



Global Knowledge®

# Formation BIG DATA

## Les fondamentaux

# Sommaire

## 1. Introduction

- a- Les objectifs et les attentes de la formation
- b- De la donnée au Big Data

## 2. Présentation du BIG DATA

- a- Qu'est-ce que le BIG Data
- b- Les trois dimensions du BIG DATA
- c- Les sources de données et les opens data
- d- Améliorer les résultats de l'entreprise grâce au BIG DATA
- e- Use Case et application pour le marketing
- f- Un “nouveau” métier : le Data Scientist
- g- Les algorithmes utilisés en Data Science

# Sommaire

## 4. Big Data vs Business Intelligence

- a- Rappel de l'architecture de la BI et ses limites
- b- Définition du Data Lake
- c- Architecture du BIG DATA
- d- Big Data dans le cloud
- e- Choix de solution Big Data Cloud

# Sommaire

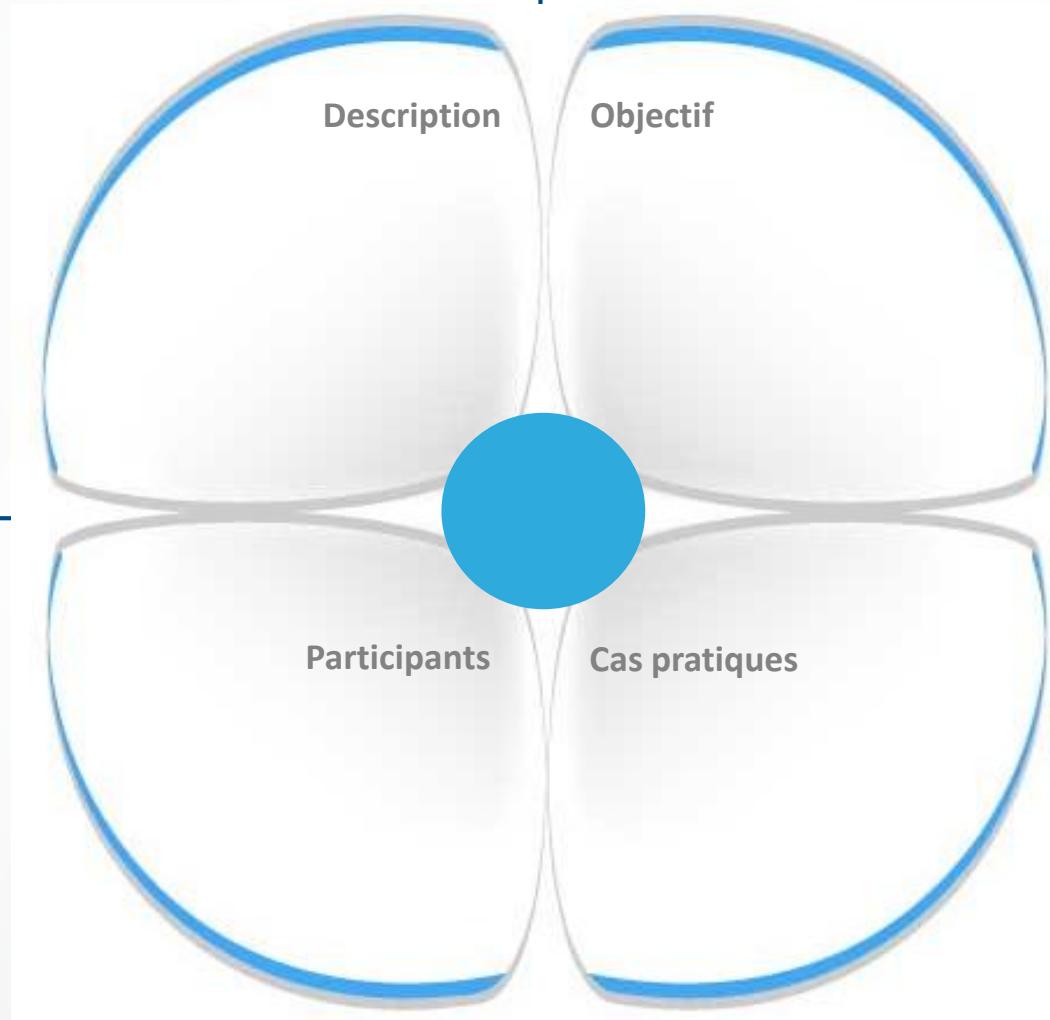
## 4. EchoSystème Hadoop

- a- Qu'est ce que Hadoop
- b- Historique
- c- Les distributions
- d- HDFS
- e- MapReduce, Yarn
- f- HIVE
- g- Sqoop, Flum
- h- NoSQL
- i- HBASE
- j- PIG
- k- Spark
- l- Conclusion

# Formation BIG DATA : Les fondamentaux

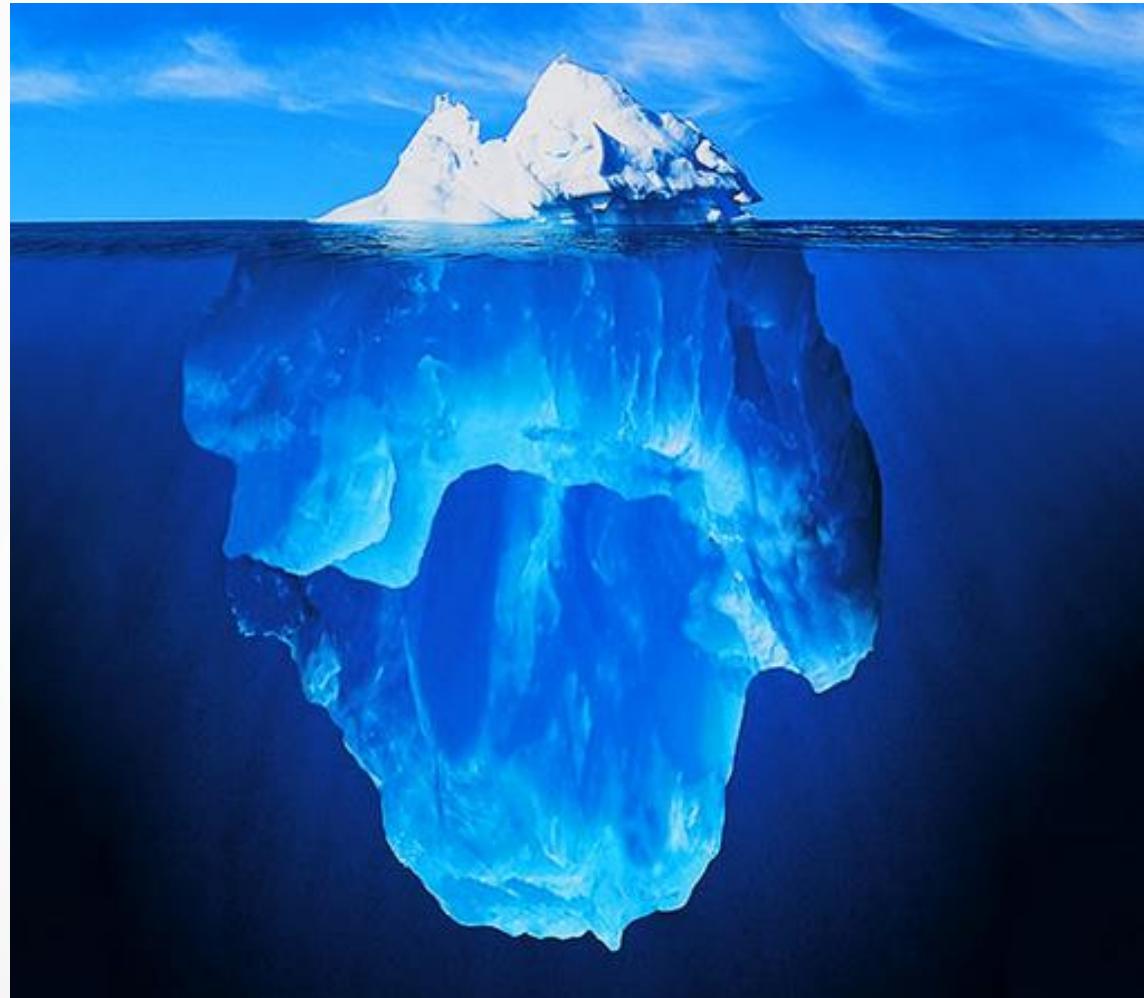
Cette formation présente l'écosystème BIG DATA : Enjeux et Solutions

- ▶ Consultants en analyse décisionnelle
- ▶ Responsables de services
- ▶ Chefs de projet
- ▶ Elèves BI, Data Scientists
- ▶ ..



- ▶ Comprendre les enjeux de BIG DATA
- ▶ Acquérir des connaissances pour exploiter les nouveaux outils dédiés au BIG DATA
- ▶ Apprendre les techniques de :
  - Stockage des informations
  - Traitement & analyse des données
  - Prise de décisions métiers éclairées
- ▶ MapReduce
- ▶ Charger des données dans HDFS
- ▶ Interrogation et Traitement avec Hive
- ▶ Interrogation et Traitement avec HBASE NOSQL

# 80% des données sont inexploitées



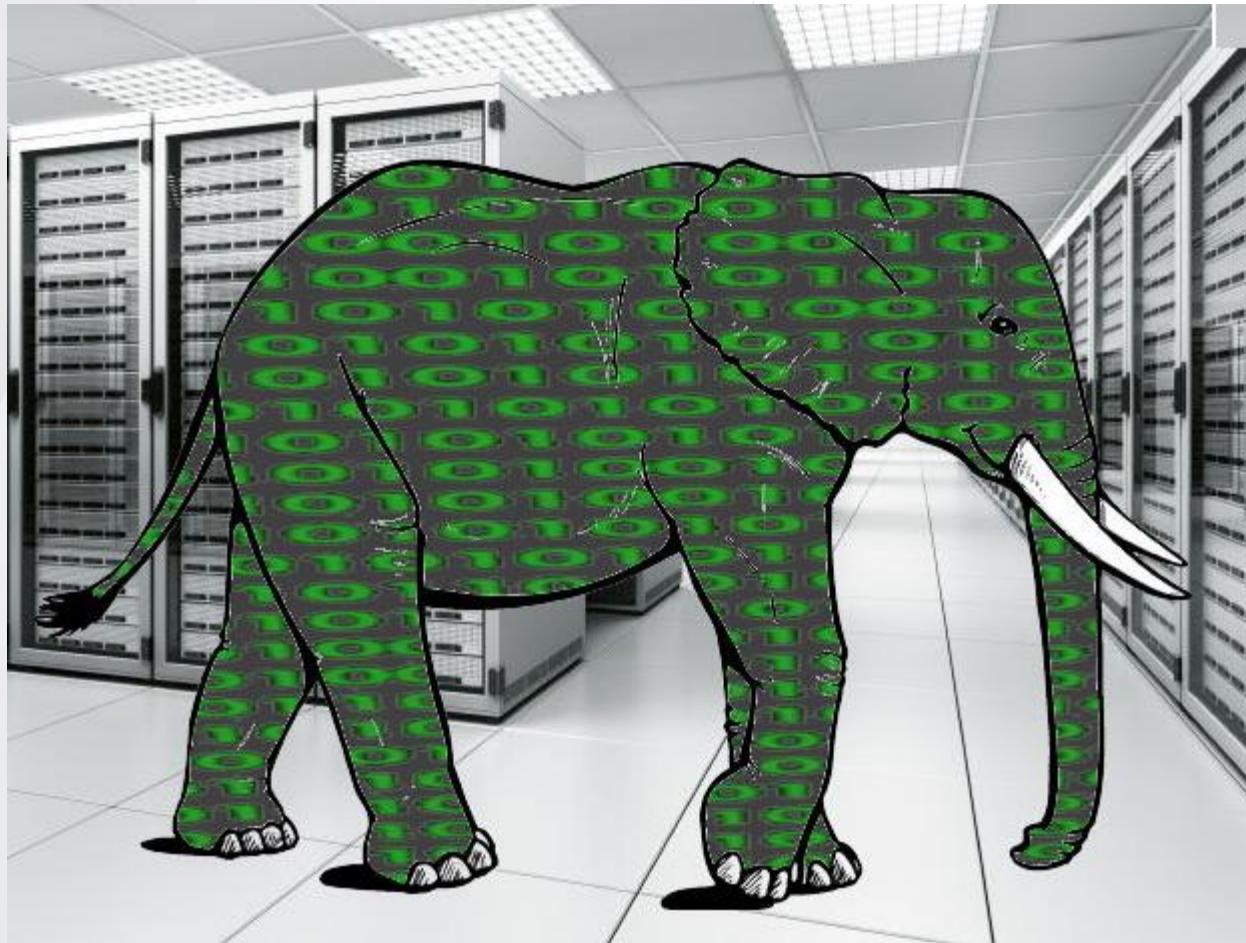
# La data est le nouveau pétrole



# Le déluge de données, beaucoup trop de données



# Nous avons un problème !





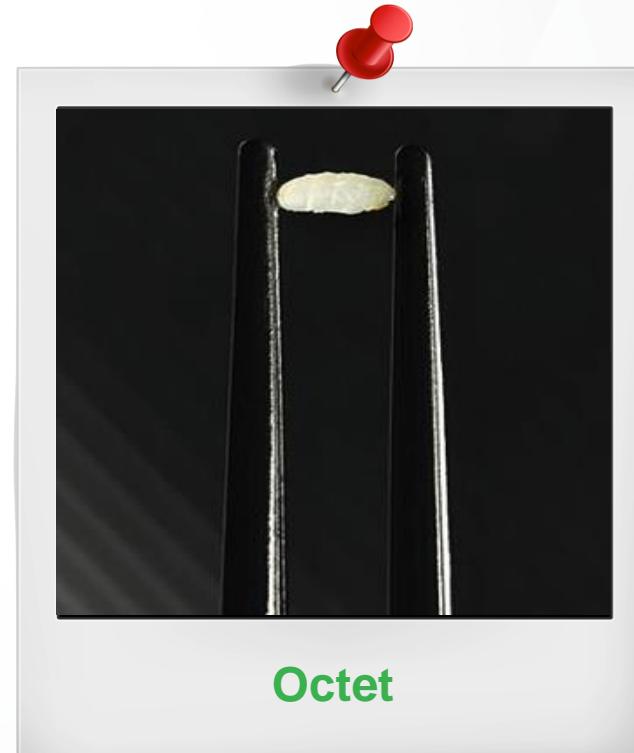
Global Knowledge®



L'histoire de Google  
Et du grain de riz

# De la donnée au Big Data

Octet : Grain de riz



# De la donnée au Big Data

Octet : Grain de riz  
**Kilo-octet : Bol de riz**



Kilo-octet

# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

**Méaoctet : 8 sacs de riz**



**Méaoctet**

# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

Méaoctet : 8 sacs de riz

**Gigaoctet : 3 semi-remorques**



# De la donnée au Big Data

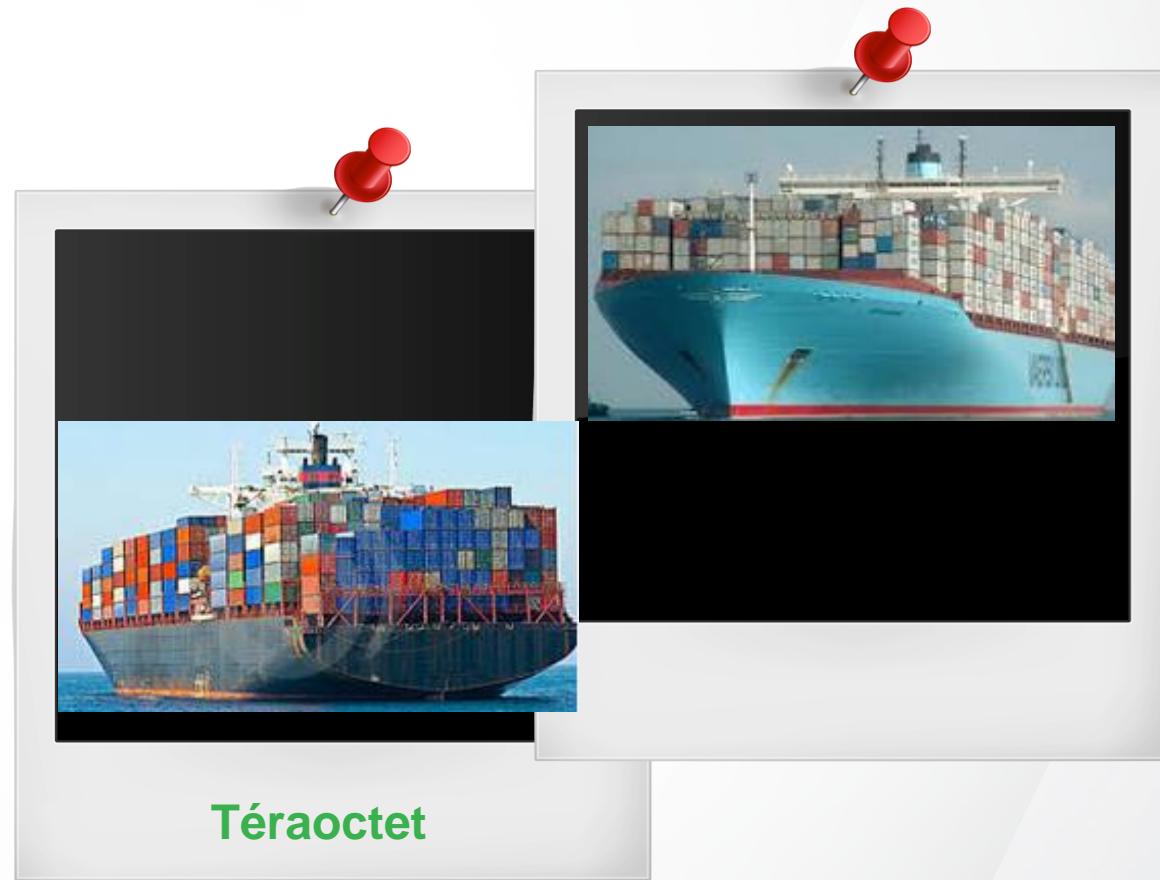
Octet : Grain de riz

Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

Gigaoctet : 3 semi-remorques

**Téraoctet : 2 porte-containers**



# De la donnée au Big Data

Octet : Grain de riz

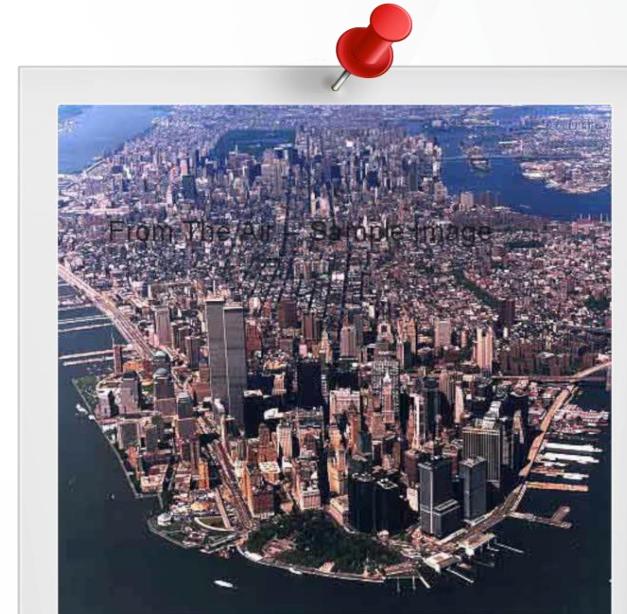
Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

Gigaoctet : 3 semi-remorques

Téraoctet : 2 porte-containers

**Pétaoctet : Manhattan couverte de riz**



**Pétaoctet**

# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

Gigaoctet : 3 semi-remorques

Téraoctet : 2 porte-containers

Pétaoctet : Manhattan couverte de riz

**Exaoctet : Côte ouest des USA couverte de riz**



# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

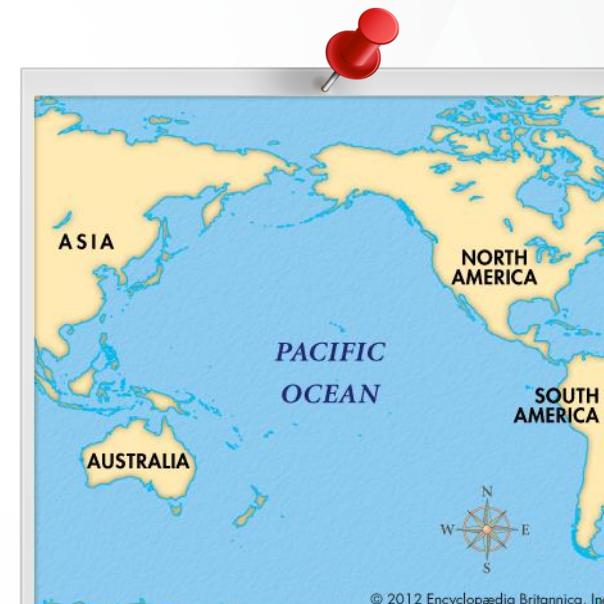
Gigaoctet : 3 semi-remorques

Téraoctet : 2 porte-containers

Pétaoctet : Manhattan couverte de riz

Exaoctet : Côte ouest des USA couverte de riz

**Zétaoctet : couvrir l'océan pacifique de riz**



**Zétaoctet**

# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

Gigaoctet : 3 semi-remorques

Téraoctet : 2 porte-containers

Pétaoctet : Manhattan couverte de riz

Exaoctet : Côte ouest des USA couverte de riz

Zétaoctet : Remplir l'océan pacifique de riz

**Yottaoctet : Remplir la terre de riz**



# De la donnée au Big Data

## Octet : Grain de riz



## Kilo-octet : Bol de riz

## Mégaoctet : 8 sacs de riz

**Gigaoctet** : 3 semi-remorques



## Téraoctet : 2 porte-conteneurs

## Pétaoctet : Manhattan couverte de riz



## Exaoctet : Côte ouest des USA couverte de riz

## Zétaoctet : Remplir l'océan pacifique de riz



## **Yottaoctet : Remplir le volume de la terre de riz**



# De la donnée au Big Data

Octet : Grain de riz

Kilo-octet : Bol de riz

Mégaoctet : 8 sacs de riz

Gigaoctet : 3 semi-remorques

Téraoctet : 2 porte-containers

Pétaoctet : Manhattan couverte de riz

**Exaoctet : Côte ouest des USA couverte de riz**

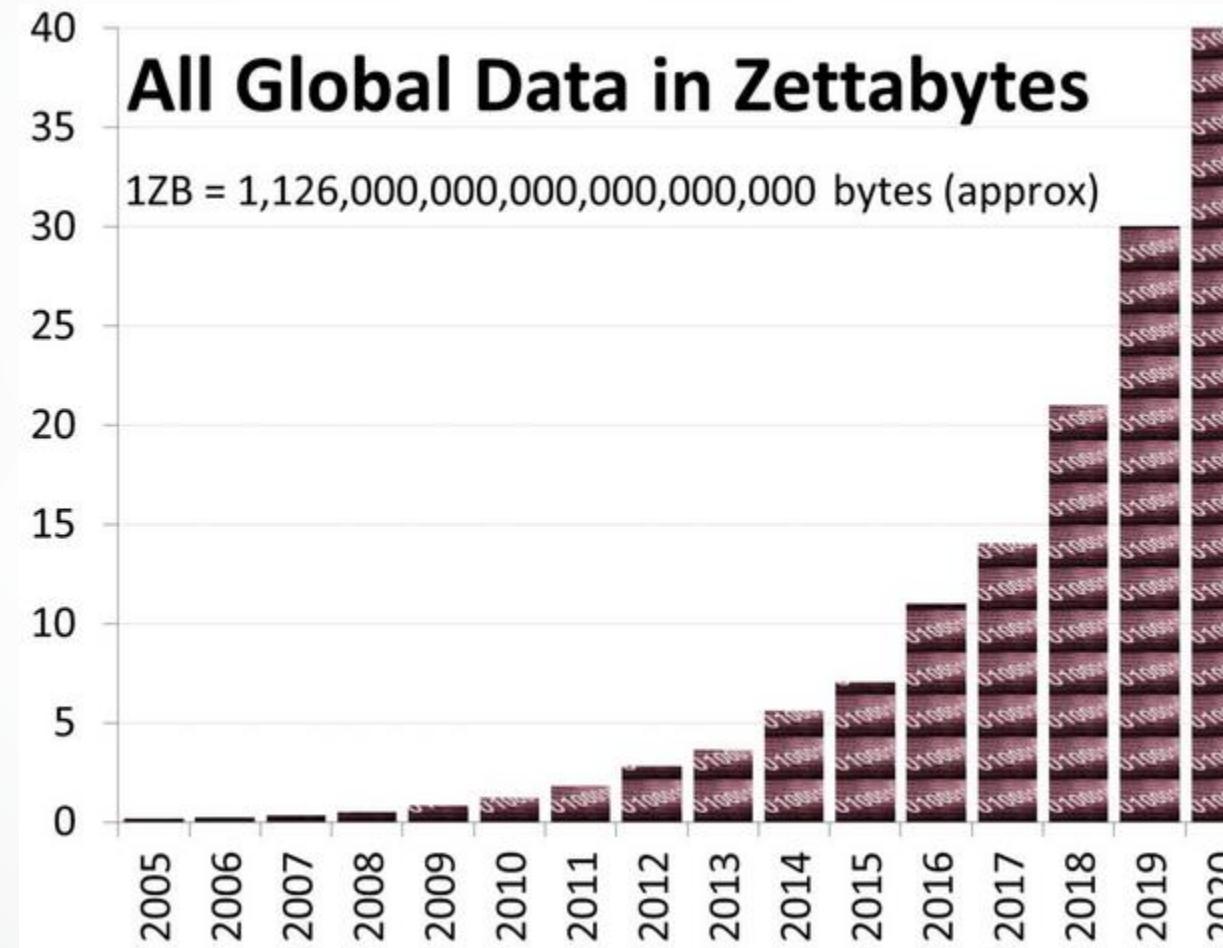
**Zéttaoctet : Remplir l'océan pacifique de riz**

Yottaoctet : Remplir le volume de la terre de riz



*Tous les 2 jours, nous générerons 5  
Exaoctet*

# De la donnée au Big Data





Global Knowledge®



Qu'est-ce que le Big Data ?

# Big Data ?

## **Big Data is like teenage sex**

“Everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it ..” Dan Ariel

# Big Data : définition

Le Big Data désigne **la problématique** d'avoir un ensemble de données à traiter tellement **volumineux** qu'il devient **très difficile**, voire **impossible**, de le faire **avec les outils existants**.

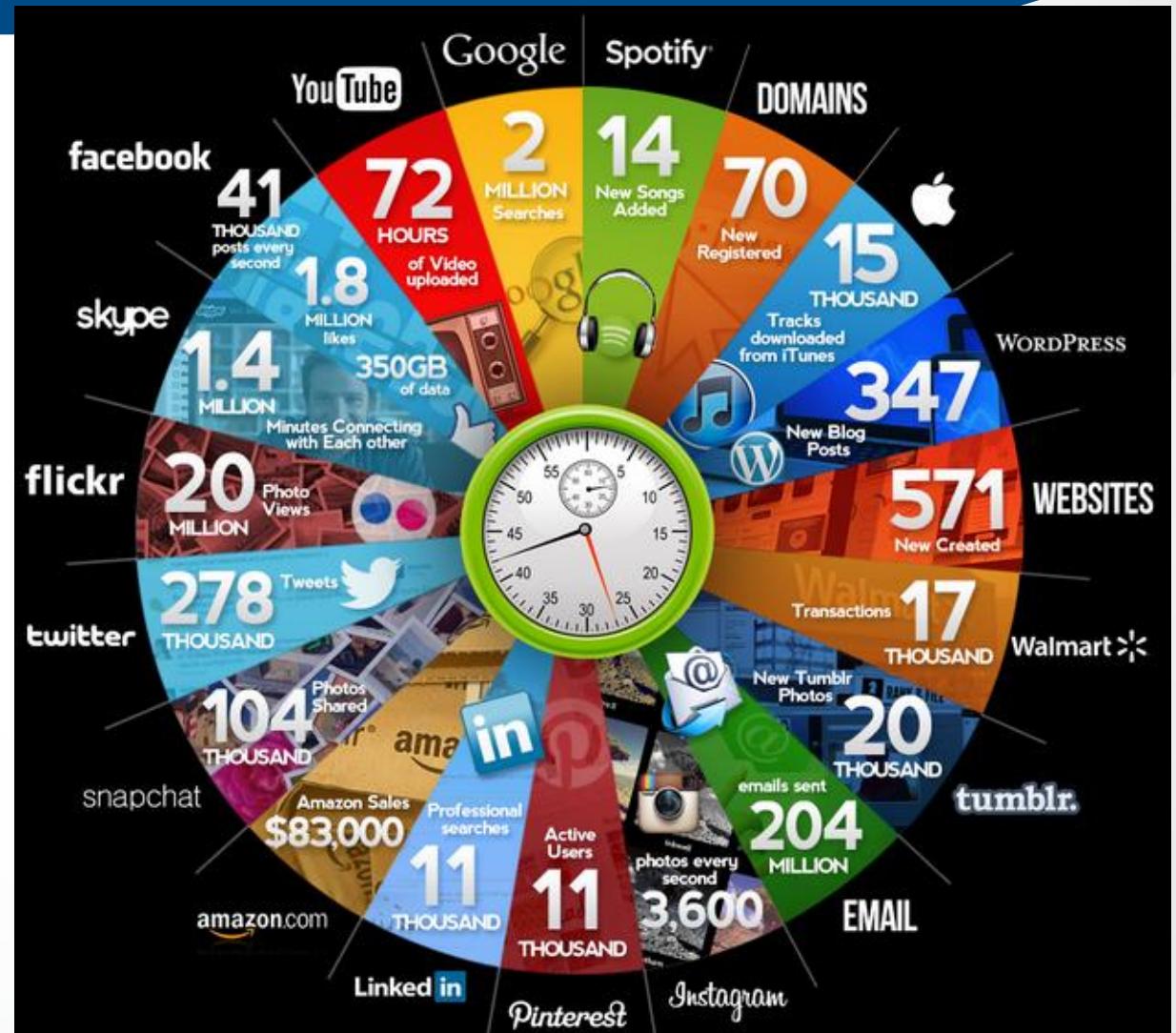
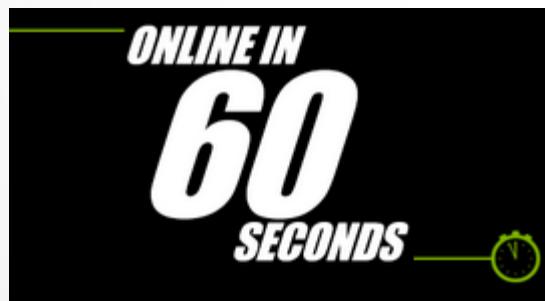
# Big Data : Objectifs

---

- Aider les entreprises à réduire les risques
- Faciliter la prise de décision.
- Créer la différence grâce à l'analyse prédictive.
- Offrir une expérience client plus personnalisée.

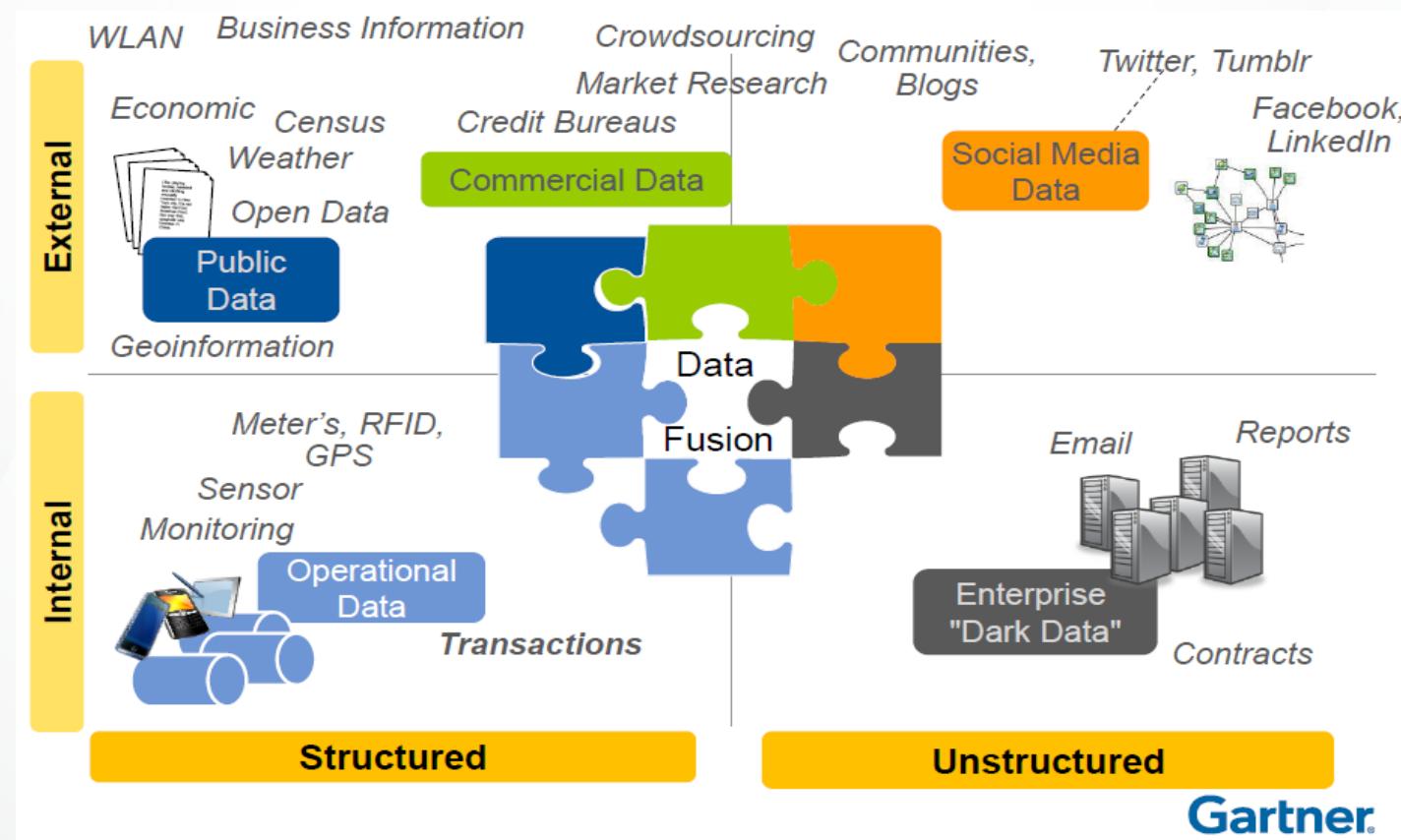
# Les 3 V du Big Data

## 1. Volume de données



# Les 3 V du Big Data

## 2. Variété de données



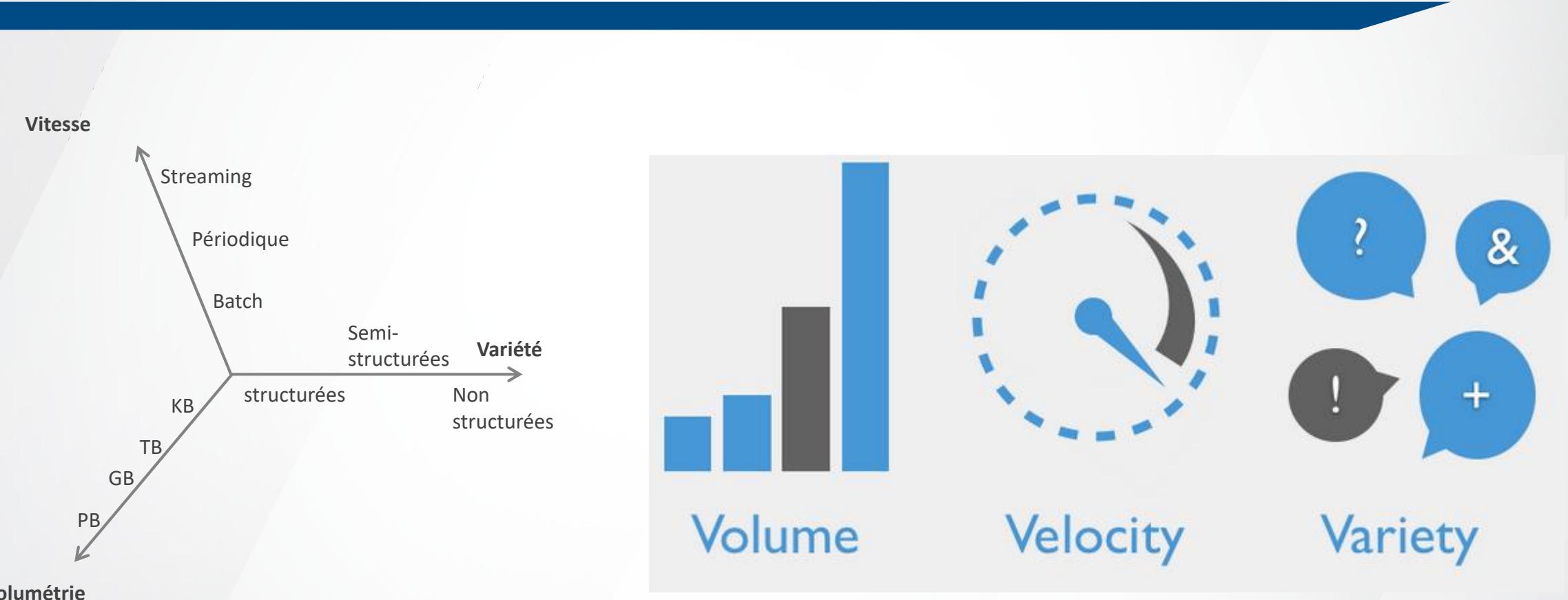
# Les 3 V du Big Data

## 3. Vélocité de données

Production et collecte en temps réel



# Bienvenue dans l'ère du Big Data





Global Knowledge.®



Les sources de données

# Les sources de données

## Les données internes :

- ERP
- CRM
- Billing
- Email
- Log



# Les sources de données

## Web / Réseaux sociaux / Mobile :

- Facebook
- Tweets
- Géolocalisation
- ClickStream



# Les sources de données

## Les données externes :

- INSEE ( Données juridiques )
- Météo
- Entreprises privées



# Les sources de données

## Open data :

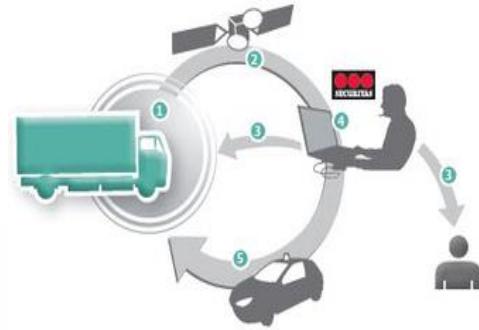
- Données administratives
- Villes
- RTAP
- SNCF
- La poste



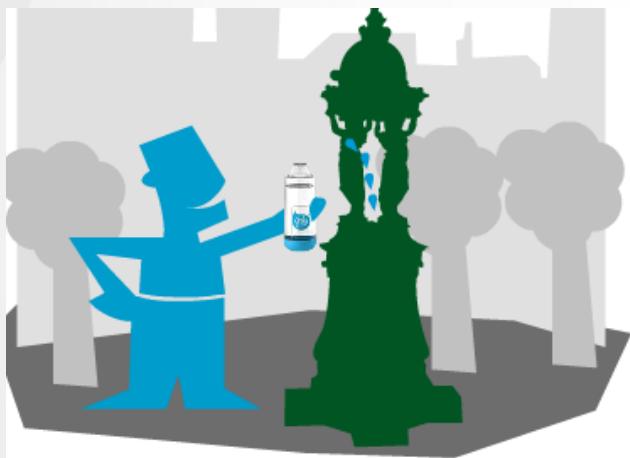
# Les sources de données

## Objets connectés

- 40 milliards d'objet connecté en 2015, 80 milliards en 2020.
- Signaux GPS
- Appli
- Image
- Vidéo
- Capture
- Géolocalisation



# Projets issus de l'open data



#JAISOIF





Global Knowledge®

Le Big Data pour le Marketing

# Marketing : où en sommes-nous ?

En plein Révolution Numérique « Digitalisation »

Phénomène d'Uberisation



# Marketing : où en sommes-nous ?

## Relation client Multicanal



# Marketing : mais il faut savoir aussi !

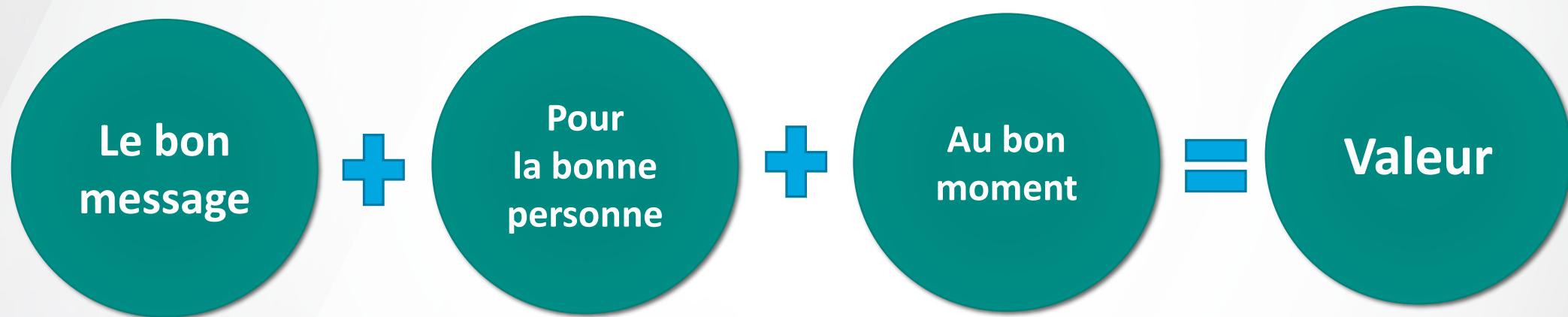
Le big data est la nouvelle monnaie du marketing



Le big data est le meilleur ami des marqueteurs



# Ce qu'apporte le Big Data pour le Marketing

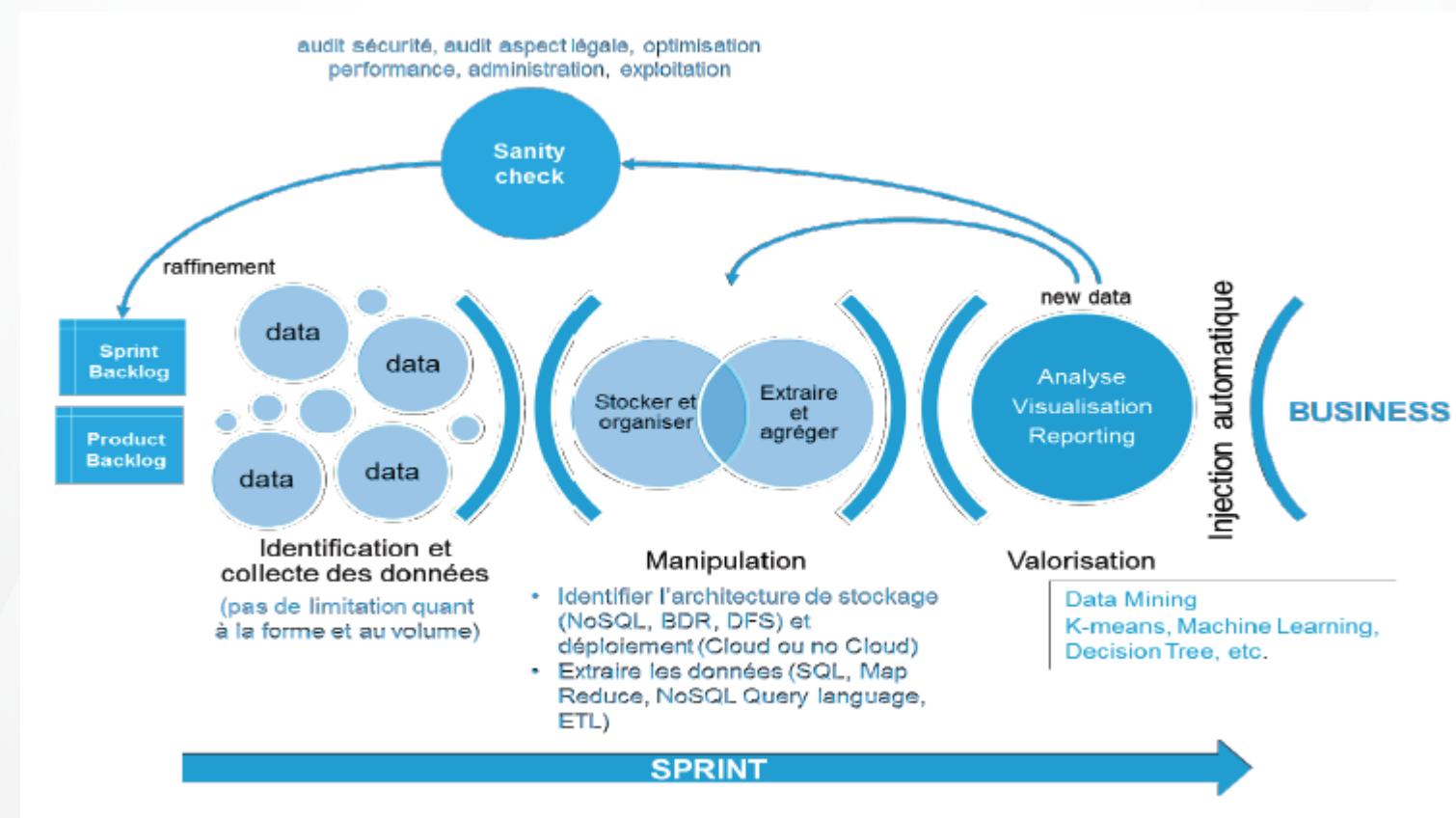


# Big Data en action

- Une approche à la fois technologique et organisationnelle qui rend possible :
  - La collecte
  - Le traitement
  - L'analyse simultanée et en temps réel d'importants volumes de données.
- Afin de mettre en place des campagnes marketing plus personnalisées et performantes.

