c't Teil 1: Containerkompetenzoffensive

Kubernetes-Experten sind gefragt und viele Docker-Nutzer würden die andere Seite des Container-Universums gern mal kennenlernen – wäre das Ökosystem nicht so groß und undurchsichtig. Mit unserer ausführlichen Praxis-Reihe gelingt der Umstieg: Der erste Teil zeigt, wie Sie aus drei Linux-Servern einen Cluster bauen.

Von Jan Mahn

c't 22/2022, Seite 164

kompakt

- Kubernetes führt dieselben Container-Abbilder aus wie Docker, mehrere Server können als Cluster zusammenarbeiten.
- Cluster kann man fertig eingerichtet mieten oder auf eigenen virtuellen Maschinen betreiben. Entscheidend ist die Wahl der Kubernetes-Distribution.
- Gesteuert wird Kubernetes per Kommandozeile oder grafischer Oberfläche aus der Ferne.

Das Softwareprojekt ist zu groß geworden für einen einzigen Docker-Server, Ihre Chefs erwarten von Ihnen jetzt Kubernetes-Erfahrung oder Sie wollen aus eigenem Antrieb verstehen, wie man seine Container mit der Software betreibt, die auch Schwergewichte wie Netflix, Spotify und Banken im Einsatz haben. Gründe, sich heute an den Einstieg in Kubernetes zu wagen, gibt es viele – Voraussetzung ist lediglich ein souveräner Umgang mit Docker oder einer anderen Container-Umgebung wie Podman. Wenn Sie noch nicht überzeugt sind, warum Sie Kubernetes brauchen und lernen sollten, finden Sie Argumente auf Seite 166. Aber ohne dass man einige Monate lang Container betrieben, Abbilder heruntergeladen und eigene gebaut hat, sollte man die Finger von Kubernetes lassen; Frust wäre garantiert. Eine Einführung in Docker und den aktuellen Stand lesen Sie in [1].

Auch für erfahrene Docker-Nutzer führt der naheliegendste Weg in die Kubernetes-Welt leider schnell in eine Sackgasse. Beim ersten Blick auf die offizielle Kubernetes-Dokumentation wird man ziemlich zuverlässig erschlagen. Die liegt unter docs.kubernetes.io und wird später ein zuverlässiger Begleiter. Wie Sie vielleicht schon mitbekommen haben, stammt Kubernetes ursprünglich aus dem Hause Google und wird jetzt als branchenübergreifendes Open-Source-Projekt entwickelt. Daher arbeitet ein Team aus Dokumentationsprofis daran, die Texte auf dem

aktuellen Stand zu halten und leistet gute Arbeit. Die Doku verrät jedes Detail und ist ein unverzichtbares Nachschlagewerk, denn auswendig lernen kann niemand alle Funktionen von Kubernetes. Für Einsteiger ist dieses Werk jedoch keine Empfehlung – das liegt auch daran, dass Sie neben Kubernetes auch gleich ein ganzes Ökosystem aus Open-Source-Projekten kennenlernen müssen, die im Zusammenspiel mit Kubernetes funktionieren. Und oft gibt es auch mehrere Projekte, die dasselbe Problem lösen. Die Kubernetes-Doku allein enthält also nur einen Teil der Wahrheit. Im Ökosystem gibt es aber so viele Pfade und Verzweigungen, dass man sich allzu leicht verlaufen kann.

Darum Kubernetes

Kubernetes ist weit mehr als eine Docker-Alternative, die im Cluster-Betrieb läuft. Selbst im Ein-Server-Betrieb kann Kubernetes weit mehr als Docker: Zunächst sind da die Konfigurationsmöglichkeiten, die den Lebenszyklus eines Containers von Anfang bis Ende kontrollierbar machen. Was in einer Docker-Compose-Datei eine Zeile ist, kann man bei Bedarf in einer Kubernetes-YAML-Datei oft auch in 20 Zeilen haarklein steuern. Sie haben zum Beispiel Ärger mit der Startreihenfolge Ihrer Container, die voneinander abhängen, und die Werkzeuge von Docker reichen nicht aus, dass sie korrekt aufeinander warten? Kubernetes hat Mittel dagegen. Durch diese Steuerung des Containerlebenszyklus sind auch perfekte Rolling Updates kein Problem mehr. Einmal richtig eingestellt, können Sie Anwendungen aktualisieren, ohne dass Nutzer einen Ausfall bemerken. Mit etablierten Werkzeugen wie Helm wird auch das Installieren und Weitergeben von Containerzusammenstellungen viel einfacher als mit Docker-Compose-Dateien.

