STOCKAGE PERSISTANT

Gestion des volumes dans Kubernetes

Comprendre les mécanismes de stockage persistant
pour vos applications conteneurisées

TYPES DE VOLUMES

Les différentes options de stockage dans Kubernetes

emptyDir

Volume temporaire créé lorsqu'un pod est assigné à un nœud. Survit aux redémarrages de conteneurs mais est supprimé lorsque le pod est supprimé.

hostPath

Monte un fichier ou un répertoire du système de fichiers du nœud hôte dans le pod. Utile pour accéder aux fichiers du nœud, mais présente des risques de sécurité.

nfs

Monte un partage NFS dans le pod. Permet le partage de données entre pods sur différents nœuds.

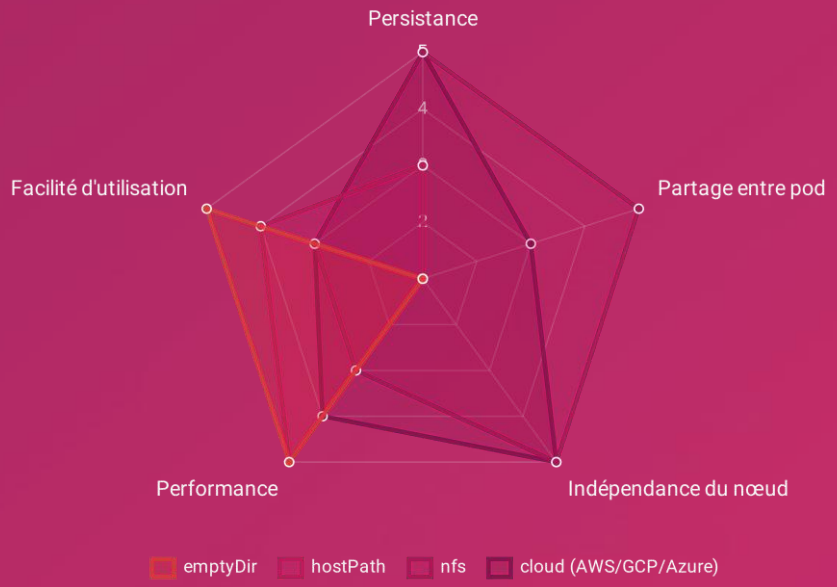
awsElasticBlockStore / gcePersistentDisk / azureDisk

Volumes spécifiques aux fournisseurs cloud pour monter des disques persistants dans les pods.

secret / configMap

Fournissent un moyen de monter des données de configuration ou des informations sensibles dans les pods.

Caractéristiques des types de volumes



PERSISTENT VOLUMES (PV)

Ressources de stockage indépendantes du cycle de vie des pods

Qu'est-ce qu'un Persistent Volume ?

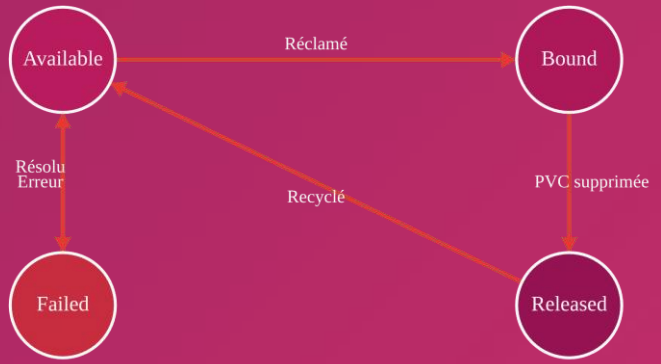
Un Persistent Volume (PV) est une ressource de stockage dans le cluster qui a été provisionnée par un administrateur ou dynamiquement via des Storage Classes. C'est une ressource au niveau du cluster, indépendante du cycle de vie des pods qui l'utilisent.

