



```
FROM employees, WITH ORDINALITY  
    WHERE empName IN  
        SELECT DISTINCT empName  
        FROM population  
        WHERE Country = "TH"  
    AND empSalary >=  
        SELECT AVG(salary)  
        FROM employees
```

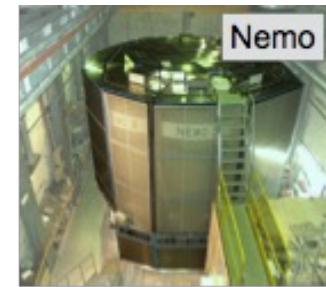
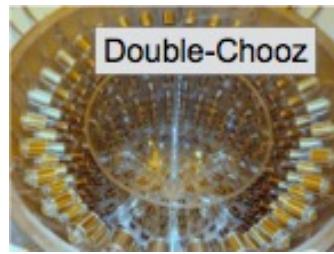
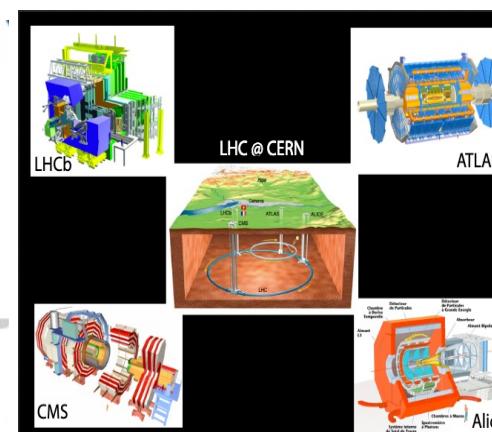
Programme

- Who am I ?
- Back to the future
- Les systèmes de fichiers
- Le stockage objet
- Les bases de données
- Les bases de données orientées Graphe
- Les bases de données orientées Document

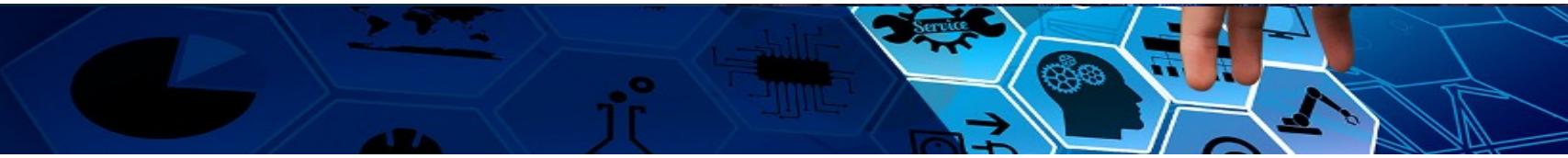
**WHO
AM I**



Les expériences IN2P3



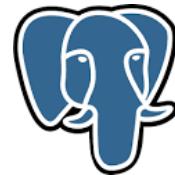
Qui suis je ?



• Mariadb



• PostgreSQL



• MySQL



• Oracle

ORACLE®

• ElasticSearch



• MongoDB



• Neo4J



Plus de 1 000 bases de données :
Volumétrie > 50 To
32 serveurs de bases de données



Animateur du réseau de Base De Données (rBDD)

Objectif : Fédérer, développer et partager les compétences autour des bases de données

Un des 26 réseaux soutenu par la Mission pour les Initiatives Transverses et Interdisciplinaires du CNRS.

Le réseau (> 750 membres) :



Action Nationale de formation 2024

- UML

Action Nationale de formation 2025

- OpenSearch et vos données

Action Nationale de formation 2026

- Introduction au NoSQL
- PostgreSQL administration



Webinaires 2024 - 2025

- TemBoard
- PostgREST
- Découvrir les fondements de l'IA
- L'IA et les bases de données



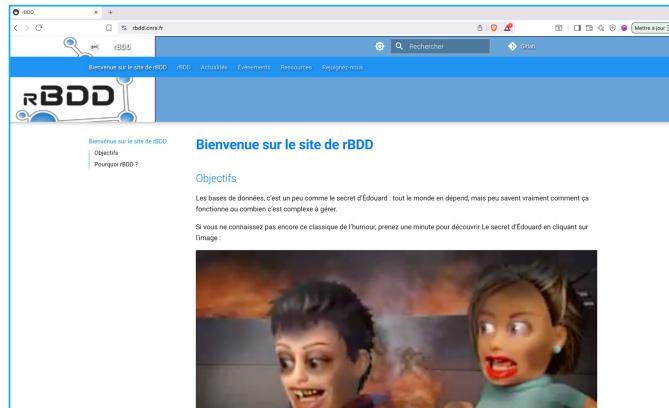
Groupe de Travail

- Atelier des données
- Observabilité (RESINFO + DEVLOG)
- Sauvegarde

Animateur du réseau de Base De Données (rBDD)

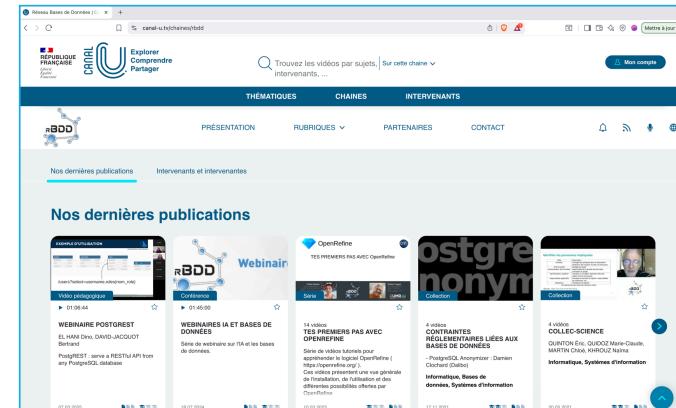
Vous êtes intéressé(e) par les bases de données, leur gestion, leurs usages dans la recherche ou les infrastructures ?
Vous avez envie de contribuer à une dynamique collective et porter des projets utiles à la communauté ?

N'attendez plus rejoignez le COPIL rBDD : rbdd-cp@services.cnrs.fr



The screenshot shows the homepage of the rBDD website. The header features the rBDD logo and navigation links for "Accueil", "Actualités", "Événements", "Ressources", and "Rejoignez-nous". The main content area has a blue header "Bienvenue sur le site de rBDD" and a section titled "Objectifs" with a sub-section "Pourquoi rBDD ?". It includes a cartoon illustration of two people looking surprised.

<https://rbdd.cnrs.fr/>



The screenshot shows the "Nos dernières publications" section of the Canal-U channel page for rBDD. It displays a grid of video thumbnails with titles like "WEBINAIRE POSTGRES", "TECHNIQUES POUR FAIRE DES API", and "TECHNIQUES POUR FAIRE DES API". Each thumbnail includes a play button and a timestamp.

<https://www.canal-u.tv/chaines/rbdd>

Pas encore prêt(e) à vous impliquer dans le COPIL ?

