
3 Ein Kubernetes-Cluster deployen

Nachdem Sie nun erfolgreich einen Anwendungs-Container gebaut haben, wollen Sie doch bestimmt lernen, wie Sie ihn in ein vollständig zuverlässiges, skalierbares verteiltes System deployen können. Natürlich benötigen Sie dafür ein laufendes Kubernetes-Cluster. Aktuell gibt es eine Reihe von Cloud-basierten Kubernetes-Services, die es einfach machen, mit ein paar Befehlen an der Kommandozeile ein Cluster zu erzeugen. Wir empfehlen Ihnen dieses Vorgehen, wenn Sie gerade mit Kubernetes beginnen. Auch wenn Sie letztendlich planen, Kubernetes direkt auf Rechnern laufen zu lassen (*bare metal*), ist der Weg über eine Cloud sinnvoll, um schnell Ergebnisse zu sehen und etwas über Kubernetes selbst zu lernen. Danach können Sie dann herausfinden, wie Sie es direkt auf Rechnern installieren.

Natürlich müssen Sie für eine Cloud-basierte Lösung die Ressourcen bezahlen, zudem benötigen Sie eine aktive Netzwerkverbindung zur Cloud. Aus diesem Grund kann eine lokale Entwicklung attraktiver sein und da bietet das Tool `minikube` einen einfachen Weg, ein lokales Kubernetes-Cluster in einer VM auf Ihrem lokalen Laptop oder Desktop-PC laufen zu lassen. Das klingt zwar nett, allerdings erzeugt `minikube` nur ein Cluster aus einem Knoten, mit dem sich nicht alle Aspekte eines vollständigen Kubernetes-Clusters demonstrieren lassen. Aus diesem Grund empfehlen wir, mit einer Cloud-basierten Lösung zu starten, sofern das für Ihre Situation möglich ist. Wenn Sie tatsächlich darauf bestehen, direkt auf echten Rechnern loszulegen, finden Sie in Anhang A am Ende dieses Buches Anweisungen für den Aufbau eines Clusters auf einem Satz Raspberry-Pi-Computern. Dabei wird das Tool `kubeadm` genutzt, das sich auch für andere Rechner als Raspberry Pis einsetzen lässt.

3.1 Kubernetes auf einem öffentlichen Cloud-Provider installieren

Dieses Kapitel behandelt das Installieren von Kubernetes auf den drei großen Cloud-Providern – Amazon Web Services (AWS), Microsoft Azure und die Google Cloud Platform.

3.1.1 Google Container Service

Die Google Cloud Platform bietet einen gehosteten Kubernetes-as-a-Service namens Google Container Engine (GKE) an. Um diesen zu nutzen, benötigen Sie einen Account für die Google Cloud Platform, über den bezahlt werden kann. Zudem muss das `gcloud`-Tool (<https://cloud.google.com/sdk/downloads>) installiert sein.

Sind diese Voraussetzungen erfüllt, setzen Sie zunächst eine Default-Zone:

```
$ gcloud config set compute/zone europe-west3-a
```

Dann können Sie ein Cluster erstellen:

```
$ gcloud container clusters create kuar-cluster
```

Das dauert ein paar Minuten. Ist das Cluster fertig, können Sie sich die Credentials dazu holen:

```
$ gcloud auth application-default login
```

Jetzt sollten Sie ein fertig konfiguriertes Cluster haben, mit dem Sie loslegen können. Sofern Sie es nicht vorziehen, Kubernetes woanders zu installieren, können Sie zum Abschnitt 3.4 springen.

Haben Sie Probleme, finden Sie die vollständigen Anweisungen zum Erstellen eines GKE-Clusters in der Dokumentation zur Google Cloud Platform unter <http://bit.ly/2ver7Po>.

3.1.2 Kubernetes auf dem Azure Container Service installieren

Microsoft Azure bietet einen gehosteten Kubernetes-as-a-Service als Teil der Azure Container Services. Am einfachsten nutzen Sie dazu die im Azure-Portal eingebaute Azure Cloud Shell. Diese erreichen Sie über einen Klick auf das Shell-Symbol  in der oberen rechten Toolbar. In der Shell ist das `az`-Tool schon installiert und konfiguriert.

