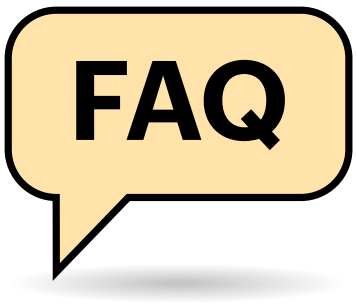


Container verwalten mit Kubernetes

Docker vereinfacht den Betrieb von Serversoftware, hat im Funktionsumfang aber Grenzen – wenn die Anforderungen steigen, wechseln viele zu Kubernetes und stehen vor neuen Fragen. Antworten auf die häufigsten von ihnen haben wir zusammengetragen.

Von Jan Mahn

Darum Kubernetes

Ich arbeite bereits länger mit Docker und habe nur eine grobe Ahnung, was Kubernetes macht. Warum braucht man Kubernetes genau?

Wenn Sie bisher mit dem Funktionsumfang von Docker zufrieden sind und nicht vorhaben, Ihre Infrastruktur zu skalieren, brauchen Sie Kubernetes wahrscheinlich nicht. Es lohnt sich immer dann, wenn Sie mit Docker-Bordmitteln nicht mehr weiterkommen. Ein typisches Beispiel: Docker kann Container mit dem Attribut `depends_on` in einer Compose-Datei nur ganz grob in der richtigen Reihenfolge starten (etwa zuerst die Datenbank, dann den Webserver). Dabei kann es mal passieren, dass eine Anwendung nach Updates ein paar Sekunden oder Minuten nicht erreichbar ist. Bei häufigen Updates und sehr anspruchsvollen Nutzern wollen Sie so etwas verhindern – Kubernetes beherrscht unter anderem sogenanntes Rolling Update, bei dem immer ein funktionierender Container bereitsteht und wirklich keine einzige Anfrage ins Leere läuft.

Anders als manchmal behauptet, braucht Kubernetes nicht zwangsläufig mehrere Server, die im Cluster laufen, sondern kann auch auf einem einzelnen Server Docker ersetzen. Im Cluster mit mehreren Maschinen spielt es seine Stärken aber so richtig aus. Während Docker die Container nur starten, stoppen und löschen kann, orchestriert Kubernetes sie. Soll heißen: Kubernetes kann entscheiden, auf welchem Server ein Container am besten läuft und wie viele identische Container gebraucht werden. Kommt ein Cluster mal an seine Grenzen, beschaffen Sie einfach weitere Hardware und erweitern den Serververbund.

Kann man nicht auch auf Docker Swarm wechseln, um mehrere Server mit Containern zu betreiben?

Docker Swarm löst ein ähnliches Problem und es ist in der Tat einfacher, die bekannte Syntax von Docker-Compose einfach weiterzuverwenden und mehrere Server als Docker-Swarm zu betreiben. Das Problem ist die etwas ungewisse Zukunft: Docker Swarm wanderte im Rahmen eines Abverkaufs zusammen mit der Enterprise-Sparte von der Docker Inc. zur Firma Mirantis. Die verdient ihr Geld überwiegend mit Dienstleistung und Produkten rund um Kubernetes und trotz aller Beteuerungen, Swarm nicht einzustellen, kann niemand garantieren, dass es Swarm in fünf Jahren noch gibt.

Kubernetes dagegen hat Google an die CNCF, eine Stiftung unter dem Dach der Linux Foundation, übergeben. Die Software ist der Branchenstandard, Open Source und auf alle Fälle zukunftssicher. Und noch ein Argument spricht für Kubernetes: Das Ökosystem ist überragend. Egal, welches Problem Sie mit Kubernetes lösen wollen – fast immer hatte schon jemand ein ähnliches und fast immer hat schon jemand ein passendes Open-Source-Projekt mit einer Lösung gebaut.

Wir nutzen bereits Docker für viele Serverdienste und denken darüber nach, auf Kubernetes umzusteigen. Wie schwierig ist das?

Technisch ist der Umstieg von Docker auf Kubernetes überschaubar. Das liegt daran, dass die Images (die zum Beispiel aus dem Docker-Hub oder anderen Registries kommen) auch von der Container-Runtime in Kubernetes genutzt werden. Es gibt also kein eigenes Kubernetes-Image-Format. Zeit investieren müssen Sie lediglich, um Docker-Compose-YAML durch Kubernetes-YAML zu ersetzen. Weil Kubernetes viel mehr Einstellmöglichkeiten kennt, werden die neuen YAML-Dateien deutlich länger und es gibt ein paar neue Konzepte zu lernen (für Volumes, Netzwerk und Konfigurationen). Einen ausführlichen Einstieg in Kubernetes für Docker-Kenner lesen Sie in [1].

