# Introduction à l'ecosystème Hadoop

## Objectif de la séance

Mise en place d'une plateforme Hadoop afin de :

- stocker des fichiers dans HDFS ;
- observer leur répartition ;
- vérifier la réplication ;
- simuler une panne ;
- lancer un traitement distribué.

À retenir :

```
Hadoop = un écosystème
HDFS   = stocker les données
YARN   = gérer les ressources
MapReduce = traiter les données
```

## 1. Pourquoi Hadoop ?

Une plateforme e-commerce collecte quotidiennement :

- des commandes ;
- des événements de navigation ;
- des journaux applicatifs ;
- des données clients et produits.

Lorsque les volumes deviennent importants, une seule machine peut atteindre ses limites :

- capacité de stockage limitée ;
- temps de traitement trop important ;
- risque de perte en cas de panne ;
- difficulté à exécuter plusieurs traitements simultanément ;
- augmentation de capacité difficile.

Hadoop propose d’utiliser plusieurs machines pour :

- répartir le stockage ;
- répartir les calculs ;
- améliorer la tolérance aux pannes ;
- augmenter progressivement la capacité.

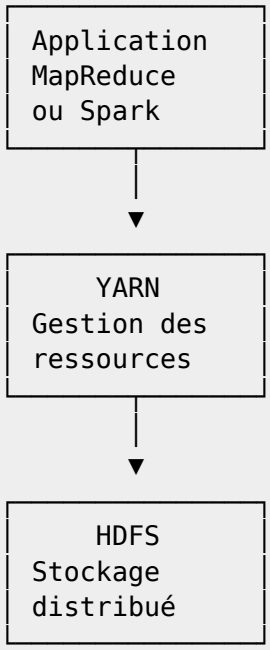


Hadoop permet de stocker et de traiter de grandes quantités de données sur plusieurs machines.

---

# 2. Hadoop en une image

Hadoop regroupe plusieurs composants qui remplissent des rôles différents :




| Composant | Rôle |
|---|---|
| Hadoop | Écosystème de technologies distribuées |
| HDFS | Stockage des fichiers sur plusieurs nœuds |
| YARN | Gestion des ressources et des applications |
| MapReduce | Modèle de traitement distribué |
| Spark | Moteur de traitement distribué étudié dans la suite du module |



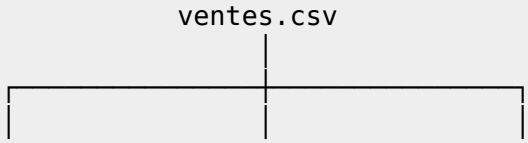
Hadoop ne désigne pas uniquement HDFS. HDFS est la partie stockage, YARN gère les ressources et MapReduce fournit un modèle de traitement.

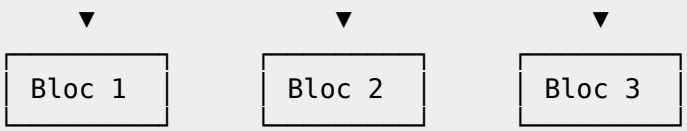
---

# 3. HDFS : le stockage distribué

## 3.1. Découpage en blocs

Lorsqu'un fichier est déposé dans HDFS, il est découpé en blocs.



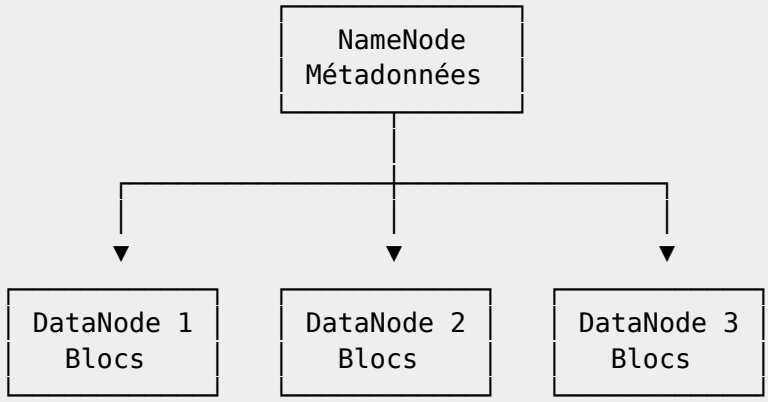



Les blocs sont ensuite répartis sur les différents DataNodes.

