

Big Data

Plan

- Introduction à Big Data
- Hadoop
 - ✓ Généralité
 - ✓ Architecture HDFS
 - ✓ Algorithme MapReduce
- Utilisation de Hadoop
 - ✓ Manipulation de HDFS
 - ✓ Développement d'une application MapReduce
- L'INTÉGRATION DES DONNÉES DANS LE HDFS
 - ✓ Cinématique de l'écriture de fichier
 - ✓ Cinématique de la lecture de fichier
 - ✓ L'API REST WebHDFS
 - ✓ Apache Flume
 - ✓ Apache Sqoop
- L'ECOSYSTEME HADOOP
 - ✓ Hue
 - ✓ Hive
 - ✓ Pig

Chapitre 1

Introduction à Big Data



BIG DATA: Définition

- La notion de Big Data est un concept s'étant popularisé en 2012 pour traduire le fait que les entreprises sont confrontées à des volumes de données à traiter de plus en plus considérables et présentant un fort enjeux commercial et marketing.
- Ces Grosses Données en deviennent difficiles à travailler avec des outils classiques de gestion de base de données.
- Il s'agit donc d'un ensemble de **technologies, d'architecture, d'outils** et de **procédures** permettant à une organisation de très rapidement capter, traiter et analyser de larges quantités et contenus hétérogènes et changeants, et d'en extraire les **informations pertinentes** à un **coût accessible**.

BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (1/2)

- Gartner (2001) – 3Vs



- IBM (2012) – 4Vs



- 2015 : encore plus de V

VALEUR

BIG DATA: Caractéristiques

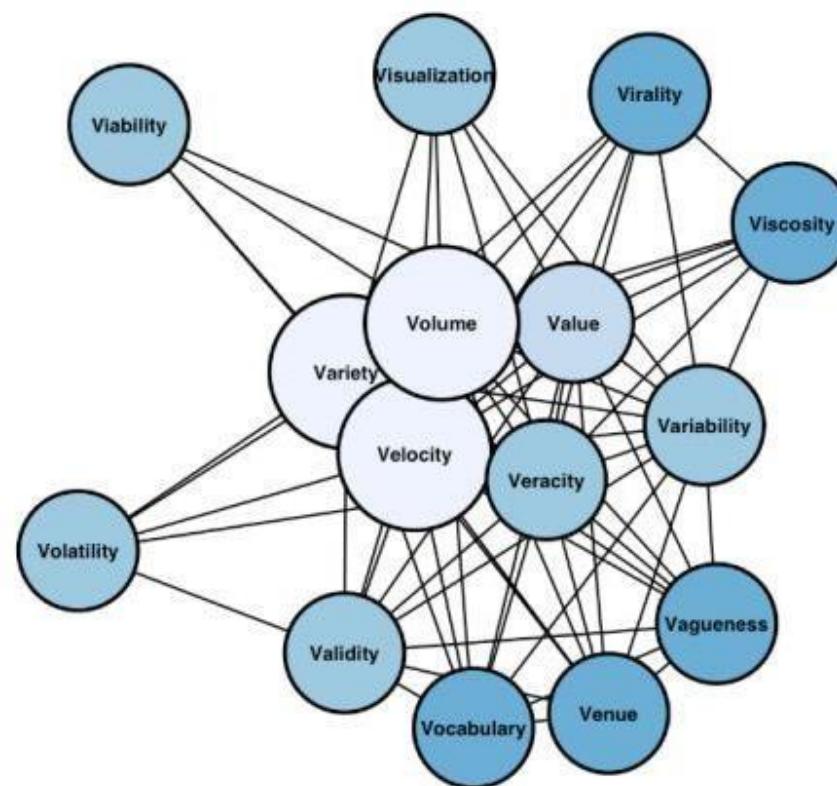
- COUVERTURE DE CINQ DIMENSIONS - 5Vs (2/2)

- 2016 : 7V ?

VARIABILITÉ

VISUALISATION

- 2017 : 10V ?

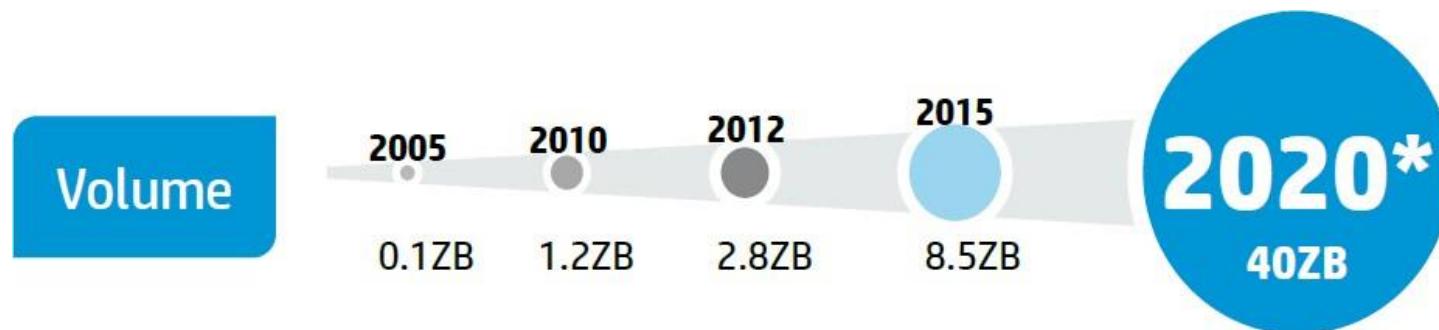


BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (1/7)

- Volume (1/2)

- Croissance sans cesse des données à gérer de tout type, souvent en teraoctets voir en petaoctets.
 - Chaque jour, 2.5 trillions d'octets de données sont générées.
 - **90% des données créées dans le monde l'ont été au cours des 2 dernières années (2014).**
 - Prévision d'une croissance de 800% des quantités de données à traiter d'ici à 5 ans.



BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (2/7)

- Volume (2/2)

- 4,4 zettaoctets de données
= 4,4 trillion de gigaoctets
 - En 2013, il y a autant de données que les étoiles connues dans tout l'univers.
 - 44 zettaoctets de données
= 44 milliards gigaoctets
 - 62 fois le nombre de tous les sables dans toutes les plages de la terre.

octets

1 Mégaoctet = 10^6 octets

1 Gigaoctet = 10^9 octets

1 Téraoctet = 10^{12} octets

1 Pétaoctet = 10^{15} octets

1 Exaoctet = 10^{18} octets

1 Zettaoctet = 10^{21} octets

40

zettaoctets
de données
en 2020

50

milliards
d'objets connectés
à la même date

BIG DATA: Caractéristiques

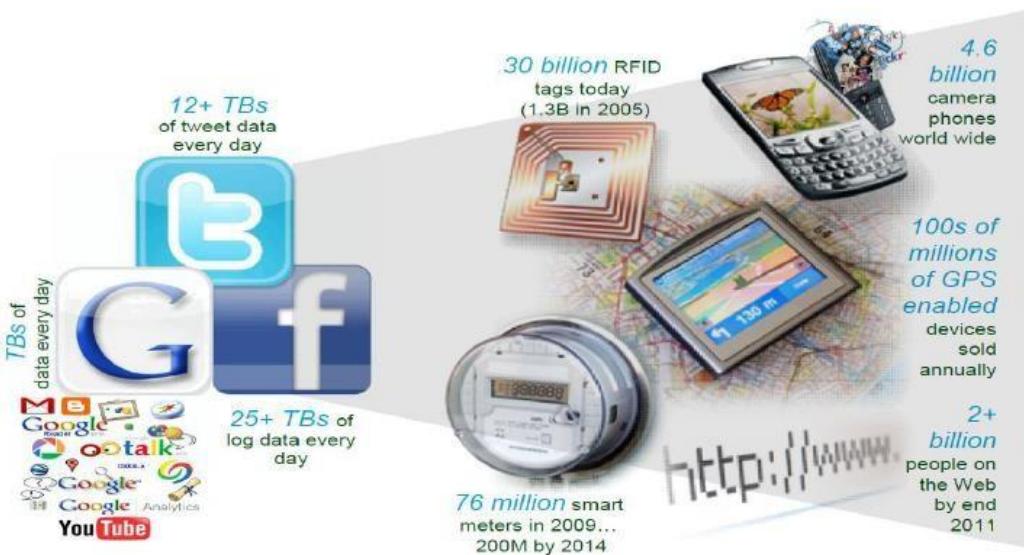
• COUVERTURE DE CINQ DIMENSIONS - 5Vs (3/7)

– Variété

- Traitement des données sous forme structurée (bases de données structurée, feuilles de calcul venant de tableur, ...) et non structurée (textes, sons, images, vidéos, données de capteurs, fichiers journaux, medias sociaux, signaux,...) qui doivent faire l'objet d'une analyse collective.



- Diversité des données



BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (4/7)

- Vitesse (Velocity)

- Utilisation des données en temps réel (pour la détection de fraudes, analyse des données, ...).
 - Fait référence à la vitesse à laquelle de nouvelles données sont générées et la vitesse à laquelle les données sont traitées par le système pour être bien analysées.
 - La technologie nous permet maintenant d'analyser les données pendant qu'elles sont générées, sans jamais mettre en bases de données.

- Streaming Data
 - des centaines par seconde
 - 100 Capteurs
 - dans chaque voiture moderne pour la surveillance



BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (5/7)

- Vérité

- Fait référence à la qualité de la fiabilité et la confiance des données.
 - Données bruités, imprécises, prédictives, ...
 - La génération des données par Spambots est un exemple digne de confiance.
 - L'élection présidentielle de 2012 au Mexique avec de faux comptes Twitter.
 - DES MILLIONS DE DOLLARS \$ PAR AN
 - Ce que la pauvre qualité des données coutent pour l'économie des Etats-Unis.
 - 1 à 3 CHEFS D'ENTREPRISE
 - Ne font pas confiance à l'information qu'ils utilisent.



Veracity

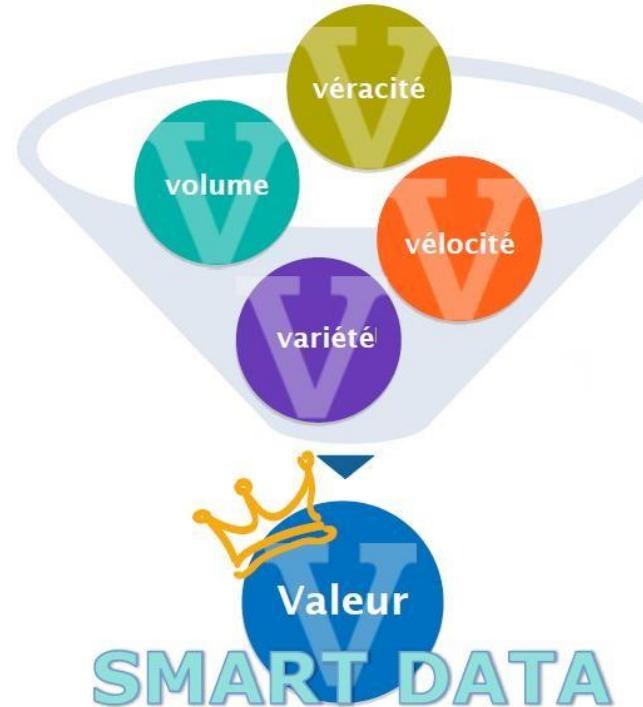
Data in Doubt

BIG DATA: Caractéristiques

- COUVERTURE DE CINQ DIMENSIONS - 5Vs (6/7)