Am besten lernen Sie direkt nach den ersten Gehversuchen mit Kubernetes-YAML die Syntax eines Paketmanagers wie Helm. Dann wird das Installieren und Aktualisieren von Kubernetes-Umgebungen schlagartig einfacher [2].

Kubernetes-API

Kubernetes hat ein API, funktioniert das ähnlich wie der Docker-Socket?

Das Kubernetes-API erfüllt einen ähnlichen Zweck, man kann darüber Kubernetes-Objekte anlegen und bearbeiten. Der zentrale Unterschied zur Docker-Welt: Das API kennt Authentifizierung und Autorisierung. Cluster-Admins können also genau steuern, wer welche Objekte sehen und ändern kann. Kubernetes ist damit für Umgebungen vorbereitet, in denen größere Teams an einem Cluster arbeiten und nicht jeder alles ändern darf. Dazu gehört auch, dass die meisten Objekte in sogenannten Namespaces liegen, über die man einen großen Cluster aufteilen kann.

In meinen Docker-Umgebungen nutze ich öfter den Trick, den Docker-Socket als Volume mit der Definition `/var/run/docker.sock:/var/run/docker.sock` in einen Container hereinzureichen, damit

der Container andere Container sehen und bearbeiten kann. Geht so etwas auch mit Kubernetes?

Ja, auch in Kubernetes können Container auf das Kubernetes-API zugreifen und weil es eine Berechtigungsverwaltung gibt, können Sie auch viel genauer steuern, was der Container darf – das ist also wesentlich sicherer. Viele Anwendungen aus dem Kubernetes-Ökosystem nutzen dieses Konzept auch fleißig aus. Allerdings funktioniert das Kubernetes-API völlig anders als der Docker-Socket, Sie können also nicht einfach denselben Code im Container nutzen.

Versionen und Distributionen

Kubernetes wird ja ständig weiterentwickelt und es gibt regelmäßig neue Releases. Wie wichtig ist es, immer die neueste Version im Cluster zu nutzen?

Die Versionierung orientiert sich am Konzept Semantic Versioning [3], die Versionsnummer besteht aus Major-, Minor- und Patch-Version. Die Major-Version 1 wurde seit dem ersten Release im Juli 2015 noch nicht verändert, obwohl man einige Änderungen und Abkündigungen schon als Breaking-Change einstufen könnte. Stattdessen gibt es immer drei Minor-Versionen, die parallel unterstützt und mit Patches versorgt werden. Aktuell sind das die Versionen 1.26, 1.27 und 1.28. Anders als bei Anwendungen für Endnutzer, bei denen man sich meist auf neue Features freut, muss man bei einer Infrastruktursoftware wie Kubernetes nicht sofort jede neue Minor-Version installieren. Die neuen Funktionen lösen oft sehr spezielle Probleme, die viele noch nie hatten. Im Gegenzug werden mit jeder Minor-Version lang verkündete Deprecations umgesetzt, es fallen also Dinge weg – ein Update erfordert also manchmal kleine Anpassungen an Ihren YAML-Dateien.

Zügig installieren sollten Sie aber jeweils die Patch-Versionen unterhalb Ihrer Minor-Version. Die Patches beseitigen Probleme und schließen Sicherheitslücken. Als Cluster-Admin sollten Sie sich außerdem im Kalender notieren, bis wann die verwendete Minor-Version Support bekommt, und rechtzeitig den Umstieg vorbereiten. Version 1.28 bekommt zum Beispiel noch bis zum 28. Oktober 2024 Updates, die neue Version 1.29 soll am 5. Dezember 2023 erscheinen. In der Regel dauert es vier Monate bis zur nächsten Minor-Version. Mehr Informationen zu den Terminen finden Sie über ct.de/y13s.