Alternativ können Sie das CLI-Tool `az` auf Ihrem lokalen Rechner installieren (<https://github.com/Azure/azure-cli>).

Läuft die Shell, können Sie folgenden Befehl eingeben:

```
$ az group create --name=kuar --location=westus
```

Wurde die Ressourcen-Gruppe erzeugt, können Sie ein Cluster anlegen:

```
$ az acs create --orchestrator-type=kubernetes \  
--resource-group=kuar --name=kuar-cluster
```

Das dauert ein paar Minuten. Ist das Cluster erzeugt worden, können Sie sich die Credentials über diesen Befehl holen:

```
$ az aks kubernetes get-credentials --resource-group=kuar \  
--name=kuar-cluster
```

Haben Sie das kubectl-Tool noch nicht installiert, können Sie es wie folgt erreichen:

```
$ az aks kubernetes install-cli
```

Die vollständige Anleitung für das Installieren von Kubernetes auf Azure finden Sie in der Dokumentation unter <http://bit.ly/2veqXYl>.

3.1.3 Kubernetes auf den Amazon Web Services installieren

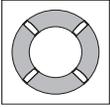
AWS bietet erst in Q1 2018 mit EKS einen gehosteten Kubernetes-Service an. Die Landschaft zum Managen von Kubernetes auf AWS verändert sich zurzeit schnell und es werden häufig neue und verbesserte Tools eingeführt. Hier ein paar Optionen für einen leichten Start:

- Am einfachsten starten Sie ein kleines Cluster für das Experimentieren mit Kubernetes im Rahmen dieses Buches über den »Quick Start for Kubernetes« von Heptio (<http://amzn.to/2veAy1q>). Dabei handelt es sich um ein einfaches CloudFormation-Template, das ein Cluster über die AWS-Konsole starten kann.
- Eine umfangreichere Management-Lösung bietet ein Projekt namens kops. Ein Tutorial für das Installieren von Kubernetes auf AWS über kops finden Sie auf GitHub unter <http://bit.ly/2q86l2n>.

3.2 Kubernetes mit minikube lokal installieren

Möchten Sie auch lokale Erfahrungen sammeln oder wollen Sie nicht für Cloud-Ressourcen bezahlen, können Sie ein einfaches Cluster mit einem Knoten über minikube installieren. Dies ist zwar eine gute Simulation eines Kubernetes-Clusters, aber eigentlich nur für die lokale Entwicklung sowie zum Lernen und Experimentieren gedacht. Weil es nur in einer VM auf einem einzelnen Knoten läuft, bietet es nicht die Zuverlässigkeit eines verteilten Kubernetes-Clusters.

Zudem ist für manche in diesem Buch beschriebenen Features die Integration mit einem Cloud-Provider notwendig. Diese Features stehen bei minikube entweder nicht zur Verfügung oder sie funktionieren nur sehr eingeschränkt.

**Tipp**

Um minikube nutzen zu können, muss auf Ihrem Rechner ein Hypervisor installiert sein. Für Linux und macOS ist das im Allgemeinen virtualbox (<https://virtualbox.org>). Unter Windows ist der Hypervisor Hyper-V die Standardoption. Stellen Sie sicher, dass Sie den Hypervisor installiert haben, bevor Sie minikube einsetzen.

Sie finden minikube auf GitHub unter <https://github.com/kubernetes/minikube>. Es gibt Binaries für Linux, macOS und Windows, die Sie herunterladen können. Ist das Tool installiert, können Sie ein lokales Cluster wie folgt erzeugen:

```
$ minikube start
```

Damit wird eine lokale VM erstellt, mit Kubernetes provisioniert und eine lokale kubectl-Konfiguration erzeugt, die auf dieses Cluster zeigt.

Benötigen Sie das Cluster nicht mehr, können Sie die VM stoppen:

```
$ minikube stop
```

Wollen Sie das Cluster ganz entfernen, nutzen Sie:

```
$ minikube delete
```

3.3 Kubernetes auf dem Raspberry Pi ausführen

Wollen Sie mit einem realistischen Kubernetes-Cluster experimentieren, aber nicht so viel Geld ausgeben, können Sie ein sehr nettes Kubernetes-Cluster mithilfe von Raspberry-Pi-Computern aufbauen, ohne sich allzu sehr in Unkosten stürzen zu müssen. Die Details eines solchen Aufbaus gehen über den Rahmen dieses Kapitels hinaus, aber Sie finden sie in Anhang A am Ende dieses Buches.

