Big Data

Chapitre 2: Hadoop

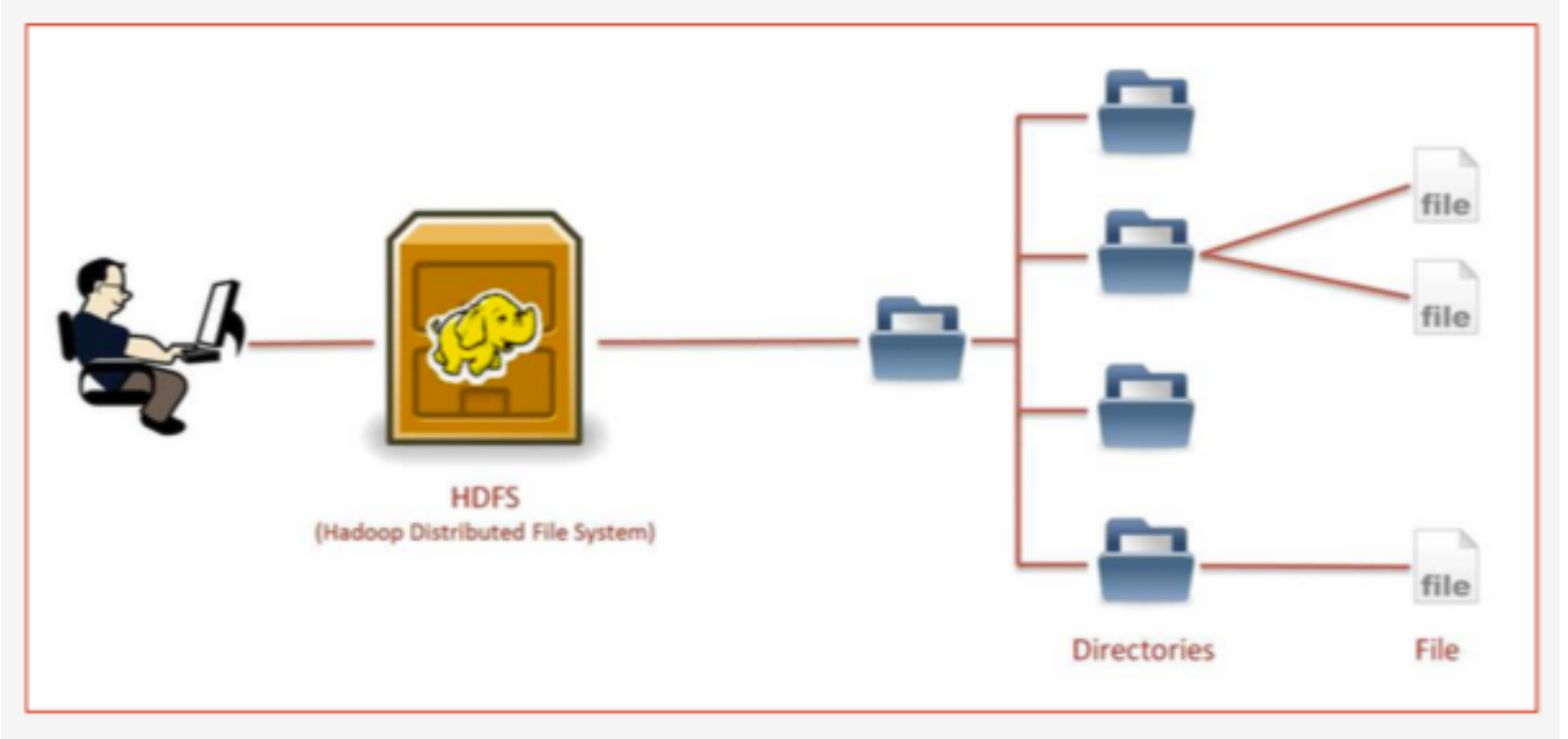
Présentation du framework

HDFS

Map Reduce

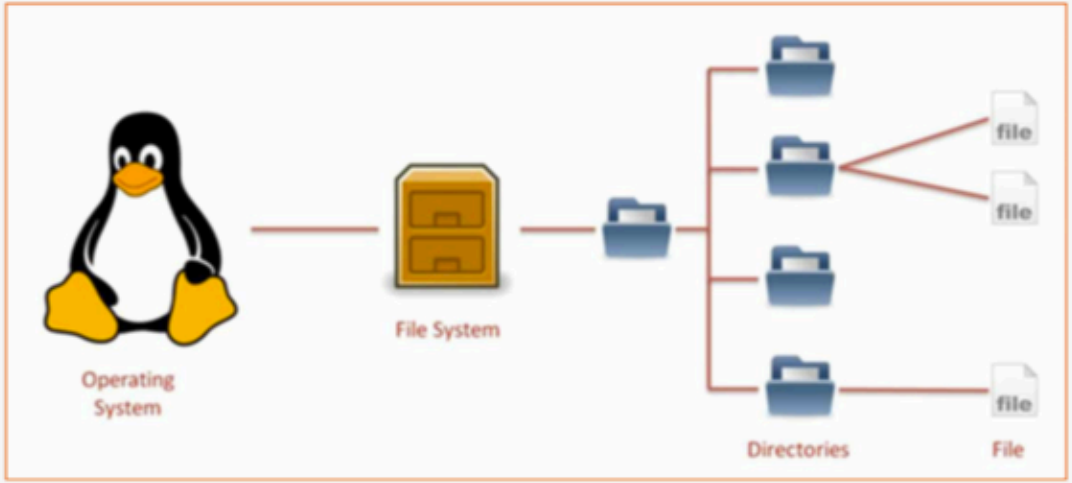
HDFS : Hadoop Distriuted File System

- Solution de stockage de données massives (des PO) d'une manière distribuée.
- Tolérant aux pannes avec la réplication des données.