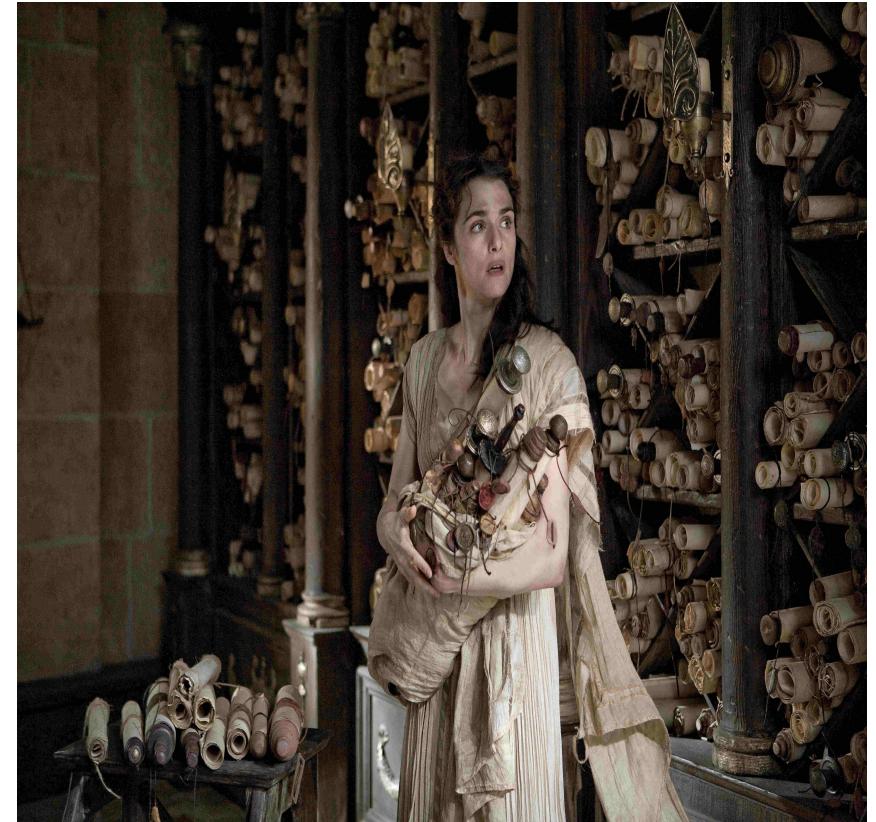
Vous pouvez aussi **vous inscrire à la mailing list** du réseau pour suivre nos actualités, recevoir les appels à projets et être informé(e) de nos actions : <https://rbdd.cnrs.fr/rejoignez-nous/>



BACK TO THE FUTURE

Les premiers systèmes de stockage

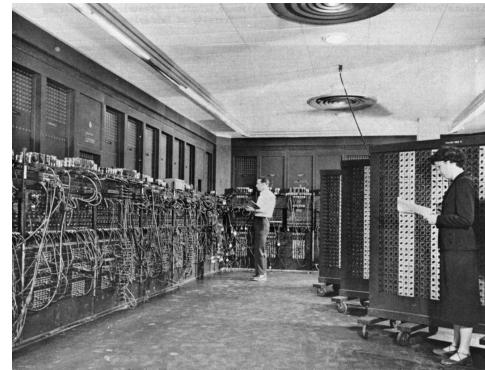
- Centraliser les données
- Stocker les données
- Organiser les données pour faciliter la recherche



Les défis technologiques

Le premier calculateur est né en 1946

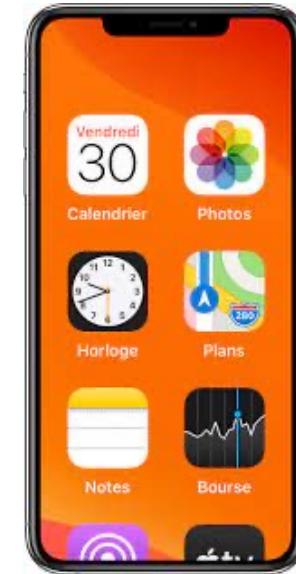
- ENIAC : *Electronic Numerical Integrator And Computer*
- 400 flops
- 167 m²
- 30 Tonnes



1955 : Stockage par bande magnétique



Et votre smartphone ???

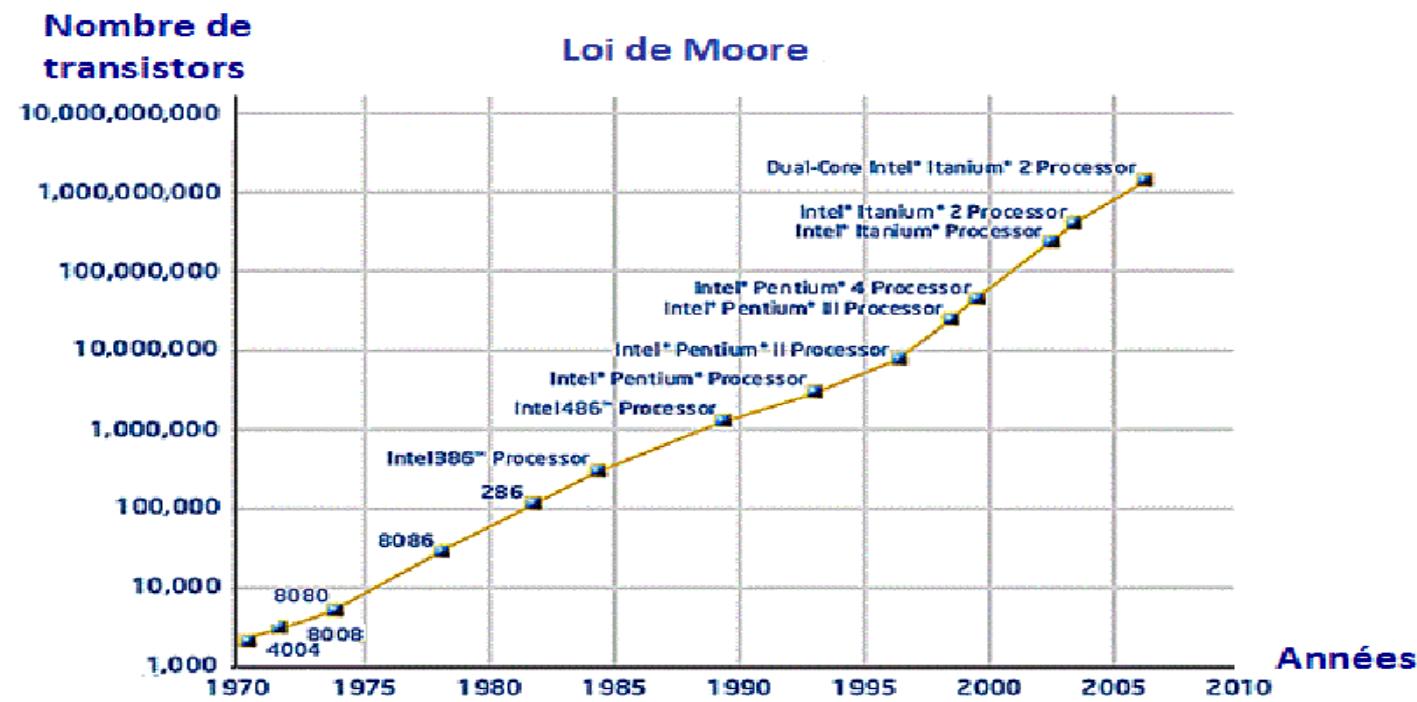


Smartphone 2022 :

