Chapitre 6
Persistance des données

1. Objectifs du chapitre et prérequis

Ce chapitre présente la persistance des données dans Kubernetes. En effet, le contenu d'un container n'a pas vocation à perdurer. Ce mécanisme est là pour permettre de conserver une information lorsque c'est nécessaire (bases de données, serveurs de fichiers, etc.).

Le chapitre abordera deux points de vue :

– l'utilisateur faisant appel aux volumes persistants,

– l'administrateur mettant en place ce mécanisme.

2. Persistance des données

2.1 Origine du besoin

Dans ce qui a précédé, vous avez pu aborder le cycle de vie du container dans Kubernetes. Un point important à retenir est qu'un container a une durée de vie relativement courte et, qu'en l'état, il n'est pas possible de conserver de la donnée.

Pour répondre à ce besoin, Kubernetes peut mettre à disposition des espaces de stockage externes au cluster. Ce mécanisme s'appuie sur la notion de volume de données persistant (*Persistent Volume*).

2.2 Utilisation d'un volume persistant externe

L'utilisation d'un volume persistant se fait au sein de la déclaration d'un pod. Une première solution pourrait être de passer par un service externe comme avec un point de montage NFS.

Cette déclaration de persistance de données se définira en deux parties :

– un référencement au niveau du pod dans le champ `volumes`,

– une indication de l'emplacement du montage au sein du container.

Le référencement d'un volume NFS au niveau d'un pod se présentera sous la forme suivante :

```
spec:
 volumes:
   - name: nfs
     nfs:
       # URL for the NFS server
       server: 192.168.0.1
       path: /
```

Le montage au niveau du container se présentera ainsi :

```
spec:
 containers:
   - name: mailhog
     image: mailhog/mailhog

     # Mount the NFS volume in the container
     volumeMounts:
       - name: nfs
         mountPath: /maildir
```

2.3 Volumes persistants

2.3.1 Structure du volume persistant

En plus de faire appel à des services externes, il est possible de référencer des objets de type volume persistant (*Persistent Volume*). Ces derniers ont la structure suivante :

- une version et un type (`apiVersion` et `kind`),
- des métadonnées (champ `metadata`),
- une spécification contenant :
  - le type d'accès,
  - la capacité de stockage,
  - la classe du volume persistant (positionner à `manual` pour l'exemple),
  - les caractéristiques du stockage.

Ci-dessous un exemple de stockage portant le nom de pv-mailhog s'appuyant sur un répertoire de la machine hôte. Ce volume a une capacité déclarée de 10 Mo :

```
apiVersion: v1
kind: PersistentVolume
metadata:
   name: pv-mailhog
spec:
   accessModes:
     - ReadWriteOnce
   storageClassName: manual
   capacity:
     storage: 10Mi
   hostPath:
     path: /tmp/pv-mailhog
```

Sauvegardez cette déclaration dans le fichier **pv-mailhog.yaml**.

2.3.2 Création du volume persistant

La déclaration réalisée, l'application de cette déclaration se fera à l'aide de la commande `kubectl` suivie des indications suivantes :

– le mot-clé apply,

– l'option -f suivie du nom de fichier.

Ci-dessous la commande correspondante :

```
$ kubectl apply -f pv-mailhog.yaml
```

Cette commande doit alors renvoyer la valeur suivante :

```
persistentvolume/pv-mailhog created
```

Le volume est déclaré. Reste maintenant à voir comment l'utiliser dans un container.

2.4 Persistance de données avec MailHog

2.4.1 Opérations à réaliser

Par défaut, lorsque MailHog reçoit un message, ce dernier le stocke en local dans un répertoire de travail. Malheureusement, lorsque le pod est relancé ou redémarré, la donnée se perd.

Afin de faire face à ce problème, ajoutez un point de montage dans le container de MailHog.

Pour cela, deux modifications seront réalisées dans l'objet déploiement :

– utilisation d'un volume persistant,

– modification des options de lancement de MailHog.

En plus de ces modifications dans le déploiement, vous devrez également déclarer un objet permettant de référencer le volume persistant.

Cet objet fera le lien entre le volume persistant et la déclaration d'utilisation.

2.4.2 Déclaration de l'objet PersistentVolumeClaim

La demande de volume persistant (`PersistentVolumeClaim` ou son raccourci `pvc`) réclame un certain nombre d'informations :

– les champs `apiVersion` et `kind`,
– les métadonnées (`metadata`),
– les spécifications de l'objet :
  – le mode d'accès,
  – la classe précisée précédemment (`manual`),
  – la quantité de ressources demandées,
  – le nom du volume persistant.

Dans le cas d'un objet `PersistentVolumeClaim` portant le nom de `pvc-mailhog` pour un volume de 10 Mo, la déclaration sera la suivante :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: pvc-mailhog
spec:
   accessModes:
     - ReadWriteOnce
   storageClassName: manual
   resources:
     requests:
       storage: 10Mi
   volumeName: pv-mailhog
```

Stockez cette déclaration dans le fichier **pvc-mailhog.yaml** et appliquez cet objet à l'aide de la commande suivante :

```
$ kubectl apply -f pvc-mailhog.yaml
```

Cette commande doit alors renvoyer la valeur suivante :

```
persistentvolumeclaim/pvc-mailhog created
```

2.4.3 État des objets de volume persistant

La consultation de ces objets se fait traditionnellement avec les méthodes `get` ou `describe`.

Ci-dessous la commande pour consulter la liste des volumes persistants (`PersistentVolume` ou `pv`) :

```
$ kubectl get persistentvolume
```

Ci-dessous une première partie de la sortie de cette commande :

```
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS ...
pv-mailhog    10Mi       RWX            Retain           Bound  ...
```

Et ci-dessous les champs suivants :

```
NAME          ... CLAIM                  STORAGECLASS    ...
pv-mailhog    ... default/pvc-mailhog    manual          ...
```

L'ensemble de ces champs va donner les informations suivantes :

- `NAME` : nom du volume persistant.
- `CAPACITY` : taille du volume persistant.
- `ACCESS MODES` : types d'accès autorisés.
- `RECLAIM POLICY` : permet de spécifier le comportement à adopter en cas de suppression de l'objet `PersistentVolumeClaim` (garder le volume ou suppression automatique).
- `STATUS` : si le volume est associé (`Bound`) ou détaché (`Released`) vis-à-vis d'un objet `PersistentVolumeClaim`.
- `CLAIM` : nom de l'objet `PersistentVolumeClaim` associé.
- `STORAGECLASS` : classe utilisée par le volume.

Le volume persistant pv-mailhog est bien lié à l'objet `PersistentVolume-Claim pvc-mailhog`.