Im Hintergrund stellt Kubernetes eine Programmierschnittstelle bereit, auf die Fernsteuerungssoftware von außen und auch Container selbst zugreifen können. Neben den Zuständen von Containern kann das API auch alle Formen von Konfigurationen verwalten – und anders als der Docker-Socket hat es eine Benutzer- und Berechtigungsverwaltung.

Eigene Erfahrungen statt Theorie

Dieser Artikel möchte einen möglichen Weg durch das Profi-Container-Dickicht aufzeigen. Nicht den einzigen Weg und sicher nicht den besten Weg für alle erdenklichen Umgebungen, aber einen, der sich für Docker-Kenner bewährt hat. Wo es angebracht ist, erhalten Sie Hinweise auf alternative Routen. Dieser Artikel ist Teil einer Serie, denn nach einem einzelnen Text sind Sie noch kein Kubernetes-Experte. Im Mittelpunkt steht das Ausprobieren und Nachbauen: Anhand einer Anwendung, die aus einer Ein-Server-Docker-Umgebung in die Kubernetes-Welt umziehen soll, lernen Sie Kubernetes-Konzepte, Werkzeuge aus dem Ökosystem und erprobte Lösungsansätze kennen. Auf dem Weg verinnerlichen Sie Begriffe und Kommandozeilenbefehle ganz automatisch.

> Kubernetes allein ist nicht der Schlüssel zum Erfolg – es ist das Ökosystem aus Open-Source-P
>
> Kubernetes allein ist nicht der Schlüssel zum Erfolg – es ist das Ökosystem aus Open-Source-Projekten. Die Landkarte der Cloud Native Computing Foundation (landscape.cncf.io) zeigt, was es im Kubernetes-Universum alles zu entdecken gibt.

Die Voraussetzungen zum Nachvollziehen dieser Einführung sind für Administratoren und Entwickler mit etwas Linux-Erfahrung keine unüberwindbare Hürde: Sie brauchen drei (virtualisierte) Server, die bestenfalls über öffentliche IP-Adressen im Internet ansprechbar sind und sich – sofern möglich – auch über ein internes Netz erreichen. Außerdem eine Domain, für die Sie Subdomains verwalten können. Nur dann können Sie später auch Experimente mit TLS und der Zertifikatsbeschaffung nachvollziehen.

Theoretisch könnten Sie auch mit einer einzigen Maschine Ihre Kubernetes-Karriere beginnen und selbst die lokale Entwicklermaschine mit installiertem Docker Desktop reicht aus, um einen Kubernetes-Cluster zu simulieren. Wenn Sie unter Windows, macOS und neuerdings auch Linux den Einstellungsdialog von Docker Desktop öffnen, können Sie beim Menüpunkt Kubernetes einen Single-Node-Cluster hochfahren. Die Funktion richtet sich vor allem an Entwickler, die testen müssen, wie sich ihr Container unter Kubernetes-Bedingungen verhält. Zum Lernen der Grundlagen ist das aber nicht die beste Wahl, weil Kubernetes erst im richtigen Cluster spannend wird.

In diesem ersten Artikel soll es nicht um Anwendungsentwicklung in Containern gehen, sondern um den Bau eines richtigen Clusters aus mehreren Maschinen. Am besten suchen Sie sich für die Experimente einen Cloudprovider und ordern dort drei kleine virtuelle Linux-VMs zum Stundentarif – für diesen Artikel kommt Ubuntu Server 20.04 LTS zum Einsatz. In [2] finden Sie eine Marktübersicht europäischer und US-Anbieter in diesem Geschäft. Trotz gestiegener Energiekosten bekommen Sie für unter 20 Euro im Monat (bei Dauerbetrieb) eine Testumgebung mit drei Maschinen. Auch wenn Sie schon absehen können, dass Sie sich beruflich niemals mit der Installation und dem Betrieb eines Kubernetes-Clusters beschäftigen müssen, weil Ihr Unternehmen ein Managed-Kubernetes-Produkt eines Providers nutzt, ist es fürs Verständnis ungemein hilfreich, mal selbst einen Cluster gebaut zu haben.

