Cloud Computing — Praktikum 2
Thema: Einführung in Minikube — Sommersemester 2025
Prof. Dr. Christoph Karg — Hochschule Aalen

Dieses Praktikum ist eine kurze Einführung in Minikube, einer Kubernetes Distribution für Desktop Computer und Laptops. Die Webpage zu Minikube findet man über den folgenden Link. Die Einführung basiert auf dem Minikube Kubernetes 101 Tutorial.

1 Installation

Die Systemanforderungen für den Betrieb von Minikube sind niedrig. Die Mindestanforderungen sind ein Prozessor mit zwei Kernen, 2 GB Speicher und 20 GB freier Platz auf der Festplatte. Zur Installation ist eine Internet-Verbindung erforderlich. Ferner muss eine Container-Software wie Docker, QEmu oder Podman auf dem Rechner installiert sein. Eine Anleitung zur Installation findet man auf der Minikube Webpage (Link). Bei gängigen Linuxdistibutionen kann Minikube direkt über den Paketmanager installiert werden. Unter MacOS installiert man Minikube am einfachsten mit Homebrew.

Ein wichtiges Werkzeug zur Arbeit mit Kubernetes im Terminal ist Kubectl-Software. Das Werkzeug ist mit Minikube-Distribution enthalten. Der entsprechende Befehl lautet:

```
> minikube kubectl
```

Alternativ kann Kubectl auch separat installiert werden. Informationen hierzu findet man hier. Kubectl liest die Konfiguration zum Zugriff auf das Kubernetes-Cluster aus einer Konfigurationsdatei. Die Default-Datei ist `$HOME/.kube/config$`. Mit der Option `--kubeconfig` kann eine alternative Konfigurationsdatei genutzt werden. Alternativ kann die Konfigurationsdatei über die Umgebungsvariable `KUBECONFIG` festgelegt werden. Eine Übersicht über die wichtigsten Befehle von Kubectl findet man hier.

Ein lokaler Minikube-Cluster wird mit folgendem Befehl gestartet:

```
> minikube start
```

Falls noch nicht geschehen, werden die erforderlichen Komponenten aus dem Internet geladen und installiert. Die Container-Plattform wird automatisch ausgewählt. Um Docker oder Podman zu nutzen, muss diese Software vor dem Start von Minikube gestartet werden. Andernfalls wird in der Regel QEmu verwendet.

Benötigt man eine spezielle Version von Kubernetes, dann kann diese beim Start von Minikube über eine Option festgelegt werden. Beispiel:

```
> minikube start --kubernetes-version=v1.32.1
```

Der Befehl zum Beenden des Clusters lautet:

```
> minikube stop
```

Der Befehl zur Abfrage des Status des Clusters lautet:

```
> minikube status
```

Will man den Cluster löschen, dann nutzt man folgenden Befehl:

```
> minikube delete
```

Der Minikube-Befehl beinhaltet zahlreiche weitere Unterbefehle. Die komplette Liste kann man sich mit folgendem Befehl anzeigen lassen:

```
> minikube --help
```

Mit dem Befehl

```
> kubectl cluster-info
```

erhält man Informationen über das Cluster. Die Knoten des Clusters werden mit folgendem Befehl angezeigt:

```
> kubectl get nodes
```

Minikube stellt diverse Addons für Kubernetes bereit. Will man beispielsweise den Metrics-Server nutzen, dann aktiviert man diesen mit folgendem Befehl:

```
> minikube addons enable metrics-server
```

Das Dashboard-Addon stellt eine Webpage mit Informationen über den Cluster bereit. Befehl:

```
> minikube dashboard
```

Eine Liste mit allen zur Verfügung stehenden Addons erhält man mit dem Befehl:

```
> minikube addons list
```

2 Ausführung einer App

Um eine Anwendung in einem Kubernetes-Cluster auszuführen, muss diese bereitgestellt werden. Man nennt diesen Vorgang ein *Deployment*. Folgender Befehl rollt die eine einfache Beispielanwendung aus:

```
> kubectl create deployment kubernetes-bootcamp \
    --image=gcr.io/k8s-minikube/kubernetes-bootcamp:v1
```

Hierzu wird aus der Google-Cloud das entsprechende Image geladen und im Cluster gestartet. Das Ergebnis ist ein sogenannter Pod, in dem der Container ausgeführt wird. Mit folgendem Befehl werden alle Pods im Default-Namespace angezeigt:

```
> kubectl get pods
```

Will man alle Pods sehen, dann nutzt man diesen Befehl:

```
> kubectl get pods --all-namespaces
```