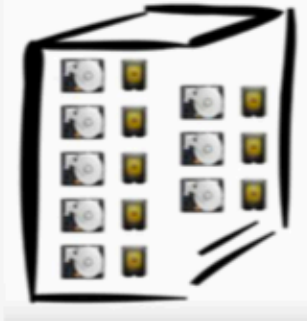


Système de fichiers classique

- Un Système de fichiers permet de
  - Gérer les répertoires et fichiers
    - Créer, Copier, Supprimer, renommer, déplacer
    - Gérer les droits d'accès
- Les fichiers sont organisés dans une structure hiérarchique dans des répertoires.
- Les dossiers et les fichiers sont stockés dans des unités de stockage (Disque dur, Disques amovibles, etc.) locales organisées en partitions
- Chaque OS dispose de son propre système de fichiers
- Quand la quantité de données gérées atteint les limites de stockage de la machine, on cherche à étendre les capacités en ajoutant d'autres disques durs supplémentaires : Vertical Scaling (Scalabilité Verticale)

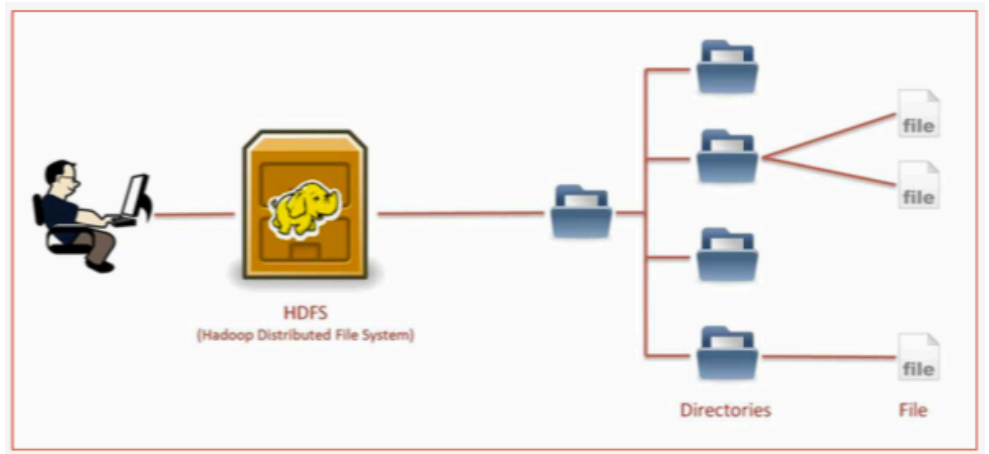


Vertical Scaling



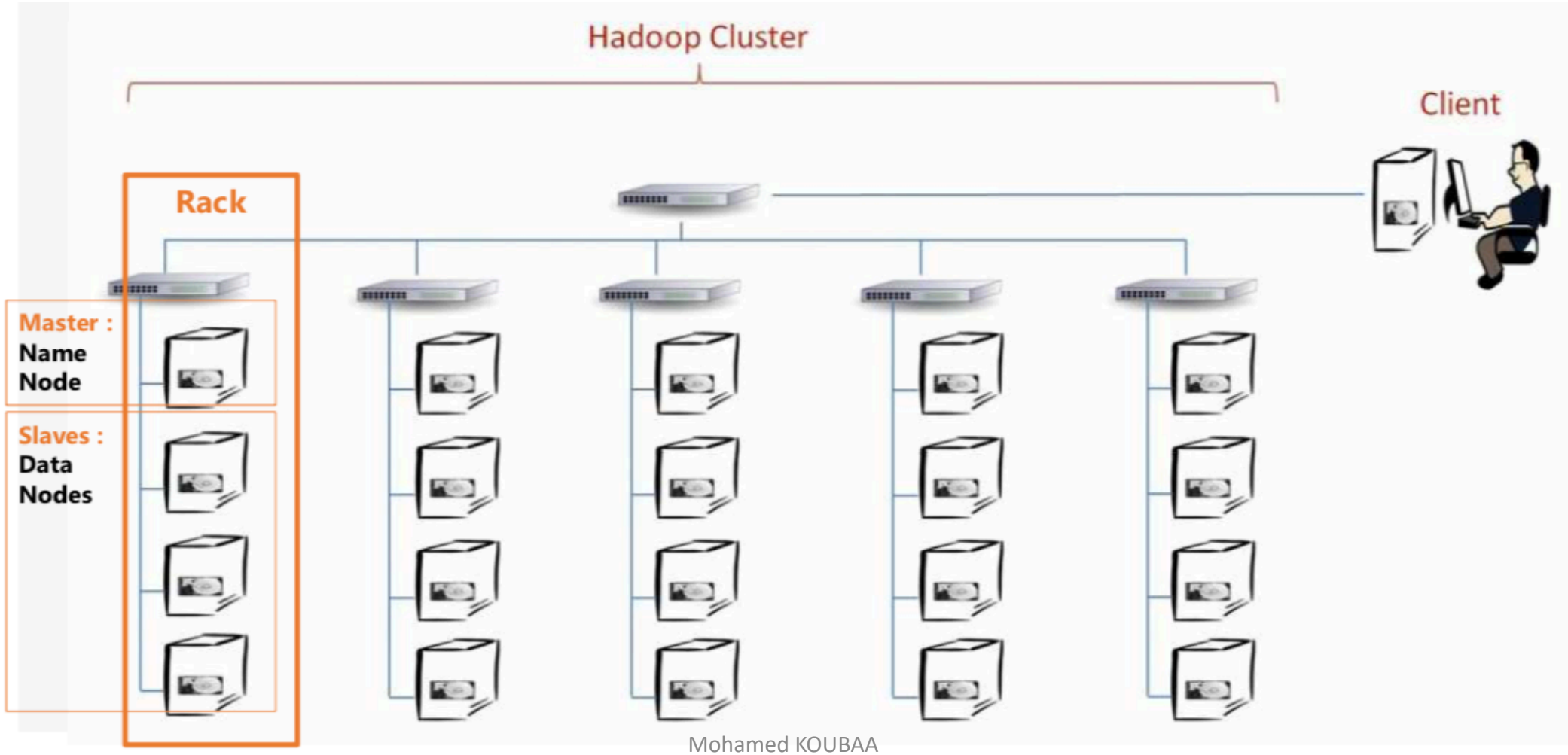
HDFS

- HDFS est aussi un système de fichiers
- Système de Fichiers Distribué
- HDFS permet de combiner les capacités de plusieurs machines distribuées sous forme d'un unique système très large.
- Il permet de créer des répertoires et stocker les données dans des fichiers d'une manière distribuée dans plusieurs machines
- Caractéristiques de HDFS
  - Distribué
  - Scalable Horizontalement
  - Cost effective(Commencer par de simple machines)
  - Fault Tolerant (Tolérent aux panes)