Drei Server, ein Cluster

Entstehen soll im Folgenden ein Zusammenschluss aus mehreren Servern, die am selben Ziel arbeiten: Ihre containerisierte Anwendung stabil, skalierbar und redundant auszuführen. Aber warum gleich drei Maschinen, ein Testcluster könnte man doch auch mit zwei Maschinen günstiger simulieren – oder nicht? Die Zahl drei werden Sie in der Kubernetes-Welt noch öfter lesen und dafür gibt es einen guten Grund: Kubernetes nutzt (in den allermeisten Fällen) die Key-Value-Datenbank etcd für die Verwaltung des Cluster-Zustands. Diese Datenbank kann redundant über mehrere Maschinen verteilt laufen und setzt auf den Konsens-Algorithmus Raft – und der wiederum funktioniert am besten mit einer ungeraden Anzahl Maschinen. Warum das so ist und was das mit Demokratie unter Servern zu tun hat, lesen Sie in [3].

Ihr erster Cluster soll gleich ausfallsicher arbeiten. Daher bekommen alle drei Maschinen (in der Kubernetes-Welt Nodes genannt) die Master-Rolle und eine etcd-Kopie. Das versetzt Sie in die Lage, dass Sie die Server (etwa bei Updates) problemlos nacheinander neu starten können. Den Ausfall einer Maschine steckt der Cluster dank Raft-Algorithmus weg, die anderen sind weiter arbeitsfähig. Servern mit der Master-Rolle kommt im Cluster eine besondere Aufgabe zu: Sie stellen ein API nach innen und auf Wunsch auch nach außen bereit, über das man den Zustand des Systems abfragen und verändern kann. Nach außen nutzen alle Verwaltungswerkzeuge für Admins

dieses API, nach innen steht es privilegierten Containern zur Verfügung, die darüber den Zustand von anderen Objekten und Containern erfahren und verändern können. Docker-Nutzer kennen dieses Prinzip von speziellen Containern, die den Unix-Socket /var/run/docker.sock als Volume bekommen, um andere Container zu steuern – die grafische Oberfläche Portainer ist ein weit verbreitetes Beispiel aus der Docker-Welt.

Neben diesen Master-Nodes kennt Kubernetes reine Worker-Nodes, die von dem oder den Mastern kontrolliert werden und nur Container ausführen, ohne sich mit etcd und Verwaltung zu belasten – für den Einstieg reicht es aus, die Master auch als Worker einzusetzen. In vielen produktiven Clustern laufen drei Master (das ist für etcd eine gute Größe) und beliebig viele Worker (die theoretische Obergrenze liegt bei 5000 Nodes pro Cluster).

Zum Steuern Ihrer Cluster müssen Sie sich nach der Einrichtung nicht mehr per SSH auf Ihren Server begeben, auch die Werkzeuge auf Ihrer lokalen Maschine nutzen dieses API. Und anders als der Docker-Socket ist das Kubernetes-API mit Authentifizierung ausgestattet und darf veröffentlicht werden. Damit Sie von der lokalen Maschine aus mit dem Cluster arbeiten können, brauchen Sie darauf das offizielle Kubernetes-Kommandozeilenwerkzeug `kubectl`. Wer Docker Desktop unter Windows oder macOS nutzt und den lokalen Cluster aktiviert, hat Kubectl damit direkt installiert, unter macOS bekommt man es ansonsten schnell über den Paketmanager Homebrew:

```
brew install kubernetes-cli
```

Ubuntu-Nutzer finden es über Snap:

```
sudo snap install kubectl --classic
```

Für alle anderen Betriebssysteme (mit und ohne Paketmanager) verrät die Kubernetes-Doku, wie Sie das kleine Werkzeug herunterladen, in den Programmpfad verschieben und ausführbar machen (siehe ct.de/yepd).

