

# Les Volumes

# Les ressources en jeu

- Volume : partage de données entre les containers d'un Pod
- PersistentVolume (PV) : stockage provisionné par un administrateur
- PersistentVolumeClaim (PVC) : demande de stockage, consomme un PV
- StorageClass : provisionnement dynamique du stockage
- StatefulSet

# Volume


- Découple les données du cycle de vie d'un container
- Permet aux containers d'un même Pod de partager des données
- Nombreux types disponibles



# Volume, ex : EmptyDir

- Lié au cycle de vie d'un Pod
- Vide à la création


```
apiVersion: v1
kind: Pod
metadata:
  name: mongo
spec:
  containers:
  - image: mongo:4.0
    name: mongo
    volumeMounts:
    - mountPath: /data/db
      name: data
  volumes:
  - name: data
    emptyDir: {}
```



mongo-emptydir.yaml

```
$ kubectl create -f mongo-emptydir.yaml
$ kubectl exec -ti mongo -- bash
```

```
root@mongo:/# mongo
> use test
switched to db test
> db.k8s.insert({ok: 1})
```

```
root@mongo:/# kill 1 
```

```
$ kubectl exec -ti mongo -- bash
root@mongo:/# mongo
> use test
switched to db test
> db.k8s.find()
```

# Volume, ex : hostPath

- Montage d'une ressource de la machine hôte dans un Pod (répertoire, fichier, ..
- Exemples: monitoring, communication avec le daemon Docker, ...

```
apiVersion: v1
kind: Pod
metadata:
  name: mongo
spec:
  containers:
    - image: mongo:3.6
      name: mongo
      volumeMounts:
        - mountPath: /data/db
          name: data
  volumes:
    - name: data
      hostPath:
        path: /data-db
```



mongo-hostpath.yaml

```
$ kubectl create -f mongo-hostpath.yaml

# Connection ssh dans la VM de Minikube
$ minikube ssh

# Inspection du répertoire /data-db
# cd /data-db/
# ls
WiredTiger
_mdb_catalog.wt
index-3-2054357870167412243.wt
WiredTiger.lock
Collection-0-2054357870167412243.wt
journal
...
```

# Volume, ex : Amazon awsElasticBlockStore (EBS)

- Montage d'un EBS dans un Pod
- Quelques contraintes
  - Disponible pour des VMs sur AWS
  - utilisé par une instance EC2 à la fois
  - même région et AZ

```
$ aws ec2 create-volume \  
  --availability-zone=eu-west-1a \  
  --size=10 \  
  --volume-type=gp2
```



```
apiVersion: v1  
kind: Pod  
metadata:  
  name: mongo  
spec:  
  containers:  
  - image: mongo:3.6  
    name: mongo  
    volumeMounts:  
    - mountPath: /data/db  
      name: data  
  volumes:  
  - name: data  
    awsElasticBlockStore:  
      volumeID: EXISTING_VOLUME_ID  
      fsType: ext4
```



# Volume, ex : Google gcePersistentDisk (PD)

- Montage d'un PD dans un Pod
- Quelques contraintes
  - disponible pour des VMs sur GCP
  - une seule VM avec un accès en écriture
  - même projet et zone que le PD

```
$ gcloud compute disks create \  
  --size=500GB \  
  --zone=us-central1-a \  
  DISK_NAME
```



```
apiVersion: v1  
kind: Pod  
metadata:  
  name: mongo  
spec:  
  containers:  
  - image: mongo:3.6  
    name: mongo  
    volumeMounts:  
    - mountPath: /data/db  
      name: data  
  volumes:  
  - name: data  
    gcePersistentDisk:  
      pdName: DISK_NAME  
      fsType: ext4
```

# PersistentVolume

- Stockage provisionné statiquement, ou dynamiquement
- Découplage workload / stockage