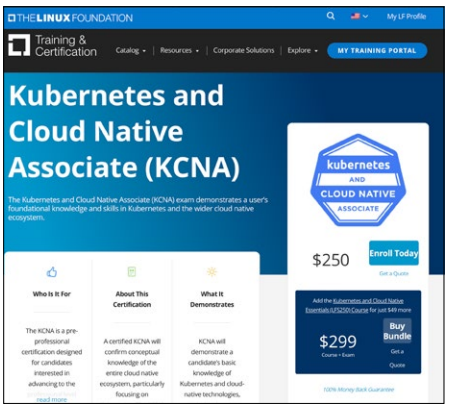
Wie Linux wird Kubernetes in Form von Distributionen angeboten. Welche sollte ich da nehmen?

Kubernetes ist eben keine einzelne Binärdatei, die man auf kubernetes.io herunterladen und installieren könnte. Zu einem Cluster gehören mehrere Komponenten, die zusammenspielen müssen. Daher gibt es Distributionen, zu denen auch eine Installationsroutine gehört. Welche Distribution Sie nutzen, hängt von der Umgebung ab. Wenn Sie Kubernetes als Komplettprodukt (Managed Kubernetes) bei einem großen Cloudprovider wie Google, Amazon oder Azure mieten, bekommen Sie die Kubernetes-Engine des jeweiligen Anbieters und müssen sich um Einrichtung und Updates gar nicht kümmern. Für eine Distribution entscheiden müssen Sie sich nur, wenn Sie den Cluster selbst auf eigener oder gemieteter Hardware (oder in gemieteten VMs) einrichten. Für kleine und mittlere Umgebungen empfehlen wir die Distribution k3s, die mittlerweile ein CNCF-Projekt ist. Damit haben Sie schnell einen Cluster eingerichtet. Für größere Umgebungen ist Rancher einen Blick wert – dazu gehört eine grafische Oberfläche im Browser, über die Sie Server einrichten und verwalten. Eine Liste mit weiteren Distributionen finden Sie über ct.de/y13s.

Qualifikation

Ich arbeite jetzt schon länger mit Kubernetes. Wie weise ich gegenüber einem potenziellen Arbeitgeber nach, dass ich von der Materie etwas verstehe?

Kubernetes-Experten sind verständlicherweise gefragt. Um die Fähigkeiten nachzuweisen, hat die Linux Foundation ein mehrteiliges Zertifizierungsprogramm entwickelt. Das Prinzip ist immer gleich: Sie bereiten sich im Heimstudium oder bei einem Schulungsanbieter auf eine Prüfung vor und absolvieren diese daheim am Computer. Dabei müssen Sie die Webcam aktivieren und einem Prüfer durch



Die Linux Foundation bietet ein Zertifizierungsprogramm für Kubernetes-Kenner an. Mit der Prüfung zum KCNA beginnt der Zertifizierungspfad.

Drehen der Webcam beweisen, dass Sie keine Spickzettel versteckt haben. Als Hilfsmittel zugelassen ist bei den meisten Prüfungen nur die offizielle Kubernetes-Doku. Geprüft wird also mehr Anwendungswissen und weniger die Fähigkeit, Dinge stumpf auswendig zu lernen.

Die einfachste und mit 250 US-Dollar günstigste Prüfung ist die zum „Kubernetes and Cloud Native Associate“ (KCNA). Wesentlich anspruchsvoller ist der „Certified Kubernetes Administrator“ (CKA), der sich an Cluster-Admins richtet. Der „Certified Kubernetes Application Developer“ (CKAD) weist nach, dass er auch Anwendungen entwickeln kann, die mit dem Kubernetes-API sprechen, und dass er weiß, wie man Anwendungen am besten für den Betrieb in Containern vorbereitet. Als „Certified Kubernetes Security Specialist“ (CKS) müssen Sie die Sicherheitsfunktionen durchschauen. 400 US-Dollar für die Prüfungen und ein paar Monate Vorbereitungszeit müssen Sie einplanen. Offiziell sind die Zertifikate drei Jahre gültig.

Eine Übersicht über die Kubernetes-Zertifizierungen der Linux Foundation finden Sie über ct.de/y13s. *(jam@ct.de)*

Literatur

[1] Jan Mahn, Containerkompetenzoffensive, Auf dem Lernpfad zum Kubernetes-Kenner, Teil 1, c’t 22/2022, S. 162
[2] Jan Mahn, Containerverpacker, Kubernetes-Anwendungen mit Helm paketieren, c’t 11/2023, S. 164
[3] Jan Mahn, Bedeutung 2.0.0, Warum Versionsnummern nicht willkürlich sind, c’t 24/2021, S. 128

Dokumentation: **ct.de/y13s**