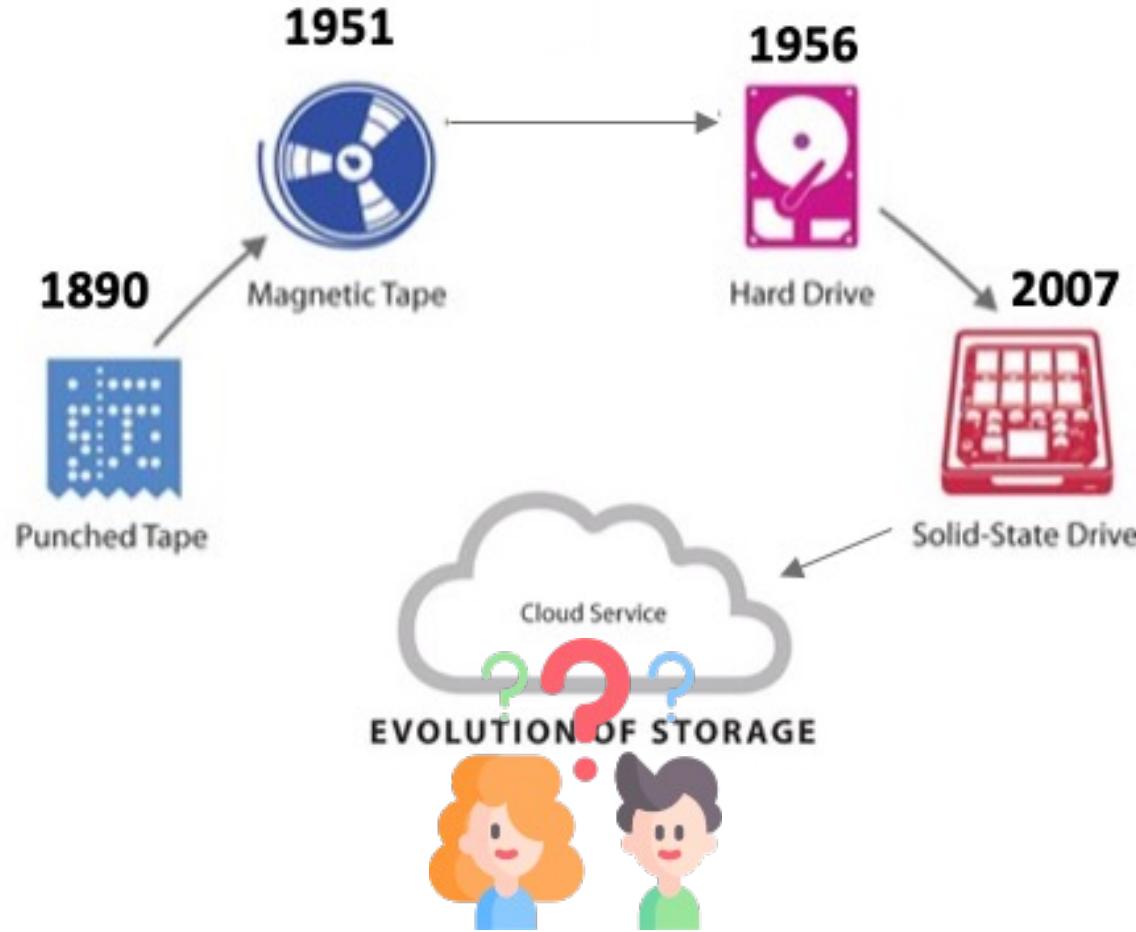
- Stockage 1To
- 400 Gflops

L'évolution matérielle

Loi de MOORE : Le nombre de transistors sur une puce de silicium double tous les 18 à 24 mois, ce qui entraîne une augmentation exponentielle de la puissance de calcul à coût équivalent.