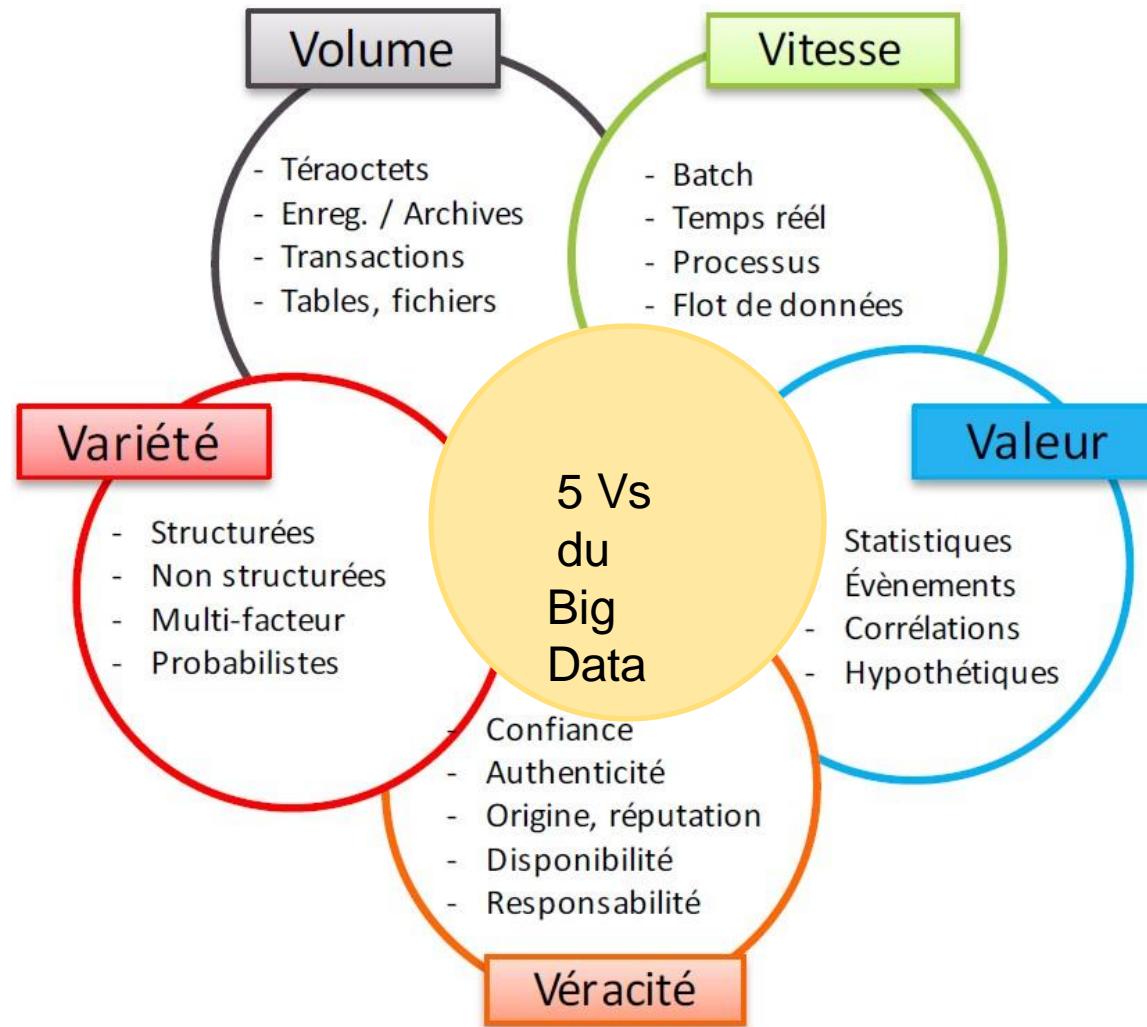
- Valeur

- La démarche Big Data n'a de sens que pour atteindre des objectifs stratégiques de **création de valeur** pour les clients et pour l'entreprise; dans tous les domaines d'activité : commerce, industrie, services ...
 - Le succès d'un projet Big Data n'a d'intérêt aux utilisateurs que s'il **apporte de la valeur ajoutée** et de nouvelles connaissances.



BIG DATA: Caractéristiques

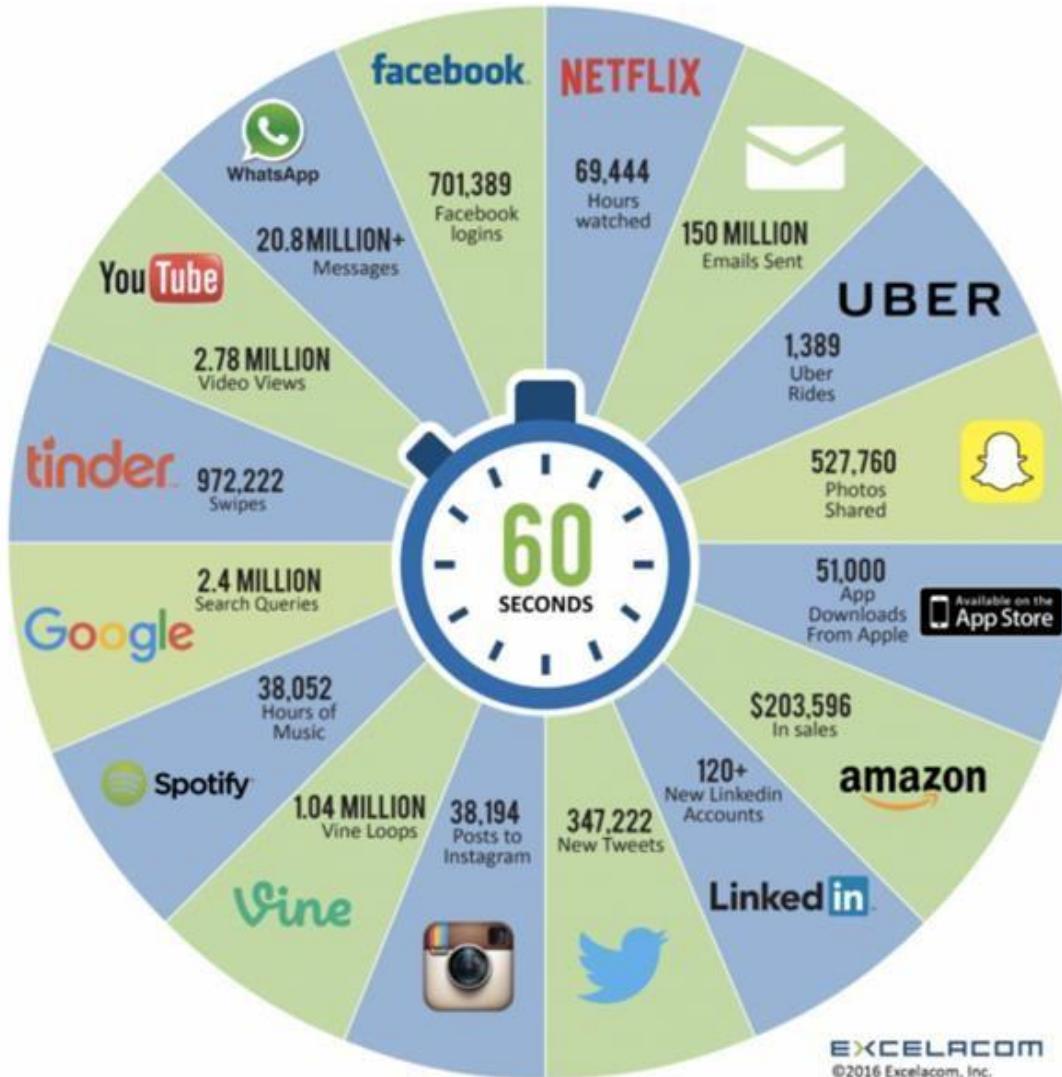
• COUVERTURE DE CINQ DIMENSIONS - 5Vs (7/7)



BIG DATA: Généralités

• DIVERSITE ET VOLUME DES SOURCES DE DONNEES (1/4)

Une minute sur internet en 2016 ?



BIG DATA: Généralités

• DIVERSITE ET VOLUME DES SOURCES DE DONNEES (2/4)



BIG DATA: Généralités

• DIVERSITE ET VOLUME DES SOURCES DE DONNEES (3/4)

– Outils numériques plus performants et plus connectés:
ordinateurs et smartphones



– Open Data

- Réseaux sociaux: facebook
- Données d'administrations publiques

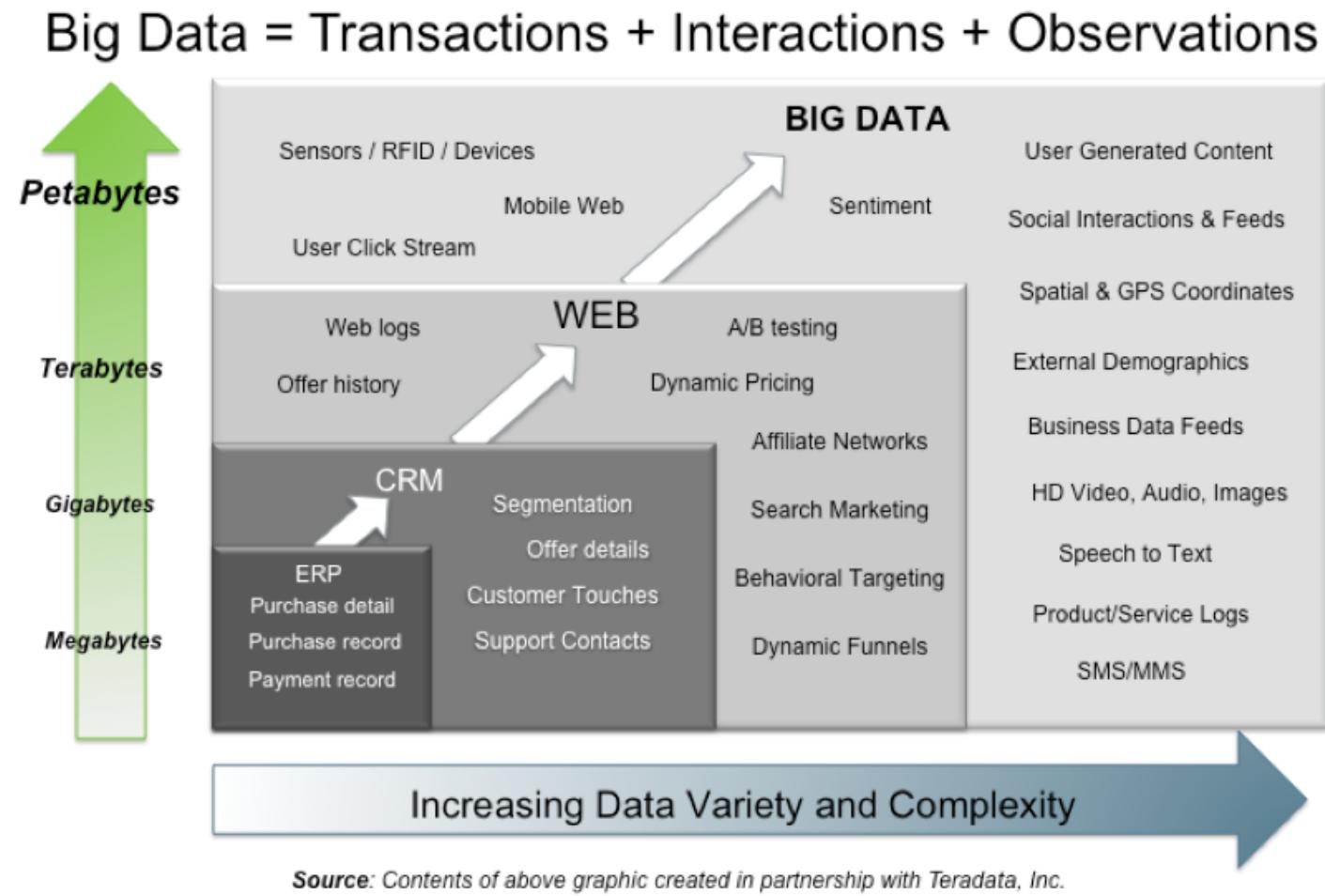


– Internet des Objets

- RFID (cartes de transport, codes bar, ...)
- le nombre d'objets connectés estimé dans le monde à 50 milliards en 2020 (12 milliards en 2013)

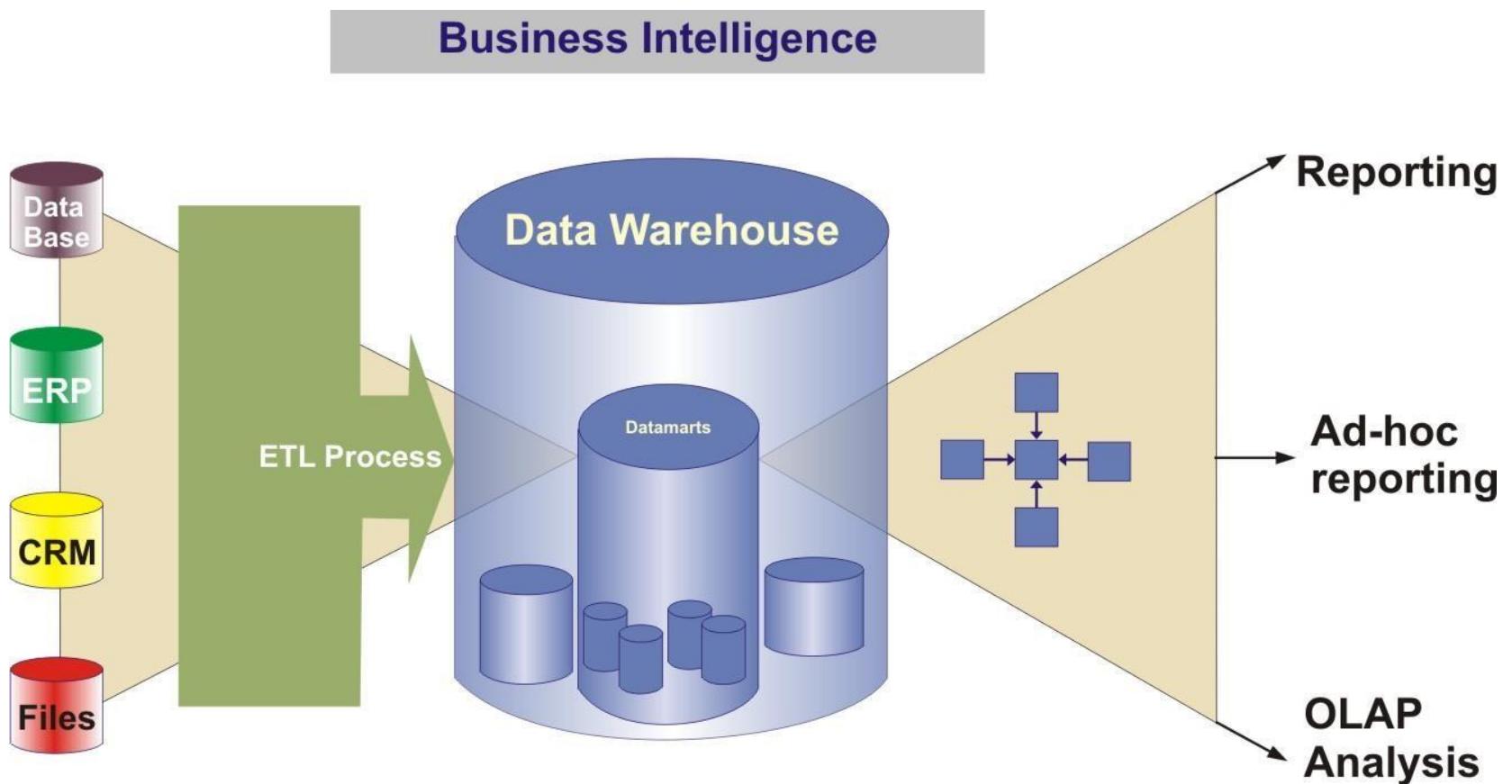
BIG DATA: Généralités

• DIVERSITE ET VOLUME DES SOURCES DE DONNEES (4/4)