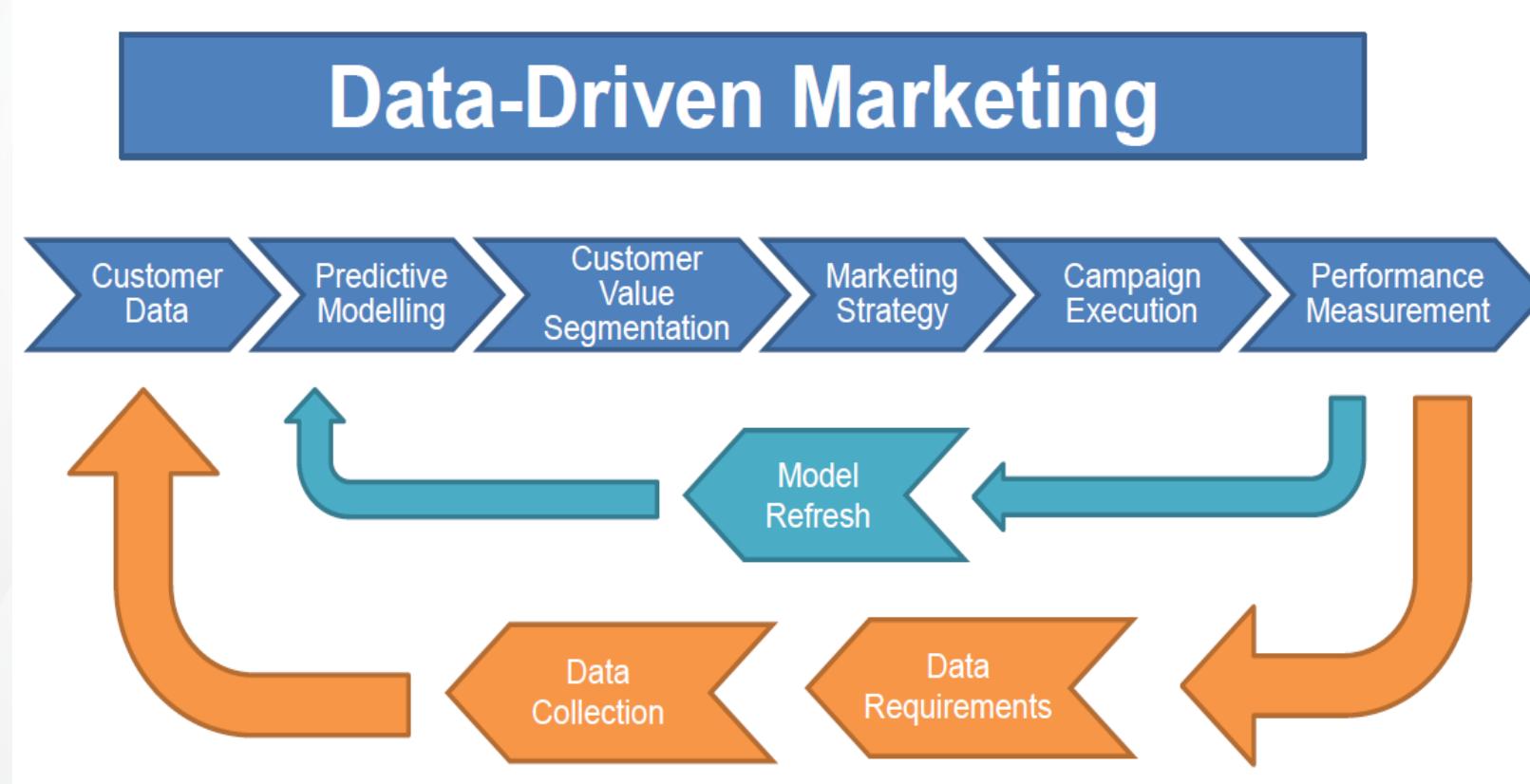
# Big Data en action

## ➤ Une nouvelle démarche



# Big Data en action

## ➤ Data-Driven Marketing





Global Knowledge®



Les cas d'usage

# Cas d'usage

## Industrie



- Produit comme un service
- Qualité, innovation R&D
- Maintenance préventive

## Distribution



- Offres temps réel et service personnalisés
- Optimisation de l'expérience magasin
- Pricing dynamique

## Banques



- Parcours clients multi-canaux
- Fraude, anti blanchiment
- Partage des données consommateurs pour personnalisation

## Assurance



- Fraudes et risques
- Connaissance client, Vision 360°, scoring temps-réel
- Recommandation client
- Tarification personnalisée

## Santé



- Gestion des effets indésirables
- Traitements personnalisés.
- Amélioration des diagnostics

## Transports, loisirs



- Planification et gestion des evts liés à la logistique
- Service client temps réel
- Economie d'énergie
- Pricing dynamique

## Secteur public



- Services informationnels
- Fraudes, abus
- Sécurité publique

## Telecom



- Parcours clients multi-canaux
- Partage de données de géo localisation
- Fraudes et analyse du comportement client

## Produits gde conso.



- Analyse de sentiments et retour produits
- Relation personnalisée avec le consommateur

# Améliorer les résultats de l'entreprise grâce au Big Data

- Une amélioration du chiffre d'affaires grâce au ciblage marketing ;
- Une réduction des coûts grâce à une optimisation des plannings et une diminution des erreurs ;
- Un développement vers des activités innovantes à forte valeur ajoutée ;
- Des gains de parts de marché liées à l'avantage concurrentiel d'être le premier détenteur de ces données stratégiques.

# Les bénéfices du Big Data pour le Marketing 1/2

Le Big data explore déjà de nouveaux usages dans plusieurs spécialités comme :

- Le marketing relationnel avec la segmentation et le ciblage plus fins des clients (les plus rentables, les plus à risque...)
- Le web analytics avec l'optimisation des parcours en ligne.
- Le marketing prédictif avec l'anticipation des besoins et des évolutions.

# Les bénéfices du Big Data pour le Marketing 2/2

Le Big Data va contribuer à transformer le marketing en améliorant ses capacités d'exécution :

- Accroître la réputation de la marque.
- Implémenter des promotions ou des offres ciblées
- Anticiper les comportements plutôt que réagir aux situations.
- Comprendre chaque client dans son unicité.
- Développer les produits de demain.
- Explorer toutes les informations clients disponibles.

# Exemples du Big Data dans le Marketing

## Revenu publicitaire sur son site

- Optimiser la sélection des annonces en fonction des profils
- Identifier et bannir les robots



# Exemple : publicité sur You Tube



# Exemples du Big Data dans le Marketing

## E-commerce

- Identifier les facteurs qui influent (négativement ou positivement) sur une vente.
- Réorganiser le contenu afin de valoriser le plus attrayant ou celui avec le meilleur ROI

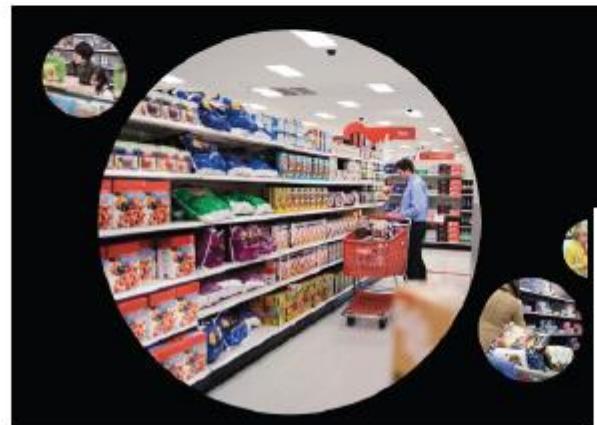


# Exemples du Big Data dans le Marketing

## Connaissance client et fidélisation

- Identifier des comportements d'achat/d'usage, segmenter sa clientèle
- Calculer un score de satisfaction client en fonction de différents indices
- Travailler sur la recommandation.. etc.

# Le Big Data : capitaliser sur la connaissance du client



# Big Data : comment décongestionner le trafic urbain ?



« big data » et les technologies numériques révolutionnent l'organisation de la ville.

GPS

App Géolocalisation

Gestion du trafic



Stationnement

Transport public

Smart City

# Un supermarché apprend la grossesse d'une femme avant son propre père



Comment votre supermarché sait-il que vous êtes enceinte?

- Comment le département marketing du supermarché a développé son outil de prédiction de grossesse :



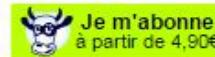
- Analysant les données démographiques et historiques d'achat des millions de clients
- Identifier une liste de 25 produits que les femmes enceintes sont plus susceptibles d'acheter
- **Score de prévisibilité de grossesse**



# Le Big Data au service de Barak Obama

## Le pouvoir du "Big data" : Obama premier Président élu grâce à sa maîtrise de traitement de données ?

Le "Big data", la collecte et le traitement automatisé de gigantesques quantités de données, a aidé Barack Obama à remporter un second mandat. Une méthode déjà utilisée depuis longtemps dans le monde de l'économie et de la finance.



Ajouter au classeur

Suivre ce contributeur

Lecture zen

A<sup>+</sup>

A<sup>-</sup>



# Le Big Data au service de Barak Obama

## Le data mining, l'arme secrète d'Obama pour gagner

Moisés Naím | [Elections-US-2012](#) | [USA 2012](#) | [Monde](#) | 25.10.2012 - 18 h 32 | mis à jour le 25.10.2012 à 18 h 32



# Le Big Data au service de Barak Obama

## ➤ Campagnes ultra-ciblées



# Le Big Data au service de Barak Obama

## 1. Collecte des données

Collecter des informations sur n'importe quelle personne qui interagit avec la campagne

- Les données des militants (porte à porte, meetings, conventions, marchés..).
- Les donateurs, les bénévoles, les électeurs, les internautes.
- Les réseaux sociaux
- Les données publiques, les données privées
- « Phoning » et « mailing » se comptent en millions



# Le Big Data au service de Barak Obama

## 2. Traitement des données

### Datamining, algorithme et analyse prédictive

- Repérer les zones abstentionnistes.
- Identifier les personnes favorables ou défavorables au candidat selon ses promesses.
- Identifier les potentiels donateurs.
- Segmentation



# Lancement de la campagne

## 3. Stratégie

### Mise en place des actions

- Ouverture des bureaux dans les quartiers ciblés.
- Mailing, publicités optimisés et ciblés ( donateurs, bénévoles..)
- Action micro-ciblée sur le terrain le jour de scrutin



# Résultat

Baraka Obama a gagné les élections de 2012 grâce à la Big Data :

- Obtention de 1 million de dollar auprès des donateurs.
  - 700.000 postes bénévoles dans les 4 derniers jours





Global Knowledge®



Un « nouveau » métier  
le Data Scientist

# Un « nouveau » métier : le Data Scientist

Au vu des besoins, un “nouveau métier” va exister : **Data Scientist.**

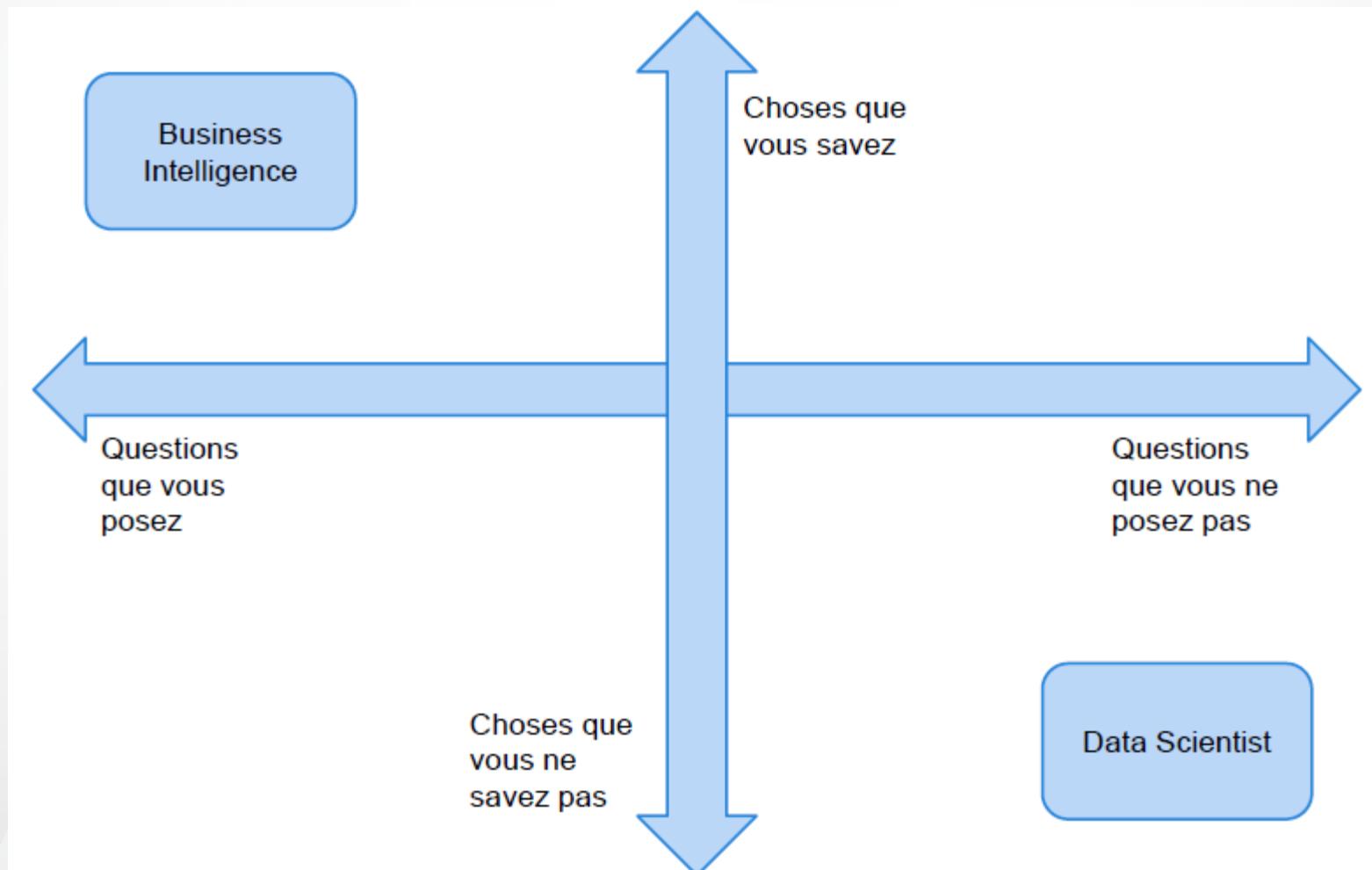
Josh Wills le définit de la façon suivante : *Une personne qui est meilleure en statistiques que n'importe quel développeur et qui est meilleure en développement que n'importe quel statisticien.*

# Un « nouveau » métier : le Data Scientist

## CV :

- Statistiques, Probabilité, Machine learning.
  - Connaissances en développement logiciel.
  - Connaît le métier de l'entreprise.
  - Capacité de présentation et d'imagination.
- 
- Hadoop, HDFS...
  - Spark, MapReduce
  - Java, R, Python...
  - ETL...
  - SQL, Excel...

# Un « nouveau » métier : le Data Scientist





Global Knowledge®



Les algorithmes utilisés

# La classification

- La **classification** consiste à prédire, pour chaque individu d'une population, à quelle classe cet individu appartient.

Exemple : “parmi mes clients, lesquels pourraient répondre à une offre spécifique ?”. Dans cet exemple il y aura deux classes “répondra” et “ne répondra pas”.

# La régression

**La régression** (estimation de valeur) essaye d'estimer ou de prédire, pour chaque individu la valeur numérique de certaines variables de cet individu. Exemple : “A quel point ce client utilisera ce service ?”.

Dans cet exemple, la valeur de l'usage du service sera générée en regardant l'usage du service par d'autres individus similaires. Par rapport à la classification qui prédit si quelque chose va se produire, la régression permet de dire à quel point la chose va se prédire.

# La recherche de similarité

---

**La recherche de similarité** essaye d'identifier des individus similaires à partir des éléments que l'on a sur eux.

Par exemple, IBM utilise cette technique pour trouver des compagnies similaires à leurs clients les plus rentables afin que leurs commerciaux se concentrent sur eux.

# Le Clustering

Le Clustering essaye de classifier des individus par leurs similarités mais sans tenir compte de leurs attributs. Par exemple, pour savoir si il y a des segments ou des groupes de clients similaires.

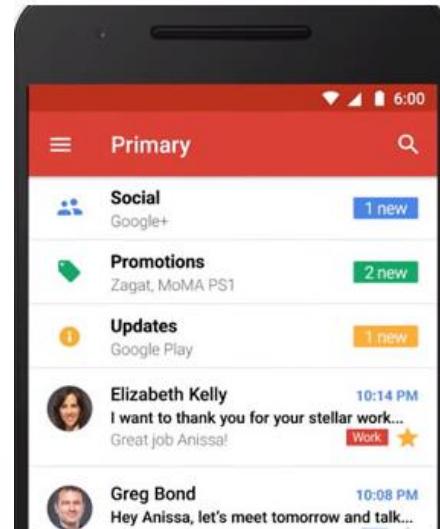
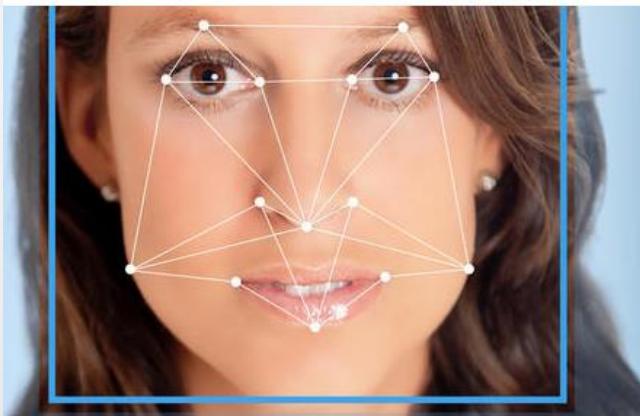
Cela permet de faire de l'exploration afin de voir vers quels types d'analyses nous devons faire.

# Le Profilage

**Scoring/profilage** essaye de caractériser un comportement typique d'un individu, d'un groupe ou d'une population. Par exemple "Quel est l'usage type de leur téléphone de tel segment de clientèle ?".

Le profilage est souvent utilisé pour la détection d'anomalies (fraudes, intrusions...) Par exemple, si l'on sait quels genres d'achats une personne fait généralement via une carte de crédit, on peut déterminer si tel ou tel achat correspond.

# Quelques exemples pratiques de Machine Learning





Global Knowledge.®

Qu'est-ce que cela change ?

# Qu'est-ce que cela change ?

---

**Scénario** : je dirige une chaîne de magasins qui vend des jeux vidéos.

La saison de Noël approche et ma réussite dépend d'une chose principalement : Avoir assez de stocks, au bon endroit, sur les produits qui vont le mieux se vendre.

Nous sommes quelques mois avant noël.

# Qu'est-ce que cela change ?

---

Quel est l'objectif ? Savoir ce qui va se vendre, où et dans quelles quantités.

Je vais utiliser les données suivantes :

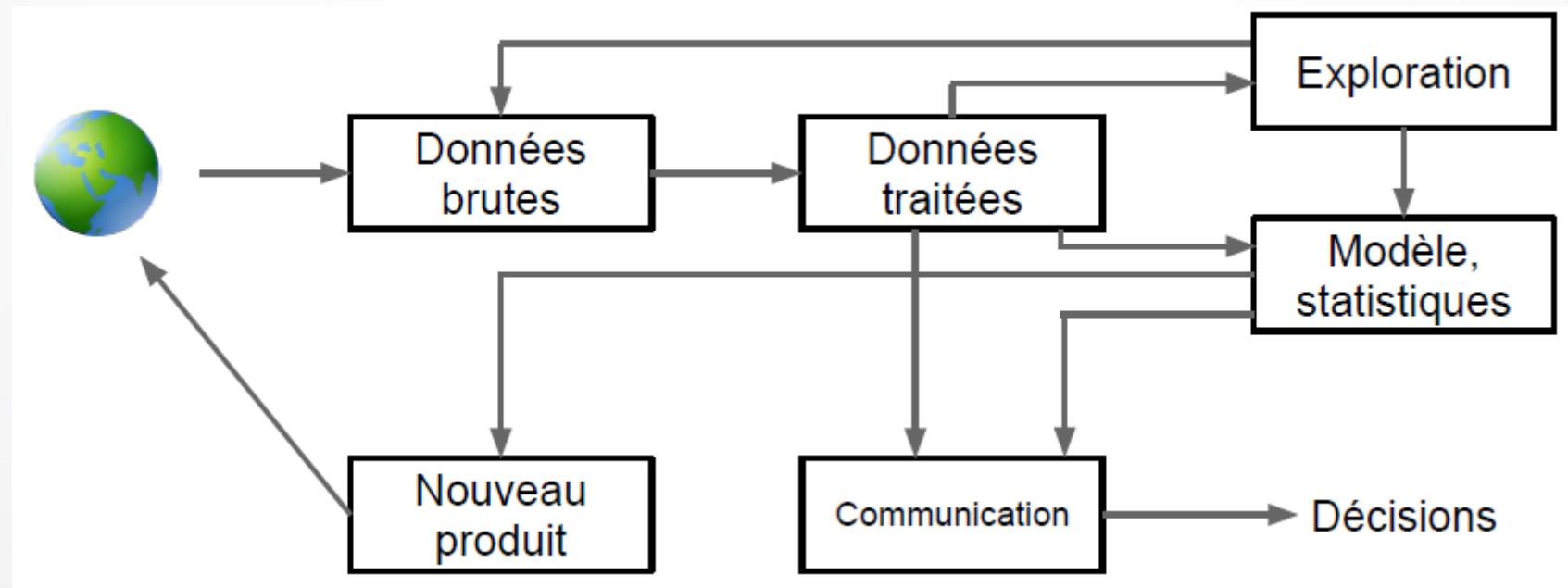
- Recherches google.
- Tweets qui parlent de jeux vidéos.
- Budgets dépensés par l'industrie du jeu.
- Tests de remises sur des joueurs “types” qui sont parmi mes clients.

# Qu'est-ce que cela change ?

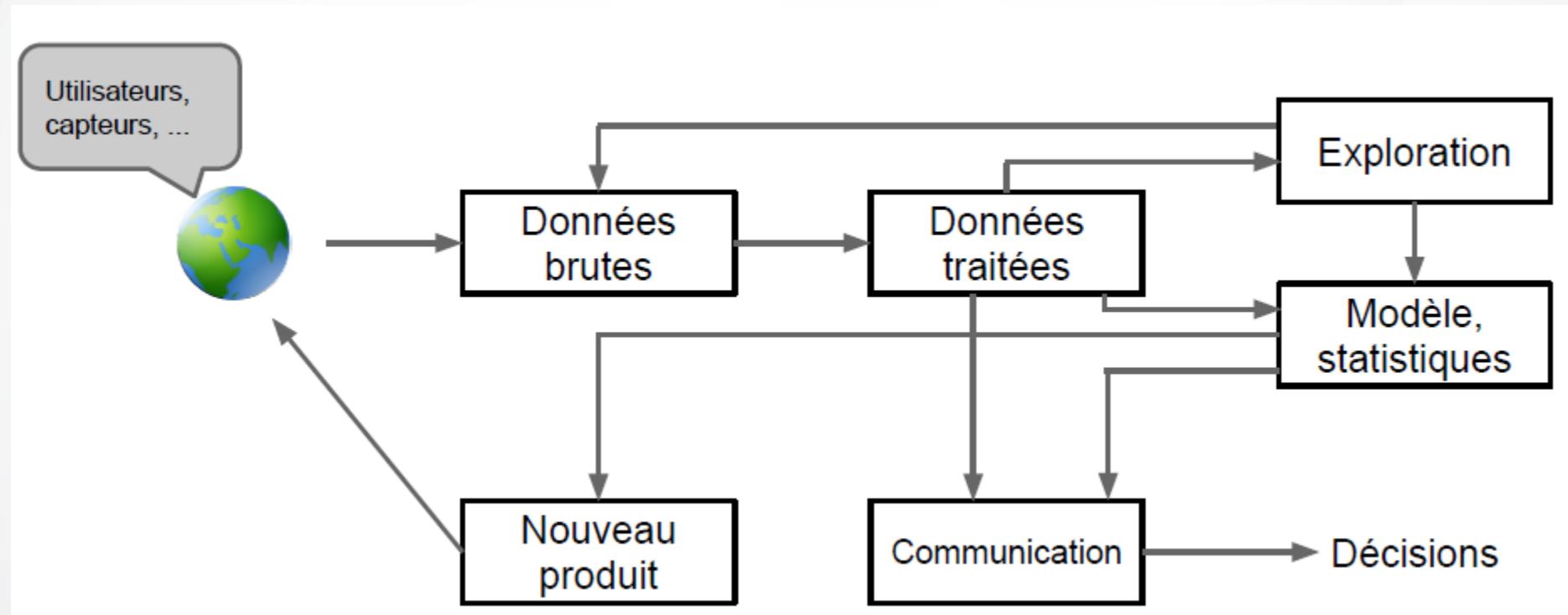
Grâce aux analyses que j'ai, je vais pouvoir :

- Identifier les jeux qui semblent avoir le plus de succès et donc piloter mes achats.
- Mieux gérer mes commandes/stocks.
- Mieux gérer mes campagnes commerciales.
- Resegmenter ma base en fonction des précommandes et faire des contacts ciblés.

