rasch skalieren und sind vor allem portierbar. Ein auf Ubuntu erstellter Container kann schnell und ohne die geringsten Änderungen auf Red Hat genutzt werden. Administratoren können containerisierte Anwendungen in Millisekunden starten und in kurzer Zeit auf Hunderte von Instanzen skalieren.

Container waren bereits Bestandteil moderner Unix-basierter Betriebssysteme wie etwa Linux (LXC), FreeBSD (Jails) und Solaris (Zones), doch Docker, Inc. machte diese Technologie für Entwickler zugänglich. Das Unternehmen revolutionierte damit die Entwicklung und Bereitstellung von Anwendungen.

Mit der Kombination aus IaaS und Containern in Public Clouds stellte es Organisationen ein völlig neues Maß an Skalierbarkeit und Flexibilität in Aussicht. Da nun Dutzende – und in einigen Fällen sogar Hunderte – von Containern in einer virtuellen Maschine gestartet werden konnten, konnten die CPU und die Speicherressourcen der virtuellen Maschinen (VM) maximal ausgenutzt werden. Und mit in Public Clouds bereitgestellten Containern ließen sich Workloads in der Größenordnung des Internets zu bezahlbaren Kosten ausführen. Die vielversprechende Kombination aus IaaS und Containern wurde für Hyperscale-Start-ups zum Trumpf.

Als Docker, Inc. die Containerlaufzeitumgebung und die Tools zur Verwaltung der Lebenszyklen von Containern bereitstellte, wurde Branchenexperten bewusst, dass auch eine Plattform zur Verwaltung von Hunderten von Containern auf Hunderten von virtuellen Maschinen erforderlich war. Dies führte zur Entwicklung von Docker Swarm durch Docker, Inc. und DC/OS durch D2iQ (ehemals Mesosphere).

Schon bevor Container in Entwicklerkreisen bekannter wurden, führte Google einige seiner

wichtigsten Webdienste in Linux-Containern aus. Während einer Präsentation auf der GlueCon 2014 behauptete Joe Beda (einer der Gründer von Kubernetes), Google würde mehr als zwei Milliarden Container pro Woche starten. Google konnte so viele Container verwalten, weil es ein internes Verwaltungstool für Rechenzentren namens Borg nutzte.

Im Juni 2014 stellte Google Kubernetes vor, eine Open-Source-Softwareplattform zur Verwaltung zahlreicher Container. Dabei handelte es sich um eine flexiblere Variante von Borg. Google integrierte das Beste aus Borg in Kubernetes und schloss dabei gezielt Schwachstellen, die von Borg-Benutzern über die Jahre identifiziert wurden.

Im Jahr 2015 wurde Kubernetes 1.0 der Linux Foundation beigesteuert. Diese rief daraufhin die Cloud Native Computing Foundation (CNCF) zur Verwaltung und Leitung des Projekts ins Leben. Heute verwaltet die CNCF mehrere Open-Source-Projekte im Zusammenhang mit Containern, darunter Containerd, Envoy, Prometheus und andere.

Wie passte Docker in dieses Bild? Docker sollte die Entwicklung moderner Anwendungen vereinfachen. Die Entwicklerin installierte die Docker Engine auf ihrer Workstation und nutzte dann Docker-APIs (Anwendungsprogrammierschnittstellen) und -Tools zur Verwaltung der Lebenszyklen der containerisierten Anwendungen. Docker entwickelte Compose und Swarm als Erweiterung zur eigentlichen Docker Engine. So wurde es möglich, mit dem gewohnten Workflow und der gewohnten Toolchain Workloads mit mehreren Containern auf mehreren Knoten bzw. Maschinen bereitzustellen und zu verwalten.

Der nächste Fortschritt kam von Google: Auf Kubernetes konnten zahlreiche containerisierte Workloads unterschiedlicher Art ausgeführt werden. Dank seiner Erweiterbarkeit, seiner Skalierbarkeit und seiner Auswahl an Bereitstellungsumgebungen war Kubernetes schon bald bei Entwicklern und Betreibern beliebt.

Apache Mesos, ein an der University of California in Berkeley entwickeltes Open-Source-Projekt, war eine der ersten verfügbaren Rechnerarchitekturen zur Verwaltung von Anwendungs-Workloads auf Compute-Clustern. Aber da die Branche und die Open-Source-Community das Kubernetes-Ökosystem bevorzugten, verlor das Projekt mit der Zeit an Relevanz. Mesos ist eine ausgereifte Rechenumgebung, doch sie eignet sich eher für große Systeme mit Hunderten von

Knoten. Ursprünglich wurde Mesos für verteilte Anwendungen entwickelt, die auf Apache Hadoop, Apache Spark und Kafka basierten. Die Containerorchestrierung kam erst viel später mit dem Plug-in Marathon dazu. Kubernetes hingegen wurde speziell für die Containerisierung entwickelt und sollte auf kleinen und großen Clustern gleichermaßen einfach und flexibel ausgeführt werden können.

Docker, Inc. entschied sich ebenfalls für Kubernetes als bevorzugtes Orchestrierungstool und integrierte es in Docker Desktop und Docker Enterprise. Im November 2019 kaufte Mirantis, ein OpenStack-basiertes Infrastrukturunternehmen, die Enterprise-Plattform von Docker und die dazugehörigen Kubernetes-as-a-Service-Optionen.

Dank seiner Benutzerfreundlichkeit, Verfügbarkeit und Skalierbarkeit entwickelte Kubernetes sich zur beliebtesten Plattform für das Containermanagement. Es ist sogar eines der am schnellsten wachsenden Open-Source-Projekte in der Geschichte der Informatik. Moderne und Greenfield-Anwendungen nutzen Kubernetes mehr und mehr, was zur Entwicklung von Unternehmensplattformen für das Containermanagement wie beispielsweise Google Anthos, Red Hat OpenShift und VMware Tanzu geführt hat. Außerdem gibt es mittlerweile zahlreiche verwaltete CaaS-Services in Public Clouds, etwa Amazon Elastic Kubernetes Service, Azure Kubernetes Service und Google Kubernetes Engine.

Die Rolle der Umgebungsgröße

Während die ersten Herausforderungen bei der Containernutzung, wie die Speicherverwaltung, inzwischen bewältigt wurden, sehen sich die Benutzer von Kubernetes weiterhin mit Fragen bezüglich Sicherheit und Kulturwandel konfrontiert. Auch müssen sie sich an neue Entwicklungen wie etwa das Service Mesh gewöhnen, in dem häufig Sidecar-Proxys zur Steuerung von Microservices in Kubernetes-Umgebungen verwendet werden. Das Service Mesh soll die Steuerung des Anwendungsverkehrs, die Beobachtbarkeit und die Sicherheit verteilter Systeme verbessern. Werfen wir nun einen Blick auf die jährlichen Umfragedaten der CNCF.

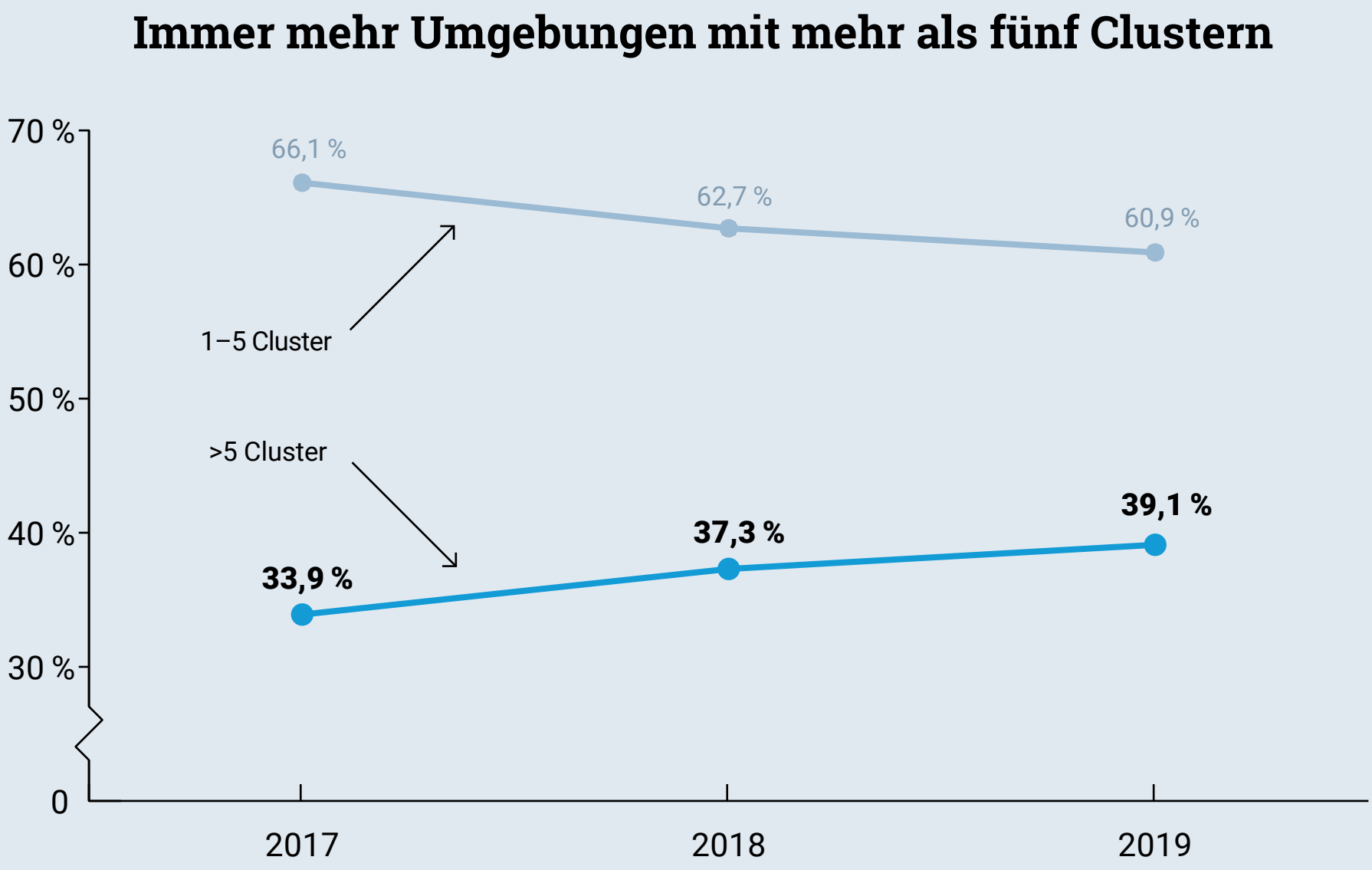
An den Zahlen erkennt man den Weg von der Experimentierphase bis hin zu größeren

Kubernetes-Bereitstellungen. Im Jahr 2017 verwendeten 64 Prozent der mehr als 550 Teilnehmer an der jährlichen Umfrage der CNCF mindestens ein Kubernetes-Projekt in der Produktion. 2018 nahmen bereits über 2.400 Personen, die Kubernetes in Erwägung zogen, an der Umfrage teil. Darin spiegelt sich das steigende Interesse an Kubernetes wider. 2019 setzten 78 Prozent der 1.300 Umfrageteilnehmer Kubernetes produktiv ein und bestätigten damit erneut die Beliebtheit von Kubernetes in der Cloud Native-Community.

Was war zuerst da, die Henne oder das Ei? Die zunehmende Nutzung von Containern generierte zunächst eine Nachfrage nach Lösungen für die Containerorchestrierung, doch in einer einmal eingerichteten Kubernetes-Infrastruktur ist es einfacher, mehr Container auszuführen. Typischerweise verwalten Kubernetes-Benutzer mehr Container, aber die Anzahl der verwendeten Container ist ungleich verteilt.

Der Anteil der Kubernetes-Bereitstellungen mit mehr als fünf Clustern steigt. Wie aus

Abb. 1.1: *Größere Bereitstellungen sind nun weiter verbreitet. Der Anteil der Kubernetes-Benutzer mit mehr als fünf Clustern stieg von 34 Prozent im Jahr 2017 auf 39 Prozent im Jahr 2019.*



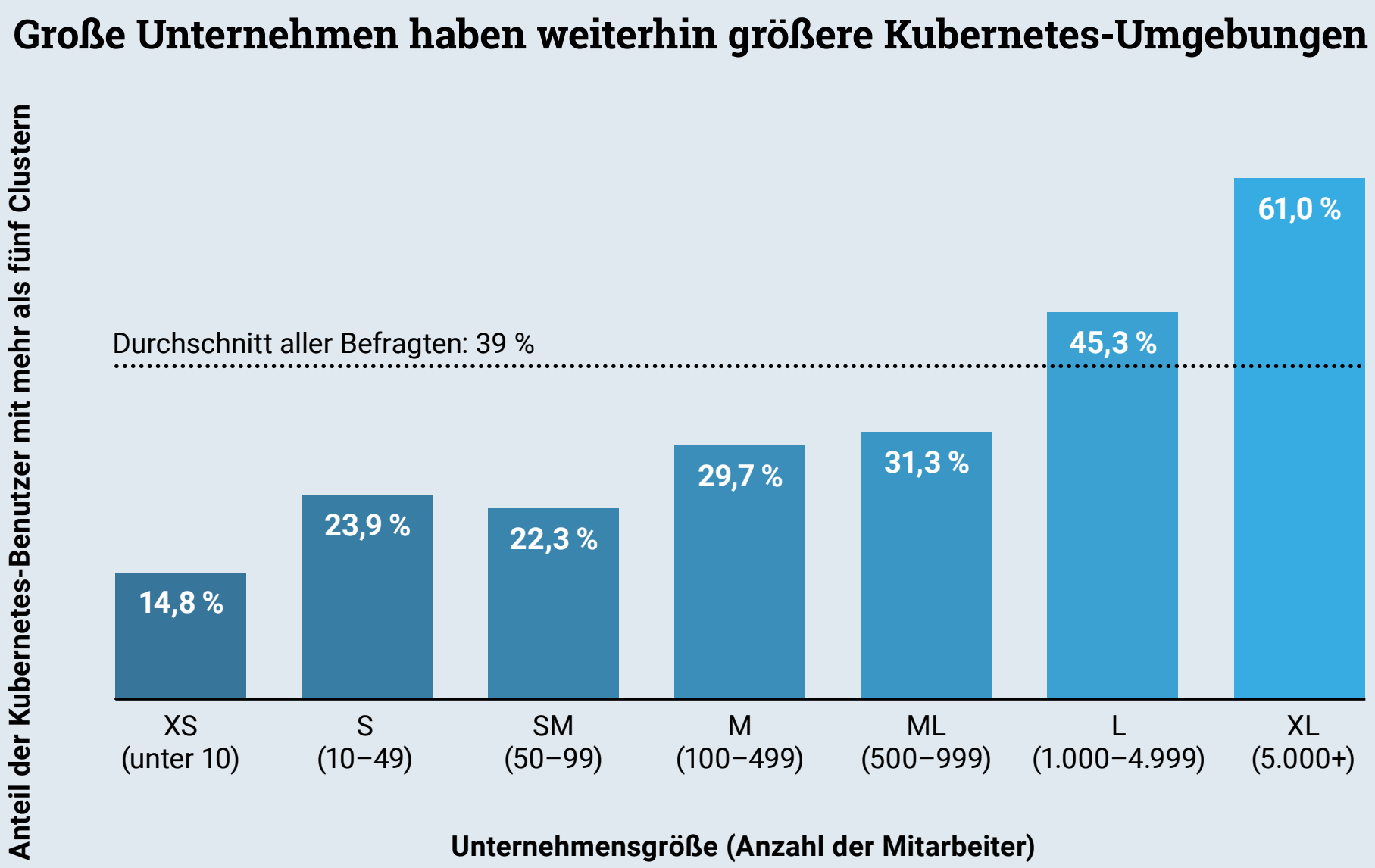
Quelle: Auswertung der CNCF-Umfragen von 2017, 2018 und 2019 durch The New Stack.
Frage: Mit welchen Herausforderungen sind Sie bei der Verwendung/Bereitstellung von Containern konfrontiert?
Bitte wählen Sie alle zutreffenden Antworten aus. Bei der Auswertung wurden nur die Antworten von Umfrageteilnehmern mit mindestens einem Kubernetes-Cluster berücksichtigt. 2017, n=454; 2018, n=1.475; 2019, n=1.197.

Abbildung 1.1 ersichtlich lag er 2017 bei 34 Prozent und 2019 bei 39 Prozent. Tatsächlich führt mehr als die Hälfte dieser Gruppe mindestens 1.000 Container aus. Der Anteil an sehr großen Umgebungen ist also ebenfalls gestiegen, von 55 Prozent im Jahr 2017 auf 62 Prozent im Jahr 2019.

Fazit:

- Große Unternehmen sind die größten Anwender. Für die Kubernetes-Bereitstellung bleiben sie tendenziell bei ihren bewährten Anbietern.
- Die durchschnittliche Größe von Containerumgebungen ist gestiegen, aber Unternehmen mit fünf oder weniger Kubernetes-Clustern melden mehr Probleme bei der Anpassung.
- Zahlreiche etablierte und neue Lösungen für Speicher, den eingehenden Datenverkehr und das Service Mesh sind im Einsatz. Die neuen Tools werden tendenziell eher in Betracht gezogen, wenn es unbewältigte Herausforderungen gibt. Generell bleiben Kunden aber bei Anbietern, zu denen sie bereits Beziehungen aufgebaut haben.

Abb. 1.2: *61 Prozent der Kubernetes-Benutzer in Unternehmen mit mindestens 5.000 Mitarbeitern haben mehr als fünf Cluster. Der Studiendurchschnitt liegt bei 39 Prozent.*



Quelle: Auswertung der CNCF-Umfrage von 2019 durch The New Stack. Frage: Falls Sie Kubernetes verwenden: Wie viele Produktionscluster haben Sie? Unter 10, n=81; 10–49, n=142; 50–99, n=121; 100–499, n=239; 500–999, n=80; 1.000–4.999, n=170; 5.000 oder mehr, n=364.

Abbildung 1.2 zeigt, dass die Mitarbeiterzahl ein besserer Indikator für große Kubernetes-Umgebungen ist als die Zahl der Container. 61 Prozent der Kubernetes-Benutzer in Unternehmen mit mindestens 5.000 Mitarbeitern haben mehr als fünf Cluster.

Darüber hinaus haben Unternehmen mit mindestens 1.000 Mitarbeitern mit größerer Wahrscheinlichkeit mehr als fünf Cluster. Das traf 2017 auf 51 Prozent und 2019 auf 56 Prozent der entsprechenden Unternehmen zu.

Diese Zahlen sind von Bedeutung, da Unternehmen mit großen Kubernetes-Umgebungen inzwischen viel weniger Probleme mit Containern haben.

Organisationen mit fünf oder weniger Clustern haben größere Schwierigkeiten bei der Verwaltung von vielen Containern.

Was ist Orchestrierung?

Zwei Trends haben moderne Infrastrukturen erheblich beeinflusst: Container und DevOps. Das DevOps-Ökosystem ermöglicht eine kontinuierliche Integration, kontinuierliche Tests, eine kontinuierliche Bereitstellung und ein kontinuierliches Monitoring und konnte so zur Beschleunigung der Softwareentwicklung beitragen. Die Nutzung von Containern in Verbindung mit bewährten Best Practices für DevOps begünstigt außerdem die schnelle Bereitstellung großer Umgebungen.

Während Entwickler mit Containern produktiver werden können, profitieren Unternehmen, die die Investitionen in ihre DevOps und Prozesse optimieren möchten, von Orchestrierungstools. Containerorchestrierung bietet unter anderem folgende Vorteile:

- Effiziente Ressourcenverwaltung
- Nahtlose Skalierung von Diensten
- Hochverfügbarkeit
- Relativ geringe Betriebskosten in großen Umgebungen

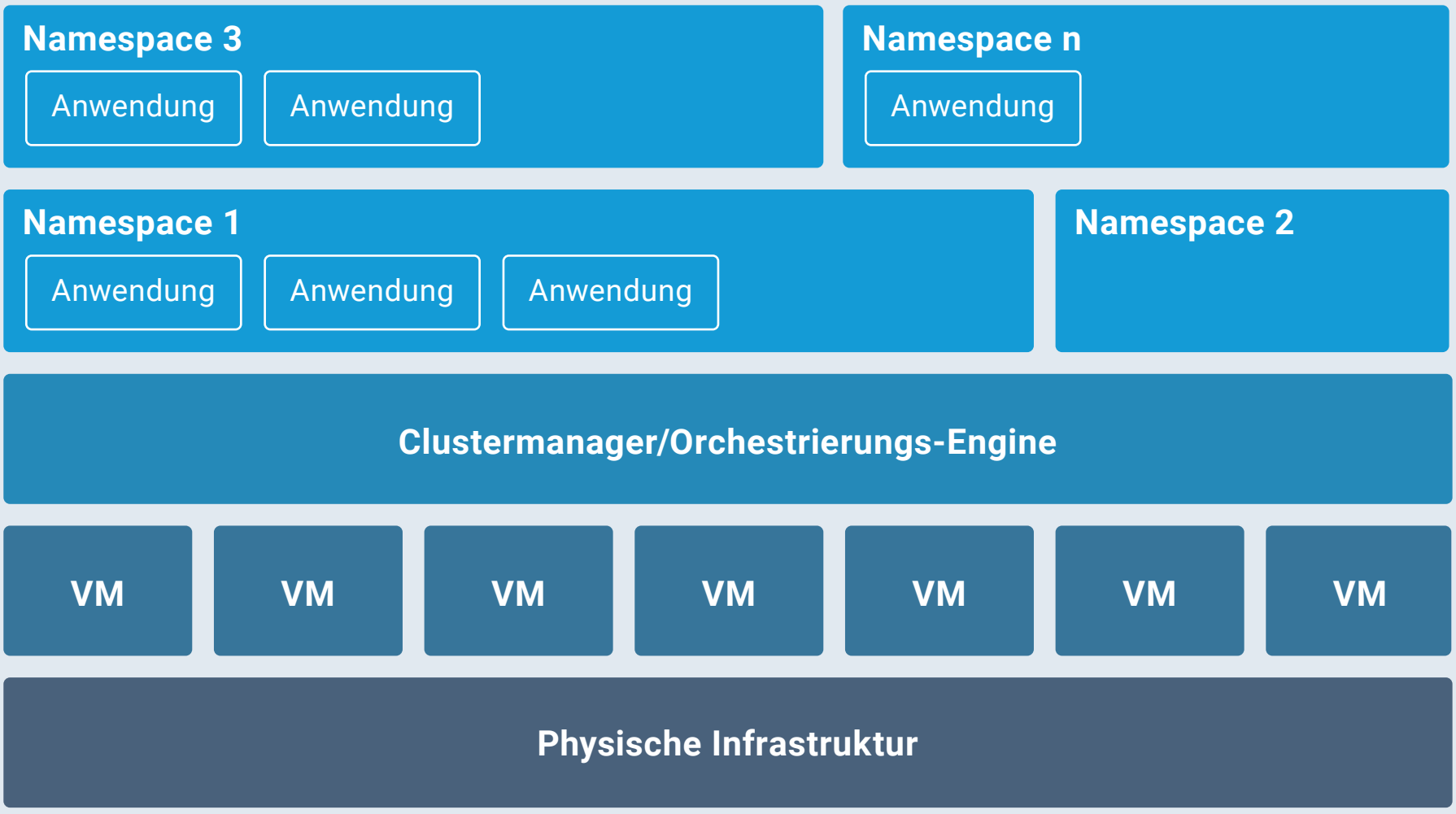


Abb. 1.3: *Die Ressourcenschichten eines Systems aus der Perspektive der Engine zur Containerorchestrierung*

- Ein deklaratives Modell für die meisten Orchestrierungstools, das eine autonomere Verwaltung ermöglicht
- Prozessbasierte Infrastructure-as-a-Service (IaaS), die wie eine Platform-as-a-Service (PaaS) verwaltet werden kann
- Einheitlicher Betrieb in On-Premises- und Public-Cloud-Umgebungen

Betreiber nutzen IaaS, wenn sie ihre Umgebungen besser steuern oder automatisieren wollen. Entwickler bevorzugen PaaS-Dienste, weil diese mehr Flexibilität, Skalierbarkeit und Produktivität bieten. Containerorchestrierungstools vereinen das Beste aus beiden Welten: Automatisierung und Skalierbarkeit.