Distributionskunde

Nach diesen Vorbereitungen kann die Installation des Clusters beginnen – fehlt nur noch Kubernetes selbst. Auf der offiziellen Seite kubernetes.io werden Sie aber vergeblich nach einem Download-Button fahnden. Mit Kubernetes verhält es sich wie mit dem Linux-Kernel: Den können Sie auch irgendwo aus einem schmucklosen Archiv herunterladen, werden damit aber zunächst wenig anstellen können. Wie auch Linux wollen Sie Kubernetes in Form einer Kubernetes-Distribution haben. Die bündelt all das, was zum Betrieb notwendig ist und verdrahtet die Komponenten schon mal sinnvoll. Etcd zum Beispiel wollen Sie nicht per Hand an ein nacktes Kubernetes-Binary anbinden. Kubernetes-Distributionen gibt es mittlerweile viele und ihre Wahl ist zur Wissenschaft geworden. Die meistgenutzten fallen für Selbstbetreiber schon mal raus, sie heißen GKE (Google Kubernetes Engine), AKS (Azure Kubernetes Service) und EKS (Amazon Elastic Kubernetes Service) und stecken in den Managed-Kubernetes-Angeboten der drei Branchenriesen im Cloudgeschäft.

> Die gut gepflegte Kubernetes-Dokumentation hat auf fast jede Frage eine Antwort. Für den Ein
>
> Die gut gepflegte Kubernetes-Dokumentation hat auf fast jede Frage eine Antwort. Für den Einstieg ist das aber zu umfangreich.

Was Sie suchen, ist Kubernetes für „Bare-Metal-Umgebungen“. So nennt man in der Szene Installationen auf eigenen Servern, virtualisiert oder physisch. Bei der Suche nach Bare-Metal-Kubernetes werden Sie früher oder später auf das Unternehmen Rancher stoßen. Das ehemalige Start-up gehört heute zum deutschen Linux-Distributor Suse, die Rancher-Produkte funktionieren aber unabhängig von Suses Linux-Distros. Ranchers Hauptprodukt, das auch schlicht Rancher heißt, können Sie sich direkt für später merken. Es handelt sich um eine Verwaltungsoberfläche, mit der Sie Kubernetes-Cluster im eigenen Rechenzentrum oder bei den oben genannten Cloudprovidern verwalten. Rancher selbst ist ein Docker-Container, den Sie auch auf der lokalen Maschine starten können und von da aus Cluster in aller Welt hochfahren und verwalten. Der Ansatz eignet sich für Firmen, die eine Multi-Cloud-Strategie planen, ist aber überdimensioniert für den Einstieg.

Für die ersten Gehversuche (und auch für kleine und mittelgroße Kubernetes-Cluster) empfehlen wir die Distribution k3s, die ursprünglich aus dem Hause Rancher stammt und heute von der CNCF verwaltet wird. Von dieser Organisation werden Sie auf dem Weg noch öfter hören: Die Cloud Native Computing Foundation ist eine Tochter der Linux Foundation, ihr gehört unter anderem der Open-Source-Code von Kubernetes. Außerdem verwaltet sie viele Projekte aus dem Ökosystem.

k3s ist mit dem Ziel angetreten, das Betreiben von Kubernetes-Clustern zu vereinfachen; ein paar Nischenfunktionen, die kaum jemand vermisst, sind daher rausgeflogen. Den ersten Cluster haben Sie mit k3s in wenigen Minuten einsatzbereit, weil die Distribution die Installation in ein komfortables Installationsskript verpackt hat. Öffnen Sie am besten je eine SSH-Sitzung auf Ihren drei Maschinen und platzieren Sie die Fenster für die nächsten Schritte schon einmal nebeneinander. Doch Achtung: Die nächsten Anweisungen müssen Sie zunächst nur auf einer der drei Maschinen ausführen.

Loslegen

Die offizielle k3s-Anleitung zur Installation besteht aus einem Einzeiler, von dem wir ohne weitere Vorbereitungen aber abraten, weil spätere Anpassungen dadurch schwieriger werden:

```
curl -sfL https://get.k3s.io | sh -
```

Die Zeile lädt ein Installationsskript herunter und führt es direkt aus. Wenn Sie wissen wollen, was da passiert, öffnen Sie die Adresse get.k3s.io im Browser. Das Skript lädt die k3s-Bestandteile nach und richtet einen Dienst für systemd oder OpenRC ein. Danach hat das Skript ausgedient und Kubernetes läuft rund um die Uhr im Hintergrund als Dienst mit dem Namen `k3s`, den Linux-Admins auch mit systemctl-Befehlen wie `systemctl status k3s` verwalten können.