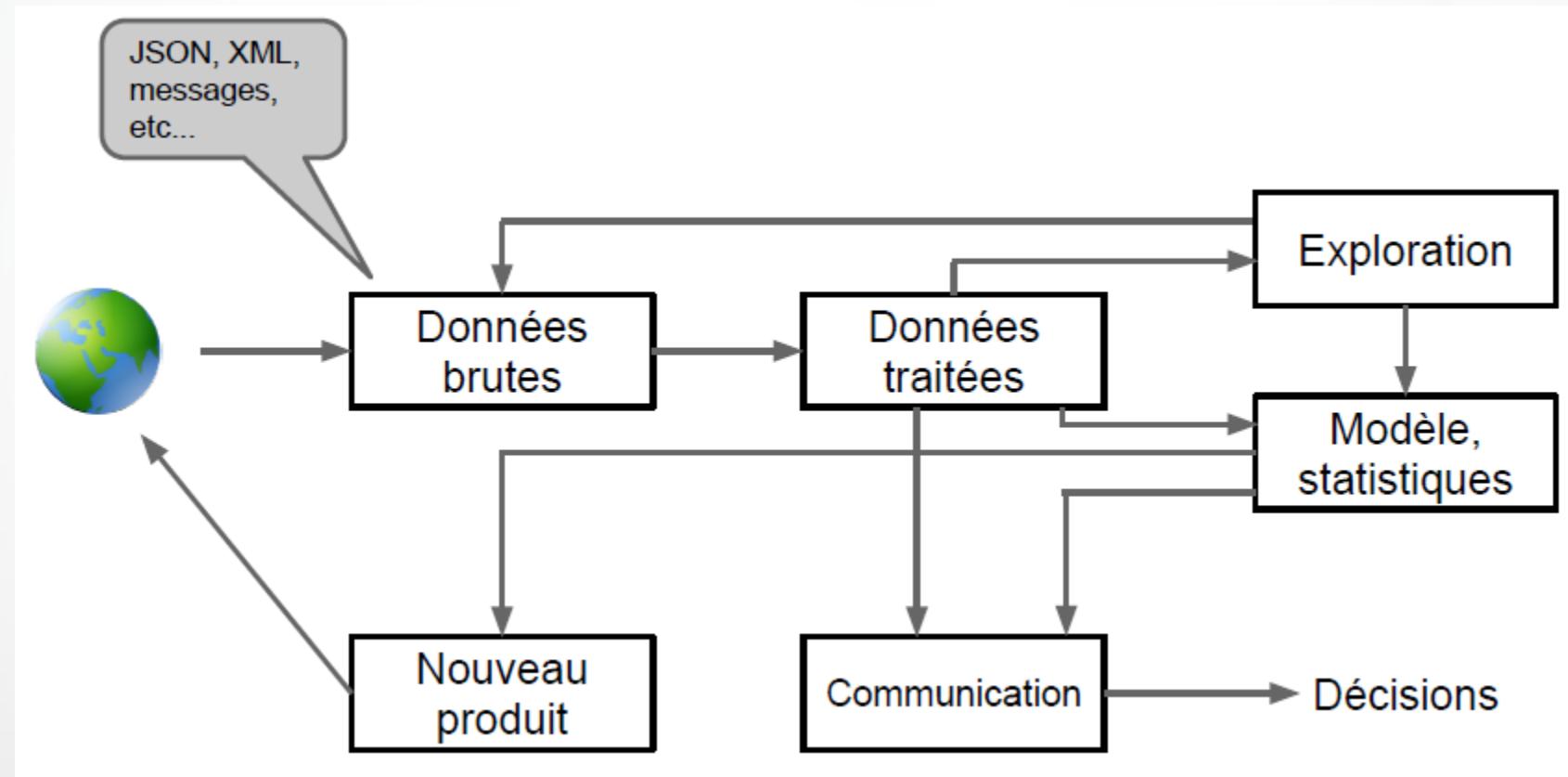
# Data Science : exemple



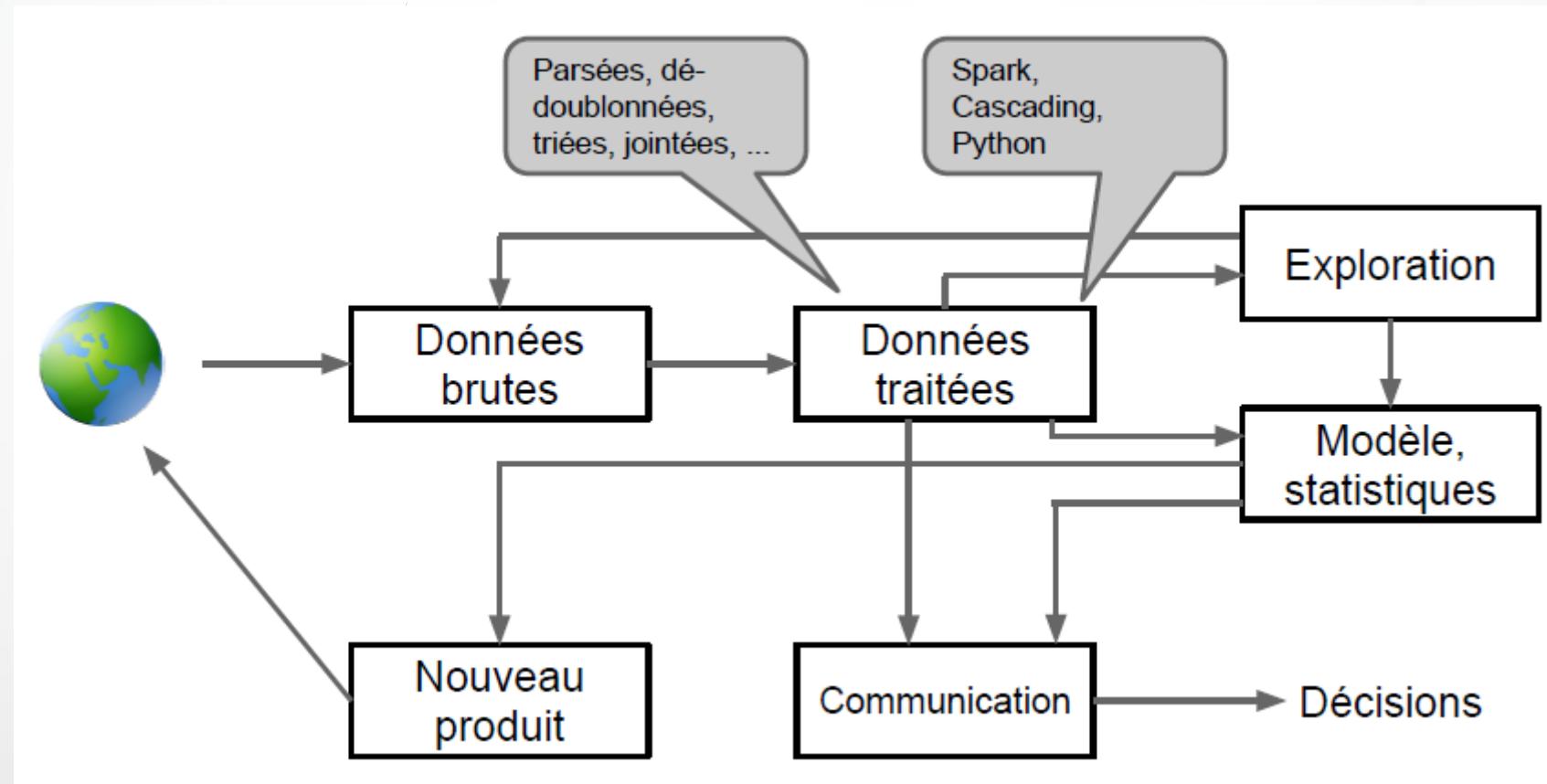
# Data Science : exemple



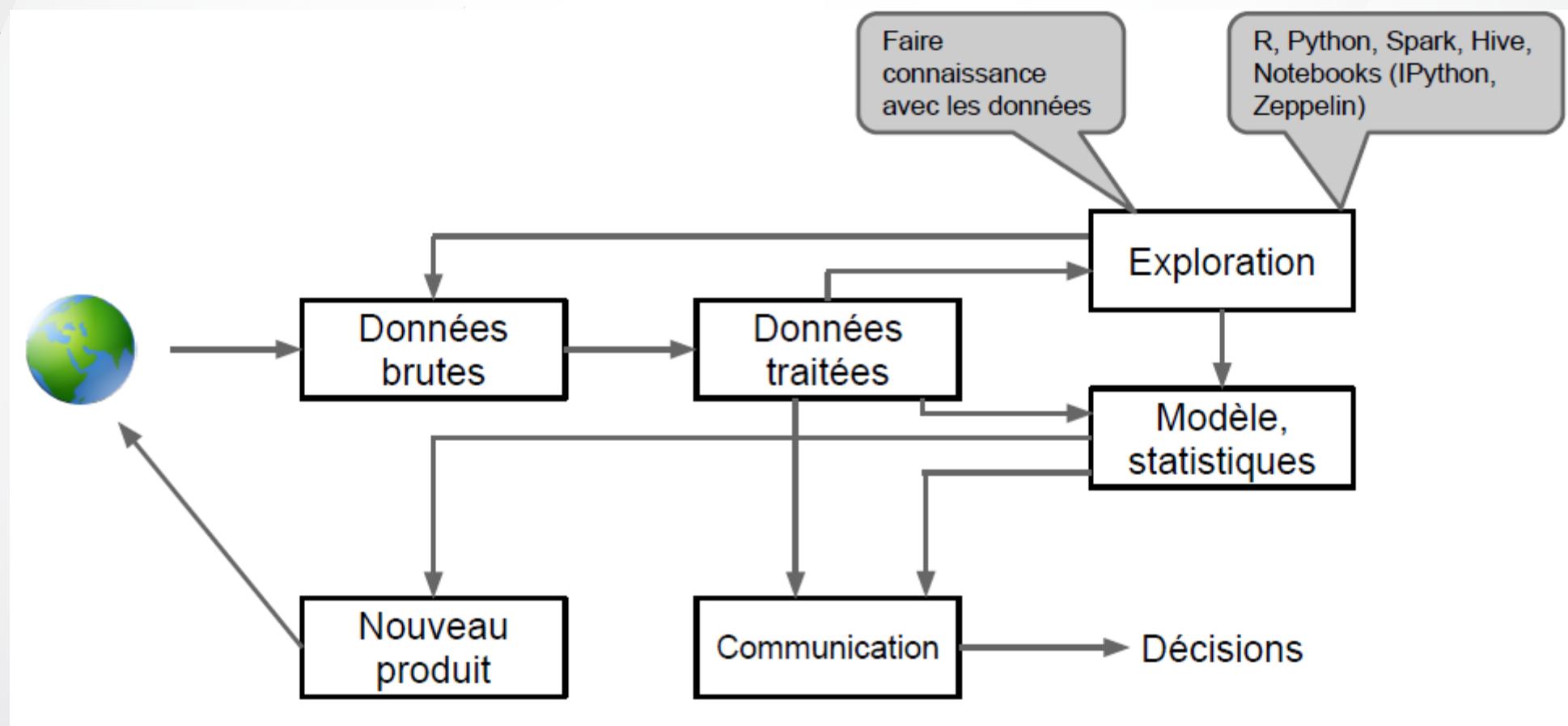
# Data Science : exemple



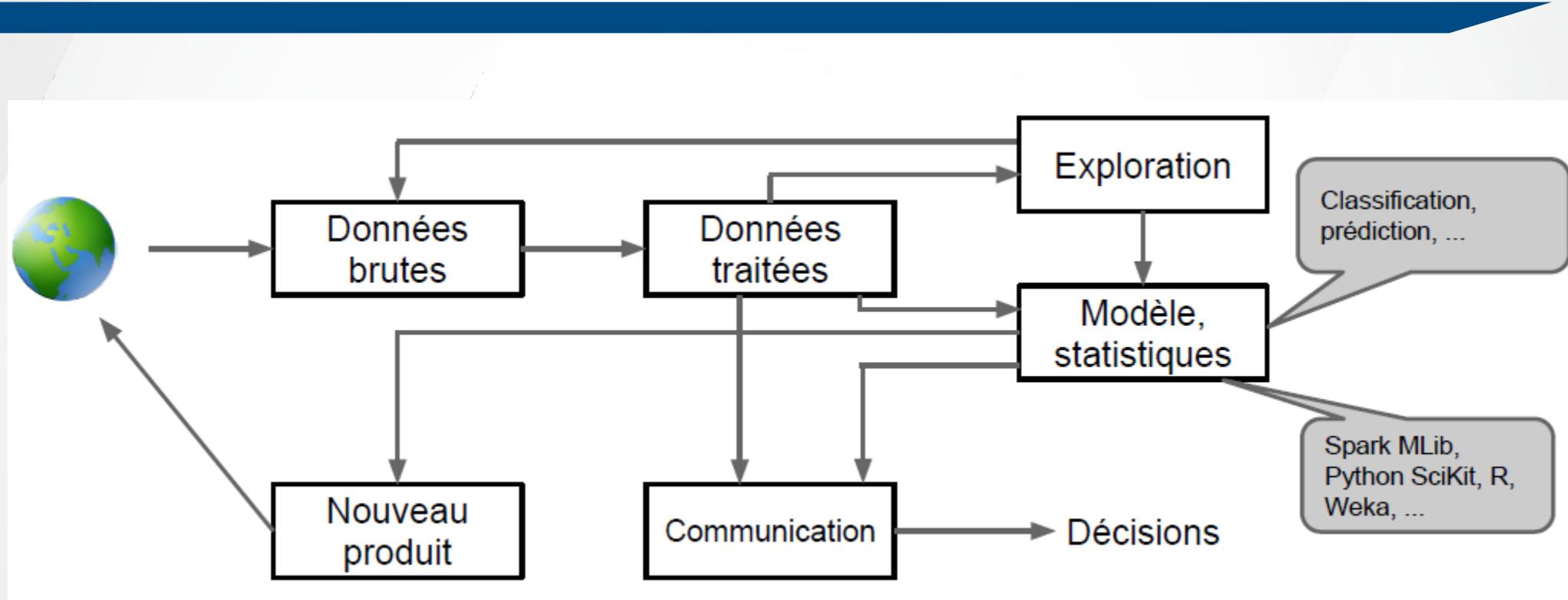
# Data Science : exemple



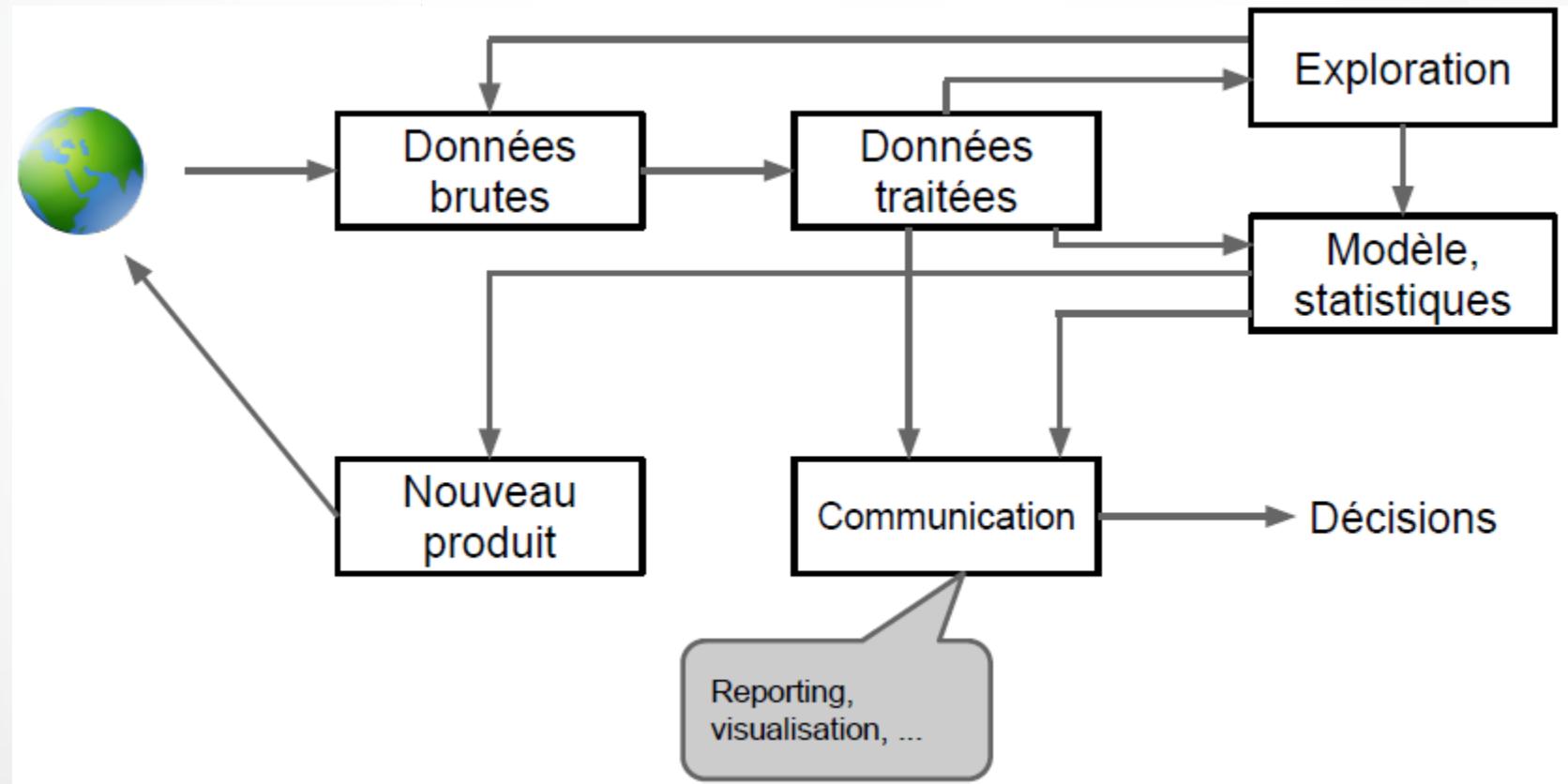
# Data Science : exemple



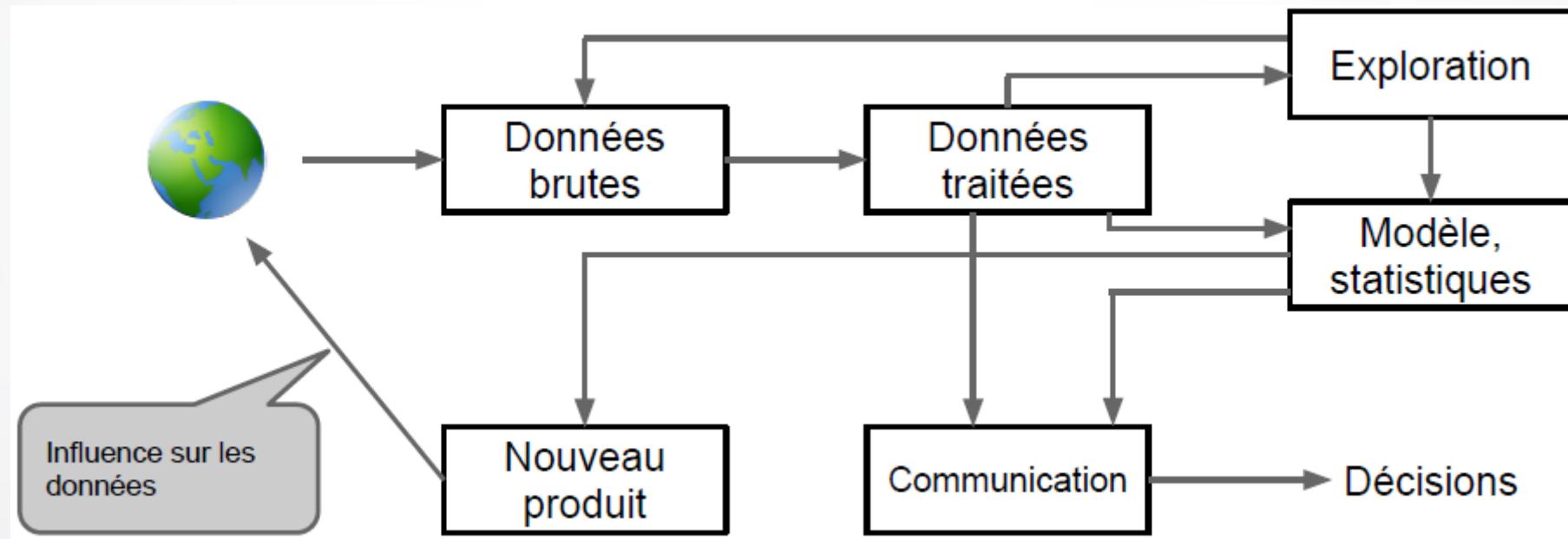
# Data Science : exemple



# Data Science : exemple



# Data Science : exemple

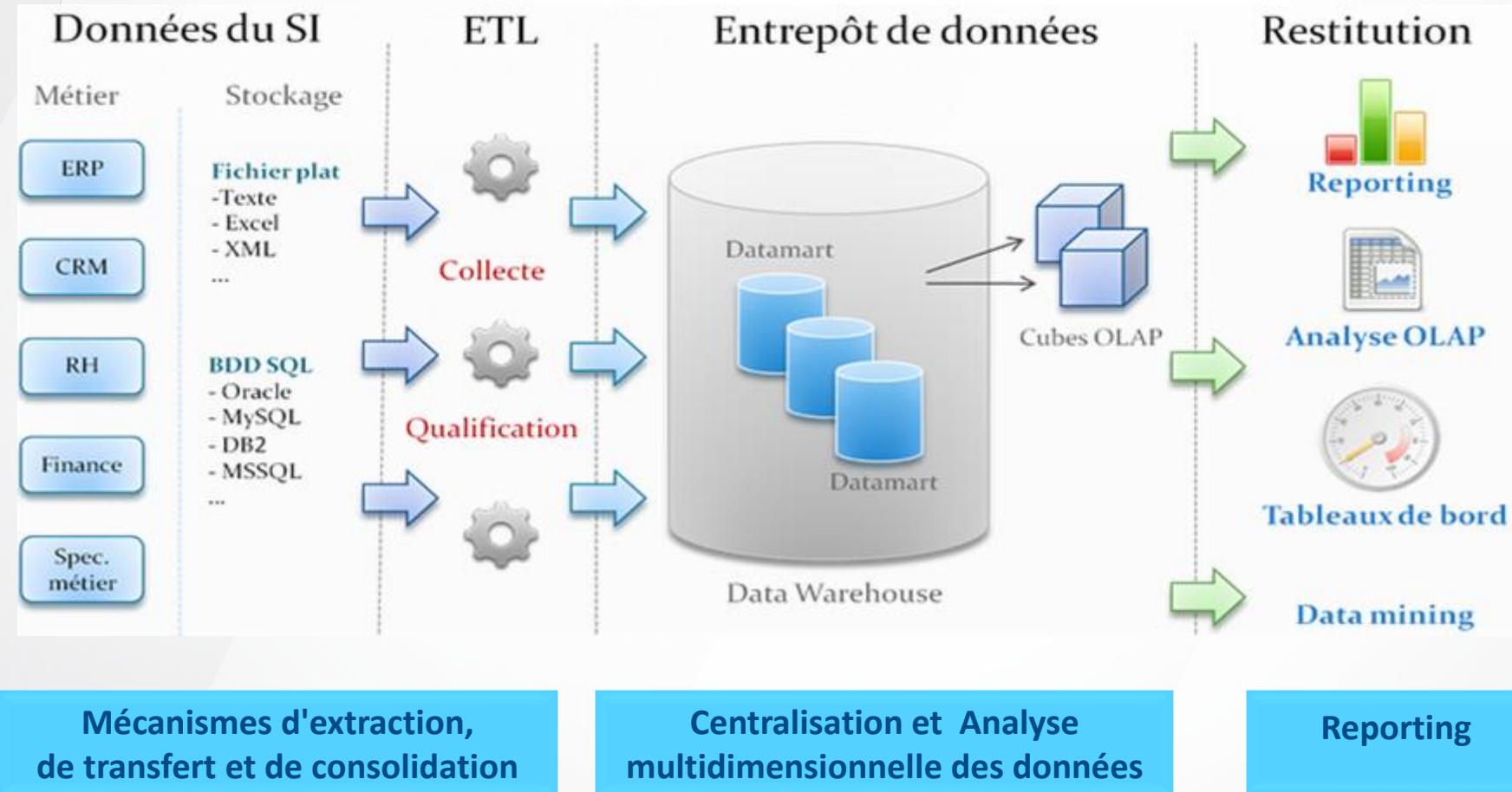




Global Knowledge®

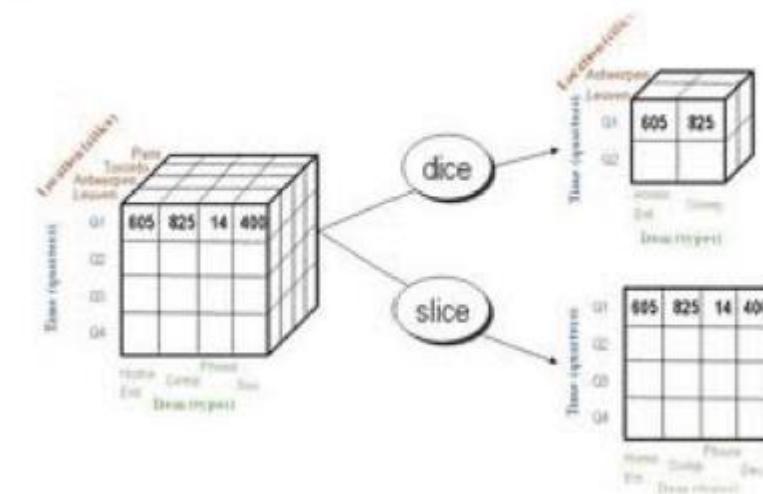
La Business Intelligence

# Architecture de la BI traditionnelle



# La BI traditionnelle

- Les analystes veulent couper leur données en « **slice et dice** »
- Les analystes ont besoin d'accéder à des données **très agrégées**.
- Les analystes veulent parcourir les données en **profondeur** (aller du général vers les détails)



# La BI traditionnelle : ses limites

- L'analyse des données en **temps réel** est **impossible**.
- Les données non structurées ne peuvent être **stockées** dans ces bases de données.
- Ces bases de données **n'adaptent pas leur capacité de stockage en fonction de la montée en charge** en fonction du volume.





Global Knowledge®

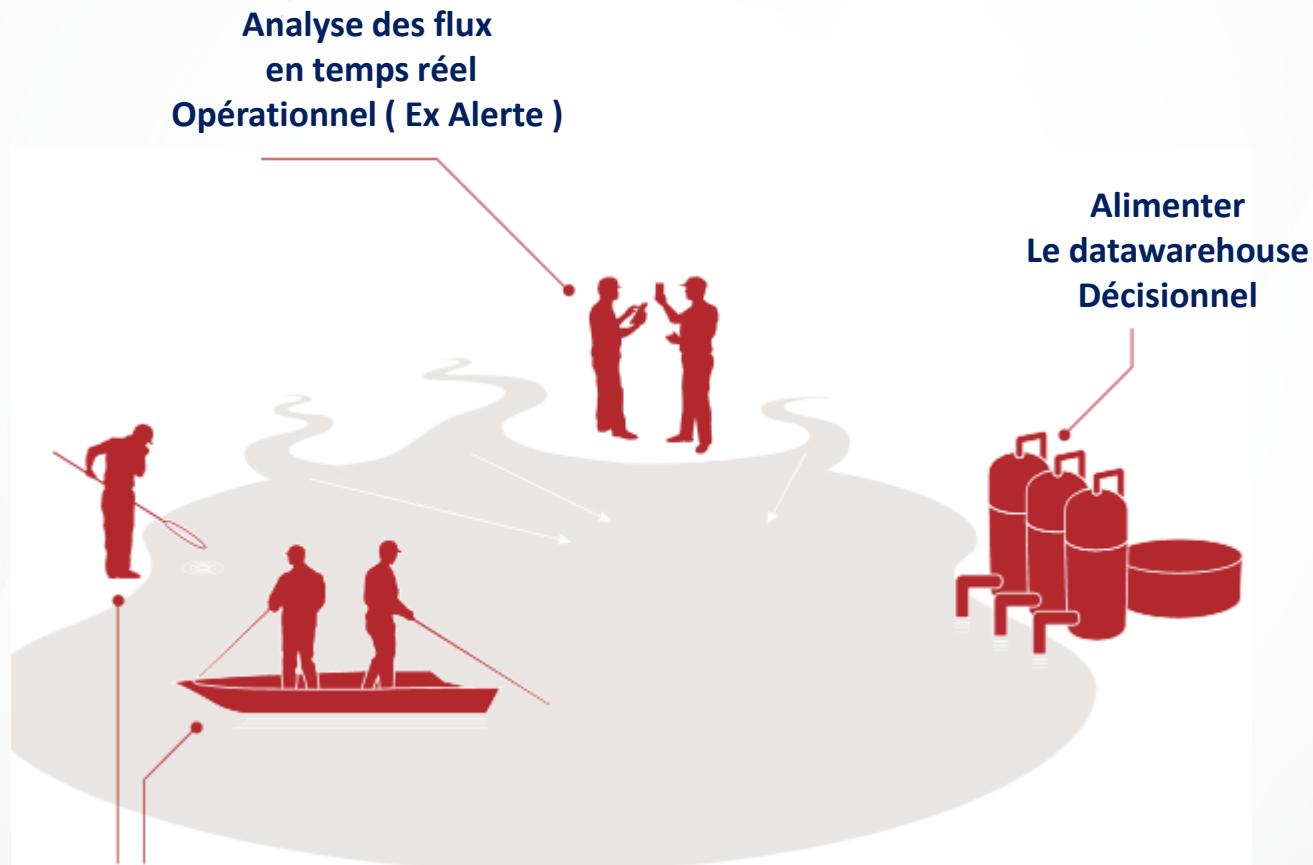
Le Data Lake

# Big Data : Data Lake

“If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.”

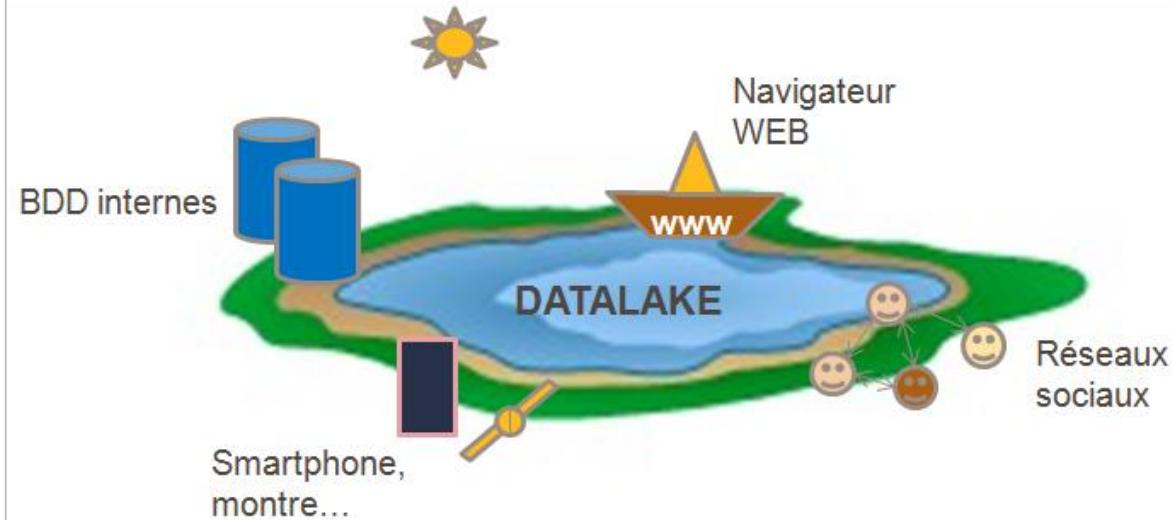
# Big Data : Data Lake

Les Data Scientists  
explore les données  
pour découverte  
et idéation  
Big Data



# Big Data : Data Lake

- Regrouper et exploiter ces données de manière simple sur une même et unique plateforme.
- Mettre à disposition toutes les données les rendant accessibles depuis un même endroit.
- Le Data Lake est en perpétuelle extension.
- Simplicité de croiser les données et de les enrichir.
- Pour le stockage des données volumineuses et les persister, ce Data Lake peut reposer sur Hadoop File System (HDFS) sur un cluster de Machines.



# Big Data : Data Lake

➤ ... mais pourquoi tout stocker sans savoir pourquoi ?

**#1 : ça peut servir un jour. En plus, ils le font tous dans la Silicon Valley.**

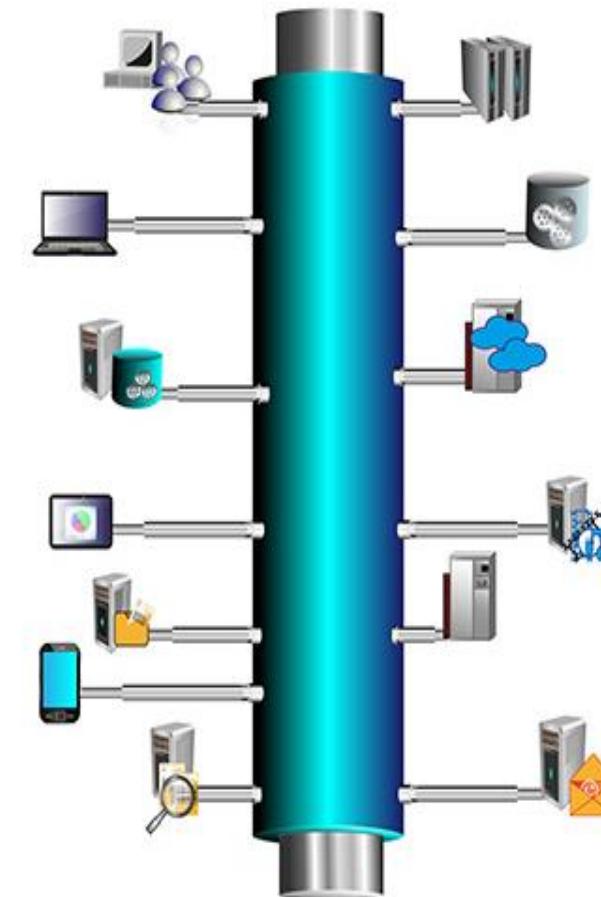


**#2 : et puis, vu ce que ça coûte maintenant, ce serait dommage de s'en priver.**

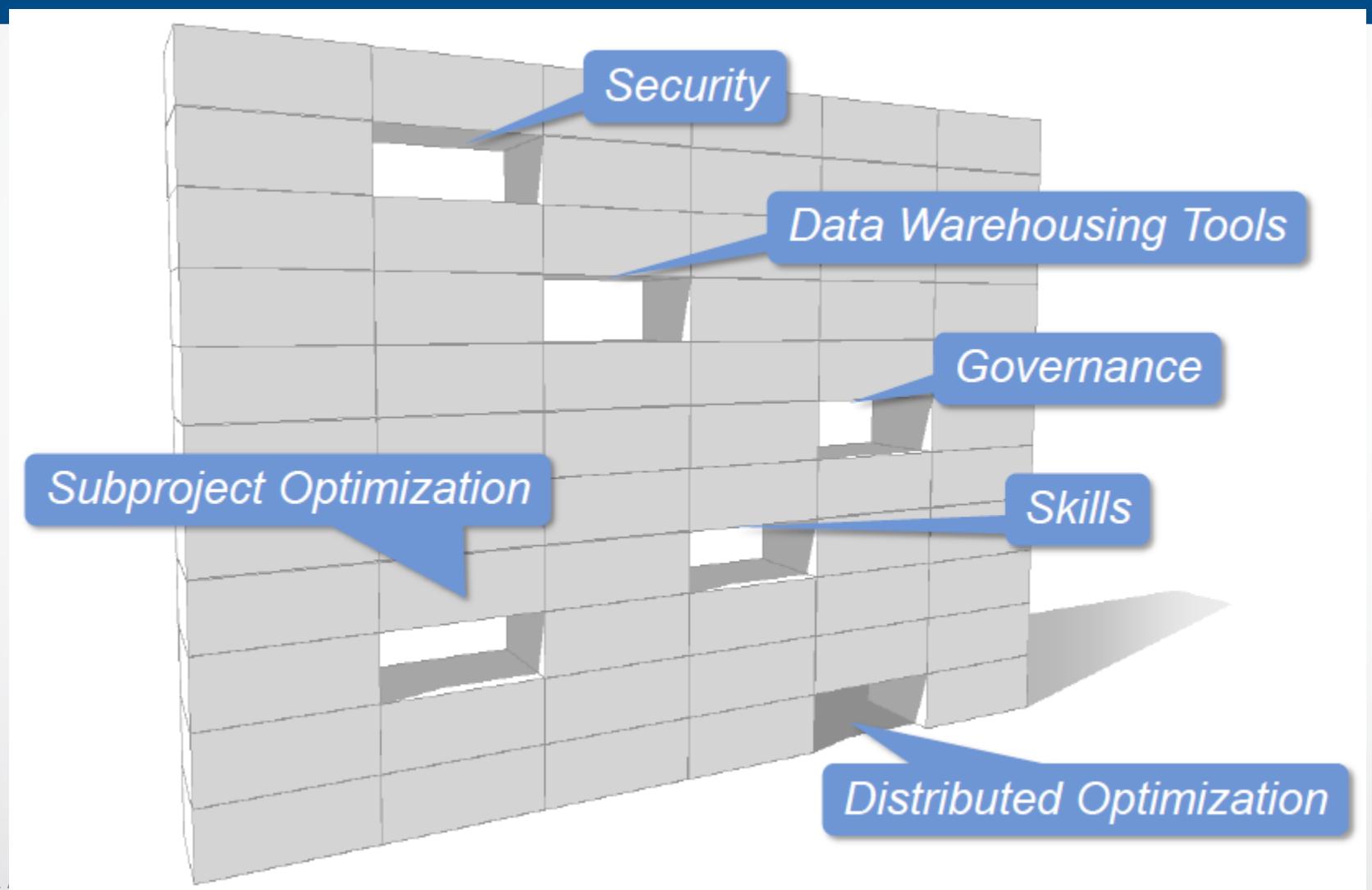


# Data Lake : Les avantages

- Plateforme logicielle flexible
- Ajout de la capacité événementielle
- Agilité & évolution facile
- Supporte tous les utilisateurs
- Supporte toutes les données
- Supporte tout type de données cibles



# Risques liés à un projet Big Data



# Impact sur les droits et libertés du Big Data

- Identifier le besoin ou pas de masquer ou de rendre anonyme les données du fait de leur sensibilité
- Identifier les liens entre les nouvelles données et les données existantes et leur impact selon les directives de la CNIL
- Identifier les mesures de sécurité requises

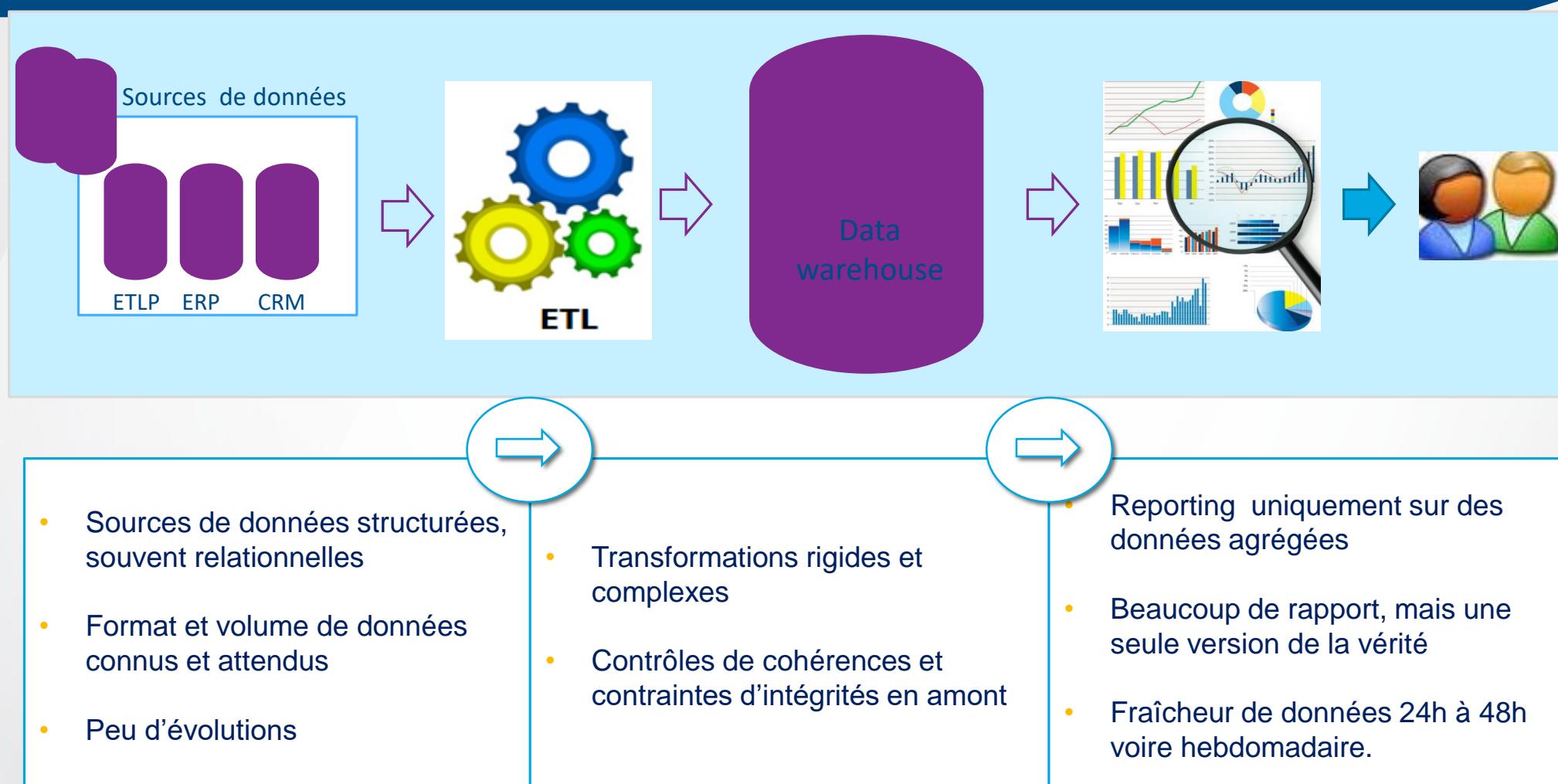


Global Knowledge®

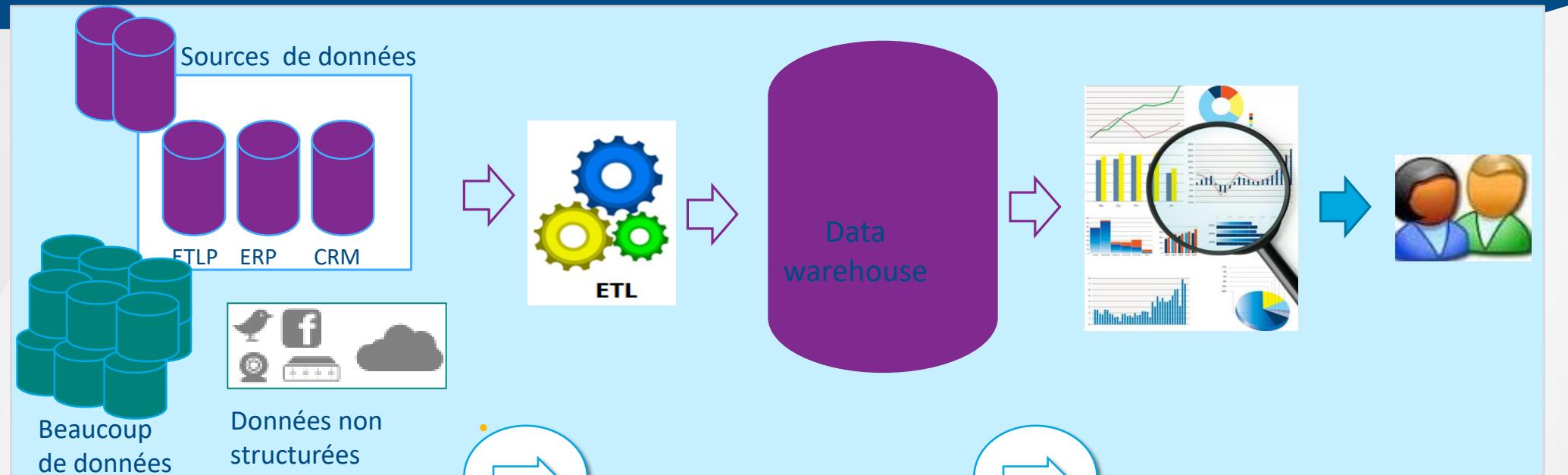


L'architecture Big Data

# Approche traditionnelle (BI)

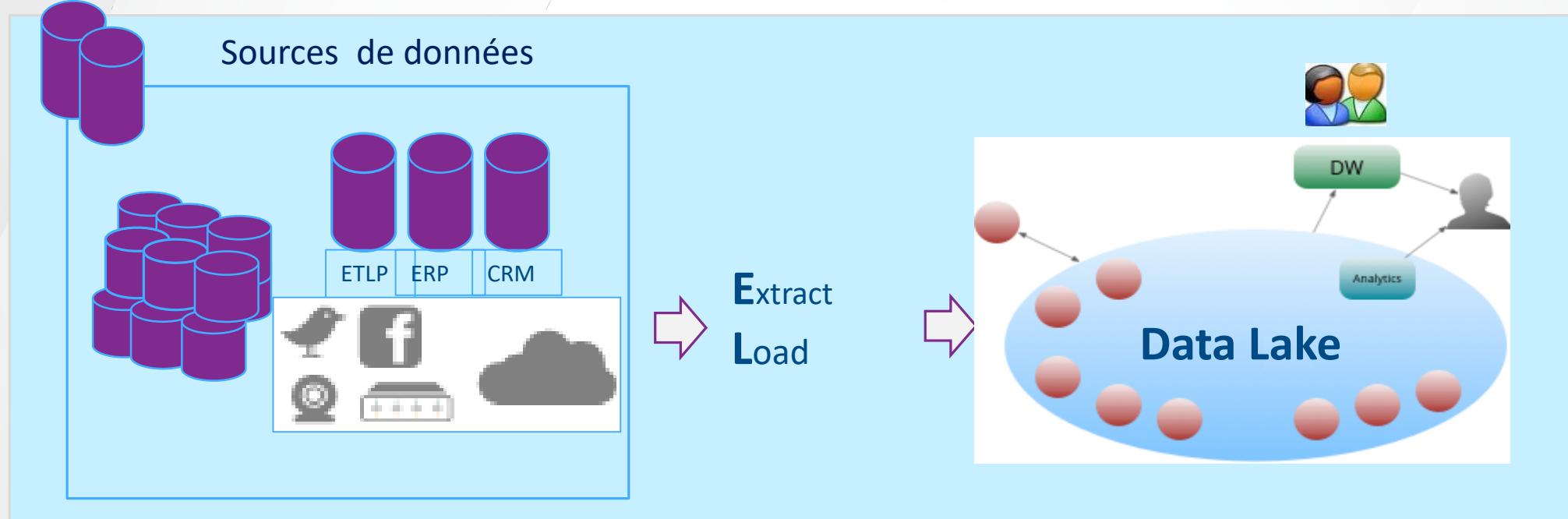


# Approche traditionnelle (BI)



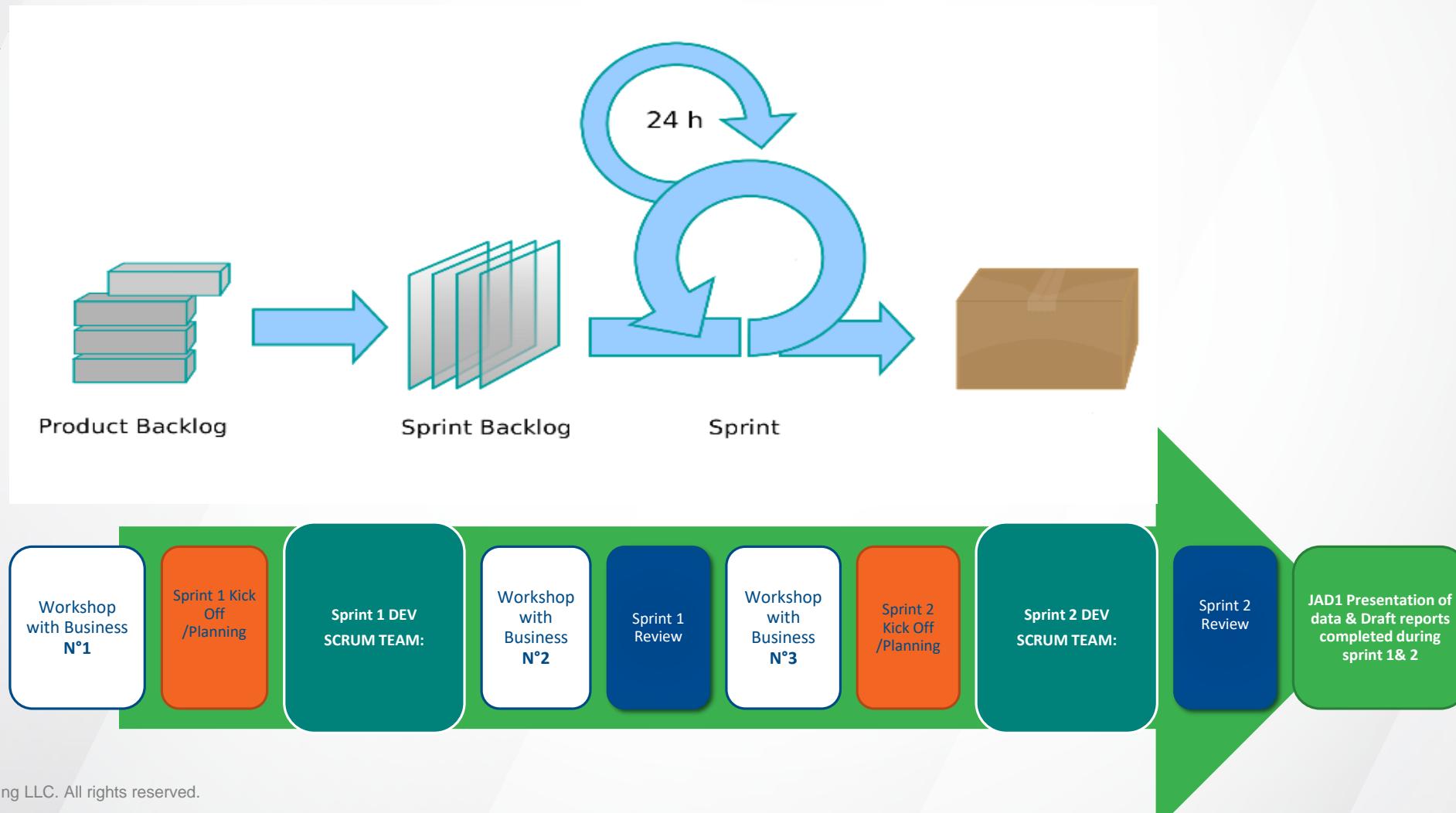
- Augmentation de la variété et du type de sources de données.
- Augmentation du volume de données (Ex Open data .. )
- Pression sur la chaîne d'alimentation
- Technologie inadaptée, retard d'alimentation de données et incapacité de transformer ces volumes
- Nécessite des évolutions dans la chaîne d'alimentation ( ETL )
- Historisation coûteuse (historique vs données ?)
- Reporting invalide ou inutilisable.
- Retard dans les rapports
- Reporting restreint

# Nouvelle approche : Data Lake (ELT et pas ETL)

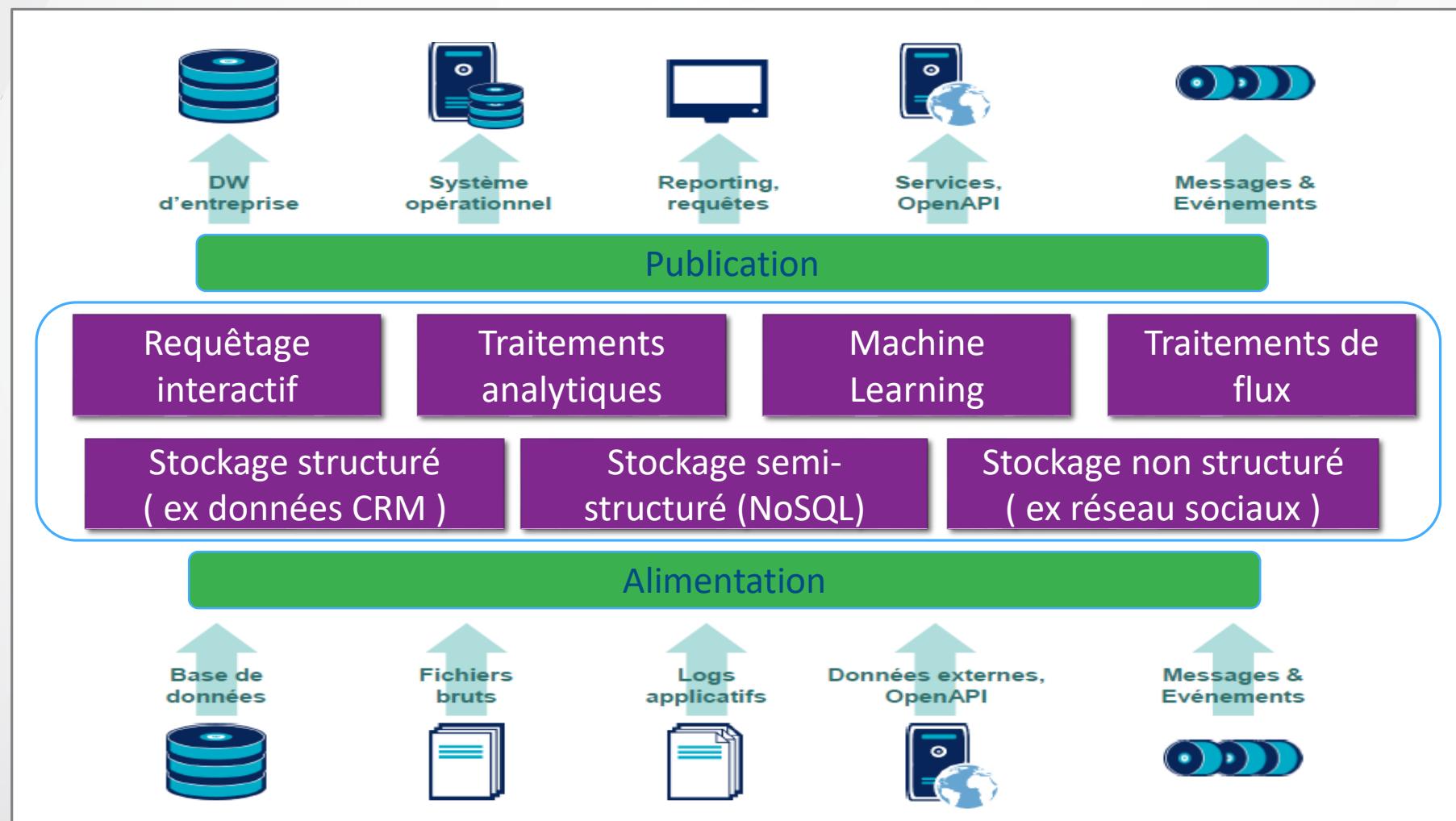


**Rapide, peu couteux et illimité**

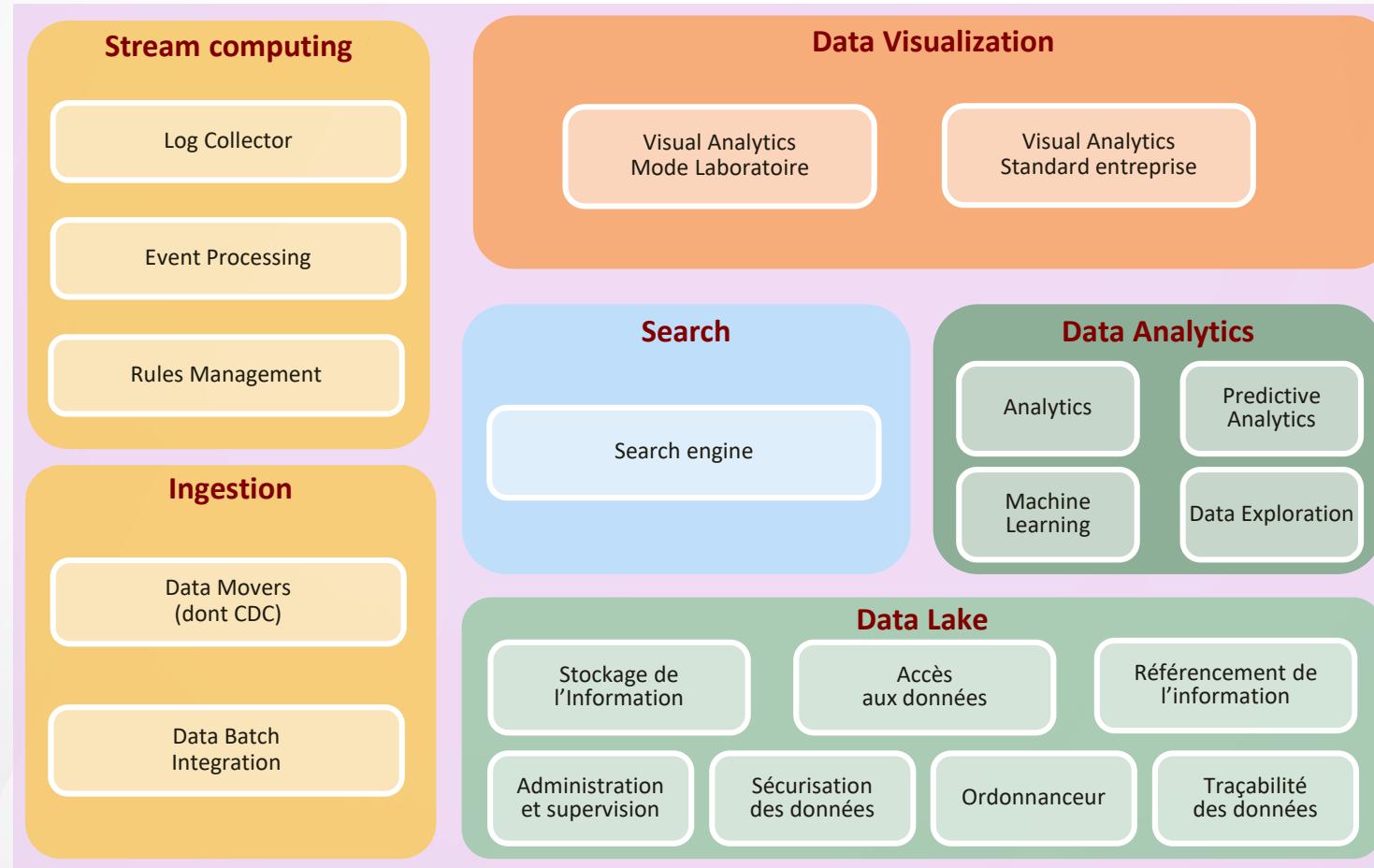
# Méthodologie Agile



# Architecture Data Lake



# Big Data : Les fonctions à couvrir



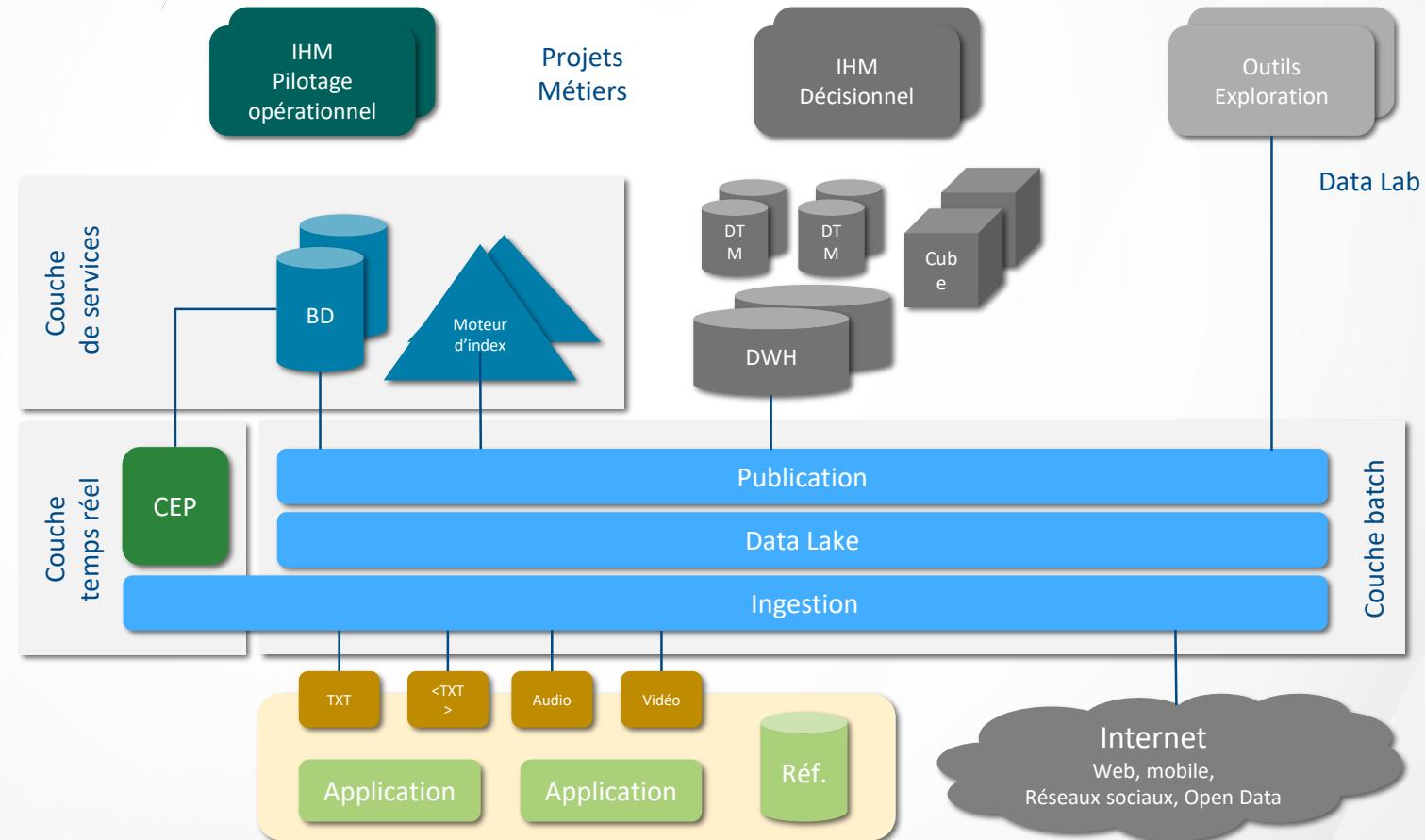
# Big Data : Architecture applicative

Restitution (DataViz)

Stockage spécialisé

Stockage massif

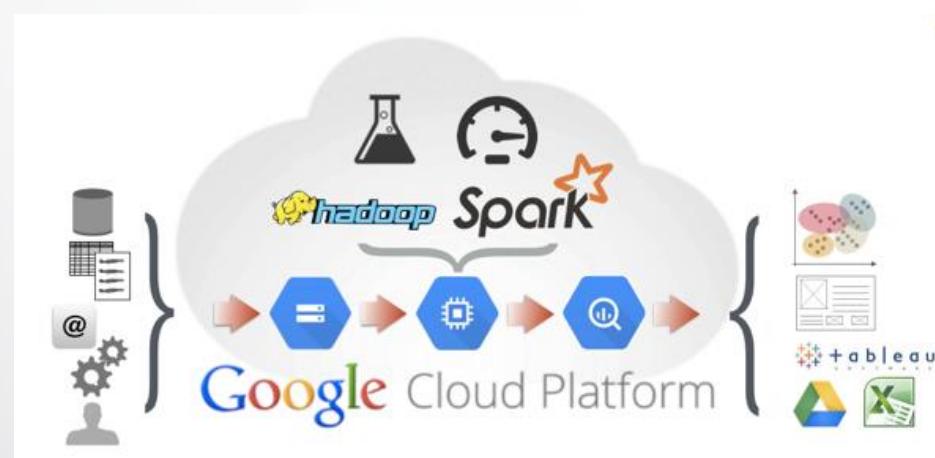
Référentiels et applications opérationnelles



# Big Data dans le Cloud

- Catégorie IaaS
- Catégorie « Hadoop-as-a-Service »
- Catégorie « Analytics-as-a-Service »

# Big Data dans le Cloud



# Choix du Big Data dans le Cloud

1

Certaines DSI ont des contraintes d'hébergement (**place disponible, coût, approvisionnement, compétences**) qui ne leur permettent pas de mettre en place **rapidement** un lot de machines adaptées à Hadoop (commodity hardware, disques JBOD) pour construire un cluster de petite ou moyenne taille.