Container begünstigen einen nahtlosen DevOps-Workflow – und somit eine erhebliche Steigerung der Entwicklerproduktivität. Entwickler können Container-Images erstellen, einen Container starten und Code in diesem Container entwickeln, der dann in lokalen Rechenzentren oder Public-Cloud-Umgebungen bereitgestellt werden kann. Diese nahtlose Entwicklung führt jedoch nicht automatisch zu effizienteren Abläufen in Produktionsumgebungen.

Die Produktionsumgebung unterscheidet sich häufig stark von der lokalen Umgebung auf dem Laptop eines Entwicklers. Sowohl für große Bereitstellungen herkömmlicher dreistufiger Anwendungen als auch für auf Microservices basierende Anwendungen erweist sich die Verwaltung zahlreicher Container und des ihnen zugrunde liegenden Knotenclusters als schwierige Aufgabe. Skalierbarkeit setzt Orchestrierung voraus, denn in großen Umgebungen ist Automatisierung unverzichtbar.

Unsere Wahrnehmung der Infrastruktur virtueller Maschinen hat sich durch das von Natur aus verteilte Cloud Computing grundlegend geändert. Das Konzept „Cattle vs Pets“ nutzt Rinder und Haustiere als Sinnbilder, um zu verdeutlichen, dass Container mehr wie austauschbare Nutztiere und weniger wie geliebte Haustiere behandelt werden sollten. Diese Sichtweise trug dazu bei, dass Container und Infrastrukturen nun in einem anderen Licht betrachtet werden.

Sowohl die Container als auch die Infrastruktur, auf der sie ausgeführt werden, gelten als unveränderlich, denn ein Grundsatz der Containerisierung ist, dass Container und Server nach deren Bereitstellung niemals geändert werden sollten. Wenn Aktualisierungen, Korrekturen oder Änderungen erforderlich sind, werden neue Container oder Server anhand einer Vorlage eingerichtet und bereitgestellt, um die alten zu ersetzen. Das ist vergleichbar mit der Verwaltung von Vieh auf einem Milchbauernhof.

Demgegenüber werden herkömmliche Server und auch virtuelle Maschinen nicht als unveränderlich behandelt – sie sind eher wie Haustiere, von denen man sich nicht trennt. Daher sind ihre Wartungskosten hoch, denn sie erfordern die ständige Aufmerksamkeit des Betriebsteams.

Eine unveränderliche Infrastruktur ist programmierbar – und damit automatisierbar. Infrastructure-as-Code (IaC) ist eines der wichtigsten Merkmale einer modernen Infrastruktur. Mit diesem Ansatz kann eine Anwendung eine Infrastruktur programmgesteuert bereitstellen, konfigurieren und nutzen und sich somit selbst ausführen.

Die Kombination aus Containerorchestrierung, unveränderlicher Infrastruktur und IaaC-basierter Automatisierung sorgt für Flexibilität und Skalierbarkeit.

Der Einsatz großer Containerumgebungen in der Praxis begünstigte die Erweiterung und

Weiterentwicklung der Konzepte der „Skalierbarkeit“ und „Ressourcenverfügbarkeit“.

Eine typische Containerorchestrierungsplattform bietet die folgenden grundlegenden Funktionen:

- Planung
- Ressourcenverwaltung
- Serviceinventarisierung
- Prüfung des Systemzustands
- Automatische Skalierung
- Updates und Upgrades

Der Containerorchestrierungsmarkt wird derzeit von Kubernetes dominiert. Kubernetes ist bei Unternehmen, Plattformentwicklern, Cloud-Anbietern und Infrastrukturfirmen gleichermaßen beliebt.

Containerorchestrierung fördert die Nutzung von Microservice-Architekturen, in denen sich eine Anwendung aus kleineren, atomaren, unabhängigen Diensten zusammensetzt, von denen jeder nur eine Aufgabe hat. Jeder Microservice wird als Container gepackt und mehrere Microservices, die logisch zur gleichen Anwendung gehören, werden zur Laufzeit von Kubernetes orchestriert.

Die wachsende Beliebtheit von Kubernetes hat zur Entstehung neuer Marktsegmente geführt, in denen Produkte angeboten werden, die die Plattform zur Containerorchestrierung und -verwaltung unterstützen sollen. Vom Speicher über das Netzwerk und Monitoring bis hin zur Sicherheit bieten neuartige Unternehmen und Start-ups containerbasierte Produkte und Dienste an. Im folgenden Kapitel gehen wir auf einige der Komponenten cloudbasierter Plattformen und Start-ups in diesem neuen Ökosystem ein.

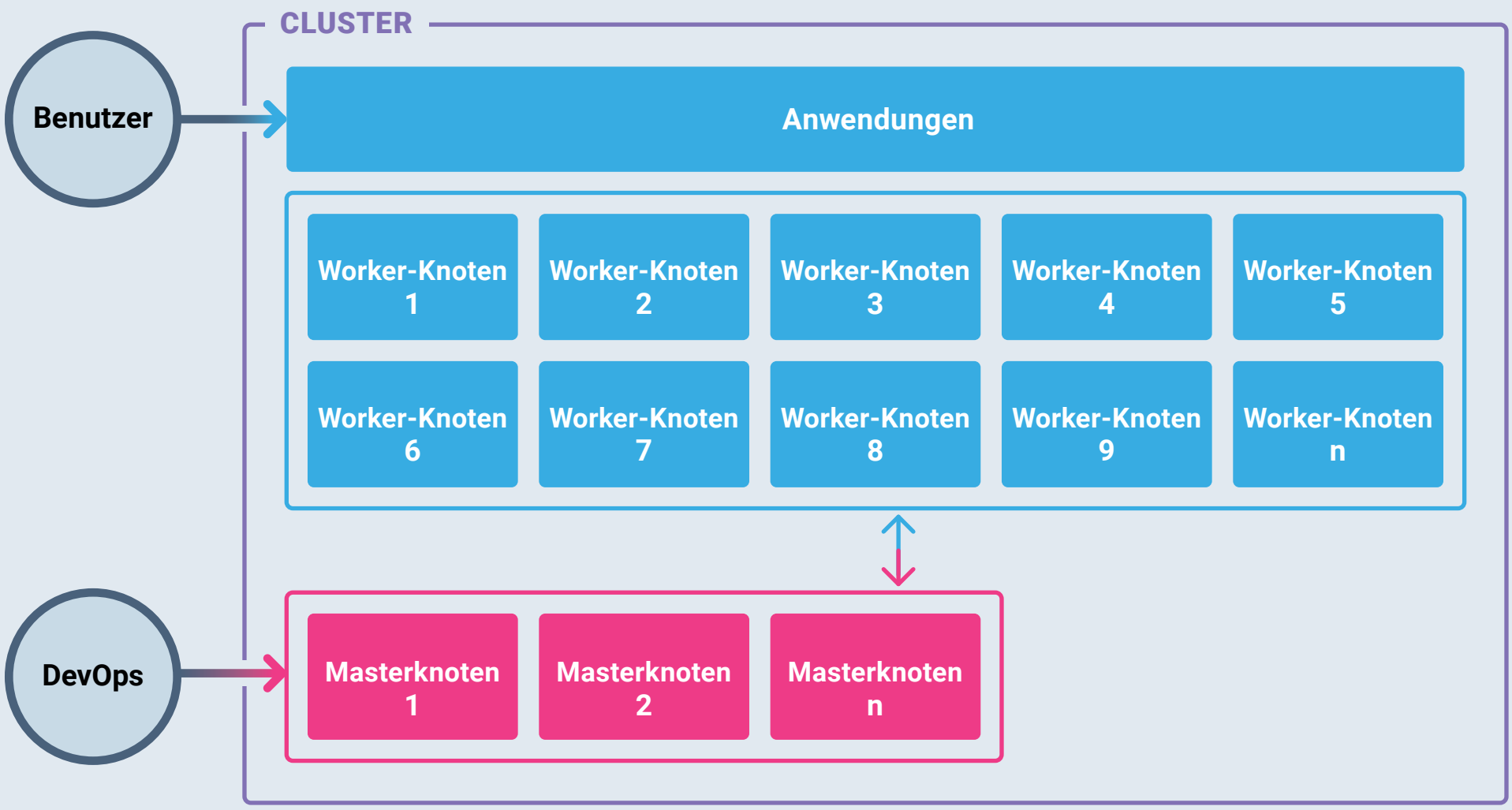
Die Kubernetes-Architektur

Eine moderne, als Containerset gepackte und in Form von Microservices bereitgestellte Anwendung benötigt eine Infrastruktur, die stabil genug ist, um die Anforderungen beim

Clustern zu erfüllen und der Belastung durch die dynamische Orchestrierung standzuhalten. Eine solche Infrastruktur sollte elementare Funktionen (Primitiven) für die hostübergreifende Planung, Überwachung, Aktualisierung und Containerverschiebung bereitstellen. Sie muss die zugrunde liegenden Rechen-, Speicher- und Netzwerkprimitiven als Ressourcenpool behandeln. Jede containerisierte Workload sollte die für sie sichtbaren Ressourcen einschließlich der CPU-Kerne, der Speichereinheiten und der Netzwerke nutzen können.

Kubernetes (siehe Abbildung 1.4) ist ein verteiltes Open-Source-System, das die zugrunde liegende physische Infrastruktur abstrahiert und somit die Ausführung containerisierter Anwendungen in großem Maßstab erleichtert. Eine über ihren gesamten Lebenszyklus in Kubernetes verwaltete Anwendung setzt sich aus Containern zusammen, die als Set erfasst und in einer Einheit koordiniert werden. Mit einer effizienten Schicht zur Clusterverwaltung kann Kubernetes diese Anwendung, wie in der Abbildung unten dargestellt, von ihrer Infrastruktur entkoppeln. Sobald die Kubernetes-Infrastruktur vollständig konfiguriert ist, können sich DevOps-Teams auf das Workload-Management konzentrieren, statt sich um den zugrunde

Abb. 1.4: *Ein Kubernetes-Cluster im Überblick*



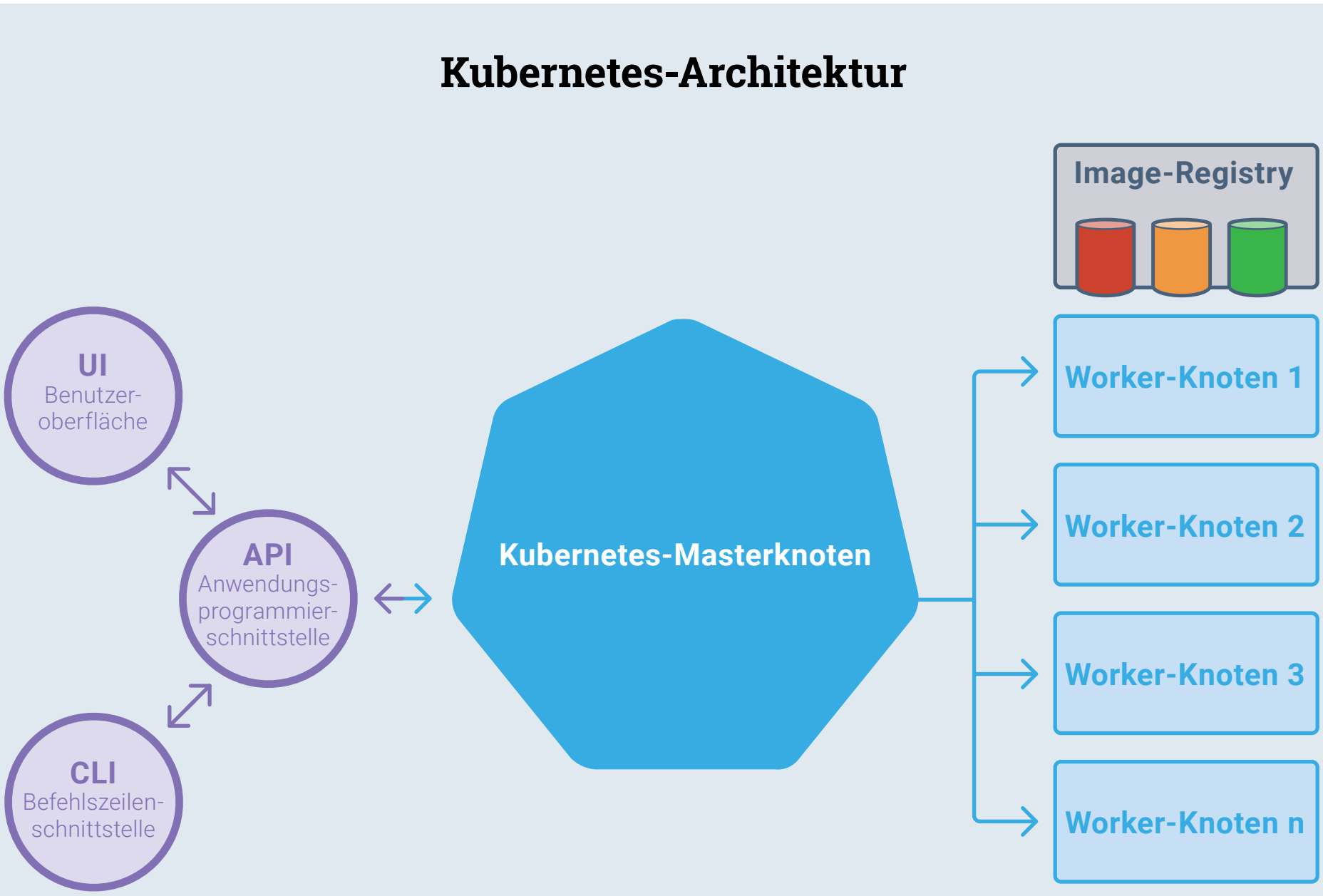
Quelle: Janakiram MSV

liegenden Ressourcenpool (CPU und Speicher) kümmern zu müssen, da dieser von Kubernetes verwaltet wird.

Kubernetes ist ein Beispiel für ein verteiltes System mit ausgefeilter Architektur. Die Plattform behandelt alle Maschinen in einem Cluster als gemeinsamen Ressourcenpool. Sie übernimmt die Rolle eines verteilten Betriebssystems und verwaltet in effizienter Form die Planung, die Ressourcenzuweisung, das Monitoring des Infrastrukturstatus und sogar die Beibehaltung des gewünschten Zustands der Infrastruktur und der Workloads. Kubernetes ist ein Betriebssystem, auf dem moderne Anwendungen cluster- und infrastrukturübergreifend in Cloud-Diensten und privaten Rechenzentren ausgeführt werden können.

Wie jedes andere ausgereifte verteilte System hat auch Kubernetes zwei Schichten: Master- und Worker-Knoten (siehe Abbildung 1.5). In den Masterknoten wird typischerweise die Steuerungsebene ausgeführt, die für die Planung und Verwaltung des Lebenszyklus von Workloads verantwortlich ist. Die Worker-Knoten sind die „Lasttiere“, auf denen die Anwendungen laufen.

Abb. 1.5: *Die Rolle eines Masterknotens in der Kubernetes-Architektur*



Quelle: Janakiram MSV

Eine aus Master- und Worker-Knoten bestehende Umgebung wird als Cluster bezeichnet.

Die für das Clustermanagement verantwortlichen DevOps-Teams kommunizieren über eine Befehlszeilenschnittstelle (CLI) oder über Tools von Drittanbietern mit der API der Steuerungsebene. Die Benutzer greifen auf die Anwendungen zu, die auf den Worker-Knoten ausgeführt werden. Die Anwendungen bestehen aus einem oder mehreren in einer zugänglichen Image-Registry abgelegten Container-Images.

Steuerungsebene

Auf der Steuerungsebene werden die Kubernetes-Komponenten ausgeführt, die die Kernfunktionen bereitstellen: den Zugang zur Kubernetes-API, das Workload-Scheduling, das Clustermanagement und die systemweite Steuerung des Datenverkehrs. Wie in Abbildung 1.5 dargestellt, überwacht der Masterknoten die in allen Knoten ausgeführten Container und den Zustand aller registrierten Knoten. Container-Images, die als einsatzfähige Artefakte fungieren, müssen dem Kubernetes-Cluster über eine private oder öffentliche Image-Registry zur Verfügung gestellt werden. Die Knoten sind für die Planung und Ausführung der Anwendungen verantwortlich und greifen über die Containerlaufzeitumgebung auf die Images in der Registry zu.

Wie in Abbildung 1.6 dargestellt, laufen auf dem Kubernetes-Masterknoten die folgenden Komponenten der Steuerungsebene:

etcd

etcd ist ein persistenter, schlanker, verteilter Datenspeicher für Schlüssel-Wert-Paare, der zur Speicherung der Konfigurationsdaten von Clustern verwendet wird. Entwickelt wurde etcd von CoreOS, das später von Red Hat übernommen wurde. Der Gesamtzustand des Clusters ist zu jedem beliebigen Zeitpunkt in etcd enthalten, das als „Single Source of Truth“ fungiert. Mehrere andere Komponenten und Dienste reagieren auf alle Änderungen in etcd, um den gewünschten Anwendungszustand beizubehalten. Der gewünschte Zustand wird durch eine deklarative Richtlinie definiert – ein Dokument, in dem die optimale Umgebung für die jeweilige Anwendung beschrieben wird. Ziel des Orchestrators ist es, diese Umgebung zu schaffen. Die Richtlinie legt auch fest, wie der Orchestrator mit verschiedenen Anwendungseigenschaften umgehen sollte, beispielsweise mit der Anzahl der Instanzen, den Speicheranforderungen und der

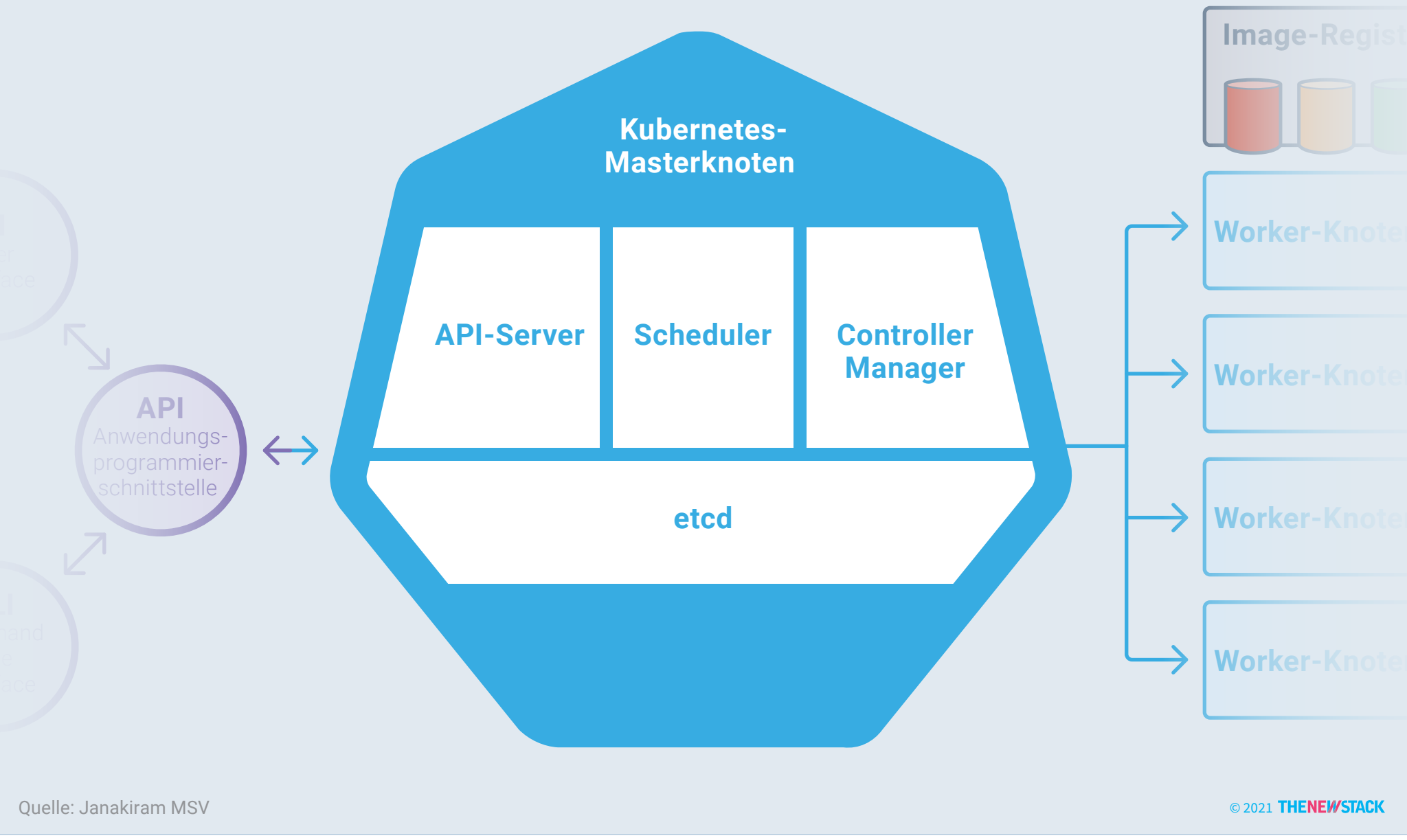


Abb. 1.6: *Komponenten eines Masterknotens*

Ressourcenzuweisung.

Auf die etcd-Datenbank kann nur über den API-Server zugegriffen werden. Jede Clusterkomponente, die aus etcd lesen oder in etcd schreiben muss, tut dies über den API-Server.

API-Server

Der API-Server macht die Kubernetes-API via JSON und HTTP zugänglich und stellt damit eine Representational State Transfer-(REST-)Schnittstelle für die internen und externen Endpunkte des Orchestrators bereit. Aufrufe an den API-Server sind über die Befehlszeilenschnittstelle, die webbasierte Benutzeroberfläche (UI) oder ein anderes Tool möglich. Der Server bearbeitet und validiert den Aufruf und aktualisiert anschließend den Status der API-Objekte in etcd. Auf diese Weise können Workloads und Container über mehrere Worker-Knoten hinweg konfiguriert werden.

Scheduler

Der Scheduler entscheidet aufgrund der verfügbaren Ressourcen, auf welchem Knoten jede

Workload ausgeführt werden soll, und verfolgt dann die Ressourcennutzung, um sicherzustellen, dass kein Pod mehr Ressourcen nutzt, als ihm zugewiesen wurden. Er verwaltet und verfolgt den Ressourcenbedarf, die verfügbaren Ressourcen sowie zahlreiche benutzerdefinierte Bedingungen und Richtlinien wie etwa die Quality-of-Service (QoS), die (Anti-)Affinitätsanforderungen und die Datenlokalität. Wenn ein Mitarbeiterteam beispielsweise das Ressourcenmodell deklarativ definiert, interpretiert der Scheduler diese Deklarationen als Anweisungen für die Bereitstellung und Zuweisung der richtigen Ressourcen zu den einzelnen Workloads.

Controller Manager

Der Teil des Masterknotens, der Kubernetes seine Flexibilität verleiht, ist der Controller Manager. Er sorgt mithilfe eines ausgefeilten Controllers dafür, dass das Cluster den gewünschten Anwendungszustand jederzeit beibehält. Ein Controller ist eine Steuerungsschleife, die den Freigabestatus des Clusters über den API-Server überwacht und gegebenenfalls ändert, um vom aktuellen zum gewünschten Zustand zu gelangen.

Der Controller erhält den stabilen Status der Knoten und Pods aufrecht, indem er den Systemzustand des Clusters und der auf diesem Cluster ausgeführten Workloads kontinuierlich überwacht. Wenn sich beispielsweise der Systemzustand eines Knotens verschlechtert, kann eventuell nicht mehr auf die Pods zugegriffen werden, die auf diesem Knoten laufen. In einem solchen Fall muss der Controller die gleiche Anzahl an neuen Pods in einem anderen Knoten einplanen. So wird sichergestellt, dass das Cluster den erwarteten Zustand konsistent beibehält.

