Kubernetes Volume

- Objectif :
  - Comprendre la notion de volume dans K8S
  - Exporter les systèmes de fichiers avec NFS

- Intro : L’exemple de vSphere et du SAN
  - VM : sont d

K8S volume : Contexte

- Docker a un concept de volumes.
  - Avec Docker, un volume est simplement un dossier sur le disque ou dans un autre conteneur.
  - Les durées de vie ne sont pas gérées et, jusqu'à très récemment, seuls les volumes supportés par un disque local l'étaient.
  - Docker fournit maintenant des pilotes de volume, mais la fonctionnalité est très limitée pour le moment (par exemple, à partir de Docker 1.7, seulement un pilote de volume est autorisé par conteneur et il n'est pas possible de passer des paramètres aux volumes).

K8S volume : Contexte

- Un volume Kubernetes
  - a une durée de vie explicite - la même que le Pod qui l'inclut.
  - Par conséquent, un volume survit aux conteneurs qui s'exécutent à l'intérieur du Pod et les données sont préservées lorsque le conteneur redémarre.
  - Bien sûr, lorsqu'un Pod cesse d'exister, le volume va également cesser d'exister.
  - Kubernetes supporte de nombreux types de volumes et un Pod peut en utiliser plusieurs simultanément.
  - A titre d'exemple, il existe de volume pour les différents fournisseur de cloud (AWS, Azure, Google Compute Engine, etc)
  - Nous utiliserons le type NFS

Source : https://kubernetes.io/fr/docs/concepts/storage/volumes/

PersistentVolume

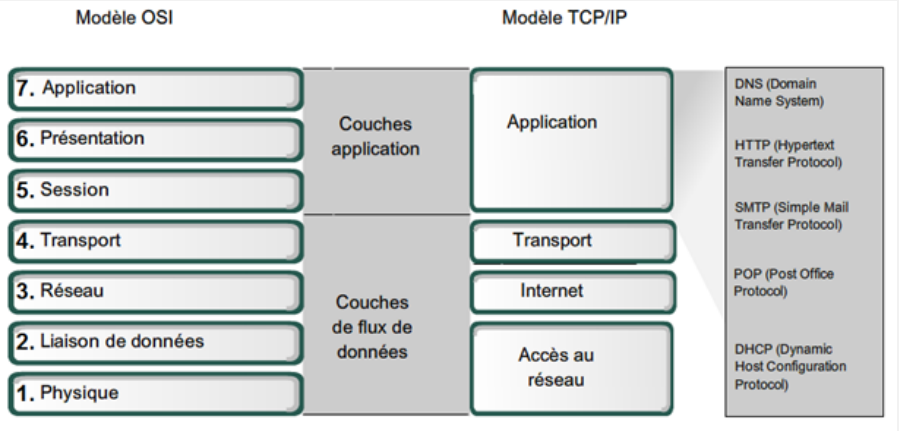
- Dans kubernetes, la persistance est généré à partir des volumes persistants (PersistantVolume)

- Pour utiliser un stockage persistant, vous devez créer un objet PersistantVolume (PV) pour réserver de l'espace. Cette opération est généralement faite par un administrateur système qui peut en créer avec différentes caractéristiques : vitesse, type de support, redondance, etc.

- Une fois le volume persistant créé, les développeurs font un objet de type PersistantVolumeClaim (PVC)

NFS

- NFS (*Network File System*) est un protocole client-serveur de partage de fichiers et répertoires, créé par Sun Microsystems en 1984, et devenu un standard de fait en environnement Unix. Défini par une série de RFC, il a été porté sur Linux ainsi que sur Windows.
- NFS est un protocole de niveau applicatif. Il s'appuie sur un protocole de niveau session, les RPC (*Remote Procedure Call*, appel de procédure distante), également créé initialement par Sun Microsystems.
- NFS permet à un serveur de définir des répertoires partagés, accessibles à travers le réseau à des clients NFS.



Les ***requests for comments*** (**RFC**), littéralement « demande de commentaires », sont une série numérotée de documents décrivant les aspects et spécifications techniques d'Internet, ou de différents matériels informatiques (routeurs, serveur DHCP). Peu de RFC sont des standards, mais tous les documents publiés par l'IETF sont des RFC. (Source Wikipedia)

NFS sur le serveur

- Le serveur NFS est constitué de plusieurs daemons et dépend de l'activation préalable du daemon gérant la couche RPC, rpcbind (ou portmapper).
  - Il faut donc au moins trois daemons actifs pour que le serveur NFS soit opérationnel :
    - rpcbind (ou portmapper) : ce daemon gère les requêtes RPC.
    - nfsd : ce daemon gère les demandes des clients NFS.
    - mountd : ce daemon gère les demandes de montage des clients NFS.