2

Dans d'autre cas, il s'agit de projets d'innovation pour lesquels **on ne sait pas déterminer** précisément à l'avance **la taille du cluster** dont on aura besoin, ni quel sera son taux d'utilisation.



Global Knowledge®



Hadoop

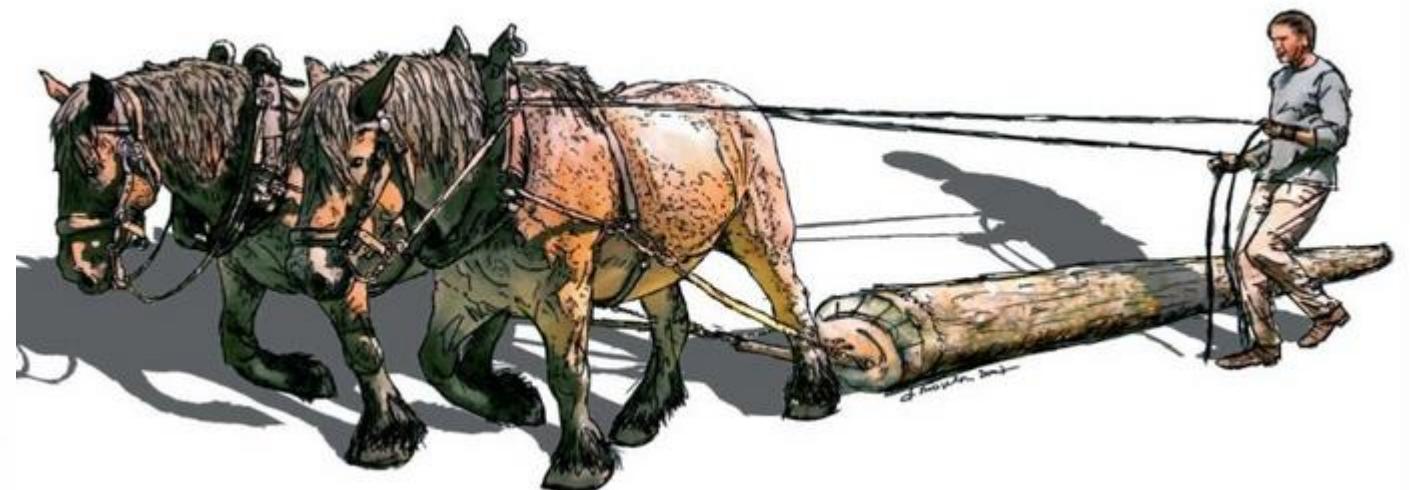
# Big Data – Difficultés à grande échelle

- Besoin : conserver et traiter des données à l'échelle du PB

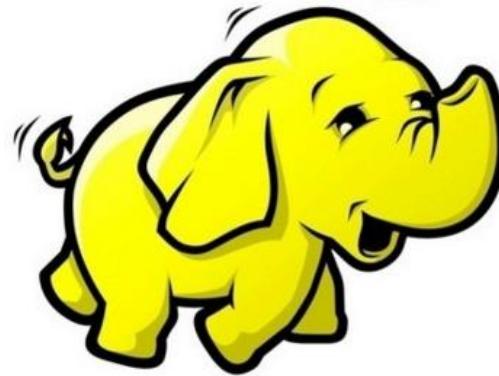


- Besoin : Entreposer des données résistantes aux défaillances
  - Haute disponibilité (availability)
  - Matériel efficace qui gère les défaillances automatiquement
- Besoin : Un framework logiciel résistant aux défaillances
  - Certaines tâches peuvent prendre plusieurs jours

# Hadoop



# Hadoop : Définition



“

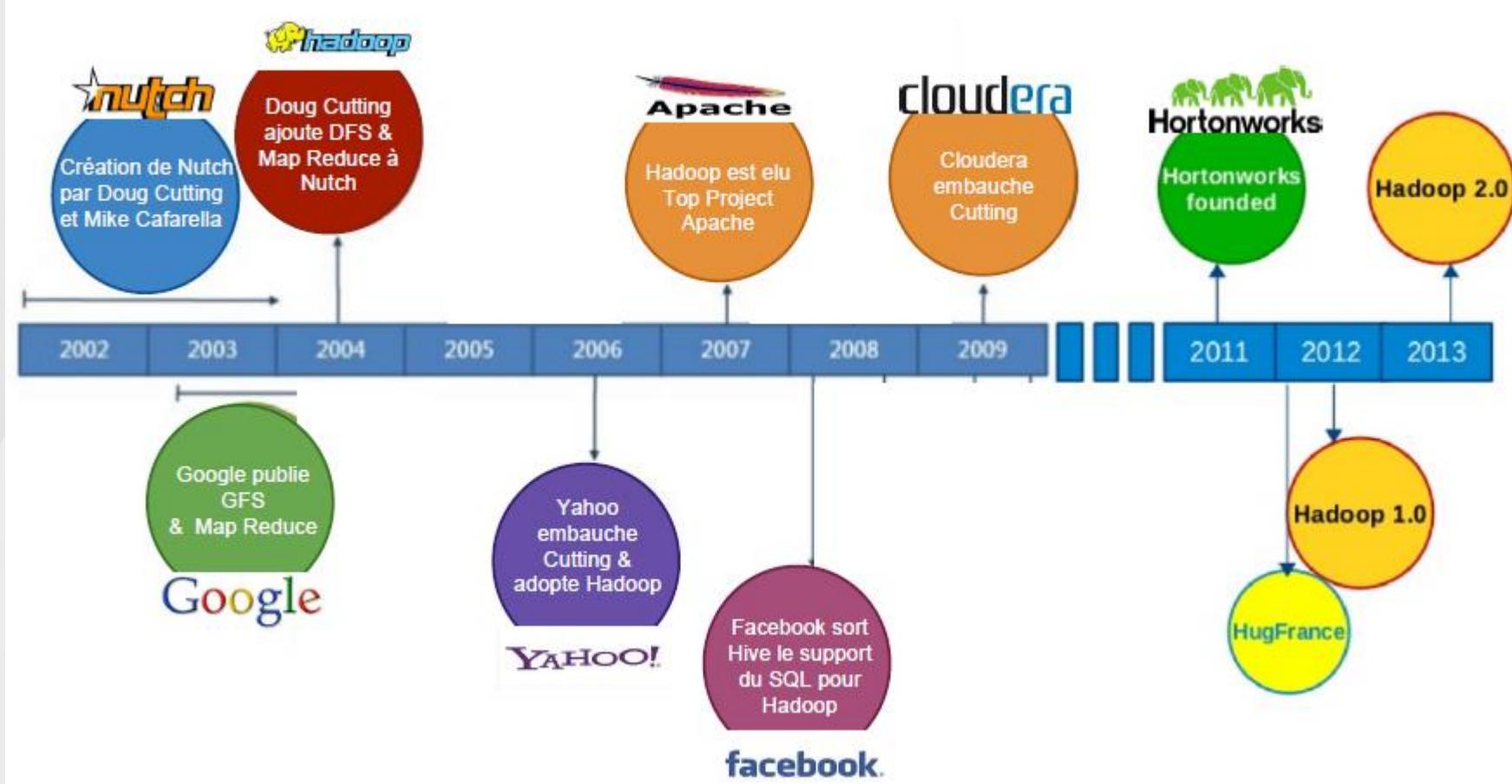
Apache Hadoop est un Framework qui va permettre le **traitement de données massives et distribué** sur un cluster allant d'une à **plusieurs centaines de machines**.

”

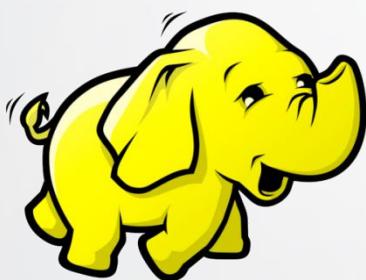
# Hadoop

- Le projet Hadoop consiste en deux grandes parties:
  - Stockage des données : **HDFS** (*Hadoop Distributed File System*)
  - Traitement des données : **MapReduce**
- Principes :
  - Diviser les données
  - Les sauvegarder sur une collection de machines, appelées *cluster*
  - Traiter les données directement là où elles sont stockées, plutôt que de les copier à partir d'un serveur distribué
- Il est possible d'ajouter des machines à votre cluster, au fur et à mesure que les données augmentent

# Hadoop : Historique



# Hadoop : un nouveau paradigme



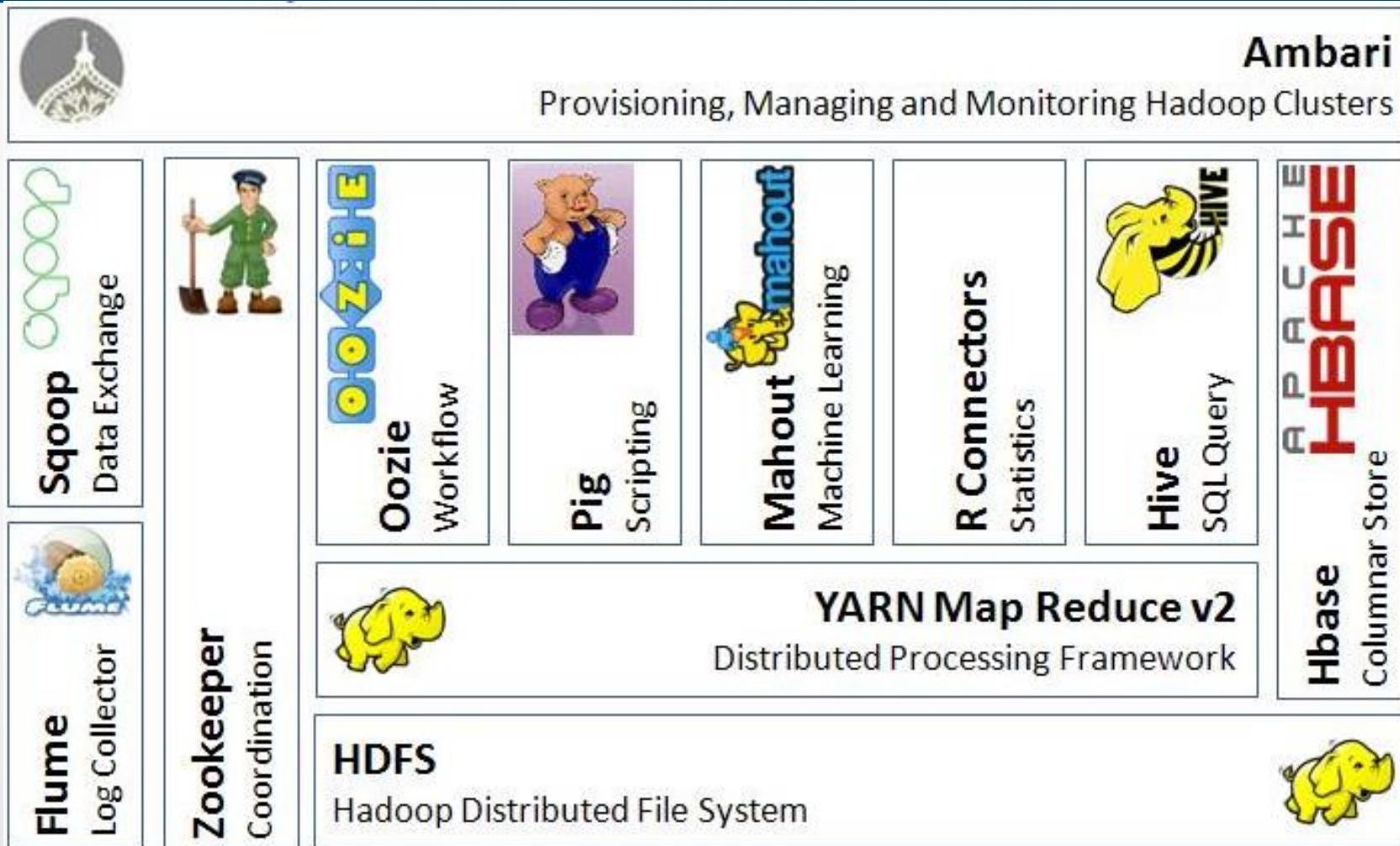
- Traitement à grande échelle et haute performance
  - Peut évoluer de 10 nœuds à 10,000 nœuds
  - Plus facile, gratuit, ouvert
- Efficace
  - Puissance de computation CPU, mémoire
  - Stockage sur disques locaux
- Nouveau
  - De nouvelles fondations
- Actuel
  - Presque tous les leaders du web 2.0
  - La grande entreprise Fortune 500

# Hadoop est robuste !

- Une solution logicielle
- Ratio coût-puissance intéressant
- Évolutif
  - On peut toujours ajouter des nœuds pour plus de capacités
    - Computation
    - Stockage
- Général
  - S'applique à une variété de problèmes utiles
  - Programmation parallèle simplifiée
- Pas de barrières pour commencer
  - Pas de schéma ou de design requis.
  - Charger des fichiers « raw » et lancer une applications

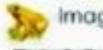
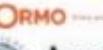
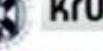
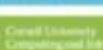
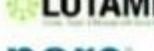
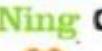
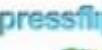
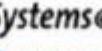
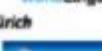
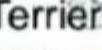
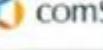
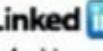
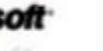
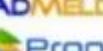
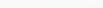
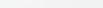


# Ecosystème Apache Hadoop



# Hadoop : Références

➤ En 10 ans, son adoption ne fait plus aucun doute !

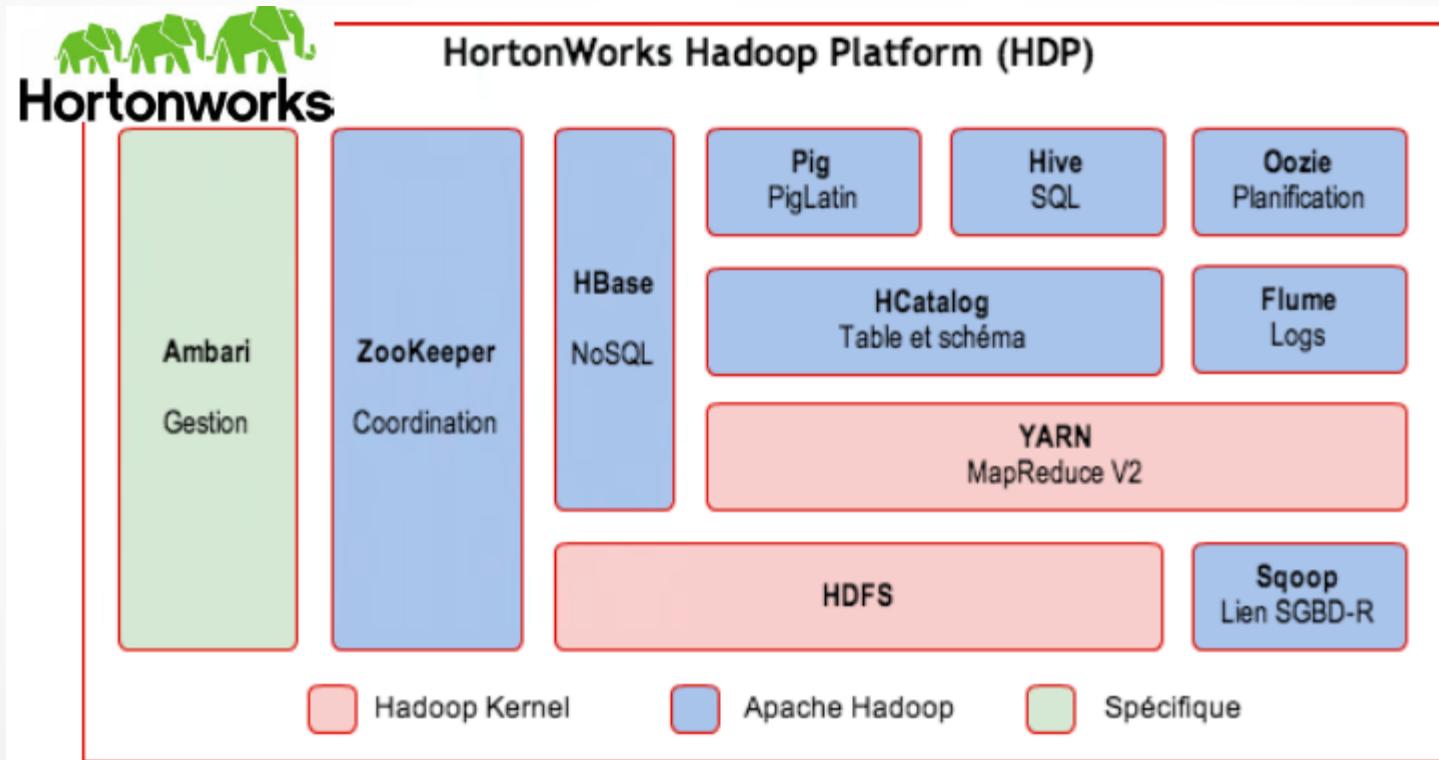
2007	2008	2009	2010
 	                  	                         	                             

# Hadoop : Les distributions

- Les principaux

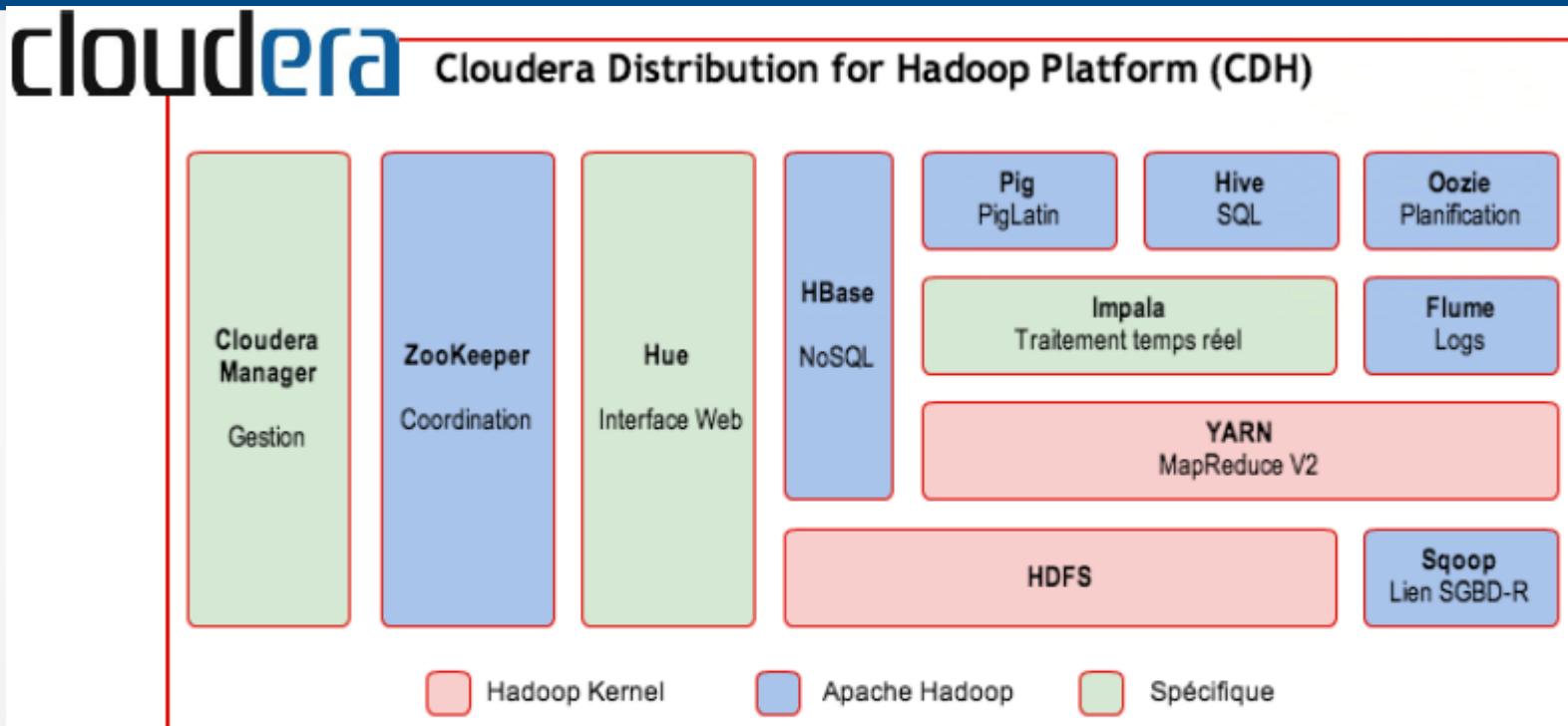


# Hadoop : Hortonworks



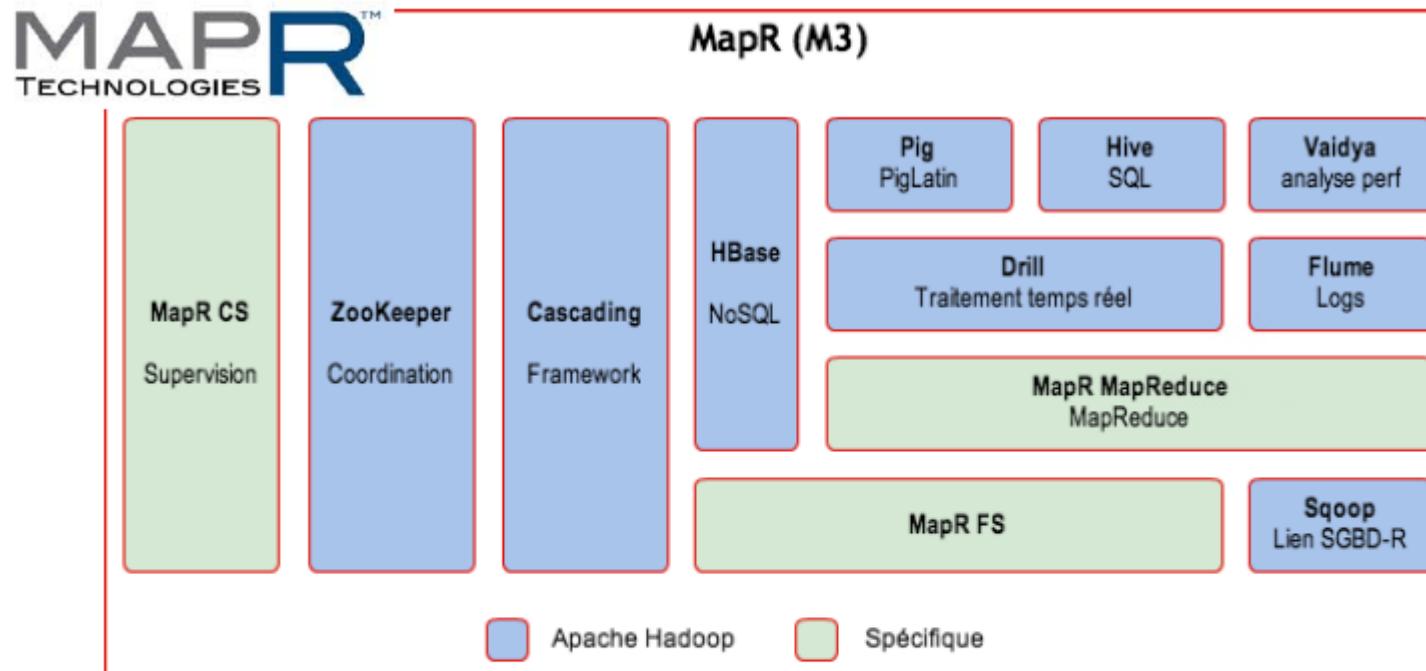
- 2011 Yahoo.
- Composants **Open Source**.
- **1:1 avec Hadoop**.
- Projets reversés à Hadoop :
  - YARN
  - HCatalog
  - Ambari

# Hadoop : Cloudera



- Commercial et Open Source.
- Hadoop + Composants Maison.
  - **Composants propriétaires :**
    - Impala (requêtes temps réel)
    - Cloudera Manager (gestion du cluster, déploiement..)
  - **Projets reversés à Hadoop :**
    - Hadoop Common (Utilitaires)
    - Hue (SDK IHM)
    - Whirr (SAAS Hadoop)

# Hadoop : MAPR



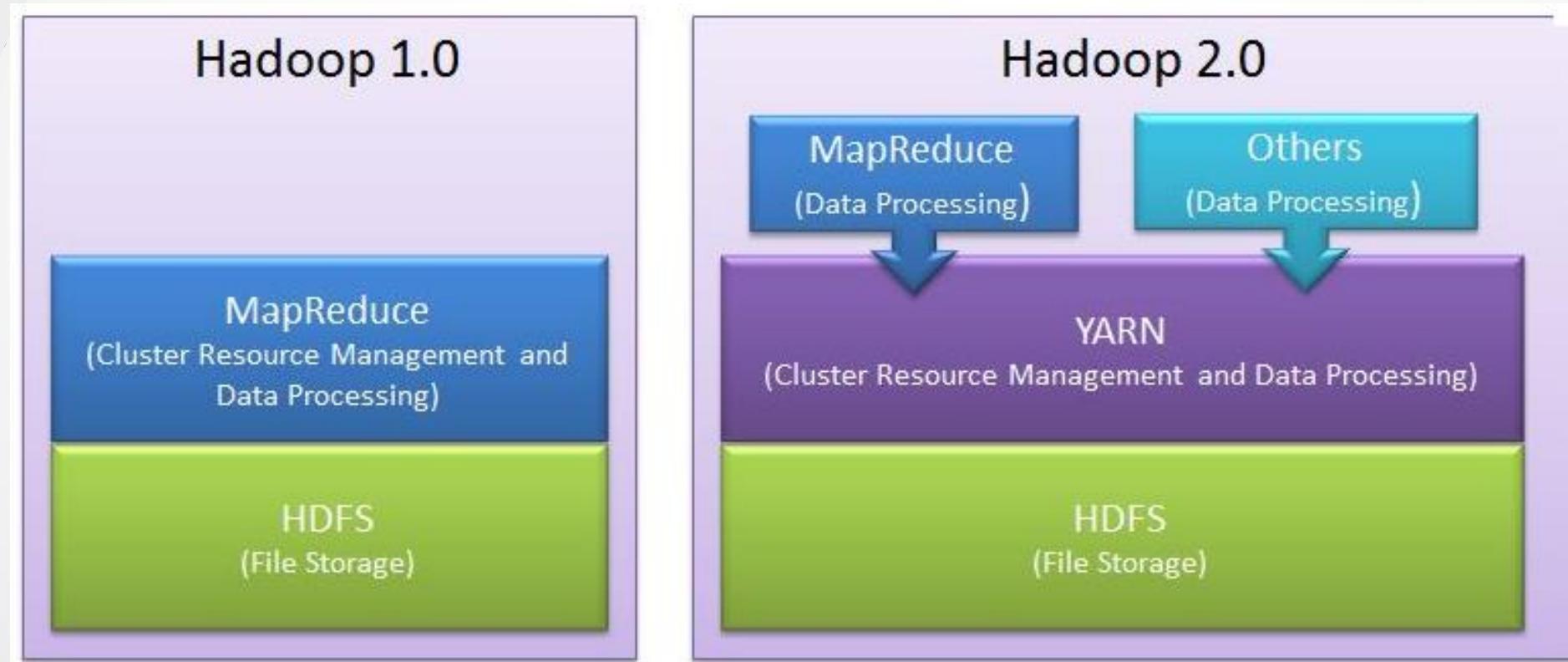
- 2009, Commercial et Open Source
- Refonte du cœur de la plateforme (MapR FS et MapR MapReduce)
- Optimisé pour Hbase.
- **Composants propriétaires :**
  - Cascading (Usine de dev Java)
  - Vaidya (Perf, Benchmark)
  - Drill (Requêtes temps réel)
- **Projets reversés à Hadoop :**
  - Hbase, Pig, Hive, Mahout
  - Sqoop, Flume.

# Hadoop : Les distributions

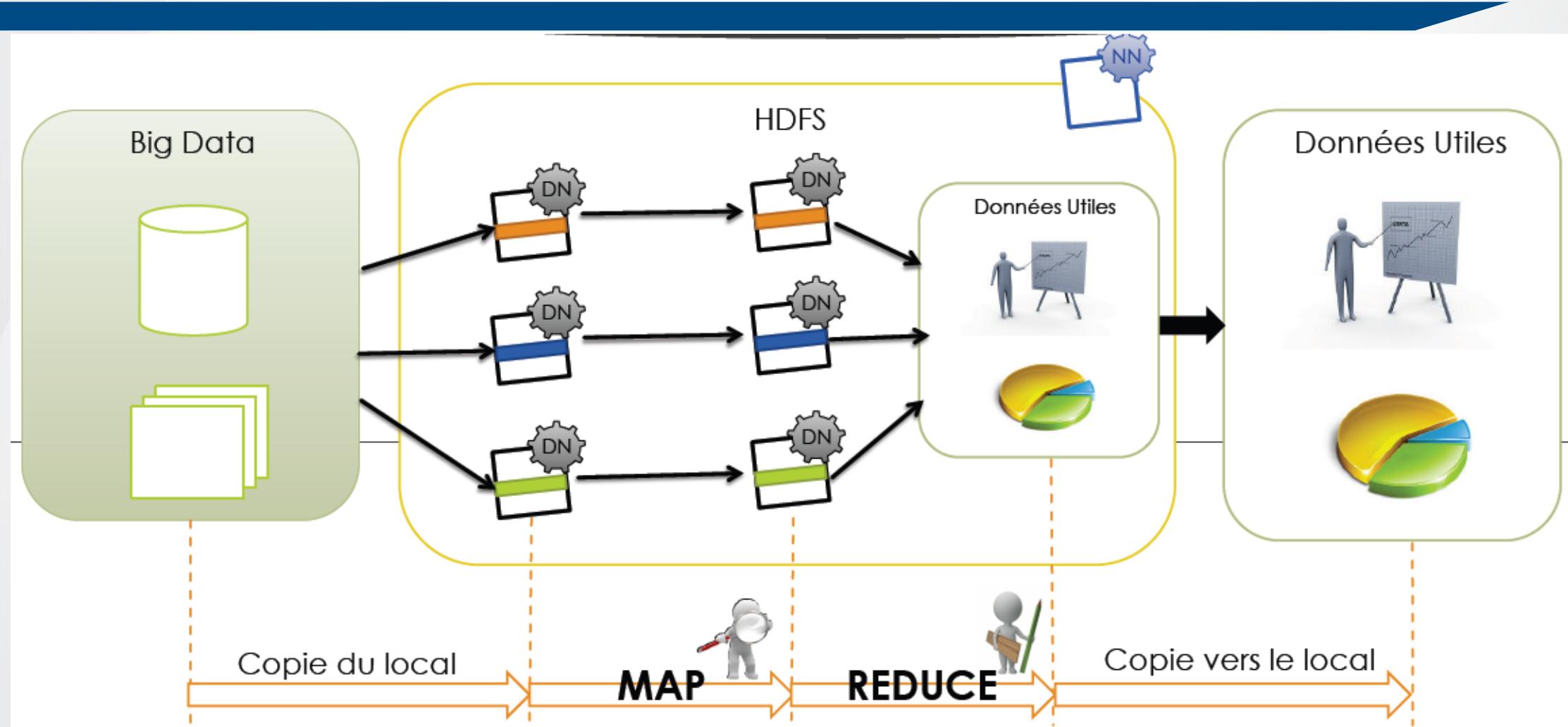
## ➤ Comment choisir une solution Hadoop ?

- Modèle économique (Open Source, Commercial..).
- Les composants.
- Maturité de la solution, le support, la documentation, le retour d'expériences.
- Le rapport avec Hadoop, la rapidité des évolutions.
- Partenariats (hébergeurs...), compatibilité.

# Hadoop 1.0 => Hadoop 2.0

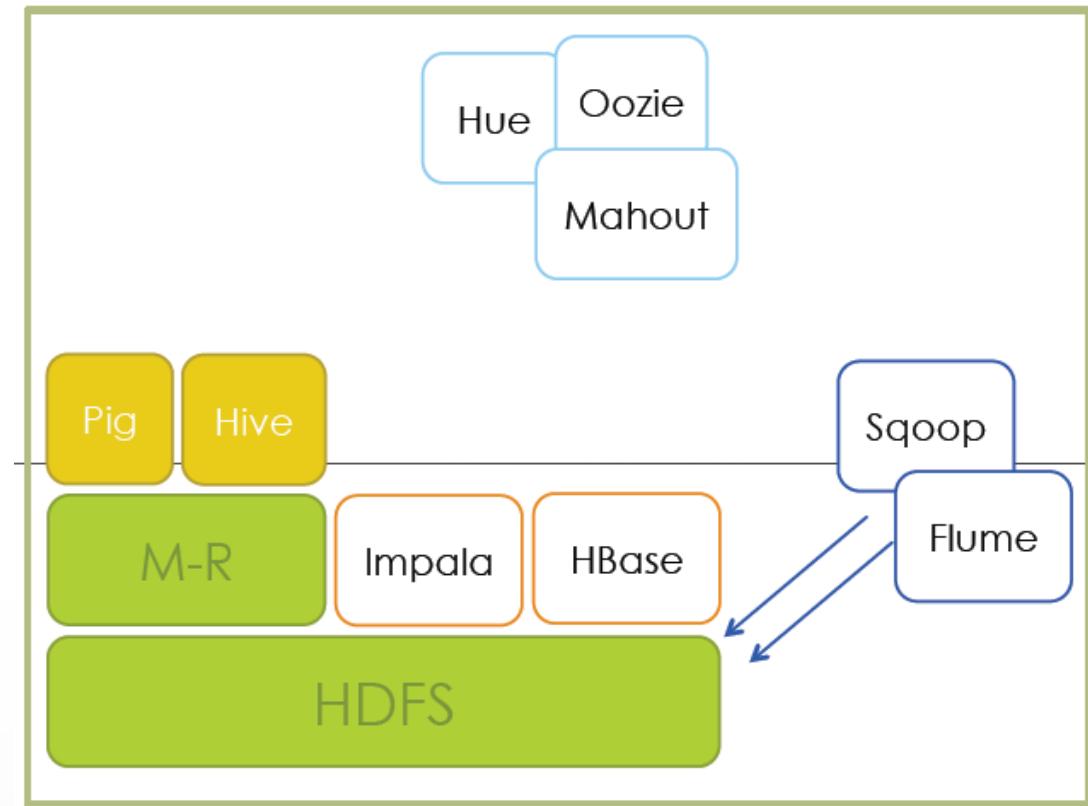


# Hadoop, HDFS et MapReduce



# Hadoop, HDFS et MapReduce

- MapReduce utilise des langages de programmation pour traiter les données :
  - **Java, Ruby, Python...**
- Plusieurs outils facilitant le travail
- Au dessus du MapReduce : langage plus simple traduit plus tard en Mappers et Reducers :
  - **PIG** : script simple
  - **Hive** : requêtes SQL

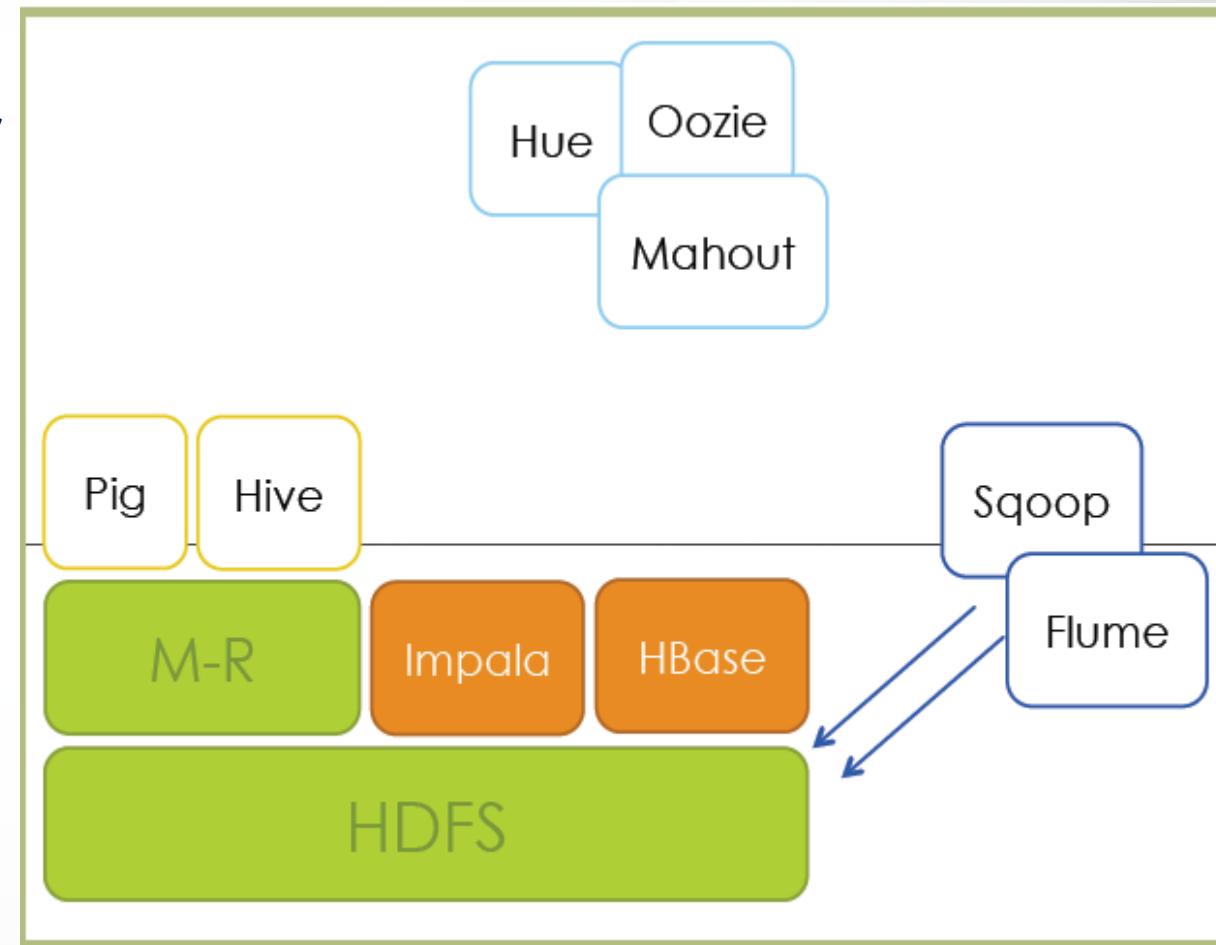


# Ecosystème de Hadoop

Les jobs MapReduce peuvent prendre beaucoup de temps pour s'exécuter sur de larges quantités de données

## Autres projets pour simplifier :

- Impala :
  - Extraction des données directement à partir de HDFS avec SQL
  - Requêtes plus rapides que Hive
- HBase :
  - Base de données temps réel
- Spark



# Ecosystème de Hadoop

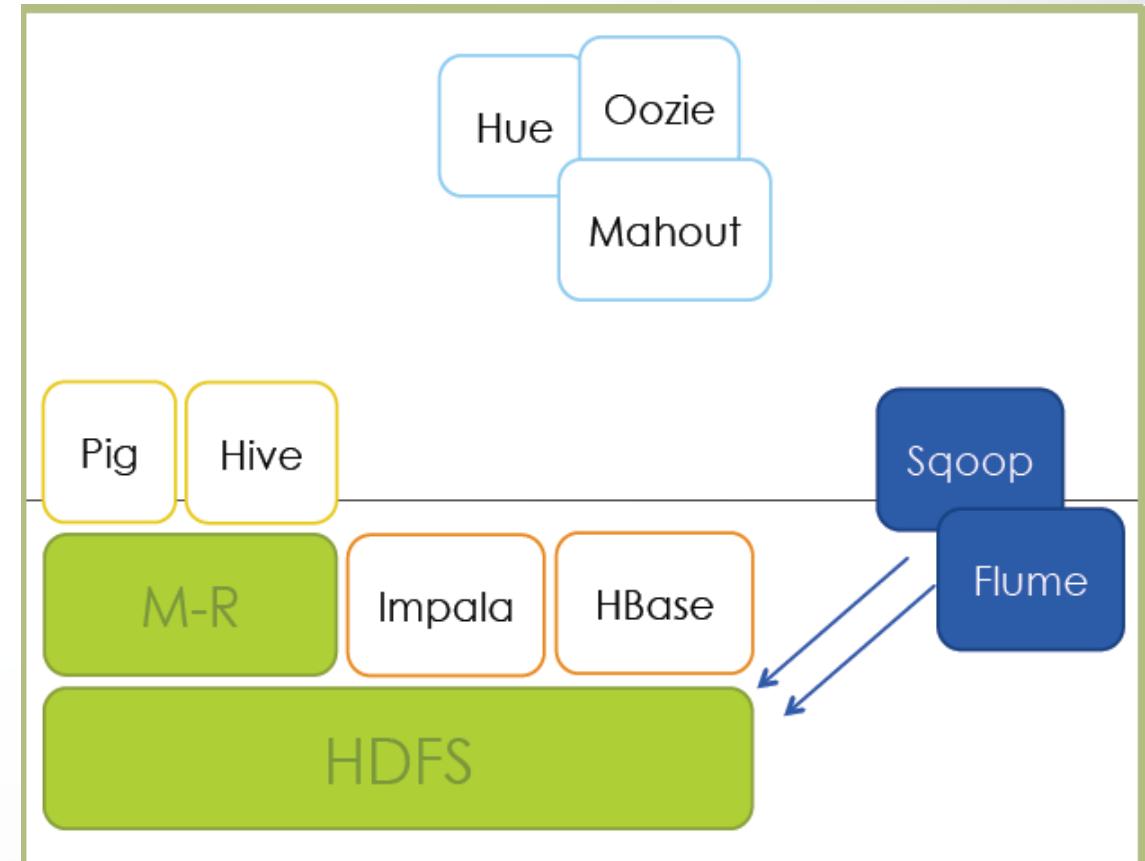
Connexion du HDFS à partir d'outils externes.