- **Indépendance** : Existe indépendamment des pods qui l'utilisent
- **Abstraction** : Masque les détails de l'implémentation du stockage
- **Modes d'accès** : ReadWriteOnce, ReadOnlyMany, ReadWriteMany

Exemple de définition de PV

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-example
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: standard
  nfs:
    server: nfs-server.example.com
    path: "/exports"
```

Cycle de vie d'un Persistent Volume



Politiques de réclamation

- **Delete** : Le PV est supprimé automatiquement lorsque la PVC est supprimée
- **Retain** : Le PV est conservé avec ses données, mais doit être nettoyé manuellement avant réutilisation
- **Recycle** : (Déprécié) Le contenu du volume est supprimé et le PV est disponible pour une nouvelle reclamation.

PERSISTENT VOLUME CLAIMS (PVC)

Demandes de stockage par les applications

Qu'est-ce qu'un Persistent Volume Claim ?

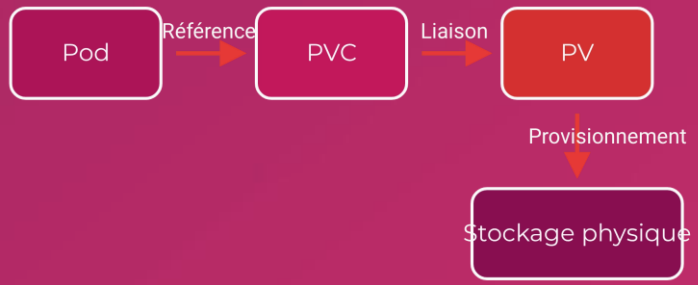
Un Persistent Volume Claim (PVC) est une demande de stockage par un utilisateur ou une application. Il représente une requête pour un Persistent Volume avec des caractéristiques spécifiques (taille, mode d'accès, classe de stockage).

- **Abstraction** : Les développeurs n'ont pas besoin de connaître les détails de l'infrastructure de stockage sous-jacente
- **Liaison** : Un PVC est lié à un PV qui répond à ses exigences (taille, mode d'accès)
- **Namespace** : Les PVCs sont liés à un namespace spécifique, contrairement aux PVs qui sont des ressources au niveau du cluster

Exemple de définition de PVC

```
apiVersion: v1 kind: PersistentVolumeClaim metadata: name: mysql-data-claim namespace: application spec: accessModes: - ReadWriteOnce resources: requests: storage: 5Gi storageClassName: standard
```

Relation entre PV et PVC



Utilisation dans un Pod

```
apiVersion: v1 kind: Pod metadata: name: mysql namespace: application spec: containers: - name: mysql image: mysql:5.7 volumeMounts: - name: mysql-data mountPath: /var/lib/mysql volumes: - name: mysql-data persistentVolumeClaim: claimName: mysql-data-claim
```

PERSISTENT VOLUME CLAIMS (PVC)

Exemple de définition de PVC

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: application
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:8.0
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: "yourpassword"
        volumeMounts:
        - name: mysql-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-storage
        persistentVolumeClaim:
          claimName: mysql-data-claim  # ⬅️ Référence au PVC
```

Utilisation dans un Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql
  namespace: application
spec:
  containers:
  - name: mysql
    image: mysql:5.7
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "examplepassword"
    volumeMounts:
    - name: mysql-data
      mountPath: /var/lib/mysql
  volumes:
  - name: mysql-data
    persistentVolumeClaim:
      claimName: mysql-data-claim
```

⚠️ **Important** : Si aucun PV ne correspond aux exigences du PVC, celui-ci reste en état "Pending" jusqu'à ce qu'un PV correspondant soit disponible

CLASSES DE STOCKAGE (STORAGECLASS)

Provisionnement dynamique des volumes persistants

Qu'est-ce qu'une StorageClass ?

Une StorageClass est une ressource Kubernetes qui définit comment les volumes persistants sont provisionnés dynamiquement lorsqu'une PVC les réclame. Elle permet aux administrateurs de décrire les "classes" de stockage qu'ils offrent.

- **Provisionnement dynamique** : Création automatique des PV à la demande
- **Paramètres personnalisables** : Type de stockage, performances, réplication, etc.
- **Abstraction** : Masque les détails d'implémentation du stockage aux utilisateurs
- **Politique de réclamation par défaut** : Delete, Retain ou Recycle

Modes de liaison des volumes

- **Immediate**: Le volume est provisionné dès que la PVC est créée (comportement par défaut)
- **WaitForFirstConsumer** : Le volume est provisionné uniquement lorsqu'un pod utilisant la PVC est créé

Le mode **WaitForFirstConsumer** est particulièrement utile dans les environnements multi-zones, car il permet de provisionner le volume dans la même zone que le pod qui l'utilise, évitant ainsi les problèmes de latence ou d'accessibilité.

Exemple de StorageClass

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fast
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
  fsType: ext4
  encrypted: "true"
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```