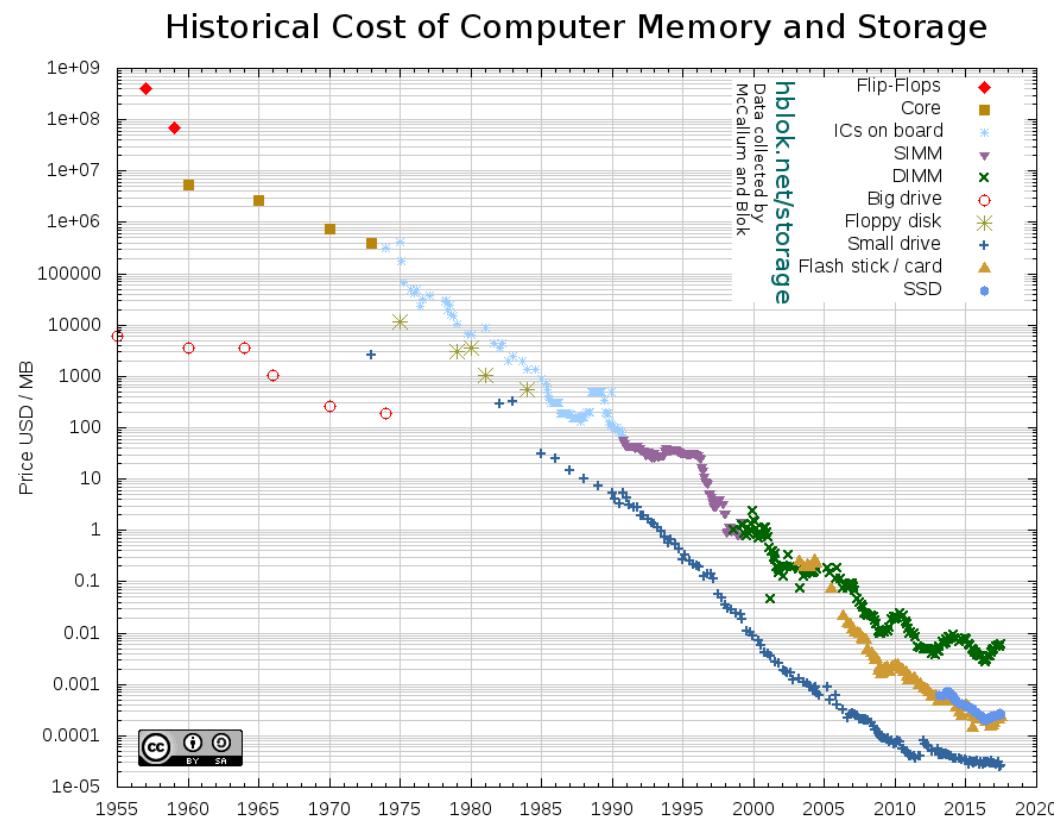


L'évolution matérielle



L'évolution matérielle

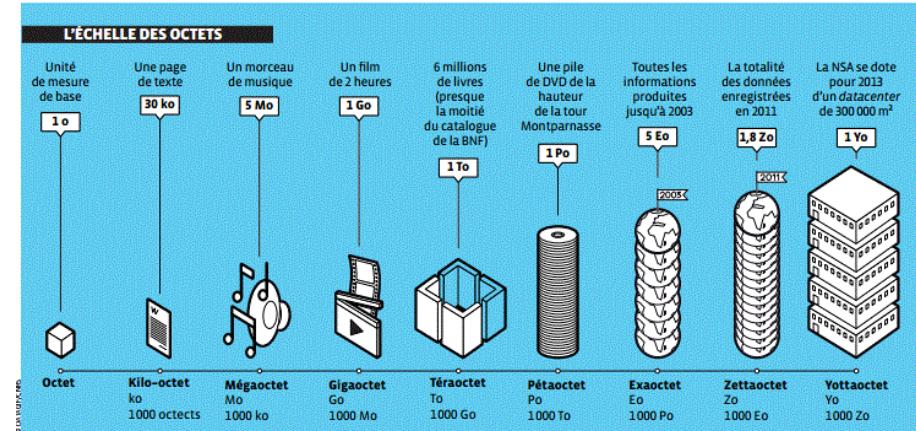
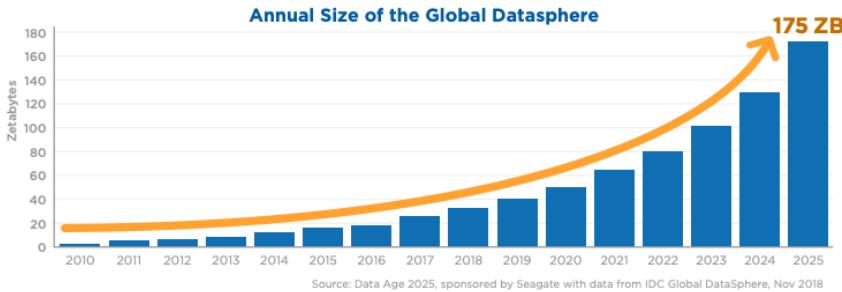
- A partir des années 2000, le matériel devient accessible à tous.



L'évolution matérielle

Des volumétries astronomiques sont en jeux :

Figure 1 – Annual Size of the Global Datasphere



Pourquoi stocker autant de données ?

- Le savoir est dans les données et « Le savoir, c'est le pouvoir » (Dan Abnett).
- Aider les compagnies / la science à mieux comprendre leur métier / notre univers.
- A mieux comprendre les clients.
- Découvrir de nouvelles logiques métiers.
- Prendre de rapides et de bonnes décisions.
- En 2019, les données ont une valeur non négligeable.
- Pour certains, la croissance des données de l'entreprise est un facteur de développement.



Graphisme et illustration : Clément Barbé

Portes ouvertes du Centre de Calcul de l'IN2P3

★ Au programme de 10h à 18h :

- Visites guidées du Centre de Calcul
- Conférences sur l'histoire des réseaux informatiques
- Découverte du musée de l'informatique
- Stands pédagogiques et animations autour de la donnée scientifique et du calcul haute performance

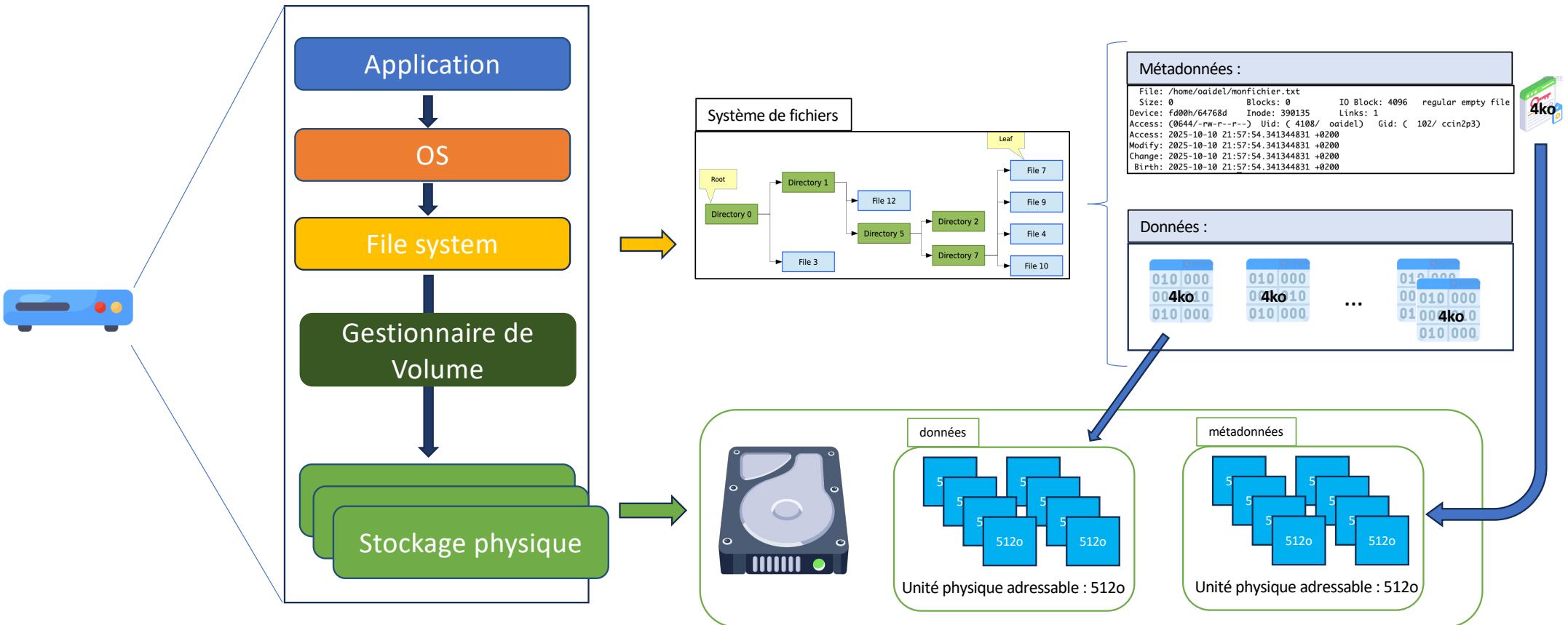
★ Entrée gratuite & Food Truck



Les systèmes de fichiers



Les systèmes de fichiers locaux



Les systèmes de fichiers locaux

❑ Métadonnées & structures

- Sans métadonnées, le système de fichiers ne saurait pas retrouver ou gérer un fichier.
- Un fichier contient les données, les métadonnées décrivent : son nom, emplacement, droits, dates, etc.
- La taille des blocs du volume varie selon le système de fichiers (4K, 8K...).

❑ Volumes logiques et gestion du stockage

- Un volume logique peut être construit à partir de plusieurs disques ou partitions.
- Il peut être agrandi à chaud avec des outils comme ZFS ou LVM.
- La tolérance aux pannes peut être assurée au niveau du gestionnaire de volume (ex : RAID logiciel).

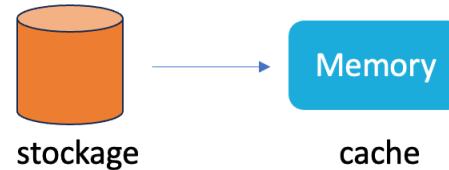
❑ Snapshots & sauvegardes

- Certains systèmes de fichiers (ex : ZFS, Btrfs) proposent nativement des snapshots.

