Praktische Einführung in Kubernetes mit Minikube

1. Starten des Minikube-Clusters

- Führe minikube start im Terminal aus.
- Dies startet einen lokalen Kubernetes-Cluster auf deinem Computer. Minikube simuliert einen Cluster, der es dir ermöglicht, Kubernetes-Funktionen zu üben, ohne eine echte Cloud-Infrastruktur zu benötigen.
- Verwende minikube status um zu überprüfen, ob der Cluster läuft.

2. Kurze Einführung in kubectl

- kubectl ist das Kommandozeilenwerkzeug für Kubernetes.
- Mit kubectl version kannst du die Version überprüfen.
- kubectl get nodes zeigt die Knoten im Cluster an.

3. Erstellung eines simplen nginx-Pods

- Erstelle eine YAML-Datei namens nginx-pod.yaml mit folgendem Inhalt:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx
```

- Ein Pod ist die kleinste Einheit in Kubernetes. Hier erstellst du einen Pod, der einen einfachen nginx-Webserver ausführt.
- YAML-Dateien werden verwendet, um Kubernetes-Ressourcen deklarativ zu definieren.
- Führe kubectl apply -f nginx-pod.yaml aus.
- Verwende kubectl get pods um den Status des Pods zu sehen.
- Mit kubectl describe pod nginx-pod erhältst du detaillierte Informationen.

4. Erstellung eines Replica Sets

- Ein Replica Set stellt sicher, dass eine bestimmte Anzahl von Pod-Kopien (Replikas) immer läuft. Dies ist nützlich für Redundanz und Skalierung.
- Erstelle nginx-replicaset.yaml:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
```

```
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx
```

- Führe kubectl apply -f nginx-replicaset.yaml aus um die das ReplicaSet im Cluster zu deployen.
- Verwende kubectl get replicaset und kubectl describe replicaset nginx-replicaset.
- Lösche jetzt den Pod ohne ReplicaSet und einen der ReplicaSet Pods und überprüfe das Verhalten:

```
kubectl delete pod nginx-pod
kubectl get pod
kubectl delete pod nginx-replicaset-{{HIER DEN NAMEN VON EINEM DER PODS EINFÜGEN}}
kubectl get pod
```

5. Erstellung eines Deployments

- Verwende kubectl create deployment:

```
kubectl create deployment nginx-deployment --image=nginx
```

- Dies erstellt ein Deployment mit einem nginx-Container.
- Der kubectl create Befehl kann auch genutzt werden um eine Yaml Datei zu erzeugen
- Nutze kubectl get deployments und kubectl describe deployment nginx-deployment um die Erstellung zu überprüfen.

6. Einführung in Kubernetes Explorer (k9s / lens)

- k9s / lens sind Anwendungen, die das Verwalten von Kubernetes-Clustern vereinfachen.
- Installiere eines dieser Tools und verbinde es mit deinem Minikube-Cluster.
- k9s ist eine Terminal Anwendung und ist in der Verwendung relativ ähnlich wie vim. (Meine Empfehlung)
- Lens/OpenLens ist ein GUI Programm was mehr Features hat aber auch langsamer ist.
- Öffne eines dieser Programme und verbinde dich mit dem Minikube Cluster. Stelle sicher, dass dein Deployment verfügbar ist.
- Richte einen Port Forward ein um auf einen der nginx Pods zugreifen zu können.

7. Erstellung eines Services für das Deployment

- Ein Service in Kubernetes definiert, wie man auf eine Gruppe von Pods (in diesem Fall dein nginx-Deployment) zugreifen kann. Services ermöglichen die Netzwerkkommunikation zu und zwischen Pods.
- Aus diesem Grund kann das Verhalten mit einem LoadBalancer verglichen werden. Wir erhalten einen Hostname und können auf alle 3 Pods unseres Deployments zugreifen.
- Die Verbindung zwischen Pod und Service entsteht durch ein Label Selector.
- Erstelle nginx-service.yaml und wende es mit kubectl apply -f nginx-service.yaml an.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
```

```
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

- Führe kubectl apply -f nginx-service.yaml aus.
- Nutze kubectl get services und kubectl describe service nginx-service.

8. Erstellung einer ConfigMap und Mounting im Deployment

- ConfigMaps erlauben es dir, Konfigurationsdaten von deinen Anwendungscode und Containern zu trennen. Durch das Einbinden dieser ConfigMap in dein Deployment kannst du Konfigurationseinstellungen dynamisch an deine Anwendung weitergeben.
- Es kann jede Datei als ConfigMap erstellt werden und dann im Container gemounted werden.
- Außerdem können Daten als Environment Variables in den Container gereicht werden.
- Erstelle eine neue nginx-index.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-index
data:
  index.html: |
    <h1>Hallo von der ConfigMap</h1>
```

- Erstelle eine ConfigMap mit kubectl apply -f nginx-index.yaml. Aktualisiere dein Deployment, um die ConfigMap zu nutzen.
- Verwende kubectl get configmaps und kubectl describe configmap nginx-index.