Mit Umgebungsvariablen greifen Sie in den Einrichtungsprozess ein und legen Einstellungen fest, die dann in die Konfiguration des Dienstes gelangen. Wenn man daran nach der Installation etwas ändern will, muss man per Hand an der systemd-Konfiguration schrauben oder das Installationsskript erneut mit neuen Einstellungen drüberlaufen lassen. Sinnvoller ist es, direkt von Anfang den tiefer in der Doku versteckten Weg zu gehen und die k3s-Konfiguration in eine YAML-Datei namens /etc/rancher/k3s/config.yaml zu schreiben. Hat man später etwas daran geändert, reicht `systemctl restart k3s`, damit die Änderungen übernommen werden. Erzeugen Sie also auf dem ersten Ihrer drei Server das Verzeichnis für diese Datei:

```
mkdir -p /etc/rancher/k3s/
```

Legen Sie darin (zum Beispiel mit dem Texteditor Nano) die YAML-Datei config.yaml an:

```
nano /etc/rancher/k3s/config.yaml
```

Fürs Erste reicht darin eine Zeile:

```
disable: traefik
```

Die verhindert, dass k3s den HTTP-Proxy Traefik direkt startet, nicht weil Traefik ein schlechter HTTP-Proxy wäre, sondern damit Sie später selbst lernen, Traefik manuell zu installieren. Auf dem ersten Server ist damit alles bereit für die Installation von k3s:

```
curl -sfL https://get.k3s.io | sh -s - server --cluster-init
```

Die Leerzeichen in diesem Befehl sehen vielleicht falsch aus, haben aber ihre Richtigkeit. Das Installationsskript wird an `sh` übergeben und mit dem Befehl `server --cluster-init` ausgeführt. Der letzte Parameter führt dazu, dass k3s eine frische etcd-Instanz einrichtet. Nach einer Minute ist die Einrichtung erledigt und Ihr Cluster läuft. Glauben Sie nicht? Ihr erster Befehl mit `kubectl` auf dem Server selbst beweist es:

```
kubectl get nodes
```

Wenn Kubectl einen Zertifikatsfehler präsentiert, geben Sie der Installation noch ein bisschen Zeit. Sobald „Ready“ in der Tabelle erscheint, ist der Single-Node-Cluster hochgefahren. Sollte der Befehl mit einer Fehlermeldung scheitern, sind Sie nicht Root auf dem Server – k3s schränkt die Rechte auf die Konfigurationsdatei stark ein (was man mit der Zeile `write-kubeconfig-mode: "064"` in der Konfiguration ändern könnte). Mit einem vorangestellten `sudo` können Sie zugreifen. Wie schon erwähnt: Im Alltag greifen Sie selten über eine SSH-Sitzung vom Server selbst zu, sondern per `kubectl` vom heimischen Arbeitsplatz.

Bevor Sie kubectl lokal einrichten, soll der Cluster aber um zwei weitere Mitglieder erweitert werden. Erzeugen Sie die oben angelegte Konfigurationsdatei auf den anderen beiden Maschinen. Damit auch die anderen Server als Master in den Cluster aufgenommen werden dürfen, brauchen Sie ein Token, das das k3s auf der ersten Maschine automatisch angelegt und in eine Datei geschrieben hat. Sie finden es mit folgendem Befehl:

```
cat /var/lib/rancher/k3s/server/token
```

Kopieren Sie die komplette zurückgegebene Zeichenkette in die Zwischenablage. Nun brauchen Sie nur noch die IP-Adresse des ersten Servers, der schon ein Cluster eröffnet hat. Im besten Fall können sich die drei Clustermitglieder über eine interne IP-Adresse erreichen, zur Not klappt es für die Experimentierumgebung auch mit den externen Adressen (für ein produktives System müssen Sie da später noch mal nachbessern). Fügen Sie Token und IP-Adresse in den folgenden Befehl ein und setzen Sie diesen auf den Servern zwei und drei ab:

```
curl -sfL https://get.k3s.io | K3S_TOKEN=<Token> sh -s - server --server https://<IP Server 1>:6
```

Der Befehl setzt voraus, dass sich die Server untereinander über Port 6443 erreichen, außerdem braucht etcd die TCP-Ports 2379 und 2380. Die Ports müssten Sie in Ihren Firewalls öffnen – gute Gründe für ein internes Netz. Am Ende der Zeremonie sollte `kubectl get nodes` (auf einem der Server abgesetzt) drei gesunde Master-Nodes anzeigen.