❑ Bonnes pratiques pour les applications

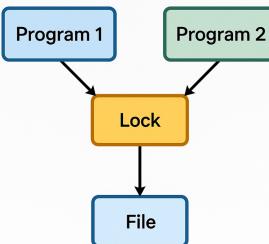
- Pour que votre application reste indépendante du système de fichiers (ext4, ZFS, NTFS...), utilisez des bibliothèques comme pathlib en Python.
- Pour assurer la portabilité entre systèmes Unix (Linux, BSD, macOS...), privilégiez les bibliothèques respectant le standard POSIX.

Les systèmes de fichiers locaux



- Sur la plupart des systèmes d'exploitation, une copie non persistante des données récemment utilisées est conservée en mémoire (RAM).
- Le cache accélère les lectures répétées.
- Les petites écritures fréquentes sont coûteuses pour le disque (surtout SSD ou disques mécaniques). Le système les regroupe en mémoire puis les écrit en bloc optimisé (write-back).
- Le système anticipe les lectures séquentielles et précharge les blocs suivants (read-ahead) .
- L'utilisateur ou l'administrateur a peu de moyens directs pour forcer la manière dont le système gère le cache.

Les systèmes de fichiers locaux



- ❑ Dans un système multi-tâche (comme Linux), plusieurs processus peuvent lire/écrire simultanément sur un même fichier. Cela peut poser problème si ces accès ne sont pas coordonnés, surtout en cas d'écriture.
- ❑ Si plusieurs programmes modifient un même fichier en même temps et sans précaution, les écritures peuvent écraser, intercaler ou corrompre les données des autres.
- ❑ Un verrou est un mécanisme de synchronisation qui bloque temporairement l'accès au fichier pour les autres processus.
 - verrou en lecture (read lock) : plusieurs lecteurs autorisés, pas d'écrivain.
 - verrou en écriture (write lock) : exclusif, un seul processus peut écrire (et lire).
- ❑ Les processus doivent attendre leur tour pour accéder au fichier (selon le type de verrou), assurant une cohérence des données.

Les systèmes de fichiers partagés (non distribué)

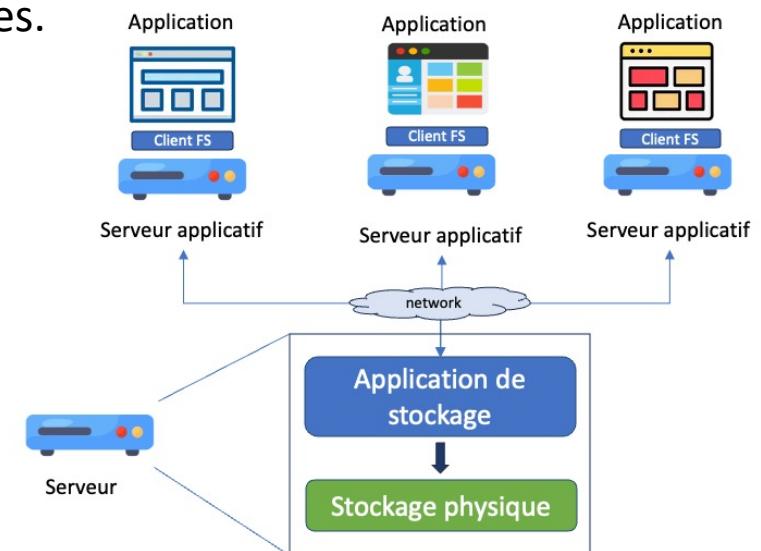
- ❑ Un système de fichiers partagé est un espace de stockage accessible simultanément depuis plusieurs machines via le réseau, comme s'il s'agissait d'un disque local.
- ❑ Il permet d'étendre l'accès aux fichiers au-delà des limites physiques d'une seule machine, sans avoir à copier les données.

❑ Accès transparent

- Les fichiers sont accessibles via les API POSIX standard (open(), read(), write(), etc.).
- Les programmes n'ont pas besoin d'être modifiés selon l'emplacement des données.

❑ Fonctionnalités typiques

- In-place updates : modification d'un fichier sans le recopier entièrement
- Exécution de programmes : exécution d'un script ou binaire stocké sur le FS
- Lectures/écritures partielles : accès à des fragments de fichiers



Les systèmes de fichiers partagés (non distribué)

❑ Gestion des utilisateurs

- Nécessite une identification cohérente sur toutes les machines (UID/GID identiques).
- S'appuie souvent sur un service d'annuaire : LDAP, NIS...
- Les droits POSIX (chmod, chown, umask) s'appliquent de manière uniforme.
- Permet une comptabilité fine des accès (qui lit/écrit quoi).

❑ Performance

- Toutes les entrées/sorties passent par le réseau : sensible à la latence et à la bande passante.
- Le serveur central peut devenir un goulot d'étranglement.
- Le client utilise des caches pour accélérer les lectures, écritures et accès aux métadonnées. Cela améliore les performances, mais peut introduire des délais de synchronisation, ce qui rend la cohérence entre clients plus complexe à garantir.

❑ Cohérence et verrous

- La gestion des accès concurrents est assurée par le serveur .
- Les opérations sur les fichiers passent par le réseau vers le serveur.
- Les latences réseau et les risques de déconnexion complexifient la gestion des verrous.

Les systèmes de fichiers partagés (non distribué)

❑ Tolérance aux pannes

- Pas de redondance native
- Point de défaillance unique (SPOF) : si le serveur tombe, tout le FS devient indisponible.

❑ Pré-requis

- Installation du client
- Mettre à jour les ACL réseaux
- L'adresse IP du client doit être autorisée
- Le mapping des utilisateurs (UID/GID) doit être cohérent (NIS)

Système de fichiers partagé distribué

Scalabilité horizontale

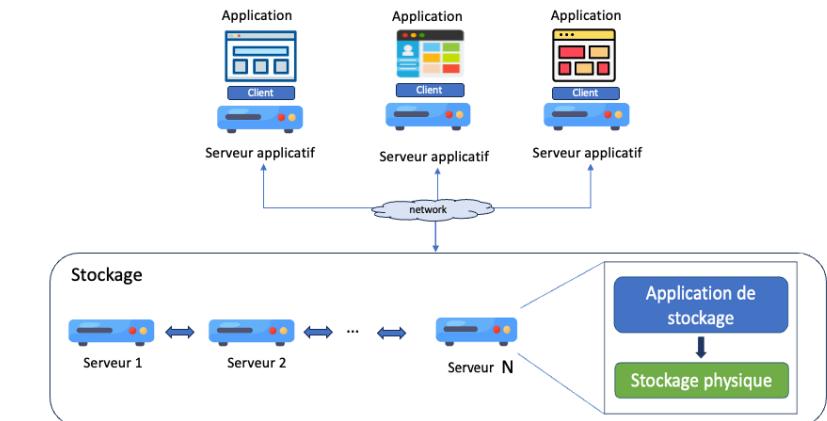
- Ajout de nœuds = augmentation automatique de la capacité et des performances
- Capacité de stockage & bande passante évolutives