Installation sur comme service

Installation : $sudo apt install nfs-kernel-server

On cherche les scripts
de démarrage init system V :

On vérifie l'état du service de RPC, rpcbind :

On vérifie le serveur NFS et on l'active si nécessaire :

```
/etc/init.d/nfs-common  /etc/init.d/nfs-kernel-server  /etc/init.d/rpcbind
jpduches@bilbo:~$ systemctl status rpcbind
● rpcbind.service - RPC bind portmap service
     Loaded: loaded (/lib/systemd/system/rpcbind.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-04-27 10:22:37 EDT; 4min 47s ago
TriggeredBy: ● rpcbind.socket
       Docs: man:rpcbind(8)
   Main PID: 15138 (rpcbind)
      Tasks: 1 (limit: 19090)
     Memory: 596.0K
        CPU: 6ms
     CGroup: /system.slice/rpcbind.service
             └─15138 /sbin/rpcbind -f -w

avr 27 10:22:37 bilbo systemd[1]: Starting RPC bind portmap service...
avr 27 10:22:37 bilbo systemd[1]: Started RPC bind portmap service.
jpduches@bilbo:~$ systemctl status nfs-server
● nfs-server.service - NFS server and services
     Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
     Active: active (exited) since Wed 2022-04-27 10:22:42 EDT; 9min ago
   Main PID: 15725 (code=exited, status=0/SUCCESS)
        CPU: 5ms

avr 27 10:22:41 bilbo systemd[1]: Starting NFS server and services...
avr 27 10:22:41 bilbo exportfs[15724]: exportfs: can't open /etc/exports for reading
avr 27 10:22:42 bilbo systemd[1]: Finished NFS server and services.
jpduches@bilbo:~$
```

Le partages des répertoires

- Le fichier /etc/exports est un fichier texte dans lequel sont définis les répertoires à partager, avec leurs paramètres de partage. Ces partages sont automatiquement activés au démarrage du serveur NFS.
- Un partage est défini sur une ligne, selon le format suivant 0
  : CheminRep [IdentClient1[(Options)] ... IdentClientN[(Options)]]

`CheminRep`	Chemin d'accès du répertoire à partager.
`IdentClient`	Nom d'hôte, de domaine, adresse IP ou réseau de(s) client(s) NFS. Par défaut : tous les clients.
`Options`	Options de partage pour le(s) client(s) spécifié(s).

Les principales options sont les suivantes :

`ro`	Accès en lecture seule (par défaut).
`rw`	Accès en lecture et écriture.
`sync`	Accès en écriture synchrone. Les données sont écrites immédiatement.
`async`	Accès en écriture asynchrone (par défaut).
`root_squash`	Le compte super-utilisateur (UID 0) côté client est remplacé côté serveur par un compte utilisateur non super-utilisateur (par défaut).
`no_root_squash`	Le compte super-utilisateur (UID 0) côté client le reste côté serveur.

Partage du serveur K8S maitre (10.100.2.90)

```
dev@ivk8sc02m01:/dev$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/srv/exports 10.100.0.0/16(rw,sync,all_squash,anonuid=1000,anongid=1000)
dev@ivk8sc02m01:/dev$
```

Côté client

- sudo apt install nfs-common
- mount -t nfs [-o Options] IdServeur:/CheminRep PointMontage

```
jpduches@VM-DevOpsJPD:~$ sudo mount -t nfs 10.100.2.90:/srv/exports/jpd /home/jpduches/nfs
```

Utilisation de NFS dans un manifeste

PV

- Voici un exemple de configuration de PV qui utilise notre serveur NFS et se lie au chemin jpd/nginx en réservant 200 Mo :

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: monpvnfs
spec:
  storageClassName: manual
  capacity:
    storage: 200Mi
  accessModes:
  - ReadWriteMany
  nfs:
    server: 10.100.1.6
    path: "/srv/exports/jpd/nginx"
```

PVC

- Pour faire une réclamation de volume persistant, il suffit de spécifier le nom du PVC et du PV :

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mon-pvc-nfs
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 100Mi
  volumeName: monpvnfs
```

Utilisation de NFS dans un manifeste

- Une fois le PVC créé, vous pouvez l'utiliser et le monter dans un conteneur. Ici on monte le PVC "mon-pvc-nfs" vers le chemin /usr/share/nginx/html.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-deploy-test-volume
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /usr/share/nginx/html
      name: www
  volumes:
  - name: www
    persistentVolumeClaim:
      claimName: mon-pvc-nfs
```