Anschließend können Sie die Fernsteuerung auf Ihrer lokalen Maschine einrichten. Dafür müssen Sie insgesamt vier Werte hinterlegen: die externe Adresse eines Ihrer Kubernetes-Master, dessen TLS-Zertifikatsinformationen sowie den öffentlichen und den privaten Schlüssel für den Benutzeraccount im Cluster. Die externe IP-Adresse kennen Sie, die anderen Informationen liegen auf allen drei Master-Servern. Mit folgendem Befehl bekommen Sie diese zu sehen:

```
cat /etc/rancher/k3s/k3s.yaml
```

Der Inhalt der Datei sieht auf den ersten Blick kompliziert aus und ist es aus gutem Grund auch: Kubectl ist dafür konzipiert, mit mehreren Clustern zu arbeiten – die meisten Nutzer haben mindestens ein Produktiv- und ein Entwicklungscluster oder gar Cluster bei mehreren Kunden, daher kann man zwischen Kontexten wechseln (und muss bei schreibenden Befehlen immer sicherstellen, dass man im richtigen Kontext unterwegs ist). Ein Kontext ist immer eine Kombination aus einem Cluster und einem Benutzer mit seinen Zugangsdaten. Verwaltet werden diese entweder alle in einer YAML-Datei oder in mehreren Dateien, wir stellen hier die Strategie mit einer Datei vor, andere Herangehensweisen beschreibt die Doku (zu finden über ct.de/yepd).

Sofern Sie kubectl über Docker Desktop bekommen haben, liegt in Ihrem Benutzerverzeichnis bereits der Ordner .kube, unter Linux und macOS also unter ~/.kube, unter Windows in %USERPROFILE%\.kube. Weil der Ordnername mit einem Punkt beginnt, blenden ihn viele grafische Dateiexplorer aus, über die Kommandozeile finden Sie ihn aber. Gibt es den Ordner noch nicht, weil Sie kubectl per Hand installiert haben, legen Sie ihn zunächst an.

> Kubernetes kann man nicht nur auf der Kommandozeile verwalten. Die Desktop-Anwendung L
>
> Kubernetes kann man nicht nur auf der Kommandozeile verwalten. Die Desktop-Anwendung Lens verbindet sich mit dem Cluster und stellt seine Details grafisch dar.

Kubectl erwartet in diesem Ordner eine Datei namens config (ohne Endung). Gibt es sie noch nicht, legen Sie sie an und kopieren den kompletten Inhalt der Datei k3s.yaml vom Server hinein. Ändern

müssen Sie dann nur die IP-Adresse 127.0.0.1:6443 durch die externe IP-Adresse eines Servers (oder durch einen DNS-Namen) mit dem Port 6443 am Ende. Geben Sie dem Kontext in Zeile 11 noch einen sprechenderen Namen als `default` geben – zum Beispiel `dev-k3s`. Anschließend sind Sie einsatzbereit.

Gibt es die Datei bereits, hat Docker sie angelegt und mit den Werten für die lokale Umgebung befüllt. Dann müssen Sie die Abschnitte vom Server einzeln in die Datei kopieren (am besten mit einem grafischen Texteditor). Zunächst die Clusterinformationen (mit angepasster IP-Adresse). Aus dem Clusternamen `default` machen Sie einen sprechenden Namen wie `dev-k3s`. Dann den Abschnitt für den User, dessen Namen Sie ebenfalls von `default` in `dev-k3s` ändern sollten. Als dritten Schritt fügen Sie einen Block unter `contexts` hinzu und kombinieren nach dem schon angelegten Schema das Cluster `dev-k3s` mit dem gleichnamigen Benutzer zu einem Kontext mit ebendiesem Namen. Wenn Sie im YAML-Salat den Überblick verloren haben, finden Sie ein Beispiel (ohne gültige Zugangsdaten) über ct.de/yepd.

Weisen Sie kubectl jetzt an, den konfigurierten Kontext zu nutzen:

```
kubectl config use-context dev-k3s
```

Der schon bekannte Befehl `kubectl get nodes` sollte jetzt auch aus der Ferne die drei gesunden Nodes anzeigen. Wenn nicht, könnte eine Firewall Port 6443 auf Ihrem Node blockieren. Sollten Sie Fehler beim Zusammenbau der YAML-Datei gemacht haben, beschwert sich der YAML-Parser von kubectl mit einer hilfreichen Fehlermeldung.