Accès parallèle optimisé

- Plusieurs serveurs répondent **simultanément** aux requêtes
- Très bon débit (**IOPS / bande passante**), idéal pour l'HPC ou le Big Data

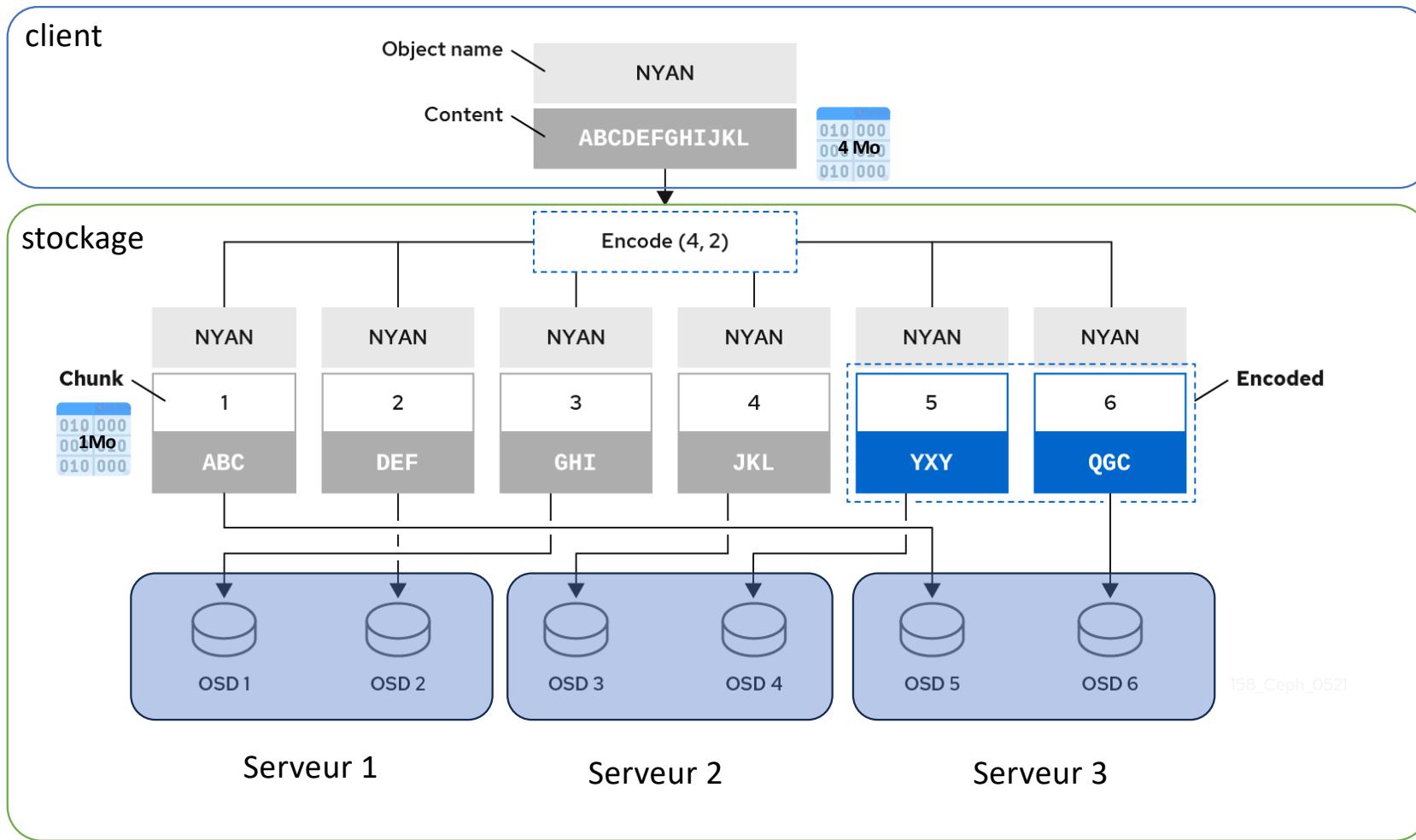
Tolérance aux pannes native

- Redondance via **réPLICATION** ou **erasure coding**
- Perte d'un nœud ou d'un disque = pas de perte de données ni d'indisponibilité



CephFS, GPFS, ...

Système de fichiers partagé distribué



Système de fichiers partagé distribué

❑ Administration et maintenance plus complexe

- Gestion fine des services (OSD, MDS, MON...)
- Surveillance, mises à jour, maintenance

❑ Cohérence et verrous distribués

- Coordination des accès entre clients
- Synchronisation obligatoire pour garantir la cohérence
- ! Cela limite l'extensibilité horizontale à grande échelle

❑ Pré-requis

- Installation du client (cephFS, GPFS ...)
- Mettre à jour les ACL réseaux
- **Adresse IP autorisée** (ACL réseau, /etc(exports))
- **Mapping UID/GID cohérent** entre les machines (via NIS, LDAP...)



Et le stockage objet
?

Le stockage objet



- Contrairement aux systèmes de fichiers classiques, il n'y a pas d'arborescence de fichiers, pas de dossiers, pas de chemin.
 - Chaque objet est **totalement indépendant**.
 - On l'identifie via une **clé unique (object ID ou nom)**.
 - Pas besoin de synchronisation entre clients donc une **scalabilité bien plus grande**
- Un objet est un fichier enrichi de métadonnées (sous forme de paires clé/valeur)
- On peut dire qu'un objet est une sorte de « **super fichier auto-descriptif** », sans dépendance hiérarchique.
- Concept introduit dans les années 90, puis **Amazon S3** en 2006 démocratise le modèle avec un accès via HTTP + REST API.
- Le **stockage objet** marque le passage à une **abstraction logicielle, scalable et cloud-native**.

Le stockage objet

□ Qui utilise le stockage objet ?