3.4 Der Kubernetes-Client

Der offizielle Kubernetes-Client ist kubectl: ein Befehlszeilentool für die Interaktion mit der Kubernetes-API. Mit kubectl können Sie Kubernetes-Objekte wie zum Beispiel Pods, ReplicaSets und Services managen. kubectl dient auch dazu, den Status des gesamten Clusters zu untersuchen und zu überprüfen.

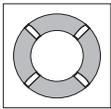
Wir werden das kubectl-Tool verwenden, um das gerade erstellte Cluster zu erforschen.

3.4.1 Den Cluster-Status prüfen

Als Erstes können Sie die Version des Clusters prüfen, das bei Ihnen läuft:

```
$ kubectl version
```

Dabei werden zwei verschiedene Versionen angezeigt – die des lokalen `kubectl`-Tools und die Version des Kubernetes-API-Servers.



Tipp

Machen Sie sich keine Sorgen, wenn die Versionen verschieden sind. Die Kubernetes-Tools sind rückwärts- und vorwärtskompatibel zu den verschiedenen Versionen der Kubernetes-API, solange Sie im Rahmen von zwei Minor-Versionen der Tools und des Clusters bleiben und nicht versuchen, neuere Features auf ein älteres Cluster anzuwenden. Kubernetes folgt den semantischen Versionierungsspezifikationen, und diese Minor-Version ist die mittlere Zahl (zum Beispiel die 5 in 1.5.2).

Nachdem wir nun dafür gesorgt haben, dass Sie mit Ihrem Kubernetes-Cluster kommunizieren können, wollen wir es uns genauer anschauen.

Als Erstes rufen wir eine einfache Diagnose ab. So können Sie schnell prüfen, ob Ihr Cluster prinzipiell in Ordnung ist:

```
$ kubectl get componentstatuses
```

Die Ausgabe sollte in etwa wie folgt aussehen:

NAME	STATUS	MESSAGE	ERROR
scheduler	Healthy	ok	
controller-manager	Healthy	ok	
etcd-0	Healthy	{"health": "true"}	

Sie sehen hier die Komponenten, aus denen das Kubernetes-Cluster besteht. Der `controller-manager` ist dafür verantwortlich, die diversen Controller laufen zu lassen, die das Verhalten im Cluster steuern – zum Beispiel wird so sichergestellt, dass alle Kopien (Replicas) eines Service verfügbar sind und laufen. Der `scheduler` kümmert sich darum, die verschiedenen Pods auf die Knoten im Cluster zu verteilen. Und schließlich ist der `etcd`-Server für das Storage des Clusters zuständig, auf dem alle API-Objekte abgelegt werden.

3.4.2 Worker-Knoten in Kubernetes auflisten

Als Nächstes können wir alle Knoten in unserem Cluster ausgeben lassen:

```
$ kubectl get nodes
```

NAME	STATUS	AGE
kubernetes	Ready,master	45d
node-1	Ready	45d
node-2	Ready	45d
node-3	Ready	45d

Sie sehen, dass es sich hier um ein Cluster aus vier Knoten handelt, das seit 45 Tagen läuft. In Kubernetes werden die Knoten unterteilt in `master`-Knoten, die

Container wie den API-Server, Scheduler und so weiter enthalten und das Cluster managen, und in worker-Knoten, auf denen Ihre Container laufen werden. Kubernetes teilt master-Knoten normalerweise keine Arbeiten zu, damit die Workload der Anwender nicht die Funktionsfähigkeit des Clusters beeinträchtigt.

Sie können den Befehl `kubectl describe` verwenden, um mehr Informationen über einen bestimmten Knoten (wie zum Beispiel `node-1`) zu erhalten:

```
$ kubectl describe nodes node-1
```

Als Erstes werden Basisinformationen über den Knoten ausgegeben:

```
Name:                node-1
Role:
Labels:              beta.kubernetes.io/arch=arm
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=node-1
```

Sie sehen hier, dass dieser Knoten unter Linux auf einem ARM-Prozessor läuft.