➤ **Sqoop :**

- Prend les données à partir d'une base de données traditionnelle, et les met dans HDFS, comme étant des fichiers délimités, pour être traitées avec d'autres données dans le cluster

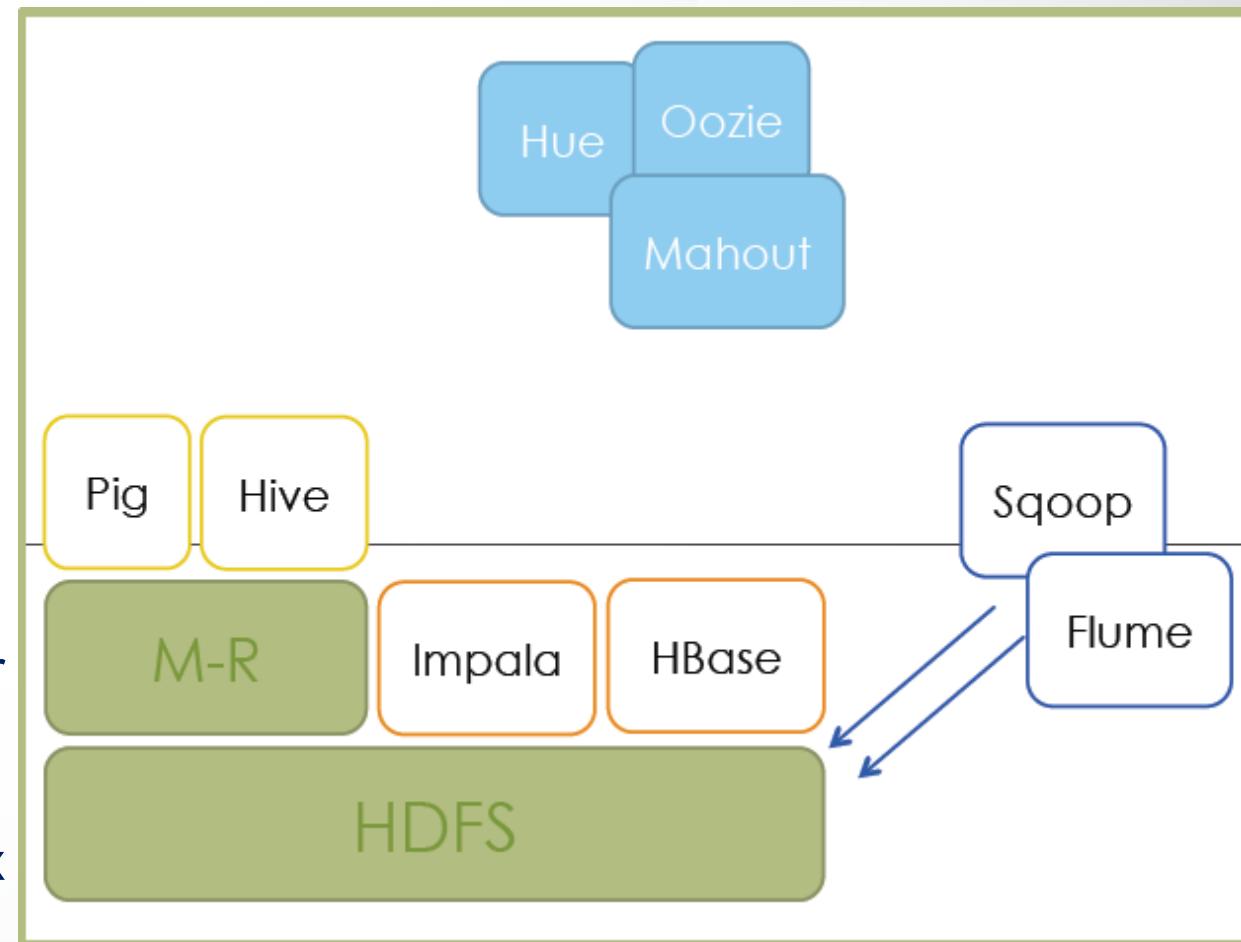
➤ **Flume :**

- Système distribué permettant de collecter, regrouper et déplacer efficacement un ensemble de données (des logs) à partir de plusieurs sources vers le HDFS



# Ecosystème de Hadoop

- **Hue :**  
Front-end graphique pour le cluster, fournit :
  - Un navigateur pour HDFS et HBase
  - Des éditeurs pour Hive, Pig, Impala et Sqoop
- **Oozie :**
  - Outil de gestion de Workflow
  - Permet de gérer les jobs Hadoop
- **Mahout :**
  - Bibliothèque d'apprentissage automatique
- **Permet de :**
  - Déterminer des éléments qu'un utilisateur pourra apprécier selon son comportement
  - Grouper des documents
  - Affecter automatiquement des catégories aux documents

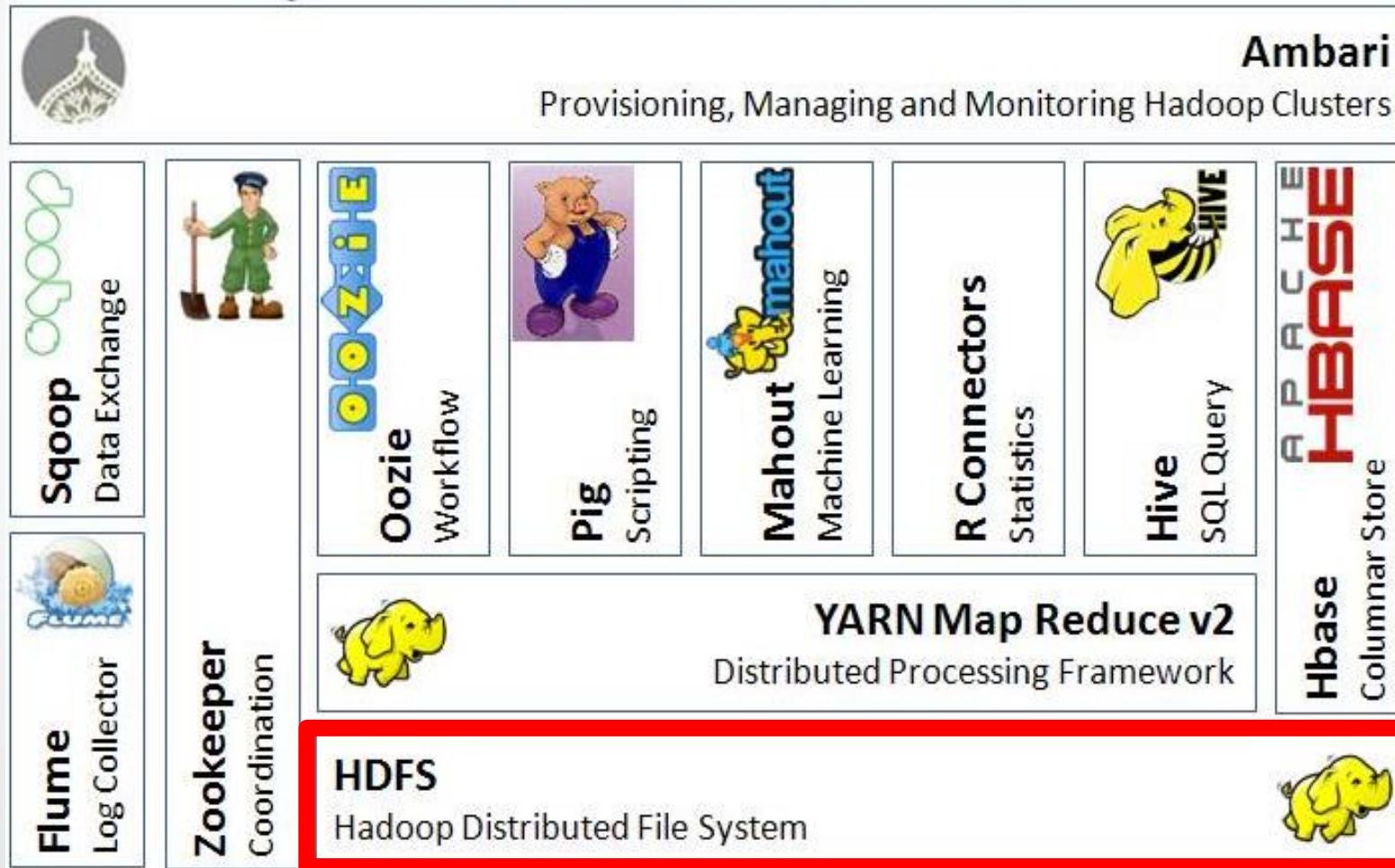




Global Knowledge®

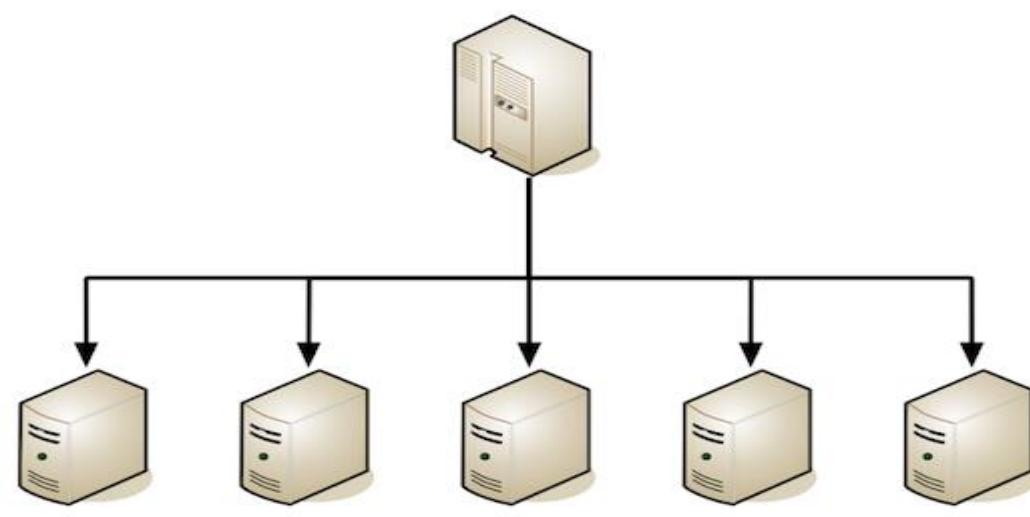
HDFS

# HDFS : Hadoop Distributed File System



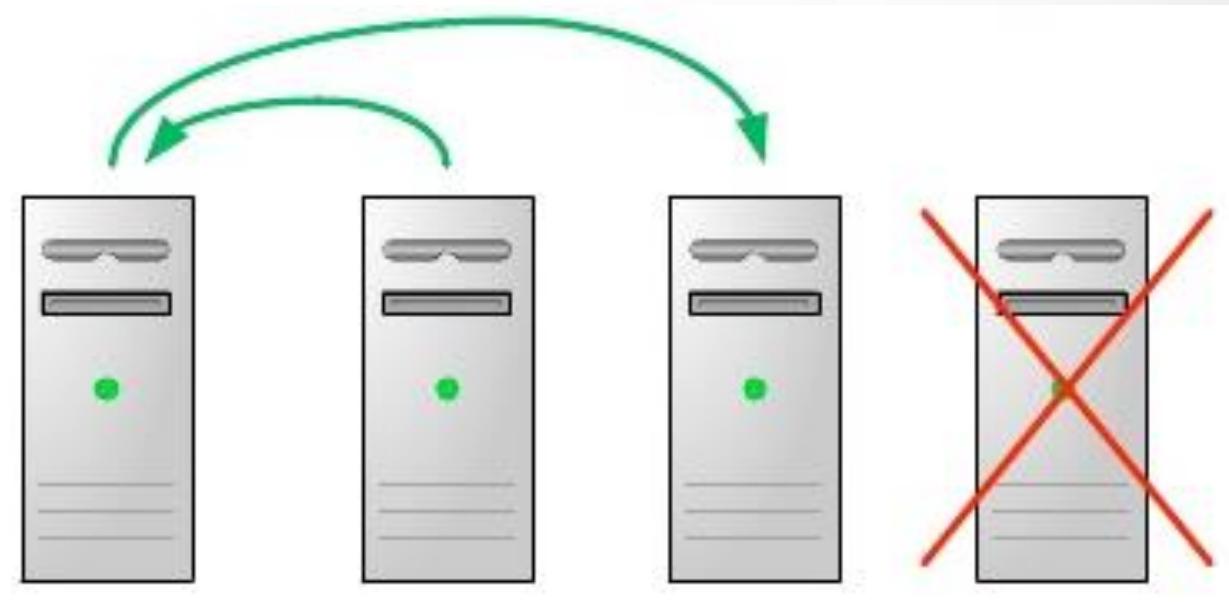
# HDFS : Hadoop Distributed File System

- Economique
- Tolérant aux pannes
- Evolutif



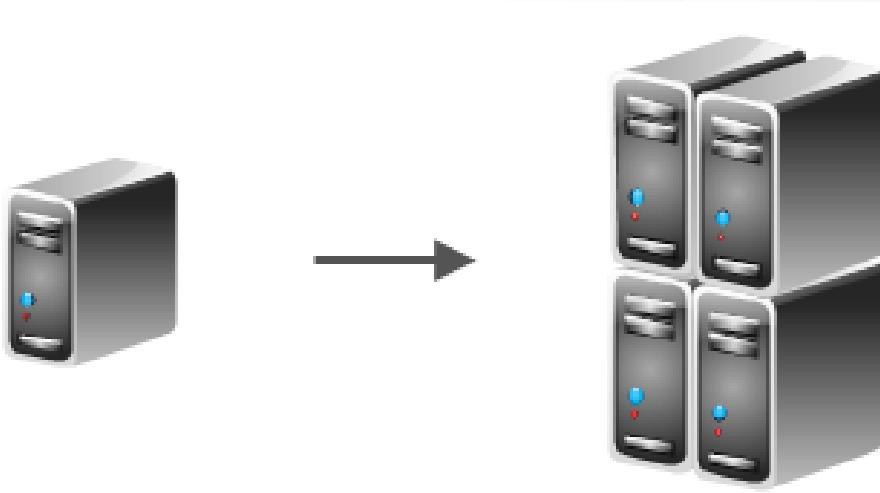
# HDFS : Avantages

- Tolérant aux pannes

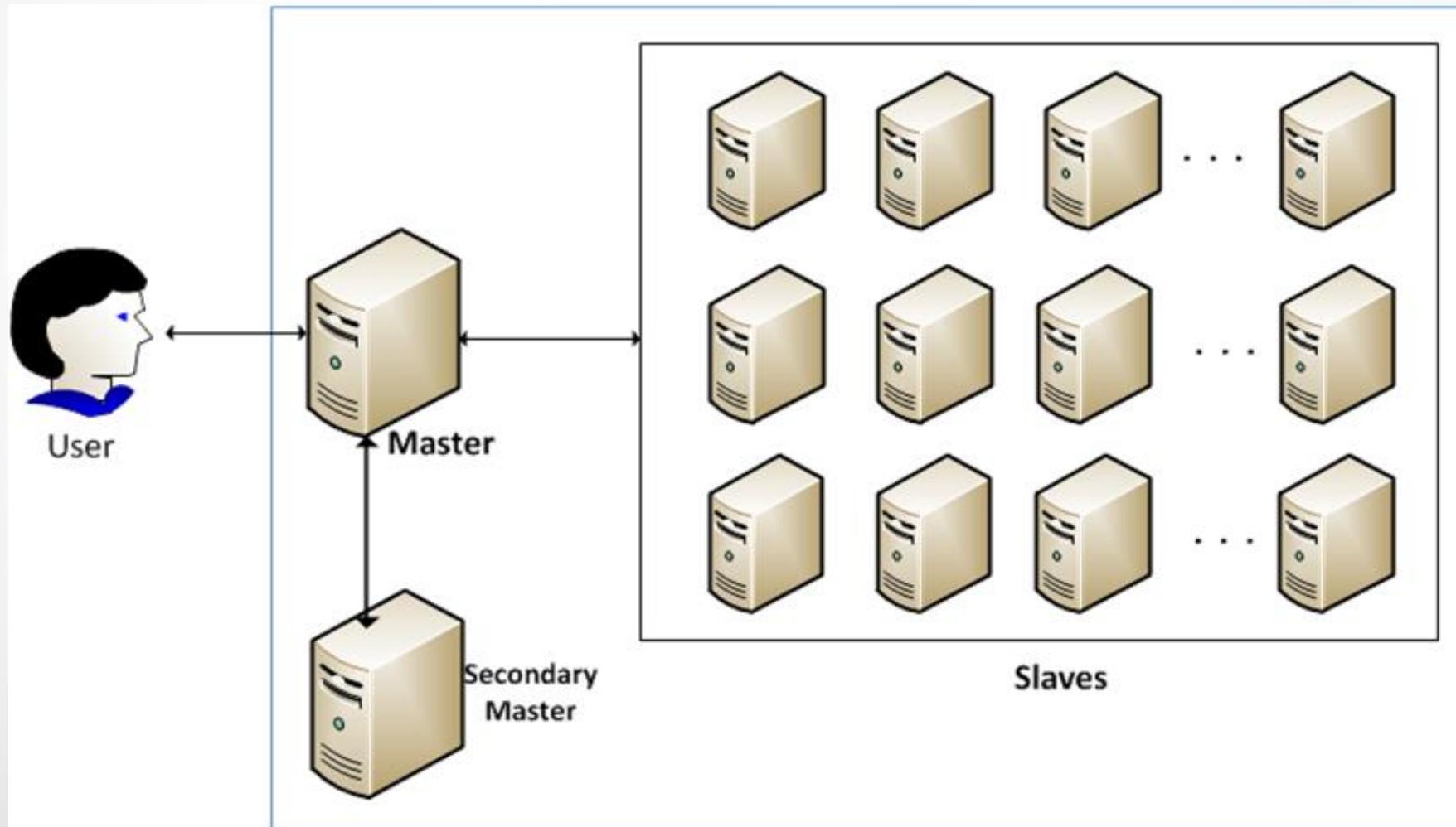


# HDFS : Avantages

- Evolution horizontale

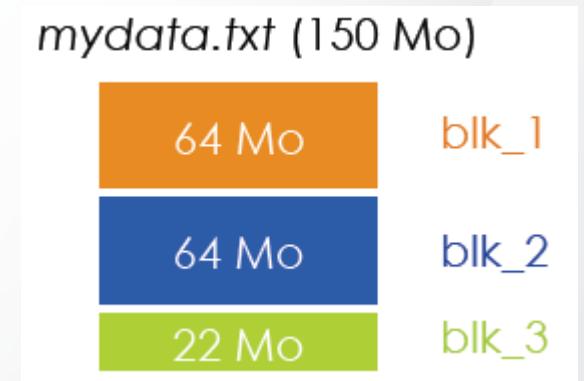
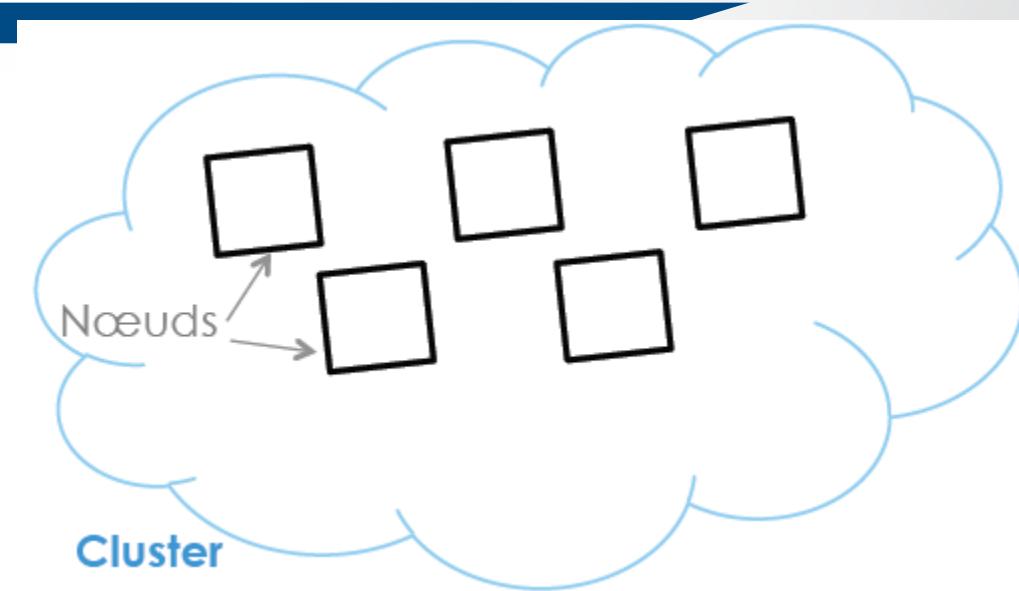


# Cluster Hadoop



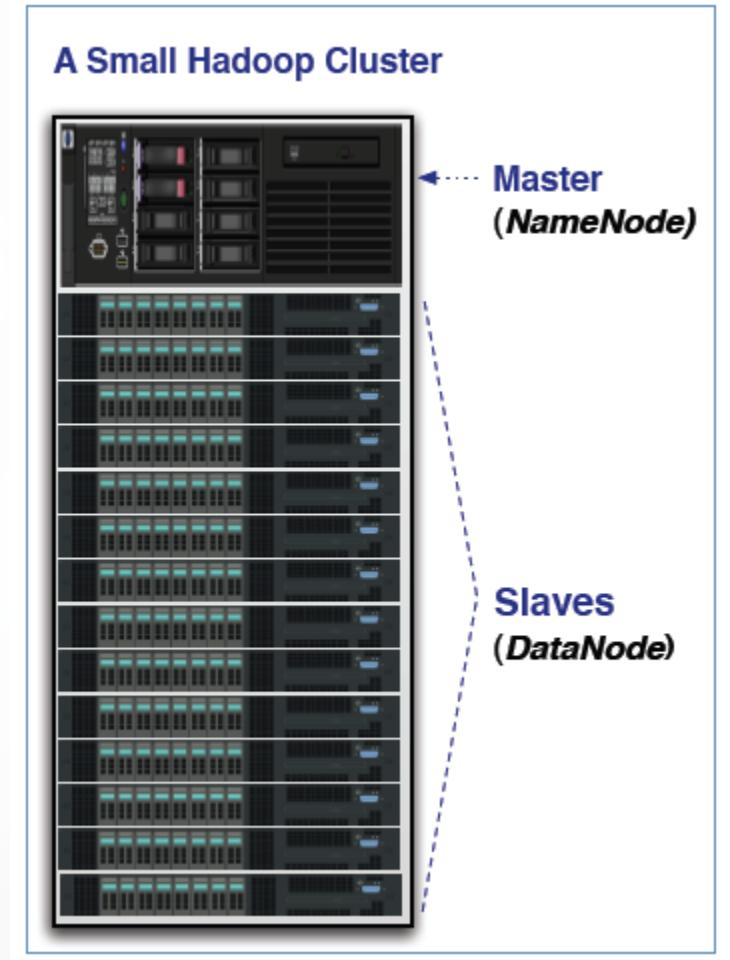
# HDFS : Hadoop Distributed File System

- HDFS est un système de fichiers distribués, extensible et portable
- Ecrit en Java
- Permet de stocker de très gros volumes de Données sur un grand nombre de machines (nœuds) équipées de disques durs banalisés (Cluster)
- Quand un fichier mydata.txt est enregistré dans HDFS, il est décomposé en grands blocs (par défaut 64Mo), chaque bloc ayant un nom unique: blk\_1, blk\_2...



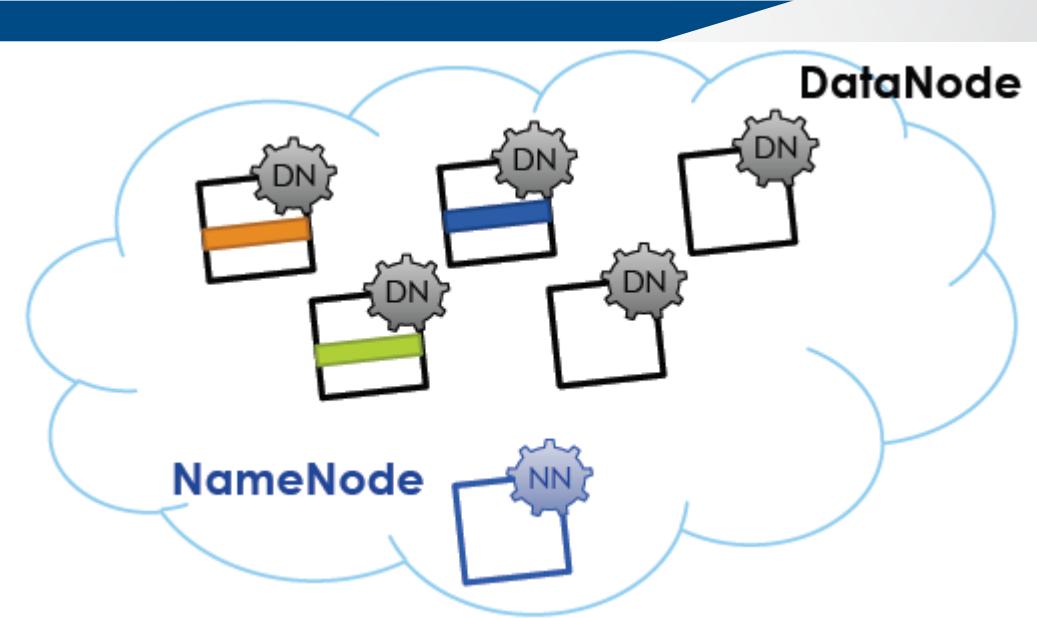
# HDFS : Hadoop Distributed File System

- Chaque bloc est enregistré dans un nœud différent du cluster
- DataNode : démon sur chaque noeud du cluster
- NameNode :
  - Démon s'exécutant sur une machine séparée
  - Contient des *méta-données*
  - Permet de retrouver les noeuds qui exécutent les blocs d'un fichier

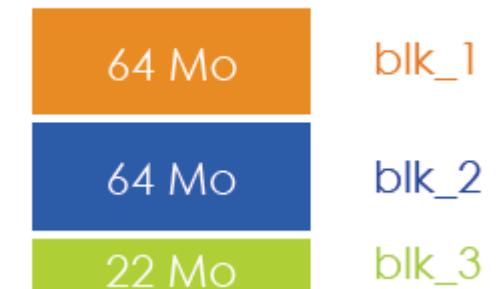


# HDFS : Hadoop Distributed File System

- Quels sont les problèmes possibles?
  - Panne de réseau ?
  - Panne de disque sur les DataNodes ?
  - Panne de disque sur les NameNodes ?

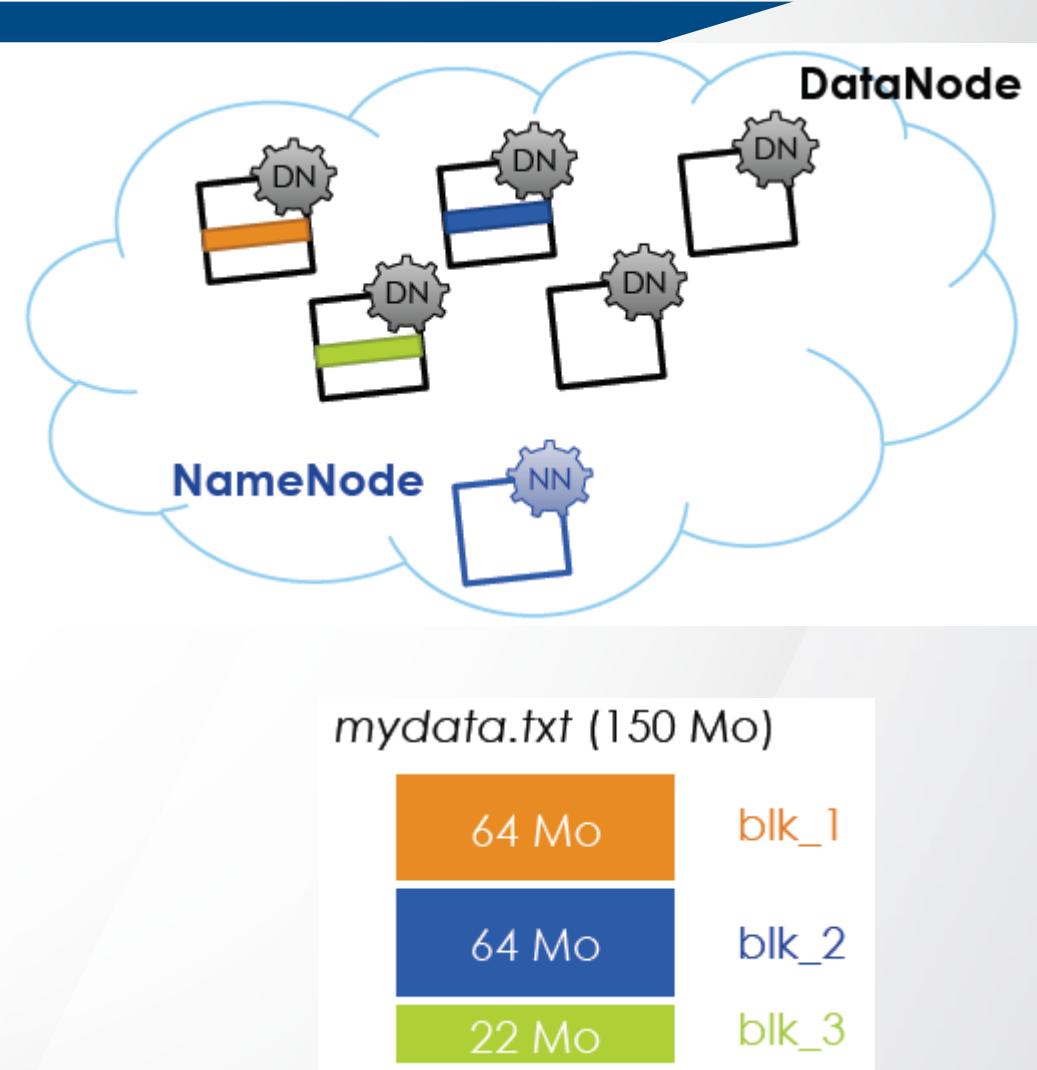


mydata.txt (150 Mo)



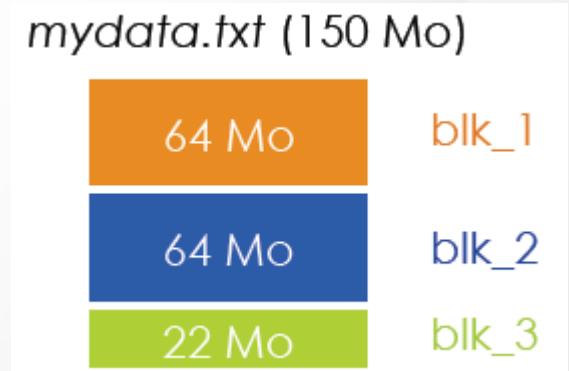
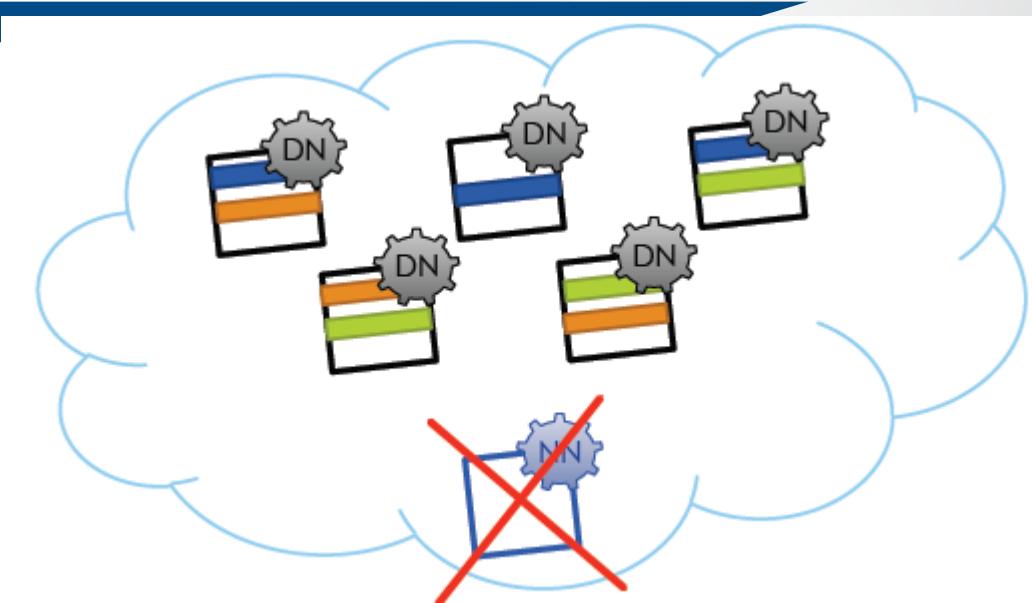
# HDFS : Hadoop Distributed File System

- Si l'un des nœuds a un problème, les données seront perdues
  - Hadoop réplique chaque bloc 3 fois
  - Il choisit 3 nœuds au hasard, et place une copie du bloc dans chacun d'eux.
  - Si le nœud est en panne, le NN le détecte, et s'occupe de répliquer encore les blocs qui y étaient hébergés pour avoir toujours 3 copies stockées
  - Concept de Rack Awareness (rack = baie de stockage)



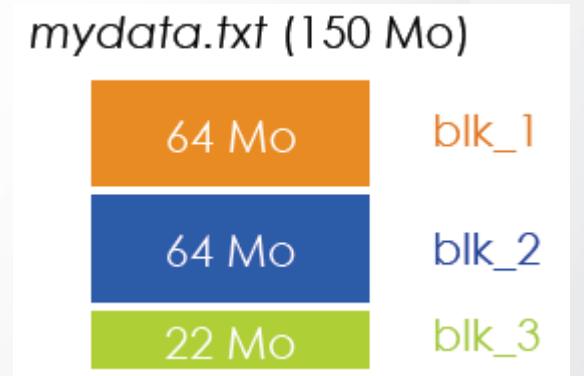
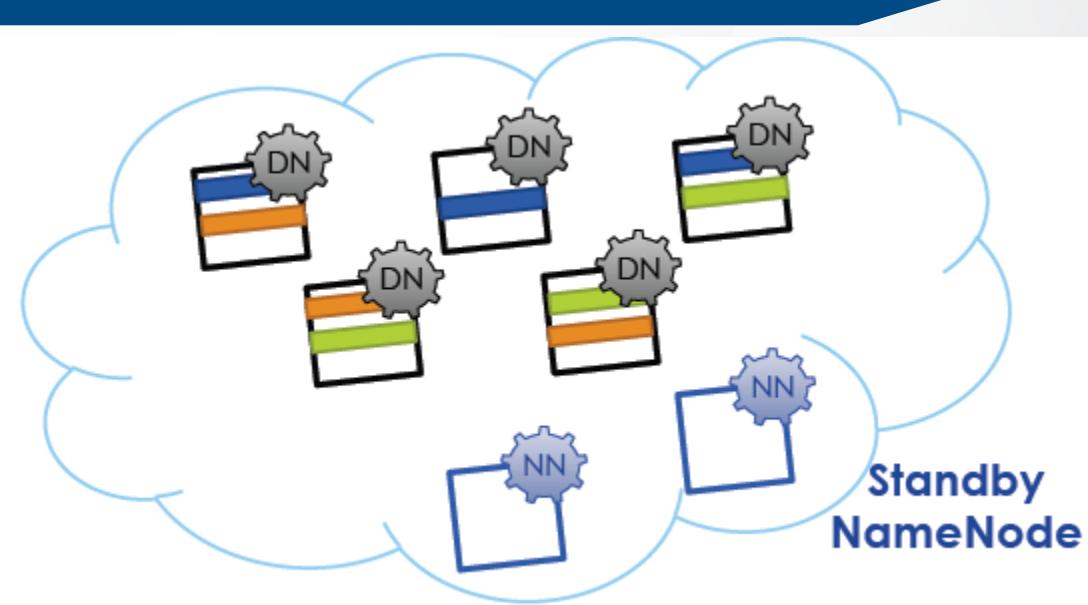
# HDFS : Hadoop Distributed File System

- Si le NameNode a un problème ?
  - Données perdues à jamais
  - Pas de problème
- Si c'est un problème d'accès (réseau), les données sont temporairement inaccessibles



# HDFS : Hadoop Distributed File System

- Pour éviter cela :
  - Le NameNode sera dupliqué, non seulement sur son propre disque, mais également quelque part sur le système de fichiers du réseau.
- Définition d'un autre NN (Secondary namenode)
  - Reprendre le travail si le NameNode actif est défaillant



# HDFS : Les commandes

- Il y a environ 30 commandes pour manipuler HDFS

- Lancer les commandes HDFS : **hadoop fs**

**-cat** : affiche le contenu d'un fichier

**-text** : comme cat mais sait afficher des données compressées

**-chgrp, -chmod, -chown** : modification des permissions

**-put, -get, -copyFromLocal, -copyToLocal** : import / export entre le système de fichier local et HDFS

**-ls, -lsr** : liste les fichiers / répertoires

**-mv, -moveFromLocal, -moveToLocal** : déplace les fichiers

**-stat** : informations statistiques sur les ressources (taille des blocs, nombre de blocs, type de fichiers, etc.)

# HDFS : Les commandes

- Il y a environ 30 commandes pour manipuler HDFS
- Lancer les commandes HDFS : **hadoop fs**

**-cat** : affiche le contenu d'un fichier

**-text** : comme cat mais sait afficher des données compressées

**-chgrp, -chmod, -chown** : modification des permissions

**-put, -get, -copyFromLocal, -copyToLocal** : import / export entre le système de fichier local et HDFS

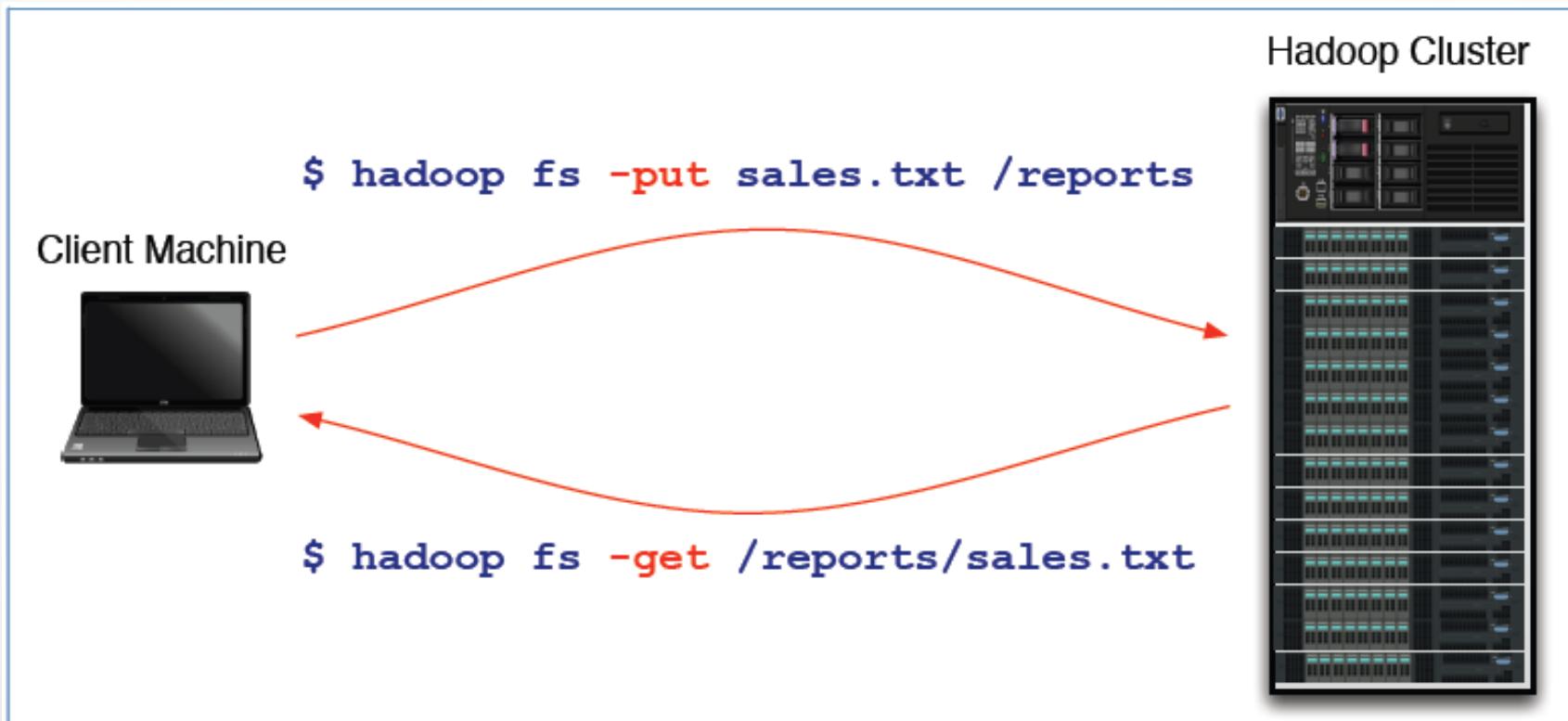
**-ls, -lsr** : liste les fichiers / répertoires

**-mv, -moveFromLocal, -moveToLocal** : déplace les fichiers

**-stat** : informations statistiques sur les ressources (taille des blocs, nombre de blocs, type de fichiers, etc.)

# HDFS : Les commandes

## ➤ Quelques commandes pour HDFS



# HDFS : Les commandes

## ➤ Quelques commandes pour HDFS

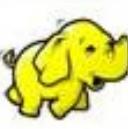
```
$ hadoop fs –ls /user/brian/  
$ hadoop fs -lsr  
$ hadoop fs –mkdir notes  
$ hadoop fs –put ~/training/commands.txt notes  
$ hadoop fs –chmod 777 notes/commands.txt  
$ hadoop fs –cat notes/commands.txt | more  
$ hadoop fs –rm notes/*.txt
```

# HDFS

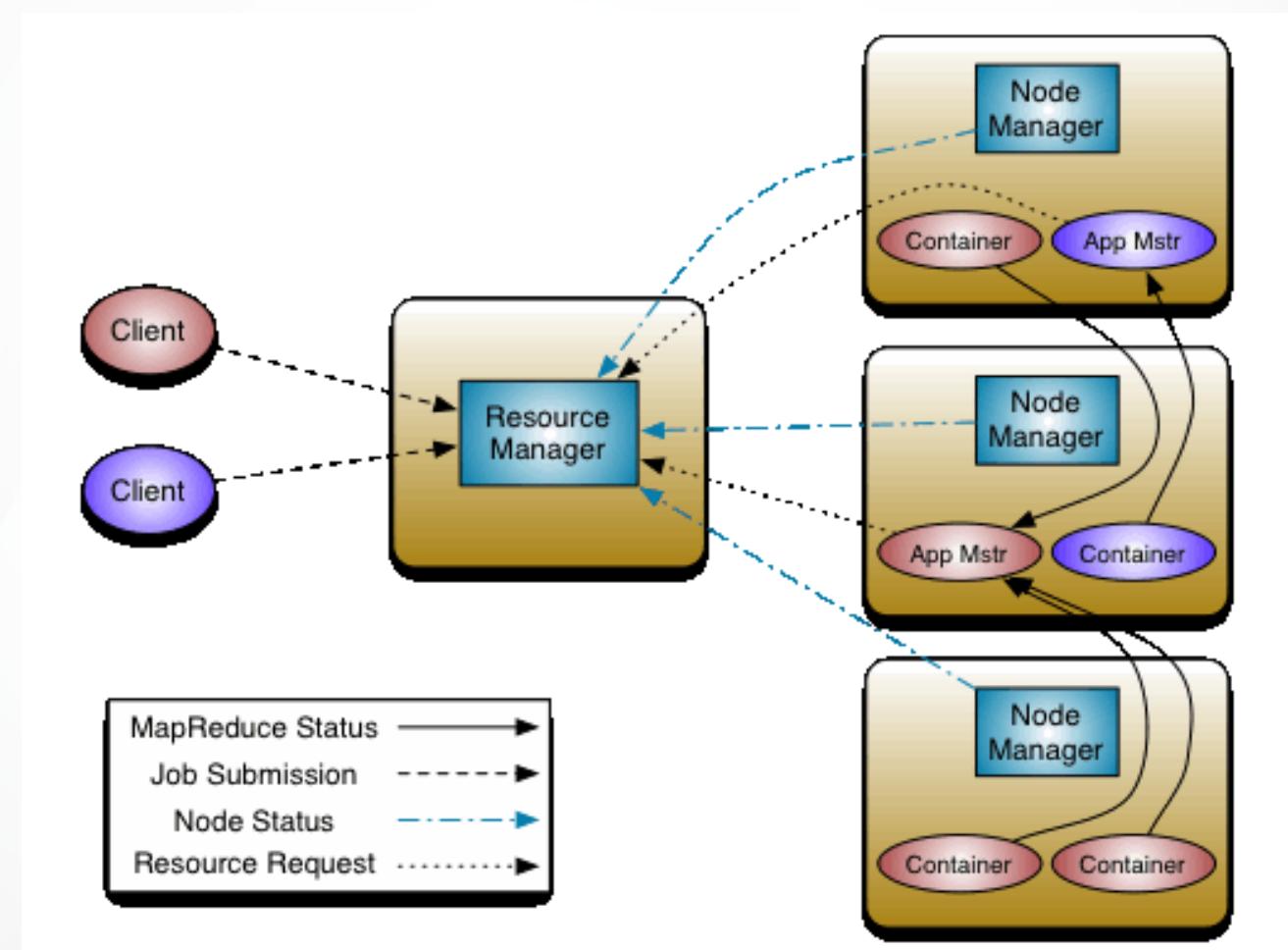


# Yarn

 **Ambari**  
Provisioning, Managing and Monitoring Hadoop Clusters

 <b>Sqoop</b> Data Exchange	 <b>Zookeeper</b> Coordination	 <b>Oozie</b> Workflow	 <b>Pig</b> Scripting	 <b>Mahout</b> Machine Learning	 <b>R Connectors</b> Statistics	 <b>Hive</b> SQL Query	 <b>Hbase</b> Columnar Store
 <b>Flume</b> Log Collector				 <b>YARN Map Reduce v2</b> Distributed Processing Framework			 <b>HDFS</b> Hadoop Distributed File System

# Yarn : Architecture



## ➤ **Le ResourceManager :**

- Le ResourceManager est le remplaçant du JobTracker du point de vue du client qui soumet des jobs (ou plutôt des applications en Hadoop 2) à un cluster Hadoop.
- Il n'a maintenant plus que deux tâches bien distinctes à accomplir :
  - Scheduler
  - ApplicationsManager

## ➤ **Le Scheduler :**

- Le Scheduler est responsable de l'allocation des ressources des applications tournant sur le cluster.
- Il s'agit uniquement d'ordonnancement et d'allocation de ressources.
- Les ressources allouées aux applications par le Scheduler pour leur permettre de s'exécuter sont appelées des Containers.
- Un Container désigne un regroupement de mémoire, de cpu, d'espace disque, de bande passante réseau, ...