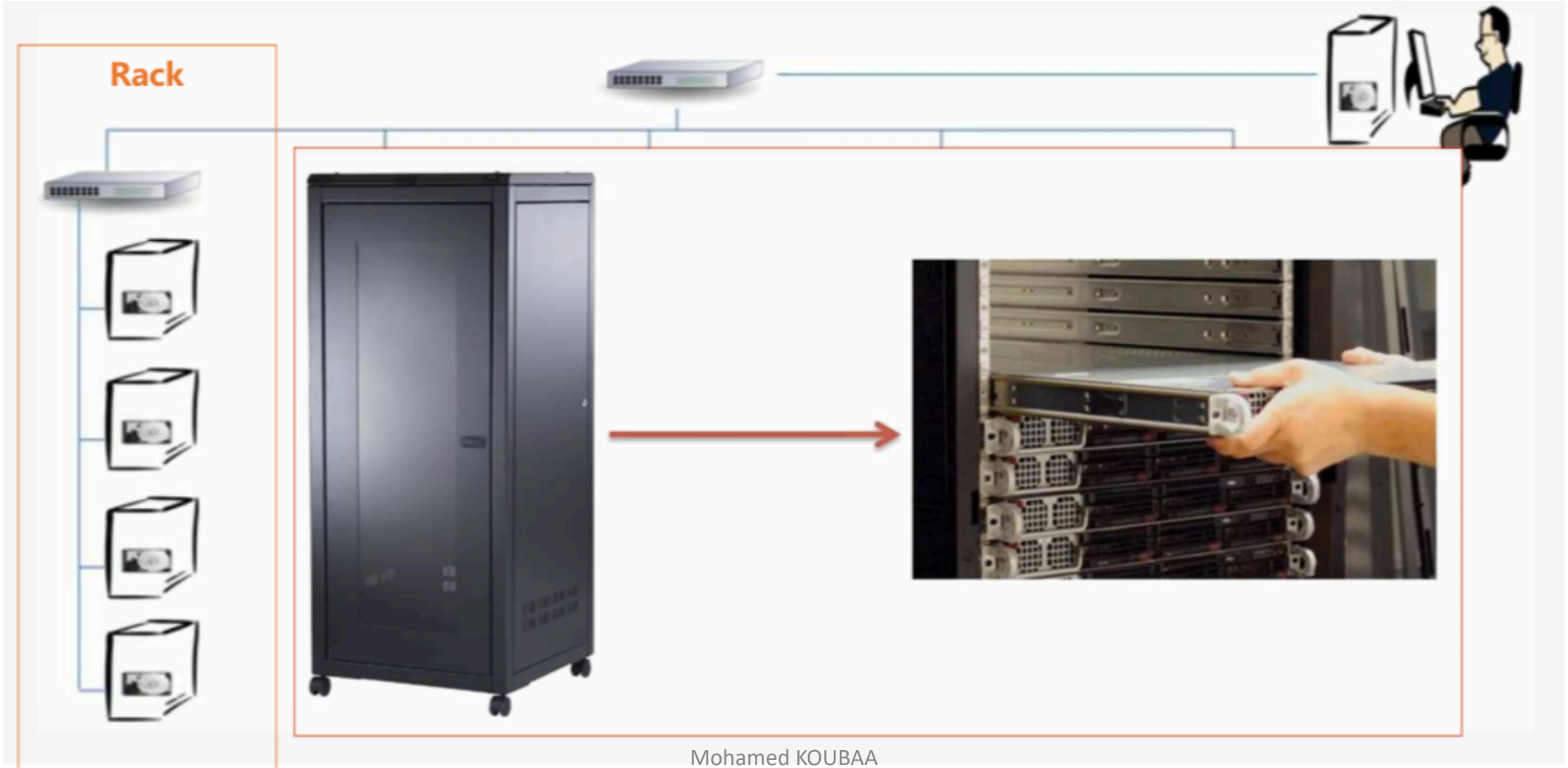


Horizontal Scaling

Hadoop cluster

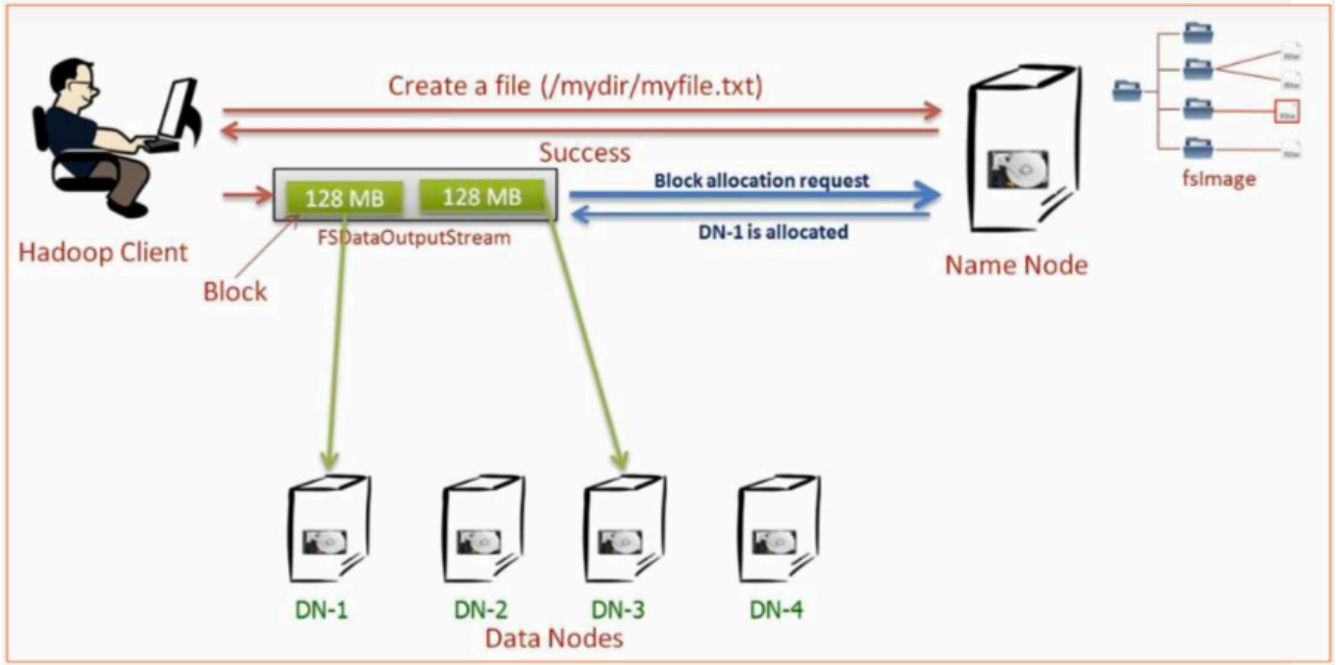


Hadoop cluster



HDFS

1. Le client Hadoop envoie une requête au NN pour demander de créer un fichier
2. Le NN procède à quelques vérifications : Existence, droits d'accès en utilisant In Memory FS Image.
3. Si les vérifications passent, Le NN Crée le dossier dans FSImage et retourne success
4. Le client écrit les données dans FSDataOutputStream
5. Pour chaque block de 128 MB, le client HDFS demande au NN l'allocation du block.
6. Le NN assigne un DN pour ce block.
7. Le streamer envoie les données du block au DN
8. Une fois toutes les données sont écrites, le NN commite les modifications dans le Système de fichier