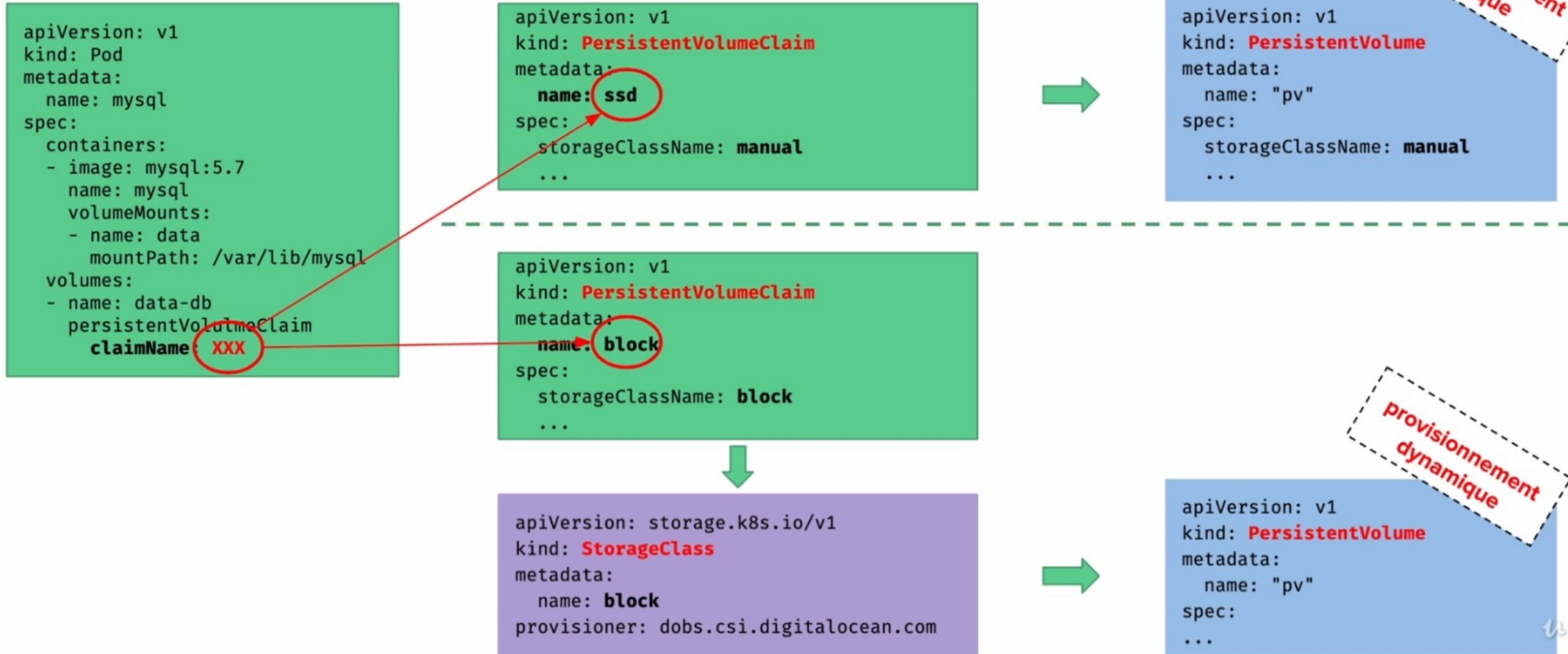
A word cloud of various storage technologies and protocols. The words are arranged in a circular pattern, with some appearing more frequently than others. The colors of the words vary, including red, orange, yellow, green, and blue. The words include: GCEPD, AWSEBS, CephFS, iSCSI, Flocker, RBD, ScaleIO, Quobyte, StorageOS, AzureDisk, CSI, Flexvolume, AzureFile, HostPath, Glusterfs, Cinder, NFS, and Portworx.

# PersistentVolumeClaim

- Demande de stockage
- Spécifie des contraintes
  - taille
  - type
  - mode d'accès
- Consomme un PV existant ou provisionnement dynamique via une StorageClass
- Utilisé par un Pod



# PV / PVC / SC : vue d'ensemble



# Provisionnement statique : création d'un PV

Exemple sur  
Minikube

```
$ cat <<EOF | kubectl apply -f -
```

```
apiVersion: v1
```

```
kind: PersistentVolume
```

```
metadata:
```

```
  name: pv
```

```
spec:
```

```
  storageClassName: manual
```

```
  capacity:
```

```
    storage: "1Gi"
```

```
  accessModes:
```

```
    - "ReadWriteOnce"
```

```
  hostPath:
```

```
    path: /data/pv
```

```
EOF
```

```
$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv	1Gi	RWO	Retain	Available		manual		3s

# Provisionnement statique : création d'un PVC

Exemple sur  
Minikube

```
$ cat <<EOF | kubectl apply -f -
```

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
```

```
metadata:
```

```
  name: claim
```

```
spec:
```

```
  storageClassName: manual
```

```
  accessModes:
```

```
    - ReadWriteOnce
```

```
  resources:
```

```
    requests:
```

```
      storage: 1Gi
```

```
EOF
```

```
$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
claim	<b>Bound</b>	pv	1Gi	RWO	manual	50s

# Provisionnement statique : utilisation du PVC

Exemple sur  
Minikube