Als Nächstes kommen Informationen über den Status von `node-1` selbst:

```
Conditions:
  Type           Status LastHeartbeatTime Reason Message
  ----           -
  OutOfDisk      False Sun, 05 Feb 2017... KubeletHasSufficientDisk kubelet...
  MemoryPressure False Sun, 05 Feb 2017... KubeletHasSufficientMemory kubelet...
  DiskPressure   False Sun, 05 Feb 2017... KubeletHasNoDiskPressure kubelet...
  Ready          True  Sun, 05 Feb 2017... KubeletReady kubelet...
```

Man sieht hier, dass der Knoten ausreichend Plattenplatz und Speicher besitzt und an den Kubernetes-Master seinen positiven Gesundheitszustand berichtet.

Als Nächstes kommen nun Informationen über die Kapazität der Maschine:

```
Capacity:
  alpha.kubernetes.io/nvidia-gpu: 0
  cpu:                             4
  memory:                          882636Ki
  pods:                             110
Allocatable:
  alpha.kubernetes.io/nvidia-gpu: 0
  cpu:                             4
  memory:                          882636Ki
  pods:                             110
```

Jetzt folgen Informationen über die Software auf dem Knoten, einschließlich der Version von Docker, Kubernetes und dem Linux-Kernel, und manches mehr:

```
System Info:
  Machine ID:          9989a26f06984d6dbadc01770f018e3b
  System UUID:        9989a26f06984d6dbadc01770f018e3b
  Boot ID:            98339c67-7924-446c-92aa-c1bfe5d213e6
  Kernel Version:    4.4.39-hypriotos-v7+
  OS Image:          Raspbian GNU/Linux 8 (jessie)
```

```

Operating System:      linux
Architecture:         arm
Container Runtime Version: docker://1.12.6
Kubelet Version:      v1.5.2
Kube-Proxy Version:   v1.5.2
PodCIDR:              10.244.2.0/24
ExternalID:           node-1

```

Und schließlich kommen noch Informationen über die Pods, die aktuell auf diesem Knoten laufen:

```

Non-terminated Pods:          (3 in total)
Namespace   Name      CPU Requests  CPU Limits  Memory Requests  Memory Limits
-----
kube-system kube-dns... 260m (6%)    0 (0%)     140Mi (16%)    220Mi (25%)
kube-system kube-fla... 0 (0%)       0 (0%)     0 (0%)         0 (0%)
kube-system kube-pro... 0 (0%)       0 (0%)     0 (0%)         0 (0%)
Allocated resources:
(Total limits may be over 100 percent, i.e., overcommitted.)
CPU Requests  CPU Limits  Memory Requests  Memory Limits
-----
260m (6%)    0 (0%)     140Mi (16%)    220Mi (25%)
No events.

```

Anhand dieser Ausgabe sehen Sie die Pods auf dem Knoten (zum Beispiel den Pod `kube-dns`, der die DNS-Services für das Cluster bereitstellt), CPU und Speicher, die jeder Pod vom Knoten anfordert, und die Gesamtressourcen, die angefordert wurden. Es sei hier darauf hingewiesen, dass Kubernetes bei jedem Pod auf der Maschine sowohl die *angeforderten Ressourcen* als auch die *Obergrenze* dokumentiert. Dieser Unterschied zwischen Anforderung und Obergrenze ist genauer in Kapitel 5 beschrieben, aber kurz ausgedrückt sind Ressourcen, die von einem Pod *angefordert* werden, auf diesem Knoten garantiert, während die Obergrenze die maximale Menge einer Ressource ist, die ein Pod nutzen kann. Diese Grenze kann höher als die angeforderte Menge sein – dann werden die zusätzlichen Ressourcen auf Best-Effort-Basis zugeteilt. Es ist aber nicht garantiert, dass sie auf dem Knoten vorhanden sind.

3.5 Cluster-Komponenten

Einer der interessantesten Aspekte von Kubernetes ist, dass viele der Komponenten, aus denen das Cluster besteht, durch Kubernetes selbst deployt werden. Wir werden uns ein paar von ihnen anschauen. Diese Komponenten nutzen eine Reihe von Konzepten, die wir in späteren Kapiteln einführen werden. Alle diese Komponenten laufen im Namensraum `kube-system`.²

2. Wie Sie im nächsten Kapitel lernen werden, ist ein Namensraum in Kubernetes eine Entität für das Organisieren von Kubernetes-Ressourcen. Stellen Sie sich ihn wie einen Ordner in einem Dateisystem vor.