- **L'ApplicationsManager :**
  - Lancer et au besoin relancer des jobs
  - Négocier les Containers nécessaires auprès du Scheduler
  - Superviser l'état et la progression des jobs.

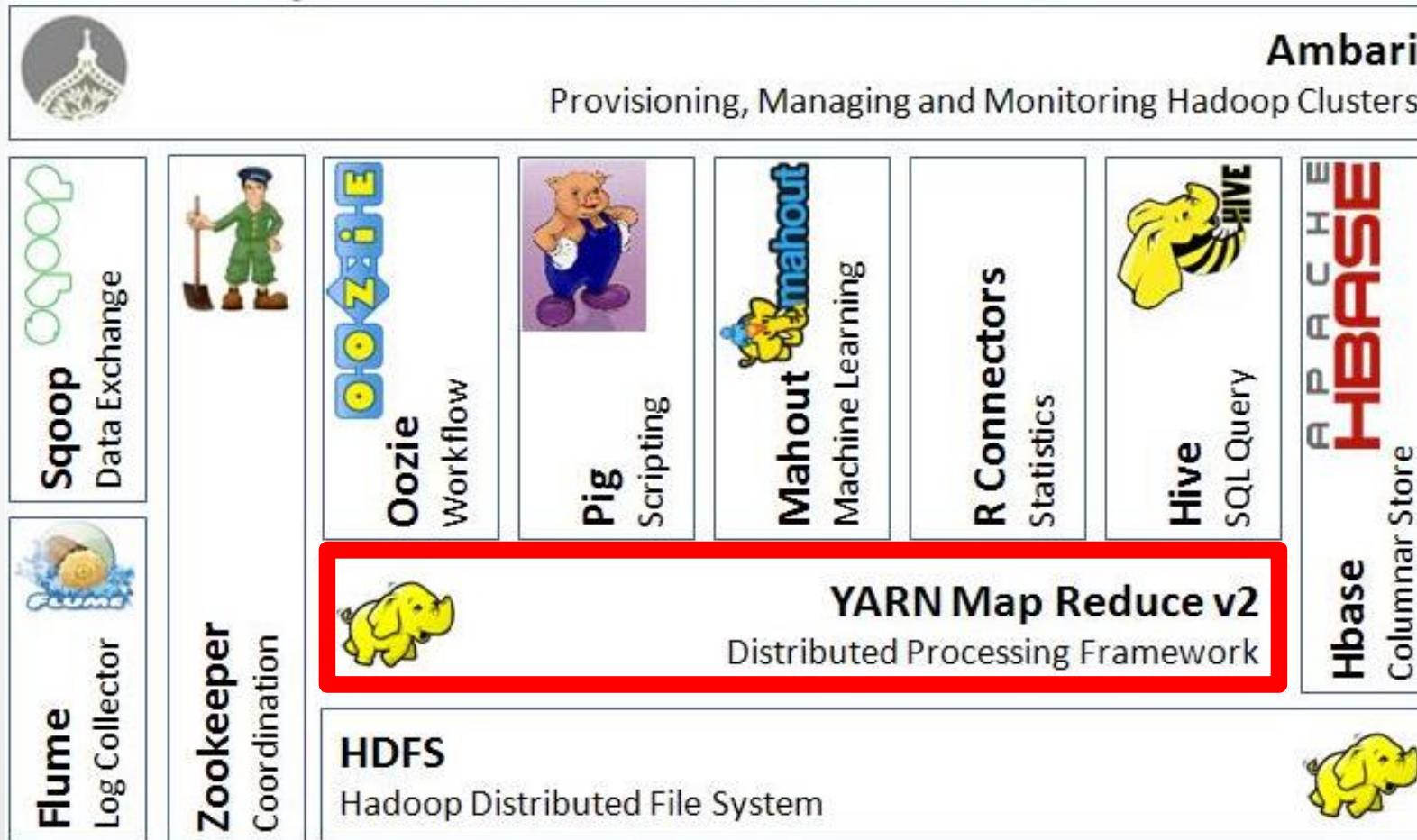


Global Knowledge®



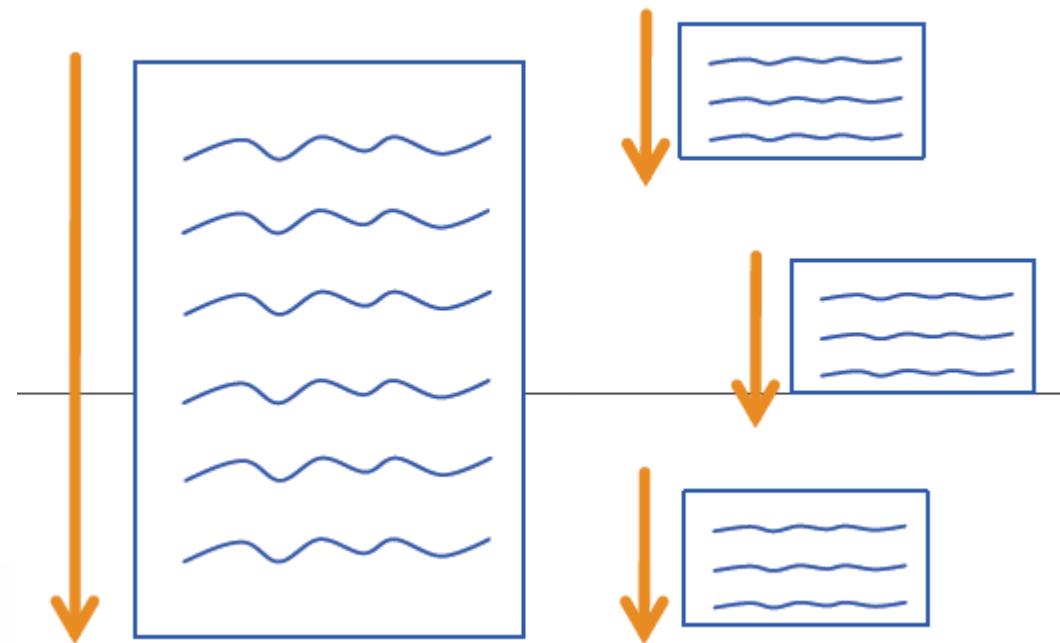
MapReduce

# MapReduce



# MapReduce

- Patron d'architecture de développement permettant de traiter des données volumineuses de manière parallèle et distribuée.
- A la base, le langage Java est utilisé, mais grâce à une caractéristique de Hadoop appelée Hadoop Streaming, il est possible d'utiliser d'autres langages comme Python ou Ruby.
- Au lieu de parcourir le fichier séquentiellement (bcp de temps), il est divisé en morceaux qui sont parcourus en parallèle.



# MapReduce

## ➤ Map

- Itérer sur un grand nombre d'enregistrements
- Extraire quelque chose ayant un intérêt de chacun d'eux

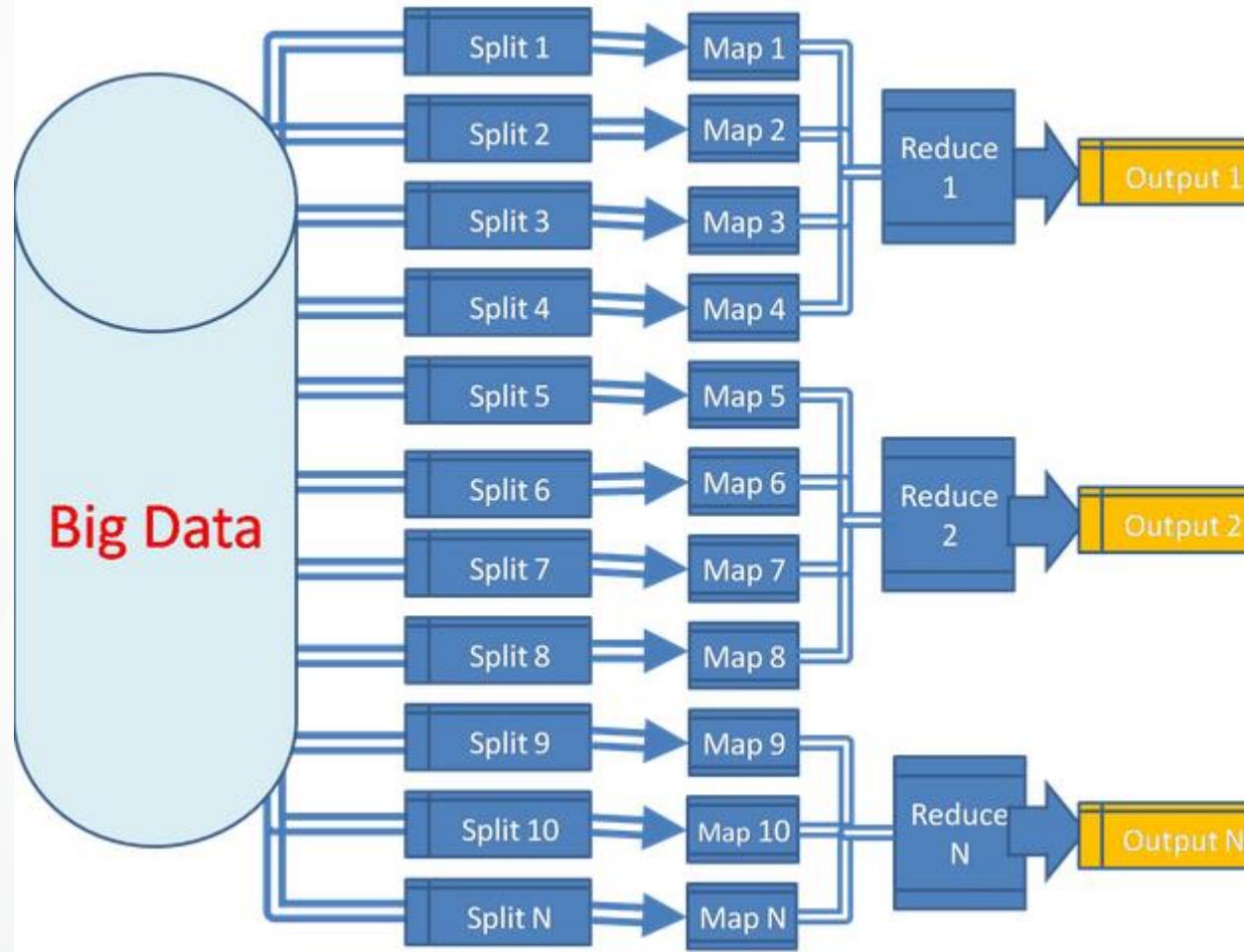
## ➤ Reduce

- Regrouper et trier les résultats intermédiaires
- Agréger ces résultats
- Générer le résultat final

# MapReduce

- Traitement parallélisé
  - Le nombre de « maps » est déterminé par le nombre de blocks DFS du fichier d'entrée (~10-100 maps/node)
  - Le nombre de « reduces » est par défaut 1 sinon peut être renseigné dans le programme Reduce (~1000 X taille du buffer)

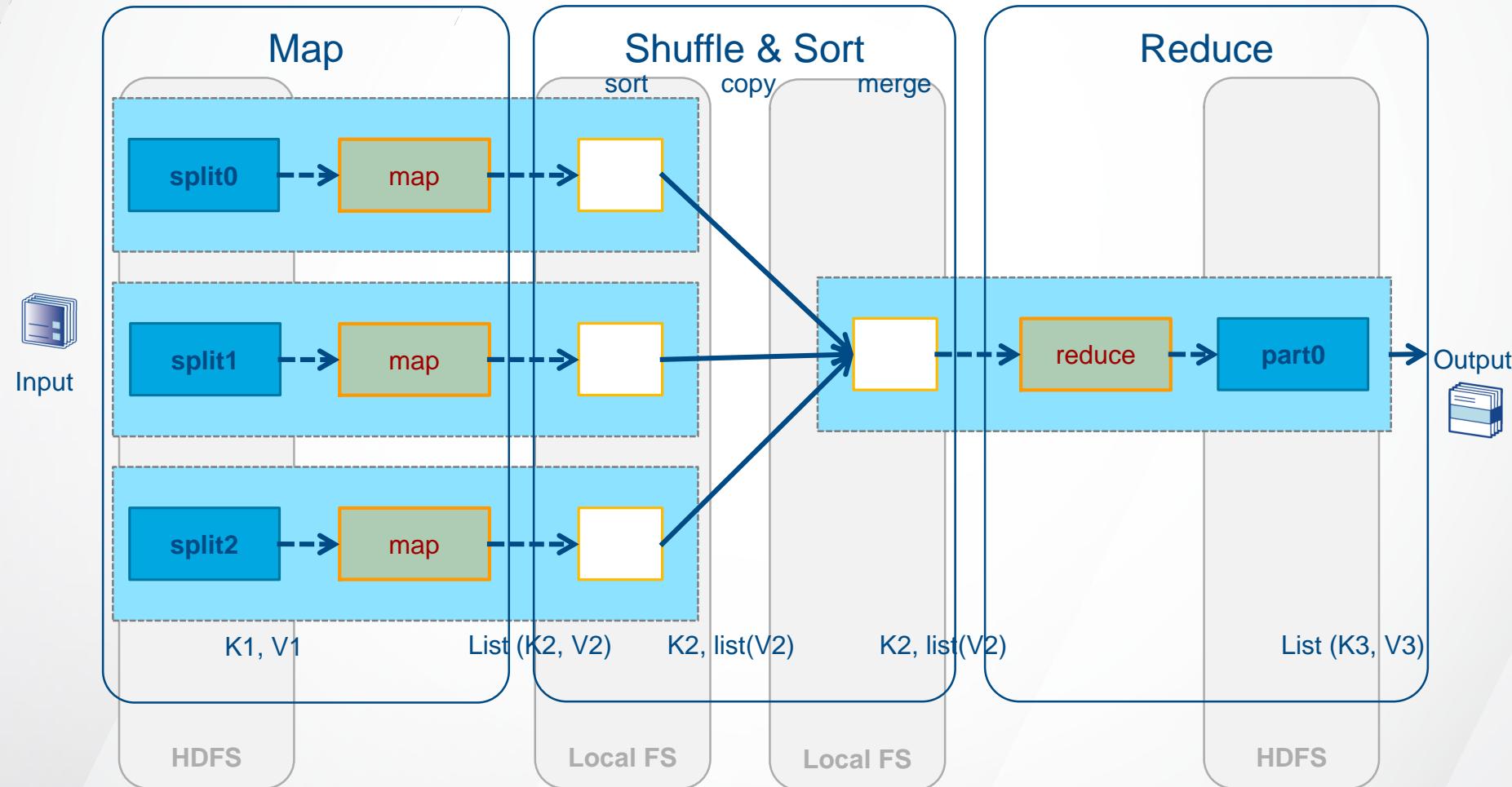
# MapReduce : Principles



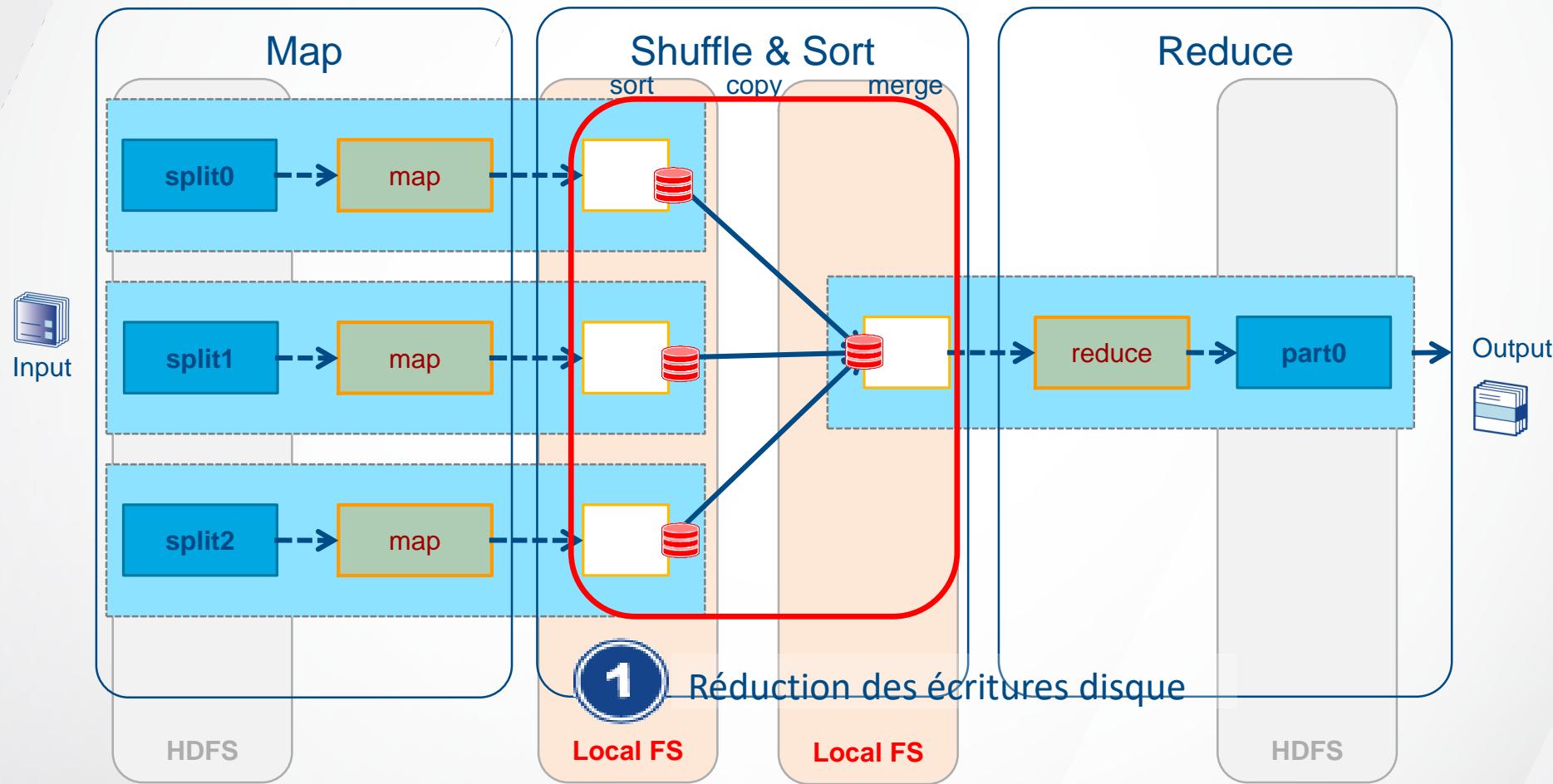
# MapReduce : Exemple 1



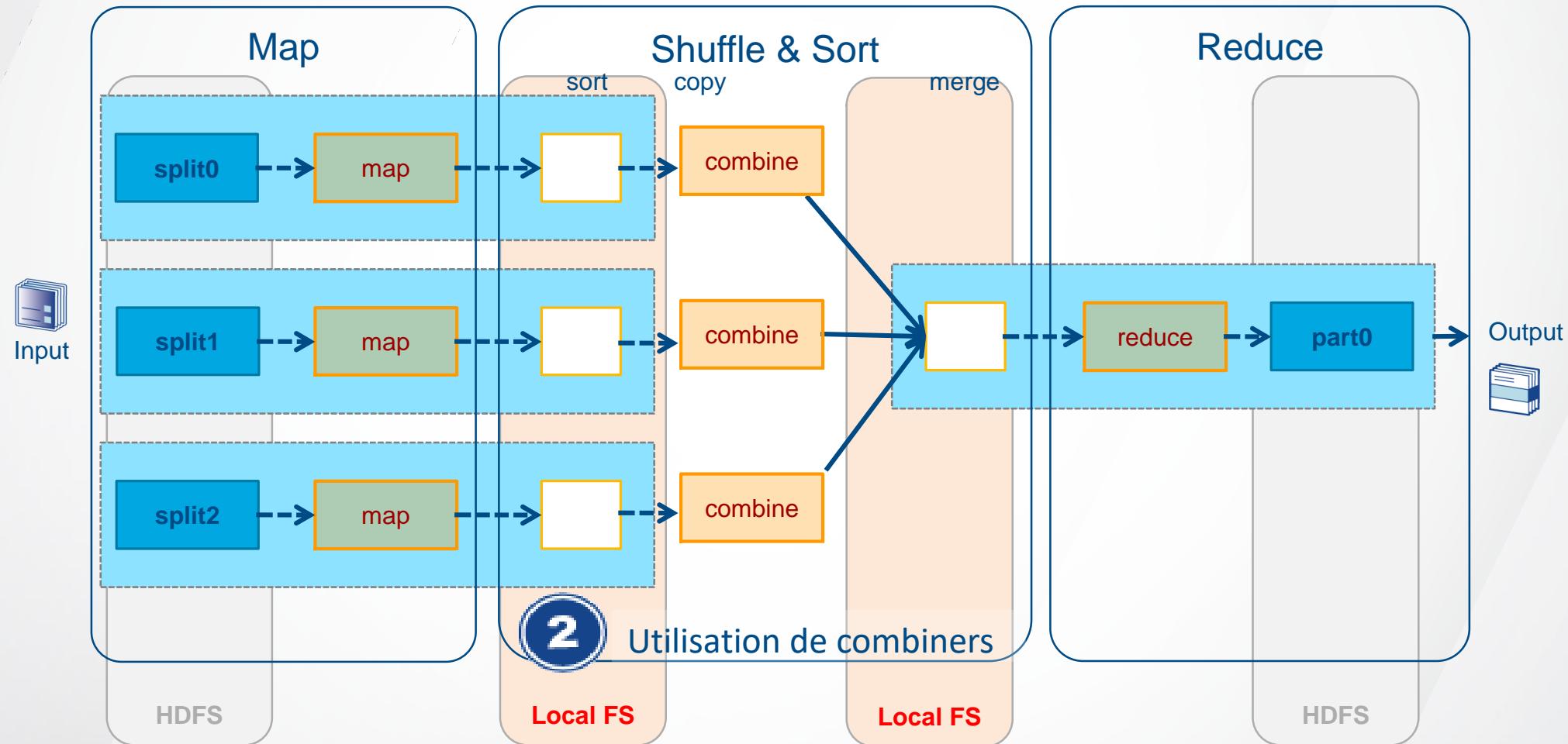
# MapReduce : Principe



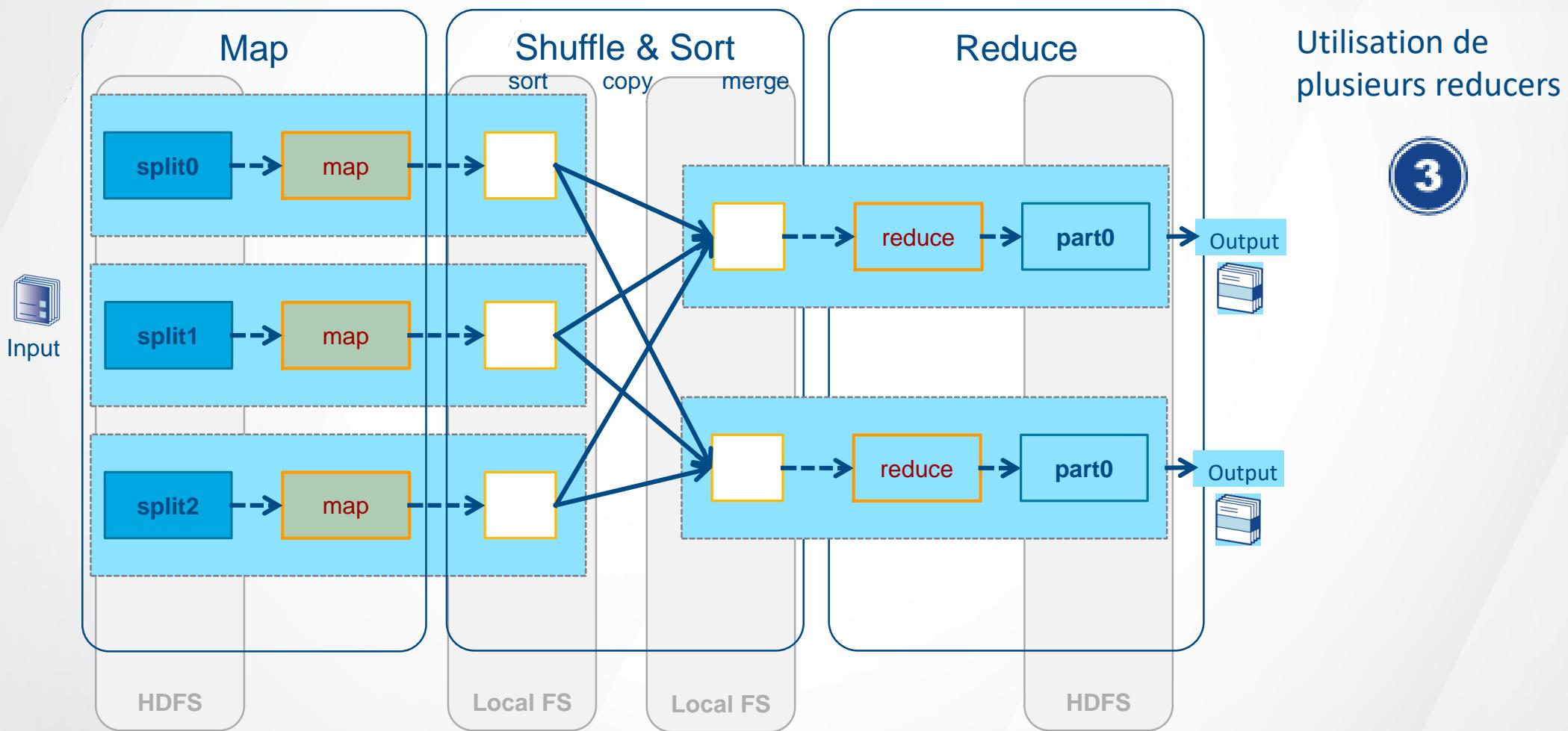
# MapReduce : Principe



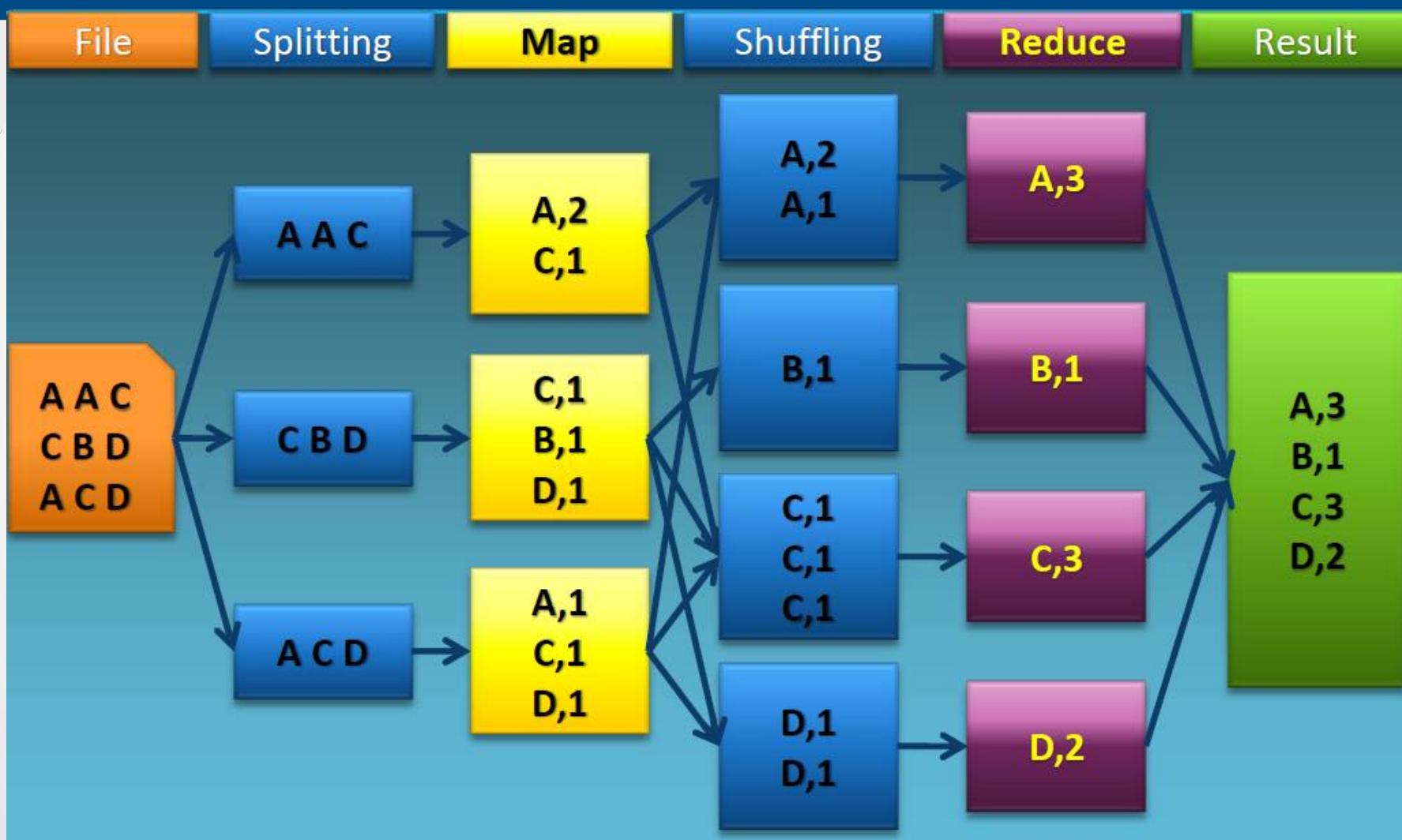
# MapReduce : Principe



# MapReduce : Optimisation



# MapReduce : Exemple 2



# Comptage des mots avec Java et Hadoop

```
import java.io.IOException;

public class WordCount {

    public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
}
```



Global Knowledge®

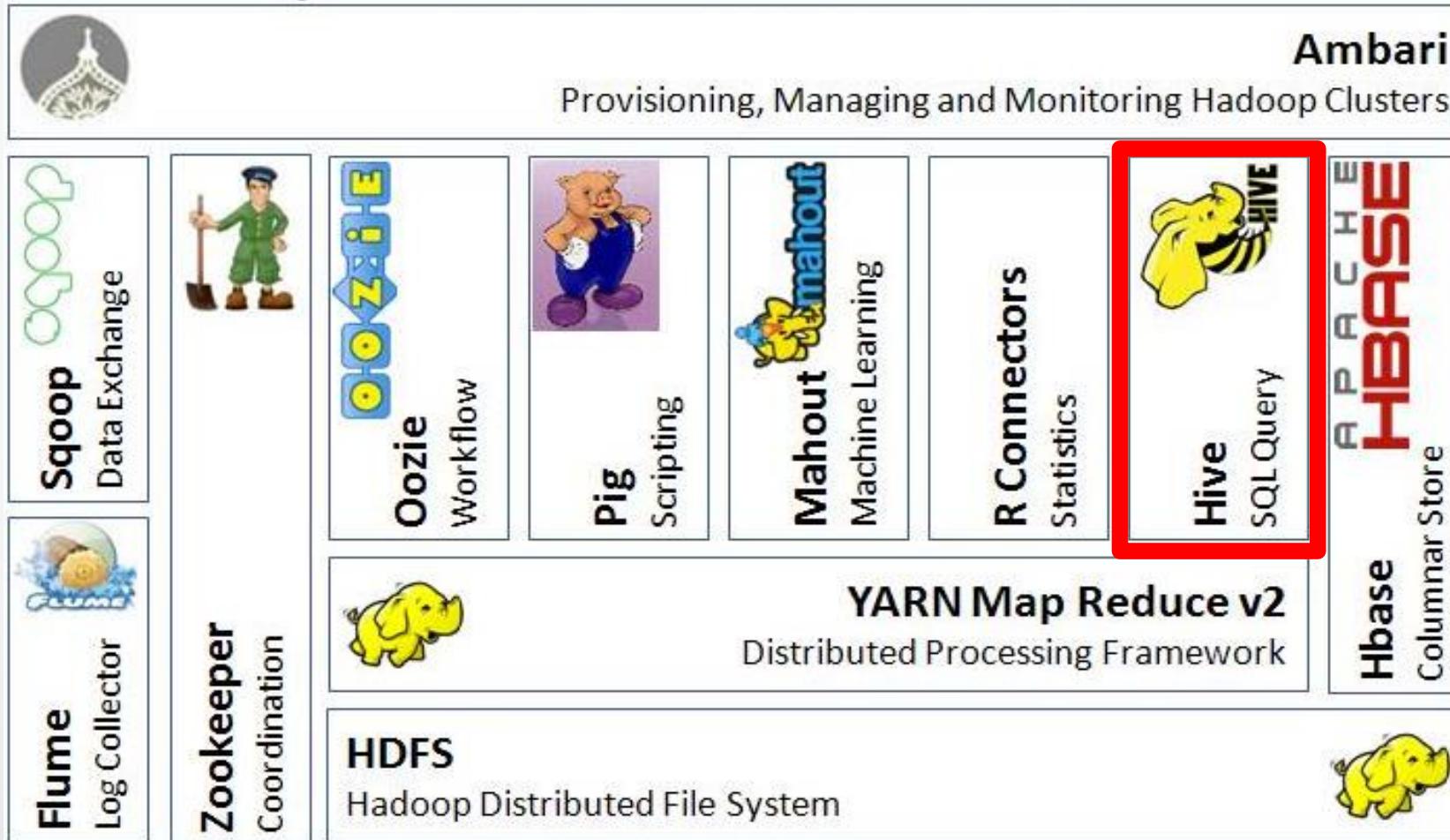
HIVE



# Hive

- Introduction de Hive
- Gestion des tables
- Gestion des données
- Gestion des requêtes (Query)
- Architecture
- Jointures

# Hive

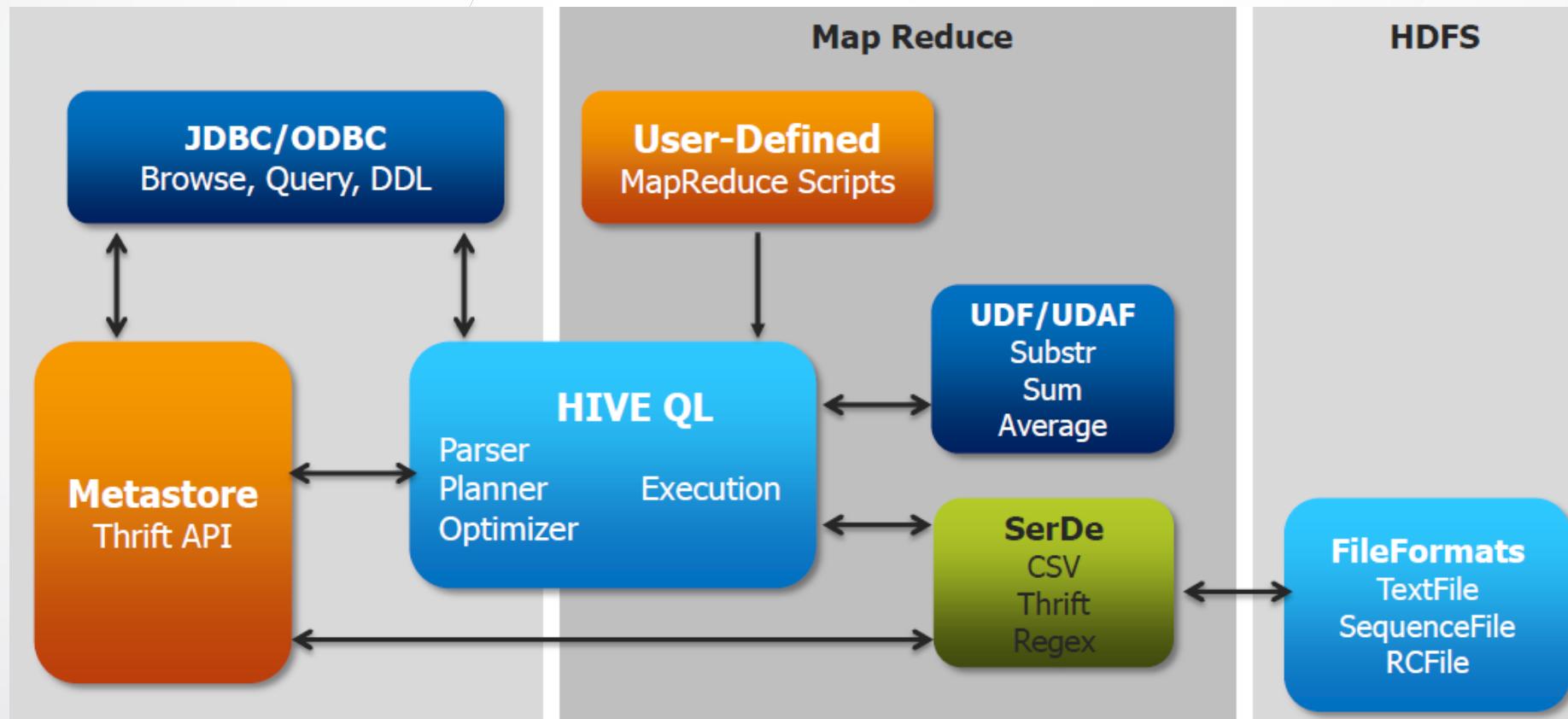


# Qu'est-ce que Hive ?

---

- Entrepôt de données pour Hadoop Langage semblable à SQL, appelé HiveQL (HQL)
- N'est pas conçu pour :
  - Le traitement des transactions en ligne
  - Des requêtes temps réel

# Architecture de Hive



# Utilisation de Hive

## ➤ HiveQL supporte :

- DDL (Create, Alter, Drop)
- DML (Load, Insert, Select)
- Fonctions utilisateurs
- Appel à des programmes externe MapReduce

# Hive : Gestion des tables

## ➤ Utilisation d'opérations Hive Data Definition Language (DDL)

Opération	Commande
Création	CREATE TABLE db_name;
Description	DESCRIBE db_name;
Liste	SHOW TABLES;
Modification	ALTER TABLE db_name ADD COLUMNS (field1 STRING);
Suppression	DROP TABLE db_name;

# Hive : exemple de création de table

```
CREATE DATABASE IF NOT EXISTS airlinedb ;  
CREATE EXTERNAL TABLE IF NOT EXISTS airlinedb.airlinetable  
  (AirlineID int  
  ,AirlineName string  
  ,AirlineAlias string  
  ,IATA string  
  ,ICAO string  
  ,CallSign string  
  ,Country string  
  ,Active string)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

# Hive : Gestion des données

## ➤ Utilisation d'opérations Hive Data Manipulation Language (DML)

Opération	Commande
Chargement des données locales	<code>LOAD DATA LOCAL INPATH './files/data_db_name.txt' OVERWRITE INTO TABLE db_name;</code>
Chargement des données HDFS	<code>LOAD DATA INPATH '/user/myname/data_db_name.txt' OVERWRITE INTO TABLE db_name;</code>

# Hive : Gestion des requêtes (Query)

## ➤ Utilisation d'opérations SQL

Opération	Commande
Retrieving information	SELECT from_columns FROM table WHERE conditions;
All values	SELECT * FROM table;
Some values	SELECT * FROM table WHERE rec_name = "value";
Multiple criteria	SELECT * FROM TABLE WHERE rec1 = "value1" AND rec2 = "value2";
Selecting specific columns	SELECT column_name FROM table;
Retrieving unique output records	SELECT DISTINCT column_name FROM table;
Sorting	SELECT col1, col2 FROM table ORDER BY col2;
Counting rows	SELECT COUNT(*) FROM table;
Grouping with counting	SELECT owner, COUNT(*) FROM table GROUP BY owner;

# Hive : Créer une table

```
hive> CREATE TABLE employes (id INT, prenom STRING, nom STRING)
```

1ere ligne : créer une table de 3 colonnes

```
> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ;
```

2eme ligne : indique que le séparateur  
entre les champs est une virgule

La commande doit finir par un  
point virgule pour l'exécuter

---

```
hive> DESCRIBE employes;
```

Affiche la structure de la table

id	int
prenom	string
nom	string

# Hive : Charger les données dans la table

```
hive> LOAD DATA LOCAL INPATH 'data/data-employes.txt'
```

Copier les valeurs à partir d'un fichier local data-employes.txt

```
> OVERWRITE INTO TABLE employes;
```

Les enregistrements éventuels de la table employes sont supprimés

---

```
$ hdfs dfs -cat /user/hive/warehouse/employes/data-employes.txt
```

```
100,jean,martin
```

```
200,robert,poulain
```

```
300,michel,dunot
```

Afficher les enregistrements copiés.

Les tables Hive sont stockées par défaut dans le répertoire /user/hive/warehouse

# Hive : Effectuer des requêtes simples

```
hive> SELECT COUNT(*) FROM employes;
```

Compter le nombre d'enregistrements de la table employes

```
hive> SELECT * FROM employes WHERE prenom = "Robert";
```

Afficher les enregistrements satisfaisant un critère

# Hive : Supprimer une table

```
hive> DROP TABLE employes;
```

Supprimer la table sélectionnée

```
$ hdfs dfs -ls /user/hive/warehouse/
```

La table a été supprimée

# Techniques performantes avec Hive

- Record Columnar File format (RCFile)
  - Format row-columnar hybride qui permet une analyse efficace lorsque seul un sous-ensemble des données est nécessaire
- Partition : définir ses propres colonnes, son stockage et sa sérialisation
  - Les partitions permettent d'accélérer les requêtes des partitions
  - Sont physiquement stockées dans des répertoires séparés dans HDFS
- Buckets : découpage des partitions
  - Optimisation pour les jointures

# Hive : Exemple de partitions