```
$ cat <<EOF | kubectl apply -f -
kind: Pod
apiVersion: v1
metadata:
  name: mongo
spec:
  containers:
    - name: mongo
      image: mongo:4.0
      volumeMounts:
        - mountPath: /data/db
          name: data-db
  volumes:
    - name: data-db
      persistentVolumeClaim:
        claimName: claim
EOF
```



```
$ minikube ssh

# ls -l /data/pv
WiredTiger
WiredTiger.lock
WiredTiger.turtle
WiredTiger.wt
WiredTigerLAS.wt
_mdb_catalog.wt
collection-0-6097105695909219062.wt
collection-2-6097105695909219062.wt
diagnostic.data
index-1-6097105695909219062.wt
index-3-6097105695909219062.wt
journal
mongod.lock
sizeStorer.wt
storage.bson
...
```



# Provisionnement dynamique : storageClass

Exemple sur  
DigitalOcean

```
$ kubectl get sc
```

NAME	PROVISIONER	AGE
do-block-storage (default)	dobs.csi.digitalocean.com	4m2s

```
$ kubectl get sc/do-block-storage -o yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: do-block-storage
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: dobs.csi.digitalocean.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
...
```

```
$ kubectl get sc
```

NAME	PROVISIONER	AGE
standard (default)	k8s.io/minikube-hostpath	5m1s

```
$ kubectl get sc/standard -o yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: k8s.io/minikube-hostpath
...
```



# Provisionnement dynamique : création d'un PVC

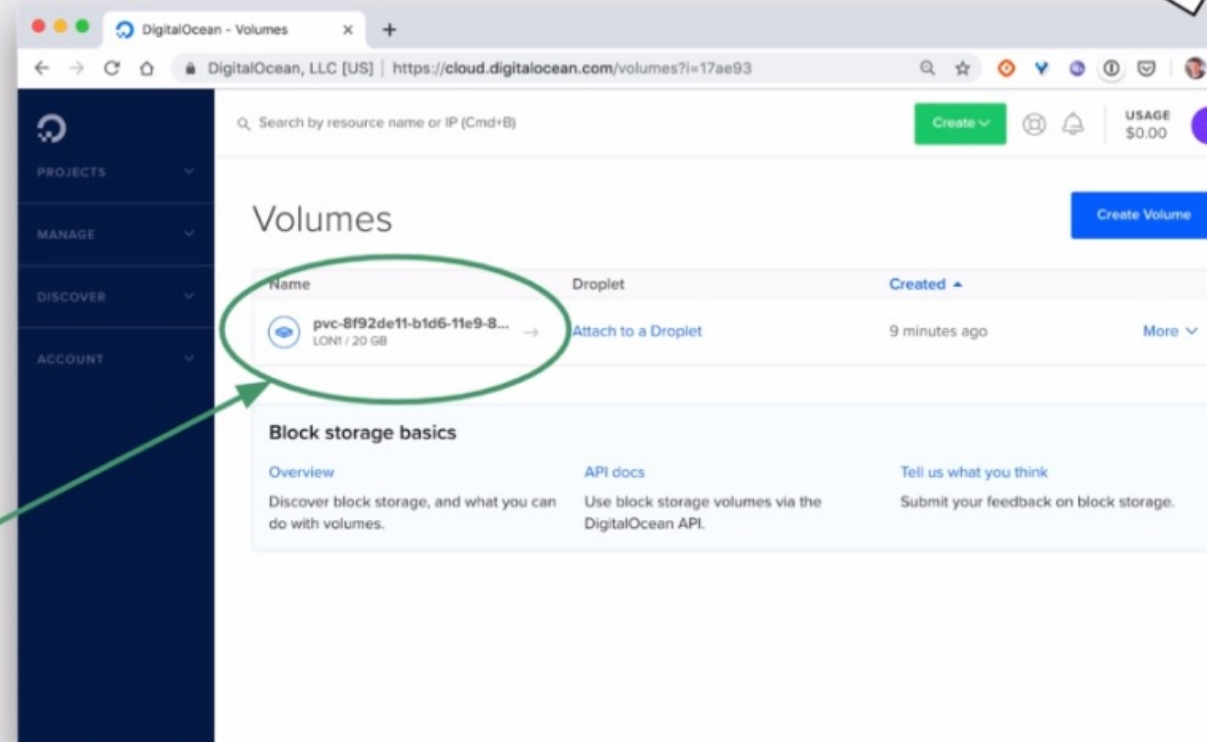
Exemple sur  
DigitalOcean