BIG DATA: Généralités

- Business Intelligence (BI)



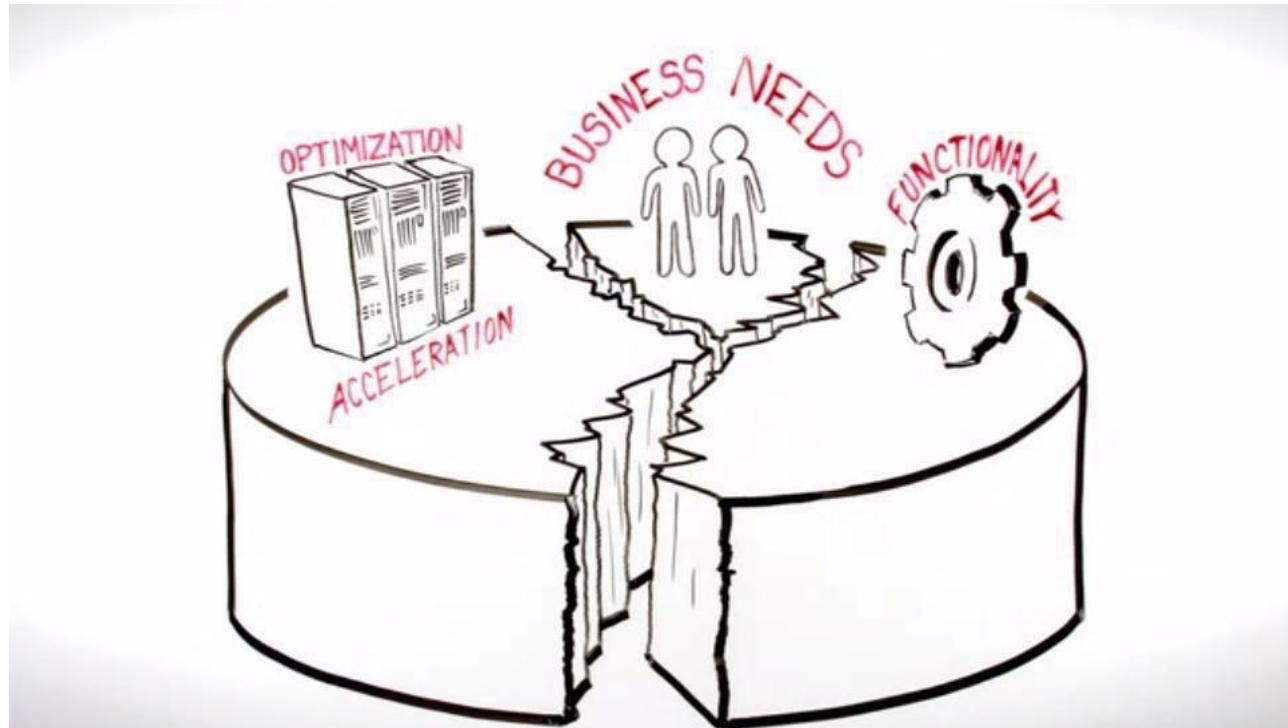
BIG DATA: Généralités

- **Quel est le problème posé par ces énormes quantités de données?**
 - Auparavant, quand les systèmes d'application de gestion de base de données ont été réalisés, ils ont été construits avec une **échelle à l'esprit** (limité). Même les organisations **n'ont pas été préparées à l'échelle** que nous produisons aujourd'hui.
 - Comme les exigences de ces organisations ont augmenté au fil du temps, ils doivent repenser et réinvestir dans l'infrastructure. Actuellement, le **coût** des ressources impliquées dans **l'extension de l'infrastructure, s'augmente** avec un facteur **exponentiel**.
 - De plus, il y aurait une **limitation** sur les différents facteurs tels que la **taille** de la machine, **CPU**, **RAM**, etc. qui peuvent être mis à l'échelle (scaled up). Ces systèmes **traditionnels** ne seraient pas en mesure de soutenir l'échelle requise par la plupart des entreprises.

BIG DATA: Généralités

- **ADAPTABILITE**

- Dans ce nouveau contexte, les **méthodes de traitement de ces données** (capture, stockage, recherche, partage, analyse, visualisation) **doivent être redéfinies** car l'ensemble de ces données deviennent **difficilement manipulables par les outils classiques**.

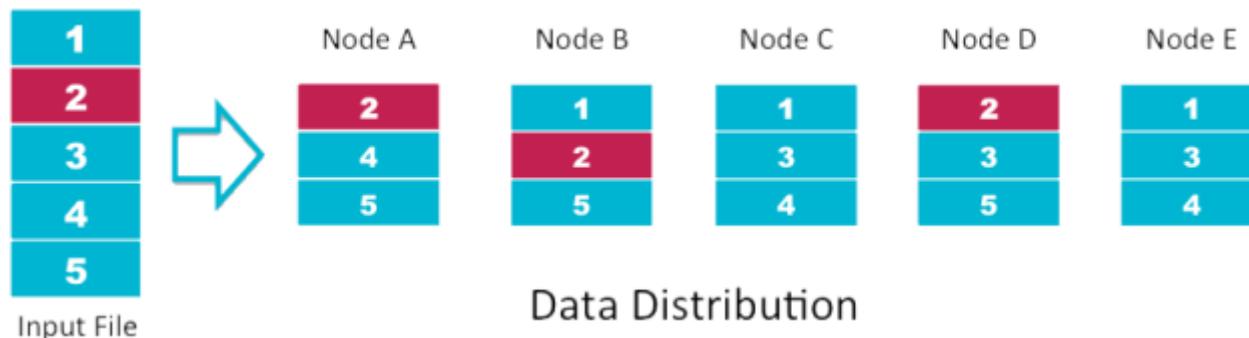


BIG DATA: Généralités

- **Comment le Big Data gère ces situations complexes? (1/3)**

- La plupart des outils et des frameworks de Big Data sont construits en gardant à l'esprit les caractéristiques suivantes :

- **La distribution des données:** Le grand ensemble de données est divisé en morceaux ou en petits blocs et réparti sur un nombre N de nœuds ou de machines. Ainsi les données sont réparties sur plusieurs noeuds et sont prêtes au traitement parallèle. Dans le monde du Big Data, ce type de distribution des données est réalisé à l'aide d'un Système de Fichiers Distribués - DFS (Distributed File System).



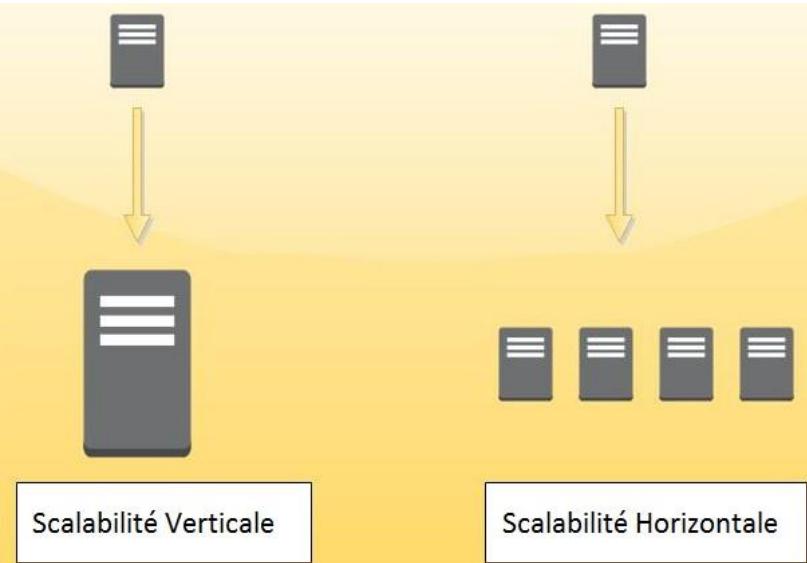
BIG DATA: Généralités

- **Comment le Big Data gère ces situations complexes? (2/3)**
 - **Le traitement en parallèle :** Les données distribuées obtiennent la puissance de N nombre de serveurs et de machines dont les données résident. Ces serveurs travaillent en parallèle pour le traitement et l'analyse. Après le traitement, les données sont fusionnées pour le résultat final recherché. (Actuellement ce processus est réalisé par MapReduce de Google qui sera détaillé dans une section ultérieure).
 - **La tolérance aux pannes:** En général, nous gardons la réplique d'un seul bloc (ou chunk) de données plus qu'une fois. Par conséquent, même si l'un des serveurs ou des machines est complètement en panne, nous pouvons obtenir nos données à partir d'une autre machine ou d'un autre « data center ». Encore une fois, nous pouvons penser que la réPLICATION de données pourrait coûter beaucoup d'espace. Mais voici le quatrième point de la rescousse.

BIG DATA: Généralités

- **Comment le Big Data gère ces situations complexes? (3/3)**

- **L'utilisation de matériel standard :** La plupart des outils et des frameworks Big Data ont besoin du matériel standard pour leur travail. Donc nous n'avons pas besoin de matériel spécialisé avec un conteneur spécial des données « RAID ». Cela réduit le coût de l'infrastructure totale.
- **Flexibilité, évolutivité et scalabilité :** Il est assez facile d'ajouter de plus en plus de nœuds dans le cluster quand la demande pour l'espace augmente. De plus, la façon dont les architectures de ces frameworks sont faites, convient très bien le scénario de Big Data.



BIG DATA: Généralités

- Exemple Simple (1/4)
 - Analyser une data set de 1TB

```
ABCSE,B7J,2008-10-28,6.48,6.74,6.22,6.72,44300,5.79
ABCSE,B7J,2008-10-27,6.21,6.78,6.21,6.40,55200,5.51
ABCSE,B7J,2008-10-24,6.39,6.66,6.21,6.40,67400,5.51
ABCSE,B7J,2008-10-23,6.95,6.95,6.50,6.59,59400,5.68
ABCSE,B7J,2008-10-22,6.92,7.17,6.80,6.80,55300,5.86
ABCSE,B7J,2008-10-21,7.20,7.30,7.10,7.10,54400,6.11
ABCSE,B7J,2008-10-20,6.94,7.31,6.94,7.12,45700,6.13
ABCSE,B7J,2008-10-17,6.43,6.93,6.42,6.90,57700,5.94
ABCSE,B7J,2008-10-16,6.61,6.69,6.21,6.53,83200,5.62
ABCSE,B7J,2008-10-15,6.84,6.90,6.36,6.36,78900,5.48
ABCSE,B7J,2008-10-14,7.15,7.32,6.93,6.96,74700,5.99
ABCSE,B7J,2008-10-13,6.00,6.57,6.00,6.57,75700,5.66
ABCSE,B7J,2008-10-10,5.05,5.72,4.79,5.72,158400,4.93
ABCSE,B7J,2008-10-09,6.30,6.41,6.00,6.02,140500,5.18
ABCSE,B7J,2008-10-08,5.60,6.47,5.60,6.28,292000,5.41
ABCSE,B7J,2008-10-07,7.59,7.59,6.66,6.69,89900,5.76
ABCSE,B7J,2008-10-06,7.83,7.90,7.00,7.40,159600,6.37
```

BIG DATA: Généralités

- Exemple Simple (2/4)

- Temps d'accès



BIG DATA: Généralités

- Exemple Simple (3/4)

Temps d'accès (lecture) > 2h

+

Temps de calcul ~ 1h

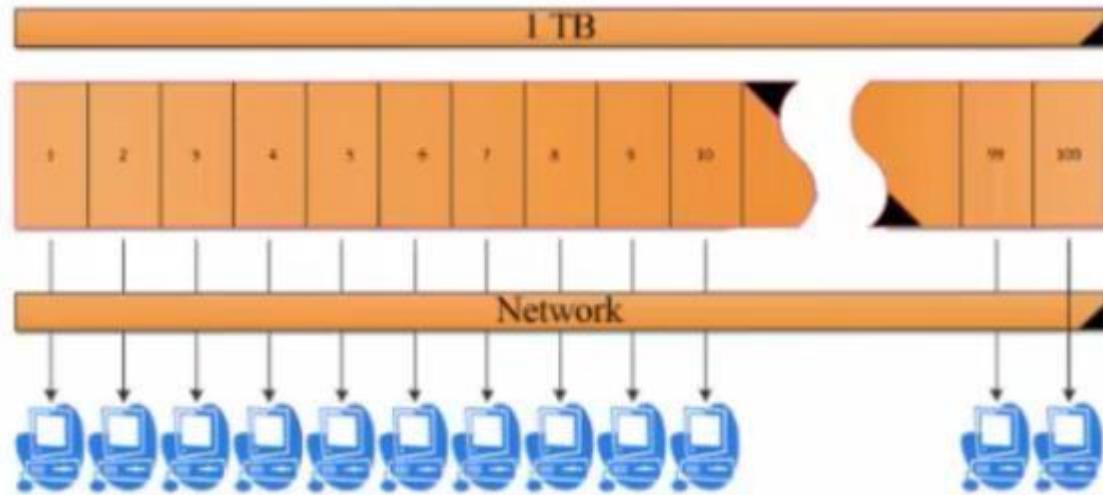
+

Bandé passante du réseau etc

> 3h

BIG DATA: Généralités

- Exemple Simple (4/4)
 - Division d'un fichier de 1TB en 100 blocs égaux
 - Lecture Parallèle