## 3.2. NameNode et DataNodes

HDFS repose principalement sur deux types de nœuds :

- le **NameNode** ;
- les **DataNodes**.




#### NameNode

Le NameNode conserve les métadonnées du système de fichiers :

- noms des fichiers ;
- arborescence des répertoires ;
- taille des fichiers ;
- emplacement des blocs ;
- facteur de réplication.

Le NameNode ne contient généralement pas le contenu des fichiers.

#### DataNodes

Les DataNodes stockent physiquement les blocs de données.

Ils sont également chargés de :

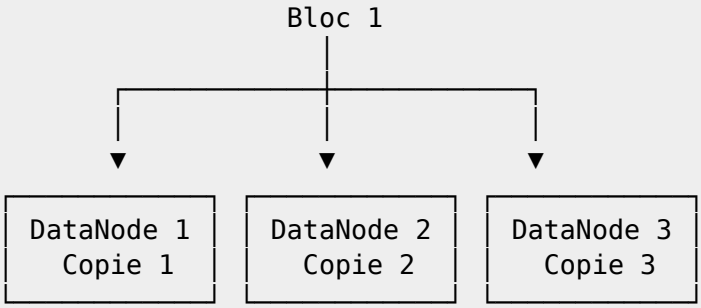
- répondre aux demandes de lecture et d’écriture ;
- signaler régulièrement leur état au NameNode ;
- gérer les copies des blocs.



Le NameNode sait où se trouvent les blocs. Les DataNodes stockent réellement les blocs.

### 3.3. Réplication des blocs

Pour éviter la perte de données en cas de panne, HDFS conserve plusieurs copies de chaque bloc.




Avec un facteur de réplication égal à 3, chaque bloc possède trois copies.

Si un DataNode tombe en panne, HDFS peut encore lire le bloc depuis un autre DataNode.



La réplication améliore la tolérance aux pannes, mais elle augmente l’espace de stockage nécessaire.

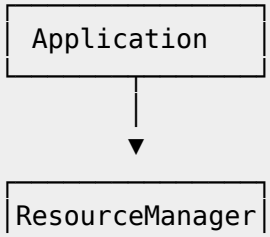
---

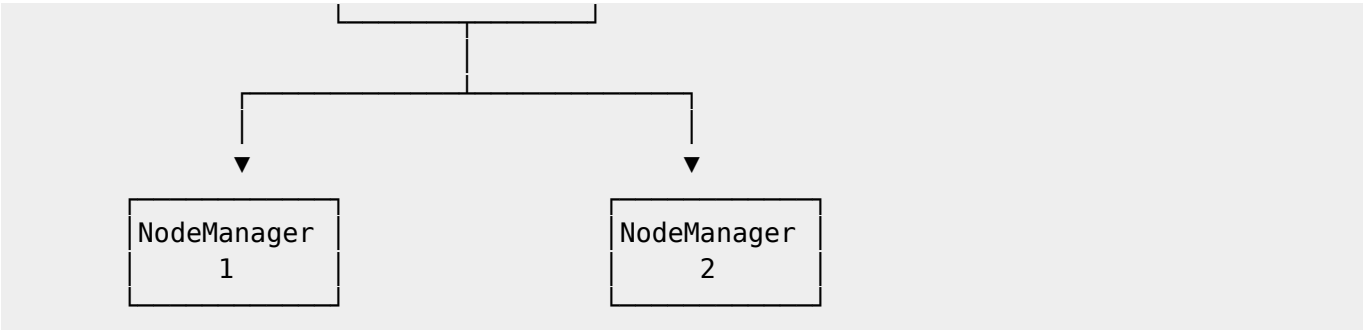
## 4. YARN : la gestion des ressources

YARN est chargé de gérer les ressources disponibles sur le cluster.

Il décide notamment :

- quelles applications peuvent s’exécuter ;
- sur quels nœuds elles s’exécutent ;
- quelle quantité de mémoire leur est attribuée ;
- combien de cœurs leur sont attribués.






#### ResourceManager

Le ResourceManager est le composant central de YARN.

Il :

- connaît les ressources disponibles ;
- attribue ces ressources aux applications ;
- planifie l'exécution des traitements.

#### NodeManager

Chaque machine exécutant des tâches possède un NodeManager.