Für Docker-Desktop-Nutzer gibt es noch einen zweiten Weg, den Kontext zu wechseln: Mit einem Klick auf das Wal-Logo in der Taskleiste öffnen sie ein Menü, das den Punkt Kubernetes enthält. Dahinter verbergen sich alle erkannten Kontexte für den schnellen Wechsel per Mausklick. Und noch eine Abkürzung ist dringend empfohlen: Während Ihrer nächsten Kubernetes-Lernschritte werden Sie `kubectl` oft eingeben. Kubernetes-Intensivnutzer sparen sich viel Tipperei, wenn sie sich einen Alias wie `k` dafür anlegen. `kubectl get nodes` legen Kubernetes-Profis gern auf den Alias `kgn`.

Neben kubectl raten wir Ihnen zu einem weiteren Werkzeug, mit dem Sie auf Ihre Cluster zugreifen können: Lens (k8slens.dev) ist eine grafische Oberfläche, die als Anwendung auf Ihrer Maschine läuft und die Kontexte aus der kubectl-Konfigurationsdatei übernimmt. Die Software ist kostenlos, schnell eingerichtet und läuft unter Windows, Linux und macOS. Welche Details Lens zeigen kann, sehen Sie im Bild oben.

Aufbauen und abreißen

Der folgende Tipp mag etwas befremdlich wirken, zahlt sich aber später aus: Wenn Ihr Cluster läuft, sollten Sie ihn möglichst bald wieder abreißen und den Bau, soweit es geht, automatisieren – das ist eine Herangehensweise, an die man sich im Cloud-Native-Umfeld früh gewöhnen sollte. Erzeugen Sie Ihre Infrastruktur immer reproduzierbar und schaffen Sie handgeknüpfte Strukturen ab. Zum restlosen Entfernen von k3s gibt es den Befehl

```
/usr/local/bin/k3s-uninstall.sh
```

Für eine reproduzierbare Installation verpacken Sie alle Schritte im einfachsten Fall in Bash-Skripte; wenn Sie mit Werkzeugen wie Ansible und Terraform vertraut sind, nutzen Sie diese für Einrichtung von Servern, Firewalls, DNS-Einträgen und schließlich k3s.

> Nicht nur Einsteiger finden in der grafischen Oberfläche Lens wichtige Informationen, die auf d
>
> Nicht nur Einsteiger finden in der grafischen Oberfläche Lens wichtige Informationen, die auf der Kommandozeile schnell untergehen.

Zum Schluss

Glückwunsch, Sie sind jetzt Betreiber eines selbst gebauten Clusters mit echter Redundanz dank verteilter Master-Rolle. Fahren Sie zum Test mal einen der Server herunter und testen, ob `kubectl get nodes` auf den anderen beiden noch funktioniert. Im Cluster läuft aber (abgesehen von ein paar System-Containern) noch nichts. Einen ersten Container, der eine einfache Website auf Port 30.000 veröffentlicht, haben Sie schnell in Betrieb: Über ct.de/yepd finden Sie eine YAML-Datei namens first-pod.yml zum Download und fürs Selbststudium. Laden Sie diese auf Ihre lokale Maschine mit installiertem kubectl, navigieren Sie auf der Kommandozeile in den Ordner mit der Datei und installieren die Zusammenstellung im Cluster:

```
kubectl -f first-pod.yml apply
```

Wenig später sollten alle drei Maschinen auf ihren externen IP-Adressen auf Port 30.000 mit einer Website antworten. In einer der nächsten c’t-Ausgaben erfahren Sie im nächsten Teil dieser Reihe, warum Container in Kubernetes in sogenannten Pods stecken, wie Sie solche erzeugen und mit der Außenwelt verdrahten. *(jam@ct.de)*

1. Literatur

2. Jan Mahn, Zu neuen Ufern, Nach dem Hype: Docker verstehen und loslegen, c’t 24/2021, S. 146
3. Holger Bleich und Jan Mahn, Wolkenangebotsvielfalt, Sechs europäische Cloudprovider im Überblick, c’t 21/2021, S. 62
4. Jan Mahn, Gemeinsame Wahrheit, Wie verteilte Systeme dank Raft-Algorithmus zusammenarbeiten, c’t 21/2022, S. 146

Dokumentation und Beispiele:ct.de/yepd

Revision #1
Created 2026-09-26 13:20:53 UTC by ralf

Updated 2026-09-26 13:20:53 UTC by ralf