In Kubernetes sind mehrere Controller integriert, die im Controller Manager ausgeführt werden. Diese Controller bieten Primitiven, die auf eine bestimmte Klasse von Workloads wie etwa Stateless, Stateful und Scheduled CronJobs sowie Run-to-Completion-Jobs abgestimmt sind. Sowohl Entwickler als auch Betreiber können diese Primitiven nutzen, wenn sie Anwendungen in Kubernetes packen und bereitstellen.

Worker-Knoten

Der Knoten ist das Lasttier eines Kubernetes-Clusters. Er ist für die Ausführung containerisierter Workloads, für zusätzliche Komponenten zur Protokollierung, Überwachung und Serviceinventarisierung sowie für optionale Add-ons verantwortlich. Seine Aufgabe ist es, den

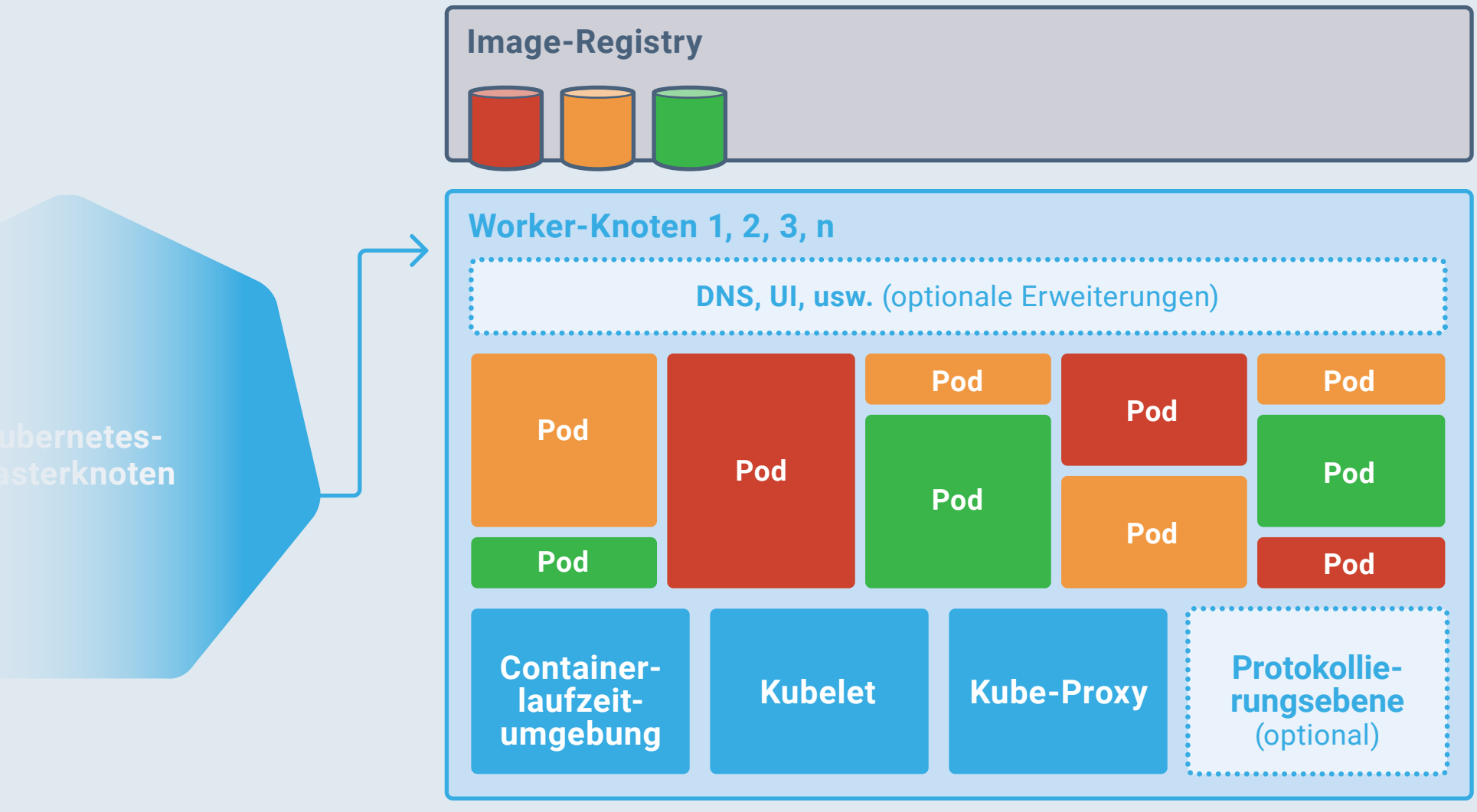


Abb. 1.7: *Komponenten der Worker-Knoten*

Anwendungen Rechen-, Netzwerk- und Speicherressourcen zugänglich zu machen. Jeder Knoten umfasst eine Containerlaufzeitumgebung wie etwa Docker sowie einen Agenten (Kubelet), der mit dem Masterknoten kommuniziert. Ein Knoten kann eine in einer Cloud ausgeführte virtuelle Maschine (VM) oder ein Bare-Metal-Server in einem Rechenzentrum sein.

Wie in Abbildung 1.7 dargestellt, umfasst jeder Knoten folgende Elemente:

Containerlaufzeitumgebung

Die Containerlaufzeitumgebung verwaltet die Lebenszyklen aller in dem Knoten ausgeführten Container. Wenn die Ausführung eines Pods auf dem Knoten eingeplant ist, lädt die Laufzeitumgebung die in diesem Pod genannten Images aus der Registry. Wenn ein Pod beendet wird, deaktiviert die Laufzeitumgebung die Container, die zu diesem Pod gehören. Kubernetes kann mit jeder OCI-konformen Containerlaufzeitumgebung kommunizieren, beispielsweise mit Docker und CRI-O.

OCI (Open Container Initiative) ist ein Standard, der die Laufzeit- und die Image-Spezifikation

definiert und damit zur Standardisierung von Containerlaufzeitumgebungen und Image-Formaten beiträgt.

Kubelet

Das Kubelet ist der Kubernetes-Agent, der für die Interaktion mit der Containerlaufzeitumgebung zuständig ist und beispielsweise Prozesse zum Aufrufen, Beenden und Pflegen von Containern ausführt.

Jedes Kubelet überwacht auch die Status der Pods. Wenn der Status eines Pods vom definierten gewünschten Status abweicht, kann dieser Pod auf demselben Knoten neu gestartet werden. Der Status des Knotens wird im Abstand von wenigen Sekunden anhand von Taktmeldungen an den Masterknoten übermittelt. Wenn der Masterknoten einen Knotenfehler erkennt, registriert der Replikationscontroller diese Statusänderung und plant die Pods auf anderen Knoten ein, die fehlerfrei arbeiten.

Kube-Proxy

Der `Kube-Proxy` fungiert als Netzwerkproxy und Load Balancer. Er orchestriert das Netzwerk und routet Serviceanforderungen an die richtigen Pods. Als Entscheidungsgrundlage dienen dabei der Servicename und die Portnummer der jeweiligen eingehenden Serviceanforderung. Der Kube-Proxy nutzt zudem betriebssystemspezifische Netzwerkfunktionen, indem er die in `iptables` definierten Richtlinien und Regeln anpasst. Jeder `Kube-Proxy` kann in Netzwerkschichten wie etwa Calico oder Flannel integriert werden.

Protokollschicht

Durch häufige Protokollierung erfasst der Orchestrator die Ressourcennutzungs- und Leistungskennzahlen der Container auf den einzelnen Knoten, unter anderem in Bezug auf die CPU, den Speicher, die Dateien und das Netzwerk. Die Cloud Native Computing Foundation hostet eine Softwarekomponente, die eine einheitliche Protokollschicht namens Fluentd zur Nutzung für Kubernetes und andere Orchestrators bereitstellt. Diese Komponente generiert Kennzahlen, um den Kubernetes-Hauptcontroller über verfügbare Clusterressourcen und den Zustand der gesamten Infrastruktur zu informieren.

Add-ons

Kubernetes unterstützt weitere Dienste in Form von Add-ons. Diese optionalen Dienste, wie etwa das Dashboard, werden ebenso wie andere Anwendungen bereitgestellt, sind jedoch in andere Kernkomponenten des jeweiligen Knotens integriert, beispielsweise in die Protokollschicht oder den `Kube-Proxy`. So nutzt zum Beispiel das Dashboard Kennzahlen aus dem Kubelet, um die Ressourcennutzung grafisch darzustellen. Und das auf `kube-dns` oder CoreDNS basierende Add-on DNS unterstützt den `Kube-Proxy` durch Namensauflösung.

Workloads

Während die Steuerungsebene und die Worker-Knoten die Kerninfrastruktur des Clusters bilden, handelt es sich bei den Workloads um die in Kubernetes bereitgestellten containerisierten Anwendungen.

Wenn die Entwickler einen Microservice entwickelt und getestet haben, packen sie ihn als Container, also als die kleinste, als Pod gepackte Bereitstellungseinheit. Mehrere Container, die zur gleichen Anwendung gehören, werden in Kubernetes zusammengefasst, gepackt, bereitgestellt und verwaltet.

Kubernetes stellt Primitiven für die Ausführung dieser Microservices bereit und übernimmt gleichzeitig die kontinuierliche Skalierung, Inventarisierung und Überwachung des Zustands. Die logische Trennung verschiedener Anwendungen voneinander erfolgt typischerweise mithilfe von Namespaces. Diese fungieren als logische Cluster, die die Grenzen und die Größen aller zu einer Anwendung gehörenden Ressourcen und Dienste genau festgelegen.

In einem Namespace werden die folgenden Kubernetes-Primitiven bereitgestellt:

Pods

Ein Pod ist die grundlegende Ausführungseinheit einer Kubernetes-Anwendung, die kleinste und einfachste Einheit im Kubernetes-Objektmodell und das kleinste, planbare Element in einer Kubernetes-Anwendung. Wenn Kubernetes ein Betriebssystem ist, dann entspricht ein Pod mehreren auf einem Cluster ausgeführten Prozessen, wobei jeder Prozess einem Container zugeordnet ist.

Der Pod wird in Kubernetes als Kerneinheit des Workload-Managements genutzt und fungiert zugleich als logische Grenze für Container, die sich einen Ausführungskontext und entsprechende Ressourcen teilen. Die Gruppierung von zusammengehörenden Containern zu Pods trägt zur Bewältigung der Herausforderungen bei, die beim Übergang von der Virtualisierung der ersten Generation zur Containerisierung entstanden, denn durch diese Gruppierung können mehrere voneinander abhängige Prozesse gleichzeitig ausgeführt werden.

Jeder Pod umfasst einen oder mehrere Container, die via Interprozesskommunikation (IPC) miteinander kommunizieren und möglicherweise einen Speicher und einen Netzwerkstack gemeinsam nutzen. In Anwendungsszenarien, in denen mehrere Container (beispielsweise ein Webservercontainer und ein Cachecontainer) gekoppelt und am selben Ort ausgeführt werden müssen, bietet es sich an, diese Container in einen Pod zu packen. Das horizontale Skalieren eines Pods kann entweder manuell oder mithilfe einer Richtlinie erfolgen, die mit der Funktion Horizontal Pod Autoscaling (HPA) definiert wird. Mit dieser Methode wird die Zahl der Pods in einer Bereitstellungsumgebung proportional und unter Berücksichtigung der Ressourcen-verfügbarkeit erhöht.

Pods ermöglichen eine funktionale Trennung zwischen der Entwicklung und der Bereitstellung. Während sich die Entwickler auf die Programmierung konzentrieren, können die Betreiber den Fokus auf das große Ganze setzen und entscheiden, welche zusammengehörenden Container in einer gemeinsamen Funktionseinheit verpackt werden sollten. Daraus ergibt sich der optimale Grad an Portabilität, da es sich bei einem Pod lediglich um ein Manifest aus einem oder mehreren gemeinsam verwalteten Container-Images handelt.

Controller

In Kubernetes erweitern Controller Pods um zusätzliche Kapazitäten wie etwa den gewünschten Konfigurationszustand und Laufzeiteigenschaften.

Eine Bereitstellung ermöglicht deklarative Aktualisierungen für Pods. Durch Protokollierung wird sichergestellt, dass bei allen an der Bereitstellung beteiligten Pods stets der gewünschte Zustand beibehalten wird. Jede Bereitstellung verwaltet ein ReplicaSet mit einer stabilen Menge replizierter Pods, die dem gewünschten Zustand entsprechend ausgeführt werden.

Durch die Skalierbarkeit, die Bereitstellungshistorie und die Rollbackfunktionen verleihen Bereitstellungen Pods PaaS-ähnliche Eigenschaften. Wenn eine Bereitstellung mit mindestens zwei Kopien konfiguriert ist, stellt Kubernetes sicher, dass immer mindestens zwei Pods ausgeführt werden und das System somit fehlertolerant ist. Selbst wenn ein Pod mit nur einer Kopie bereitgestellt wird, wird dringend empfohlen, einen Bereitstellungscontroller anstelle einer schlichten Pod-Spezifikation zu verwenden.

Ein StatefulSet ist vergleichbar mit einer Bereitstellung, wurde aber für Pods konzipiert, die Persistenz, einen wohldefinierten Identifier und eine garantierte Erstellungsreihenfolge benötigen. Für Workloads wie etwa Datenbankcluster erstellt ein StatefulSet-Controller ein hochverfügbares Podset in einer vorgegebenen Reihenfolge und mit vorhersehbarer Benennungskonvention. Statusbehaftete Workloads, die hochverfügbar sein müssen, beispielsweise Cassandra, Kafka, SQL Server und ZooKeeper, werden in Kubernetes als StatefulSets bereitgestellt.

Mit einem DaemonSet-Controller kann erzwungen werden, dass ein Pod auf jedem Knoten eines Clusters ausgeführt wird. Da Kubernetes in neu bereitgestellten Worker-Knoten automatisch ein DaemonSet einplant, ist dieses perfekt geeignet, um die Knoten zu konfigurieren und auf Workloads vorzubereiten. Wenn beispielsweise ein vorhandenes Network File System (NFS) oder eine Gluster-Dateifreigabe vor der Bereitstellung einer Workload auf dem Knoten gemountet werden soll, sollte der Pod als DaemonSet gepackt und bereitgestellt werden. Auch für Monitoringagenten empfiehlt sich ein DaemonSet, um sicherzustellen, dass sie auf jedem Knoten laufen.

Bei Stapelverarbeitungs- und Zeitplanungsaufträgen können Pods für einen Run-to-Completion-Job oder einen CronJob gepackt werden. Ein Job erstellt einen oder mehrere Pods und stellt sicher, dass eine festgelegte Anzahl von ihnen erfolgreich abgeschlossen wird. Für Run-to-Completion konfigurierte Pods führen die Jobs aus und werden danach beendet, während ein CronJob den Job basierend auf dem Zeitplan ausführt, der im Crontab-Format definiert wurde.

Controller legen die Lebenszyklen von Pods in Abhängigkeit von den Workload-Eigenschaften und dem zugehörigen Ausführungskontext fest.

Dienste und Serviceinventarisierung

Das Servicemodell in Kubernetes stellt die einfachste und zugleich wichtigste Funktion für Microservices bereit: die Inventarisierung.

Jedem API-Objekt in Kubernetes (einschließlich der Knoten und der Pods) können Schlüssel-Wert-Paare zugeordnet sein, also zusätzliche Metadaten zur Identifizierung und Zusammenfassung von Objekten mit gemeinsamen Attributen oder Eigenschaften. In Kubernetes werden diese Schlüssel-Wert-Paare als Label und Annotations (Anmerkungen) bezeichnet. Bei der Serviceinventarisierung werden diese Label und Selektoren genutzt, um einen Dienst mit einem Pod-Set zu assoziieren.

Ein einzelner Pod beziehungsweise ein einzelnes ReplicaSet kann internen oder externen Clients über Dienste zur Verfügung gestellt werden, die ein Pod-Set mit einem bestimmten Kriterium assoziieren. Jeder Pod, dessen Label zu dem im Dienstmanifest definierten Selektor passen, wird automatisch vom Dienst erkannt. Diese Architektur bietet einen flexiblen, lose gekoppelten Mechanismus zur Serviceinventarisierung.

Wenn ein Pod erstellt wird, wird ihm eine IP-Adresse zugewiesen, auf die nur innerhalb des Clusters zugegriffen werden kann. Es gibt jedoch keine Garantie, dass die IP-Adresse eines Pods während seines gesamten Lebenszyklus gleich bleibt. Kubernetes kann Pods zur Laufzeit verschieben oder neu instanziieren, was zu einer neuen IP-Adresse für den Pod führt.

Zum Ausgleich sorgen Dienste dafür, dass der Datenverkehr immer zum richtigen Pod im Cluster geroutet wird, unabhängig davon, auf welchem Knoten er eingeplant ist. Jeder Dienst stellt eine IP-Adresse und eventuell einen DNS-Endpunkt bereit – und beide bleiben immer unverändert. Interne und externe Verbraucher, die mit einem Pod-Set kommunizieren müssen, verwenden entweder die IP-Adresse des Dienstes oder seinen besser bekannten DNS-Endpunkt. Auf diese Weise fungiert der Dienst als Leim zwischen den Pods.

Eine Bereitstellung entscheidet anhand der Label und Selektoren, welche Pods an einem Skalierungsvorgang beteiligt sein werden. Jeder Pod, dessen Label zu dem im Dienst definierten Selektor passt, wird an seinem Endpunkt zur Verfügung gestellt. Anschließend sorgt ein Dienst für einen elementaren Lastenausgleich, indem er den Datenverkehr an die in Frage kommenden

Pods leitet.

Ein Selektor ist eine Art Kriterium, nach dem Kubernetes Objekte abfragt, die einem Labelwert entsprechen. Mit dieser eleganten Methode ist es möglich, Objekte locker miteinander zu koppeln. So können neue Objekte generiert werden, deren Label dem Wert des Selektors entsprechen. Label und Selektoren bilden den wichtigsten Gruppierungsmechanismus in Kubernetes zur Identifizierung von Komponenten, auf die ein Vorgang angewendet wird.

Zur Laufzeit können Pods mithilfe von ReplicaSets skaliert werden, um sicherzustellen, dass jede Bereitstellung immer die gewünschte Anzahl an Pods ausführt. Jedes ReplicaSet enthält stets ein vorab definiertes Podset. Wenn eine Bereitstellung einen Skalierungsvorgang einleitet, empfangen die von diesem Vorgang neu erstellten Pods sofort Daten.

Kubernetes bietet drei Schemata für die Servicebereitstellung:

1. **ClusterIP:** Für die Kommunikation zwischen Pods in einem Cluster. So steht beispielsweise ein Datenbank-Pod, der über einen ClusterIP-basierten Dienst zugänglich ist, den Webserver-Pods zur Verfügung.
2. **NodePort:** Dieses Schema wird genutzt, um einen Dienst auf allen Knoten eines Clusters über denselben Port zur Verfügung zu stellen. Ein interner Routingmechanismus sorgt dafür, dass alle Serviceanfragen an den entsprechenden Knoten des jeweiligen Pods weitergeleitet werden. Dieser Mechanismus wird typischerweise für Dienste mit externen Verbrauchern verwendet.
3. **LoadBalancer:** Das Schema LoadBalancer erweitert den NodePort-Dienst durch Load Balancer auf den Ebenen vier (L4) und sieben (L7). Dieses Schema wird häufig für Cluster in Public-Cloud-Umgebungen verwendet, die die automatisierte Bereitstellung von softwaredefinierten Load Balancern unterstützen.

Wenn mehrere Dienste einen Load Balancer oder einen externen Endpunkt gemeinsam nutzen müssen, wird die Verwendung eines Ingress-Controllers empfohlen. Dieser verwaltet den externen Zugriff auf die Dienste in einem Cluster – typischerweise über HTTP – und bietet Lastenausgleich, Secure Socket Layer-(SSL-)Beendigung und namenbasiertes virtuelles Hosting.

Ingress wird immer häufiger zur Ausführung von Workloads in Kubernetes genutzt. Mit Ingress können mehrere Microservices einer Anwendung denselben Endpunkt nutzen, der über einen Load Balancer, ein API-Gateway oder einen Application Delivery Controller (ADC) erreichbar ist.

Netzwerk und Speicher

Rechenleistung, Netzwerk und Speicher bilden die Grundlagen für jeden Infrastrukturdienst. In Kubernetes repräsentieren die Knoten die Rechenleistung, die den in den Clustern ausgeführten Pods bereitgestellt wird. Die Netzwerk- und Speicherdienste werden von softwaredefinierten, containerbasierten Plug-ins bereitgestellt, die speziell für Kubernetes entwickelt wurden.

Die Netzwerkkomponente unterstützt die Kommunikation von Pod zu Pod, Knoten zu Pod, Pods zu Diensten und externen Clients zu Diensten. Zur Implementierung der Netzwerkkomponente nutzt Kubernetes ein Plug-in-Modell. Das standardmäßig verwendete Netzwerk-Plug-in heißt Kubenet und lässt sich leicht konfigurieren. Es wird gewöhnlich in Single-Node-Umgebungen oder in Kombination mit einem Cloud-Anbieter eingesetzt, der Routingregeln für die Kommunikation zwischen den Knoten konfiguriert.

Kubernetes kann zahlreiche Plug-ins unterstützen, die auf der Spezifikation Container Network Interface (CNI) basieren. CNI definiert die Netzkonnektivität von Containern und übernimmt das Management der Netzwerkressourcen, wenn Container gelöscht werden. Es gibt viele CNI-Implementationen, darunter Calico, Cil3ium, Contiv und Weave Net. Die CNI-Spezifikation unterstützt auch virtuelles Networking in Public Clouds, sodass die Netzwerktopologie und die Subnetze auf die Kubernetes-Cluster ausgedehnt werden können.

Einige der CNI-kompatiblen Netzwerk-Plug-ins, wie etwa Calico, implementieren Richtlinien, die strenge Routingvorgaben durch die Isolierung von Pods durchsetzen. Sie nutzen firewallartige Regeln für die Pods und Namespaces eines Kubernetes-Clusters.