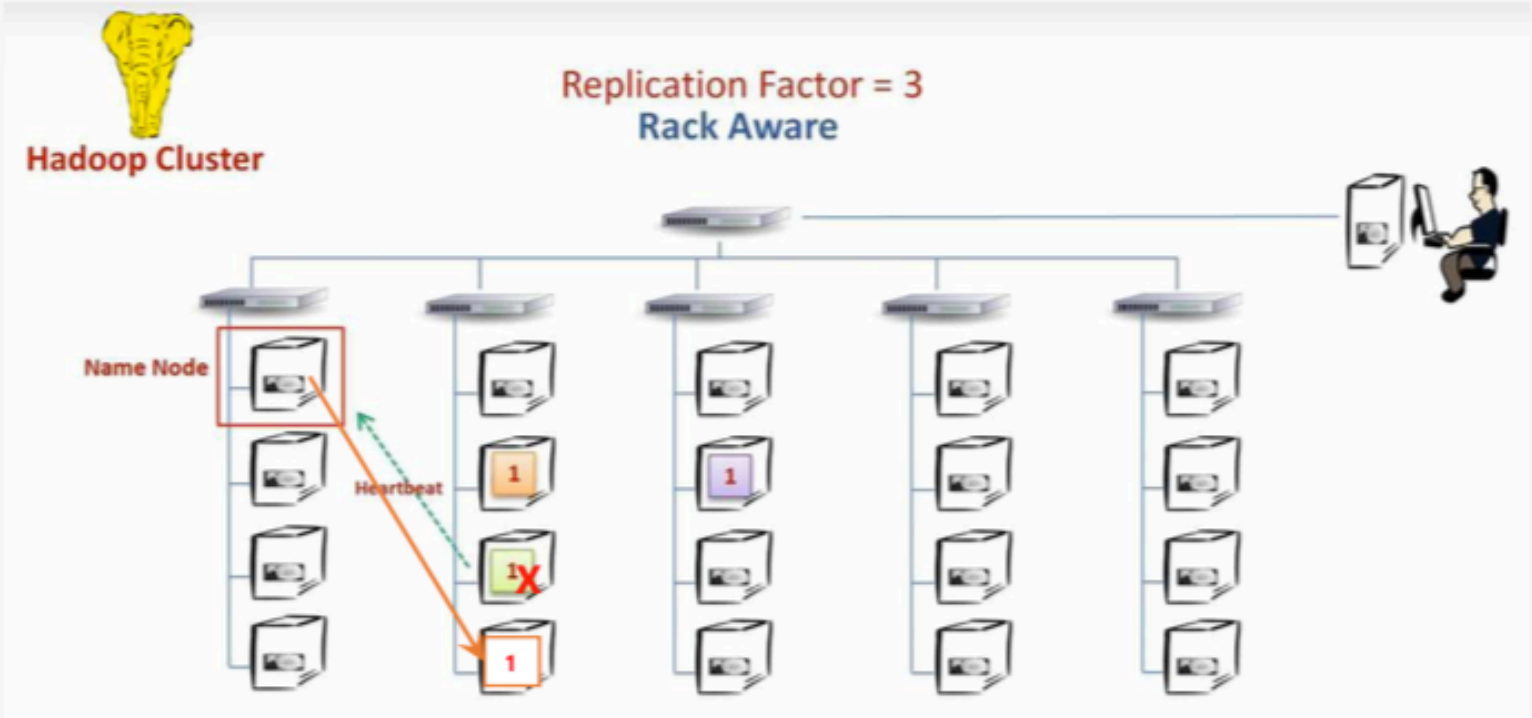
HDFS architecture

- Un Cluster Hadoop se compose de un à plusieurs Racks
- Il est conçu avec une architecture Master Esclaves
  - Un Master : Name Node
  - Un à plusieurs Esclaves : Data Nodes
- Le NN Gère l'espace de noms du système de Fichiers (Manage Namespaces)
- Toutes les interactions du client HDFS commencent avec le NameNode
- Les DataNodes stockent les données des fichiers (Manage Data)

- NN maintient un fichier contenant une table d'adressage des blocks dans FSImage.
- Le fichier est divisé en plusieurs blocks enregistrés dans des DN.
- La taille par défaut d'un block est de 128MB (version 2).
- On peut changer la taille d'un block pour un fichier.
- Les DN envoient des Heartbeats et le rapport des blocks au NN
- Le client interagit directement avec les DN pour la lecture et l'écriture des données de bocks.