Temps de lecture = $150 \text{ min} / 100 < 2\text{min}$

Temps de calcul = $60 \text{ min} / 100 < 1 \text{ min}$

BIG DATA: Généralités

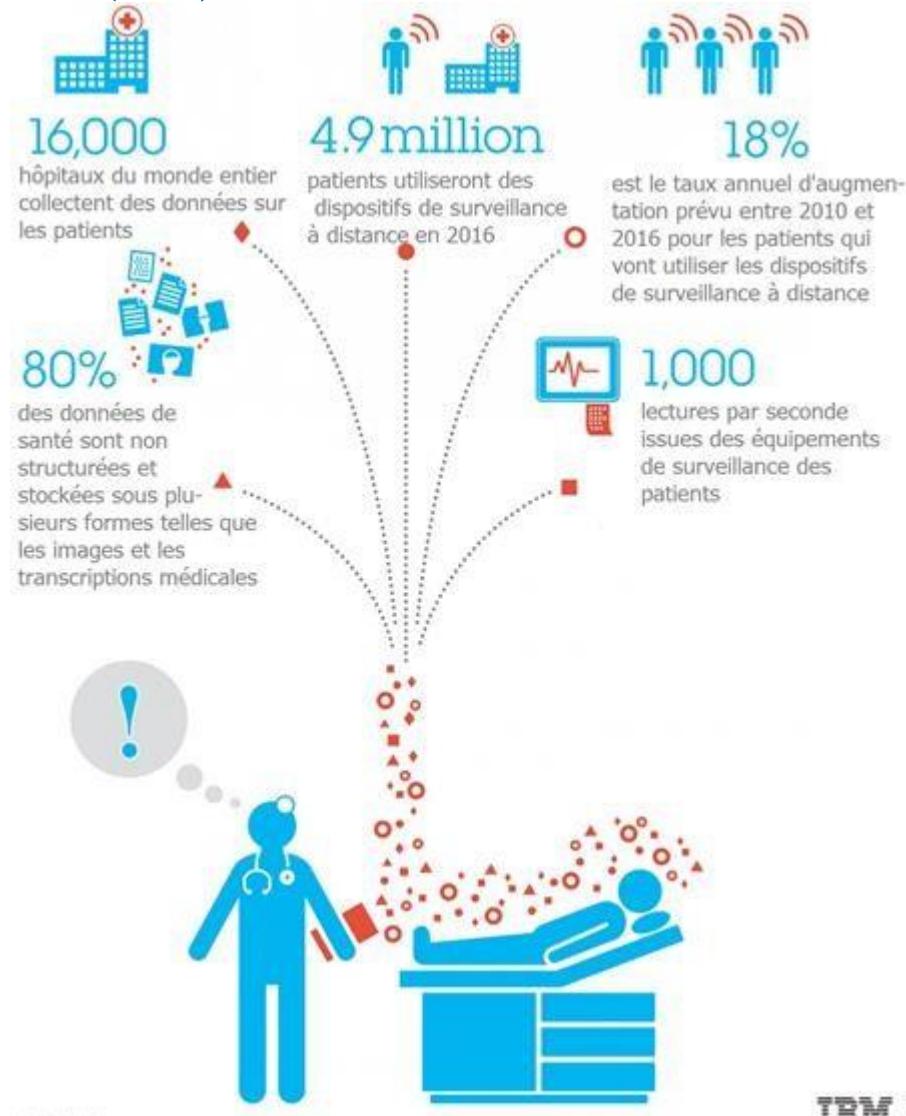
• PERSPECTIVES ET DOMAINES D'APPLICATION

- Les perspectives d'utilisation de ces données sont énormes, notamment pour l'analyse d'opinions politiques, de tendance industrielles, la génomique, la lutte contre la criminalité et la fraude, les méthodes de marketing publicitaire et de vente etc ...



BIG DATA: Généralités

- Cas d'utilisation : Santé (1/2)



BIG DATA: Généralités

- **Cas d'utilisation : Santé (2/2)**

- Analyse des données globales des patients et des résultats pour comparer l'efficacité des différentes interventions.
- Analyse des résultats de santé d'une population pour prévoir les maladies et les épidémies, savoir les causes environnementales et prendre les prévention nécessaire dans les stages primaires.
- Déploiement de systèmes d'aide à la décision clinique pour améliorer l'efficacité et la qualité des opérations.
- Télésurveillance des patients. La collecte de données pour les patients souffrant de maladies chroniques et l'analyse des données résultant pour surveiller la conformité et pour améliorer les futures options de médicaments et de traitement.

BIG DATA: Généralités

- **Cas d'utilisation : Marketing (1/2)**

- Plus d'intelligence pour plus de ventes.
- **Analyse prédictive** : En analysant l'historique des achats du client ou les fichiers Logs qui contiennent les pages visitées, l'entreprise peut prévoir ce que le client cherche et les mettre dans les zones des offres et publicités afin d'augmenter les achats.
- **Analyse des sentiments** : De Nombreuses sociétés utilisent les échanges sur les réseaux sociaux comme le reflet de l'opinion publique. Celle-ci devient une nouvelle source d'informations en temps réel directement fournie par le consommateur. Les questions d'e-réputation « à quoi est associée mon image ? » ou « comment est accueilli le nouveau produit que je viens de lancer ? » peuvent être analysées avec ces données. Le Big Data permet de prendre le pouls quasiment en direct, mesurer l'impact de sa marque, savoir comment est perçue la société par le public et anticiper les mauvaises critiques.

BIG DATA: Généralités

- **Cas d'utilisation : Marketing (2/2)**

- **Analyse des comportements** : L'analyse du comportement des clients en magasin permet d'améliorer l'aménagement du magasin, le mix produit et la disposition des produits dans les rayons et sur les étagères. Les dernières innovations ont également permis de suivre les habitudes d'achat (compter le nombre de pas effectués et le temps passé dans chaque rayon du magasin), géolocaliser en temps réel les clients,... Les données issues des tickets de caisse, captées depuis longtemps, peuvent maintenant être analysées et révèlent les habitudes d'achat des clients.



BIG DATA: Généralités

- Cas d'utilisation : Analyse de tweets en temps réel

The dashboard illustrates the analysis of tweets related to the London 2012 Cycling event. It features five main sections:

- Event Info:** Shows "London2012: Cycling" and "1- C. Froome (GBR)" with a time of "51:47.87".
- News Ticker:** Lists recent news items:
 - 97 min Froome target enters and stands as interim leader
 - 95 min split time at km 30
 - 94 min Rogers target enters and stands as interim leader
 - 92 min split times at mile 30
 - 90 min Cancellara passes over a minute of Wiggins at kilometer 30
- Tweets:** Displays a tweet from "LiamCleary1" (@ChristopherFroome) about Team GB leading the cycling race.
- Social Stats:** Provides a summary of tweet sentiment counts:
 - totalTweets = 6259
 - positiveTweets = 1101
 - neutralTweets = 4825
 - negativeTweets = 281It also lists the top hashtags:
 - #olympics2012
 - #london2012
 - #ironlungs
 - #teamgb
 - #cycling
- Social Map:** A map of the United Kingdom showing tweet locations with red dots.

Annotations on the dashboard highlight specific features:

- Thèmes les plus abordés** (Topics most discussed) points to the Tag Cloud section.
- Analyse de sentiments** (Sentiment analysis) points to the Social Stats section.
- Tweets sur l'évènement** (Tweets about the event) points to the Tweets section.
- Localisation des tweets** (Location of tweets) points to the Social Map section.

BIG DATA: Généralités

- **Cas d'utilisation : Politique**

- L'analyse de Big Data a joué un rôle important dans la campagne de réélection de Barack Obama, notamment pour analyser les opinions politiques de la population.
- Depuis l'année 2012, le Département de la défense américain investit annuellement sur les projets de Big Data plus de 250 millions de dollars.
- Le gouvernement américain possède six des dix plus puissants supercalculateurs de la planète.
- La National Security Agency est actuellement en train de construire le Utah Data Center. Une fois terminé, ce data center pourra supporter des yottaoctets d'information collectés par la NSA sur internet.

BIG DATA: Généralités

- **Cas d'utilisation : Sport (1/2)**

- La première source de données recueillie s'appuie sur des capteurs intégrés aux protège-tibias ou aux chaussures. Ces minuscules composants remontent des informations biométriques sur les joueurs :
 - la distance parcourue
 - les vitesses en sprint
 - les accélérations
 - le nombre de ballons touchés
 - le rythme cardiaque, etc.

→ A terme et quand l'analyse en temps réel sera réellement possible, on peut très bien imaginer qu'une alerte remonte lorsqu'un joueur fatigue afin que l'entraîneur le remplace.



BIG DATA: Généralités

- **Cas d'utilisation : Sport (2/2)**

- Une deuxième source de récolte de données provient de caméras installées en hauteur autour du terrain. Tous les déplacements des joueurs et leurs positions les uns par rapport aux autres sont ainsi filmés et enregistrés. Lors de son briefing, le tacticien peut ainsi comparer plusieurs fois par match la position géométrique de son équipe au moment des temps forts, quand l'équipe se montre offensive, s'ouvre des occasions et marque des buts.
- Le tacticien a également la capacité d'analyser le comportement de son équipe en fonction de la réaction de l'équipe concurrente. Ces données peuvent ensuite être agrégées avec d'autres sources telles que l'historique des matchs joués ou les données recueillies pendant les entraînements.



BIG DATA: Généralités

- **Cas d'utilisation : Sécurité publique (1/2)**
 - Aujourd’hui, avec le Big Data, la vidéosurveillance va beaucoup plus loin : elle permet d’analyser automatiquement les images et les situations, de croiser les informations, et d’envoyer des alertes.
 - Cette analyse de vidéo avancée est utilisée en particulier pour :
 - la sécurité du trafic (routier, ferroviaire, maritime et aérien)
 - la protection des espaces et des bâtiments publics
 - la sécurité personnelle.
 - Il est aujourd’hui possible à travers l’analyse des images vidéo de faire de :
 - la reconnaissance d’objets et de mouvements
 - la lecture de plaques minéralogiques
 - la détection de véhicule non autorisé
 - la reconnaissance faciale
 - l’auto-surveillance avec possibilité de déclenchement d’alertes ou autres actions automatisées.

BIG DATA: Généralités

- Cas d'utilisation : Sécurité publique (2/2)



- A titre d'exemple la ville de Londres avait, quant à elle, mis en place un système de reconnaissance faciale lors des jeux olympiques de 2012 organisés dans la capitale, afin de lutter contre le terrorisme pour lequel l'alerte était à son maximum.

BIG DATA: Acteurs et Solutions

- Les grands acteurs du web tel que **Google, Yahoo, Facebook, Twitter, LinkedIn** ... ont été les premiers à être confrontés à des volumétries de données extrêmement importantes et ont été à l'origine des premières innovations en la matière portées principalement sur deux types de technologies :
 - ! **1. Les plateformes de développement et de traitement des données (GFS, Hadoop, Spark,...)**
 - 2. Les bases de données (NoSql)**

BIG DATA: Historique

- **Historique : Big Data, Google : Le système de fichier GFS**



- Pour stocker son index grandissant quelle solution pour Google ?