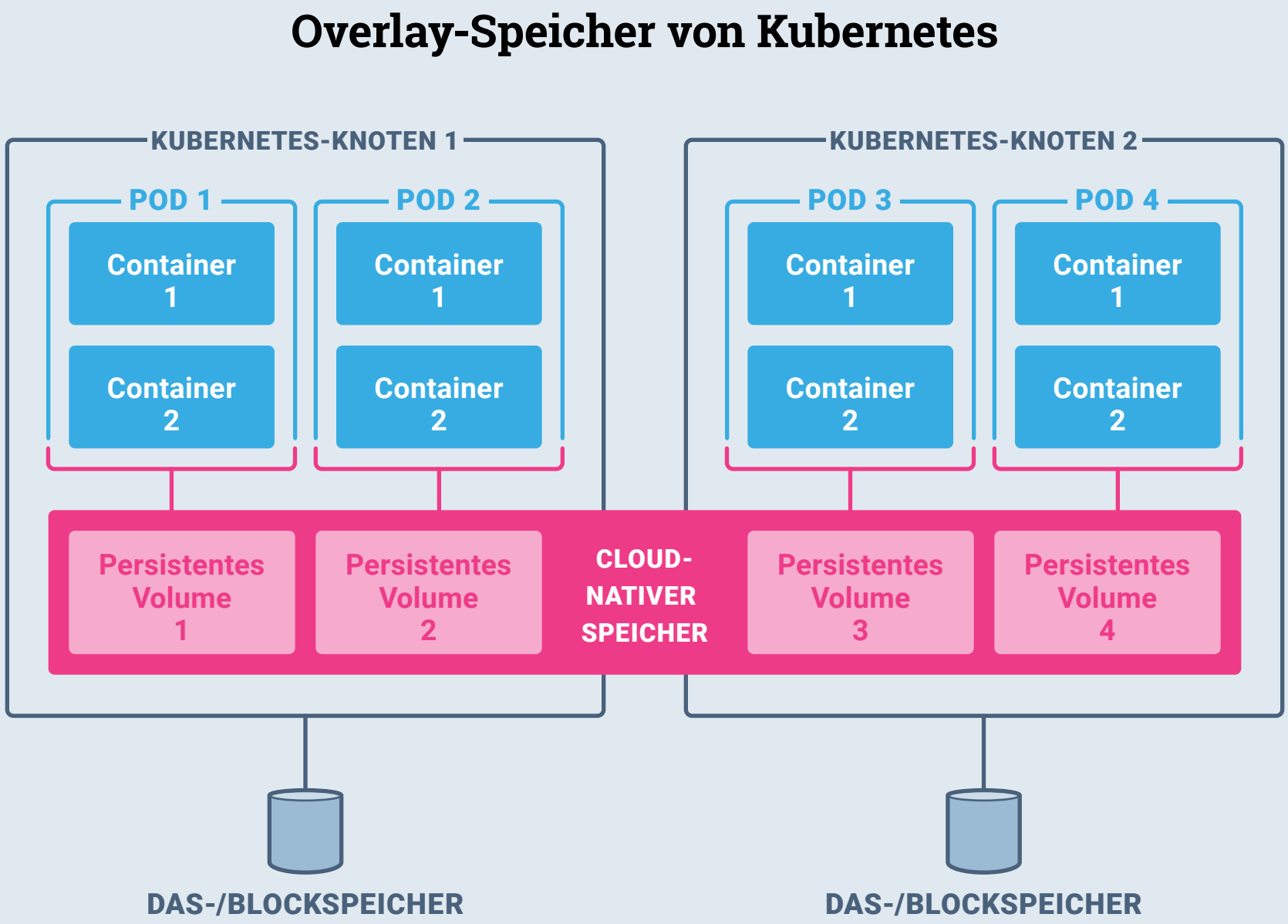
Persistenter Speicher wird in Kubernetes durch den Zugriff auf persistente Volumes unterstützt. Diese Volumes werden Pods zugewiesen, wenn diese mit einem Persistent Volume Claim (PVC) persistenten Speicher anfordern. Speicheradministratoren stellen Speicher bereit, indem sie persistente Volumes auf einem vorhandenen NAS (Network Attached Storage), SAN (Storage Area

Network), DAS (Direct Attached Storage), SSD (Solid State Drive), NVMe (Non-Volatile Memory Express) oder Flash Disk Array einrichten. Entwickler and DevOps-Teams können mit PVCs beschreiben, wie der persistente Speicher für jeden ihrer Pods beschaffen sein soll.

In Kubernetes sind Speicherprimitiven für den Zugriff auf Speicher auf vorhandenen Knoten enthalten. Eine dieser Primitiven ist ein Volume-Typ, der Pods Zugang zum zugrunde liegenden Speicher bietet. Beispiele für Volume-Typen sind `emptyDir` und `hostPath`. Diese werden für bestimmte Anwendungsszenarien verwendet: `emptyDir` zur temporären Speicherung und `hostPath`, um lokale Volumes für Pods verfügbar zu machen. Aufgrund der engen Kopplung mit dem Knoten sind sie jedoch weder hochverfügbar noch fehlertolerant. Bei der Overlay-Speichertechnik (siehe Abbildung 1.8) werden Poolspeicher-Volumes aus Blockspeichern, NAS und SAN überlagert, um Kubernetes-Objekten externen Speicherplatz bereitzustellen.

Zudem gibt es in Kubernetes Plug-ins für Speicheranbieter, mit denen diese containerisierten Workloads Zugang zu ihren Plattformen – und damit hochverfügbare, containerbasierte

Abb. 1.8: *Bereitstellung von Speicher für Pods und Container*



Quelle: Janakiram MSV

Speicherfunktionen – anbieten können. Die Open-Source-Upstreamdistributionen von Kubernetes umfassen Plug-ins für Blockspeicher von Public-Cloud-Anbietern, verteilte Dateisysteme basierend auf NFS und GlusterFS sowie einige kommerzielle Speicherplattformen. Speicheradministratoren erstellen Speicherklassen für jede Art von Speicher-Engine in Abhängigkeit von deren Performance und Geschwindigkeit. Diese Speicherklassen dienen dann als Ausgangspunkt für die Einrichtung von Persistent Volumes und Persistent Volume Claims für unterschiedliche Workloads. So kann beispielsweise ein relationales Datenbankmanagementsystem (RDBMS) mit einer Speicherklasse mit mehr Ein- und Ausgabevorgängen pro Sekunde (IOPS) assoziiert werden, während ein Content Management System (CMS) über eine andere Speicherklasse auf eine verteilte Speicher-Engine zugreift.

Angelehnt an CNI hat die Kubernetes-Community Spezifikationen für die Speicherung im Container Storage Interface (CSI) definiert, um den Weg für eine standardisierte, portierbare Implementierung und Nutzung von Speicherdiensten durch containerisierte Workloads zu bahnen.

Hyperskalierung mit Kubernetes

Da Kubernetes aus Borg weiterentwickelt wurde, ist es von Haus aus gut für die Hyperskalierung von Workloads geeignet. Dank seiner modernen Architektur unterstützt es die optimale Ausnutzung der Infrastrukturressourcen. Vorhandene Cluster können problemlos und fast ohne Konfigurationsänderungen um zusätzliche Worker-Knoten erweitert werden. Workloads können die CPU und die Speicherressourcen dieser neuen Knoten unverzüglich nutzen.

Die Zusammenfassung von Containern zu Pods und deren Verwaltung als Bereitstellungs- und Skalierungseinheit steigert die Leistung. Wenn beispielsweise ein Webserver und mehrere Cachecontainer im gleichen Pod untergebracht werden, sinkt die Latenz und die Performance steigt. Alle Container in einem Pod nutzen denselben Ausführungskontext und können daher via Interprozesskommunikation (IPC) effizient miteinander kommunizieren.

Pods, die zum gleichen ReplicaSet und zur gleichen Bereitstellung gehören, lassen sich schnell skalieren. Es dauert nur wenige Sekunden, bis eine Bereitstellung auf Hunderte von Pods skaliert ist. Die Pods werden in Abhängigkeit von den Ressourceneigenschaften und dem gewünschten

Konfigurationszustand auf den Knoten geplant. Durch die Konfiguration eines Horizontal Pod Autoscalers (HPA) kann Kubernetes eine Bereitstellung automatisch hoch- oder herunterskalieren.

In elastischen Infrastrukturumgebungen kann Kubernetes den Cluster Autoscaler nutzen, um Knoten zu einem Cluster hinzuzufügen oder aus ihm zu entfernen. Mit dieser Methode und einem HPA lässt sich die dynamische Autoskalierung sowohl der Workloads als auch der Infrastruktur effizient steuern.

Der schlanke Netzwerkstack und die Serviceinventarisierung von Kubernetes wurden für die Skalierung konzipiert. Sie können Zehntausende von Endpunkten verwalten, die von Diensten zur internen und externen Nutzung angeboten werden.

Das Kubernetes-Ökosystem und die Kubernetes-Community arbeiten kontinuierlich an der Weiterentwicklung der Plattform, um die Verarbeitung von Workloads in der Größenordnung des Internets weiter zu verbessern.

Der Einsatz von Kubernetes

Kubernetes wird in Produktionsumgebungen als Containerorchestrierungssystem, als PaaS und als Kerninfrastruktur zur Verwaltung von cloudbasierten Anwendungen eingesetzt. Diese Anwendungsfälle schließen sich nicht gegenseitig aus. So können Betreiber mit Kubernetes beispielsweise die gesamte Verwaltung von Anwendungslebenszyklen an eine PaaS-Schicht delegieren. Parallel dazu können sie eine eigenständige Kubernetes-Bereitstellung zur Verwaltung von Anwendungen implementieren, die über die vorhandene CI/CD-Toolchain bereitgestellt werden. Kunden, die bei der Anwendungsentwicklung bei null anfangen, können Kubernetes nutzen, um moderne, auf Microservices basierende, cloudnative Anwendungen mit Features wie rollierenden Upgrades und Canary-Bereitstellungen zu erstellen.

Kunden, die Kubernetes bewerten oder für Produktionsbereitstellungen verwenden möchten, haben zahlreiche Distributionen und Varianten zur Auswahl.

Unten finden Sie eine Liste der wichtigsten auf dem Markt erhältlichen Kubernetes-Distributionen.

Upstreamdistributionen

Die in GitHub verfügbare Open-Source-Upstreamdistribution von Kubernetes kann in Rechenzentren, Public- und Private-Cloud-Umgebungen genutzt werden. Die Codebasis umfasst die Kernbausteine und die wichtigsten Komponenten für die Ausführung von Workloads auf Kubernetes.

Zudem gibt es in Kubernetes ein Bereitstellungstool namens kubeadm, das sich problemlos installieren lässt und die Konfiguration von Clustern basierend auf CentOS, Red Hat Enterprise Linux, SUSE, Ubuntu und anderen Linux-Distributionen unterstützt.

Kunden können auch automatisierte Bereitstellungstools wie etwa kops, kubespray und Rancher Kubernetes Engine nutzen, um Cluster zu installieren und zu konfigurieren. Diese Tools bieten mehr oder weniger Unterstützung für die Anpassung der Konfiguration der Container-laufzeitumgebungen sowie der Netzwerk- und Speicher-Plug-ins.

Die Upstreamdistribution ist kostenlos erhältlich, transparent und unterstützt einen vollständig konfigurierbaren Ansatz zum Installieren von Kubernetes. Mitarbeiter und Organisationen, die den Einsatz von Kubernetes mit dieser Distribution planen, sollten jedoch über fundiertes Wissen verfügen.

Kommerzielle Distributionen

Einige Firmen bieten eine anpassbare und optimierte Variante von Kubernetes mit Professional Services und Support an. Sie nutzen dabei ein bewährtes Modell zur Kommerzialisierung von Open-Source-Software mit Serviceangeboten.

Kommerzielle Distributionen beschleunigen nicht nur die Konfigurierung und Bereitstellung von Kubernetes-Clustern, sondern versprechen auch Patches, Hot Fixes und nahtlose Upgrades auf neuere Versionen der Plattform.

Canonical, D2iQ, HPE, Mirantis, Rancher, VMware und andere Firmen bieten kommerzielle Kubernetes-Distributionen an.

Containers-as-a-Service

Kubernetes ist auch als vollständig gehostete, verwaltete Plattform verfügbar. Anbieter von Containers-as-a-Service (CaaS) bieten komplette Kubernetes-Umgebungen mit Service-Level Agreements (SLAs) für Unternehmensrechenzentren, Colocation-Infrastrukturen und Public-Cloud-Umgebungen an.

Nahezu alle großen Cloud-Anbieter haben inzwischen eine CaaS-Option in ihrem Serviceportfolio. Alibaba Container Service for Kubernetes, Amazon EKS, Azure AKS, DigitalOcean Kubernetes, Google Kubernetes Engine, Huawei Cloud Container Engine und IBM Kubernetes Service sind Beispiele für CaaS in Public Clouds.

Mirantis, NetApp und Platform 9 bieten CaaS für Rechenzentren und Private-Cloud-Umgebungen an.

Platform-as-a-Service

Kunden nutzen PaaS vorrangig, um ihre Entwicklungs- und Bereitstellungsumgebungen zu standardisieren. Mit einer Kubernetes-basierten PaaS sind sie in der Lage, sowohl herkömmliche Branchen- als auch neuartige Anwendungen zu entwickeln. Etablierte PaaS-Anbieter nutzen Kubernetes inzwischen, um umfassende Plattformfunktionen für ihre Firmenkunden bereitzustellen.

Kubernetes-basierte PaaS-Lösungen bauen auf den bekannten Kernfunktionen der Containerorchestrierung auf und verwalten den gesamten Lebenszyklus containerisierter Anwendungen.

Red Hat OpenShift und VMware Tanzu Kubernetes Grid sind Beispiele für PaaS-Lösungen, die auf Kubernetes basieren.

Kubernetes als universelle Steuerungsebene

Kubernetes kristallisiert sich zudem als eine der besten Steuerungsebenen im Kontext moderner Anwendungen und Infrastrukturen heraus. Der leistungsstarke Scheduler, der ursprünglich entwickelt wurde, um Pods auf geeigneten Knoten zu platzieren, hat noch ganz andere

Fähigkeiten. Er kann viele Probleme lösen, die in herkömmlichen verteilten Systemen auftreten.

Kubernetes ist inzwischen die bevorzugte Steuerungsebene für die Planung und Verwaltung von Aufgaben (Jobs) in hochverteilten Umgebungen, darunter die Bereitstellung von virtuellen Maschinen auf physischen Hosts, die Platzierung von Containern auf Edge-Geräten und sogar die Erweiterung der Steuerungsebene auf andere Scheduler wie etwa serverlose Umgebungen.

Von Bare-Metal-Servern über virtuelle Maschinen und Geräte im Internet der Dinge (IoT) bis hin zu verwalteten Cloud-Diensten – Kubernetes wurde über Container und Pods hinaus entwickelt, um auch andere Bereitstellungs- und Planungsherausforderungen zu meistern.

Es folgen einige Beispiele:

Crossplane

Mit Crossplane soll das Infrastruktur- und Anwendungsmanagement mit demselben API-zentrierten, deklarativen Konfigurierungs- und Automatisierungsverfahren standardisiert werden, das erstmalig mit Kubernetes zum Einsatz kam. Es handelt sich dabei um eine einheitliche Steuerungsebene, die nahtlos in vorhandene Tools und Systeme integriert werden kann und eine problemlose Konfigurierung von Richtlinien, Kontingenten und Protokollberichten ermöglicht.

Crossplane fungiert als Brücke zwischen Kubernetes und herkömmlichen Workloads wie beispielsweise Datenbanken und sogar verwalteten Diensten in Public Clouds. DevOps-Teams können externe Ressourcen und native Kubernetes-Anwendungen mit der gleichen YAML-Spezifikation definieren. Dieses Verfahren unterstützt die Konfiguration als Code, da die Versionierung, die kontinuierliche Integration und die Bereitstellung auf Ressourcen außerhalb von Kubernetes ausgeweitet wird.

K3s

K3s von Rancher ist eine zertifizierte Kubernetes-Distribution für Produktions-Workloads, die in Umgebungen mit begrenzten Ressourcen (wie dem IoT und Edge-Bereitstellungen) ausgeführt werden.

K3s kann auf den meisten virtuellen Maschinen in Public-Cloud-Umgebungen und sogar auf einem Raspberry Pi bereitgestellt werden. Seine Architektur erfüllt sämtliche Anforderungen der Kubernetes-Konformitätstests der CNCF, ist aber sehr stark für nicht überwachte, entfernte Bereitstellungen auf ressourcenarmen Geräten optimiert.

Mit K3s wird Kubernetes schlank genug für Edge-Computing-Umgebungen.

KubeEdge

Auf der KubeCon+CloudNativeCon 2018 in Seattle stellte Huawei KubeEdge vor, das offizielle Projekt, mit dem Kubernetes und seine leistungsstarken Funktionen in Edge-Umgebungen Einzug halten sollten.

KubeEdge basiert auf der Intelligent Edge Fabric (IEF) von Huawei – einer kommerziellen Edge-Plattform für das IoT, die auf den IoT-PaaS-Optionen von Huawei basiert. Ein großer Teil der IEF wurde für KubeEdge angepasst und als Open Source bereitgestellt. Version 1.3 von KubeEdge läuft stabil und ist für die wichtigsten IoT- und Edge-Anwendungsfälle geeignet. Sie kann auf einer unterstützten Linux-Distribution oder einem ARM-Gerät wie etwa einem Raspberry Pi installiert werden.

Das KubeEdge-Projekt ist Teil der CNCF-Sandbox.

KubeVirt

KubeVirt, ein Add-on zur Verwaltung virtueller Maschinen auf Kubernetes, ermöglicht die parallele Ausführung von VMs und Containern in Kubernetes- oder OpenShift-Clustern. Über die CustomResourceDefinitions-(CRD-)API von Kubernetes erweitert es die Plattform um Ressourcentypen für VMs. KubeVirt-VMs werden in normalen Kubernetes-Pods ausgeführt, wo sie Zugriff auf das Standard-Pod-Netzwerk und dessen Speicher haben und anhand von standardmäßigen KubeVirt-Tools wie etwa `kubectl` verwaltet werden können.

KubeVirt ist ebenfalls Teil der CNCF-Sandbox.

Virtual Kubelet

Das Projekt Virtual Kubelet von Microsoft ist die interessanteste Erweiterung des Kubelet-Agenten und der Kubernetes-API. Der Virtual Kubelet ist ein Agent, der in einer externen Umgebung ausgeführt wird, die aber als Knoten innerhalb des Kubernetes-Clusters registriert ist. Über die Kubernetes-API erstellt der Agent einen Knoten. Mithilfe von Taints und Tolerations plant der Agent Pods in einer externen Umgebung, indem er deren systemeigene API aufruft.

Virtual Kubelet ist mit den Steuerungsebenen Azure Container Instances, Azure IoT Edge und AWS Fargate kompatibel.

Kubernetes hat zwar bescheiden mit Containerorchestrierung angefangen, sich dann aber schnell zum Betriebssystem der Wahl für Cloud- und Edge-Umgebungen entwickelt. Heute ist Kubernetes die Basis moderner Infrastrukturen in Rechenzentren sowie in Hybrid-, Public- und Multi-Cloud-Umgebungen.

Im nächsten Kapitel beschäftigen wir uns mit dem wachsenden Ökosystem von Kubernetes und der Cloud-Branche.

Selfservicearchitekturen und der Kubernetes-Operator für Cassandra

In den letzten zehn Jahren haben verteilte Datenbanken, wie das quelloffene Apache Cassandra, an Boden gewonnen – nicht zuletzt, weil sie Kubernetes und dessen horizontal skalierbare Architektur sehr gut ergänzen. Dieser Trend hat das Zeitalter der reinen Selfservicearchitekturen eingeläutet, über die wir in diesem Podcast mit Kathryn Erickson und Patrick McFadin von DataStax, dem Unternehmen hinter der gleichnamigen cloudnativen NoSQL-Datenplattform, sprechen.

DataStax stellt mit einem neuen Kubernetes-Operator, der die Datenbankebenen abstrahiert, einen Wegweiser durch Cassandra bereit, sodass sich Entwickler ganz auf das konzentrieren können, was sie am besten beherrschen: programmieren.

Selfservicearchitekturen geben Entwicklern die Freiheit, Daten auf neue und originelle Art zu nutzen. Mit den Worten von McFadin: „Datenbanken [stellen] nun eine geringere kognitive Belastung für Entwickler dar.“

Auf SoundCloud anhören **Auf YouTube ansehen**

Kathryn Erickson *leitet die Selfservicegeschäftsstrategie für DataStax.*

Patrick McFadin *ist bei DataStax Chief Evangelist for Apache Cassandra und Vice President of Developer Relations.*

KI-Beobachtbarkeit durchdringt die Komplexität von Kubernetes

Mit Kubernetes sind horizontal skalierbare Anwendungen in Multicloud-Umgebungen Wirklichkeit geworden. Allerdings ist dadurch auch die Komplexität für IT-Abteilungen enorm gestiegen.

Andreas Grabner, ein DevOps-Aktivist der Software-Intelligence-Plattform Dynatrace, hat kürzlich angemerkt, dass in Kubernetes-Umgebungen von Unternehmen „Milliarden gegenseitiger Abhängigkeiten zu berücksichtigen sind“. Ja, Milliarden!

In diesem Podcast erläutert Grabner, wie man die anderenfalls überwältigende Komplexität mit KI in den Griff bekommen kann. Außerdem sprechen wir über Anwendungsfälle für KI-Beobachtbarkeit für Betreiber und Entwickler sowie darüber, welchen Nutzen Telemetriedaten den für Kubernetes zuständigen Betriebsteams bringen und wie sich mit dem Aufkommen von Kubernetes die Kultur in Entwicklerteams verändert hat.

Auf SoundCloud anhören **Auf YouTube ansehen**

Andreas Grabner *ist ein DevOps-Aktivist bei Dynatrace. Er unterstützt Entwickler, Tester, Betriebs- und xOps-Mitarbeiter dabei, ihre Aufgaben mit Dynatrace effizienter zu erledigen.*

Die Anpassung datenorientierter Anwendungen an Kubernetes-Architekturen

Kubernetes ist, ähnlich wie Microservices, von Natur aus „lose gekoppelt“. Die Komponenten sind schlank, haben einen einzigen Zweck und werden unabhängig voneinander ausgeführt. Die Aktualisierung wechselseitig abhängiger, integrierter Code-komponenten im Rahmen von Anwendungsänderungen ist im Gegensatz dazu wesentlich komplexer.

Nanda Vijaydev, Distinguished Technologist und Lead Data Scientist bei HPE, erörtert die zahlreichen Aspekte, die bei der Verwaltung datenorientierter Anwendungen in dem lose gekoppelten Kontext von Kubernetes größere Sorgfalt und Aufmerksamkeit erforderlich machen.

„Wie geht man mit etwas, das als Datenmodell konzipiert ist, dynamisch um … und implementiert es in großem Maßstab?“, fragt sie. „Das ist der Punkt, an dem sich die Technologie weiterentwickelt.“

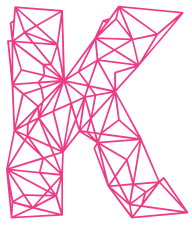
Durch die Übernahme von u. a. MapR Technologies, BlueData and Cloud Technology Partners (CTP) ist HPE nun noch besser in der Lage, Unternehmen bei der Bewältigung der Herausforderungen zu unterstützen, die bei der Kopplung von auf Kubernetes ausgeführ-ten, datenorientierten Workloads entstehen können. Mit der Softwareplattform von HPE können Benutzer die Datenverwaltung anders angehen und sie weniger als Kubernetes-spezifisches Problem, sondern vielmehr als Speicher- oder Netzwerkproblem behandeln.

Auf SoundCloud anhören

Nanda Vijaydev setzt in ihrer Tätigkeit als Lead Data Scientist und Distinguished Technologist bei HPE Technologien wie Kubernetes, TensorFlow, H2O und Spark für die Entwicklung von Lösungen für unternehmensspezifische Anwendungsfälle von maschinellem Lernen und Deep Learning ein.

KAPITEL 2

Aufbau des Kubernetes-Ökosystems



ubernetes hat den Wettlauf der Orchestrierungstools gewonnen und sich als Containermanagementplattform der Wahl etabliert. Es hat sich als die konsistente, einheitliche Infrastrukturlösung herauskristallisiert, die für Unternehmensrechenzentren, Public-Cloud-Plattformen, Hybrid-Cloud-Plattformen, Appliances in hyperkonvergenten Infrastrukturen und sogar Edge-Umgebungen geeignet ist.

Kubernetes ist die Grundlage einer modernen Infrastruktur für die Ausführung heutiger Workloads, die oft als cloudnative Anwendungen bezeichnet werden. In einer Kubernetes-Architektur wird die zentrale Infrastrukturebene durch Open-Source-Projekte und kommerzielle Software erweitert und ergänzt, die einen cloudnativen Stack bilden.

Dieses Kapitel gibt einen Überblick über cloudnative Technologien, gefolgt von einer detaillierten Diskussion der Komponenten des cloudnativen Stacks.

Wir analysieren den Stack vom Betriebssystem bis zum Entwicklererlebnis und werfen einen genaueren Blick auf die führenden Open-Source-Projekte und kommerziellen Angebote, die cloudnative Funktionalität für Unternehmen und Benutzer bereitstellen.

Dieses Kapitel befasst sich ausschließlich mit den Kernkomponenten von Kubernetes. Nicht behandelt werden beispielsweise Datenbanksysteme, die Teil des weiteren Kubernetes-Ökosystems sind. Insbesondere Databases-as-a-Service (DBaaS) ist in den letzten Jahren immer wichtiger geworden und umfasst heute Produkte wie Amazon Aurora, Azure SQL Database,

MongoDB Atlas und Redis Cloud Essentials. Eine Datenbank ist jedoch keine Kernkomponente des cloudnativen Stacks, sondern ein Workload. Die wichtigste Komponente, die Datenbanken und Stateful-Workloads unterstützt, ist der Speicher. Er ist ein wesentlicher Bestandteil des cloudnativen Stacks und wird im Folgenden behandelt.

