# Ceph Pools et RBD

Le stockage en mode bloc

# Les pools

- Ceph stocke les données dans des pools qui sont des groupes logiques permettant de découper un cluster Ceph en plusieurs zones pour le stockage des objets.
- Résilience : Chaque pool peut être configuré pour être répliqué (« replicated ») ou en codage à effacement (« erasure coded »). Pour les pools répliquées, vous devez définir le nombre de répliques dont disposera chaque objet de données. Avec le codage à effacement, vous devez définir les valeurs de k et m, k correspondant au nombre de tranches de données et m au nombre de tranches de codage.
- Groupes de placement : vous pouvez définir le nombre de groupes de placement pour un pool.
  Nombre de placement groups total = (OSD * 100) / Nombre de réplicats
- Règles CRUSH : Règle de placement des données et des réplicats. Vous pouvez créer une règle CRUSH personnalisée en fonction des classes des OSD et du domaine de défaillance (OSD, Host,Room,Dc)

# Gestion des pools

Création : ceph osd pool create POOL_NAME[<pg_num:int>] [<pgp_num:int>] [<pool_type:replicated|erasure>] [<autoscale_mode:on|off|warn>] [<target_size_bytes:int>]
ceph osd pool ls detail
ceph df
ceph osd pool rename CURRENTPOOLNAME NEWPOOLNAME

Suppression :

ceph tell mon.* injectargs --mon-allow-pool-delete=true

ceph osd pool delete poolname poolname --yes-i-really-really-mean-it

ceph tell mon.* injectargs --mon-allow-pool-delete=false

# RADOS object store

rados -p <pool-name> put <object-name> <file>

rados -p <pool-name> get <object-name> <file>

rados -p <pool-name> ls

rados -p <pool-name> rm <object-name>

rados df

rados bench

ceph osd map <pool-name> <object-name>

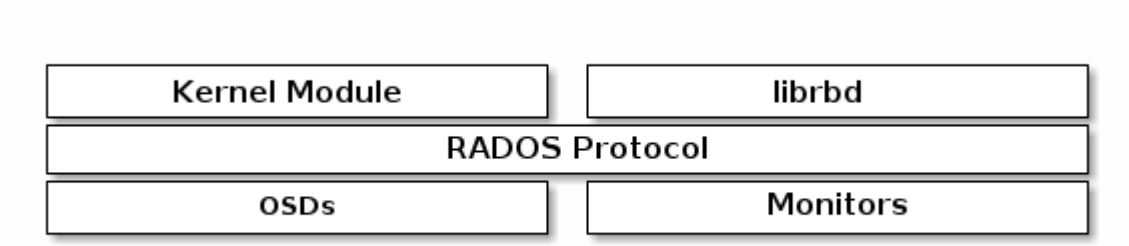
ceph osd find osd.id

Exemple :

ceph osd pool create pool1 16 16
dd if=/dev/zero of=/tmp/test bs=10M count=1
rados -p pool1 put object1 /tmp/test
rados -p pool1 ls
object1
ceph osd map pool1 object1
osdmap e15 pool ‘pool1’ (3) object ‘object1’ -> pg 3.bac5debc (3.c) -> up ([2,1,0], p2) acting ([2,1,0], p2)

# Le système de stockage en mode bloc

- Permet de monter une image RBD (RADOS Block Devices) comme un périphérique de type bloc.
- Ceph entrelace et réplique les données à travers  un pool sur le cluster automatiquement.
- Ceph RBD est généralement utilisé pour les disques de machines virtuelles.



# Les commandes de base

rbd create --size 1G mypool/myimage
rbd create mypool/myimage --size=1G --data-pool myecpool
rbd -p mypool ls
rbd -p mypool trash ls
rbd -p mypool info myimage
rbd resize --size 4096 mypool/foo
rbd resize --size 2048 mypool/foo --allow-shrink
rbd rm mypool/myimage
rbd trash mv mypool/myimage
rbd trash restore mypool/myimage

# Utilisation avancée

- Gestion des snapshots
  rbd snap create prbd/foo@snapv1
  rbd snap ls prbd/foo
  rbd snap rm prbd/foo@snapv1
  rbd snap purge prbd/foo # supprime tous les snap de l'image
- Gestion des clones
  rbd snap protect prbd/foo@snapv1
  rbd clone prbd/foo@snapv1 prbd/fooclone
  rbd rm prbd/fooclone
  rbd snap unprotect prbd/foo@snapv1
- Accès depuis le client
  rbd device map prbd/fooclone
  rbd device unmap prbd/fooclone
  rbd device list