1. Utilisation d'un SGBDR ?

- Problème de distribution des données
- Problème du nombre d'utilisateurs
- Problème de Vitesse du moteur de recherche

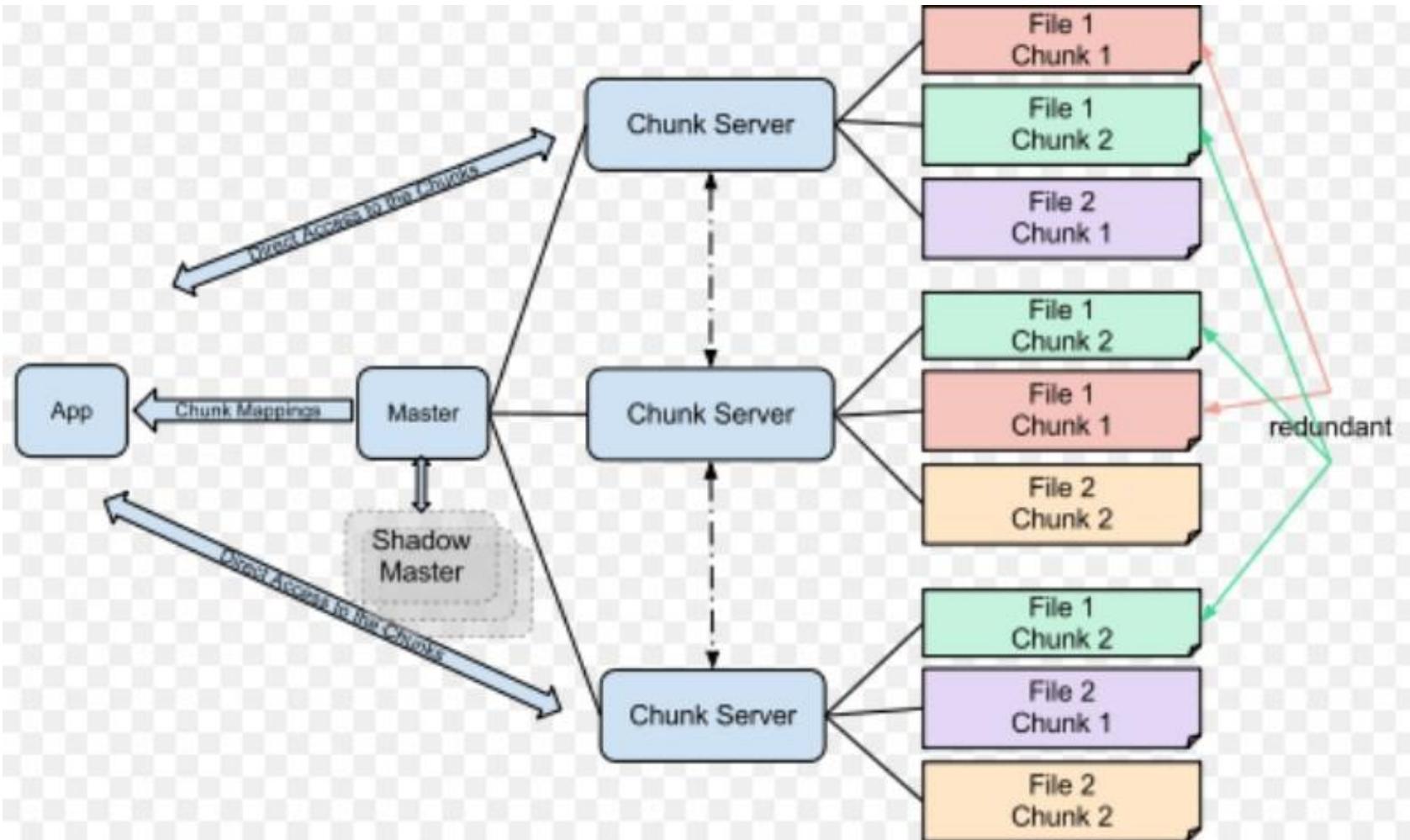


2. Invention d'un nouveau système propriétaire : GFS (Google File Système) en 2003 ✓



BIG DATA: Historique

- Historique : Big Data, Google : Le système de fichier GFS



BIG DATA: Historique

- **Historique : Big Data, Comment exploiter ce système de fichier ?**



- La notion de Big Data est intimement lié à la capacité de traitements de gros volumes.
 - Le premier Article a été publié en 2004 : Jeffrey Dean and Sanjay Ghemawat
 - **MapReduce : Simplified Data Processing on Large Clusters**
 - C'est un algorithme inventé par Google, Inc afin de distribuer des traitements sur un ensemble de machines avec le système GFS.
 - Google possède aujourd'hui plus de 1 000 000 de serveurs interconnectés dans le monde.

BIG DATA: Les Acteurs de l'Open Source

- Les contributeurs de l'implémentation Libre et Open Source :

YAHOO!

facebook

Linked in

twitter

- Ces entreprises ont décidé d'ouvrir leurs développements internes au monde Open Source.
- Un certains nombre de ces technologies comme « **Hadoop** » et « **Spark** » font partie de la fondation Apache et ont été intégrés aux offres de « **Big Data** » des grands acteurs tel que **IBM**, **Oracle**, **Microsoft**, **EMC** ...

BIG DATA: Plateforme – Technologies - Outils

Société	Technologie développée	Type de technologie
Google	Big Table	Système de base de données distribuée propriétaire reposant sur GFS (Google File System). Technologie non Open Source, mais qui a inspiré Hbase qui est Open Source.
	MapReduce	Plate-forme de développement pour traitements distribués
Yahoo	Hadoop	Plateforme Java destinée aux applications distribuées et à la gestion intensive des données. Issue à l'origine de GFSet MapReduce.
	S4	Plateforme de développement dédiée aux applications de traitement continu de flux de données.
Facebook	Cassandra	Base de données de type NoSQL et distribuée.
	Hive	Logiciel d'analyse de données utilisant Hadoop.
Twitter	Storm	Plateforme de traitement de données massives.
	FlockDB	Base de données distribuée de type graphe.
LinkedIn	SenseiDB	Base de données temps réel distribuée et semi-structurée.
	Tableau : Quelques technologies Open Source du Big Data	Quelques technologies Open Source du Big Data
	Voldemort	Base de données distribuée destinée aux très grosses volumétries

BIG DATA: Plateforme – Technologies - Outils

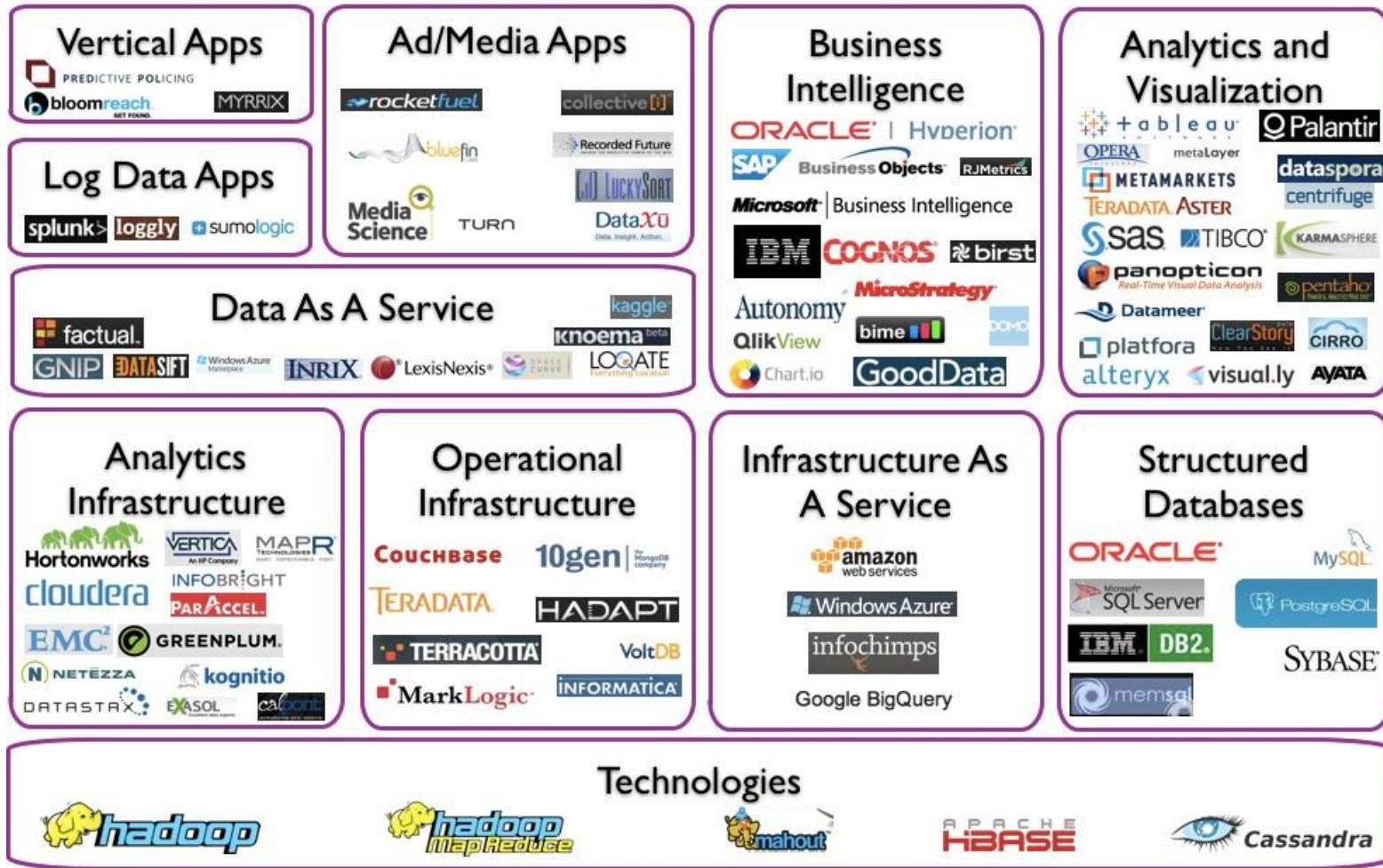
- Processing
 - Hadoop, Hive, Pig, mrjob, Caffeine
- NoSQL Databases
 - Hbase, MongoDB, Vertica, Cassandra, Neo4j, etc.
- Servers
 - EC2, Google App Engine, Elastic, Beanstalk, Heroku
- Analytics
 - R, SAS, Python scikit-learn, Spark MLLib, Apache Mahout
- Search
 - Solr/Lucene, ElasticSearch

BIG DATA: Plateforme – Technologies - Outils



BIG DATA Landscape (1/3)

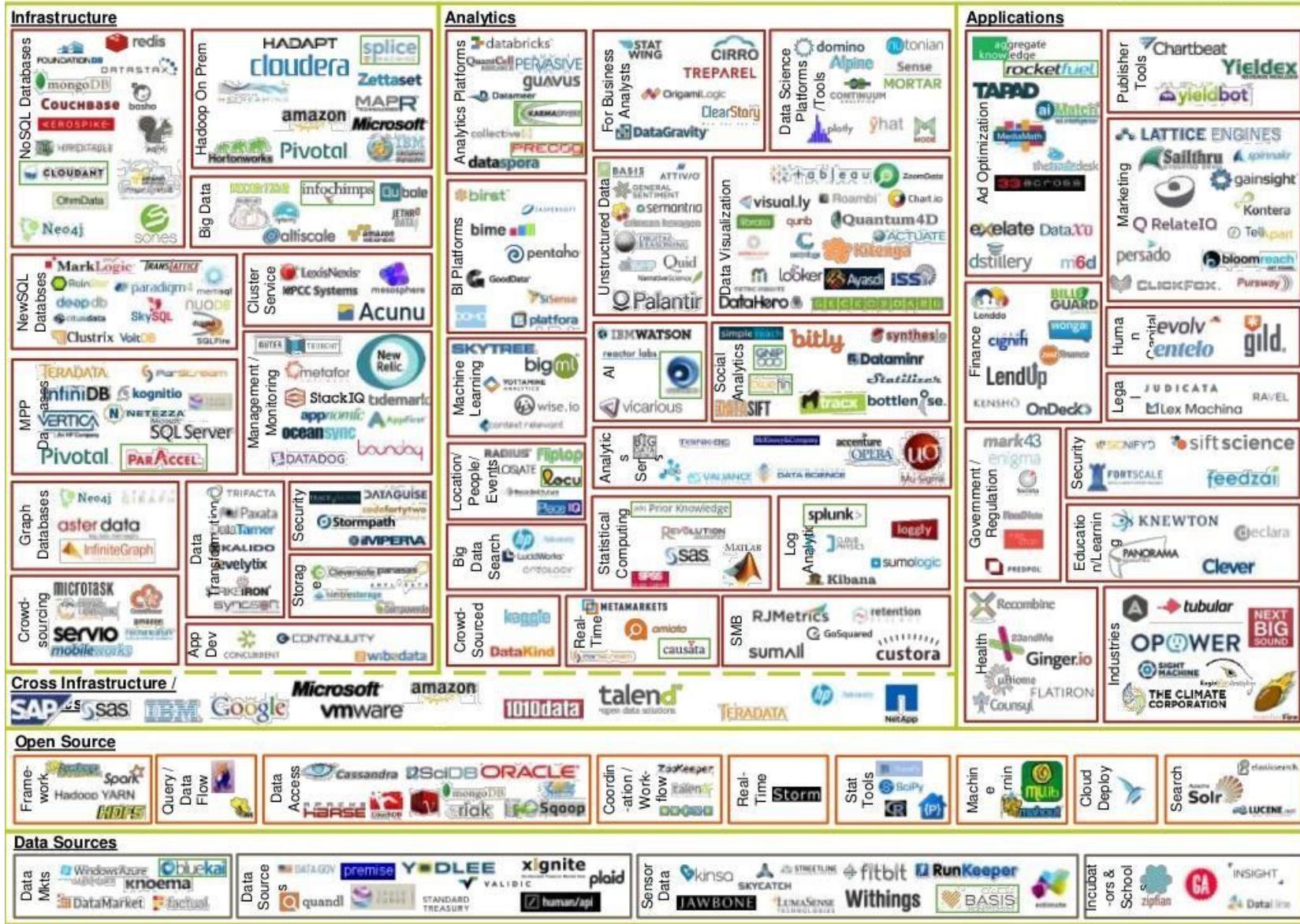
Big Data Landscape



BIG DATA Landscape (2/3)

BIG DATA LANDSCAPE, VERSION 3.0

Exited: Acquisition or
IPO



BIG DATA Landscape (3/3)