Der Aufstieg von cloudnativen Lösungen und CaaS

Laut The Linux Foundation wird beim cloudnativen Computing ein Stack von Open-Source-Software verwendet, um Anwendungen als Microservices bereitzustellen. Dabei wird jeder Teil in einem eigenen Container ausgeführt und diese Container werden dynamisch orchestriert, um die Ressourcenauslastung zu optimieren.

Mit dem Begriff „cloudnativ“ werden containerbasierte Umgebungen beschrieben. Mit cloudnativen Technologien werden Anwendungen aus containerisierten Diensten aufgebaut, die als Microservices bereitgestellt und auf einer elastischen Infrastruktur über agile DevOps-Prozesse und Workflows der kontinuierlichen Bereitstellung verwaltet werden.

Eine der wichtigsten Eigenschaften von cloudnativen Lösungen ist die Portabilität, die nur möglich ist, wenn die Infrastruktur in allen genutzten Umgebungen konsistent ist. Damit wird Kubernetes zum kleinsten gemeinsamen Nenner der Infrastruktur und zur Grundlage des cloudnativen Stacks.

Dennoch benötigen Entwickler und DevOps-Experten zusätzliche Software, um moderne Anwendungen bereitzustellen, zu skalieren und zu verwalten. Plattformanbieter wie Red Hat und VMware bieten vollständige Plattformen auf der Grundlage von Kubernetes an. Public-Cloud-Anbieter – wie Amazon Web Services (AWS), Google Cloud Platform (GCP) und Microsoft Azure – bieten Kubernetes-basierte Managed Services an, die auf vorhandener Computing-, Speicher- und Netzwerkinfrastruktur ausgeführt werden.

Integrierte, Kubernetes-basierte Containermanagementplattformen stellen eine neue Kategorie von Modellen zur Anwendungsbereitstellung dar: Containers-as-a-Service (CaaS). Ähnlich wie eine Platform-as-a-Service (PaaS) kann die Containermanagementplattform hinter der Firewall

eines Unternehmensrechenzentrums bereitgestellt oder als Managed- Cloud-Service-Angebot genutzt werden.

Mit CaaS als gemeinsamer Grundlage für das Rechenzentrum und die Public Cloud können Unternehmen hybride Anwendungen erstellen, die interne Ressourcen sicher mit der Public Cloud verbinden. Zudem entwickelt CaaS sich inzwischen zusehends zu einem Katalysator für Hybrid- und Multi-Cloud-Bereitstellungen. Entwickler und Betreiber können Anwendungen ganz einfach zwischen verschiedenen Umgebungen verschieben.

Wichtige Eigenschaften einer Containermanagementplattform

Unabhängig davon, wo sie bereitgestellt wird, muss eine Containermanagementplattform die folgenden Anforderungen erfüllen:

- **Konsistente Plattform:** Entwickler und Betreiber erwarten in Public-Cloud- und On-Premises-Umgebungen ein einheitliches Erlebnis.
- **DevOps-Prozesse:** Eine Integration in bewährte DevOps-Praktiken zur schnellen Softwarebereitstellung sollte möglich sein.
- **Sicherheit:** Containermanagementplattformen müssen für die Sicherheit von Infrastruktur und Anwendungen sorgen. Sie sollten vor dem Bereitstellen von Anwendungen Schwachstellen erkennen und die Infrastruktur fortlaufend auf mögliche Kompromittierungen überwachen.
- **Zuverlässige Infrastruktur:** Die Plattform sollte ein auf Service-Level-Agreements (SLAs) basierendes Dienstbereitstellungsmodell unterstützen, das maximale Verfügbarkeit der Infrastruktur und der Plattform vorsieht.
- **Hohe Verfügbarkeit von Workloads:** Neben der Infrastruktur müssen auch die auf der Plattform bereitgestellten Geschäftsanwendungen hochverfügbar sein.
- **Beobachtbarkeit:** Die Plattform sollte Einblicke in die Infrastruktur, Ressourcen und Anwendungen bieten, indem sie Kennzahlen, Ereignisse, Protokolle und Traces aus dem gesamten Stack erfasst und an einem zentral zugänglichen Ort speichert.

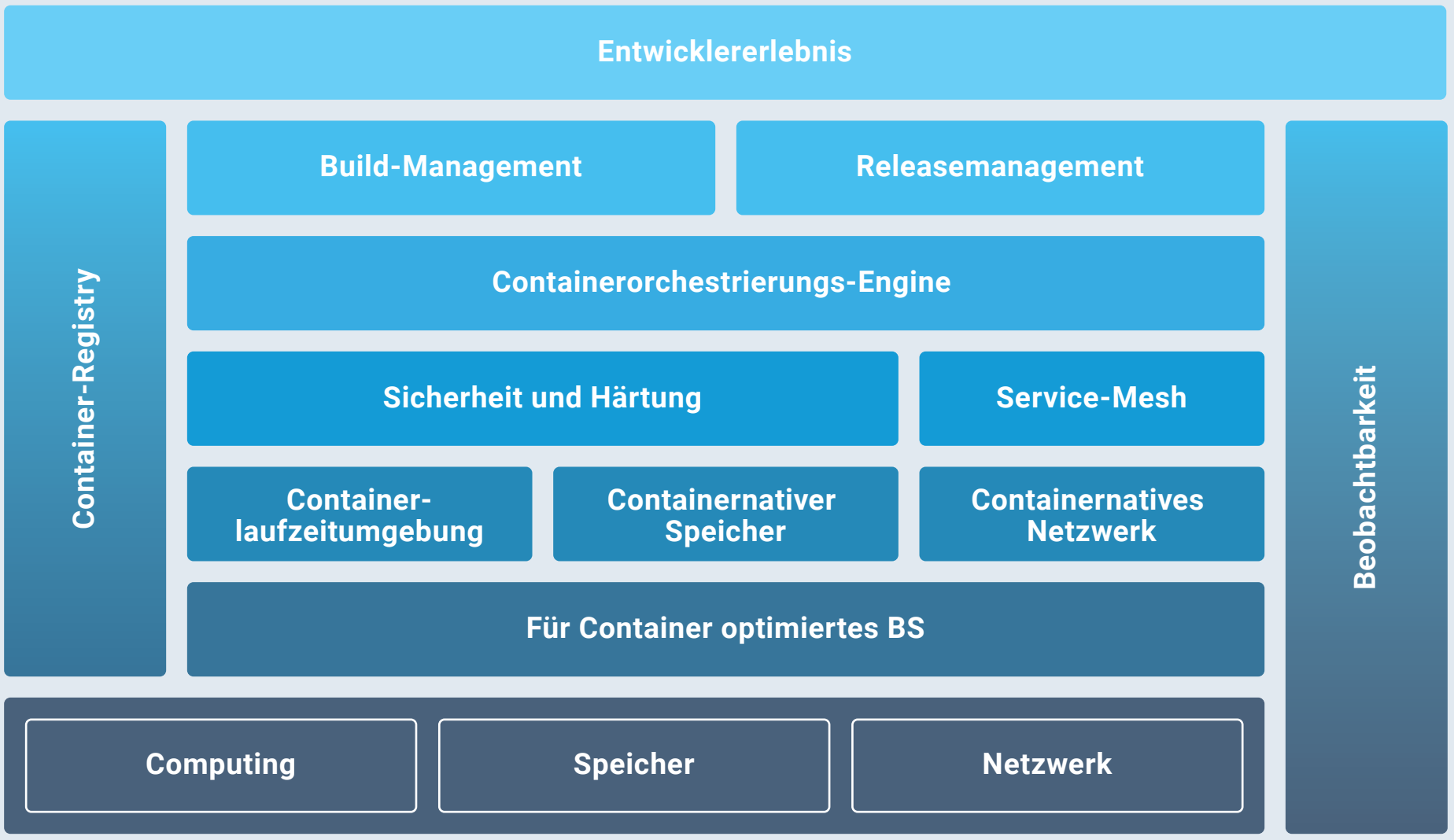
- **Mandantenfähigkeit und richtlinienbasierte Verwaltung:** Optional sollte die Containermanagementplattform die Mandanten, die sie nutzen, völlig voneinander isolieren. Die Bereitstellung und Verwaltung von Anwendungen sollten klar definierten Richtlinien unterliegen.

Überblick über eine Containermanagementplattform

Ein moderner cloudnativer Stack, der als integrierte Containermanagementplattform bereitgestellt wird, besteht aus mehreren Komponenten. Einige davon sind als Open-Source-Projekte verfügbar, andere sind kommerzielle Angebote von unabhängigen Softwareanbietern.

Die unterste Ebene besteht aus der physischen Infrastruktur des Clusters (Computing-, Speicher- und Netzwerkkomponenten). Die Plattform enthält außerdem verschiedene Abstraktionsebenen, um diese physische Infrastruktur optimal zu nutzen.

Abb. 2.1: *Der cloudnative Stack*



Quelle: Janakiram MSV

Sehen wir uns jede dieser Ebenen genauer an.

Für Container optimiertes Betriebssystem

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Fedora CoreOS	Red Hat	Flatcar Container Linux	Nicht eingereicht
Talos	Talos Systems	RancherOS	Nicht eingereicht

Mit Containern wird die Rolle des Betriebssystems (Operating System, OS) neu definiert. Da viele Aufgaben in die Containerlaufzeitumgebung verlagert werden, wird das Betriebssystem zu einer dünnen Ebene, die Zugriff auf physische Ressourcen bietet. Das hat zur Entwicklung einer neuen Kategorie von Betriebssystemen geführt, die als „für Container optimierte OS (COS)“ bezeichnet werden.

Der Umfang eines COS ist viel kleiner als der eines herkömmlichen Betriebssystems. Es enthält die wichtigsten Komponenten, die für die Ausführung der Containerlaufzeitumgebung erforderlich sind. Die Wahl des richtigen COS ist sehr wichtig für den Betrieb von CaaS.

Kunden haben die Wahl zwischen Fedora CoreOS von Red Hat, Talos von Talos Systems, Flatcar Container Linux von der Kinvolk GmbH und RancherOS von Rancher Labs (Ende 2020 von SUSE übernommen).

Die meisten Anbieter bieten ein optionales, kommerzielles Abonnement an, das regelmäßige Updates, Patches und professionellen Support umfasst.

Containerlaufzeitumgebung

Die Containerlaufzeitumgebung verwaltet den Lebenszyklus eines Containers, stellt die Ausführungsumgebung bereit und fungiert als Schnittstelle zwischen dem Workload und dem Hostbetriebssystem.

Im Jahr 2015 gründete die Linux Foundation die Open Container Initiative (OCI), um Parität zwischen den verschiedenen Containerlaufzeitumgebungen herzustellen. Die OCI definiert zurzeit zwei Spezifikationen: die Runtime Specification (runtime-spec) und die Image Format

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Docker Engine Enterprise	Mirantis	containerd	Graduated
--	--	CRI-O	Incubating
--	--	Docker-CE	Nicht eingereicht
--	--	Kata Containers	Nicht eingereicht
--	--	runC	Nicht eingereicht

Specification (image-spec).

Laut der OCI-Website beschreibt die Runtime Specification, wie ein „Dateisystembündel“ ausgeführt wird, das auf eine Festplatte entpackt wird. Abstrakt gesagt lädt eine OCI-Implementierung ein OCI-Image herunter und entpackt es zu einem Dateisystembündel für die OCI-Laufzeitumgebung.

In der Image Format Specification wird definiert, wie ein OCI-Image erstellt wird – im Allgemeinen durch ein Build-System – und wie ein Image-Manifest, eine Dateisystem-(Ebenen-) Serialisierung und eine Image-Konfiguration ausgegeben werden.

Seit der Übernahme von Docker Enterprise durch Mirantis wird die kommerzielle Edition der Docker Engine (Docker Engine Enterprise) von Mirantis vertrieben; diese bietet Support und professionelle Dienstleistungen der Enterprise-Klasse.

Das Projekt containerd hat sich als Branchenstandard für Containerlaufzeitumgebungen etabliert. Es hat den CNCF-Status „Graduated“ und wird in vielen Produktionsumgebungen eingesetzt. CRI-O ist zurzeit ein CNCF-Projekt mit dem Status „Incubating“ mit aktiver Beteiligung der Community.

Docker Engine (jetzt Docker-CE) ist eine der beliebtesten Containerlaufzeitumgebungen für Containermanagementplattformen. Frakti (ein älterer Ansatz) ist eine Hypervisor-basierte Containerlaufzeitumgebung für Kubernetes, die stärkere Isolierung bietet, indem sie Pods in dedizierten VMs ausführt. Weitere Möglichkeiten sind Kata Containers und runC.

Containernativer Speicher

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Red Hat OpenShift Container Storage	Red Hat	Ceph	Nicht eingereicht
Kubera	MayaData	Longhorn	Sandbox
Portworx	Portworx	OpenEBS	Sandbox
Robin	Robin Systems	Rook	Incubating
StorageOS	StorageOS	--	--
Trident	NetApp	--	--

Der Speicher ist eine der wichtigsten Komponenten einer CaaS-Plattform. Containernativer Speicher macht die zugrunde liegenden Speicherdienste für Container und Microservices zugänglich. Wie softwaredefinierter Speicher aggregiert und bündelt er Speicherressourcen von unterschiedlichen Medien.

Containernativer Speicher ermöglicht mithilfe persistenter Volumes die Ausführung von Stateful-Workloads in Containern. In Kombination mit Kubernetes-Primitives wie StatefulSets bietet er die Zuverlässigkeit und Stabilität, die zur Bearbeitung geschäftskritischer Workloads in Produktionsumgebungen erforderlich ist.

Obwohl Kubernetes herkömmliche, verteilte Dateisysteme wie Network File System (NFS) und GlusterFS verwenden kann, empfehlen wir die Verwendung eines containersensiblen Speicher-Fabrics, das für die Anforderungen von Stateful-Workloads im Produktionsbetrieb entworfen wurde. Kunden haben zahlreiche Open-Source-Projekte und kommerzielle Implementierungen zur Auswahl.

Das Container-Storage-Interface (CSI) bietet Speicherspezifikationen für das cloudnative Ökosystem, die einen standardisierten, portablen Ansatz für die Implementierung und Nutzung von Speicherdiensten durch containerisierte Workloads fördern.

Ceph, Longhorn, OpenEBS und Rook sind Beispiele für Open-Source-Projekte für containernativen Speicher, während Kubera von MayaData, Trident von NetApp, Portworx, Red

Hat OpenShift Container Storage, Robin von Robin System und StorageOS kommerzielle Angebote mit Support sind.

Auch etablierte Anbieter wie NetApp, Pure Storage und VMware bieten Speicher-Plug-ins für Kubernetes an.

Bewältigung von Infrastrukturherausforderungen

Verwaltete Kubernetes-Angebote können die Komplexität großer Containerbereitstellungen reduzieren und ihre Nutzung auch ohne umfassende Fachkenntnisse ermöglichen. Die Optimierung der Infrastruktur von Kubernetes-Workloads ist eines der wichtigsten Kriterien, nach denen IT-Experten ihre Technologieroadmaps bewerten.

Wir betrachten nun Daten aus der CNCF-Umfrage von 2019 zu aktuellen und zukünftigen Plänen für die Kubernetes-Einführung sowie zu den Containerherausforderungen für Kubernetes-Benutzer. Die Ergebnisse zeigten, dass bestehende Anbieterbeziehungen einen großen Einfluss darauf hatten, wo die Technologie zuerst genutzt wurde. Die Zufriedenheit war jedoch durchwachsen.

Viele Kubernetes-Benutzer zogen ihre bestehenden Speicher- und Cloud-Anbieter in die engere Wahl für cloudnative Lösungen. Sie hatten jedoch offenbar Mühe, sich zu entscheiden. Alle 38 in der Umfrage erwähnten Anbieter wurden von mindestens fünf Prozent der Kubernetes-Benutzer evaluiert.

Nach dem Aufkommen von verwaltetem Kubernetes machten Cloud-Anbieter den Blockspeicher über Speicherklassen und dynamische Bereitstellung verfügbar. Kunden konnten Volumes von Amazon Elastic Block Store (EBS) auf AWS oder Azure Managed Disks und Google Persistent Disks auf Kubernetes-Worker-Knoten in AWS, GCP und Microsoft Azure nutzen. Damit waren Cloud-Anbieter im Vorteil.

Auf die Frage, welchen cloudnativen Speicher sie verwenden (siehe Abbildung 2.2 unten), nannten Kubernetes-Benutzer am häufigsten Amazon EBS, Google Persistent Disk und Azure Disk Storage. In vielen Fällen konnten Cluster-Workloads über StatefulSets auf den Blockspeicher des Cloud-Anbieters zugreifen. Blockspeicher großer Cloud-Anbieter wurden häufig eingesetzt, obwohl sie

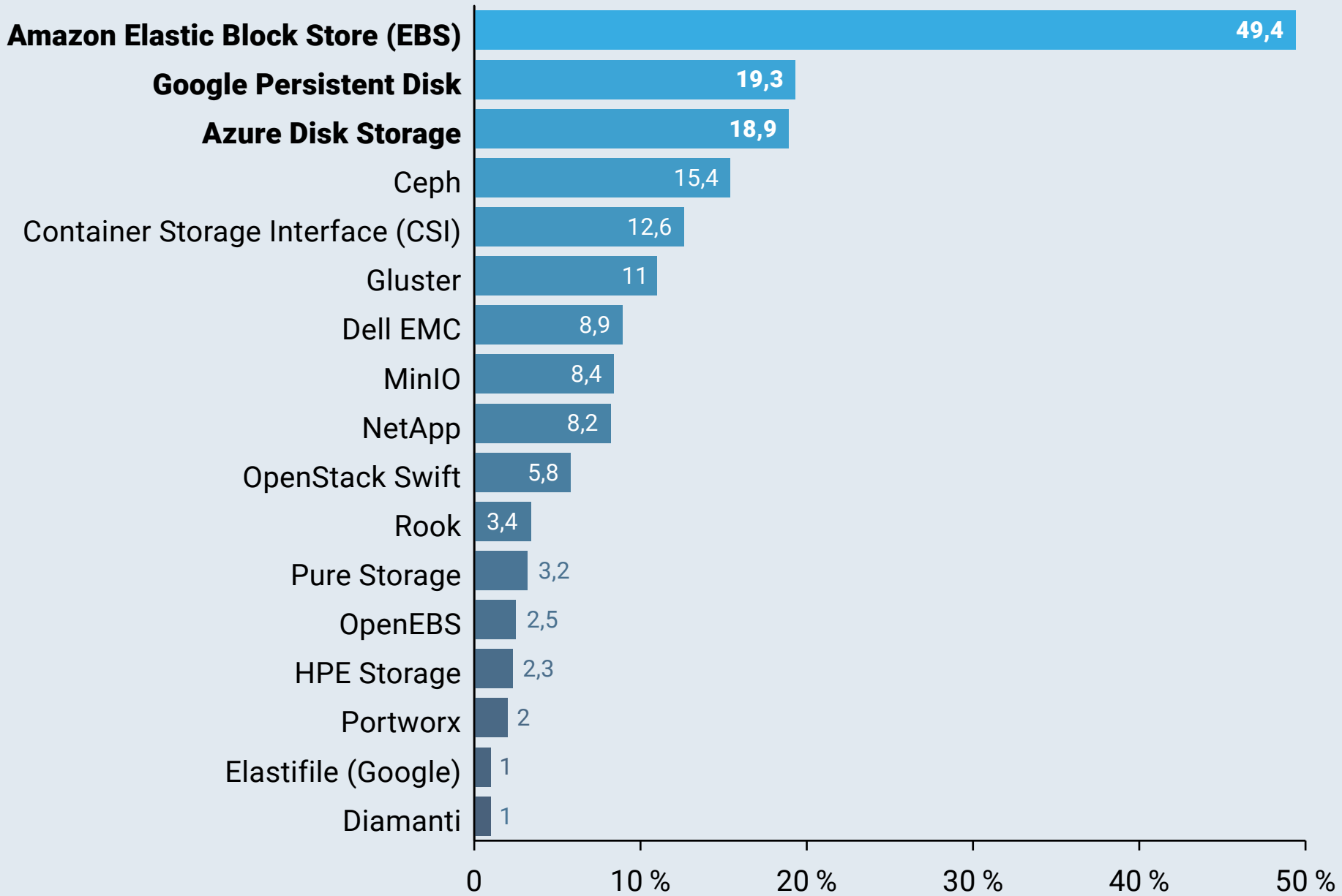


Abb. 2.2: *Dateisysteme wie Ceph werden oft als bessere Alternative zu Cloud-Speichern betrachtet als die Angebote etablierter Speicheranbieter.*

nicht speziell für Kubernetes-Workloads entwickelt wurden.

Auf den nächsten Plätzen folgten Ceph, CSI und Gluster, wobei 37 Prozent der Gluster-Benutzer auch Ceph einsetzen. Ceph und Gluster sind verteilte Dateisysteme, die eine Persistenzebene über mehrere Knoten hinweg einrichten. Sie sind jedoch nicht gut in Kubernetes-Tools und -Workflows integriert, sodass sie für Speicheradministratoren möglicherweise schwieriger zu warten und zu konfigurieren sind.

Auf weiteren Plätzen folgten Angebote etablierter Speicherunternehmen wie Dell EMC, NetApp und Pure Storage. Ursprünglich umfasste Kubernetes integrierte Volume-Plug-ins, um sich mit den Speicher-Back-Ends dieser Unternehmen zu verbinden. Die Upstream-Kubernetes-

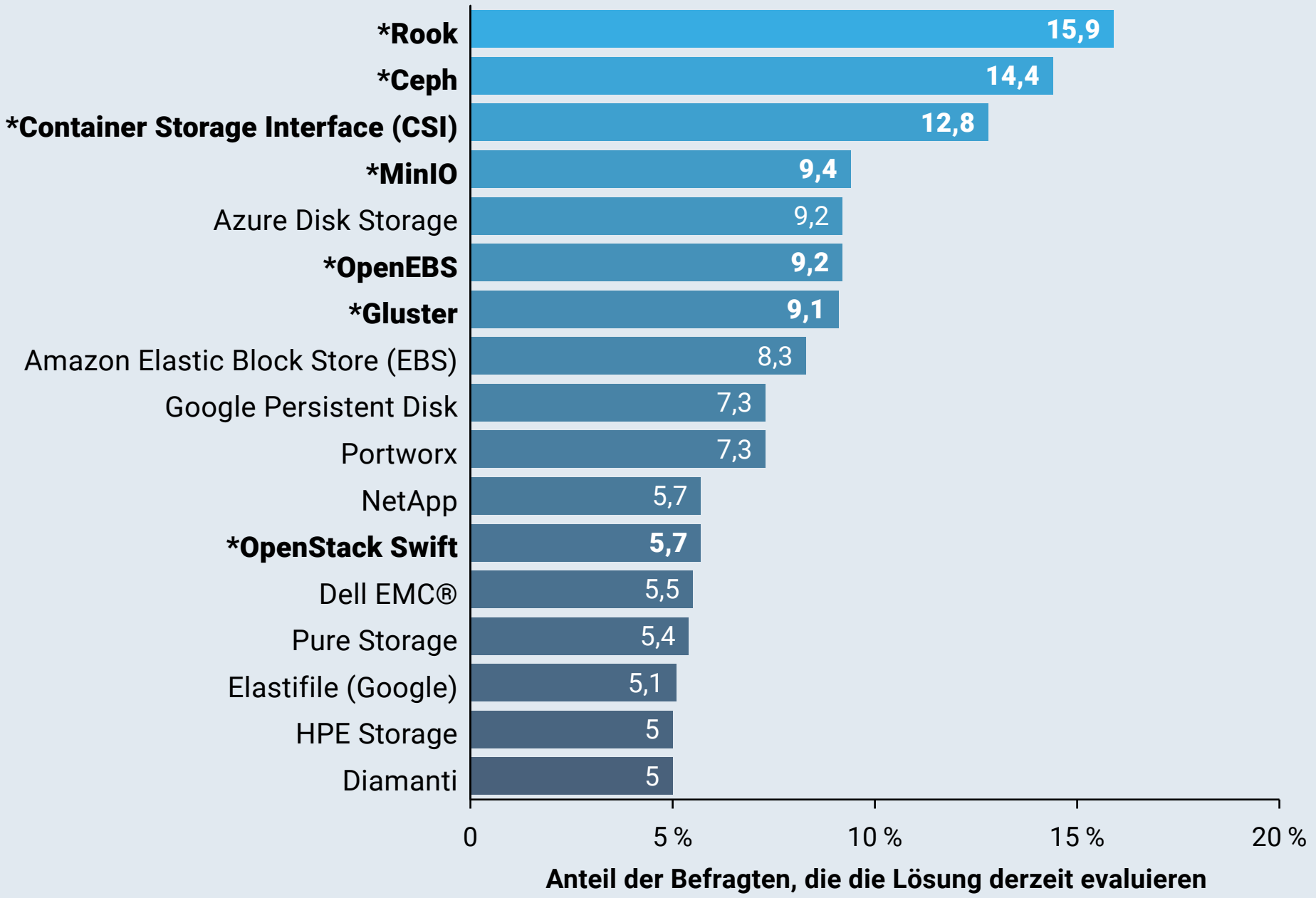
Distribution wuchs jedoch leider so sehr, dass jedes kleine Update und jede Änderung an einem Plug-in einen neuen Build mit Kompilierung des gesamten Codes erforderte.