Der folgende Befehl zeigt die Pods im Namespace `kube-system`:

```
> kubectl get pods -n kube-system
```

Die im Default-Namespace enthaltenen Deployments liefert der folgende Befehl:

```
> kubectl get deployments
```

Detailliertere Informationen über ein Deployment werden über diesen Befehl angezeigt:

```
> kubectl describe deployment kubernetes-bootcamp
```

Eine einfache Möglichkeit, um auf Dienste des Clusters zuzugreifen, ist der Start eines Proxys, der eine Verbindung zum Arbeitsplatzrechner einrichtet. Der Befehl lautet:

```
> kubectl proxy
```

Die Software gibt nach dem Start den Netzwerk-Port aus, über der Zugriff erfolgen muss. Der Proxy wird mit dem Befehl `Ctrl-C` beendet.

Bei aktivem Proxy kann mittels eines Webbrowsers diverse Informationen abfragen. Der folgende Befehl liefert Informationen zur Version des Clusters:

```
> curl http://localhost:8001/version
```

3 Arbeiten mit Pods

In diesem Abschnitt wird dargestellt, wie man Informationen über im Cluster aktive Pods sammeln und ausgeben kann.

Der Einstieg erfolgt oft über diesen Befehl:

```
> kubectl get pods
```

Hier werden Informationen über die Pods des Default-Namespaces ausgegeben. Eine vollständige Liste aller Pods erhält man mit:

```
> kubectl get pods -A
```

Um eine Liste der Pods im Namespace `kube-system` zu erhalten, führt man diesen Befehl aus:

```
> kubectl get pods -n kube-system
```

Der folgende Befehl liefert eine detaillierte Beschreibung aller Pods im Default-Namespace.

```
> kubectl describe pods
```

Die Informationen zu einem konkreten Pod erhält man mit:

```
> kubectl describe pod kubernetes-bootcamp-5fb9694484-6mgx4
```

Hinweis: Der Name des Pods ist von System zu System verschieden. Bitte den korrekten Namen verwenden!

Ein Tipp für ein effizienteres Arbeiten mit Pods (und anderen Kubernetes-Objekten) ist, den Namen des Pods in einer Umgebungsvariable zu speichern:

```
> export POD_NAME=kubernetes-bootcamp-579cb4f987-6mgx4
```

Nun kann die Abfrage der Informationen anhand der Umgebungsvariable `POD_NAME` erfolgen:

```
> kubectl describe pod $POD_NAME
```

Die Umgebungsvariable kann auch für den Zugriff auf die API genutzt werden:

```
> curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME
```

Die Log-Daten eines Pods werden mit folgendem Befehl angezeigt:

```
> kubectl logs $POD_NAME
```

Will man sich die Umgebungsvariablen eines Pods anschauen, dann verwendet man diesen Befehl:

```
> kubectl exec $POD_NAME -- env
```

Der obige Befehl ist universell nutzbar. Hinter den beiden Strichen `--` kann ein beliebiger Linux-Befehl stehen. Eine interaktive Shell auf dem Pod startet man so:

```
> kubectl exec -ti $POD_NAME -- bash
```

Die Shell ermöglicht die Arbeit auf dem Pod. Beispielsweise kann man die auf dem Pod installierte Webanwendung testen. Befehl:

```
POD# curl localhost:8080
```

Auf die Webanwendung kann man übrigens noch nicht von außerhalb des Clusters zugreifen. Dieses Thema wird im nächsten Abschnitt behandelt.

4 Anwendung für Zugriff von außen freigegeben

Die einfachste Art, den Zugriff auf eine Anwendung freizugeben, ist die Konfiguration eines `NodePort`-Services. Dieser Service erstellt einen externen Port, der mit dem Port des Pods verbunden ist. Der entsprechende Befehl lautet:

```
> kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
```

Eine Übersicht über die aktiven Services des Default-Namespaces erhält man über diesen Befehl:

```
> kubectl get services
```

Detaillierte Information über den erstellten Service erhält man mit folgendem Befehl:

```
> kubectl describe services/kubernetes-bootcamp
```

Um auf den Service zuzugreifen, ist die externe IP-Adresse des Clusters erforderlich. Diese ermittelt man mit folgendem Befehl:

```
> minikube ip
```

Dieser Befehl kann in die Webanfrage eingebaut werden:

```
> curl $(minikube ip):32594
```

Der Port muss entsprechend der Konfiguration des Services angepasst werden.

