

# CEPH

## C'est quoi ?

Ceph est une solution de stockage distribué et open-source développée pour mettre en place un stockage hautement évolutif, redondant et performant pour des systèmes de fichiers tels que les blocs de données et les objets. L'installation de CEPH peut se faire sur de nombreux appareils qui va d'un rack de serveurs avec 86 disques à un simple Raspberry Pi. Il n'y pas de limite théorique de stockage pour un cluster. Par exemple, il y a 7 ans, le CERN a testé 11 000 disques sur un cluster CEPH.

## Historique

### 2004

À la base, Ceph a été développé par Sage Weil dans le cadre de sa thèse de doctorat à l'Université de Californie. L'objectif principal était de créer un système de stockage distribué permettant d'avoir des performances élevées, une haute disponibilité et une forte évolutivité.

### 2010

Le client CEPH pour les systèmes Block et Fichier est désormais disponible dans le noyau Linux. CEPH devient ainsi bien plus accessible. Il suffit juste d'installer un simple paquet pour pouvoir l'utiliser.

### 2012

Sage Weil fonde Inktank Storage pour offrir un support commercial et de la formation sur CEPH. Le projet se professionnalise et le rend viable pour les entreprises qui cherchent un support commercial.

### 2014

Red Hat, une entreprise majeure dans l'open-source, rachète Inktank Storage pour 175 millions de dollars.

### 2018

Il y a la création de la fondation CEPH. Elle dépend de la fondation Linux. Parmi les différents membres qui la composent, on y trouve des fournisseurs de stockage : Samsung, Intel, Western Digital, des fournisseurs de cloud : Digital Ocean, OVH Cloud et des entreprises de support/conseil : Red Hat ...

## Les composants

### *CEPH Monitor*

Le CEPH Monitor surveille l'état général du serveur en gardant en mémoire différents maps (monitor, manager, OSD, MDS, CRUSH). Ces maps permettent aux daemons CEPH de se coordonner entre eux. Son nom peut porter à confusion : Il ne fait pas de monitoring. Cette Tâche revient aux OSD. Le Monitor gère toute authentification entre le client et les daemons via un système de token. En générale, il est conseillé d'avoir 3-5 monitors par cluster.

### *CEPH Manager*

Le CEPH Manager permet de surveiller/mesurer des statistiques (utilisation, performance, charge système) en temps réel des daemons. Les informations sont exposées/gérées par des modules python hébergées dans les managers. Il y a prise en charge des APIs REST. Un CEPH

Dashboard y est mis à disposition. Des plugins sont disponibles. Par exemple, un plugin permet de prédire quand les disques ne vont plus fonctionner en se basant sur les informations recueillies. Pour garantir une haute disponibilité, il est utile d'avoir au moins deux managers.

### *CEPH Object Storage Daemon*

Il gère directement l'entière du stockage des disques. Il fait du monitoring : Il surveille si les autres fonctionnent bien et si dans le cas où un OSD dysfonctionne, les autres autour vont l'expulser du cluster. Pour avoir une certaine résilience des données, il effectue de la réplication ou de l'effacement codé au choix de l'utilisateur. Normalement, on attribue un OSD par disque.

La réplication est similaire à du RAID 1. L'entière des données d'un disque est répliquée dans les autres. Cela demande un coût en stockage élevé. La vitesse est limitée au disque le plus lent. Cela garde une bonne performance et une certaine simplicité. Attention, à avoir au moins 3 disques sinon avec seulement 2 disques, quand CEPH va détecter une incohérence dans les données, il ne pourra pas déterminer quel disque est juste et il va juste choisir arbitrairement un.

L'effacement codé ressemble à du RAID 6. Chaque donnée est divisée en  $(n \text{ disques} - 1)$  parties en plus du checksum qui sont réparties dans les disques. Grâce au checksum et aux différentes parties de la données réparties à travers les disques, si un disque tombe en panne, alors il est simple de recréer la partie perdue à partir du reste. Malheureusement, il y a un coût en performance parce qu'il est nécessaire de faire une requête pour chaque disque pour reconstituer la donnée souhaitée. Cette approche est plus complexe que la réplication.