Kunden etablierter Speicherunternehmen beklagten sich wesentlich häufiger über Speicherherausforderungen. Zum Beispiel hatten 46 Prozent der Pure Storage-Kunden Schwierigkeiten beim Umgang mit containerbezogenem Speicher, gegenüber nur 27 Prozent der Kubernetes-Benutzer insgesamt. Es zeichnete sich jedoch bereits ein Hoffnungsschimmer ab, denn 13 Prozent nutzten Container-Storage-Interface (CSI). CSI wurde 2019 allgemein für Kubernetes verfügbar und löst das Problem der häufigen Upstream-Integrationen. Etablierte Speicheranbieter, Cloud-Anbieter und reine Containerspeicherunternehmen wie Portworx wechseln zurzeit zu CSI.

CSI interessierte Benutzer, die nach neuen Lösungen für ihre aktuellen Herausforderungen rund um Containerspeicher suchten. Insgesamt zogen nur 13 Prozent der Kubernetes-Benutzer CSI in Erwägung, doch unter den Benutzern mit Speicherproblemen lag dieser Anteil bei 22 Prozent.

Obwohl einige der Befragten etablierte Unternehmen in Betracht zogen, waren diejenigen, die nach neuen Speicheroptionen suchten, vor allem an Open-Source-Projekten interessiert (siehe Abbildung 2.3). Im Vergleich zum Durchschnitt der Befragten evaluierten die 26,6 Prozent der Kubernetes-Benutzer, die mit Speicherproblemen zu kämpfen hatten, eher Rook (25,9 Prozent gegenüber 15,9 Prozent), Ceph (23,4 Prozent gegenüber 14,4 Prozent), Gluster (14,6 Prozent gegenüber 9,1 Prozent), OpenEBS (14,6 Prozent gegenüber 9,2 Prozent) und MinIO (12,8 Prozent gegenüber 9,4 Prozent). Diese Open-Source-Projekte zeichneten sich insbesondere dadurch aus, dass ihre Entwickler nicht auf den Verkauf von Hardware angewiesen waren.

Sowohl die Kunden etablierter Speicherunternehmen als auch die Benutzer der neueren Generation rein cloudnativer Speicherangebote klagten mit größerer Wahrscheinlichkeit über Speicherprobleme. Mit neuen Ansätzen wie CSI gingen etablierte Speicherunternehmen jedoch auf die Bedenken ihrer Kunden ein. Obwohl viele Benutzer neuerer Angebote – wie OpenEBS von MayaData, Minio und Portworx – bejahten, dass sie mit Speicherherausforderungen konfrontiert seien, handelte es sich dabei wahrscheinlich um Probleme bei der Anbindung älterer Datenspeicher.



Quelle: Auswertung der CNCF-Umfrage von 2019 durch The New Stack. Frage: Welche dieser cloudnativen Speicherprojekte werden in Ihrem Unternehmen/Ihrer Organisation derzeit evaluiert oder bereits in der Produktion eingesetzt? Mit welchen Herausforderungen sind Sie bei der Verwendung/Bereitstellung von Containern konfrontiert? Bitte wählen Sie alle zutreffenden Antworten aus. Die Grafik zeigt nur Daten für Angebote, die von mindestens 1 % der Befragten mit mindestens einem Kubernetes-Cluster genutzt werden. n=1.149.

Abb. 2.3: *Eine genauere Analyse von Kubernetes-Benutzern, die Speicherherausforderungen angaben (Daten nicht gezeigt), zeigt, dass sie um 50 Prozent häufiger Rook, Ceph und OpenEBS in Betracht ziehen als andere Benutzer. Alle drei Lösungen haben CSI-Treiber.*

Die Implementierung innovativer Drittanbieterlösungen war für ihre ersten Kunden eine große Herausforderung. Es wird interessant sein, die Effektivität dieser neuen Akteure im Laufe der Zeit zu beurteilen. Sie könnte sich auf die Fähigkeit etablierter Speicher- und Cloud-Unternehmen auswirken, ihre Kunden in diesem Segment zu halten.

Containernative Netzwerke

Ähnlich wie der containernative Speicher abstrahiert das containernative Netzwerk die physische Netzwerkinfrastruktur, um ein flaches Netzwerk für Container verfügbar zu machen. Es ist eng in Kubernetes integriert, um die Herausforderungen der Pod-zu-Pod-, Knoten-zu-Knoten-, Pod-zu-Service- und externen Kommunikation zu bewältigen.

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Calico Enterprise	Tigera	Cilium	Nicht eingereicht
Weave Net (Enterprise-Abonnement vom Anbieter erhältlich)	Weaveworks	Contiv	Nicht eingereicht
--	--	Flannel	Nicht eingereicht
--	--	Project Calico	Nicht eingereicht
--	--	Tungsten Fabric	Nicht eingereicht
--	--	Weave Net	Nicht eingereicht

Kubernetes unterstützt zahlreiche Plug-ins gemäß der CNI-Spezifikation (Container-Network-Interface). Diese definiert die Netzwerkkonnektivität von Containern sowie die Verwaltung der Netzwerkressourcen, wenn der Container gelöscht wird. Das CNI-Projekt gehört zu den Projekten mit CNCF-Status „Incubating“.

Containernative Netzwerke bieten mehr als nur bloße Konnektivität. Sie unterstützen eine dynamische Durchsetzung von Netzwerksicherheitsregeln. Mithilfe von Richtlinien kann eine differenzierte Steuerung der Kommunikation zwischen Containern, Pods und Knoten konfiguriert werden.

Die Wahl des richtigen Netzwerkstacks ist entscheidend für die Pflege und Sicherheit der CaaS-Plattform. Kunden stehen Stacks aus Open-Source-Projekten wie Cilium, Contiv, Flannel, Project Calico, Tungsten Fabric und Weave Net zur Auswahl. Auf der kommerziellen Seite bietet Tigera Calico Enterprise an und ein Enterprise-Abonnement von Weave Net ist von Weaveworks erhältlich.

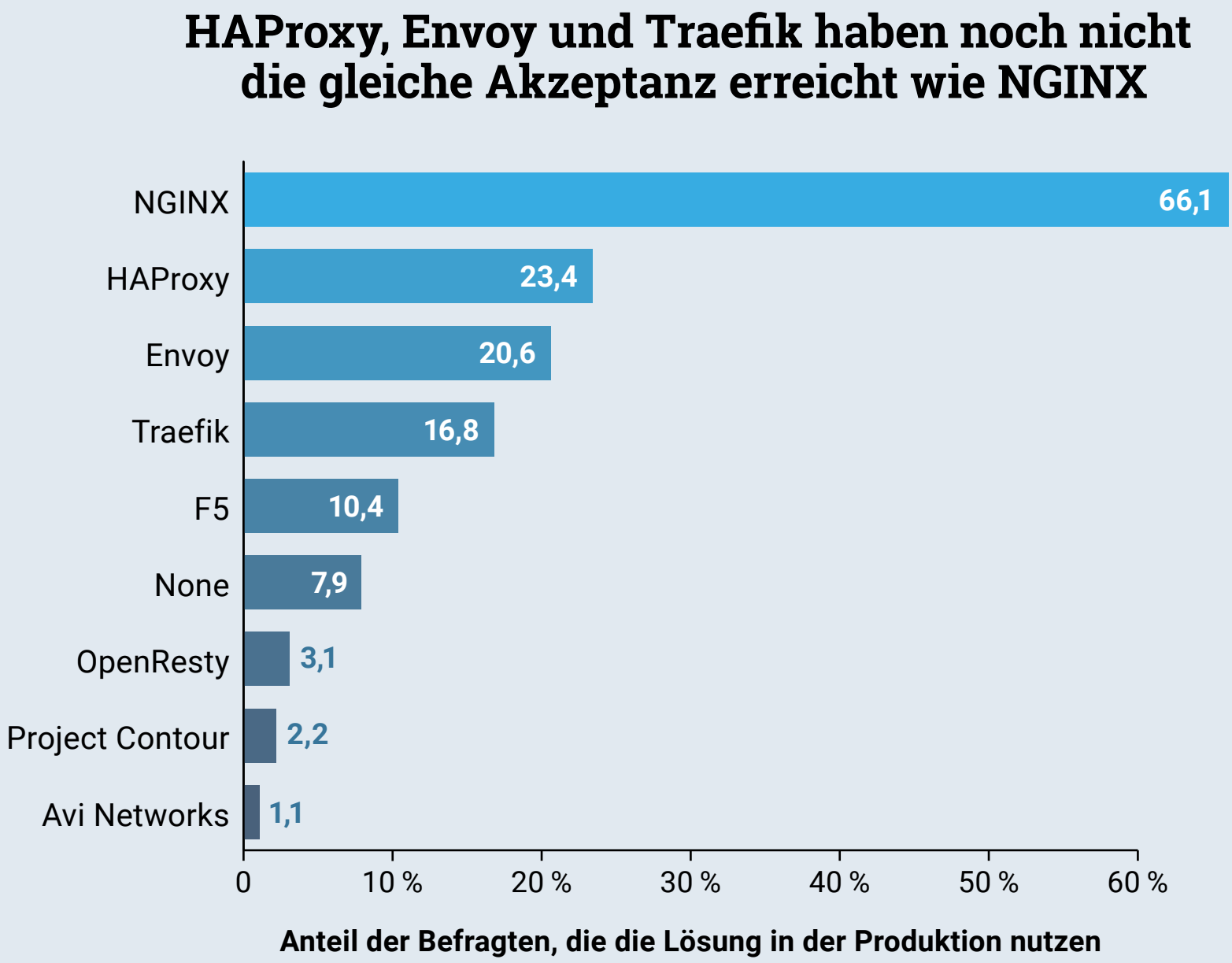
Verwaltete CaaS-Angebote von Public-Cloud-Anbietern sind eng in den vorhandenen virtuellen Netzwerkstack integriert. Beispielsweise bietet AWS ein CNI-Plug-in für Amazon Elastic Kubernetes Service (EKS) an, das auf der Amazon Virtual Private Cloud (VPC) basiert, während Microsoft das Azure Virtual Network Container Networking Interface (CNI) für Azure Kubernetes Service (AKS) entwickelt hat.

Betrachten wir nun wieder Daten aus der CNCF-Umfrage von 2019, jetzt in Bezug auf Netzwerke.

Die Netzwerkherausforderung ist im Laufe der Jahre kleiner geworden, auch wenn Kubernetes-Benutzer weiterhin Ingressanbieter evaluieren. Während ein Kubernetes-Benutzer im Durchschnitt 1,5 Ingressanbieter nutzte, hatten die 28 Prozent der Befragten, die das Netzwerk als Herausforderung nannten, im Durchschnitt drei Ingressanbieter. NGINX fand sich in 66 Prozent der Kubernetes-Stacks – konnte jedoch allein nicht die Anforderungen aller Benutzer erfüllen. HAProxy und Envoy wurden seltener eingesetzt, waren aber bei den Befragten mit Netzwerkherausforderungen überdurchschnittlich stark vertreten.

Dies legt nahe, dass neuere Marktteilnehmer herangezogen wurden, um Probleme zu lösen. Es ist anzunehmen, dass sich Angebote in Zukunft darin unterscheiden werden, welche Protokolle sie unterstützen und ob sie ein API-Gateway enthalten.

Abb. 2.4: *Trotz der Verbindung mit Ingress hat die Verwendung von NGINX mit Kubernetes kaum Auswirkungen auf Netzwerkherausforderungen.*



Quelle: Auswertung der CNCF-Umfrage von 2019 durch The New Stack. Frage: Welche Ingressanbieter (d. h. Serviceproxys) für Kubernetes verwenden Sie? Bitte wählen Sie alle zutreffenden Antworten aus. Mit welchen Herausforderungen sind Sie bei der Verwendung/Bereitstellung von Containern konfrontiert? Bitte wählen Sie alle zutreffenden Antworten aus. Die Grafik zeigt nur Daten für Angebote, die von mindestens 1 % der Befragten mit mindestens einem Kubernetes-Cluster genutzt werden. n=1.197.

Sicherheit und Härtung

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Aqua	Aqua Security Software	Clair	Nicht eingereicht
Calico Enterprise	Tigera	Falco	Incubating
Prisma Cloud	Palo Alto Networks	Open Policy Agent (OPA)	Incubating
Snyk	Snyk	Snyk (Open Source)	Nicht eingereicht
StackRox	StackRox	--	--

Containermanagementplattformen werden durch die Härtung des Netzwerks, das Scannen von Container-Images und die regelmäßige Erstellung von Risikoprofilen und Bedrohungserkennung gesichert.

Die Netzwerksicherheit ist ein wichtiger Aspekt der Plattformsicherheit. Das IT-Team sollte in eine Zero-Trust-Netzwerkebene investieren, um Sicherheits- und Compliance-Anforderungen zu erfüllen. Diese Netzwerke setzen differenzierte Richtlinien an mehreren Punkten in der Infrastruktur durch, suchen nach Anomalien und verschlüsseln und autorisieren den Datenverkehr zwischen Microservices mit sicheren Protokollen wie mTLS (mutual Transport Layer Security). Calico Enterprise von Tigera ist eine der führenden Zero-Trust-Netzwerklösungen für Kubernetes- und CaaS-Plattformen.

In einer sicheren Containerplattform kommen die folgenden Techniken für die Sicherheit der Infrastruktur und der Anwendungen zum Einsatz:

- Signierte, verifizierte, **vertrauenswürdige Images**: Diese Funktionalität ist eng in die Container-Registry-Komponente integriert, in der Container-Images gespeichert werden.
- Ein **verschlüsselter Speicher**, in dem vertrauliche Daten wie Benutzernamen, Passwörter usw. sicher gespeichert und abgerufen werden können.
- Integration in vorhandene Lightweight-DirectoryAccess-Protocol-(LDAP-) und Active-Directory-(AD-)Umgebungen, damit eine integrierte **rollenbasierte Zugriffskontrolle** (Role-Based Access Control, RBAC) konfiguriert werden kann.

Aqua, Prisma Cloud und StackRox bieten kommerzielle Sicherheitslösungen für Container und Kubernetes-Infrastrukturen. Snyk bietet sowohl kommerzielle als auch Open-Source-Lösungen an.

Open-Source-Projekte wie Clair bieten statische Schwachstellenanalysen für Anwendungscontainer.

Falco, ein Projekt mit CNCF-Status „Incubating“, ist eine verhaltensbasierte Aktivitätsüberwachung, die Anomalien in cloudnativen Anwendungen erkennt. Falco prüft ein System auf der untersten Ebene: dem Kernel. Diese Daten reichert Falco mit anderen eingehenden Datenströmen an, zum Beispiel mit Containerlaufzeitmetriken und Kubernetes-Metriken, um Benutzer anhand vorgegebener Regeln zu warnen.

Open Policy Agent (OPA), ein weiteres Projekt mit CNCF-Status „Incubating“, bietet ein einheitliches Toolset und Framework für Richtlinien im gesamten Stack. OPA bietet eine deklarative höhere Programmiersprache, in der Entwickler und Betreiber Richtlinien codieren können, sowie einfache APIs, um die Entscheidungsfindung für Richtlinien aus der Software auszulagern. OPA kann Richtlinien in Microservices, Kubernetes, CI/CD-Pipelines, API-Gateways und mehr durchsetzen.

Service-Mesh

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Aspen Mesh	F5	Consul	Nicht eingereicht
Consul Enterprise	HashiCorp	Contour	Incubating
Grey Matter	Decipher Technology Studios	Envoy	Graduated
Kong Enterprise	Kong	Istio	Nicht eingereicht
Linkerd Commercial Support	Buoyant	Kuma	Sandbox
Maesh	Containous	Linkerd	Incubating
Service Mesh Hub	Solo.io	Service Mesh Interface (SMI)	Sandbox
Tetrate	Tetrate	Zuul	Nicht eingereicht

Das Service-Mesh entwickelt sich zu einer wichtigen Komponente des cloudnativen Stacks. Es ermöglicht eine differenzierte Kontrolle über den Datenverkehr zwischen verschiedenen Microservices und bietet Einblicke in ihren Zustand.

Das Service-Mesh ergänzt das containernative Netzwerk. Es konzentriert sich auf den Ost-West-Datenverkehr, während der Nord-Süd-Verkehr von der zentralen Netzwerkebene verarbeitet wird. Das Service-Mesh weist jedem Microservice einen Proxy zu, der den ein- und ausgehenden Datenverkehr überwacht. Da er sich in unmittelbarer Nähe des Microservices befindet, kann er seinen Zustand verfolgen.

Consul, Contour, Envoy, Istio, Kuma, Linkerd, Service Mesh Interface (SMI) und Zuul sind einige gängige Open-Source-Projekte für das Service-Mesh.

Kommerzielle Implementierungen sind beispielsweise Aspen Mesh von F5, Consul Enterprise von HashiCorp, Grey Matter von Decipher Technology Studios, Kong Enterprise von Kong, Linkerd Commercial Support von Buoyant, Maesh von Containous, Service Mesh Hub von Solo.io und Tetrate.

Betrachten wir nun Daten der CNCF-Umfrage von 2019 zum Service-Mesh.

Laut den Umfrageergebnissen stellten Service-Meshes für 27 Prozent der Kubernetes-Benutzer eine Herausforderung dar. Das betraf sowohl die Auswahl des richtigen Tools als auch den täglichen Betrieb.

Consul von HashiCorp und Istio lagen nahezu gleichauf an der Spitze des Feldes der von Kubernetes-Benutzern am häufigsten genutzten Service-Meshes. Viele der Befragten, die Consul erwähnten, nutzten es jedoch möglicherweise bereits für die Serviceinventarisierung und experimentierten mit dem Anwendungsfall Service-Mesh nur.

Das war bei Istio und Linkerd nicht der Fall und beide wurden deutlich häufiger von Befragten evaluiert, die das Service-Mesh als Herausforderung empfanden.

Zuul von Netflix stand beim Interesse der Benutzer an vierter Stelle. Auffälligerweise klagten nur

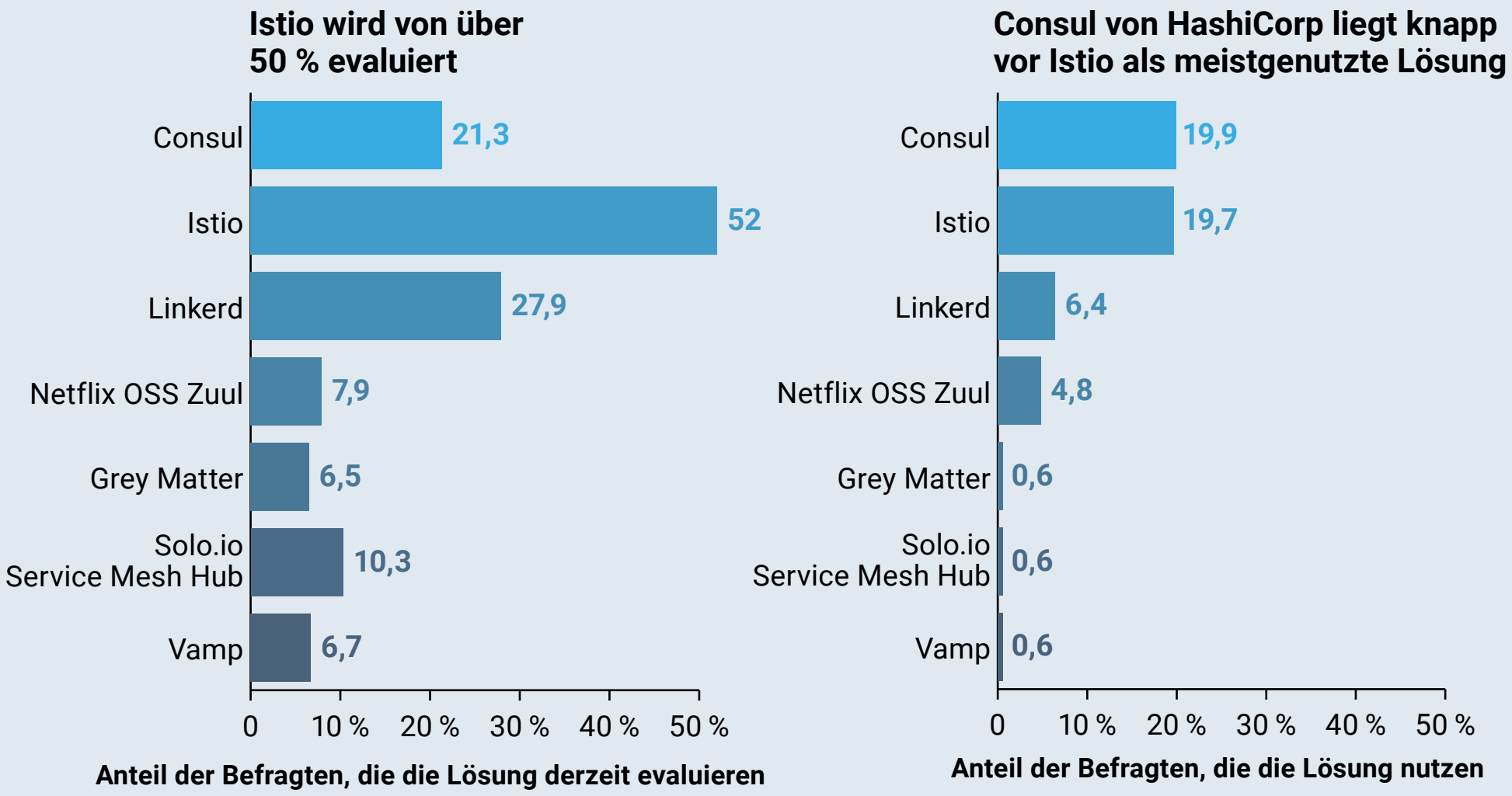


Abb. 2.5: *Eine bestehende Entwicklerbasis, die Consul für die Serviceinventarisierung verwendet, bedeutet wahrscheinlich, dass es auch für verwandte Service-Mesh-Funktionen verwendet wird. Zugleich evaluiert über die Hälfte aller Kubernetes-Benutzer Istio zum ersten Mal.*

23 Prozent der Benutzer von Zuul über Probleme mit dem Service-Mesh, im Vergleich zu 27 Prozent aller Befragten. Da der Bekanntheitsgrad von Zuul geringer war als der von Istio, hatten die Zuul-Benutzer das Produkt mit größerer Wahrscheinlichkeit bei einer gezielten Suche nach einer Lösung für ein bestimmtes Problem entdeckt.