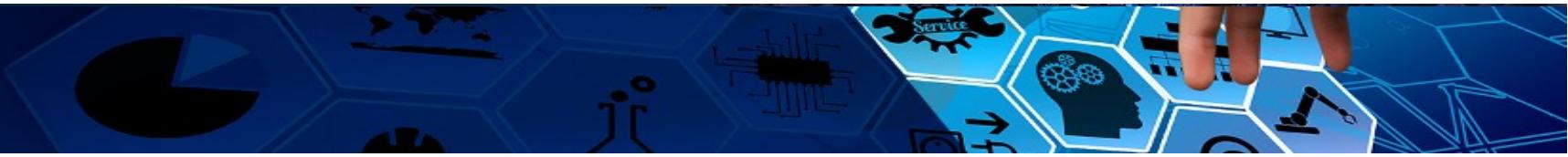
Dropbox



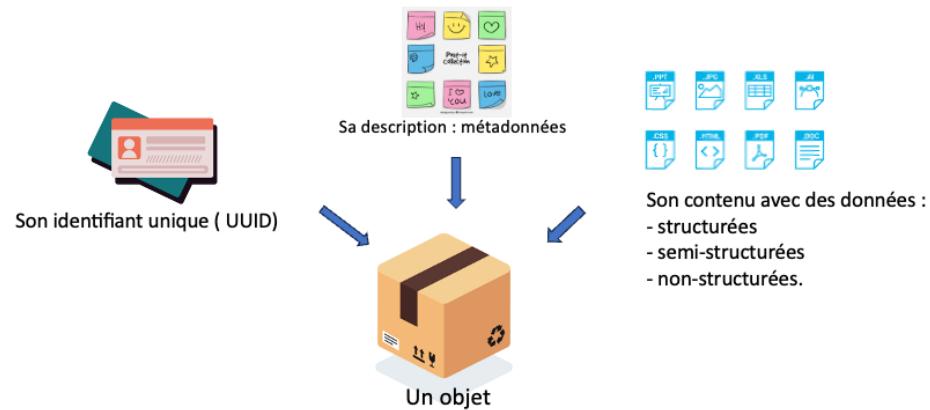
iCloud



Les objets

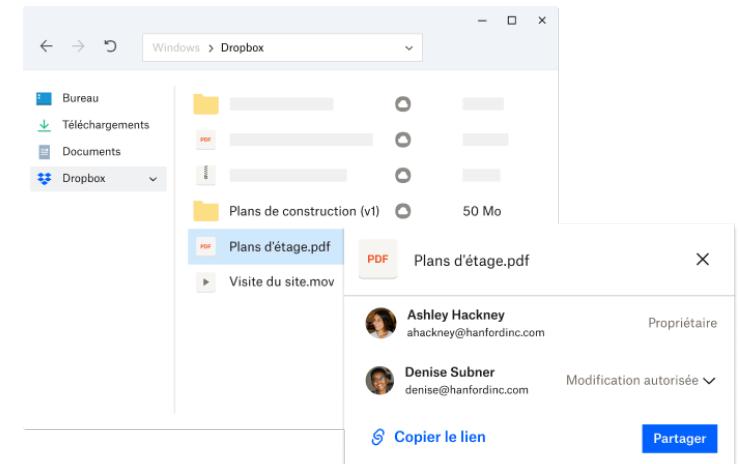
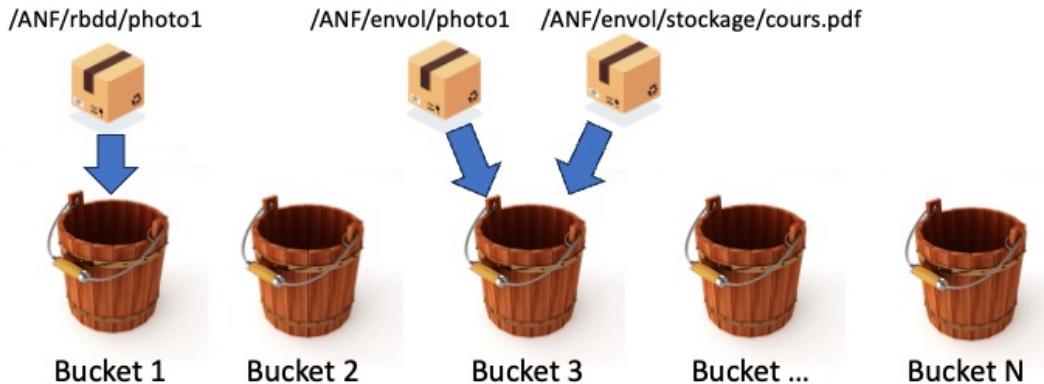


un objet est caractérisé par :



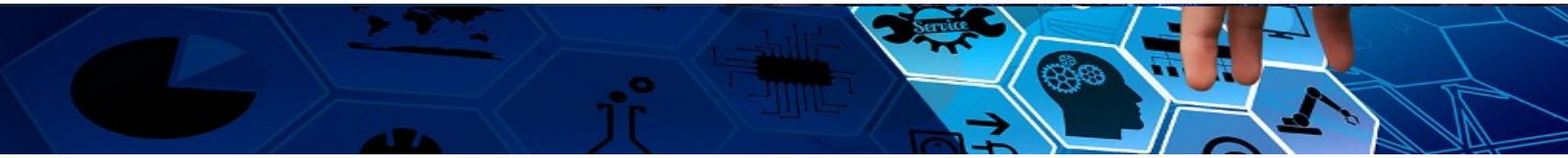
Les objets sont immuables : une fois écrits, ils ne peuvent pas être modifiés partiellement.

Les buckets



- Les objets / fichier sont organisées au sein de **conteneurs logiques** appelés **buckets**.
- Un bucket peut contenir un ensemble d'objets et ne peut pas inclure un bucket.
- Bien que le stockage objet repose sur une **organisation à plat** (sans arborescence réelle de répertoires), il est possible de **simuler une hiérarchie logique** des objets en utilisant des noms structurés par **chemin d'accès** (ex. : /ANF/envol/photo1). Cette convention permet une **meilleure lisibilité**, et facilite certaines opérations comme le **listing par préfixe** (prefix= /ANF/envol/ souvent utilisé dans les API (comme Amazon S3).

Les buckets



□ Isolation des espaces de noms

- Chaque bucket définit un namespace unique
- Deux objets peuvent avoir le même nom s'ils sont dans des buckets différents.
- Cela permet de séparer proprement les données entre utilisateurs, applications, projets...

□ Gestion des droits d'accès

- Les droits (ACL, policies, IAM, etc.) sont souvent définis au niveau du bucket mais peuvent être définis au niveau objet.

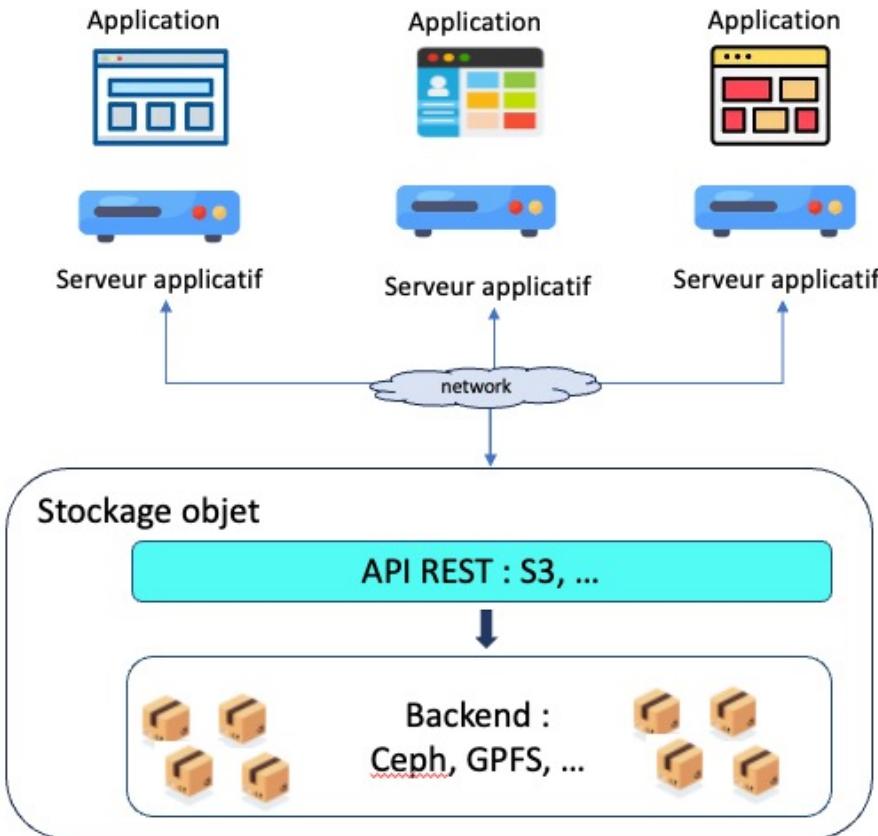
□ Topologie de placement :

- Dans certains systèmes (ex. Ceph), un bucket peut être lié à une région géographique, un data center ...
- Utile pour des raisons de performance, de latence, ou de réglementation (ex : RGPD)

□ Niveau de résilience

- Dans certains systèmes (ex. Ceph), un bucket est associé à un pool de stockage, qui définit le niveau de réPLICATION (ex 3 copies) ou un schema d'erasurement coding (ex 4+2).
- Offre une flexibilité de politique de stockage par usage.