### *RADOS*

Le RADOS est composé d'au moins un OSD et un Monitor. C'est le minimum pour avoir un cluster fonctionnel. Une fois qu'on a le RADOS, on peut utiliser les librairies Librados qui permettent d'accéder aux données CEPH avec différents langages (Python, Java, Rust, C ...). On peut désormais faire du Block qui donnent accès aux fonctionnalités suivantes :

- Snapshots (Copie d'un instant T d'un disque),
- Copy on Write (Exemple : Lancer 100 VM Ubuntu avec une seule image stockée)
- Mirroring (Synchronisation de toutes les écritures d'un cluster CEPH avec un autre)

### *CEPH Metadata Serveur*

Si on souhaite avoir un système de fichier (CephFS), il est nécessaire d'avoir un serveur de métadonnées. Avec beaucoup de RAM, le serveur va stocker un maximum de métadonnées. Côté client, des données seront mis en cache pour éviter de refaire des requêtes coûteuses si l'état du cluster n'a pas changé.

### *RADOS Gateway*

Si on veut de l'object storage (AWS S3), alors il faut un RADOS Gateway. Il permet d'agréger les clusters entre eux pour faire du multi zone. Par exemple, on éparpille des clusters à travers le monde pour qu'ils soient au plus proche des clients. Les clusters sont tous synchronisés. On souhaite juste distribuer du contenu aux plus proches des clients. Le daemon met à disposition un Gateway RESTful entre les applications et les clusters.

## Fonctionnement

### *Pool – Groupe de placement*

On ne peut pas stocker directement un objet dans le cluster. D'abord, il est nécessaire de créer un pool. Ce dernier représente un cas d'utilisation prévu. Puis, au sein de ce pool, on y définit des groupes de placement. En générale, il y en a une centaine par OSD. L'objet que l'on souhaite stocker est placé dans ce groupe de placement. Ces groupes permettent d'alléger la gestion des objets à l'OSD. Selon l'algorithme CRUSH, en fonction de la donnée voulue et de l'architecture du cluster, il va déterminer le bon OSD à communiquer directement pour y récupérer la donnée souhaitée.

### *CSI*

Le Container Storage Interface est une norme qui permet aux systèmes de stockage comme Ceph, Amazon EBS de s'intégrer de manière uniforme avec les orchestrateurs de conteneurs Kubernetes, Docker Swarm, ...

Parmi les fonctionnalités proposées

- Le provisionnement dynamique permet de créer automatiquement du stockage quand les applications en ont besoin.
- Monter et démonter des volumes sur les nœuds où les pods sont déployés
- Prendre des Snapshots des volumes
- Redimensionner en augmentant la taille des volumes existants
- Supprimer les volumes non utilisées pour libérer des ressources

## Hardware

### *Processeur CPU*

- **CEPH Métadonnée Serveur**, Le serveur est particulièrement gourmand en puissance du processeur. Ce dernier doit être à haute fréquence, mais ne nécessite pas un grand nombre de cœur pour satisfaire le serveur.
- **Monitor et Manager**, Ils ne nécessitent pas spécialement d'un CPU puissant. Un modeste leur suffit.
- **OSD**, Les nœuds OSD nécessitent une puissance de traitement suffisante pour exécuter le service RADOS, calculer le placement des données avec CRUSH et répliquer les données. Avec les disques NVMe, Ceph peut utiliser efficacement plusieurs cœurs par OSD.

### *Mémoire RAM*

- **Monitor et Manager**, 64 Go peut suffire pour un cluster moyen, sinon 128 Go pour des clusters plus grand
- **OSD**, La configuration par défaut de `osd_memory_target` est de 4 Go. Il est utile de prévoir une marge en plus pour l'OS et les tâches administratives, en allouant 8 Go par OSD utilisant BlueStore.
- **CEPH Métadonnée Serveur**, Il est recommandé de garder 1 Go. La consommation de mémoire dépend de la taille du cache configuré.

### *Stockage*

- **Disques Durs HDD**, Il est recommandé d'avoir un disque dédié pour l'OS et un pour chaque OSD.

- **Disques SSD**, Leur utilisation pour le Monitor, Manager et les pools va améliorer considérablement les performances. Faire attention de bien aligner les partitions des disques pour éviter des ralentissements au niveau des transferts de données.
- Concernant le système de fichier CephFS, la séparation du stockage des métadonnées à celle du contenu des fichiers améliore les performances.

## Configuration

### *Backend - BlueStore*

BlueStore est le backend de stockage par défaut de Ceph OSD, développé pour remplacer l'ancien backend Filestore. Il permet d'améliorer les performances, la fiabilité et l'efficacité du stockage des objets dans CEPH. Il possède plusieurs caractéristiques notamment le mécanisme de 'copy-on-write clone' et la vérification des données et métadonnées avec des checksums.

### *Sources, sections de configuration*

Les services/démons CEPH récupèrent leurs configurations à partir de différentes sources : Valeurs par défaut, Base de données de configuration, Fichiers de configuration locaux, Variables d'environnement, Arguments de ligne de commande et surcharges en temps réel définies par l'administrateur.