BIG DATA LANDSCAPE 2017

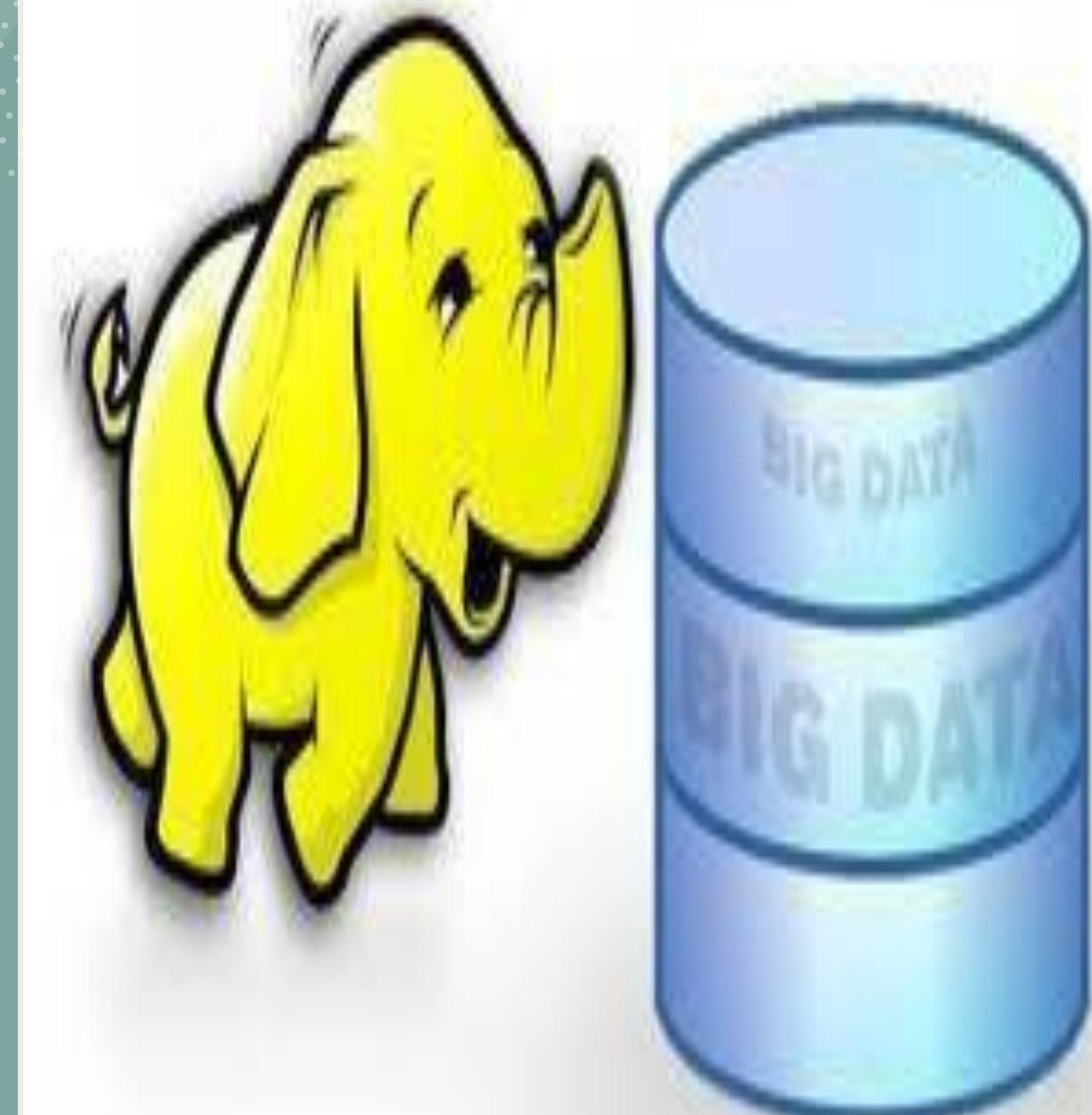


BIG DATA: Conclusion

- Nous sommes actuellement dans l'ère de la production massive de données. D'une part, les applications génèrent des données issues des logs, des réseaux de capteurs, des rapports de transactions, des traces de GPS, etc. et d'autre part, les individus produisent des données telles que des photographies, des vidéos, des musiques ou encore des données sur l'état de santé (rythme cardiaque, pression ou poids).
- Un problème se pose alors quant au stockage et à l'analyse des données. La capacité de stockage des disques durs augmente mais le temps de lecture croît également. Il devient alors nécessaire de paralléliser les traitements en stockant sur plusieurs machines.
- Plusieurs solutions, inspirées des solutions de Google, ont été proposées. Hadoop est la solution la plus répondue au monde de BigData. Cette solution sera détaillée dans le chapitre suivant.

Chapitre 2

Hadoop



Hadoop: Présentation (1/3)

- **Hadoop** est une plateforme (framework) open source conçue pour réaliser d'une façon distribuée des traitements sur des volumes de données massives, de l'ordre de plusieurs pétaoctets. Ainsi, il est destiné à faciliter la création d'applications distribuées et échelonnables (scalables). Il s'inscrit donc typiquement sur le terrain du Big Data.
- **Hadoop** est géré sous l'égide de la fondation Apache, il est écrit en Java.
- **Hadoop** a été conçu par Doug Cutting en 2004 et été inspiré par les publications MapReduce, GoogleFS et BigTable de Google.
-

Hadoop: Présentation (2/3)

- Modèle simple pour les développeurs: il suffit de développer des tâches Map-Reduce, depuis des interfaces simples accessibles via des librairies dans des langages multiples (Java, Python, C/C++...).
- Déployable très facilement (paquets Linux pré-configurés), configuration très simple elle aussi.
- S'occupe de toutes les problématiques liées au calcul distribué, comme l'accès et le partage des données, la tolérance aux pannes, ou encore la répartition des tâches aux machines membres du cluster : le programmeur a simplement à s'occuper du développement logiciel pour l'exécution de la tâche.

Hadoop: Présentation (3/3)

Hadoop assure les critères des solutions Big Data:

- **Performance:** support du traitement d'énormes data sets (millions de fichiers, Go à To de données totales) en exploitant le parallélisme et la mémoire au sein des clusters de calcul.
- **Economie:** contrôle des coûts en utilisant de matériels de calcul de type standard.
- **Evolutivité (scalabilité):** un plus grand cluster devrait donner une meilleure performance.
- **Tolérance aux pannes:** la défaillance d'un nœud ne provoque pas l'échec de calcul.
- **Parallélisme de données:** le même calcul effectué sur toutes les données.

Hadoop : Historique (1/2)

- **2002:** Doug Cutting (directeur archive.org) et Mike Cafarella (étudiant) développent Nutch, un moteur de recherche Open Source exploitant le calcul distribué. L'implémentation peut tourner seulement sur quelques machines et a de multiples problèmes, notamment en ce qui concerne l'accès et le partage de fichiers.
- **2003/2004:** le département de recherche de Google publie deux whitepapers, le premier sur GFS (un système de fichier distribué) et le second sur le paradigme Map/Reduce pour le calcul distribué.
- **2004:** Doug Cutting et Mike Cafarella développent un framework (encore assez primitif) inspiré des papiers de Google et portent leur projet Nutch sur ce framework.
- **2006:** Doug Cutting (désormais chez Yahoo) est en charge d'améliorer l'indexation du moteur de recherche de Yahoo. Il exploite le framework réalisé précédemment...

Hadoop : Historique (2/2)

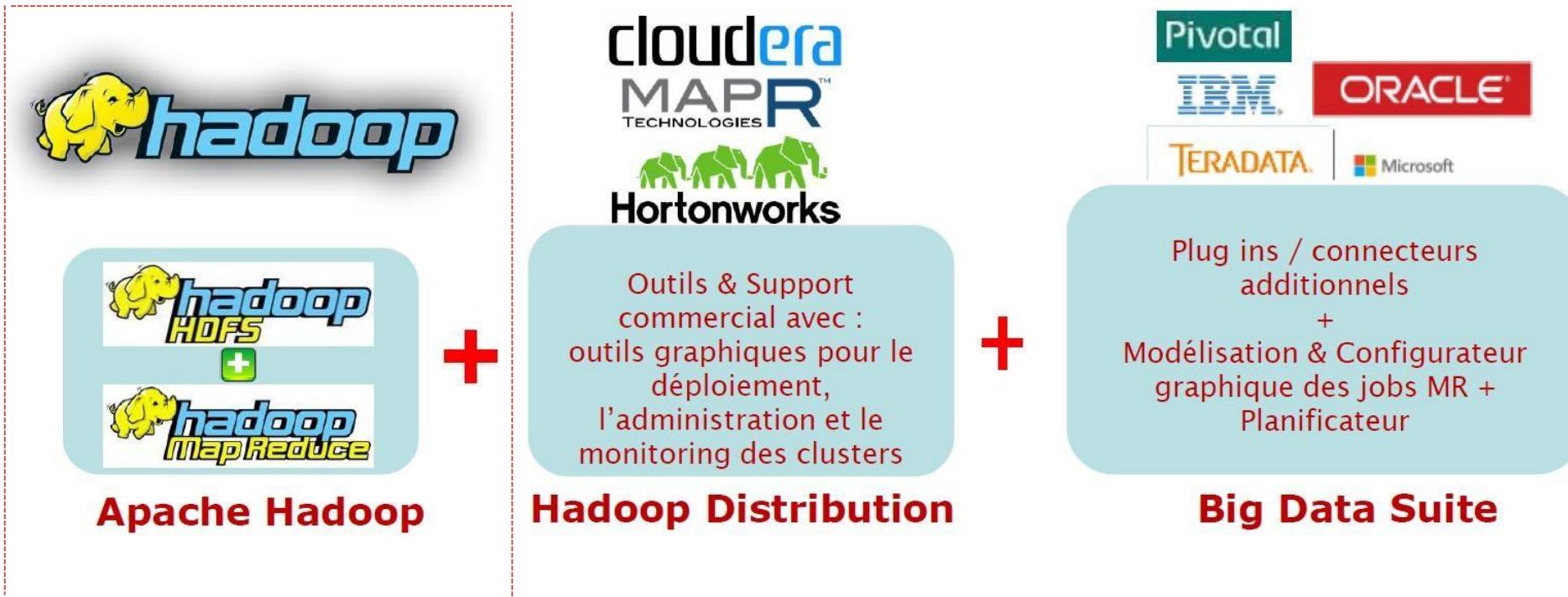
Il crée alors une nouvelle version améliorée du framework en tant que projet Open Source de la fondation Apache, qu'il nomme Hadoop (le nom d'un éléphant en peluche de son fils).

A l'époque, Hadoop est encore largement en développement – un cluster pouvait alors comporter au maximum 5 à 20 machines, etc.



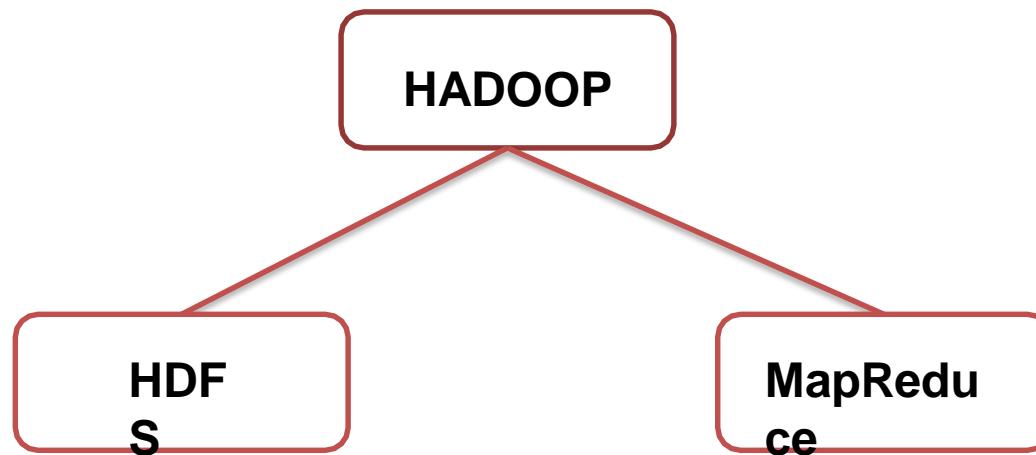
- **2008:** le développement est maintenant très abouti, et Hadoop est exploité par le moteur de recherche de Yahoo Ainsi que par de nombreuses autres divisions de l'entreprise.
- **2011:** Hadoop est désormais utilisé par de nombreuses autres entreprises et des universités, et le cluster Yahoo comporte 42000 machines et des centaines de petaoctets d'espace de stockage.

Hadoop : Choix



Hadoop: Composants fondamentaux

- Hadoop est constitué de deux grandes parties :
 - Hadoop Distributed File System – **HDFS** : destiné pour le stockage distribué des données
 - Distributed Programming Framework - **MapReduce** : destiné pour le traitement distribué des données.



HDFS : Présentation (1/3)

- HDFS (**Hadoop Distributed File System**) est un système de fichiers :
 - **Distribué**
 - **Extensible**
 - **Portable**
 - Développé par Hadoop, inspiré de GoogleFS et écrit en **Java**.
 - **Tolérant aux pannes**
 - Conçu pour stocker de **très gros volumes** de données sur un grand nombre de machines **peu couteuses** équipées de disques durs banalisés.
- HDFS permet **l'abstraction de l'architecture physique de stockage**, afin de manipuler un système de fichiers distribué comme s'il s'agissait d'un disque dur unique.
- HDFS reprend de nombreux **concept proposés par des systèmes de fichiers classiques comme ext2 pour Linux ou FAT pour Windows**. Nous retrouvons donc la notion de **blocs** (la plus petite unité que l'unité de stockage peut gérer), les **métadonnées** qui permettent de retrouver les blocs à partir d'un nom de fichier, les **droits** ou encore **l'arborescence** des répertoires.