Comment accéder à un service S3 ?



- Aujourd’hui, quand on parle de « stockage objet », on fait souvent référence à l’API (interface d’accès), et non à l’implémentation système.
- Quasiment tous les systèmes de stockage objet sont compatibles avec l’API S3 même si certains ont aussi leur propre API (OpenStack Swift, MinIO, etc.).
- L’API S3 d’Amazon est un standard de facto, propriétaire mais elle n’est pas un standard.

Comment accéder à un service S3 ?

- Une API REST pour le stockage objet
- REST signifie **Representational State Transfer** : c'est un **style d'architecture web** qui permet à des systèmes de communiquer via **HTTP** en utilisant des **verbes standard** (GET, PUT, DELETE, etc.).
- Dans le contexte du **stockage objet**, une **API REST** permet **d'interagir avec les objets et les buckets** via des appels **HTTP**, depuis n'importe quelle application, navigateur, ou script.

Action	Verbe HTTP
Télécharger un fichier	GET
Uploader un fichier	PUT
Supprimer un fichier	DELETE
Récupérer les headers d'un objet	HEAD

Comment accéder à un service S3 ?

- Via SDK

```
import boto3
from botocore.exceptions import ClientError

# === Configuration MinIO (ou AWS) ===
S3_ENDPOINT = 'http://minio:9000'
ACCESS_KEY = 'minioadmin'
SECRET_KEY = 'minioadmin'
REGION = 'eu-east-1'

# === Connexion au service S3 ===
s3 = boto3.client('s3',
                  endpoint_url=S3_ENDPOINT,
                  aws_access_key_id=ACCESS_KEY,
                  aws_secret_access_key=SECRET_KEY,
                  region_name=REGION)
|
s3.put_object('monfichier.txt', 'mon-bucket', 'chemin/monfichier.txt')
```

Comment accéder à un service S3 ?

- Via AWS cli

Fonctionnalité	aws s3 (haut niveau)	aws s3api (bas niveau)
Créer un bucket	✓ mb	✓ create-bucket
Lister les objets	✓ ls	✓ list-objects-v2
Upload (copier un fichier)	✓ cp	✓ put-object
Ajouter des métadonnées (--metadata)	✗ Non	✓ Oui (--metadata)
Modifier les métadonnées	✗ Non	✓ copy-object avec --metadata-directive REPLACE
Supprimer un objet	✓ rm	✓ delete-object
Supprimer un bucket	✓ rb	✓ delete-bucket
Télécharger un objet	✓ cp	✓ get-object
Voir les métadonnées d'un objet	✗ Non	✓ head-object
Liste des versions	✗ Non	✓ list-object-versions

```
# Création d'un bucket ( mb : make bucket)
aws s3 --endpoint-url http://minio:9000 mb s3://mon-bucket
# suppression d'un bucket ( rb : remove bucket)
aws s3 --endpoint-url http://minio:9000 rb s3://mon-bucket
# Liste les objets d'un bucket
aws s3 --endpoint-url http://minio:9000 ls s3://mon-bucket
# Copie un fichier dans un bucket
aws s3 --endpoint-url http://minio:9000 cp lsst.png s3://mon-bucket/
# télécharger un fichier
aws s3 --endpoint-url http://minio:9000 cp s3://mon-bucket/test .
```

Comment accéder à un service S3 ?

- Via console web

The screenshot shows the MINIO Object Store Community Edition interface. On the left, there's a sidebar with a 'Create Bucket' button and a 'Filter Buckets' input field. Below that is a list of buckets: demo-public, demo-tp, demo-tp-bucket, mon-bucket, and public-bucket. The main area is titled 'Object Browser' with a search bar that says 'Start typing to filter objects in the bucl'. It shows a bucket named 'mon-bucket' created on 'Sat, Oct 04 2025 17:59:23 (GMT+2)' with 'Access: PRIVATE' and '14.0 B - 1 Object'. There are buttons for 'Rewind', 'Refresh', and 'Upload'. A table lists one object: 'test' (Last Modified: Sat, Oct 04 2025 18:00 (GMT+2), Size: 14.0 B). There are also buttons for 'Create new path' and a trash icon.

Name	Last Modified	Size
test	Sat, Oct 04 2025 18:00 (GMT+2)	14.0 B

Fonctionnement du stockage objet

- Dans un système de stockage objet, les objets sont immuables : une fois écrits, ils ne peuvent pas être modifiés partiellement
- Ce comportement en fait une solution idéale pour les applications suivant un modèle d'accès WORM (*Write Once, Read Many*), telles que les sauvegardes, les médias (images, vidéos) ou les logs.
- En revanche, le stockage objet est **contre-indiqué** pour les applications qui effectuent des **accès aléatoires fréquents**, comme les bases de données ou les systèmes nécessitant une **faible latence**.
- La **taille maximale d'un objet** varie selon les implémentations, mais elle est généralement de l'ordre de **5 To** (ex. : Amazon S3 , CEPH).

Généralité sur le stockage objet



❑ Gestion des permissions

- La gestion des permissions s'effectue au niveau objet ou bucket – pas besoin d'UID/GID partagés (auth via clés/API) et donc de synchronisation entre les clients.
- Les utilisateurs sont généralement authentifiés via des tokens, clés d'API, IAM, etc., pas via un système d'OS partagé.
- Cela simplifie la gestion des accès dans des environnements hétérogènes (multi-utilisateurs, multi-cloud...).

❑ Modification impossible

- Chaque changement = nouvelle version ou réécriture complète
- Possibilité d'activer le versioning d'un objet

❑ Pas de script exécutable directement

- Objets accessibles uniquement via API REST, pas comme des fichiers montés.
- Téléchargement de l'objet puis exécution localement.

❑ Lecture partielle possible

- Télécharger juste un segment d'une vidéo, ou une portion d'un fichier binaire (via les headers HTTP)

❑ Tag personnalisables

- Attention, modifier une métadonnée revient à réécrire l'objet entièrement.
- Même si les objets sont physiquement plats, il est possible de les organiser logiquement (projet=ENVOL domain=stockage type=cours)
- Il est possible de rechercher des objets ciblés selon les tags (projet=ENVOL & domain=stockage)

TP S3