Il est chargé de :

- surveiller les ressources utilisées ;
- lancer les tâches demandées ;
- transmettre des informations au ResourceManager.

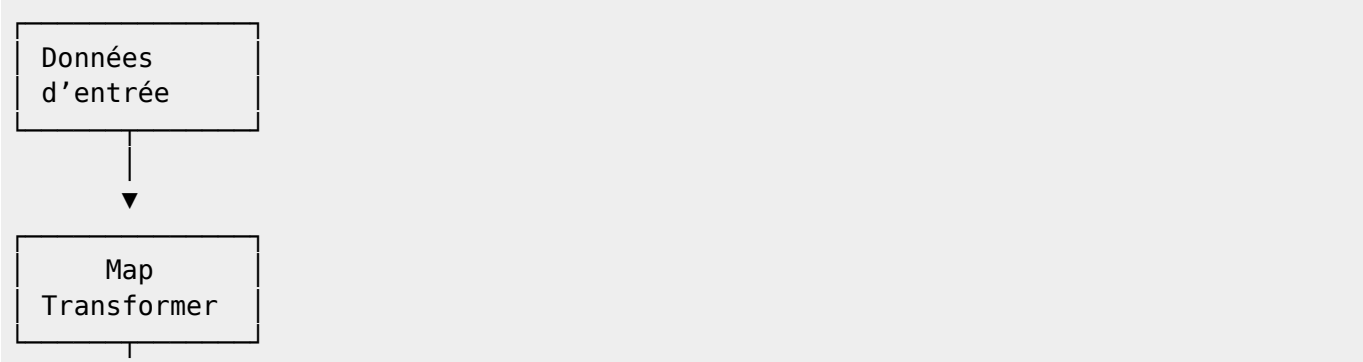


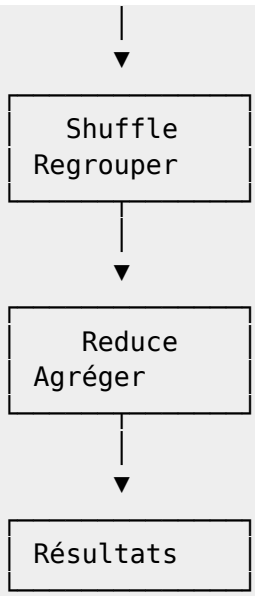
HDFS gère les données. YARN gère les ressources.

---

## 5. MapReduce : traiter les données

MapReduce permet de traiter les données en plusieurs étapes :






### Exemple : compter les catégories

Données d'entrée :

```
informatique
maison
informatique
sport
maison
```

### Étape Map

Chaque ligne devient une paire clé-valeur :

```
(informatique, 1)
(maison, 1)
(informatique, 1)
(sport, 1)
(maison, 1)
```

### Étape Shuffle

Hadoop regroupe les valeurs par clé :

```
(informatique, [1, 1])
(maison, [1, 1])
(sport, [1])
```

### Étape Reduce

Le système additionne les valeurs :

```
informatique : 2
maison       : 2
sport        : 1
```

### Intérêt de MapReduce

MapReduce permet de :

- traiter plusieurs blocs en parallèle ;
- répartir les calculs sur plusieurs machines ;
- relancer certaines tâches en cas de problème ;
- stocker les résultats dans HDFS.



MapReduce est particulièrement adapté aux traitements batch. Spark propose une approche plus flexible et souvent plus rapide pour de nombreux traitements.