```
$ cat <<EOF | kubectl apply -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: block
spec:
  storageClassName: do-block-storage
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
EOF
```

```
$ kubectl get pvc,pv
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc/block	Bound	pvc-8f9...	20Gi	RWO	do-block-storage	10s

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv/pvc-8f9...	20Gi	RWO	Delete	Bound	default/block	do-block-storage		9s



# Provisionnement dynamique : utilisation du PVC

Exemple sur  
DigitalOcean

```
$ cat <<EOF | kubectl apply -f -
kind: Pod
apiVersion: v1
metadata:
  name: mongo
spec:
  containers:
  - name: mongo
    image: mongo:4.0
    volumeMounts:
    - mountPath: /data/db
      name: data-db
  volumes:
  - name: data-db
    persistentVolumeClaim:
      claimName: block
EOF
```



```
$ kubectl get pod
NAME      READY   STATUS    RESTARTS   AGE
mongo     1/1     Running   0           43s
$ kubectl get pod/mongo -o yaml -o=jsonpath='{.spec.nodeName}'
worker-pool-j59z
```

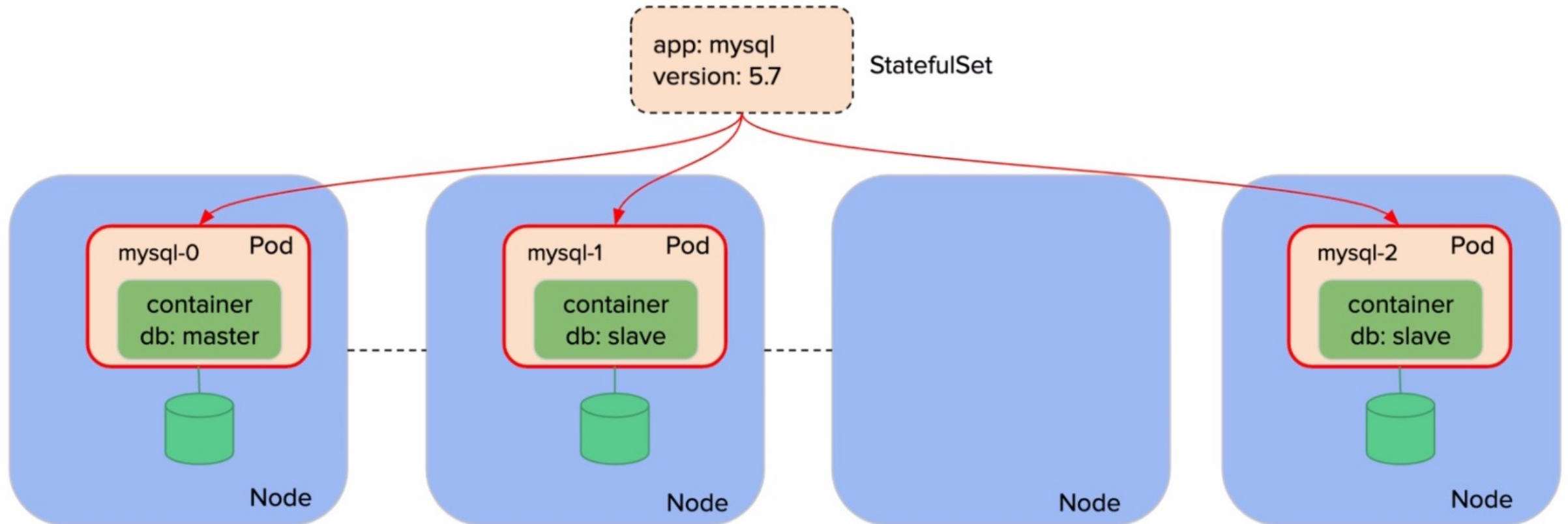
The screenshot shows the DigitalOcean Volumes page. The URL is <https://cloud.digitalocean.com/volumes?i=17ae93>. The page displays a table of volumes with the following columns: Name, Droplet, and Created. A volume named 'pvc-8f92de11-b1d6-11e9-8...' is shown, which is linked to a droplet named 'worker-pool-j59z' (4 GB / 80 GB / LON1). The volume was created 12 minutes ago. Below the table, there is a section titled 'Block storage basics' with links for 'Overview', 'API docs', and 'Tell us what you think'.