3.5.1 Kubernetes-Proxy

Der Kubernetes-Proxy ist dafür verantwortlich, Netzwerkverkehr auf durch Load Balancing bedienten Services im Kubernetes-Cluster zu routen. Dazu muss er auf jedem Knoten im Cluster vorhanden sein. Kubernetes besitzt ein API-Objekt namens `DaemonSet`, das Sie später noch kennenlernen werden und das in vielen Clustern verwendet wird, um dies umzusetzen. Nutzt Ihr Cluster den Kubernetes-Proxy mit einem `DaemonSet`, können Sie die Proxys mit folgendem Befehl anzeigen lassen:

```
$ kubectl get daemonSets --namespace=kube-system kube-proxy
NAME          DESIRED  CURRENT  READY  NODE-SELECTOR  AGE
kube-proxy    4         4        4      <none>         45d
```

3.5.2 Kubernetes-DNS

Kubernetes besitzt auch einen DNS-Server, der Naming und Discovery für die Services bereitstellt, die im Cluster definiert sind. Dieser DNS-Server läuft ebenfalls als replizierter Service im Cluster. Abhängig von der Größe Ihres Clusters laufen in Ihrem Cluster ein oder mehrere DNS-Server. Der DNS-Service wird als `Kubernetes-Deployment` ausgeführt, das diese Replicas verwaltet:

```
$ kubectl get deployments --namespace=kube-system kube-dns
NAME          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
kube-dns      1         1        1            1          45d
```

Es gibt auch einen `Kubernetes-Service`, der das Load Balancing für den DNS-Server bereitstellt:

```
$ kubectl get services --namespace=kube-system kube-dns
NAME          CLUSTER-IP  EXTERNAL-IP  PORT(S)          AGE
kube-dns      10.96.0.10  <none>       53/UDP,53/TCP   45d
```

Sie sehen hier, dass der DNS-Service für das Cluster die Adresse 10.96.0.10 besitzt. Melden Sie sich an einem Container im Cluster an, werden Sie sehen, dass dies dort in der Datei `/etc/resolv.conf` eingetragen wurde.

3.5.3 Kubernetes-UI

Die letzte Kubernetes-Komponente ist ein GUI. Das UI läuft als einzelne Replica, wird aber trotzdem aus Gründen der Zuverlässigkeit und für Updates durch ein `Kubernetes-Deployment` gemanagt. Den UI-Server sehen Sie über:

```
$ kubectl get deployments --namespace=kube-system kubernetes-dashboard
NAME          DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
Kubernetes-dashboard  1         1        1            1          45d
```

Auch das Dashboard besitzt einen Service, der sich um das Load Balancing kümmert:

```
$ kubectl get services --namespace=kube-system kubernetes-dashboard
NAME                CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
kubernetes-dashboard 10.99.104.174   <none>       80:32551/TCP    45d
```

Wir können den `kubectl-proxy` nutzen, um auf dieses UI zuzugreifen. Starten Sie den Proxy über:

```
$ kubectl proxy
```

Damit wird ein Server auf `localhost:8001` gestartet. Rufen Sie nun `http://localhost:8001/ui` in Ihrem Webbrowser auf, sollten Sie das Web-UI von Kubernetes sehen. Über diese Schnittstelle können Sie Ihr Cluster erforschen, aber auch neue Container anlegen. Es würde den Rahmen dieses Buches sprengen, auf alle Details dieser Oberfläche einzugehen, außerdem ändert sie sich noch sehr häufig.

3.6 Zusammenfassung

Hoffentlich läuft bei Ihnen nun ein Kubernetes-Cluster (oder mehrere) und Sie haben ein paar Befehle genutzt, um sich mit dem Cluster vertraut zu machen. Als Nächstes werden wir uns detaillierter mit der Befehlszeilen-Schnittstelle für dieses Kubernetes-Cluster beschäftigen und Ihnen zeigen, wie Sie mit dem `kubectl`-Tool umgehen. Im weiteren Verlauf werden Sie `kubectl` und Ihr Test-Cluster nutzen, um die diversen Objekte der Kubernetes-API kennenzulernen.