Les options configurables sont regroupées en sections pour spécifier leur portée : global (tous), mon (monitor), mgr (manager), osd (OSD), mds (serveur métadonnées) et clients (CephFS, Block, RGW).

La commande 'ceph config' permet de configurer le cluster.

### *Réseau*

Par défaut, CEPH utilise un seul réseau public pour toutes les communications. Cependant, pour avoir de meilleures performances dans les grands clusters, il est conseillé de configurer un réseau supplémentaire, appelé "cluster network", réservé aux opérations internes comme la réplication et la récupération des données. Cela permet d'isoler le trafic interne du cluster du trafic client.

### *Protocole de communication - Messenger V2*

Le protocole Messenger V2 permet aux démons et clients CEPH de communiquer entre eux de manière sécurisée et flexible. Il possède quelques caractéristiques comme une amélioration de l'encapsulation des authentifications (intégration future de Kerberos) et une meilleure annonce/négociation des fonctionnalités au début de la communication. Dans la V2, les démons écoutent sur le port 3300.

Le mode CRC est utilisé par défaut. Il fournit une authentification lors du début de la communication et réalise une vérification d'intégrité via CRC32C. Pour se protéger d'attaque 'man-in-the-middle', utiliser en plus le mode sécurisé permettra de chiffrer les données qui y transitent.

## *Protocole d'authentification – CephX*

Le protocole CephX est le mécanisme d'authentification cryptographique activé par défaut pour sécuriser les communications entre les clients et les services du cluster. Le monitor CEPH remplit le rôle d'autorité d'authentification en possédant une base de données de secrets. Un token est remis après authentification auprès du monitor qui permettra de réaliser des actions avec les services CEPH. La désactivation de CephX permet d'économiser en puissance de calcul, mais est vivement déconseillé.

## Outils de déploiement

### Cephadm

Cephadm est un outil de déploiement natif qui permet de déployer directement sur des machines virtuelles ou physiques sans avoir Kubernetes. La gestion des services sur plusieurs nœuds passe par SSH et les containers (Docker). Il permet d'automatiser le provisionnement et mise à jour du cluster. Il est idéal pour un déploiement CEPH natif sans Kubernetes. La perte et récupération d'un nœud se repose sur CEPH lui-même pour détecter la perte et restaurer les données. Sur des forums, le déploiement avec Cephadm est plus simple et direct que celui avec Rook. Cependant, la récupération d'un nœud défaillant est plus difficile.

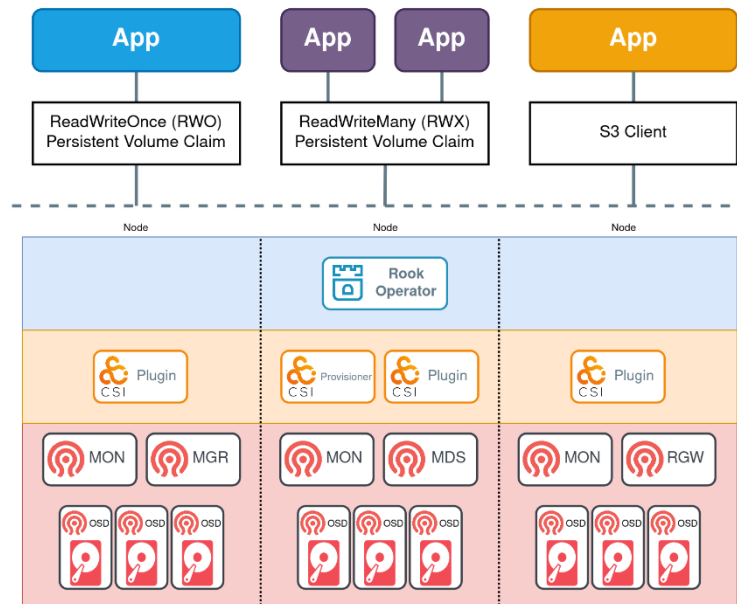
### ROOK

#### *C'est quoi ?*

Rook est un orchestrateur de stockage cloud natif qui permet de gérer et déployer CEPH sur Kubernetes de manière automatisée et simplifiée. Il agit comme un opérateur de stockage pour Kubernetes. Il permet de simplifier le déploiement, la configuration de CEPH, détecter / récupérer des défaillances et gérer des volumes persistants sur Kubernetes.

L'opérateur a été déclaré comme étant stable depuis décembre 2018 avec la version 0.9. Cela résulte en une solution prête pour la production depuis quelques années. Le projet Rook est reconnu comme 'Graduated' par la Cloud Native Computing Foundation (CNCF) attestant un haut niveau de maturité, croissance et d'adoption.