# rbd live migration

- Lors de la création d'un pool, il est nécessaire de définir celui-ci en mode réplication ou en mode erasure. Si vous devez changer de type de paramètre, il est nécessaire de migrer les données vers un autre pool avec un minimum de temps d'indisponibilité.
- Remarque: krbd ne supporte pas le live migration. ce mode fonctionne seulement avec librbd, kvm ou nbd-rbd
- Le processus de migration se déroule en trois étapes :
  - Préparer la migration : cela consiste à créer une nouvelle image cible (de destination) et de la lier à l'image source. l'image source sera ensuite en lecture seule et les écritures seront dirigées vers l'image de destination.
  - Exécuter la migration: il s'agit d'une opération en arrière-plan qui copie les blocs de l'image source vers l'image cible. il est possible de différer le lancement de cette étape pendant une période où il y a moins d'activité.
  - Terminer la migration : vous pouvez valider ou abandonner la migration une fois le processus de migration en arrière-plan terminé. La validation de la migration supprime les liens entre les images source et cible, et supprimera l'image source. L'abandon de la migration supprime les liens croisés et supprime l'image cible.
- Exemple :

```
rbd migration prepare prbd/foo prbd/foo2ec --data-pool prbdec
rbd migration execute prbd/foo2ec
rbd migration commit prbd/foo2ec
```

# rbd feature

- Différentes options:

Layering : La création de couches, ou superposition(layering), permet d'utiliser le clonage.
Striping : La segmentation des données sur plusieurs objets et contribue au parallélisme de la lecture/écriture
exclusive-lock : un client obtient un verrouillage sur l’image avant d'effectuer une écriture.
object-map : Les périphériques de bloc sont provisionnés dynamiquement, ce qui signifie qu'ils ne stockent que les données qui existent réellement.
fast-diff: Génère plus rapidement les différentiels entre les snapshots d'une image.
deep-flatten : créer une image à partir d’une image clonée pour être indépendant de l’image parente
journaling : enregistre toutes les modifications d'une image dans l'ordre où elles se produisent.

- Utilisation :

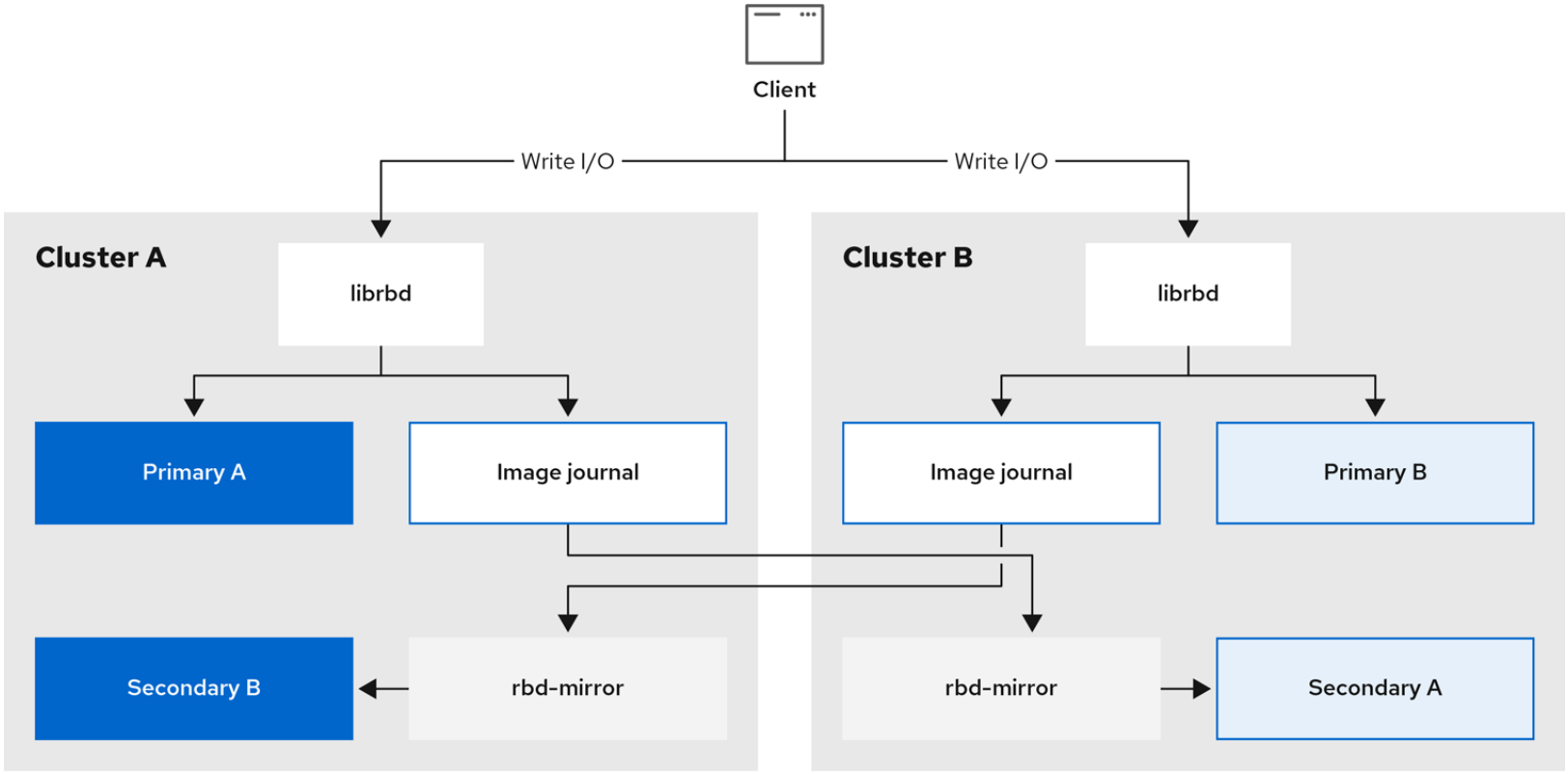
rbd feature enable mypool/image1 exclusive-lock