Grey Matter, Service Mesh Hub und Vamp hatten, wenn überhaupt, nur wenige Benutzer, aber ein ansehnlicher Bekanntheitsgrad brachte sie häufig in die engere Wahl.

Für die Zukunft erwarten wir, dass Benutzer aller Service-Meshes vor der Herausforderung stehen werden, sie ausreichend in den Rest ihres Kubernetes-Stacks zu integrieren. In der Zwischenzeit ist zu erwarten, dass die Anbieter von Istio, Linkerd und anderen Services versuchen werden, ihre Integration mit Envoy zu verbessern.

Containerorchestrierungs-Engine

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Canonical Distribution of Kubernetes (CDK)	Canonical	Kubernetes	Graduated
Cisco Container Platform	Cisco	OKD	Nicht eingereicht
Docker Enterprise	Mirantis	Rancher Kubernetes Engine (RKE)	Nicht eingereicht
HPE Ezmeral Container Platform	HPE	--	--
Mesosphere Kubernetes Engine (MKE)	D2IQ	--	--
Mirantis Cloud Platform	Mirantis	--	--
Rancher Kubernetes Engine (RKE)	Rancher Labs	--	--
Red Hat OpenShift	Red Hat	--	--
SUSE CaaS Platform	SUSE	--	--
Tanzu Kubernetes Grid	VMware	--	--

Wie in Kapitel 1 besprochen, ist Kubernetes die am häufigsten genutzte Containerorchestrierungs-Engine für CaaS-Plattformen.

Obwohl Kubernetes auf GitHub als Open-Source-Upstream-Distribution verfügbar ist, sollten Kunden möglicherweise in eine kommerzielle Distribution mit Wartungs- und Supportpaketen investieren. Canonical Distribution of Kubernetes, Cisco Container Platform, Docker Enterprise, HPE Ezmeral Container Platform, Mesosphere Kubernetes Engine, Mirantis Cloud Platform, Rancher Kubernetes Engine, Red Hat OpenShift (das auch als PaaS genutzt werden kann), SUSE CaaS Platform und VMware Tanzu Kubernetes Grid sind einige der kommerziellen Distributionen auf dem Markt.

Zudem gibt es Open-Source-Projekte, wie beispielsweise OKD und Rancher Kubernetes Engine (RKE).

Kunden mit Investitionen in Public-Cloud-basierte CaaS-Plattformen auf der Grundlage von Google Kubernetes Engine oder IBM Kubernetes Service können sich für Anthos oder IBM Cloud Paks entscheiden, die eng in die cloudbasierte Steuerungsebene integriert sind.

Die Bereitstellungsmodelle von Kubernetes wurden im vorherigen Kapitel ausführlich vorgestellt.

Container-Registry

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Docker Hub	Docker	Docker Registry	Nicht eingereicht
Docker Trusted Registry	Mirantis	Harbor	Graduated
GitLab Container Registry	GitLab	Project Quay	Nicht eingereicht
JFrog Container Registry / Artifactory	JFrog	--	--
Red Hat Quay	Red Hat	--	--

Microservices werden als Container-Images verpackt, bevor sie von der Orchestrierungs-Engine bereitgestellt und skaliert werden. Eine Container-Registry fungiert als zentrales Repository, in dem Container-Images gespeichert werden. Sie ist in Build-Management-Tools integriert und generiert jedes Mal, wenn eine neue Version des Service erstellt wird, automatisch die entsprechenden Images. Ein Releasemanagementtool ruft die neueste Version des Images aus der Registry ab und stellt sie bereit.

Docker Trusted Registry von Mirantis, GitLab Container Registry, JFrog Container Registry und Red Hat Quay sind einige der kommerziell erhältlichen Container-Registrys neben denen, die von den Cloud-Anbietern angeboten werden. Auch Docker bietet eine gehostete Registry namens Docker Hub und ihr Open-Source-Gegenstück, die Docker Registry, an.

Neben der Speicherung der Container-Images übernehmen diese Produkte in der Regel auch Schwachstellenscans für Images sowie die Authentifizierung und Autorisierung der Benutzer. In einigen Unternehmen müssen On-Premises-Repositorys genutzt werden und der Zugriff auf internetbasierte Registrys ist verboten. Für den Einsatz in derartigen Umgebungen enthalten

einige der kommerziellen Angebote interne Registrys. Alternativ ist es möglich, eine interne Registry mit ähnlichen Funktionen zu erstellen.

Cloud-Anbieter, die Kubernetes als Managed Service anbieten, bieten eine Option zum Hosten einer privaten Container-Registry innerhalb des Kundenkontos. Da sich das Cluster und die Registry in derselben Region und demselben Konto befinden, bieten sie geringe Latenz und hohe Sicherheit.

Harbor, ursprünglich von VMware entwickelt, ist eine als Open Source erhältliche Container-Image-Registry, die Images mit rollenbasierter Zugriffskontrolle sichert, auf Schwachstellen scannt und als vertrauenswürdig signiert. Project Quay ist die Open-Source-Distribution von Quay von Red Hat. Sie bietet eine Web-Benutzeroberfläche, die der von Verbrauchersoftware ähnelt, scannt Images auf Schwachstellen und bietet Datenspeicherung und -sicherheit der Enterprise-Klasse.

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
CircleCI	Circle Internet Services	GitLab	Nicht eingereicht
CloudBees CI	CloudBees	Jenkins / Jenkins X	Nicht eingereicht
Digital.ai	Digital.ai Software	--	--
GitHub Actions	GitHub	--	--
GitLab CI	GitLab	--	--
JFrog Pipelines	JFrog	--	--
SemaphoreCI	Rendered Text	--	--
Travis CI	Travis CI	--	--

Build-Management

Zur schnellen Bereitstellung von Microservices wird jedes Mal, wenn Code im Repository abgelegt wird, ein neues Container-Image erstellt und in die Container-Registry aufgenommen. Diese Images können dann in Staging- oder Testumgebungen innerhalb von CaaS für automatisierte und manuelle Tests bereitgestellt werden.

Das Build-Management umfasst die Konvertierung der neuesten Codeversion in verschiedene Artefakte wie Container-Images, Bibliotheken und ausführbare Dateien. Es bildet eine wichtige Stufe in der Pipeline für kontinuierliche Integration und Bereitstellung (CI/CD).

Das Quellcodeverwaltungssystem, das auf internen oder externen Git-Repositorys basiert, startet einen automatisierten und sicheren Build-Prozess, der ein Bereitstellungsartefakt erstellt. Das Ergebnis dieser Phase kann Container-Images, Helm-Charts und Kubernetes-Bereitstellungen umfassen.

GitLab und Jenkins sind beliebte Build-Management-Lösungen, die als Open-Source- und als kommerzielle Versionen erhältlich sind. CloudBees CI ist eine kommerzielle Version des Build-Management-Servers Jenkins.

CircleCI, Digital.ai, GitHub Actions, JFrog Pipelines, SemaphoreCI und Travis CI sind kommerzielle CI/CD-Lösungen für Microservices.

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
Armory Enterprise Spinnaker	Armory	**Argo CD**	Incubating
Open Enterprise Spinnaker	OpsMx	**Flux**	Sandbox

Releasemanagement

Das Releasemanagement ist die letzte Stufe einer CI/CD-Pipeline. Hierbei wird eine vollständig getestete Version der Microservices in der Produktionsumgebung bereitgestellt. Fortschrittliche Bereitstellungsstrategien wie Canary Releases, Blue/Green-Bereitstellungen, Rollbacks und Rollforwards sind Teil des Releasemanagement.

Spinnaker ist eines der beliebtesten Releasemanagement für Microservices. Es ergänzt Jenkins und andere Build-Management-Tools, indem es die Verwaltung und Bereitstellung von Artefakten automatisiert. Armory und OpsMx bieten kommerzielle Implementierungen von Spinnaker an.

GitOps ist eine neue Technik zur Implementierung von Configuration-as-Code (CaC).

Die YAML-Artefakte und Helm-Charts unterliegen einer Versionsverwaltung und werden in einem Git-Repository gepflegt. Mit einer Pull- oder Push-Strategie werden die an der Konfiguration vorgenommenen Änderungen auf das Cluster angewendet. Argo CD (CNCF-Status „Incubating“) und Flux (CNCF-Status „Sandbox“) sind zwei beliebte Open-Source-Projekte zum Konfigurieren von GitOps.

Kommerzielle Produkte	Anbieter	Open-Source-Projekte	CNCF-Status
AppDynamics	Cisco	Fluentd	Graduated
Datadog	Datadog	Grafana	Nicht eingereicht
Dynatrace	Dynatrace	Jaeger	Graduated
Honeycomb	Honeycomb	OpenTelemetry	Sandbox
Lightstep	Lightstep	OpenTracing	Incubating
New Relic One	New Relic	Prometheus	Graduated
Sysdig Monitor	Sysdig	--	--

Beobachtbarkeit

Zur Beobachtbarkeit gehört die Erfassung der Kennzahlen, Protokolle, Ereignisse und Traces aus dem gesamten Stack.

Beim Monitoring werden Kennzahlen aus der Infrastruktur und dem Anwendungsstack der Plattform erfasst. Eine robuste Monitoringplattform überwacht den Zustand des Kubernetes-Clusters und der bereitgestellten Anwendungen. Die auf jedem Knoten installierten Agenten sammeln detaillierte Informationen zur Ressourcenauslastung, zum Zustand des Clusters, zum Zustand des Knotens, zur Verfügbarkeit von Speicher und Arbeitsspeicher, zur Anzahl der auf jedem Knoten ausgeführten Container sowie detaillierte Kennzahlen für Pods. Kunden können Open-Source-Überwachungstools wie Grafana und Prometheus implementieren oder in kommerzielle Plattformen wie DataDog, Dynatrace, New Relic One und Sysdig Monitor investieren.

Die Protokollierung sammelt und aggregiert die Informationen, Warnungen und Fehler aus verschiedenen Komponenten der cloudnativen Plattform. Fast jede Komponente von Kubernetes

erzeugt Protokolle, die detaillierte Einblicke in den aktuellen Zustand des Clusters bieten. Im Rahmen der Best Practices sollten Entwickler die Protokollierung in ihre Microservices integrieren. Innerhalb des Clusters sind verschiedene Agenten damit beschäftigt, Protokolle zu erfassen und an ein zentrales Repository zu streamen. Fluentd ist ein beliebtes Open-Source-Tool zur Datensammlung, das in Kubernetes bereitgestellt wird und in Kombination mit Elastic und Kibana zur Visualisierung und Durchsuchung von Protokollen verwendet werden kann. Service-Mesh-Lösungen wie Istio und Linkerd sind eng in die Protokollierungsplattformen integriert.

Prometheus und Fluentd haben den CNCF-Status „Graduated".

Da cloudnative Anwendungen aus unterschiedlichen und autonomen Services zusammengesetzt sind, ist es wichtig, die Kommunikationskette und die Reaktionszeiten der einzelnen Services zu verfolgen, um die Anwendungsleistung zu überwachen und zu analysieren.

Open-Source-Tools wie Jaeger und OpenTracing können APM-Funktionen (Application Performance Monitoring) für Microservices bereitstellen. Kommerzielle Angebote sind unter anderem AppDynamics, Honeycomb und Lightstep. Die beiden letztgenannten bieten umfassende Tracing- und Überwachungsfunktionen für moderne Anwendungen.

OpenTelemetry bietet einen einheitlichen Satz von APIs, Bibliotheken, Agenten und Kollektordiensten zur Erfassung von verteilten Traces und Kennzahlen aus cloudnativen Anwendungen.

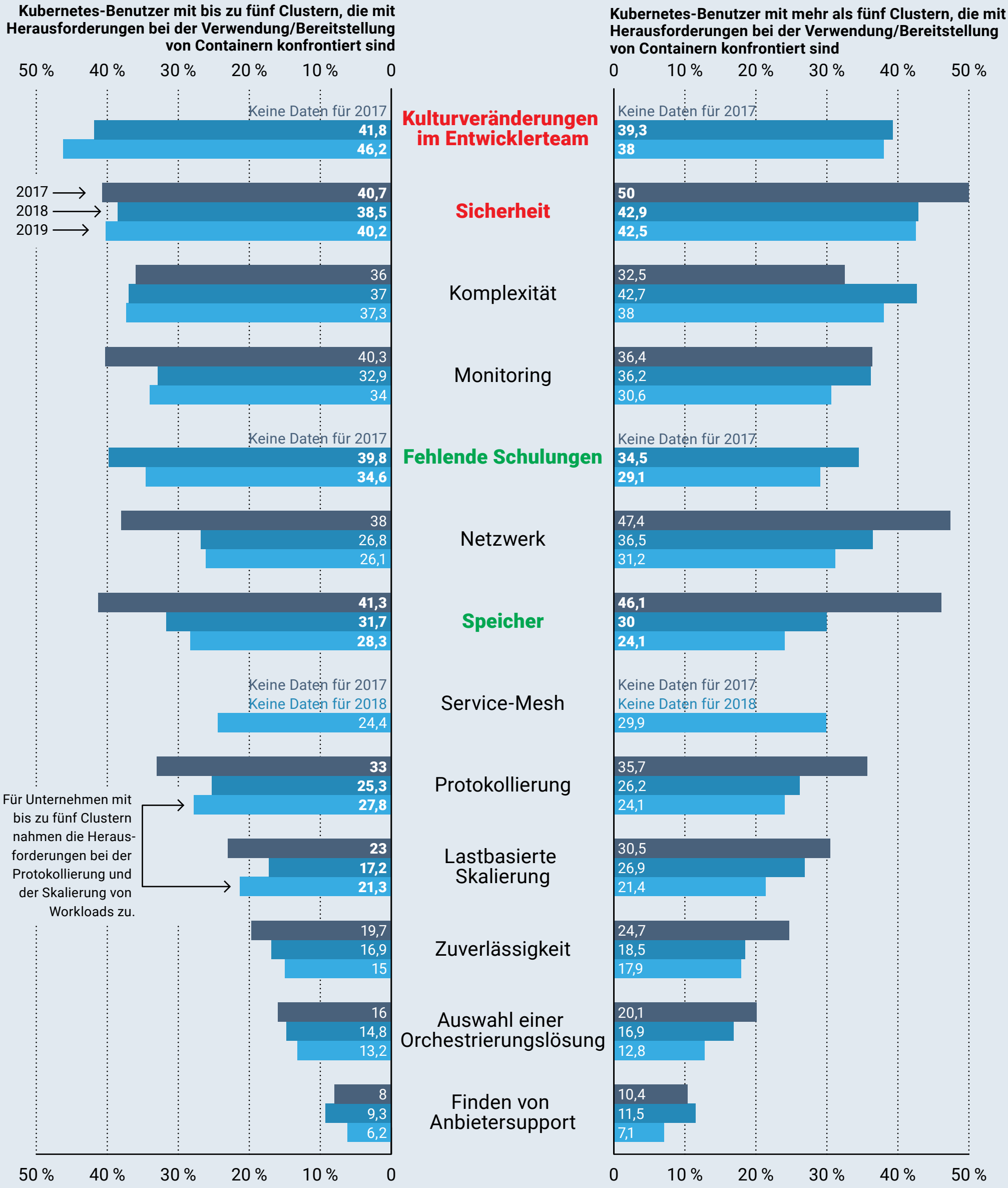
OpenTracing hat den CNCF-Status „Incubating" und OpenTelemetry den CNCF-Status „Sandbox".

Containerherausforderungen für Kubernetes-Benutzer

Wiederum anhand der Daten aus den CNCF-Umfragen von 2017–2019 sehen wir uns jetzt die Containerherausforderungen an, mit denen Kubernetes-Benutzer konfrontiert sind.

Vertreter von Unternehmen mit mehr als fünf Clustern meldeten 2018 deutlich weniger Herausforderungen im Zusammenhang mit Containern als 2017. Der rückläufige Trend hielt auch 2019 weiter an. Kubernetes-Benutzer mit einem bis fünf Clustern nannten als Antwort auf die

Schulungen und Speicherung nun einfacher, Kultur und Sicherheit noch nicht vollständig im Griff



Quelle: Auswertung der CNCF-Umfragen von 2017, 2018 und 2019 durch The New Stack. Frage: Mit welchen Herausforderungen sind Sie bei der Verwendung/Bereitstellung von Containern konfrontiert? Bitte wählen Sie alle zutreffenden Antworten aus. 1–5 Kubernetes-Cluster: 2017, n=300; 2018, n=925; 2019, n=729. >5 Kubernetes-Cluster: 2017, n=154; 2018, n=550; 2019, n=468.

Abb. 2.6: *Kubernetes-Benutzer mit bis zu fünf Clustern haben mit Herausforderungen hinsichtlich der Kultur ihrer Entwicklerteams zu kämpfen. Benutzer mit mehr als fünf Clustern verzeichneten 2019 leichte Rückgänge bei allen Herausforderungen.*

Frage nach Herausforderungen bei der Verwendung oder Bereitstellung von Containern am häufigsten die Sicherheit, Komplexität und die notwendigen kulturellen Veränderungen in Entwicklerteams.

Unabhängig von der Anzahl der genutzten Cluster sagten die Befragten, Herausforderungen bezüglich der Datenvernetzung und -speicherung würden derzeit gelöst. Der Rückgang der Speicherprobleme war möglicherweise darauf zurückzuführen, dass die CSI-Spezifikation bereits relativ ausgereift war und die meisten Speicherlösungen eine CSI-Schnittstelle besaßen. Die Netzwerkherausforderungen waren vielleicht zurückgegangen, weil auf dem Container-Networking-Interface (CNI) basierende Technologien wie Flannel zum De-Facto-Standard für die Containerkommunikation in Clustern geworden waren.

Der Erfolg der CNCF, der KubeCon + CloudNativeCon und des restlichen Open-Source-Ökosystems zeigt sich darin, dass die Zahl der Benutzer, die sich über fehlende Schulungen oder mangelnden Anbietersupport beklagen, kontinuierlich abnimmt.

Für Befragte mit mehr als fünf Clustern nahmen Herausforderungen im Zusammenhang mit der Skalierung ihrer Umgebungen weiter ab. Das Skalieren von Workloads nach oben, das Monitoring und die Protokollierung in größeren Clustern wurden im Kubernetes-Ökosystem erfolgreich angegangen.

Befragte aus Unternehmen mit fünf oder weniger Clustern malten ein weniger rosiges Bild. Für sie waren die Herausforderungen bei der Skalierung von Bereitstellungen, der Entwicklungskultur und der Protokollierung zwischen 2018 und 2019 gestiegen.

Besonders schwer fielen ihnen die erforderlichen Veränderungen in der Kultur der Entwicklerteams. Das lag vermutlich daran, dass die Neuausrichtung der Organisationsstruktur in diesen kleineren Unternehmen weniger weit fortgeschritten war. Obwohl viele Unternehmen DevOps- oder SRE-Teams eingerichtet hatten, arbeiteten die Entwickler in diesen Unternehmen weiterhin vor allem an Aspekten der Infrastrukturebene.

Containermanagementplattformen für Unternehmen

Plattform- und Betriebssystemanbieter haben kommerzielle Distributionen von Kubernetes erstellt, die in Unternehmensrechenzentren installiert werden können. Als Infrastruktur stehen physische oder Bare-Metal-Server, virtuelle Maschinen in einem Hypervisor sowie Public-Cloud-IaaS (Infrastructure-as-a-Service) zur Auswahl.

Die folgenden Mainstreamanbieter bieten kommerzielle Kubernetes-Distributionen an:

Canonical

Ubuntu von Canonical ist das am weitesten verbreitete Linux-Betriebssystem in Public Clouds. Ubuntu kann auch in privaten Clouds und auf Internet-of-Things-Geräten (Internet der Dinge, IoT) sowie als Betriebssystem für Desktop-PCs eingesetzt werden. Unternehmen schätzen Ubuntu aufgrund seiner Kompatibilität, Leistungsfähigkeit und Vielseitigkeit.

Canonical bietet zwei Varianten von Kubernetes:

- MicroK8s: Eine kompakte Distribution, die mit einem einzigen Befehl auf IoT- und Edge-Geräten installiert werden kann.
- Charmed Distribution of Kubernetes: Eine vollständig CNCF-konforme Upstreamversion von Kubernetes. Canonical bietet einen Managed Service für Unternehmen, der Schulungen, Installation, Konfiguration, Orchestrierungstools und Optimierung von Kubernetes-Clustern in der Produktion umfasst.

Mirantis

Mirantis hat sich von einem reinen OpenStack-Unternehmen zu einem Kubernetes-Akteur gewandelt. Das Unternehmen bietet jetzt eine eigene Kubernetes-Distribution an. Es spezialisiert sich auf eine offene Cloud-Infrastruktur mit einer Plattform, die virtuelle Maschinen und Container nahtlos verwalten kann.

Im Jahr 2019 übernahm Mirantis Docker Enterprise von Docker, Inc. Dazu gehören Docker Engine Enterprise Edition, Docker Desktop Enterprise, Docker Trusted Registry und Docker Enterprise CaaS.

Neben Docker Enterprise hat Mirantis auch ein Managed-Service-Angebot für Kubernetes, das als KaaS (Kubernetes-as-a-Service) vermarktet wird und auf OpenStack oder AWS bereitgestellt werden kann.

Rancher

Rancher Labs (Ende 2020 von SUSE übernommen) bietet eine Plattform zur Bereitstellung und Verwaltung von Kubernetes als Unternehmensservice.

Rancher Kubernetes Engine ist für IT-Teams in Unternehmen bestimmt, die ihren Mitarbeitern Kubernetes-as-a-Service (KaaS) bereitstellen möchten. Neben der eigenen Kubernetes-Distribution kann Rancher auch andere gehostete Container-as-a-Service-(CaaS-)Angebote – wie Amazon Elastic Container Service for Kubernetes (Amazon EKS), Azure Kubernetes Service (AKS) und Google Kubernetes Engine (GKE) – verwalten und dabei konsistente Sicherheitsrichtlinien durchsetzen. Die Plattform kann in Active Directory, LDAP und andere IT-Services integriert werden, um Authentifizierung, rollenbasierte Zugriffskontrolle und Sicherheitsrichtlinien für mehrere Kubernetes-Cluster bereitzustellen.

Red Hat

Red Hat OpenShift Container Platform ist eine Anwendungsplattform, die auf Kubernetes aufbaut. Sie basiert auf bewährten Open-Source-Technologien wie Red Hat Enterprise Linux, CRI-O und Kubernetes und stellt ein umfassendes Anwendungslebenszyklusmanagement bereit.

Red Hat OpenShift Container Platform kann in einer Public Cloud oder On-Premises bereitgestellt werden. Die Plattform automatisiert Builds, Bereitstellungen, Skalierung und Zustandsmanagement von Containern und Anwendungen. Als CNCF-zertifizierte Kubernetes-Distribution unterstützt OpenShift die Portabilität und Interoperabilität von containerisierten Workloads.

OpenShift ist für die Bereitstellung in Private-Cloud- und Hybrid-Cloud-Umgebungen sowie als Managed Service in der Public Cloud verfügbar. Red Hat arbeitet mit Public-Cloud-Anbietern wie Amazon, Microsoft und Google zusammen, um Kunden eine verwaltete OpenShift-PaaS anzubieten.

SUSE

SUSE, der bekannte Anbieter der gleichnamigen Linux-Distribution, ist ebenfalls mit einer Kubernetes-Distribution am CaaS-Markt vertreten.

Die SUSE CaaS Platform unterstützt das Bereitstellen von Kubernetes mit mehreren Masterknoten in Public- und Private-Cloud-Umgebungen. Sie enthält Sicherheitsfunktionen der Enterprise-Klasse wie rollenbasierte Zugriffskontrolle, Image-Scanning und Unterstützung von Transport Layer Security (TLS). Die Plattform umfasst auch ein speziell entwickeltes Containerhost-betriebssystem, eine Containerlaufzeitumgebung und eine Container-Image-Registry.

SUSE nutzt seine Präsenz in Unternehmen, um die CaaS-Plattform zu vermarkten. Wie seine Mitbewerber arbeitet auch SUSE daran, einen integrierten Stack bereitzustellen, der auf den besten Open-Source-Technologien basiert.