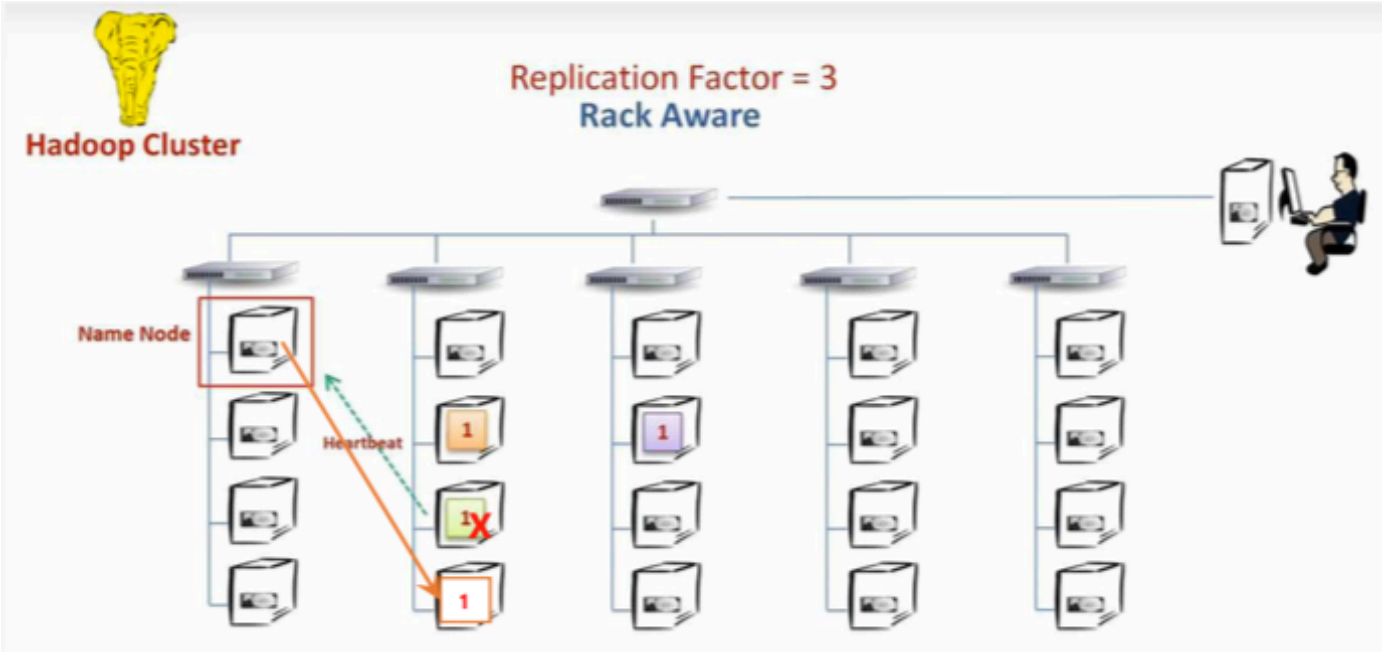
HDFS: Tolérance aux pannes

- Quand HDFS enregistre un block dans un DN, il enregistre également deux copies de backup dans d'autres DN.
- Si un DN tombe en panne, HDFS restore les blocks perdus à partir des duplicatas dans d'autres DN.
- Le nombre de copies est configurée dans le paramètre REPLICATION_FACTOR qui est par défaut égale à 2.
- La valeur de REPLICATION_FACTOR peut être changée pour chaque fichier.



HDFS: Tolérance aux pannes

- Si le mode Rack Aware est activé, HDFS s'assure que chaque copie d'un block est enregistrée dans un DN d'un Rack différent des autres copies.
- Ce qui permet de se prévenir d'un éventuel échec d'un Rack.
- Comme les DN envoient des Heart beats (pulsations) au NN, quand un DN échoue, NN crée à nouveau des copies des blocks de ce DN en vue de maintenir le REPLICATION_FACTOR.
- Le système de réplication coûte cher en terme de stockage
- Il existe des soutions pour économiser le stockage:
  - HDFS 2.0 => « Storage Policies » pour économiser le stockage
  - HDFS3.0 => « Erasure coding » comme alternative à la réplication

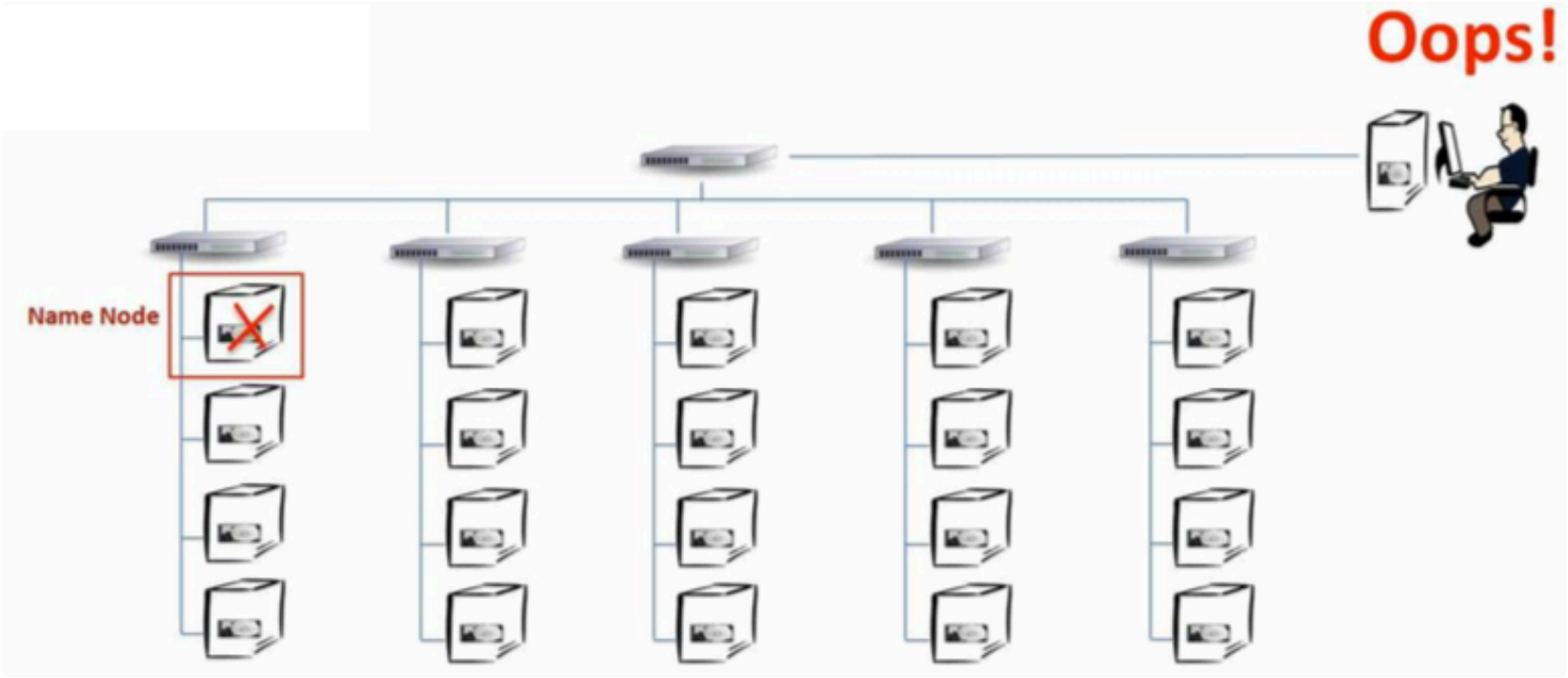


HDFS: Haute disponibilité

- **La tolérance aux pannes** (Fault Tolerance) mesure la capacité d'un système distribué à continuer à fonctionner correctement dans le cas d'une panne d'un composant du système (Disque dur, CPU, Ordinateur, Switch, Rack, etc..)
- **La haute disponibilité** (High Availability) mesure le pourcentage de disponibilité des services offerts par un système à cours, moyen et à long termes.