rbd feature disable mypool/image1 object-map

# rbd-mirroring

- Les images RBD peuvent être mises en miroir de manière asynchrone entre 2 clusters Ceph.
  - Mode basé sur un journal : Ce mode utilise la fonctionnalité de journalisation de l'image RBD afin de garantir une réplication ponctuelle, cohérente entre les clusters en cas de panne.
  - Mode basé sur des instantanés : Ce mode utilise des instantanés en miroir d'image RBD planifiés régulièrement pour répliquer des images RBD cohérentes entre 2 clusters
- Granularité :
  - Pool : Toutes les images  sont répliquées pour les pools en mirroring
  - Image : Uniquement une sélection d’images sont répliquées.

# Diagramme de fonctionnement




- Le service « rbd-mirror » est connecté aux deux clusters : Le service a besoin d'un compte sur chacun des clusters
- Il utilise le journal de l'image pour synchroniser uniquement les modifications
- Les données transitent par le service : Attention à la bande passante

# rbd-mirror, installation

- Token
  - rbd mirror pool peer bootstrap create [--site-name local-site-name] pool-name
  - bd mirror pool peer bootstrap import [--site-name local-site-name] [--direction rx-only or rx-tx] pool-name token-path
- rbd mirror pool enable [--site-name {local-site-name}] {pool-name} {mode}
- rbd mirror pool info POOL_NAME
- Remarques :
  - En mode journal il est nécessaire d’utiliser les features “exclusive-lock,journaling” avec rbd feature enable data/image1 exclusive-lock, journaling
  - Ce mode est compatible avec qemu, mais pas avec les montages rbd en mode kernel.

# rbd-mirror, utilisation

- Status : rbd mirror image status {pool-name}/{image-name} ou rbd mirror pool status {pool-name}
- Basculement :
  - rbd mirror image demote {pool-name}/{image-name} ou rbd mirror pool demote {pool-name}
  - rbd mirror image promote [--force] {pool-name}/{image-name} ou rbd mirror pool promote [--force] {pool-name}
- Resync : rbd mirror image resync {pool-name}/{image-name}
- Gestion des snapshots: ils sont gérés par le service rbd-mirror, et sont supprimés automatiquement. Il n'est pas possible de définir une période de rétention par pool ou par images
  - Utilisation du programmateur
    - rbd mirror snapshot schedule add
    - rbd mirror snapshot schedule list (... ls)
    - rbd mirror snapshot schedule remove (... rm)
    - rbd mirror snapshot schedule status

# Documentation

- https://docs.ceph.com/en/latest/rados/operations/pools/
- https://docs.ceph.com/en/latest/rbd/
- https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html/storage_strategies_guide/index
- https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html/block_device_guide/index
- https://github.com/sg4r/cephlab/blob/main/cephrbd.md