```
CREATE EXTERNAL TABLE clicks (
    hms          STRING,
    hostname     STRING,
    process      STRING,
    pid          INT,
    uid          INT,
    message      STRING)
PARTITIONED BY (
    year         INT,
    month        INT,
    day          INT);
```

# Gestion de table avec Hive





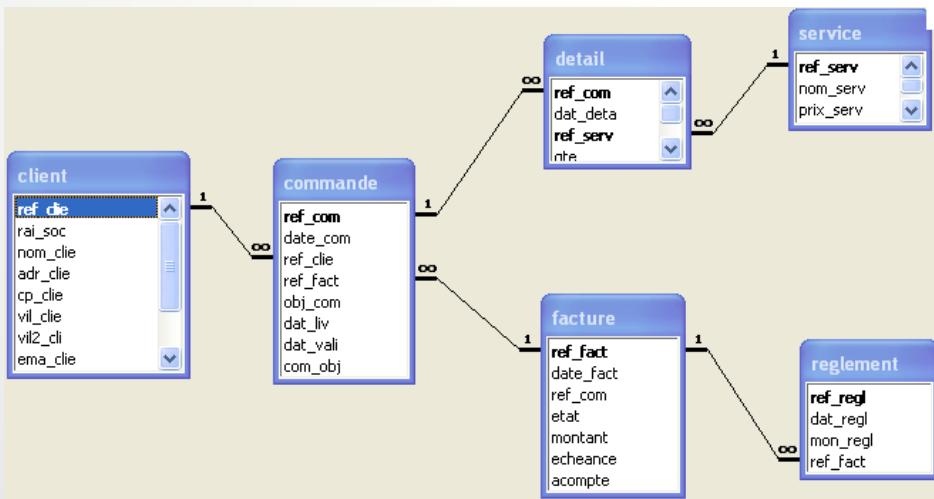
Global Knowledge®



NOSQL

# Les bases données relationnelles

- Algèbre Relationnelle
- Théorie des ensembles
- +40 ans 'optimisations



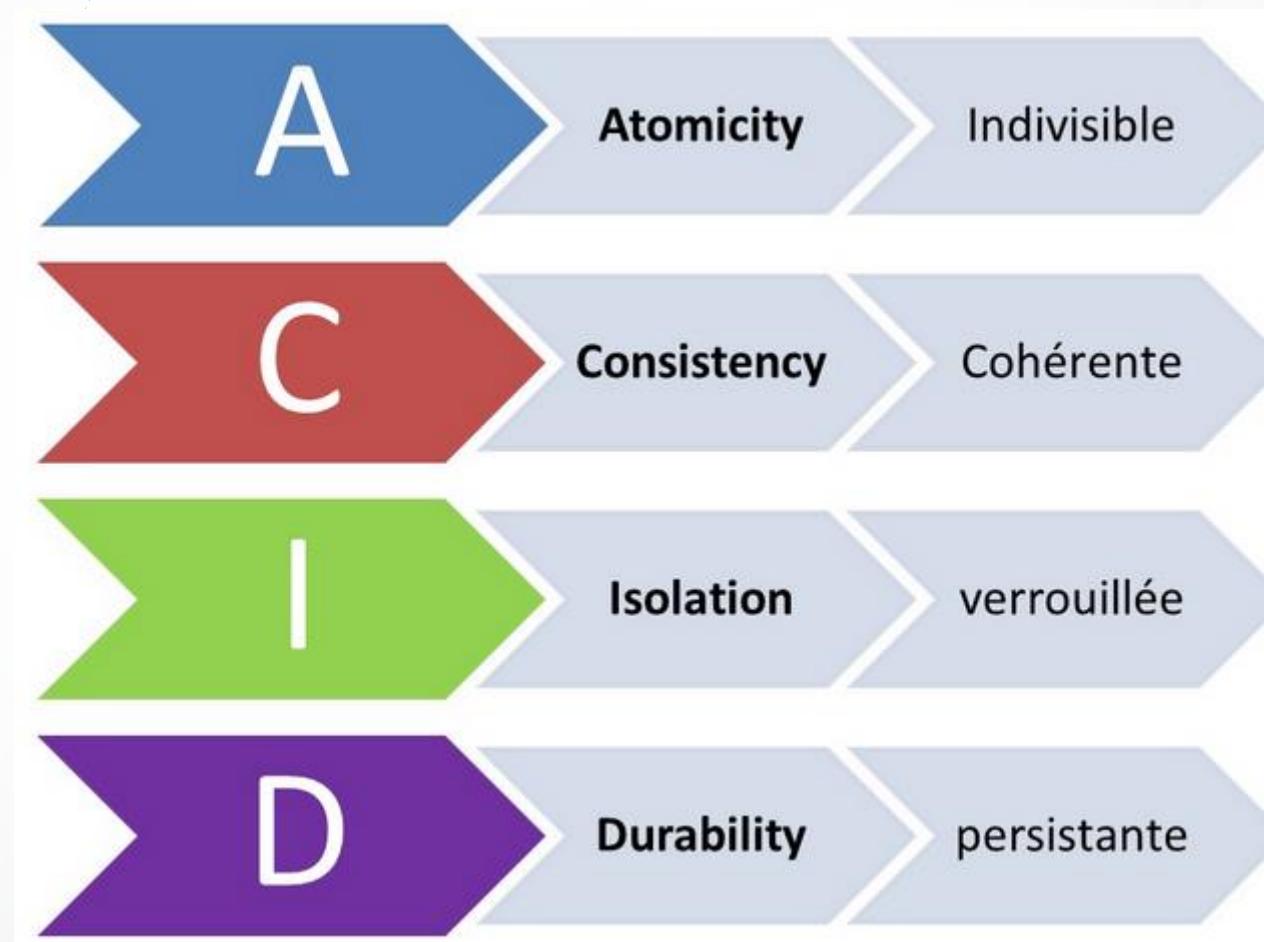
Ceux qui comprennent la théorie des ensembles

????

Ceux qui ne comprennent pas la théorie des ensembles

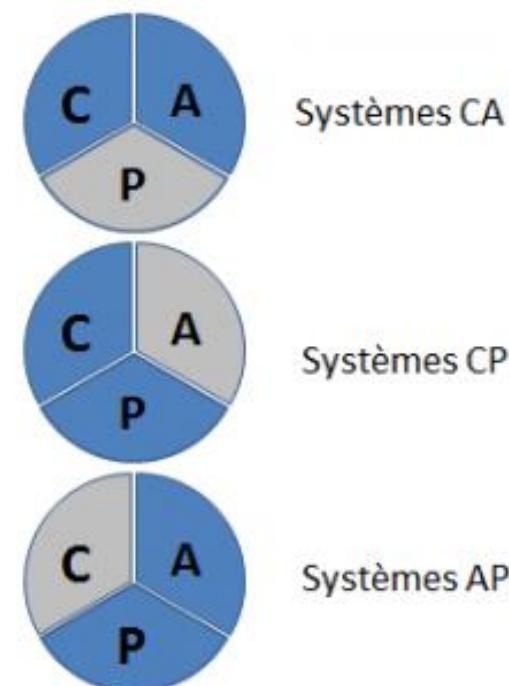
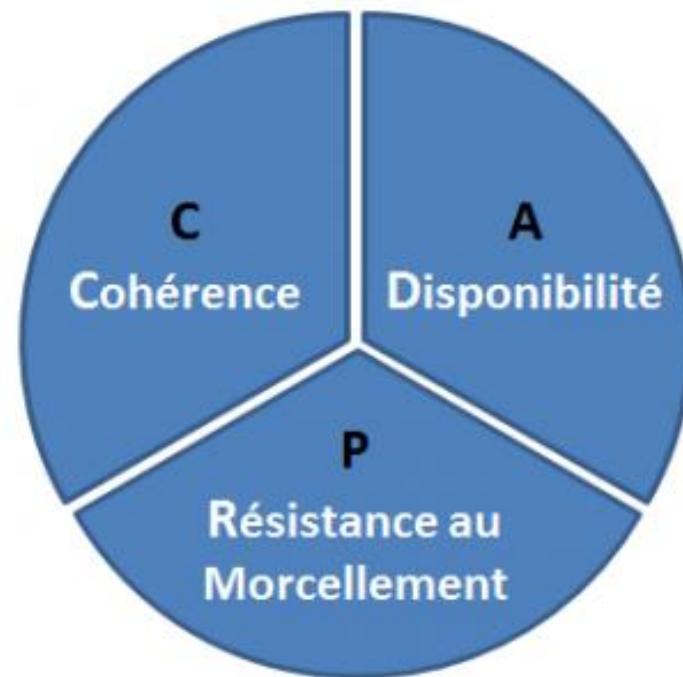
# Les bases données relationnelles

## ➤ Transaction ACID



# Les bases de données NOSQL

## ➤ Théorème de CAP

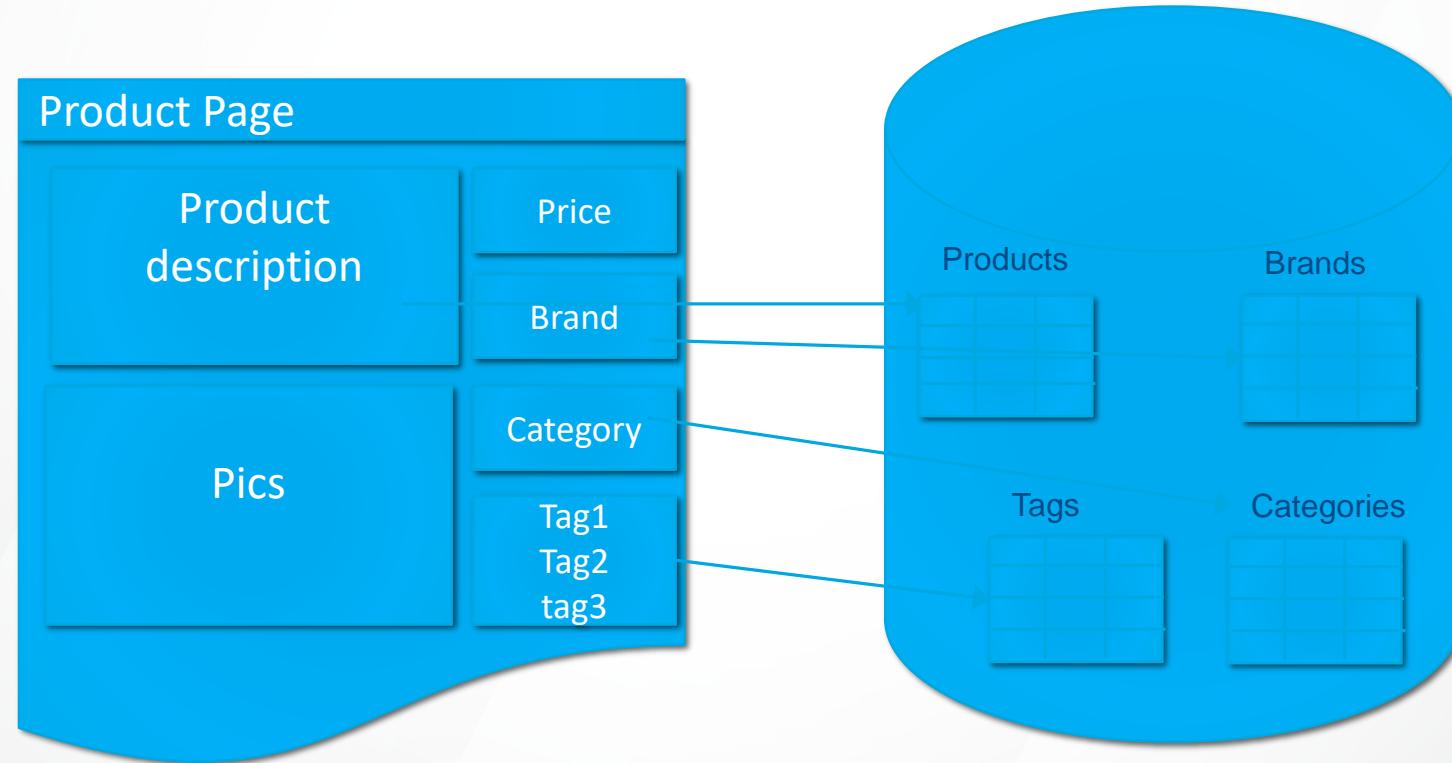


### Théorème de CAP:

On ne peut obtenir à la fois que 2 des 3 Caractéristiques dans un système réparti

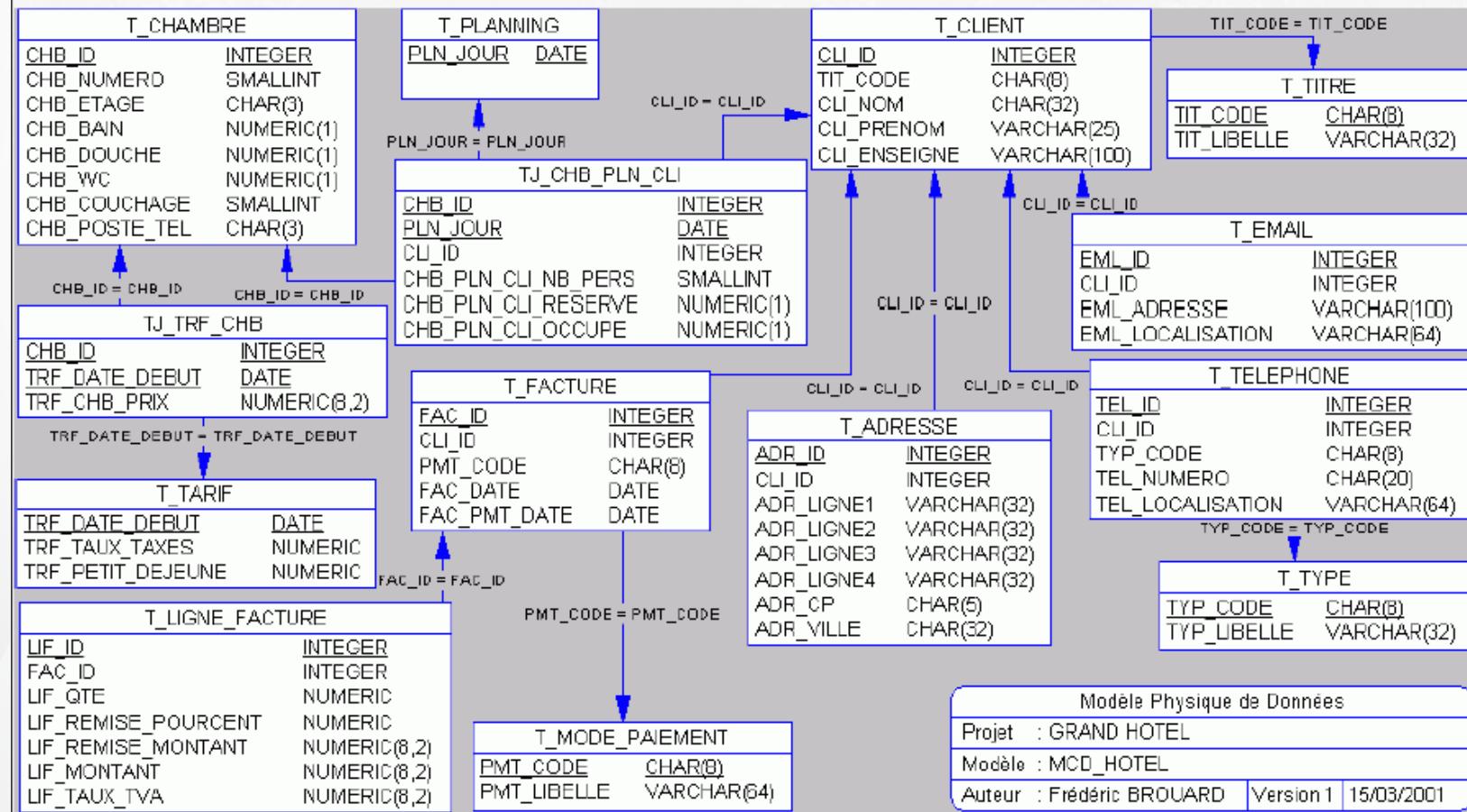
# Les limites des bases relationnelles

## ➤ Impedance mismatch



# Les limites des bases relationnelles

## ➤ Requête lent



# Les limites des bases relationnelles

- Scalabilité
- Le modèle de consistance des RDBMS empêche l'utilisation de plusieurs machines pour répartir la charge (au moins en écriture)
- Pour augmenter la performance d'accès à la base, pas d'autres moyens que d'acheter un plus gros serveur, puis un autre, puis un autre, ...



# Base de données NOSQL



Not Only SQL

# Base de données NOSQL

**NO  
SQL**

## PROBLEME

- Les données provenant de sources hétérogènes.

## CONSEQUENCE

- Données non structurées que l'on va essayer d'organiser pour pouvoir les traiter.
- La diversité initiale empêche de respecter un format rigide de donnée finale

## SOLUTION

- Besoin d'un design de la donnée performant et donc penser en fonction de l'usage plutôt que des relations

# Quel points communs à la plupart des bases NoSQL ?

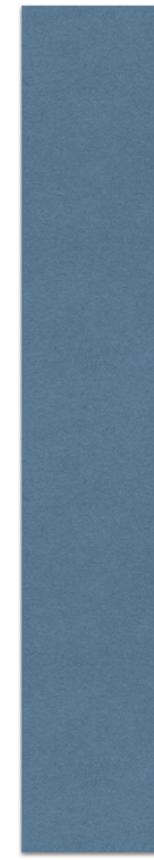
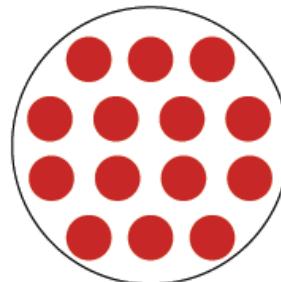
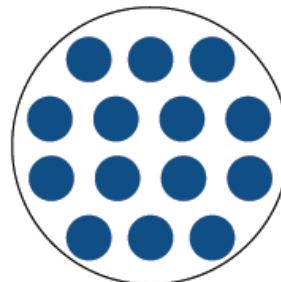
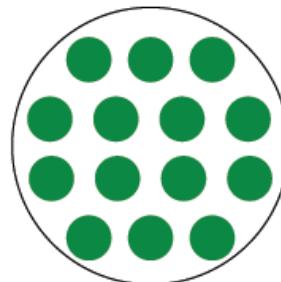
- Un schéma implicite
- L'open source
- L'absence de relations
- L'absence du langage SQL

# NOSQL : Atouts

- Principaux atouts :
  - / Évolutivité
  - Disponibilité
  - Tolérance aux fautes
- Caractéristiques :
  - Modèle de données sans schéma
  - Architecture distribuée
  - Utilisation de langages et interfaces qui ne sont pas uniquement du SQL

# Base de données NOSQL

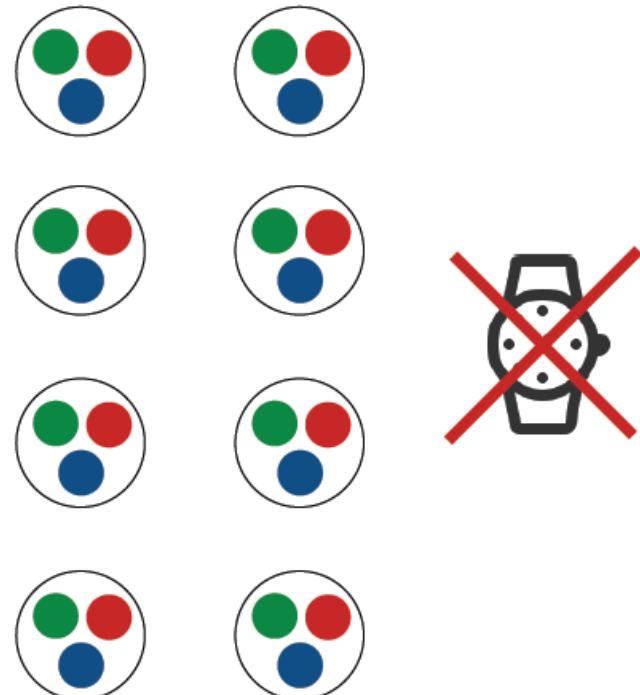
## La chocolaterie



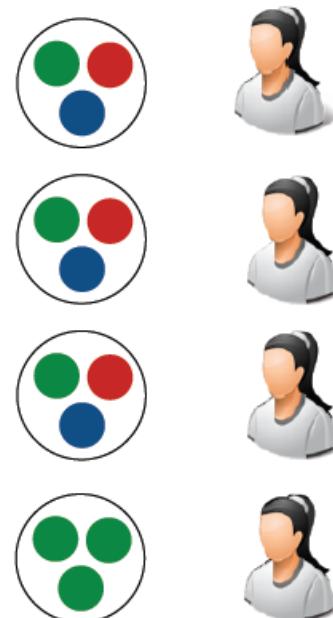
I-4. Penser usage

# Base de données NOSQL

## La chocolaterie



?

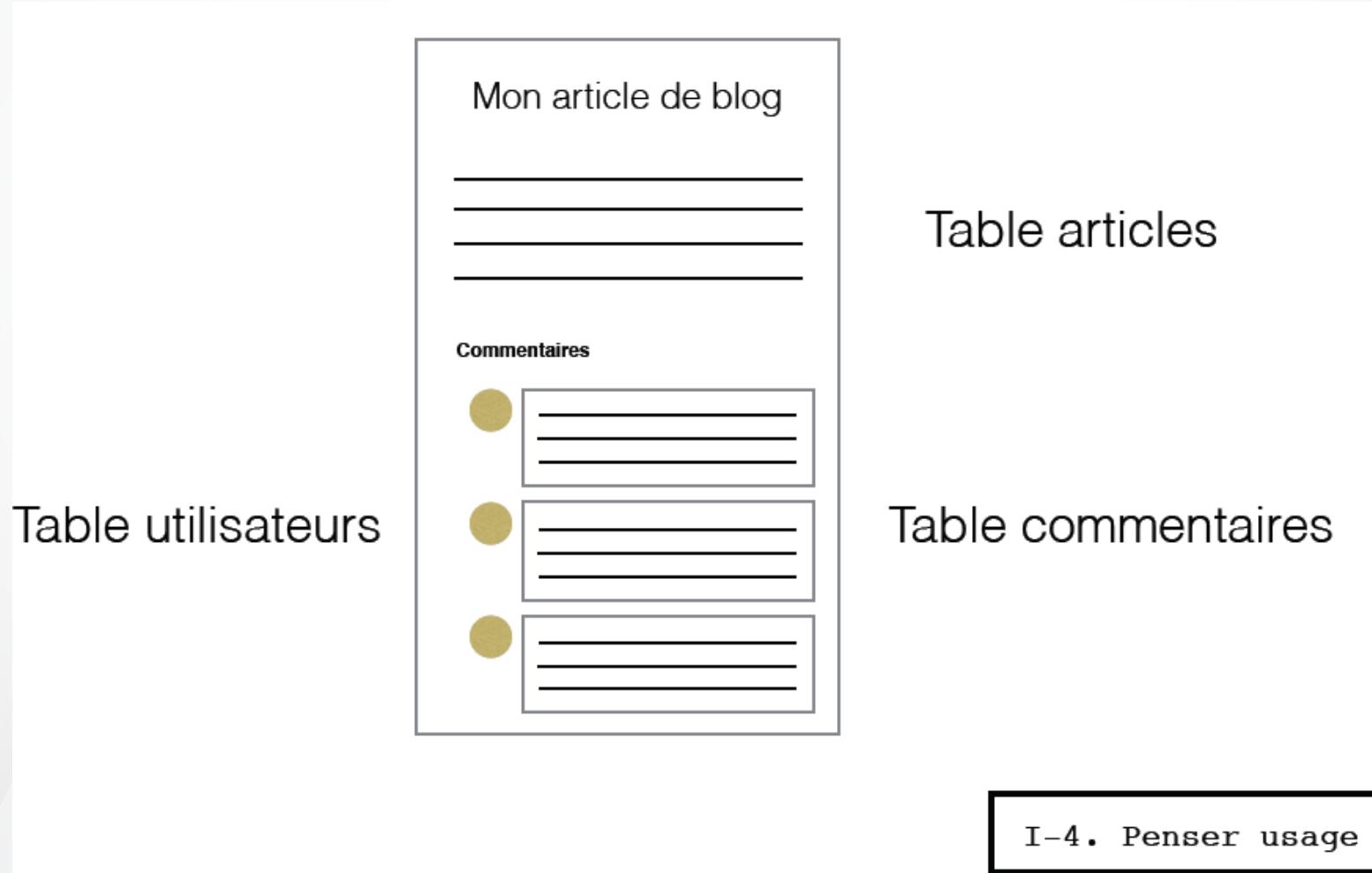


I-4. Penser usage

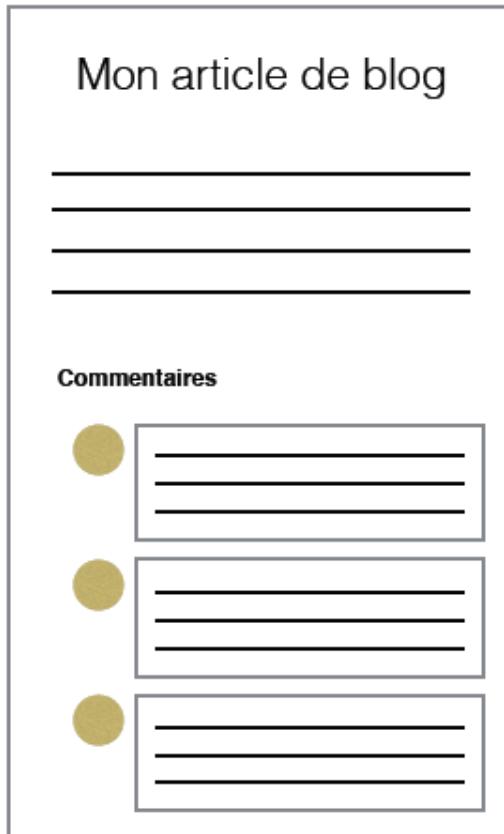
# Base de données NOSQL

- Comprendre la notion d'usage de la donnée
- Stocker les données en fonction des intentions d'usage
- Quelles sont les conséquences d'une mauvaise modélisation de la donnée

# Le relationnel dans la réalité

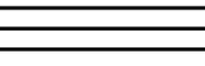


# Le relationnel dans la réalité



Stockage de donnée déjà packagées

Titre : Mon article de blog

Contenu : 

Commentaires : [  
  { user, contenu ... },  
  { user, contenu ... },  
  { user, contenu ... }]  
]

I-4. Penser usage

# A retenir

- Le NoSQL demande de penser usage :
  - Impossible de créer le schéma de la donnée sans connaître l'expérience utilisateur finale.
- NoSQL est très performant pour un usage précis :
  - En packageant toutes les données relatives à un usage précis, NoSQL ne requête qu'un emplacement au lieu de « n » emplacements.
- Le modèle relationnel est plus flexible :
  - Dans le modèle relationnel, la donnée peut être assemblée à souhait quelque soit la demande.
  - En NoSQL, un besoin non anticipé peut coûter très très cher en temps et donc en performance.

# Types de Bases de Données NOSQL

---

## ➤ Types des bases de données NOSQL

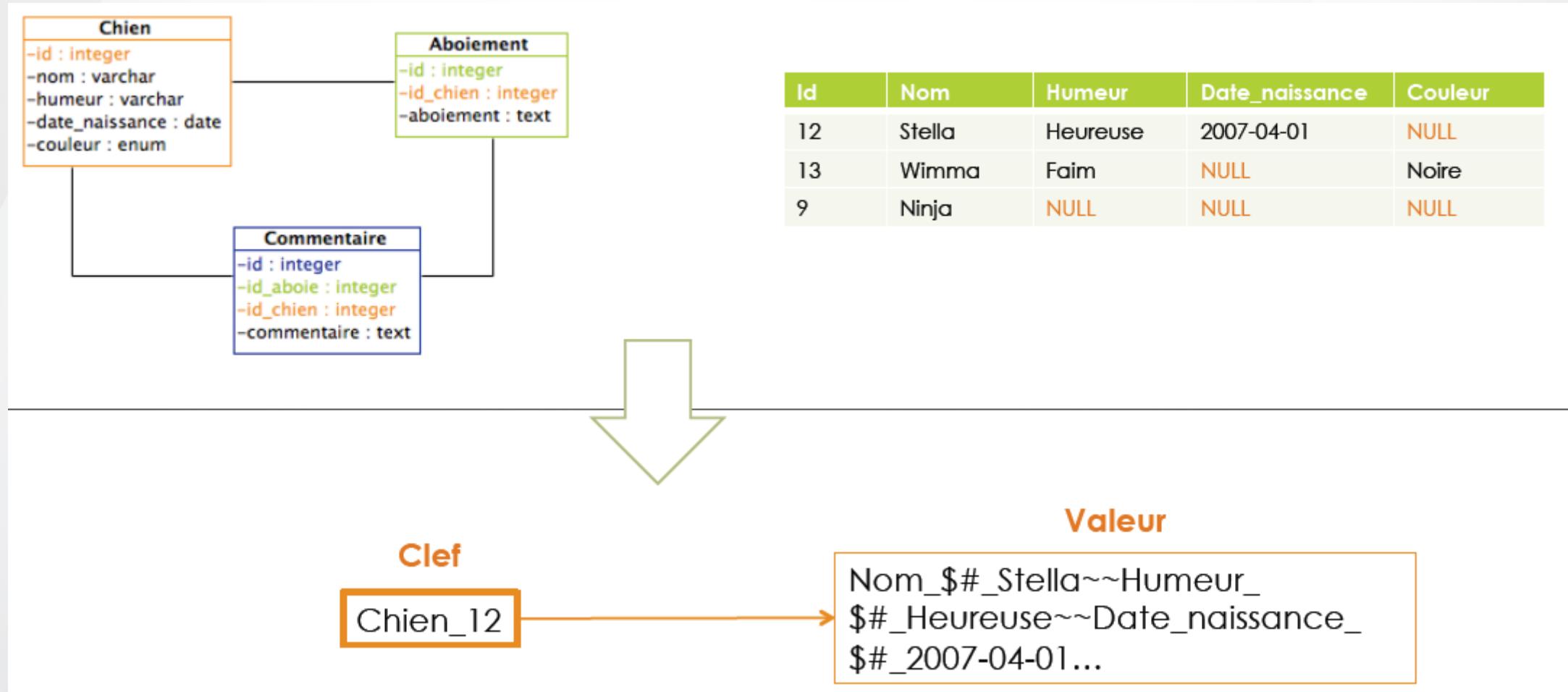
- Clef/valeur
- Orientées colonnes
- Orientées documents
- Orientées graphes

# Type de base de données NOSQL : clé/valeur

## Clé/valeur :

- Conçues pour sauvegarder les données **sans définir de schéma**
- Toutes les données sont sous forme de clé/valeur
  - La valeur peut être une chaîne de caractères, un objet sérialisé, un blob...
  - La donnée est opaque au système : il n'est pas possible d'y accéder sans passer par la clé
- Absence de typage a un impact sur le requêtage: toute l'intelligence portée avant par les requêtes devra être portée par l'applicatif qui interroge la BD
- Communications se résumant surtout aux opérations PUT, GET et DELETE  
Objectif: fournir un accès rapide aux informations, conserver la session d'un site web...
- Exemple : DynamoDB (Amazon)

# Base NOSQL Clef/valeur : Exemple



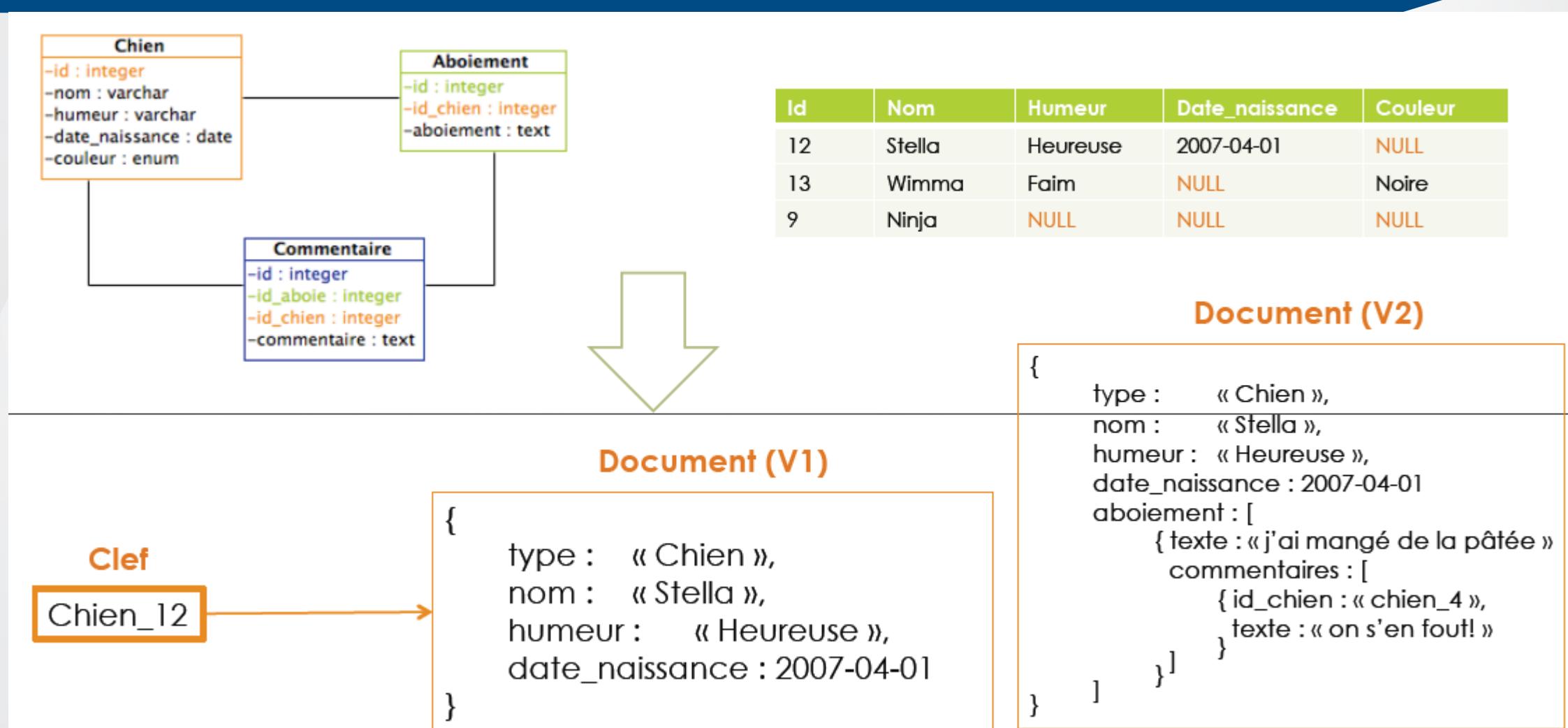
# Type de base de données NOSQL : Orientée Documents

## Orientée documents :

- Étendent le paradigme clef/valeur, avec des « documents » plus complexes à la place des données simples, et une clef unique pour chacun d'eux
- Documents de type JSON ou XML
- Chaque document est un objet, contient un ou plusieurs champs, et chaque
- champs contient une valeur typée (string, date, binary ou array)
- Permettent de stocker, extraire et gérer les informations orientées documents (données semi-structurées)
- Avantage : pouvoir récupérer, via une seule clef, un ensemble d'informations structurées de manière hiérarchique
  - Dans les bases relationnelles, cela impliquerait plusieurs jointure
- Exemples : Mongo DB (SourceForge), Couch DB (Apache)

# Base NOSQL Orientée documents :

## Exemple



# Type de base de données NOSQL : Orientée Colonne

Orientée colonnes :

- Évolution de la BD clé/valeur
- Ressemble aux SGBDR, mais avec un nombre de colonnes dynamiques, différent d'un enregistrement à un autre (pas de colonnes portant les valeurs NULL)
- Offre de très hautes performances et une architecture hautement évolutive
- Exemples : Exemples: Hbase (Hadoop), Cassandra (Facebook, Twitter), BigTable (Google)

# Type de base de données NOSQL : Orientée Colonnes

## ➤ Orientée colonnes

	A	B	C	D	E
1	Foo	Bar	Hello		
2		Tom			
3			Java	Scala	Cobol

1	A	Foo	B	Bar	C	Hello
2	B	Tom				
3	C	Java	D	Scala	E	Cobol

# Type de base de données NOSQL : Orientée Colonnes



## Requête

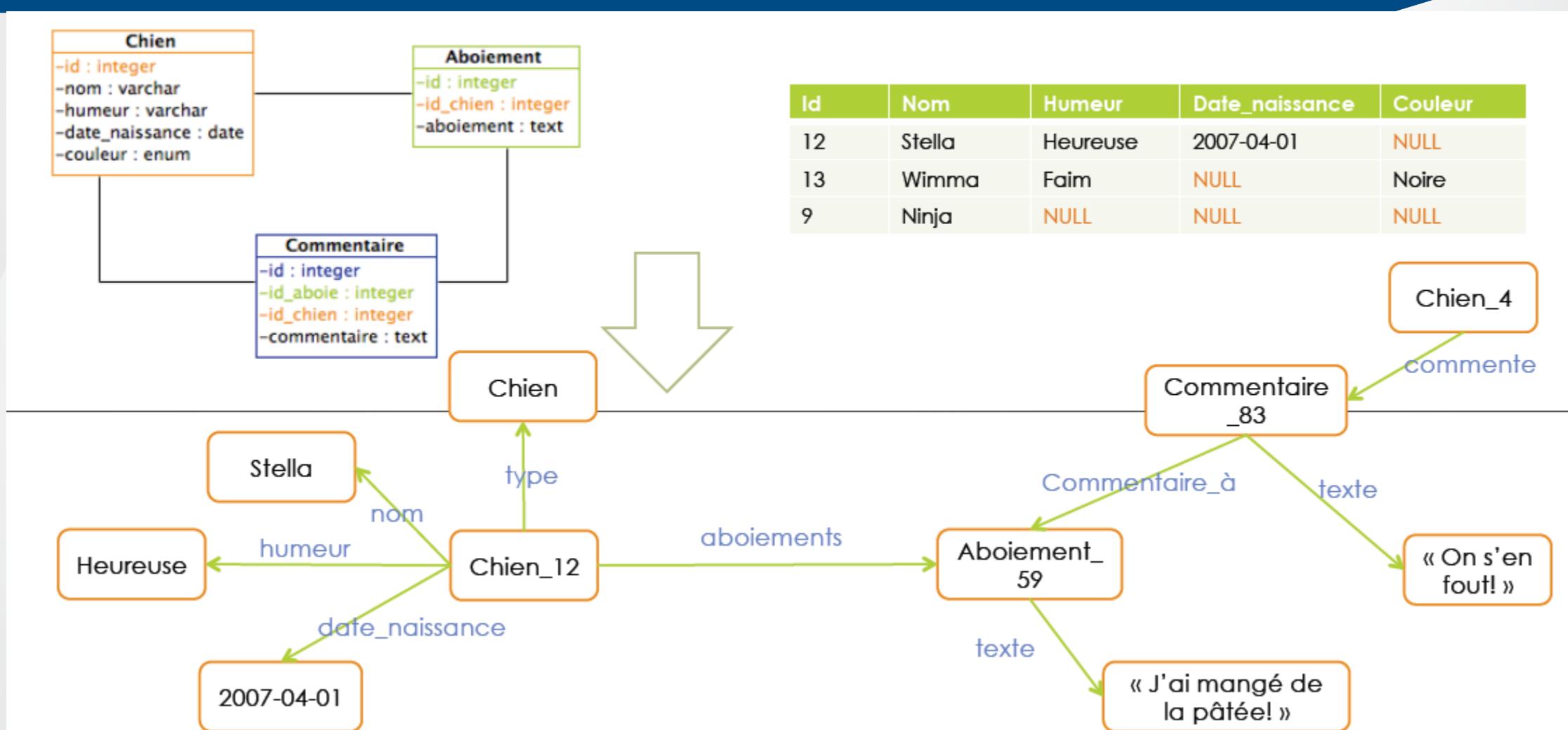
Exp : chien\_12/chien :  
nom=>Stella :hummeur=>heureuse :Date\_naissance=>2007-04-01

# Type de base de données NOSQL : Orientée Graphe

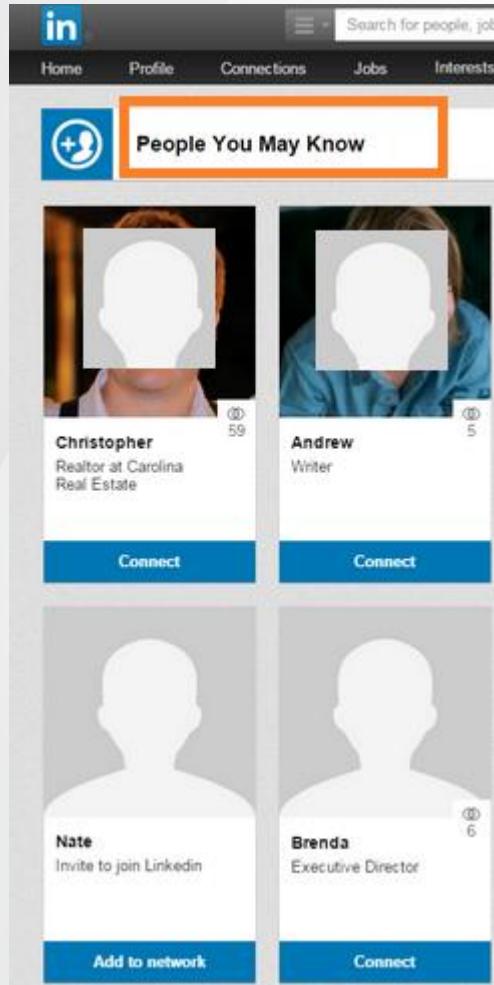
Orientée graphe :

- Basée sur les théories des graphes
- S'appuie sur les notions de nœuds, de relations et des propriétés qui leur sont rattachées
- Conçues pour les données dont les relations sont représentées comme graphes, et ayant des éléments interconnectés, avec un nombre indéterminé de relations entre elles
- Adapté aux traitements des données des réseaux sociaux
- Exemples : LinkedIn

# Type de base de données NOSQL : Orientée Graphe

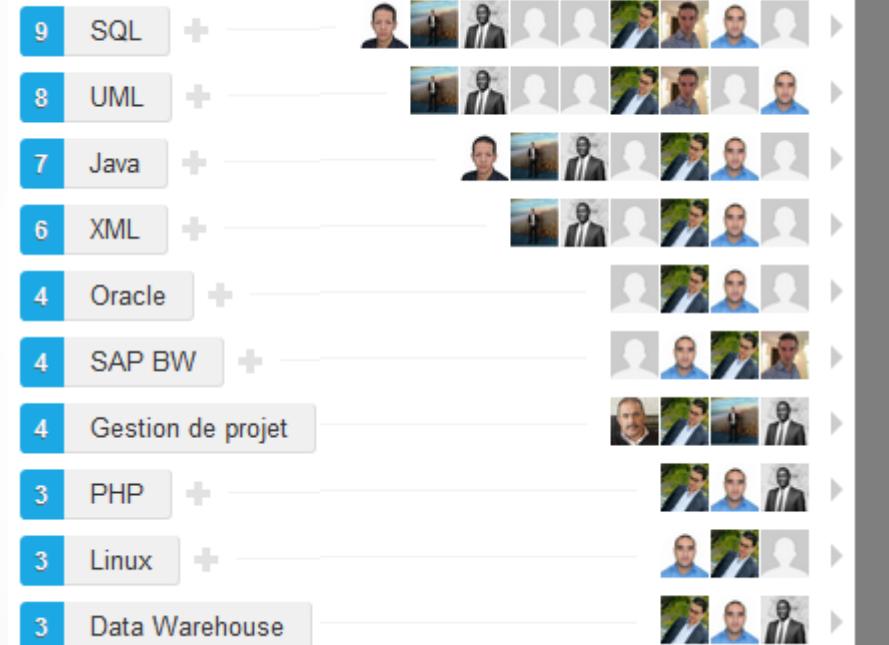


# Type de base de données NOSQL : Orientée Graphe



A screenshot of the LinkedIn homepage. At the top, there is a search bar with the placeholder "Search for people, job". Below the search bar, the navigation menu includes "Home", "Profile", "Connections", "Jobs", and "Interests". A blue button labeled "People You May Know" is highlighted with an orange border. Below this button, there are two profile cards: "Christopher" (Realtor at Carolina Real Estate) and "Andrew" (Writer). At the bottom of the page, there are two more profile cards: "Nate" (Invite to join LinkedIn) and "Brenda" (Executive Director). Each profile card has a "Connect" button.

Top des compétences



En commun avec

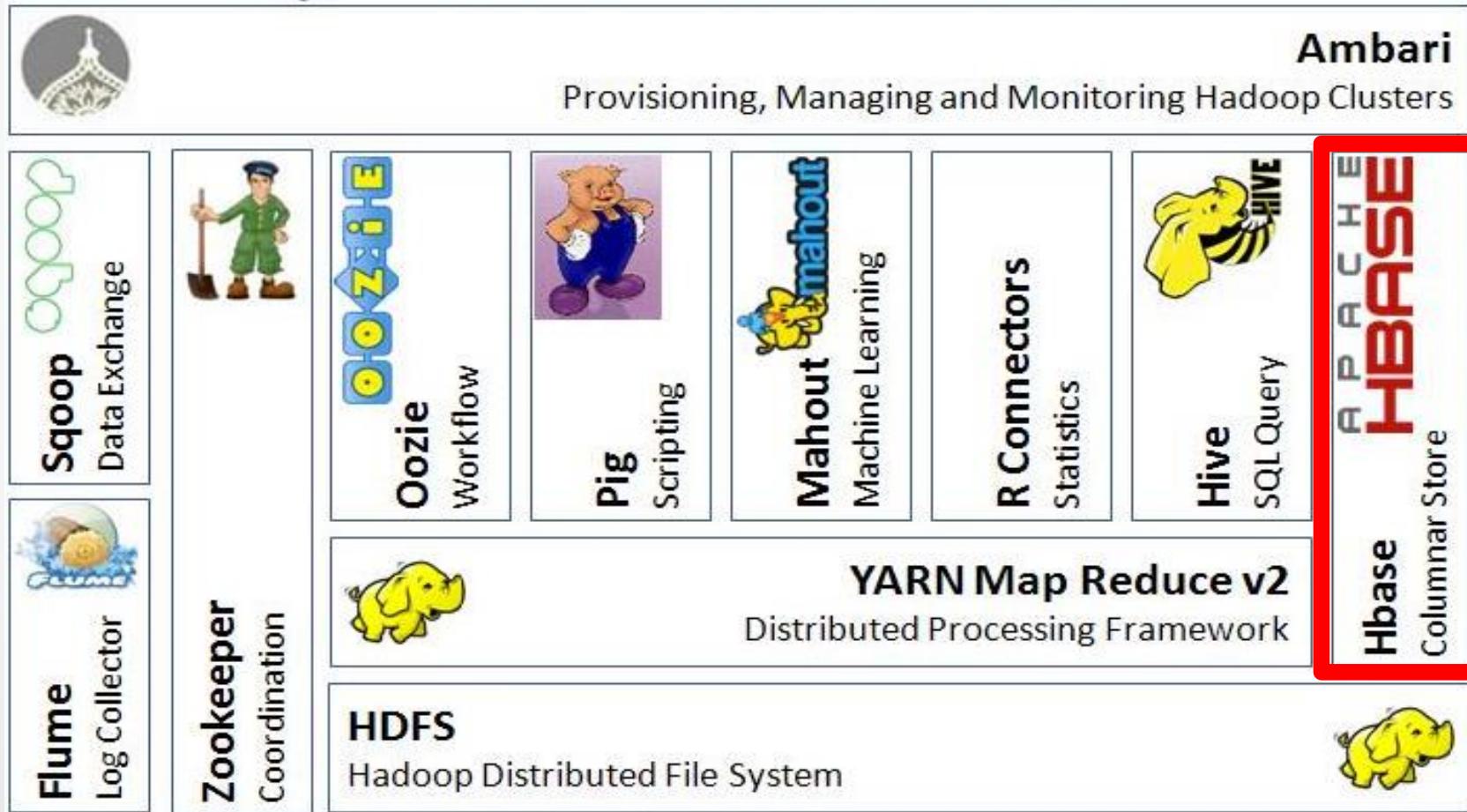




Global Knowledge®

HBase

# HBASE



# HBASE

## Un peu d'histoire...

- 2006 - Google sort son papier sur BigTable
- 2007 - Version prototype d'HBase / contribution à Hadoop
- 2007 - Première version d'HBase
- 2008 - Hadoop devient un top-level project à la fondation apache et HBase un sous-projet
- 2010 - HBase devient un top-level project à la fondation apache

# HBASE

- Distribuée
- Orientée colonne
- Multidimensionnelle
- Haute Disponibilité
- Haute Performance
- Système de stockage

# HBASE

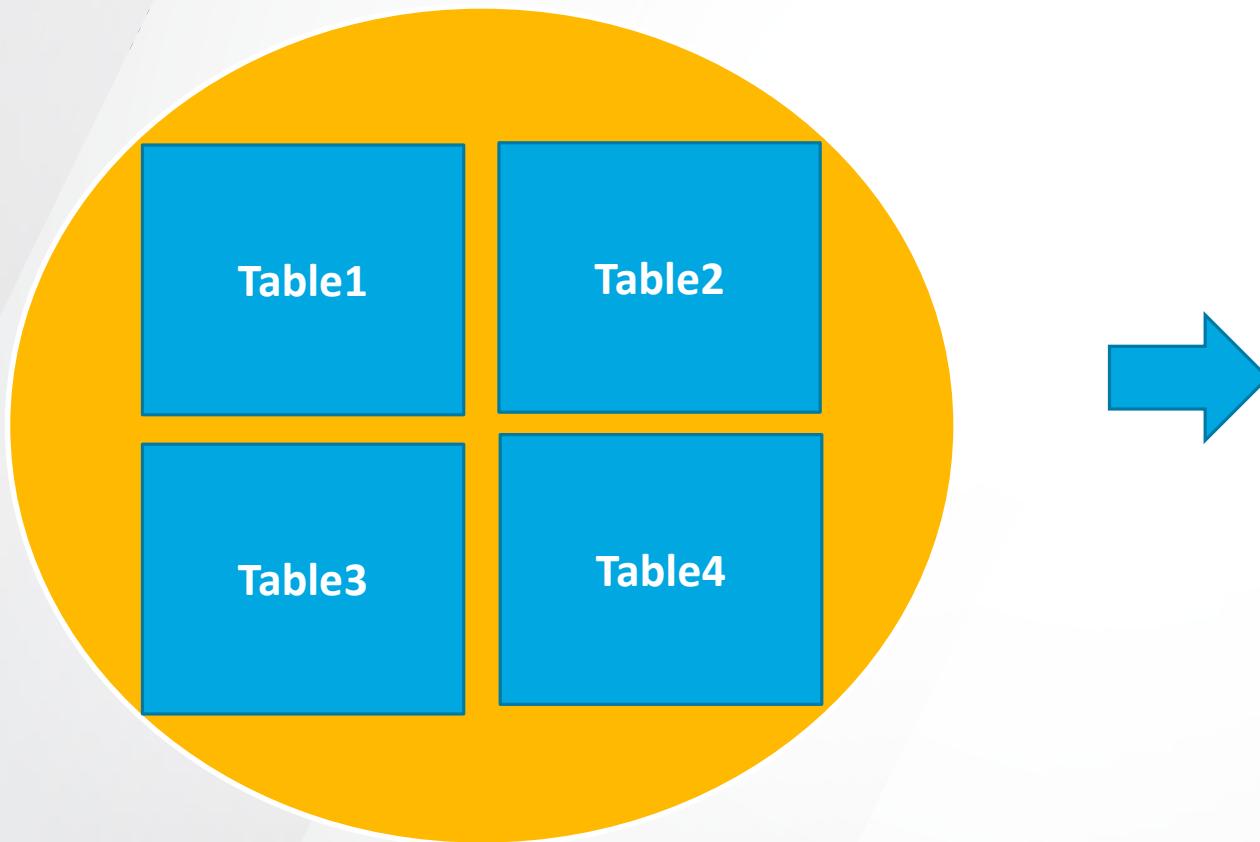
- HBASE est un système de gestion des données pour un environnement distribué qui doit être couplé au gestionnaire de fichiers Hadoop
- HBASE gère la partie **logique**, tandis qu'Hadoop gère la partie **physique**
- C'est un modèle de stockage BigData **orienté colonne**, basé sur le principe de **BigTable de Google**

# Architecture : aspect logique

- Les données sont gérées au sein de grande tables (**appelées Htable**) composées de lignes (**nommées row**) et de famille de colonnes (**Family**).
- Les familles de colonnes sont sous-divisées en colonnes (**qualifier**).
- Les lignes sont des identifiants dont la valeur est unique, soit l'équivalent de la clé primaire dans le mode relationnel.
- Comparaison avec un SGBDR :
  - **Htable => ensemble de toutes les tables de la base**
  - **family => une table**
  - **qualifier => un attribut**
  - **row => clé primaire**

**De fait comme toutes les données sont logiquement placées dans la même table, les jointures sont inutiles.**

# Architecture : aspect logique



**HTABLE**

**QF :** Tab1, Tab2, Tab3, Tab4

**Q:** col..tbl1, col..tab2..

# Architecture : aspect logique

KeyValue = row + family + qualifier + timestamp + value

HTable				
	Family-1		Family-2	
	Qualifier-1	Qualifier-2	Qualifier-1	Qualifier-3
Row-1	■			■
Row-2	■	■■	■	■
Row-3			■■■	
Row-4	■	■■		■
Row-5	■		■	■
Row-6	■	■	■	■■■

KeyValue = coordonées  
(Row + Family + Qualifier + Timestamp + Value)

Value

Timestamp  
= version

# Architecture : aspect logique

- Hbase est basé sur la même architecture physique qu'Hadoop puisque
- ce dernier gère le stockage physique.
- On retrouve donc une configuration distribuée en cluster de plusieurs noeuds qui peuvent être hétérogènes au niveau hardware.
- Le principe est donc le même que celui d'Hadoop. Seuls les noms diffèrent :

## **Architecture**

Serveur - maître  
Serveur - esclave  
Fichier de stockage

## **Hadoop**

Namenode  
Datanode  
Datafile

## **Hbase**

Regionserver  
Region  
Storefile

# Architecture : aspect physique

- Avant d'être stockées à travers le cluster, les données à stocker transitent par un Framework fourni avec Hbase :
  - Le Zookeeper. Il coordonne la destination d'archivage des données dans un système distribué.
- Ces données sont ensuite réparties sur les différents serveurs (region) du cluster, où elles sont triées et compactées.
- Pour une meilleure gestion de l'espace mémoire, les données sont toujours sauvegardées sous forme de bytes. Elles sont stockées dans les regionservers sous forme de blocs de données ( les storefiles) d'une taille configurée par défaut de 64 ( configuration faite via Hadoop)

# Architecture : aspect physique

- **La répartition des données sur les regionserveurs se fait en 2 étapes :**
  - Les lignes d'une Htable sont scindées en paquets réguliers par le regionserveur.
  - Chaque paquet est dirigé chacun vers un serveur-esclave où il est de nouveau divisé par le memestore selon les colonnes de la Htable.
- **Les données issues de la même famille de colonne sont alors stockées dans un même storefile ( ou datafile pour hadoop )**

# Comment fonctionne HBase ?

Deux types de nœuds :

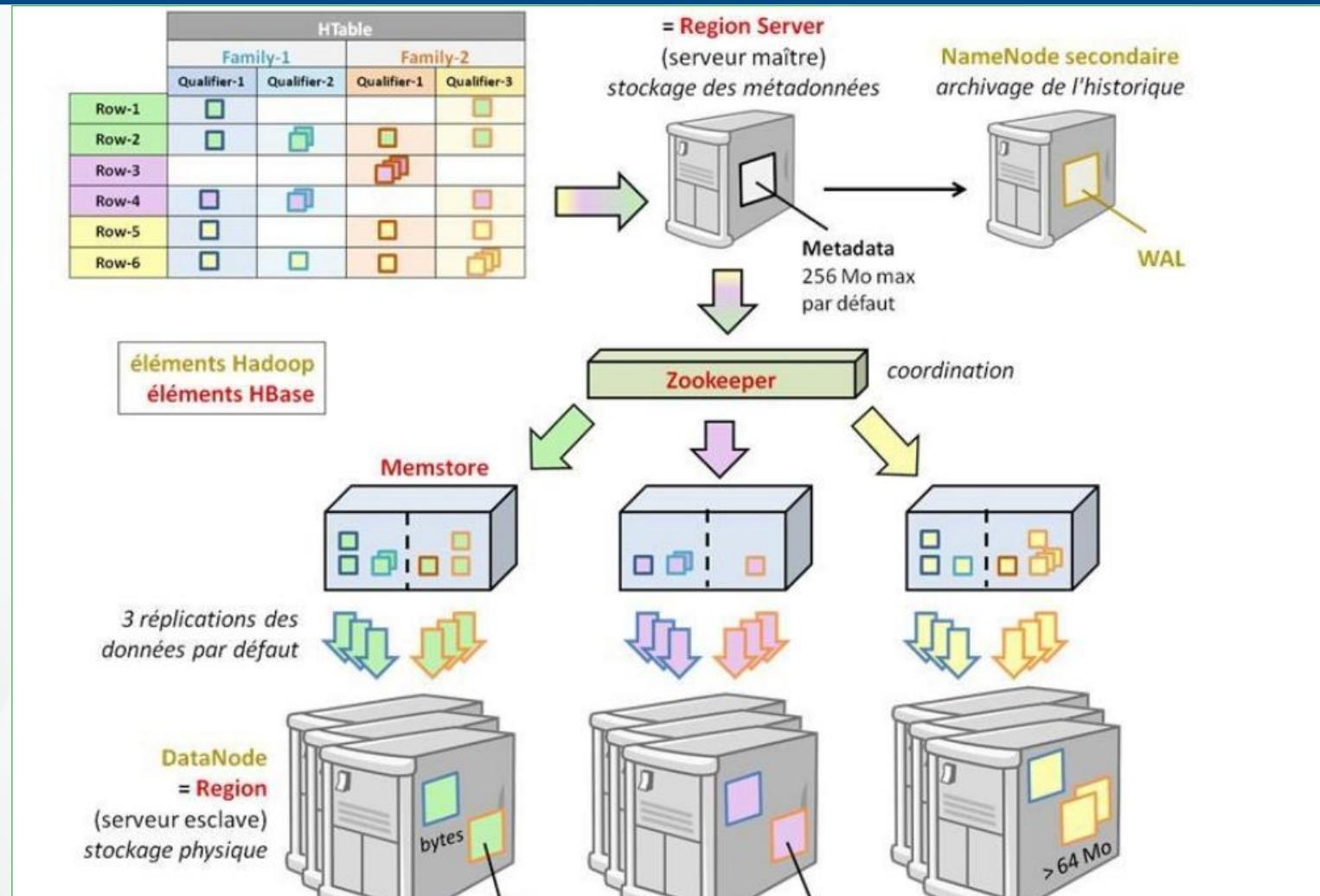
➤ **Master :**

- Master (un à la fois)
- Gère les opérations du cluster
- Affectation, répartition de la charge, fractionnement
- Haute disponibilité avec Zookeeper

➤ **RegionServer**

- Héberge les tables, exécute les lectures, écritures
- Les Clients dialoguent directement avec eux pour les lectures / écritures

# Architecture HBase



# Architecture Hbase : élément critique

## ➤ Les éléments influent sur la vitesse de la lecture des données :

## Choix des colonnes :

- Requête utilisant une seule colonne de famille.
  - Taille du nom de colonne de famille le plus petit possible

## Exemple :

Familles de colonnes = ( température d'un compostant )  $\Rightarrow t$

= ( état activé ou nom du composant ) => e

# Commande HBASE 1/6

- Entrer dans le shell de Hbase :

hbase shell

- Lister le contenu des tables Hbase :

list

- Décrire une table Hbase :

describe 'Table'

- Lire toutes les données d'une table Hbase :

scan 'Table'

# Commande HBASE 2/6

- **Créer une table T avec ses familles de colonnes F1 et F2 :**  
`create 'T', 'F1','F2'`

- **Déverrouiller une table :**  
`disable 'T'`

- **Déverrouiller toutes les tables :**  
`disable_all ' *.*'`

- **Ajouter une famille de colonne à une table**  
`alter 'T', NAME => 'F3'`

# Commande HBASE 3/6

- **Reverrouiller une table :**

enable 'T'

- **Ajouter des valeurs dans une table :**

put 'T', 'R1', 'F1:C1','V1'

- **Lire une ligne avec toutes ses valeurs:**

disable 'T'

- **Savoir si la table existe :**

exist 'T'

# Commande HBASE 4/6

- **Lire des familles de colonnes :**