Hinweis: Nutzt man Minikube mit der Docker-Engine auf einem Mac, dann ist der Zugriff auf einen Service in der oben beschriebenen Weise nicht möglich. Der Grund sind die eingeschränkten Netzwerkfähigkeiten des Docker-Systems. Minikube stellt jedoch eine Alternative bereit. Der folgende Befehl erlaubt den Zugriff auf den Service über SSH und öffnet die Webanwendung direkt im Browser.

```
> minikube service kubernetes-bootcamp
```

Ein wichtiges Hilfsmittel bei der Arbeit mit Kubernetes-Objekten sind Labels. Labels sind Paare bestehend aus einem Schlüssel und einem zugehörigen Wert. Labels kann man nutzen, um Eigenschaften von Kubernetes-Objekten zu kennzeichnen.

Um die Labels des Bootcamp-Deployments anzuzeigen, nutzt man folgenden Befehl:

```
> kubectl describe deployment kubernetes-bootcamp
```

Um die Labels der Pods im Default-Namespace anzuzeigen, gibt man folgenden Befehl ein:

```
> kubectl get pods --show-labels
```

Der folgende Befehl zeigt alle Pods an, die das angegebene Label besitzen:

```
> kubectl get pods -l app=kubernetes-bootcamp
```

Um einen Pod mit einem neuen Label zu versehen, nutzt man diesen Befehl:

```
> kubectl label pods $POD_NAME version=v1
```

Nun kann der Pod über das Label ausgewählt werden:

```
> kubectl get pods -l version=v1
```

Labels funktionieren auch mit Services. Beispielsweise zeigt der folgende Befehl den weiter oben erstellten Service an:

```
> kubectl get service -l app=kubernetes-bootcamp
```

Labels sind beispielsweise praktisch, wenn den zu einer App gehörenden Service löschen will. Befehl:

```
> kubectl delete service -l app=kubernetes-bootcamp
```

Die Webanwendung wird durch das Löschen des Services nicht beeinflusst. Check:

```
> kubectl exec -ti $POD_NAME -- curl localhost:8080
```

5 Skalierung von Apps

Eine wichtige Anforderung an eine Webanwendung ist deren Skalierbarkeit. In einem Kubernetes Cluster wird dies durch sogenannte ReplicaSets umgesetzt. Für jedes Deployment wird automatisch ein zugehöriger ReplicaSet angelegt.

```
> kubectl get rs
```

Mit folgendem Befehl werden vier Pods für das Bootcamp-Deployment erstellt:

```
> kubectl scale deployments/kubernetes-bootcamp --replicas=4
```

Die Anzahl der Pods wird auch bei der Deployment-Übersicht angezeigt. Befehl:

```
 > kubectl get deployments
```

Die Anzahl der Pods kann auch wieder verkleinert werden. Befehl:

```
> kubectl scale deployments/kubernetes-bootcamp --replicas=1
```

6 Update von Apps

Ein wichtiger Aspekt bei dem Betrieb einer Webanwendung ist das Einspielen von Updates. Bei Kubernetes wird dies zum Beispiel durch Bereitstellen eines aktualisierten Images umgesetzt. Der folgende Befehl aktualisiert das Images des Bootcamp-Deployments auf Version 2:

```
> kubectl set image deployments/kubernetes-bootcamp \
    kubernetes-bootcamp=gcr.io/k8s-minikube/kubernetes-bootcamp:v2
```

Durch den Befehl wird ein Rolling-Update durchgeführt, ð, die Pods werden im laufenden Betrieb aktualisiert. Durch eine entsprechende Richtlinie wird sichergestellt, dass die Webanwendung weiterläuft. Mit folgendem Befehl kann überprüft werden, welches Image in den Pods verwendet wird.

```
> kubectl describe pods
```

Wenn ein Update fehlschlägt, dann kann der alte Zustand der Webanwendung wieder hergestellt werden. Beispielsweise führt der folgende Befehl zu einem Fehler, da das angegebene Image nicht existiert.

```
> kubectl set image deployments/kubernetes-bootcamp \
    kubernetes-bootcamp=gcr.io/k8s-minikube/kubernetes-bootcamp:v10
```

Betrachtet man die Pods, dann stellt man fest, dass ein Teil der Pods einen Fehler im Status anzeigt. Befehl:

```
> kubectl get pods
```

Mit folgendem Befehl kann das Update wieder zurück genommen werden:

```
> kubectl rollout undo deployments/kubernetes-bootcamp
```

7 Feierabend!

Um das Bootcamp-Deployment zu löschen, gibt man folgende Befehle ein:

```
> kubectl delete svc kubernetes-bootcamp
> kubectl delete deployment kubernetes-bootcamp
```

Anschließend dann das Minikube Cluster gestoppt werden. Befehl:

```
> minikube stop
```