Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 5.4 : Kubernetes - Volumes

V2023.02

Table des matières

1 - Introduction

Il est possible d'utiliser des volumes pour utiliser des espaces disques externes aux Pods afin de conserver cet espace à la fin de vie du Pod.

Le plus habituel est le « `hostPath` » qui indique un répertoire local au nœud sur lequel fonctionne le Pod, à ne pas confondre avec « `local` » est utilisé pour l'accès à un périphérique.

La liste des drivers supportés est : `awsElasticBlockStore, azureDisk, azureFile, cephfs, cinder, configMap, csi, downwardAPI, emptyDir, fc (fibre channel), flexVolume, flocker, gcePersistentDisk, gitRepo (deprecated), glusterfs, hostPath, iscsi, local, nfs, persistentVolumeClaim, projected, portworxVolume, quobyte, rbd, scaleIO, secret, storageos, vsphereVolume`

Si il n'existe qu'un seul nœud, un volume local est suffisant, néanmoins ce cas ne présente que peu d'intérêt.

Comme nous l'avons vu avec Swarm, il faut donc impérativement utiliser un système de fichier réseau de type NFS, GlusterFS, Cephfs, … Qui peut être intégré :

- **Au niveau système** : dans ce cas c'est complètement transparent pour Kubernetes qui continue à utiliser des points de montage locaux,
- **Au niveau Kubernetes** : qui permet via des plugins, de monter des systèmes de fichiers réseaux lors de la création des pods et de services (option de mount).

2 - Volume local

Pour utiliser par exemple le « hostPath » :

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # chemin du dossier sur l'hôte
      path: /data
      # ce champ est optionnel
      type: Directory
```

3 - Volume NFS

Il est possible d'utiliser un partage NFS directement dans un pod :

```
kind: Pod
apiVersion: v1
metadata:
  name: pod-using-nfs
spec:
  volumes:
    - name: nfs-volume
      nfs:
        server: 172.16.22.10
        path: /myexport
  containers:
    - name: app
      image: alpine

      volumeMounts:
        - name: nfs-volume
          mountPath: /var/nfs

      command: ["/bin/sh"]
               args:  ["-c",  "while  true;  do  date  >>
/var/nfs/dates.txt; sleep 5; done"]
```

4 - Volumes persistants

4.1 - Introduction

Il existe deux types de ressources pour gérer les volumes persistants :

- `PersistentVolume` (PV) : Qui est un élément de stockage disponible dans le cluster, et qui doit être définit par l'administrateur de ce dernier,
- `PersistentVolumeClaim` (PVC) : Qui est la ressource demandée par une application (pod, déploiement, service, …), à prendre dans un « PersistentVolume ».

4.2 - PersistentVolume

Les modes d'accès sont :

- **ReadOnlyMany** : le volume peut être monté en lecture seule par plusieurs nœuds
- **ReadWriteOnce** : le volume peut être monté en lecture-écriture par un seul nœud
- **ReadWriteMany** : lecture-écriture pour plusieurs nœuds

La policy peut être :

- **Retain** : Persistent, remise en état manuelle,
- **Recycle** : effacement de base (rm -rf /thevolume/*, uniquement NFS et HostPath),
- **Delete** : l'élément de stockage associé tel qu'AWS EBS, GCE PD, Azure Disk ou le volume OpenStack Cinder est supprimé,

Pour lister les PV :

```
root@maitre:~# kubectl get pv
NAME   CAPACITY ACCESS MODES RECLAIM POLICY   STATUS    CLAIM  STORAGECLASS REASON AGE
pvtest 5Gi      RWO          Recycle          Available                          2m16s
```

PV local

Premier exemple de PV en local, qui s'utilise donc soit pour un seul nœud, soit avec un système de fichier lié à un répertoire monté en réseau (NFS, GlusterFS, ...) :

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvlocal
  labels:
    type: local
spec:
  storageClassName: ""
  capacity:
    storage: 30Gi
  persistentVolumeReclaimPolicy: Recycle
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/var/data"
```

NFS

Dans le cas de NFS, il est possible de monter le partage directement dans le « Claim » sans passer par un montage local :

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvnfs
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: ""
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /var/nfs/tmp
    server: 10.3.134.XXX
```

Pour GlusterFS il est possible de passer par un « endpoint » :

```
apiVersion: v1
kind: Endpoints
metadata:
  name: glusterfs-cluster
subsets:
- addresses:
  - ip: 10.3.134.187
  ports:
  - port: 49152
- addresses:
  - ip: 10.3.134.188
  ports:
  - port: 49152
- addresses:
  - ip: 10.3.134.189
  ports:
  - port: 49152
```

Pour vérifier on utilise la commande : « kubectl get ep »

Ensuite on peut créer le PV :

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvgfs
spec:
  capacity:
    storage: 20Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: ""
  glusterfs:
    endpoints: glusterfs-cluster
    path: gfs1
    readOnly: no
```

4.3 - PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 2Gi
  storageClassName: ""
```

Pour lister :

```
root@maitre:~# kubectl get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myclaim   Bound    pvtest   5Gi        RWO            slow           19s
root@maitre:~# kubectl get pv
NAME   CAPACITY ACCESS MODES RECLAIM POLICY STATUS    CLAIM           STORAGECLASS REASON AGE
pvbd   50Gi     RWO          Retain         Available                 database            8m12s
pvtest 5Gi      RWO          Recycle        Bound     default/myclaim                     2m16s
```

Exemple d'utilisation :

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: mypd
        subPath: html
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

Remarque : Pour éviter sur l'application écrive directement dans le point de montage lié au PV, il est préférable d'indiquer un « subpath » qui sera créé automatiquement.