```
scan 'T', {COLUMNS => ['F1','F2']}
```

- **Lire des colonnes :**

```
scan 'T', {COLUMNS => ['F1:C3','F2:C1']}
```

- **Lire des colonnes à partir d'une ligne précise :**

```
scan 'T', {COLUMNS => ['F1:C3','F2:C1'],STARTROW=> 'R2'}
```

- **Lire des colonnes jusqu'à une ligne précise :**

```
scan 'T', {COLUMNS => ['F1:C3','F2:C1'],STOPROW=> 'R6'}
```

# Commande HBASE 5/6

- **Lire une ligne avec toutes les valeurs :**

get 'T', 'R1'

- **Lire les valeurs d'une ligne et d'une famille de colonnes :**

get 'T', 'R1','F1'

- **Lire les valeurs d'une ligne et d'une colonne :**

get 'T', 'R1','F1:C3'

- **Vider une table :**

truncate 'T'

# Commande HBASE 6/6

- **Supprimer une table :**

drop 'T'

- **Supprimer une famille de colonnes :**

alter 'T', delete => 'F1'

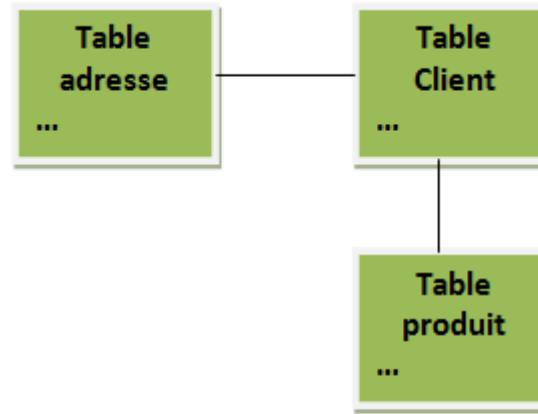
- **Supprimer une ligne entière :**

deleteall 'T','R1'

- **Supprimer une valeur via ses coordonnées :**

delete 'T','R1','F1:C2'

# Hbase : Exemple



**ID\_Client : Clef , Column Family1 : Qualifier1 = Valeur, Qualifier2 = Valeur**  
**... Column Family2 : Qualifier1 = Valeur, Qualifier2 = Valeur ...**

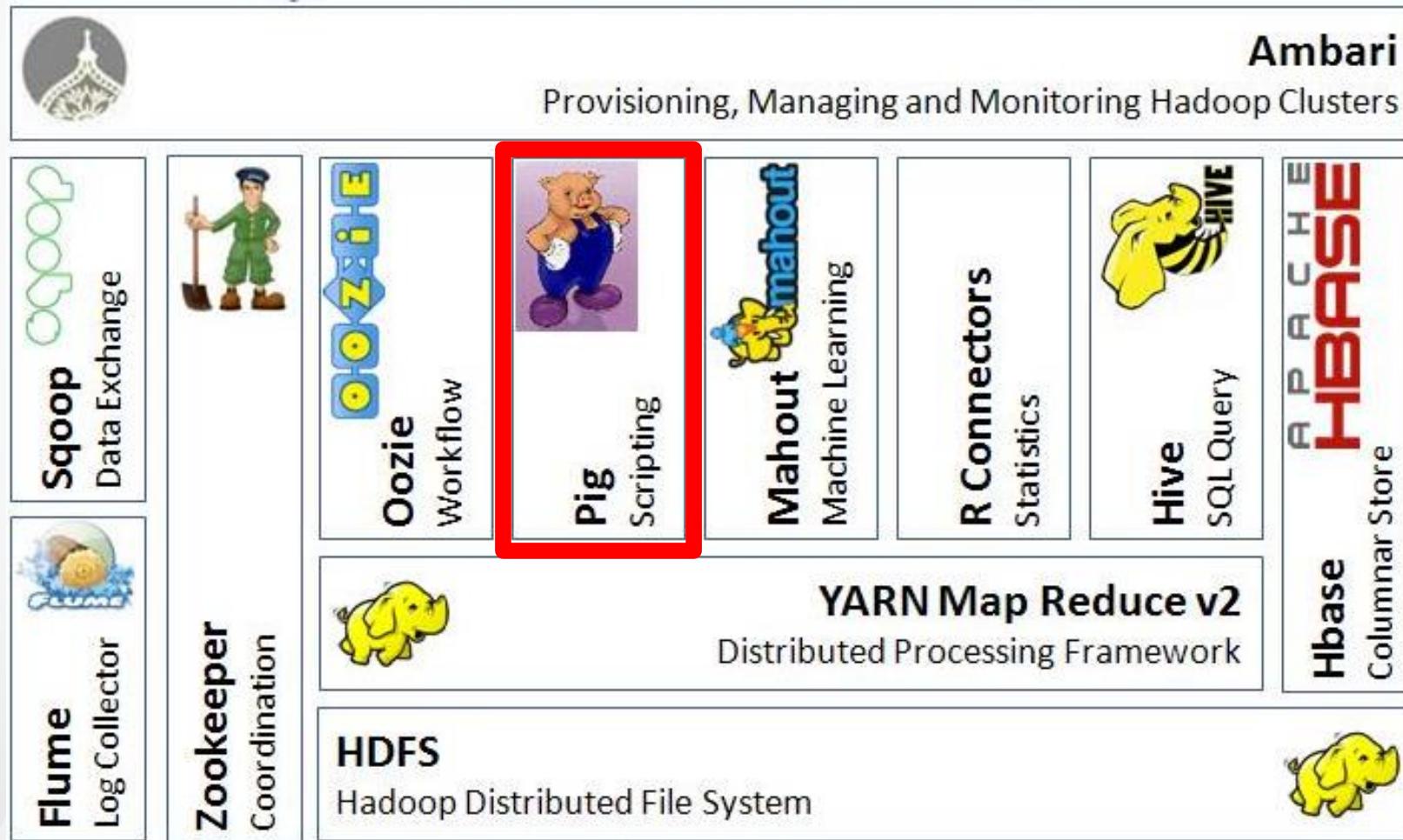
**ID\_Client : Clé, adresse : ..... , client, ..... , produit**



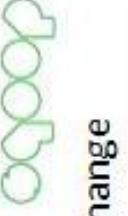
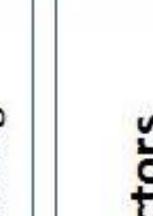
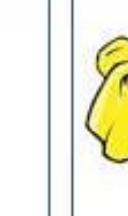
Global Knowledge®

PIG

# PIG



**Ambari**  
Provisioning, Managing and Monitoring Hadoop Clusters

 <b>Flume</b> Log Collector	 <b>Sqoop</b> Data Exchange	 <b>Zookeeper</b> Coordination	 <b>Oozie</b> Workflow	 <b>Pig</b> Scripting	 <b>Mahout</b> Machine Learning	 <b>R Connectors</b> Statistics	 <b>Hive</b> SQL Query	 <b>Hbase</b> Columnar Store
<b>YARN Map Reduce v2</b> Distributed Processing Framework								
<b>HDFS</b> Hadoop Distributed File System								

# Qu'est-ce que Pig ?

## Créé chez Yahoo!

- Une plate-forme **très simple pour traiter les Big Data**
- **PigLatin** : langage dont le traitement est en flux, simple, très efficace
- **Pig Engine** : parse, optimise et exécute automatiquement les scripts PigLatin comme une série de jobs MapReduce au sein d'un cluster Hadoop

# Qu'apporte Pig ?

## Créé chez Yahoo!

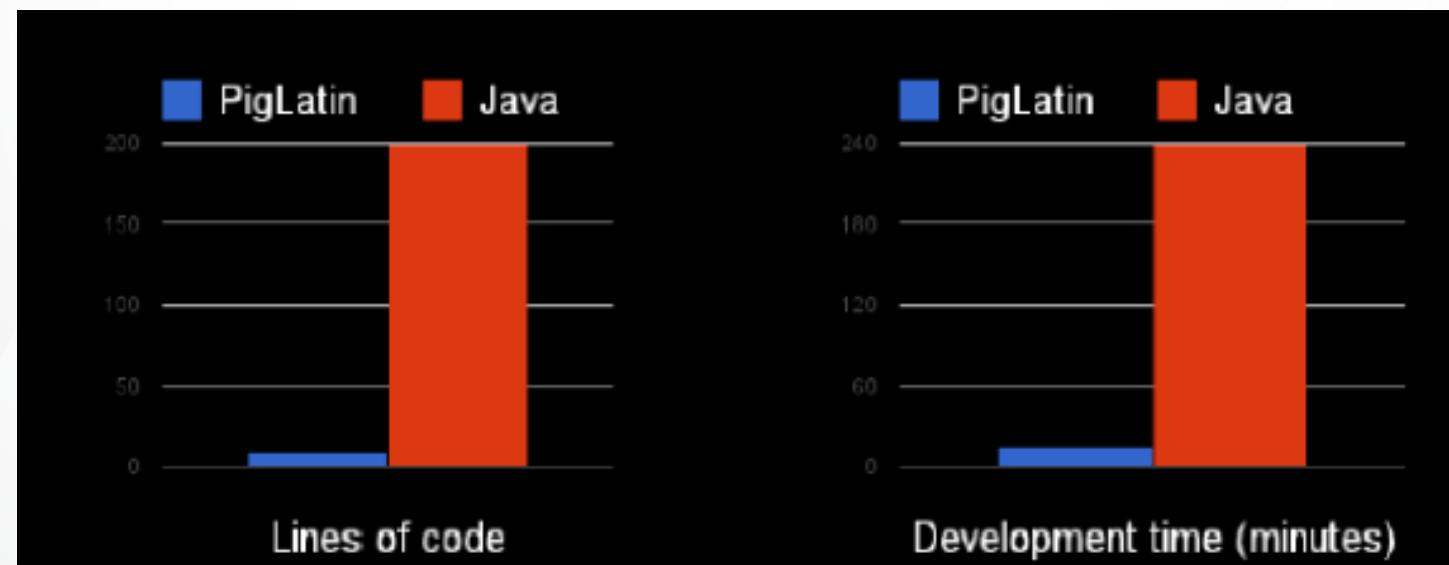
- PigLatin est :
  - Un langage de haut niveau,
  - Facile à comprendre,
  - Orienté traitement par flux (data flow)
- Il fournit les opérations standards pour la manipulation de données :
  - Filtre, jointure, tri , des types primitifs, des types complexes (tuples, bags, maps)
- Bien plus simple à comprendre pour un analyste que du MapReduce
  - Il ouvre Hadoop au non-programmeur-java

# WordCount en PigLatin ?

```
Lines = LOAD '/data/texts/*.txt' AS (line:chararray);
-- TOKENIZE splits the line into a bag of words
-- FLATTEN produces a separate record for each item
--   from a bag
Words = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
-- Group records together by each word.
Groups = GROUP Words BY word;
-- Count words.
Counts = FOREACH Groups GENERATE group, COUNT(Words) AS wordCount;
-- Store the results.
STORE Counts INTO 'counts';
```

# Si vous n'êtes pas encore convaincu ?

- Augmente dramatiquement la productivité
- 10 lignes en Pig = 200 lignes en Java
- 15 minutes en Pig = 4 heures en Java



# Top 5 des pages les plus vues en PigLatin

```
-- Load users and pages data files
Users = LOAD '/data/texts/users.txt' AS (user: chararray, age: int);
Pages = LOAD '/data/texts/pages.txt' AS (user: chararray, url: chararray);
-- Remain records with users with age between 18 and 25
Fltrd = FILTER Users BY age >= 18 and age <= 25;
-- Join data sets by a user key
Jnd = JOIN Fltrd BY user, Pages BY user;
-- Group records together by each url
Grpd = GROUP Jnd BY url;
-- Calculate click count for each group
Smmd = FOREACH Grpd GENERATE group, COUNT(Jnd) AS clicks;
-- Sort records by a numer of click
Srted = ORDER Smmd BY clicks DESC;
-- Get top 5 pages
Lmt = LIMIT Srted 5;
-- Store output records in a given directory
STORE Lmt INTO '/jobs/output/top5Pages';
```

# Les autres bénéfices de Pig

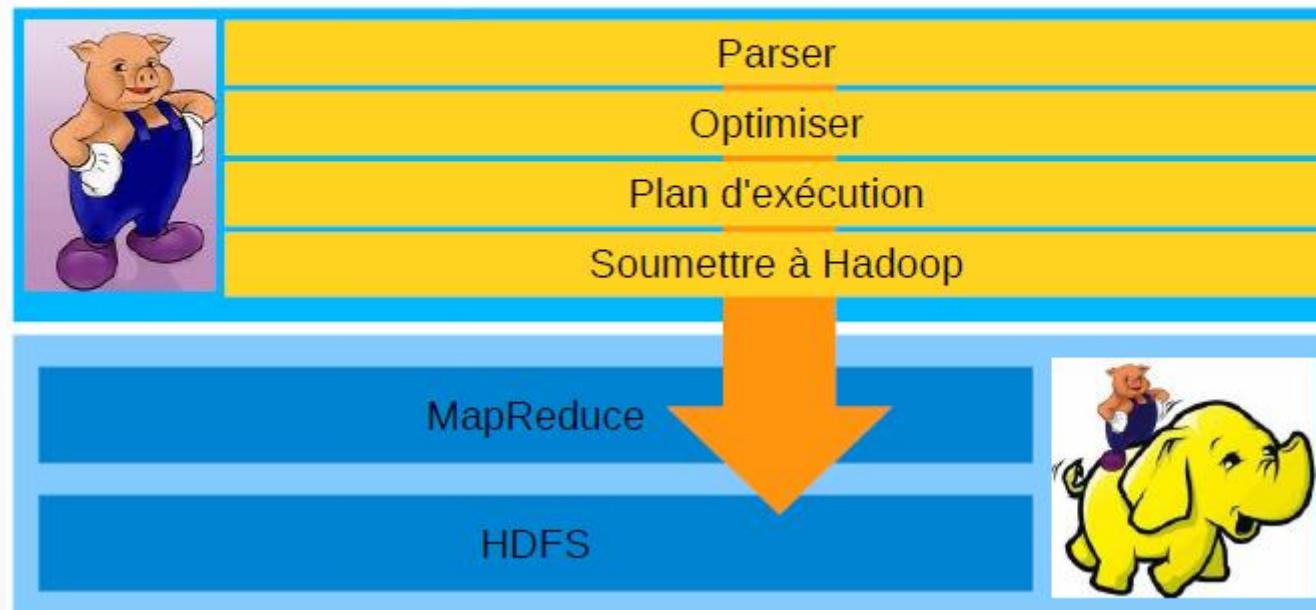
---

Gère tous les détails d'un job de la soumission jusqu'à son exécution et ce même sur des flux de données très complexes.

- Écrire des jobs qui n'ont pas d'adhérence à l'API Java d'Hadoop
- Facile à étendre avec les **UDF**
- Possibilité d'embarqué :
  - Python
  - JavaScript
- Intégré à HBase

# Comment fonctionne Pig ?

```
Lines = LOAD '/data/texts/*.txt' AS (line:chararray);
Words = FOREACH Lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE group, COUNT(Words);
STORE Counts INTO 'counts';
```



# A vos éditeurs !

## Eclipse :

- PigEditor
- Pig-pen
- Pig-Eclipse

## Plugin pour :

- VIM
- Emacs,
- Textmate

# Comment utiliser Pig ?

## ➤ Mode local

- Ni Hadoop, Ni HDFS requis
- Système de fichiers local
- Faciles à utiliser pour « prototyper », développer,

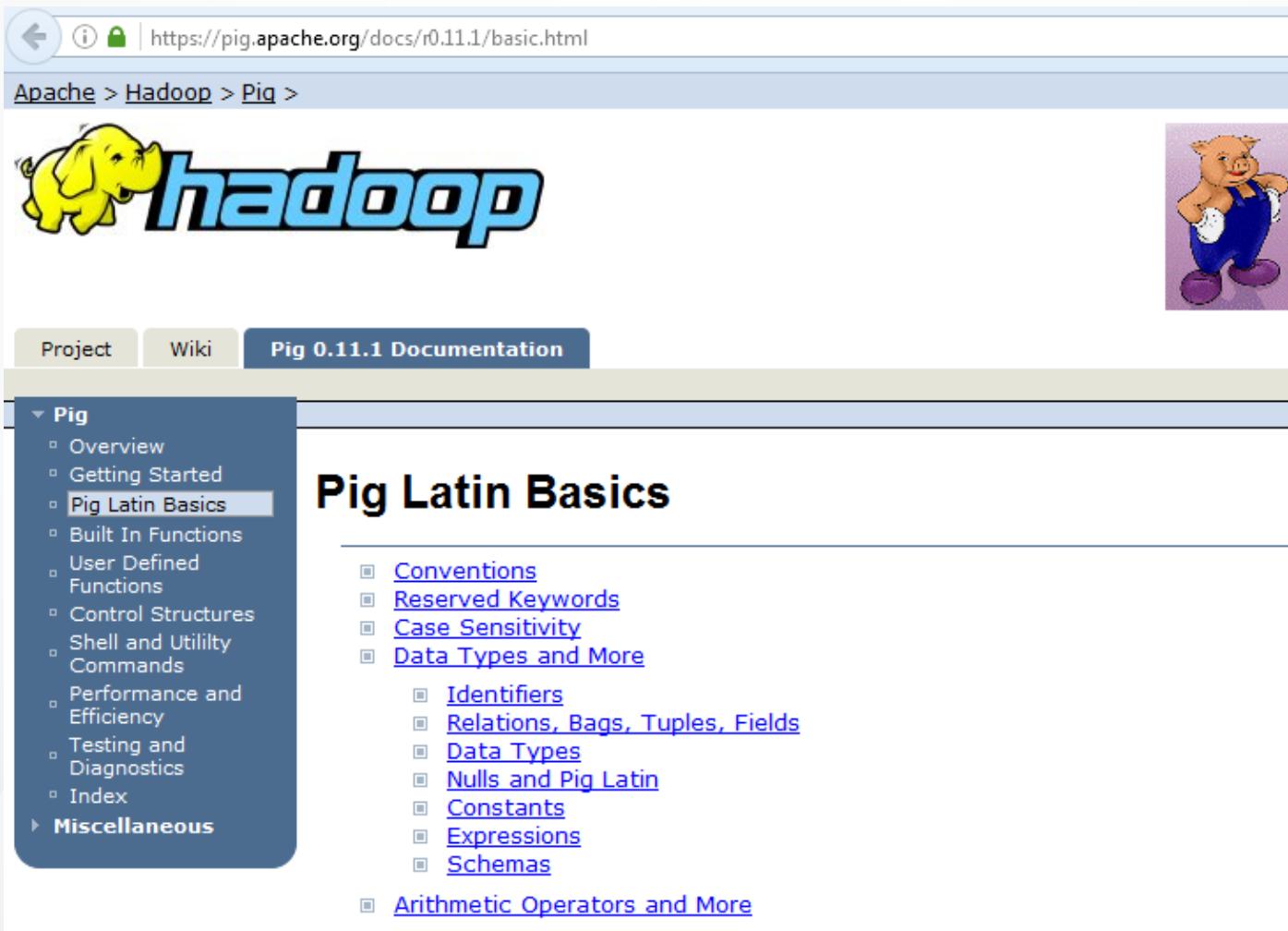
## ➤ débugger

- Mode Cluster
- Sait exécuter le même job qu'en local

# Exécuter un script Pig

- Exécuter un script pig directement – mode batch
  - *\$ pig -p input=someInput script.pig*
- *script.pig*
  - *Lines = LOAD '\$input' AS (...);*
- Grunt, le shell pour Pig – mode interactif
  - *grunt> Lines = LOAD '/data/books/' AS (line: chararray);*
  - *grunt> Unique = DISTINCT Lines;*
  - *runt> DUMP Unique;*

# Documentation



The screenshot shows a web browser displaying the Apache Pig 0.11.1 Documentation for Pig Latin Basics. The URL in the address bar is <https://pig.apache.org/docs/r0.11.1/basic.html>. The page title is "Apache > Hadoop > Pig >". The main content features the Hadoop logo (a yellow elephant) and a cartoon pig character. The navigation bar includes "Project", "Wiki", and "Pig 0.11.1 Documentation". A sidebar on the left lists "Pig" sub-sections: Overview, Getting Started, Pig Latin Basics (which is selected and highlighted in blue), Built In Functions, User Defined Functions, Control Structures, Shell and Utility Commands, Performance and Efficiency, Testing and Diagnostics, Index, and Miscellaneous. The main article title is "Pig Latin Basics". Below the title is a list of sub-topics: Conventions, Reserved Keywords, Case Sensitivity, Data Types and More, Identifiers, Relations, Bags, Tuples, Fields, Data Types, Nulls and Pig Latin, Constants, Expressions, Schemas, and Arithmetic Operators and More.

https://pig.apache.org/docs/r0.11.1/basic.html

Apache > Hadoop > Pig >

Project Wiki Pig 0.11.1 Documentation

▼ Pig

- Overview
- Getting Started
- **Pig Latin Basics**
- Built In Functions
- User Defined Functions
- Control Structures
- Shell and Utility Commands
- Performance and Efficiency
- Testing and Diagnostics
- Index
- ▶ Miscellaneous

## Pig Latin Basics

---

- [Conventions](#)
- [Reserved Keywords](#)
- [Case Sensitivity](#)
- [Data Types and More](#)
  - [Identifiers](#)
  - [Relations, Bags, Tuples, Fields](#)
  - [Data Types](#)
  - [Nulls and Pig Latin](#)
  - [Constants](#)
  - [Expressions](#)
  - [Schemas](#)
- [Arithmetic Operators and More](#)



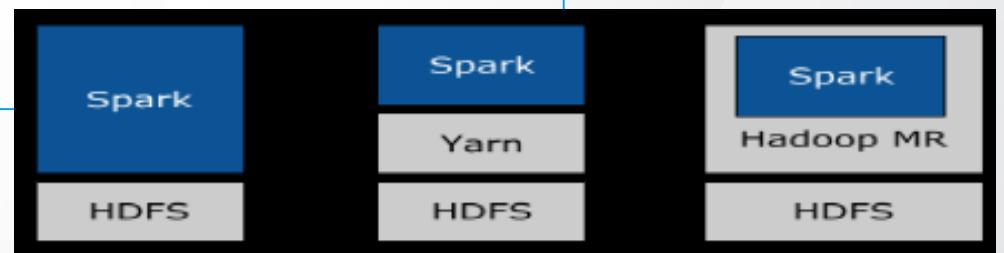
Global Knowledge®



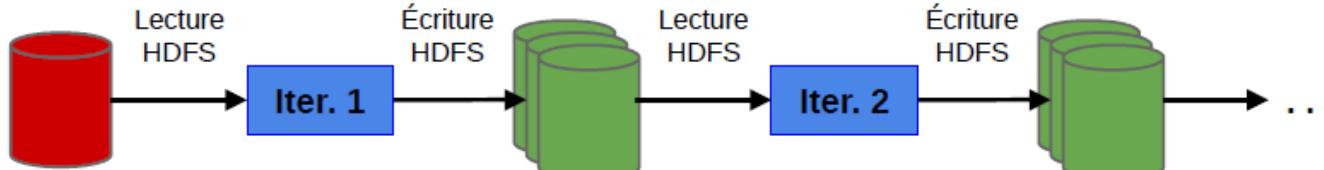
Le projet SPARK

# Hadoop : Qu'est-ce que Spark ?

- Ce n'est pas une version modifiée de Hadoop.
- C'est un moteur Map-Reduce plus évolué, plus rapide.
  - Utilisation de la mémoire pour optimiser les traitements.
  - Des API'S pour faciliter et optimiser les étapes d'analyses.
  - Permettre le traitement temps réel des données.
- Compatible avec le système de Stockage de Hadoop (HDFS, Hive, SequenceFiles...)



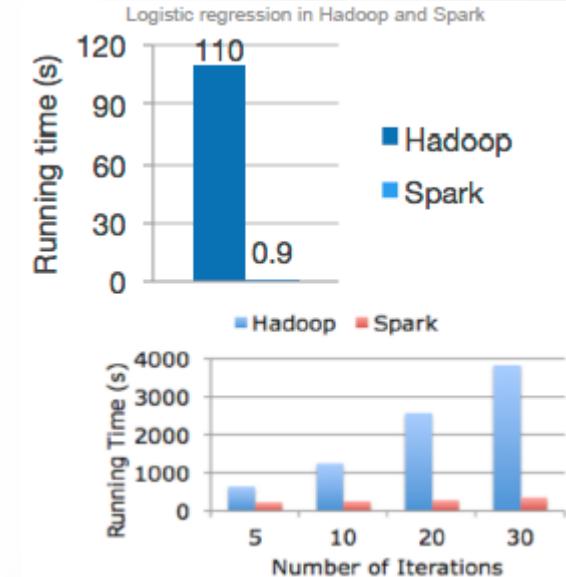
# Hadoop : Spark, plus rapide



Traitement (très) long à cause des nombreux accès à l'HDFS



Une seule lecture des données puis traitement en mémoire



## RDD's (Resilient DataSet)

- Une collection d'objets distribués, mise en cache en mémoire au travers du cluster.
- Une API' pour manipuler ces objets "Operators"

# Hadoop : Spark, facile à utiliser

- Une liste d'Operators pour faciliter la manipulation des données au travers des RDD'S :Map, filter, groupBy, sort, join, leftOuterJoin, ritghtOuterJoin, reduce, count, reduceByKey, groupByKey, first, union, cross, sample, cogroup, take, partitionBy, pipe, save.....
- Compatibles avec les langages de programmation fonctionnel Scala, Java, Python...

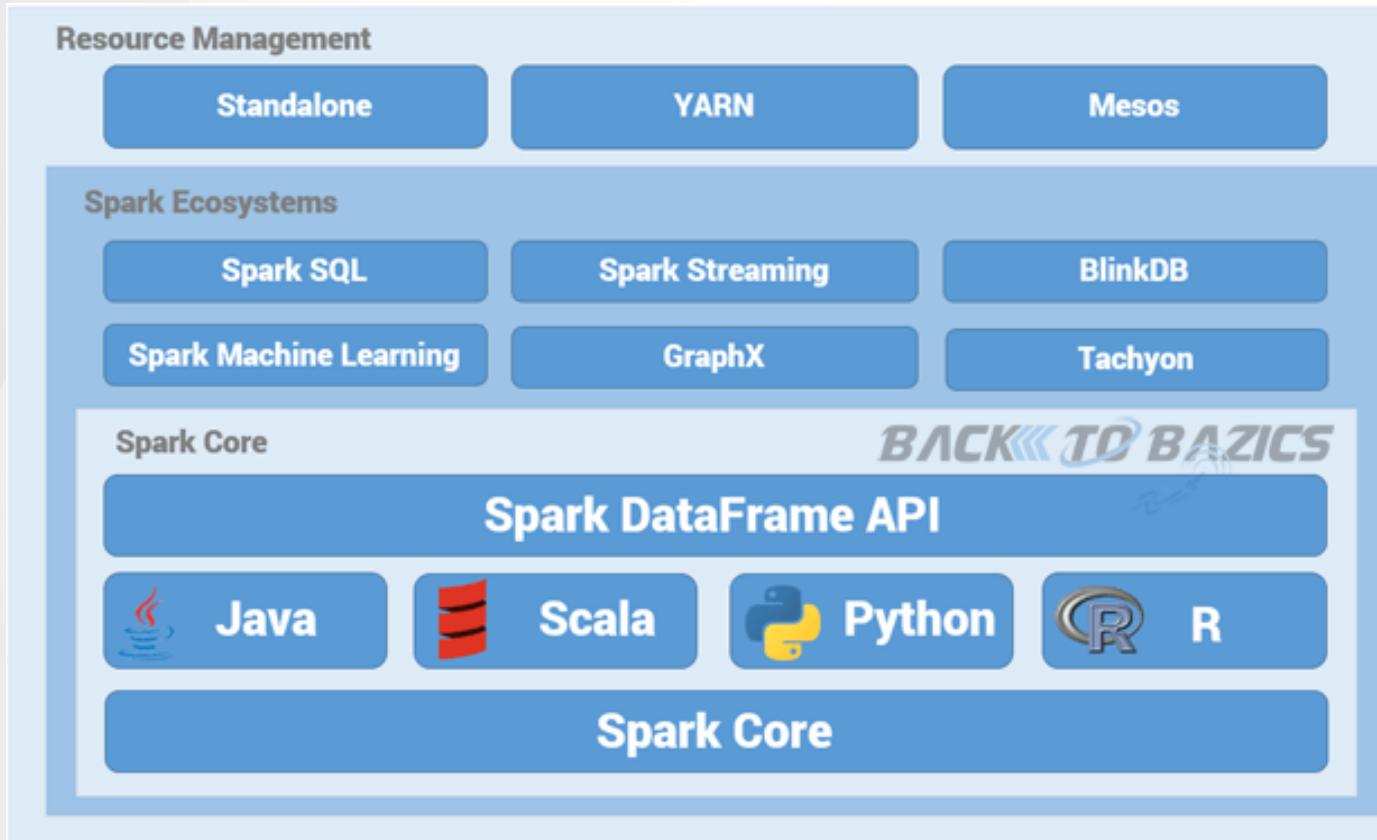
## Java Hadoop

```
public class WordCount {  
    public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {  
        private final static IntWritable one = new IntWritable(1);  
        private Text word = new Text();  
        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {  
            String line = value.toString();  
            StringTokenizer tokenizer = new StringTokenizer(line);  
            while (tokenizer.hasMoreTokens()) {  
                word.set(tokenizer.nextToken());  
                context.write(word, one);  
            } } }  
    public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {  
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {  
            int sum = 0;  
            for (IntWritable val : values) {  
                sum += val.get();  
            }  
            context.write(key, new IntWritable(sum));  
        } }  
    public static void main(String[] args) throws Exception {  
        Configuration conf = new Configuration();  
        Job job = new Job(conf, "wordcount");  
        job.setMapperClass(Map.class);  
        job.setReducerClass(Reduce.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        TextInputFormat.addInputPath(job, new Path(args[0]));  
        TextOutputFormat.setOutputPath(job, new Path(args[1]));  
        job.waitForCompletion(true);  
    } }
```

## Scala Spark

```
file = spark.textFile("hdfs://...")  
  
file.flatMap(line => line.split(" ")).  
      .map(word => (word, 1)).  
      .reduceByKey(_ + _)
```

# Hadoop : Spark, Hadoop : Spark, un Framework analytique



**Spark Streaming :** A opposer aux traitements “batch”, mise à jour d'un état dynamiquement, piloté par des évènements sur les données “event processing”. (Filtre de Spam, analyse de click, reporting temps réel...)

# Shark : Hive pour Spark

- HiveQL  $\approx$  SQL
- Compatible avec Hive data (HDFS, HBase)
- 100x plus rapide que Hive

# Shark + Spark = PL/SQL 2.0

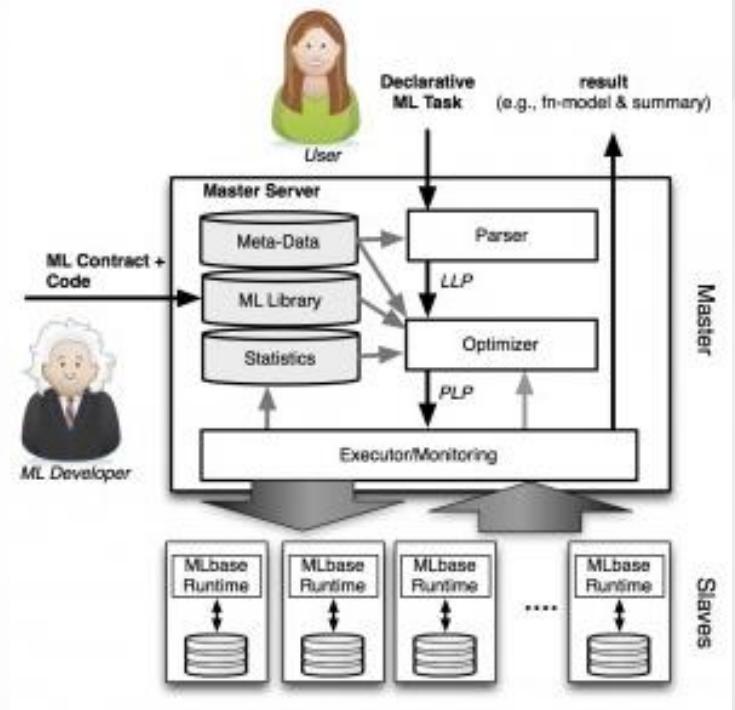
## ➤ Le meilleur de chaque langage :

- SQL pour la sélection
- Scala pour le traitement

```
val youngUsers = sql2rdd("SELECT * FROM users WHERE age < 20")  
print(youngUsers.count)  
val featureMatrix = youngUsers.mapRows(extractFeatures(_))  
kmeans(featureMatrix)
```

# MLBase (Dev)

- Système de machine learning basé sur Spark
  - Sélection automatique du meilleur algorithme
  - Optimisation des paramètres
  - Utilisation interactive





Global Knowledge.®

Questions ?