Name	Droplet	Created
pvc-8f92de11-b1d6-11e9-8... LON1 / 20 GB	worker-pool-j59z 4 GB / 80 GB / LON1	12 minutes ago

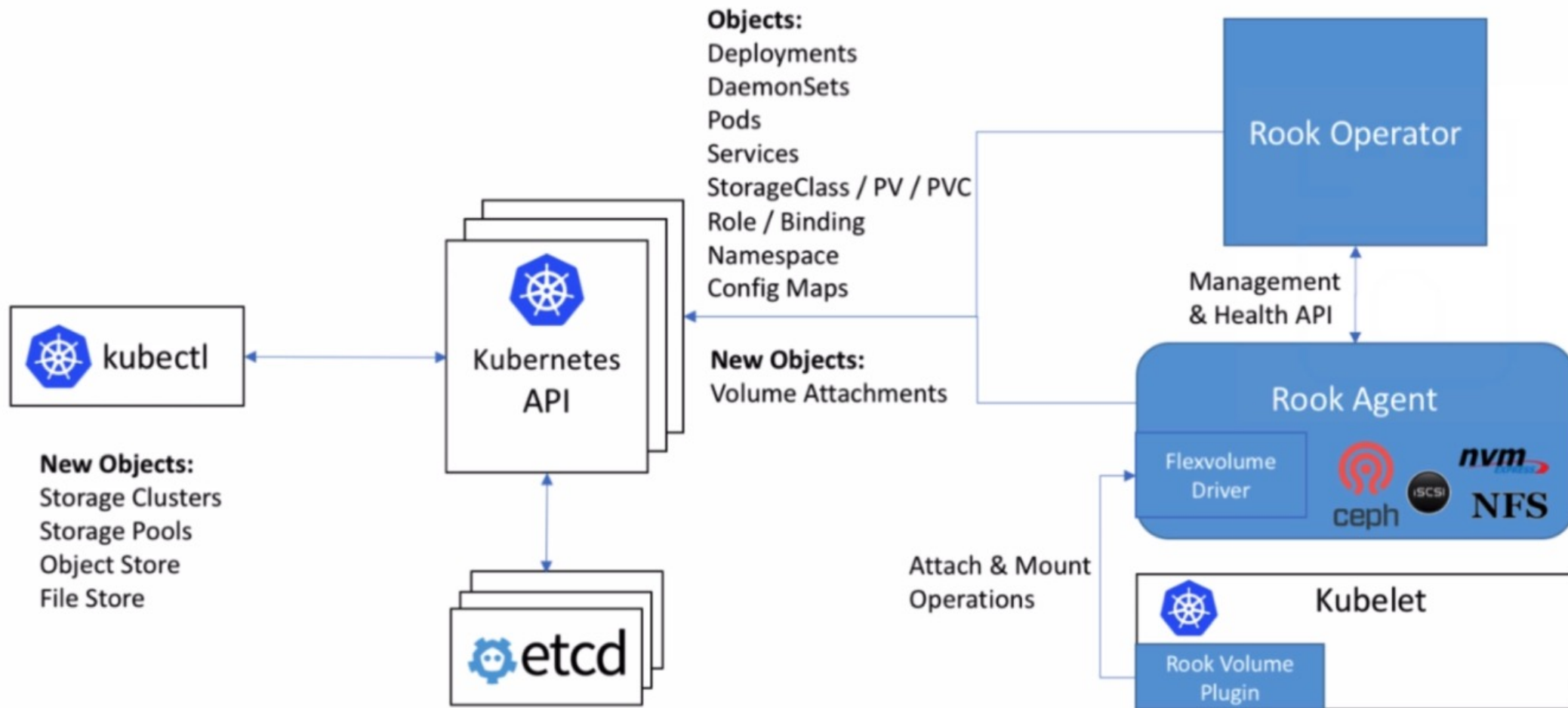
# StatefulSet

- Utilisé pour la gestion d'applications Stateful
  - Ex: cluster de base de données
- Management d'un ensemble de Pods non interchangeables
- Chaque Pod à
  - un nom constant
  - un identifiant réseau persistant
  - son propre stockage
- Mise en place / suppression de l'application selon un ordre défini

# StatefulSet : exemple d'un cluster MySQL



# Design





# Exemple : architecture Ceph