VMware

Nach der Übernahme von Pivotal Labs hat VMware den Pivotal Kubernetes Service (PKS) in VMware Tanzu Kubernetes Grid umbenannt.

Die Kubernetes-Distribution von VMware ist in zwei Varianten verfügbar: Tanzu Kubernetes Grid und Tanzu Kubernetes Grid Integrated Edition (TKGI).

Tanzu Kubernetes Grid ist eine CNCF-zertifizierte Kubernetes-Laufzeitumgebung der Enterprise-Klasse, die den Betrieb in einer Multi-Cloud-Infrastruktur optimiert. Sie kann auf vSphere (On-Premises), IaaS (Public Cloud) oder Geräten mit begrenzten Ressourcen (Edge/IoT) bereitgestellt werden.

Tanzu Kubernetes Grid Integrated Edition (TKGI) ist eine produktionsreife, Kubernetes-basierte Containerlösung mit modernen Netzwerkfunktionen, einer privaten Container-Registry und einem umfassenden Lebenszyklusmanagement. Sie ist eng in vSphere, vCenter, vSAN und NSX integriert. Sie kann in einem Rechenzentrum mit einem VMware-Stack oder in einer Public-Cloud-Umgebung mit VMware Cloud (VMC) Foundation bereitgestellt werden.

Managed Services für das Containermanagement

Die folgenden führenden Anbieter bieten Kubernetes als Managed Service an:

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service (EKS) ist ein Managed-Service-Angebot für Kubernetes von AWS. Es basiert auf den wichtigsten Komponenten von AWS, darunter Amazon Elastic Compute Cloud (EC2), Amazon EBS, Amazon Virtual Private Cloud (VPC) und Identity Access Management (IAM). AWS verfügt außerdem über eine integrierte Container-Registry in Form der Amazon Elastic Container Registry (ECR), die einen sicheren, latenzarmen Zugriff auf Container-Images bietet.

Amazon EKS ist für das Monitoring eng in CloudWatch und für die Sicherheit in IAM integriert. AWS App Mesh bietet native Service-Mesh-Funktionen für die in AWS bereitgestellten Workloads. Über AWS Fargate stellt Amazon einen serverlosen Mechanismus zur Ausführung von Containern in EKS bereit.

Zur Beschleunigung von Training und Inferenz beim maschinellen Lernen können Workloads auf Knoten geplant werden, die mit NVIDIA-Grafikprozessoren (GPUs) verbunden sind.

AWS Outposts, die Hybrid-Cloud-Plattform von Amazon, führt EKS On-Premises aus.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) ist eine verwaltete Containermanagementplattform, die in Microsoft Azure zur Verfügung steht. AKS baut auf Azure VMs, Azure Storage, Virtual Networking und Azure Monitoring auf. Azure Container Registry (ACR) kann in derselben Ressourcengruppe bereitgestellt werden wie das AKS-Cluster, um privaten Zugriff auf Container-Images zu ermöglichen.

AKS nutzt Skalierungsgruppen für virtuelle Maschinen sowie Verfügbarkeitsgruppen, um die Anzahl der Knoten in einem Cluster dynamisch zu vergrößern und zu verkleinern.

AKS ist auf Azure Stack Hub verfügbar, dem Hybrid-Cloud-Angebot von Microsoft zur Ausführung von Azure-Diensten in Rechenzentren.

Google Kubernetes Engine

Google Kubernetes Engine (GKE) war einer der ersten Managed Services für Kubernetes in der Public Cloud. Wie andere Public-Cloud-basierte CaaS nutzt GKE die zentralen Dienste der Google Cloud Platform wie Compute Engine, Persistent Disks, VPC und Stackdriver.

Google stellt Kubernetes in On-Premises-Umgebungen und in anderen Public-Cloud-Plattformen über den Service Anthos bereit. Anthos ist eine Steuerungsebene, die in GCP ausgeführt wird, aber den Lebenszyklus von Clustern in Hybrid- und Multi-Cloud-Umgebungen verwaltet.

Google hat GKE um einen verwalteten Istio-Dienst für Service-Mesh-Funktionen erweitert. Außerdem bietet es mit Cloud Run eine serverlose Plattform, – basierend auf dem Open-Source-Projekt Knative – auf der Container ausgeführt werden können, ohne ein Cluster zu starten.

IBM Cloud Kubernetes Service

IBM Cloud Kubernetes Service (IKS) ist ein Managed Service zum Aufbau von Kubernetes-Clustern aus Computing-Hosts, auf denen dann containerisierte Anwendungen in der IBM Cloud bereitgestellt und verwaltet werden können. Als zertifizierter Anbieter bietet IKS intelligente Planung, Selbstreparatur, horizontale Skalierung, Serviceinventarisierung und Lastverteilung, automatisierte Rollouts und Rollbacks sowie Geheimdaten- und Konfigurationsmanagement für moderne Anwendungen.

IBM ist einer der wenigen Cloud-Anbieter, die einen Managed Service für Kubernetes auf Bare Metal anbieten.

Dank der Übernahme von Red Hat bietet IBM die Wahl zwischen Clustern mit Community-Kubernetes und mit OpenShift, die über IKS verfügbar sind.

IBM hat außerdem Cloud Paks entwickelt, eine Reihe von hyperkonvergenten Appliances auf der Basis von Kubernetes und OpenShift, die in Unternehmensrechenzentren eingesetzt werden können.

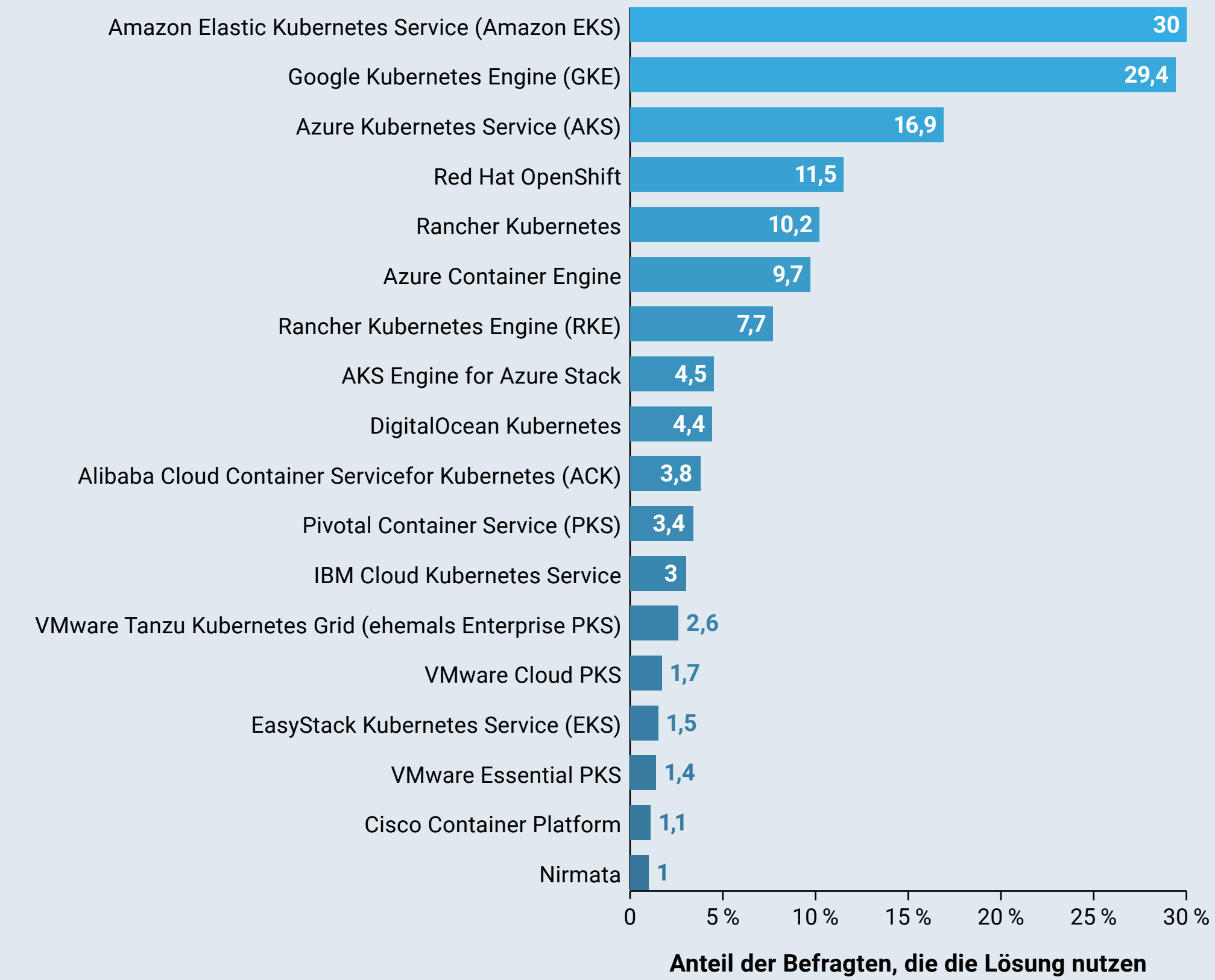


Abb. 2.7: *Die Namen vieler Angebote haben sich geändert, seit die Frage gestellt wurde. In Kombination mit den Angeboten von Pivotal und VMware könnte Tanzu bei der Veröffentlichung der Umfrageergebnisse für 2020 eine bessere Position erreichen.*

Weitere Managed Services für Kubernetes

Neben den oben aufgeführten Diensten stehen Kunden auch Managed Kubernetes Services von Alibaba Cloud Container Service, DigitalOcean Kubernetes, Huawei Cloud Container Engine und Platform9 zur Auswahl.

Kundenbasis von Kubernetes-as-a-Service von Cloud-Anbietern

Zum Zeitpunkt der CNCF-Umfrage von 2017 führten 25 Prozent der Kubernetes-Benutzer Container mit Google Kubernetes Engine (GKE) aus. AWS warb noch für seinen Elastic Container Service (ECS) und Microsoft Azure Container Service war gerade erst auf den Markt gekommen. Dennoch war bereits klar, dass Kubernetes zum Containerorchestrator der Wahl werden würde. Die teilweise noch sehr neuen Kubernetes-Angebote wurden bereits enthusiastisch eingesetzt: 52 Prozent der Kubernetes-Benutzer nutzten eine AWS-Umgebung, 34 Prozent Google Cloud Platform (GCP), 21 Prozent Azure und 15 Prozent VMware. Die Hälfte aller Kubernetes-Workloads wurde in AWS ausgeführt, obwohl dies damals nur in VMs möglich war und die AWS-Kunden ihre Kubernetes-Umgebungen selbst verwalten mussten.

Auf der AWS re:Invent dieses Jahres wurde Amazon Elastic Container Service for Kubernetes (EKS) angekündigt mit allgemeiner Verfügbarkeit ab Mitte 2018 und AWS wurde kurz darauf Mitglied der CNCF. Kubernetes hatte das Rennen um die Containerorchestrierung gewonnen, aber Unternehmen betrieben Container meist dort, wo sie auch andere Workloads ausführten. Deshalb wurden Amazon EKS und Azure Kubernetes Service (AKS) im Jahr 2019 enthusiastisch begrüßt.

In der jüngsten CNCF-Umfrage von 2019 gab es für die Frage, wie Container verwaltet wurden, mehr als 100 Auswahlmöglichkeiten, wobei Amazon EKS (30 Prozent) von Kubernetes-Benutzern etwas häufiger verwendet wurde als GKE (29 Prozent). Es folgten das Angebot von Microsoft, Red Hat OpenShift und Rancher Kubernetes.

PRISMA
BY PALO ALTO NETWORKS

Kubernetes hat sich weiter-entwickelt. Ihre Sicherheit sollte nachziehen.

Kubernetes bietet zahlreiche integrierte Sicherheitsfunktionen, ist dadurch aber nicht automatisch von Haus aus sicher. Der Schutz des Abhängigkeitsmanagement lässt noch zu wünschen übrig und es entstehen immer neue Angriffsvektoren, wie etwa schädliche Container. Trotz Fortschritten in puncto Sicherheit bleiben die APIs die von Angreifern am häufigsten genutzten Einfallstore in Kubernetes.

Die gute Nachricht ist, dass Sicherheitsteams in den letzten Jahren viel über den Schutz von Kubernetes-Umgebungen und in Containern ausgeführten Anwendungen gelernt haben. Solchen Bedrohungen kann mit einer Kombination aus Workflows und Tools, die Entwickler, Sicherheitsteams und den IT-Betrieb (DevSecOps) einbinden, einfacher entgegengewirkt werden. Zum Beispiel können schädliche Container und andere Angriffsvektoren mit Tools für Anomalieerkennung und Malwarescans leicht erkannt werden.

Robert Haynes, Cloud Security Evangelist bei Prisma Cloud von Palo Alto Networks, erläutert, was abgesehen von den nativ integrierten Funktionen in puncto Kubernetes-Sicherheit noch möglich und nötig ist. Außerdem blickt er auf die Entwicklung der Schwachstellen von Kubernetes seit den ersten API-Angriffen vor wenigen Jahren zurück. Die ersten gefährlichen Angriffe „zeigten uns etwas, das wir bereits wussten: dass Kubernetes eine Software ist, die wie jede andere Software Schwachstellen hat“, so Haynes.

Auf SoundCloud anhören

Robert Haynes *blickt auf eine Karriere zurück, die ihn vom einfachen Helpdesk-Mitarbeiter bis in die luftigen Höhen eines Unix-Systemadministrators und dann wieder zurück ins Marketing führte. Aktuell gehört er zum Prisma Cloud-Team von Palo Alto Networks, wo es ihm großes Vergnügen bereitet, Geschichten über Sicherheit, Technologie und Menschen zu erzählen.*

Mittel gegen die Kubernetes-Komplexitätsmüdigkeit

Je größer die in einem Unternehmen genutzten Kubernetes- und cloudnativen Umgebungen werden, desto komplexer gestaltet sich das Clustermanagement. Vielerorts macht sich dann „Kubernetes-Komplexitätsmüdigkeit“ breit.

In diesem Podcast sprechen Ravi Lachhman, Evangelist bei Harness, und Frank Moley, Senior Technical Engineering Manager bei DataStax, über dieses Thema und mögliche Lösungen.

Die Müdigkeit lässt sich zu einem guten Teil darauf zurückführen, dass sich DevOps-Teams bei der Umstellung in neue Fachgebiete einarbeiten müssen, oft in einem Umfang, der ihre Schmerzgrenze überschreitet. Angehörige von Infrastrukturteams müssen sich beispielsweise mit den Anforderungen von Entwicklern vertraut machen, während die Arbeit der Entwickler in zunehmendem Maße auch Aufgaben in Zusammenhang mit der Infrastruktur umfasst.

Um Ihren DevOps-Teams zu helfen, Ermüdungserscheinungen zu vermeiden, müssen Sie die eigenen Grenzen kennen. Lachhman empfiehlt, Komponenten „häppchenweise“ zu kaufen und sich im Umgang mit Lücken zu üben, um seine Ziele zu erreichen.

Auf SoundCloud anhören

Ravi Lachhman *ist Evangelist bei Harness und war davor Evangelist bei AppDynamics. Außerdem hatte er verschiedene Positionen im Verkauf und in technischen Bereichen bei Mesosphere, Red Hat und IBM inne.*

Frank Moley *ist ein Senior Technical Engineering Manager bei DataStax und Autor für LinkedIn Learning. Seine Schwerpunktthemen sind cloudnatives Computing, Microservicearchitekturen, Plattformtechnologie und Sicherheit.*

Kelsey Hightower über seinen ganz persönlichen Werdegang mit Kubernetes

In diesem Podcast sprechen wir mit einer bekannten Persönlichkeit im Kubernetes-Ökosystem: Kelsey Hightower, Principal Developer Advocate bei Google Cloud und früheres Mitglied des Technical Oversight Committees der Cloud Native Computing Foundation (CNCF), das alle CNCF-Projekte leitet, darunter auch Kubernetes. Hightower spricht über seine Rolle bei Kubernetes und die bevorstehenden Herausforderungen.

„Ich erinnere mich noch daran, wie ich die Codebasis durchging und feststellen musste, dass darin noch nicht einmal erklärt wurde, wie die Installation funktioniert“, berichtet Hightower von seiner ersten Begegnung mit Kubernetes im Jahr 2015. „Das Erste, was ich tat, war, gewissermaßen eine der allerersten Installationsanleitungen zu schreiben.“

Jetzt will Hightower mit zunehmendem Reifegrad der Plattform die Einarbeitung erleichtern: „Es sollte nicht notwendig sein, dass Entwickler Experten für das Kubernetes-Netzwerk werden, damit sie ihren Job als Softwareingenieure machen können.“

Auf SoundCloud anhören

Kelsey Hightower ist ein Principal Developer Advocate bei Google Cloud. Er hat an der Entwicklung und Verbesserung vieler Google Cloud-Produkte mitgewirkt, darunter die Kubernetes-Engine von Google, Google Cloud Functions und das API-Gateway von Apigees.

Schlusswort

THENEWSTACK

Inzwischen gibt es Kubernetes seit gut fünf Jahren und der Architekturaufbau ist weitestgehend abgeschlossen. Die Begriffe Pod, Knoten und Cluster sind im Vokabular zur Beschreibung horizontal skalierbarer Architekturen verankert. Diskussionen über Kubernetes drehen sich nicht mehr um das beeindruckende Grundgerüst, sondern darum, was auf Kubernetes aufgebaut wird.

Kubernetes wird langweilig, eben weil es nun robust und skalierbar ist. Das Grundgerüst funktioniert einfach.

Man kann leicht vergessen, dass dieses Grundgerüst noch vor wenigen Jahren alles andere als langweilig war. Aber seit der Veröffentlichung des ersten E-Books von The New Stack über Kubernetes im Jahr 2017 entstand mit einer kontinuierlichen Abfolge von Releases für die Containerorchestrierung ein iterativer Prozess zum Aufbau einer soliden Architekturbasis für horizontal skalierbare, verteilte Anwendungen.

Die Herausforderung besteht jetzt darin, optimal auf Kubernetes aufzubauen. In der Welt der horizontal skalierbaren Architekturen wird der Code auf Container verteilt. Der Code befindet sich also in Containern. Diese Container sind Prozesse, die in einem Betriebssystem ausgeführt werden, und als solche portierbar sind. Dem Konzept nach sind Container unveränderlich. Wenn der in ihnen enthaltene Anwendungscode aktualisiert wird, werden sie durch neue Container ersetzt. Die unveränderliche Infrastruktur basiert auf dem Konzept, dass eine Anwendung in einer konsistenten Umgebung bereitgestellt werden kann. Container mit aktualisiertem Code lösen Container mit älterem Code nahtlos ab und ermöglichen so die kontinuierliche Weiterentwicklung von Anwendungen.

Die Herausforderung beim Aufsetzen von Anwendungen auf Kubernetes ergibt sich durch diese Natur der Container als unveränderliche Prozesse. Ein Container ist zustandslos. Um Kubernetes für Unternehmen nutzbar zu machen, müssen Softwareingenieure Schnittstellen entwickeln, über die sie die zustandslosen Container miteinander verbinden können.

Das Container Network Interface (CNI) verknüpft Container und zugehörige Plug-ins mit dem Netzwerk, das ein Unternehmen zum Ausführen einer Anwendung nutzt. Der Mehrwert einer

solchen Umgebung ergibt sich aus der breiten Unterstützung für die Verbindung von Plug-ins mit Netzwerken. Die Unternehmen, die eine gemeinsame CNI entwickeln, sind ein Indiz für das Erfolgsrezept der Kubernetes-Community: Bei Kubernetes-Projekten tritt der Geist der Zusammenarbeit in einer offenen, gemeinschaftlichen Entwicklung zutage. Das Grundgerüst von Kubernetes mag langweilig geworden sein, aber die Menschen, die diese Architektur weiterentwickeln, stecken voller Energie und arbeiten mit Begeisterung am Ausbau einer Architektur für eines der ersten Betriebssysteme für Software und Services auf einer zukunftsfähigen, verteilten Infrastruktur.

Doch Kubernetes ist immer noch relativ neu. Clientbasierte Betriebssysteme wurden für PCs oder Server, hauptsächlich für monolithische Anwendungen, konzipiert. Im Serverbereich dominiert Linux. Container basieren auf Linux – sie sind eine Erweiterung des Linux-Kernels. Mit Kubernetes wird der Container zum Basisprozess für die Orchestrierung des Codes, der aus verschiedenen, einzeln ausgeführten Komponenten besteht. Kubernetes selbst ist eine Steuerungsebene, die von den Cloud- oder Clientarchitekturen unabhängig ist. Das ermöglicht die Verwaltung der Steuerungsebene und der zugehörigen Rechen-, Netzwerk- und Speicherkomponenten über APIs.

Die relative Neuheit von Kubernetes spüren vor allem die Entwickler. Die Kubernetes-Architektur ist noch nicht wirklich entwicklerfreundlich. Die Zeit von Platform-as-a-Service (PaaS) geht vorbei; über die letzten drei Jahre hat sie erheblich an Bedeutung verloren. PaaS ist eine Methode, um Anwendungen auf Cloud-Services und Unternehmensumgebungen auszuführen. Eine PaaS wird als Anwendungsstruktur aufgebaut und zusammengesetzt. Aber selbst die Besten sind nicht für jeden Einzelfall geeignet, insbesondere für ältere, monolithische Anwendungen.

Mit Containern wird das grundlegend anders. Code kann in einem Container ausgeführt und weitaus einfacher und mit weniger Zeitaufwand verwaltet werden. Aber Entwicklern stehen immer noch nur wenige Möglichkeiten zum Aufbau von Services auf Kubernetes zur Verfügung. Hier kommen Containers-as-a-Service (CaaS) ins Spiel, die auch als Containermanagement-plattformen bekannt sind. Mit ihnen können verschiedene cloudnative Services eingebunden werden, sodass sie sowohl auf monolithischen Unternehmensarchitekturen als auch auf

modernen Microservices-Architekturen ausgeführt werden können. Eine Managementplattform führt mehrere Services auf Kubernetes aus. Den Anfang machen die CPU und die physische Infrastruktur. Die Managementplattform umfasst das für Container optimierte Betriebssystem, die Containerlaufzeitumgebung, containernative Netzwerk-, Speicher-, Sicherheitstechnologie, die Registry, Beobachtbarkeit, die Engine für die Containerorchestrierung selbst und mehr.

CaaS-Plattformen sind noch nicht ausgereift. Managed Services werden an Bedeutung gewinnen, wodurch Entwickler entlastet werden.

Bis dahin werden sich die Rollen – sozusagen die Personas – der verschiedenen Beteiligten herauskristallisieren: der Entwickler, der für die Konfiguration der Services verantwortlichen Personen und derjenigen, die die Richtlinien für Kubernetes festlegen. Container erobern die IT-Welt und definieren neu, welche Rolle ein Betriebssystem spielt, wenn es standortunabhängig auf einer Infrastruktur ausgeführt wird.

Vielleicht ist Kubernetes doch noch nicht so langweilig. Aber es ist auf dem besten Weg dorthin. Wie Kelsey Hightower so gern sagt, besteht unsere nächste Herausforderung darin, es verschwinden zu lassen.

Offenlegung

THENEWSTACK

Zusätzlich zu den Sponsoren unseres E-Books sind die folgenden hierin erwähnten Unternehmen Sponsoren von The New Stack:

Accurics, AppDynamics, Aspen Mesh, Amazon Web Services, CircleCI, Citrix, CloudBees, Cloud Foundry Foundation, Cockroach Labs, Dell Technologies, Diamanti, Futurewei, GitLab, HAProxy, HashiCorp, Honeycomb, InfluxData, Lightbend, Lightstep, LogDNA, MayaData, MongoDB, New Relic, NS1, Packet, PagerDuty, Portworx, Red Hat, Redis Labs, Rezilion, SaltStack, SAP, Sentry, Snyk, Sonatype, The Linux Foundation, TriggerMesh und VMware.