

Dynamische Persistent Volumes mit Hilfe von NFS Client Provisioner

Manifestfiles

Namespace für den Client Provisioner anlegen

```
kubectl create ns nfs-client
```

Storage Class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-nfs-storage
provisioner: fuseim.pri/ifs # anderer Name ist möglich, muss dann aber über
einstimmen mit der env PROVISIONER_NAME im Deployment
parameters:
  archiveOnDelete: "false"
allowVolumeExpansion: true
```

RBAC

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: nfs-client
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
```

```
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources: ["events"]
  verbs: ["create", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: run-nfs-client-provisioner
subjects:
- kind: ServiceAccount
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: nfs-client
roleRef:
  kind: ClusterRole
  name: nfs-client-provisioner-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: nfs-client
rules:
- apiGroups: [""]
  resources: ["endpoints"]
  verbs: ["get", "list", "watch", "create", "update", "patch"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  namespace: nfs-client
subjects:
- kind: ServiceAccount
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: nfs-client
roleRef:
  kind: Role
  name: leader-locking-nfs-client-provisioner
  apiGroup: rbac.authorization.k8s.io
```

Deployment für den NFS Client Provisioner

```
apiVersion: apps/v1
```

```

kind: Deployment
metadata:
  name: nfs-client-provisioner
  labels:
    app: nfs-client-provisioner
  namespace: nfs-client
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nfs-client-provisioner
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nfs-client-provisioner
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image: gcr.io/k8s-staging-sig-storage/nfs-subdir-external-
provisioner:v4.0.0
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: fuseim.pri/ifs
            - name: NFS_SERVER
              value: nfs.training.lab # e.g. add your nfsserver
ip
            - name: NFS_PATH
              value: /data/dynamic/userX # e.g. nfsshare ÄNDERE DIE
USER_ID
          volumes:
            - name: nfs-client-root
              nfs:
                server: nfs.training.lab # e.g. add your nfsserver
ip
                path: /data/dynamic/userX # e.g. nfsshare ÄNDERE DIE
USER-ID

```

Testen des Provisioners

Erstellen eines Volume Claim

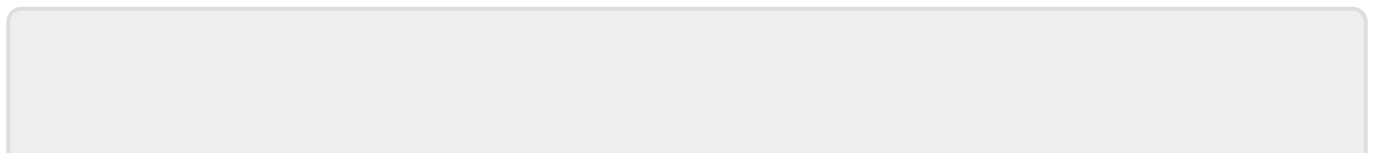
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-claim
  namespace: pv-test
  annotations:
    volume.beta.kubernetes.io/storage-class: "managed-nfs-storage"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

Pod ausrollen mit volume und volume mount

```
kind: Pod
apiVersion: v1
metadata:
  name: test-pod
  namespace: pv-test
spec:
  containers:
    - name: test-pod
      image: docker.io/library/busybox:1.35
      tty: true
      stdin: true
      volumeMounts:
        - name: nfs-pvc
          mountPath: "/mnt"
  restartPolicy: "Never"
  volumes:
    - name: nfs-pvc
      persistentVolumeClaim:
        claimName: test-claim
```

Dem Pod betreten und schauen ob der vlume mount geklappt hat.

```
kubectl exec test-pod -n pv-test -it shell -- /bin/sh
```



From:

<https://wiki.cooltux.net/> - **TuxNet DokuWiki**

Permanent link:

<https://wiki.cooltux.net/doku.php?id=it-wiki:kubernetes:nfs-client-provisioner&rev=1706685864>

Last update: **2024/01/31 07:24**