HDFS : Présentation (2/3)

- Toutefois, HDFS se démarque d'un système de fichiers classique pour les principales raisons suivantes :
 - HDFS n'est pas **solidaire du noyau du système d'exploitation**. Il assure une **portabilité** et peut être déployé sur différents systèmes d'exploitation. Un des inconvénients est de devoir solliciter une application externe pour monter une unité de disque HDFS.
 - HDFS est un système **distribué**. Sur un système classique, la **taille** du disque est généralement considérée comme la **limite globale** d'utilisation. Dans un système **distribué** comme HDFS, chaque **nœud** d'un cluster correspond à un **sous-ensemble** du volume global de données du cluster. **Pour augmenter ce volume global, il suffira d'ajouter de nouveaux nœuds.** On retrouvera également dans HDFS, un service central appelé Namenode qui aura la tâche de gérer les métadonnées.

HDFS : Présentation (3/3)

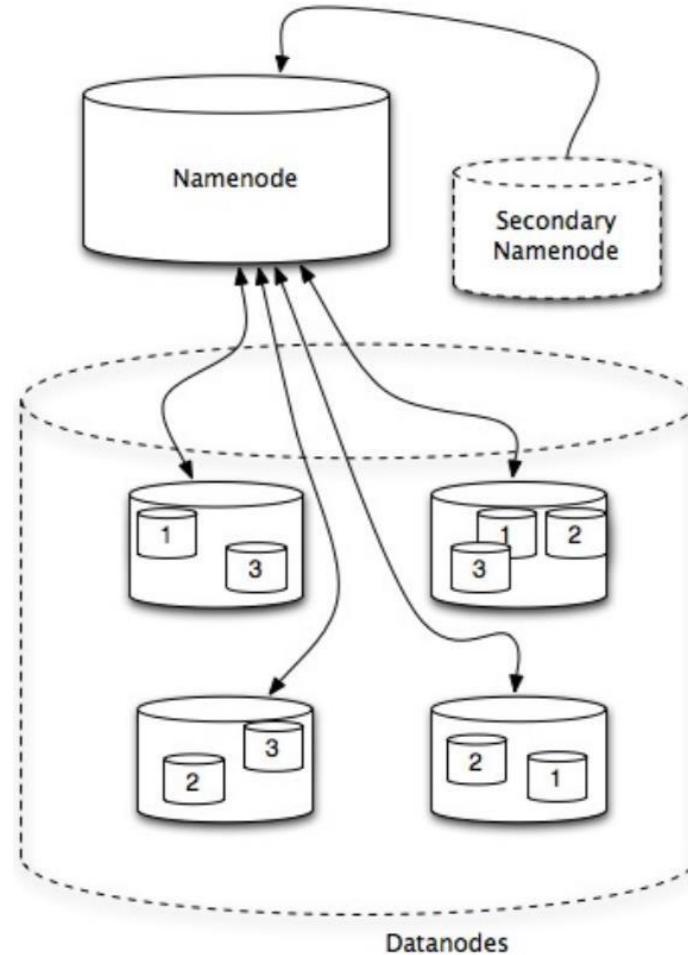
- HDFS utilise des **tailles de blocs largement supérieures** à ceux des systèmes classiques. Par défaut, la taille est fixée à **64 Mo**. Il est toutefois possible de monter à 128 Mo, 256 Mo, 512 Mo voire 1 Go. Alors que sur des systèmes classiques, la taille est généralement de **4 Ko**, l'intérêt de fournir des tailles plus grandes **permet de réduire le temps d'accès** à un bloc. Notez que si la taille du fichier est inférieure à la taille d'un bloc, le fichier n'occupera pas la taille totale de ce bloc.
- HDFS fournit un **système de réPLICATION des blocs** dont le nombre de réPLICATIONS est configurable. Pendant la phase d'écriture, chaque bloc correspondant au fichier est réPLIqué sur plusieurs nœuds. Pour la phase de lecture, si un bloc est indisponible sur un nœud, des copies de ce bloc seront disponibles sur d'autres nœuds.

HDFS : Architecture (1/3)

- Une architecture de machines HDFS (aussi appelée cluster HDFS) repose sur trois types de composants majeurs :

- **NameNode (noeud de nom)** :

Un Namenode est un **service central** (généralement appelé aussi **maître**) qui s'occupe de **gérer l'état** du système de fichiers. Il **maintient l'arborescence** du système de fichiers et **les métadonnées** de l'ensemble des fichiers et répertoires d'un système Hadoop. Le Namenode a une connaissance des Datanodes (étudiés juste après) dans lesquels les blocs sont stockés.



HDFS : Architecture (2/3)

- **Secondary Namenode :**

Le Namenode dans l'architecture Hadoop est un **point unique de défaillance** (Single Point of Failure en anglais). Si ce service est arrêté, il n'y a pas un moyen de pouvoir extraire les blocs d'un fichier donné. Pour répondre à cette problématique, un Namenode secondaire appelé Secondary Namenode a été mis en place dans l'architecture **Hadoop version2**. Son fonctionnement est relativement simple puisque le **Namenode secondaire vérifie périodiquement l'état du Namenode principal et copie les métadonnées**. Si le **Namenode principal est indisponible, le Namenode secondaire prend sa place.**

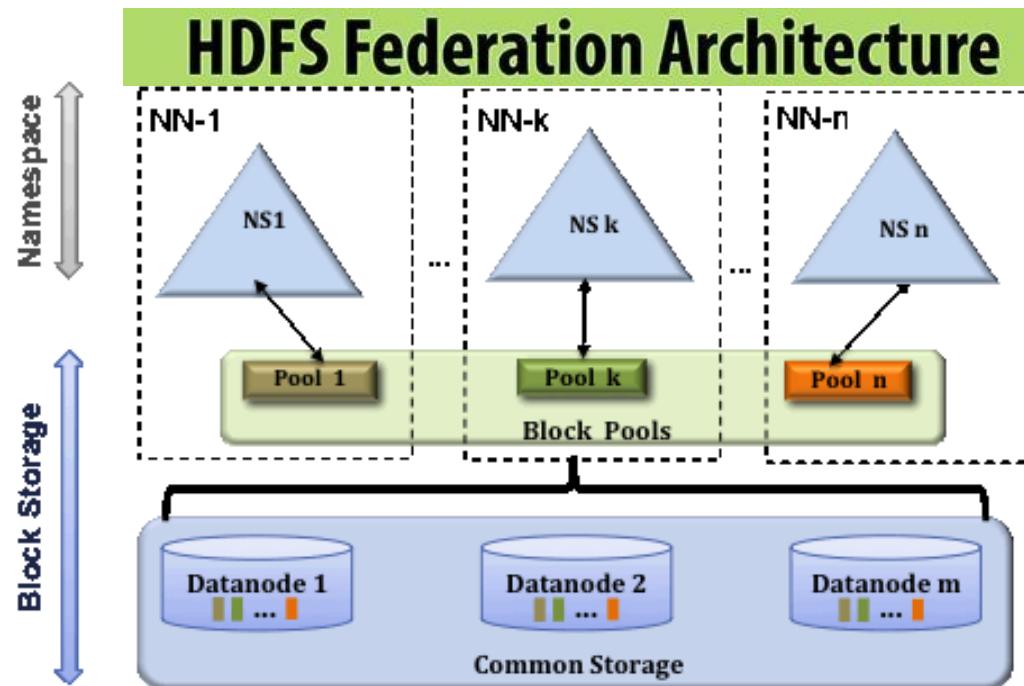
- **Datanode :**

Précédemment, nous avons vu qu'un Datanode **contient les blocs de données**. En effet, il stocke les blocs de données lui-mêmes. Il y a un DataNode pour chaque machine au sein du cluster. Les Datanodes sont sous les ordres du Namenode et sont surnommés les **Workers**. Ils sont donc sollicités par les Namenodes lors des opérations de lecture et d'écriture.

HDFS : Architecture (3/3)

- Remarque : En plus de l'ajout du Namenode secondaire, nous trouvons d'autres améliorations de HDFS dans la 2^{ème} version de Hadoop :

Les NameNodes étant le point unique pour le stockage et la gestion des métadonnées, ils peuvent être un goulot d'étranglement pour soutenir un grand nombre de fichiers, notamment lorsque ceux-ci sont de petite taille. **En acceptant des espaces de noms multiples desservis par des NameNodes séparés, le HDFS limite ce problème.** Par exemple, un NameNode pourrait gérer tous les fichiers sous /user, et un second NameNode peut gérer les fichiers sous /share.

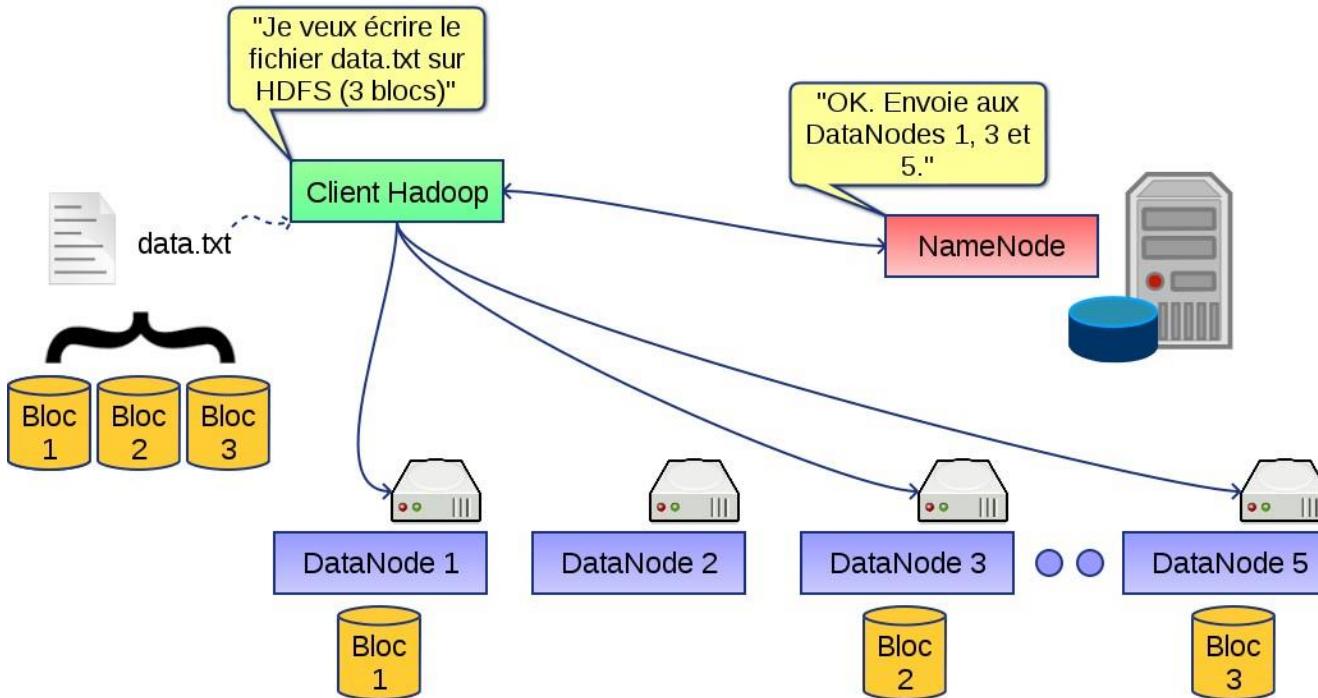


HDFS: Écriture d'un fichier (1/2)

- Si on souhaite écrire un fichier au sein de HDFS, on va utiliser la commande principale de gestion de Hadoop: **hadoop**, avec l'option **fs**. Mettons qu'on souhaite stocker le fichier `page_livre.txt` sur HDFS.
- Le programme va diviser le fichier en blocs de 64 Mo (ou autre, selon la configuration) – supposons qu'on ait ici 2 blocs. Il va ensuite annoncer au NameNode: « Je souhaite stocker ce fichier au sein de HDFS, sous le nom `page_livre.txt` ».
- Le NameNode va alors indiquer au programme qu'il doit stocker le bloc 1 sur le DataNode numéro 3, et le bloc 2 sur le DataNode numéro 1.
- Le client `hadoop` va alors contacter directement les DataNodes concernés et leur demander de stocker les deux blocs en question. Par ailleurs, les DataNodes s'occuperont – en informant le NameNode – de répliquer les données entre eux pour éviter toute perte de données.