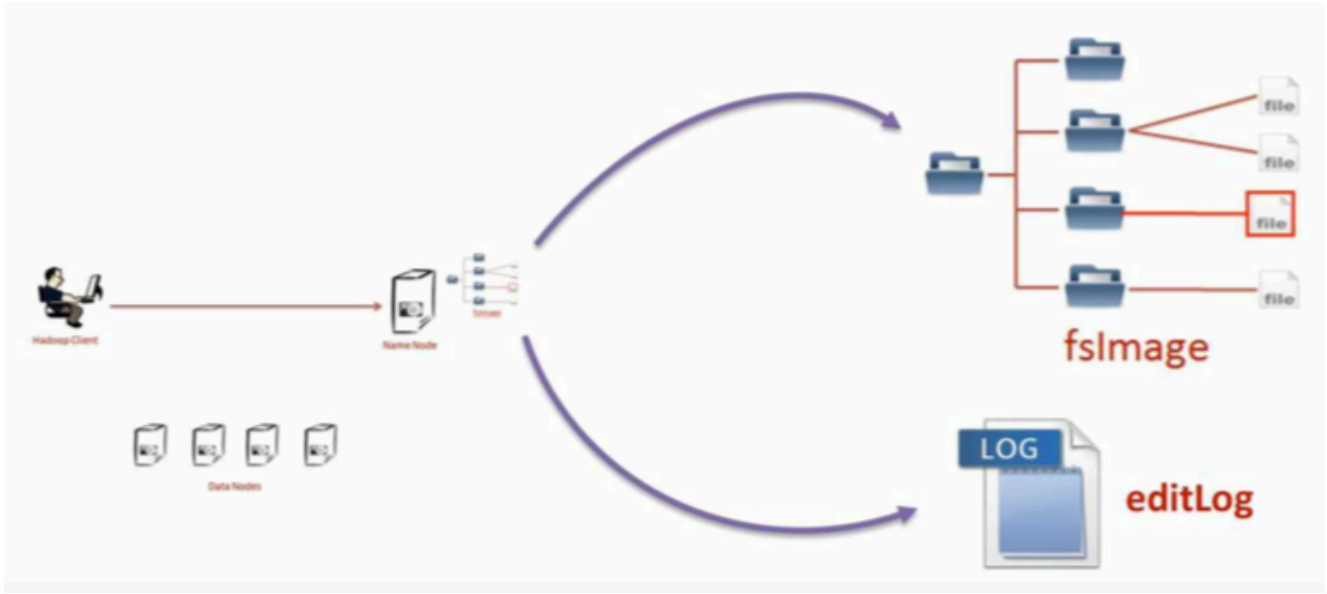
HDFS: Haute disponibilité

- Le Name Node est un point de défaillance unique dans un cluster Hadoop
- Comment protéger l'échec du Name Node?