## Rook Architecture



### EN COURS

#### Différences

La principale différence entre Cephadm et Rook réside l'environnement d'exécution. Cephadm installe CEPH de manière native alors que Rook transforme CEPH en un service Kubernetes. La gestion des services se fait avec la commande 'ceph orch' dans Cephadm. Dans Rook, les CRD permettent de gérer services CEPH qui sont dans pods Kubernetes.

Avec Cephadm, la détection d'une panne est réalisée par le monitor CEPH. La recreation des OSD et l'ajout d'un nouveau nœud doit se faire manuellement avec les commandes dédiées. Avec Rook, la détection de la panne se fait via le Node Controller. La recreation des OSD, ainsi que l'ajout d'un nouveau nœud sont automatisés.

Le choix se repose essentiellement entre un contrôle élargi malgré une configuration plus manuelle et une automatisation maximale des services dans un environnement Kubernetes déjà en place mais un contrôle légèrement amoindri.

## Kubernetes

### Kubernetes (K8s)

K8s est la plateforme d'orchestration de conteneurs la plus populaire, connue pour sa scalabilité et flexibilité. Elle est adaptée aux déploiements sur site et dans le cloud, et convient aux applications conteneurisées à grande échelle. K8s donne accès à des fonctionnalités avancées telles que la mise à l'échelle automatique, l'auto-guérison et les mises à jour progressives. Il est idéal pour les environnements de production complexes.

Dans le cas d'un déploiement sur des Raspberry Pi 3, K8s est trop lourd. Il demande beaucoup trop de ressources en RAM et CPU. Son installation et sa maintenance sont plus complexes par rapport aux deux autres versions Kubernetes.

## K3s

Développé par Rancher Labs, K3s est une version allégée de Kubernetes, conçue pour les environnements aux ressources limitées, tels que l'edge computing et l'IOT. K3s est facile à déployer et à gérer. Il garde l'essentiel des fonctionnalités de K8s tout en réduisant la complexité et les exigences matérielles. Il est optimisé pour ARM. Il supporte les Custom Resource Definitions nécessaires au fonctionnement de Rook. K3s a été certifié par la CNCF comme projet 'Sandbox' et cela assure une certaine fiabilité et sécurité.

Probablement le meilleur choix dans le cadre d'une utilisation avec un Raspberry Pi 3 : Il est très léger (moins de 100 MO). Il a été optimisé pour l'architecture ARM. Il est simple à installer et gérer. Il possède moins de dépendances que K8s. Il correspond bien à une utilisation pour un petit cluster. La plupart des exemples de déploiements de CEPH dans des Raspberry trouvés sur le web ont choisi K3s.

## K0s

Développée par Google, K0s est une distribution Kubernetes légère et sécurisée, adaptée aux environnements aux ressources limitées. Elle possède une architecture monolithique composée d'un seul binaire. Cela facilite le déploiement et réduit la consommation de ressources. Cette plateforme conteneur native a été prévue pour lancer des applications conteneurisées dans un environnement de calcul distribué.

K0s constitue comme une alternative viable à K3s. Il n'a pas de dépendances système du fait de son architecture. Il supporte l'architecture ARM du Raspberry Pi 3. Il est facile à mettre à jour et sécuriser. Sa taille est plus légère que celle de K3s. Dans sa documentation officielle, on y trouve un guide sur l'installation de CEPH avec Rook et K0s.

## Choix final – K3s

K3s reste l'option la plus populaire en ce qui concerne le déploiement dans un environnement aux ressources limitées. Malgré qu'il soit un peu moins léger par rapport à K0s, son adoption est plus répandue. Par conséquent, il a été bien éprouvé dans ce cas d'utilisation et dispose d'une communauté active avec un support fiable. Il est aussi maintenu par Rancher.

## Sources

### *CEPH ROOK*

<https://docs.ceph.com/en/reef/start/>

<https://rook.io/docs/rook/latest-release/Getting-Started/intro/>

[https://www.youtube.com/watch?v=VVsv3Ca\\_Y70](https://www.youtube.com/watch?v=VVsv3Ca_Y70)

### *Kubernetes*

<https://k3s.io/>

<https://k0sproject.io/>

<https://www.nops.io/blog/k0s-vs-k3s-vs-k8s/>

[https://www.reddit.com/r/kubernetes/comments/1i5n75f/anyone\\_using\\_k3smicrok8sk0s\\_in\\_production/](https://www.reddit.com/r/kubernetes/comments/1i5n75f/anyone_using_k3smicrok8sk0s_in_production/)