NFS-Freigabe als PVC-Speicher im Kubernetes Pod verwenden

Description

Einleitung

In diesem kurzen Artikel geht es darum, wie wir eine **NFS-Freigabe** als **persistenten (PVC) Speicher** verwenden können. Dies verwenden wir z.B. wenn wir keinen Cloud Kubernetes Cluster betreiben, sondern diesen Cluster bei uns im lokalen LAN betreiben.

Über den NFS-Share stellen wir dann sicher, dass jeder Pod Zugriff auf dieselben Daten hat. Ansonsten müssten wir auf einem anderen Weg sicherstellen, dass auf jedem Node die Daten vorhanden sind.

Info: Stelle bitte sicher, dass das Paket `nfs-common` bereits auf allen Nodes installiert ist!

Durchführung

Helm Installation

Im ersten Schritt müssen wir auf unserem **Master Node** einmal den **Helm Paketmanager** installieren. Dazu führen wir den folgenden Befehl aus:

curl -L https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash

Um zu überprüfen, ob die Installation geklappt hat, kannst du den folgenden Befehl ausführen:

helm version

Installation vom nfs-provisioner

Im nächsten Schritt installieren wir den **nfs-subdir-external-provisioner**. Dies ist eine Speicherklasse, um den **NFS-Speicher** in das **Kubernetes Cluster** einzubinden. Kubernetes hat von Haus aus keine Möglichkeit, **NFS-Speicher** anzubinden. Um die Installation durchzuführen und einen dedizierten Namespaces zu erstellen, führe die folgenden Befehle aus:

helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/ kubectl create namespace nfs-provisioner

Starten des nfs-provisioner Pods

In diesem Schritt starten wir den Pod für den **NFS-Provisioner**. Dazu passen wir noch die Serveradresse und den Pfad unserer NFS-Freigabe an.



Anlegen der Manifest-Dateien für den Storage

Im nächsten Schritt erstellen wir eine **YAML-Datei** für das *"PersistentVolume"*. Diese enthält den folgenden Inhalt:



Jetzt legen wir die **YAML-Datei** für das *"PersistentVolumeClaim"* an. Dies erhält die entsprechende Größe, welches unsere Anwendung benötigt. Die Datei sieht wie folgendermaßen aus:



Info zu PV und PVC: Ein PV wird vom Administrator definiert und präsentiert das Speichermedium (Speicher Backend) und das PVC stellt die Anfrage vom Pod an den PV um die entspreche Größe für die App vom Volume zu erhalten. In der Regel erstellt man einen PV pro Speichermedium und einen PVC pro Applikation / Pod.

Im nächsten Schritt müssen wir beide Dateien wie gewohnt mit kubectl apply -f <dateipfad> aktivieren.

Pod Konfiguration anpassen

Um jetzt den Speicher in dem Pod verfügbar zu machen, müssen wir in der **Deployment-Datei** unseres Pods im spec Segment die **volumes** definieren.



Dann können wir jetzt im Container Segment in volumeMounts definieren:



Wenn wir jetzt im Anschluss den Pod starten, und die Berechtigungen stimmen, dann sollten die Daten geschrieben werden und damit sind jetzt die Daten persistent verfügbar.

Beispiel Pod-Konfiguration



```
apiVersion: apps/v1
kind: Deployment
metadata:
   app: nextcloud
spec:
   matchLabels:
     app: nextcloud
       app: nextcloud
   spec:
     containers:
        - name: nextcloud-app
              value: "1000"
            - name: GUID
              value: "1000"
            - name: http
              containerPort: 80
          resources:
              cpu: "250m"
              memory: "4096Mi"
            - mountPath: /config
              name: nfs-volume-nextcloud-config
              name: nfs-volume-nextcloud-data
            server: 192.168.5.4
            path: /mnt/Kubernetes/Daten/nextcloud-config
        - name: nfs-volume-nextcloud-data
            server: 192.168.5.4
            path: /mnt/Kubernetes/Daten/nextcloud-data
apiVersion: v1
kind: Service
```

Category

1. Kubernetes

Date Created 09.02.2025 Author administrator