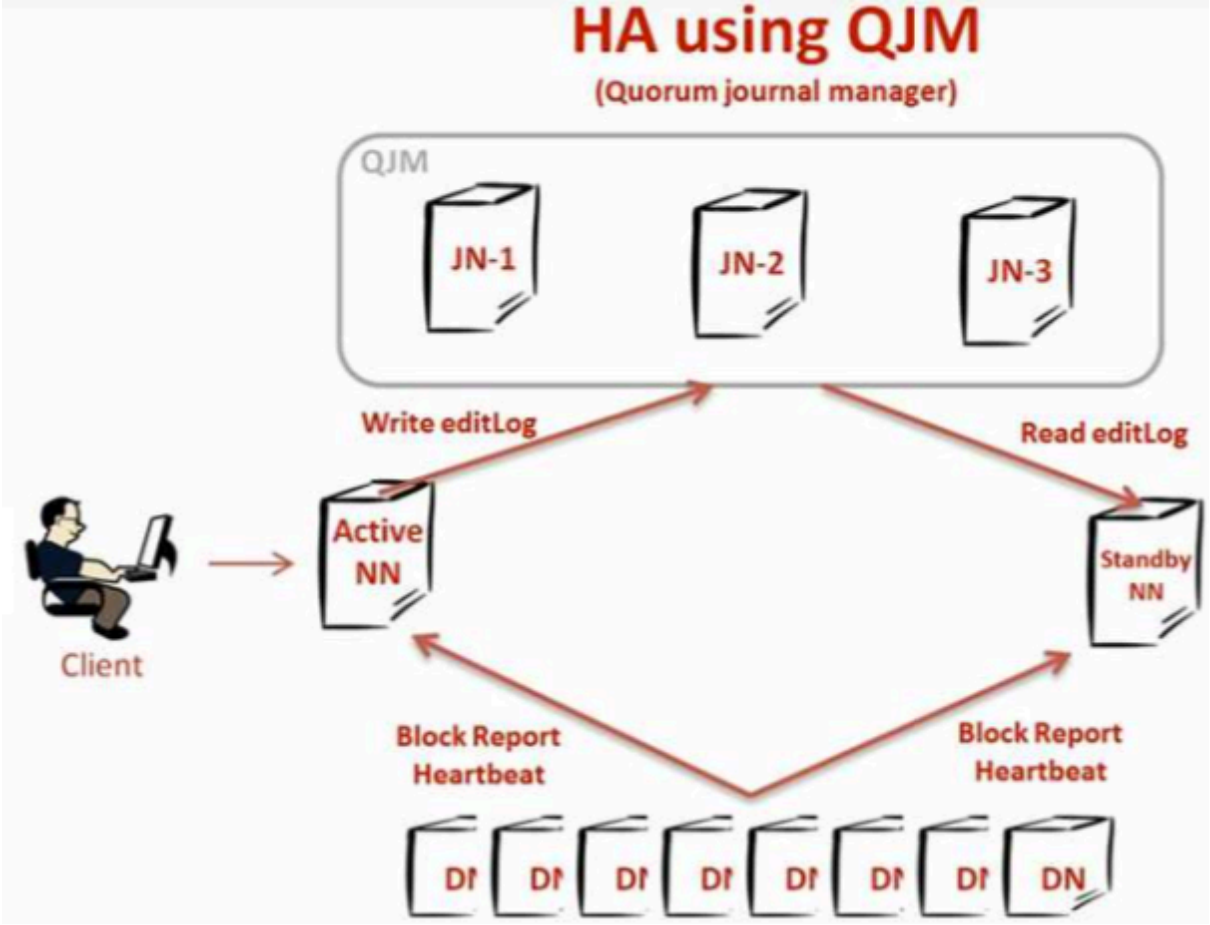
HDFS: Haute disponibilité

- Le NameNode conserve l'intégralité du système de fichiers en mémoire dans FsImage (In memory FsImage)
- Le NameNode maintient aussi le fichier editLog (journal de modifications)
- Toute modification dans le système de fichier est enregistrée dans le fichier editLog
- Avec seulement le fichier editLog, on peut reconstruire le FsImage en mémoire
- La solution est de faire un backup du fichier editLog
  - Comment?



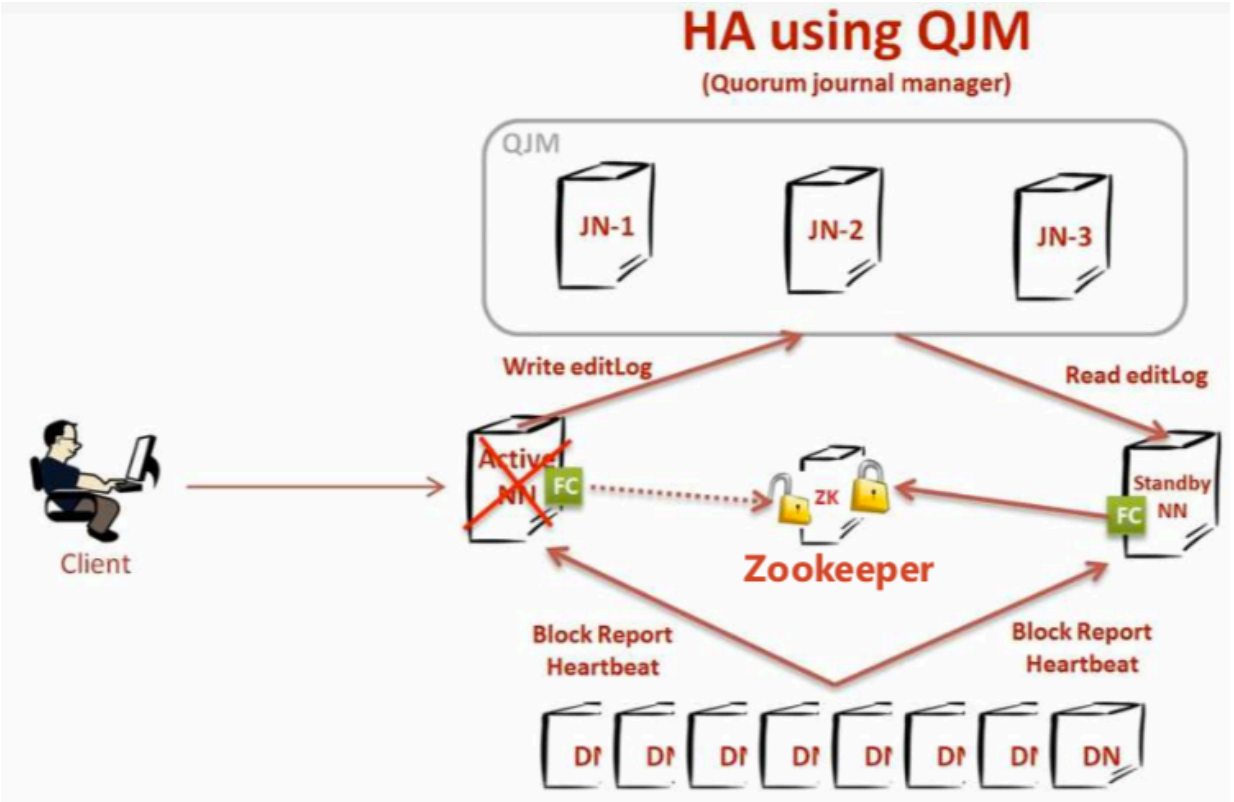
HDFS: Haute disponibilité (Quorum Journal Manager)

- Le QJM est un ensemble de machines (3 au minimum)
- Chaque machine est configurée pour exécuter un journal node daemon
- Une fois le QJM est mis en place, il faut configurer le NN pour écrire le editLog dans le QJM au lieu de l'écrire sur le disque local
- Le **standBy NN** est un nœud de cluster qui lit en permanence le editLog à partir du QJM
- Tous les DataNodes envoient leurs rapports (pulsations) simultanément aux deux NameNodes



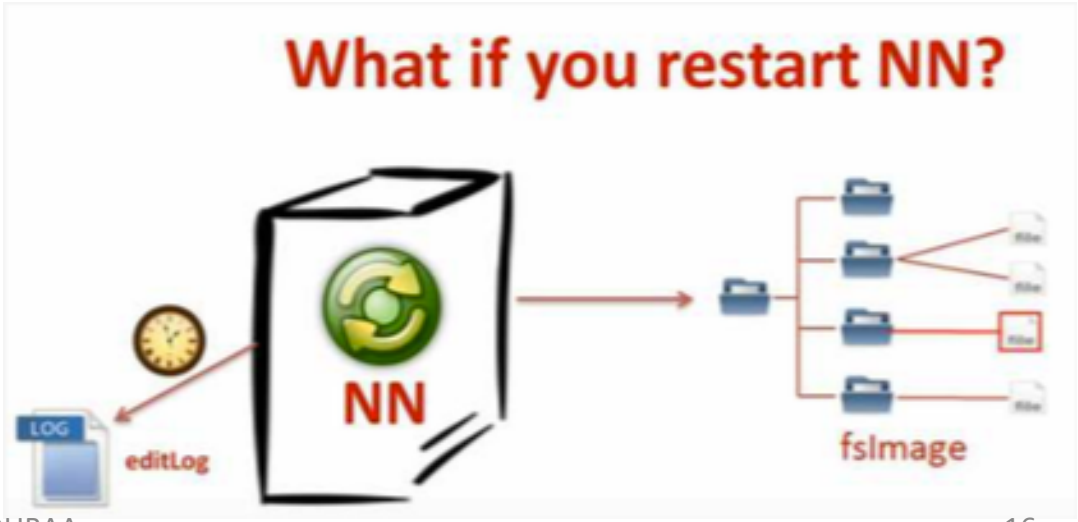
HDFS: Haute disponibilité (Quorum Journal Manager)