---

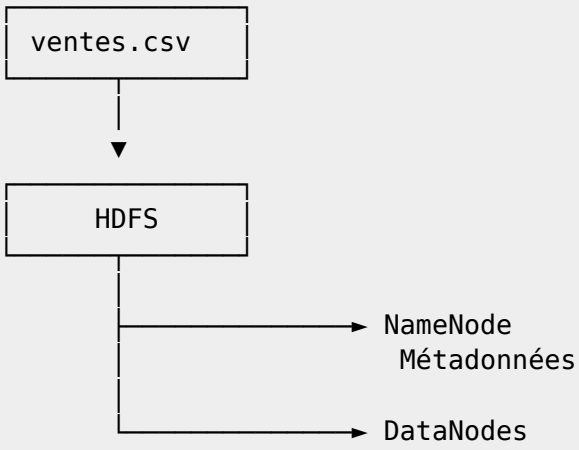
## 6. Lien avec le TD

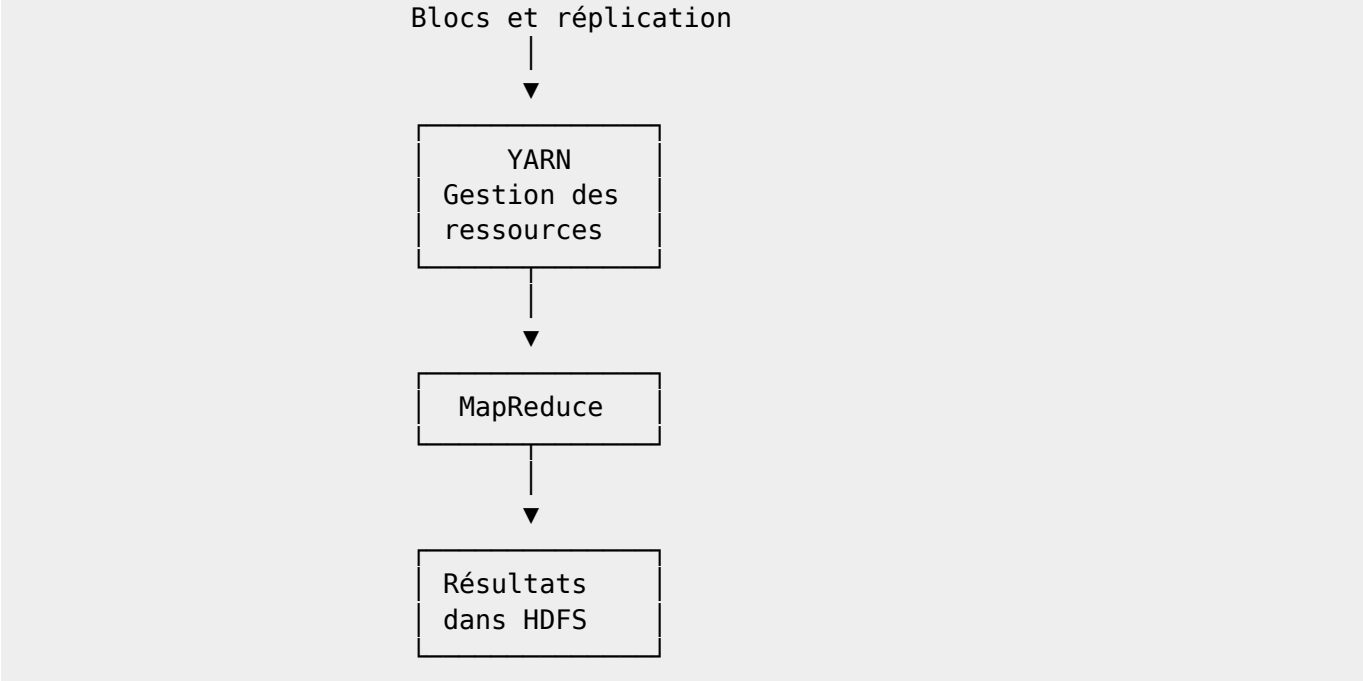
Dans le TD, nous allons :

1. lancer les conteneurs Docker ;
2. observer les nœuds Hadoop ;
3. créer des répertoires HDFS ;
4. déposer un fichier dans HDFS ;
5. observer ses blocs et sa réplication ;
6. arrêter un DataNode ;
7. vérifier que les données restent accessibles ;
8. observer YARN ;
9. lancer un traitement ;
10. lire les résultats.

---

## 7. Schéma général du TD






---

## À retenir

1. **Hadoop** est un écosystème de traitement distribué.
2. **HDFS** stocke les fichiers en les découpant en blocs répartis sur plusieurs DataNodes.
3. **Le NameNode** conserve les métadonnées et localise les blocs.
4. **YARN** attribue les ressources aux applications.
5. **MapReduce** traite les données avec les étapes Map, Shuffle et Reduce.



- HDFS stocke.
- YARN organise les ressources.
- MapReduce traite.

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
**http://slamwiki2.kobject.net/eadl/bloc5/fm3/chap1?rev=1789594213**

Last update: **2026/09/16 23:30**