- Comment le standBy NameNode va savoir que le NameNode a subi un échec et qu'il doit prendre la relève?
- Zookeeper assure la coordination
- Le NameNode principal détient un verrou dans zookeeper
- Le **standBy NN** essaie en permanence de prendre le verrou
- En cas de panne du NameNode, son verrou expire et le standBy NN réussit dans ce cas à détenir le verrou et prendre ainsi la relève
- Il passe de l'état standBy NN à Active NN



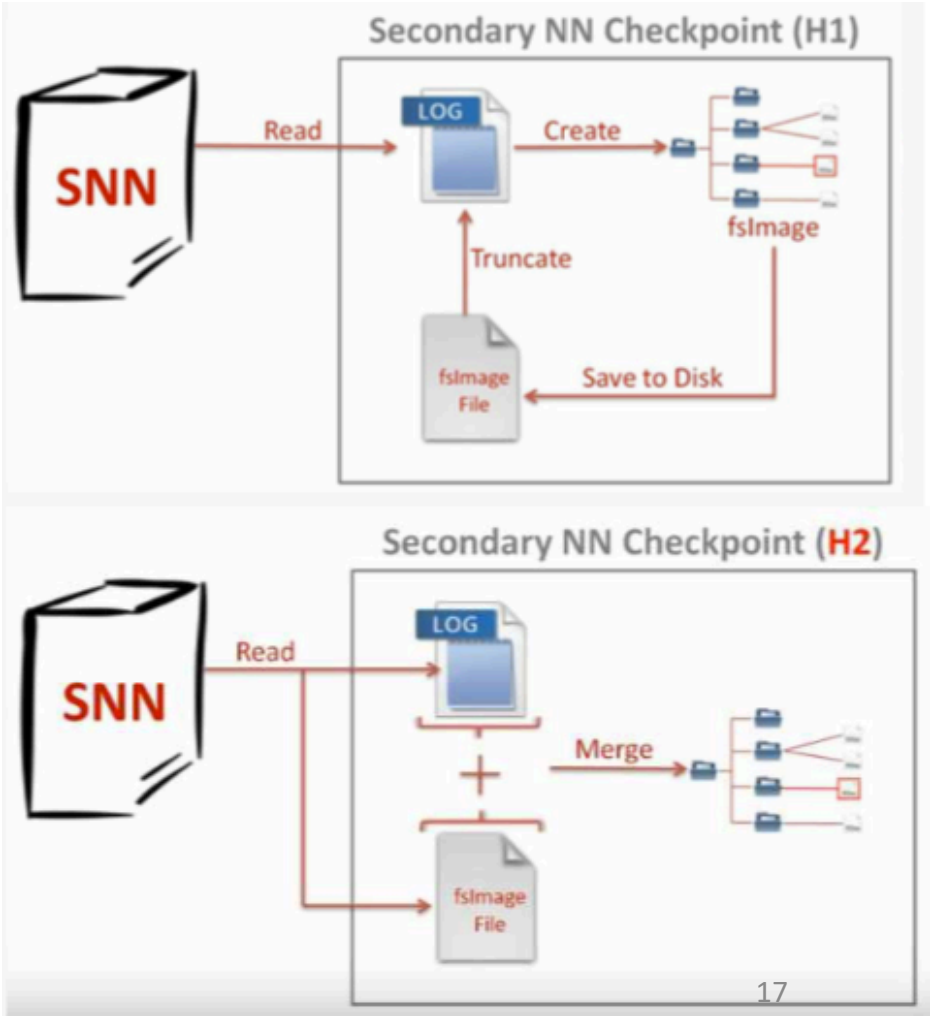
HDFS: Haute disponibilité (Secondary Name Node)

- Le Secondary Name Node a un rôle différent de celui de StandBy NN
- Il a une autre responsabilité
- Le fichier editLog est persistent, au contraire du InMemory FsImage (chargé en mémoire donc volatile)
- Au redémarrage, le InMemory FsImage est perdu
- Le NameNode reconstruit le InMemory FsImage en parsant le editLog (lecture entière du fichier)
- Opération très longue vu que le fichier editLog est très volumineux



HDFS: Haute disponibilité (Secondary Name Node)

- Le Secondary Name Node vient pour résoudre ce problème
- Toutes les heures, il fait la sauvegarde de l'état du système de fichier
- La toute première sauvegarde:
  - Lit le fichier editLog
  - Crée le dernier état du fsImage
  - Le sauvegarde sur le disque local (on Disk fsImage)
  - Efface le contenu du editLog puisque toutes les modifications ont été appliquées
- Toutes les heures suivantes:
  - Le SNN lit le fichier fsImage
  - Applique les modifications cumulées dans le editLog tout au long de la dernière heure
  - Met à jour le on Disk fsImage
  - Efface le contenu du editLog



HDFS: Haute disponibilité (Secondary Name Node)

- HDFS est un système de fichiers distribué sur la plate-forme hadoop
- Il assure le stockage de données pour les composants Big Data tels que hive, hbase, ...
- Sur le shell HDFS, nous pouvons exploiter et gérer le système de fichiers distribué.
- Il existe plusieurs types de commandes : dfs, dfsadmin, …
- Pour exécuter une commande dfs :

```
hdfs dfs commande
```

`hdfs dfs -help` permet d'afficher les détails des commandes dfs

HDFS: Haute disponibilité (Secondary Name Node)

Commande	Description
-**ls** <path>	Afficher les fichiers et répertoires se trouvant dans « path »
-**mkdir** <path>	Créer un répertoire
-**put** <local path> <hdfs path> Ou bien –**copyFromLocal**	Copier un fichier du local vers hdfs
-**get** <hdfs path> <local path> Ou bien –**copyToLocal**	Copier un fichier du hdfs vers local
-**moveFromLocal** <local path> <hdfs path>	Couper puis copier un fichier du local vers hdfs
-**cp** <hdfs path> <hdfs path>	Copier un fichier d'un emplacement à un autre
-**mv** <hdfs path> <hdfs path>	Déplacer un fichier d'un emplacement à un autre
-**rm** <hdfs path>	Supprimer un fichier
-**cat** <path>	Afficher le contenu d'un fichier
-**df** <path>	Vérifier l'espace disponible