HDFS: Écriture d'un fichier (2/2)

Ecriture HDFS

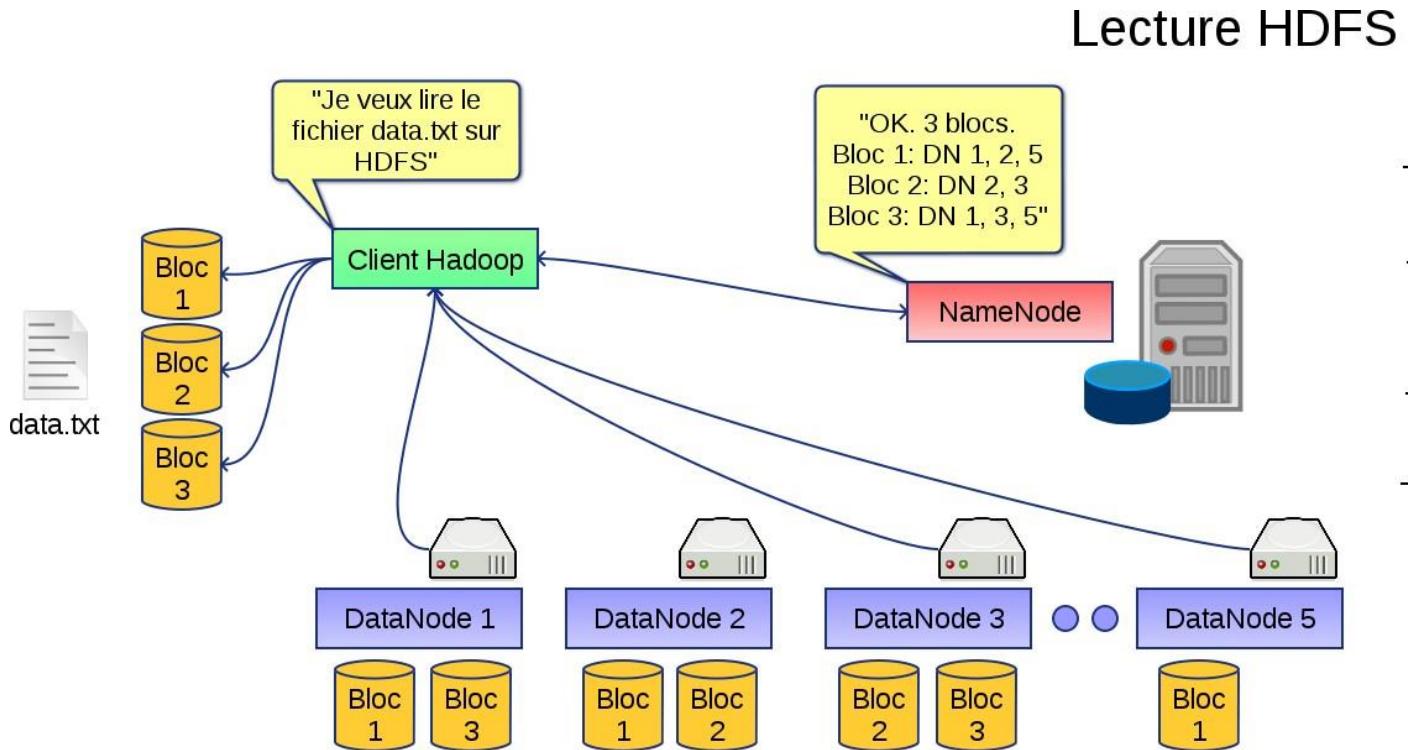


- Le client indique au NameNode qu'il souhaite écrire un bloc.
- Celui-ci lui indique le DataNode à contacter.
- Le client envoie le bloc au Datanode.
- Les DataNodes répliquent le bloc entre eux.
- Le cycle se répète pour le bloc suivant.

HDFS: Lecture d'un fichier (1/2)

- Si on souhaite lire un fichier au sein de HDFS, on utilise là aussi le client Hadoop. Mettons qu'on souhaite lire le fichier `page_livre.txt`.
- Le client va contacter le NameNode, et lui indiquer « Je souhaite lire le fichier `page_livre.txt` ». Le NameNode lui répondra par exemple « Il est composé de deux blocs. Le premier est disponible sur le DataNode 3 et 2, le second sur le DataNode 1 et 3 ».
- Là aussi, le programme contactera les DataNodes directement et leur demandera de lui transmettre les blocs concernés. En cas d'erreur/non réponse d'un des DataNode, il passe au suivant dans la liste fournie par le NameNode.

HDFS: Lecture d'un fichier (2/2)



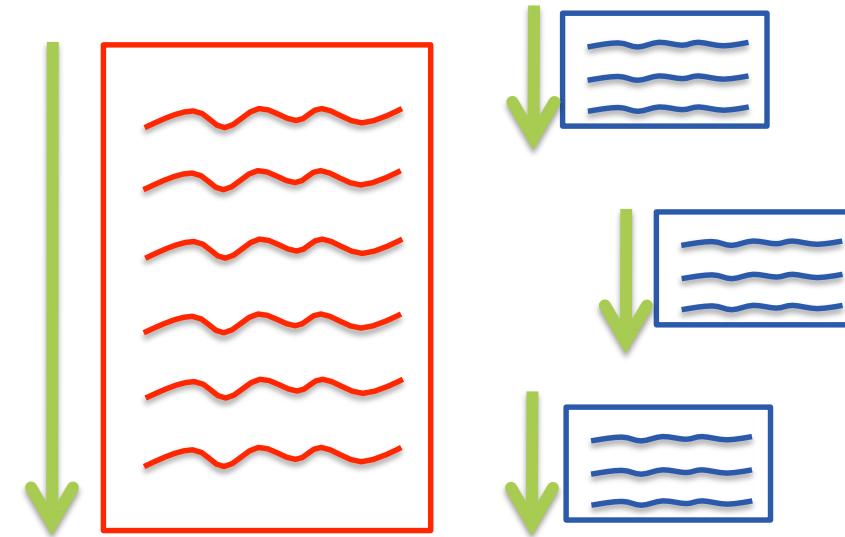
- Le client indique au NameNode qu'il souhaite lire un fichier.
- Celui-ci lui indique sa taille et les différents DataNode contenant les N blocs.
- Le client récupère chacun des blocs à un des DataNodes.
- Si un DataNode est indisponible, le client le demande à un autre.

HDFS: La commande Hadoop fs

- Comme indiqué plus haut, la commande permettant de stocker ou extraire des fichiers de HDFS est l'utilitaire console hadoop, avec l'option fs. Il réplique globalement les commandes systèmes standard Linux, et est très simple à utiliser:
 - `hadoop fs -put livre.txt /data_input/livre.txt`
 - Pour stocker le fichier `livre.txt` sur HDFS dans le répertoire `/data_input`.
 - `hadoop fs -get /data_input/livre.txt livre.txt`
 - Pour obtenir le fichier `/data_input/livre.txt` de HDFS et le stocker dans le fichier local `livre.txt`.
 - `hadoop fs -mkdir /data_input`
 - Pour créer le répertoire `/data_input`
 - `hadoop fs -rm /data_input/livre.txt`
 - Pour supprimer le fichier `/data_input/livre.txt`
 - D'autres commandes usuelles: `-ls`, `-cp`, `-rmdir`, `-du`, etc...

MapReduce : Présentation (1/5)

- 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 : Présentation (2/5)

- Pour exécuter un **problème** large de **manière distribuée**, il faut pouvoir **découper** le problème en plusieurs problèmes de taille réduite à **exécuter sur chaque machine du cluster** (stratégie algorithmique dite du divide and conquer / diviser pour régner).
- De multiples approches existent et ont existé pour cette division d'un problème en plusieurs « sous-tâches ».
- MapReduce est un **paradigme** (un **modèle**) visant à généraliser les approches existantes pour produire une **approche unique** applicable à tous les problèmes.
- MapReduce existait déjà depuis longtemps, notamment dans les langages fonctionnels (Lisp, Scheme), mais la présentation du paradigme sous une forme « **rigoureuse** », **généralisable** à tous les problèmes et orientée **calcul distribué** est attribuable à un whitepaper issu du département de recherche de **Google** publié en **2004** (« MapReduce: Simplified Data Processing on Large Clusters »).

MapReduce : Présentation (3/5)

MapReduce définit deux opérations distinctes à effectuer sur les données d'entrée:

- **La première, MAP**, va transformer les données d'entrée en une série de **couples clef/valeur**. Elle va regrouper les données en les associant à des clefs, choisies de telle sorte que les couples clef/valeur aient un sens par rapport au problème à résoudre. Par ailleurs, cette opération doit être **parallélisable**: on doit pouvoir découper les données d'entrée en plusieurs fragments, et faire **exécuter l'opération MAP à chaque machine du cluster** sur un fragment distinct.
- **La seconde, REDUCE**, va appliquer un **traitement** à toutes les valeurs de chacune des clefs distinctes produite par l'opération MAP. Au terme de l'opération REDUCE, on aura **un résultat pour chacune des clefs distinctes**. Ici, on attribuera à chacune des machines du cluster une des clefs uniques produites par MAP, en lui donnant la liste des valeurs associées à la clef. **Chacune des machines effectuera alors l'opération REDUCE** pour cette clef.

MapReduce : Présentation (4/5)

On distingue donc 4 étapes distinctes dans un traitement MapReduce:

- **Découper** (**split**) les données d'entrée en plusieurs fragments.
- **Mapper** chacun de ces fragments pour obtenir des couples (clef ; valeur).
- **Grouper** (**shuffle**) ces couples (clef ; valeur) par clef.
- **Réduire** (**reduce**) les groupes indexés par clef en une forme finale, avec une valeur pour chacune des clefs distinctes.

En modélisant le problème à résoudre de la sorte, on le rend parallélisable – chacune de ces tâches à l'exception de la première seront effectuées de manière distribuée.

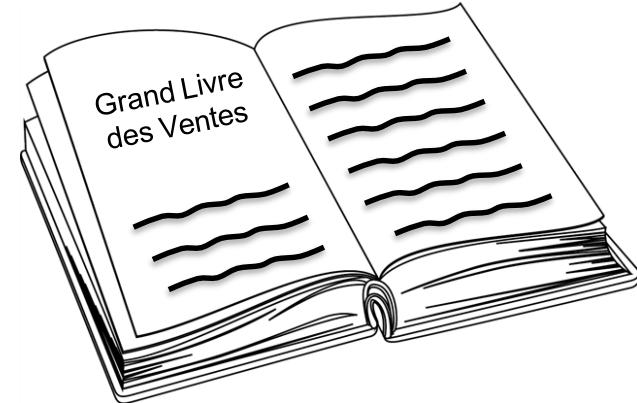
MapReduce : Présentation (5/5)

Pour résoudre un problème via la méthodologie MapReduce avec Hadoop, on devra donc:

- Choisir une manière de découper les données d'entrée de telle sorte que l'opération MAP soit parallélisable.
 - Définir quelle CLEF utiliser pour notre problème.
 - Écrire le programme pour l'opération MAP.
 - Écrire le programme pour l'opération REDUCE..
- ... et Hadoop se chargera du reste (problématiques calcul distribué, groupement par clef distincte entre MAP et REDUCE, etc.).

Map-Reduce Exemple

- Imaginons que vous ayez plusieurs magasins que vous gérez à travers le monde
- Un très grand livre de comptes contenant TOUTES les ventes
- Objectif : Calculer le total des ventes par magasin pour l'année en cours
- Supposons que les lignes du livres aient la forme suivante:
 - Jour Ville produit Prix



2012--01--	London	Clothes	25.99
-01			
2012--01--	Miami	Music	12.15
-01			
2012--01--	NYC	Toys	3.10
-02			
2012--01--	Miami	Clothes	50.00
-02			

Map-Reduce_Exemple

- Possibilité :
 - Pour chaque entrée, saisir la ville et le prix de vente
 - Si on trouve une entrée avec une ville déjà saisie, on les regroupe en faisant la somme des ventes
- Dans un environnement de calcul traditionnel, on utilise généralement des *Hashtables*, sous forme de:
 - Clef Valeur
- Dans notre cas, la clef serait l'adresse du magasin, et la valeur le total des ventes.



2012-01-01	London	Clothes	25.99
2012-01-01	Miami	Music	12.15
2012-01-02	NYC	Toys	3.10
2012-01-02	Miami	Clothes	50.00



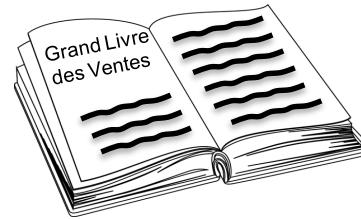
Londo	25.9
n	9
Miami	12.1
NYC	5



Londo	25.9
n	9
Miami	62.1
NYC	5

Map-Reduce Exemple

- Si on utilise les hashtable sur 1To, Problèmes ?
 - ❑ Ça ne marchera pas ?
 - ❑ Problème de mémoire ?
 - ❑ Temps de traitement long ?
 - ❑ Réponses erronées ?
- Le traitement séquentiel de toutes les données peut s'avérer très long
- Plus on a de magasins, plus l'ajout des valeurs à la table est long
- Il est possible de tomber à court de mémoire pour enregistrer cette table
- Mais cela peut marcher, et le résultat sera correct



2012-01--	London	Clothes	25.99
-01			
2012-01--	Miami	Music	12.15
-01			
2012-01--	NYC	Toys	3.10
-02			
2012-01--	Miami	Clothes	50.00
-02			

Clef	Valeur
London	25.99
Miami	62.15
NYC	3.10

Map-Reduce Exemple

- Map-Reduce : Moyen plus efficace et rapide de traiter ces données
- Au lieu d'avoir une seule personne qui parcourt le livre, si on en recrutait plusieurs?
- Appeler un premier groupe les *Mappers* et un autre les *Reducers*
- Diviser le livre en plusieurs parties, et en donner une à chaque *Mapper*
 - Les Mappers peuvent travailler en même temps, chacun sur une partie des données

Mappers



Reducers



Map-Reduce Exemple

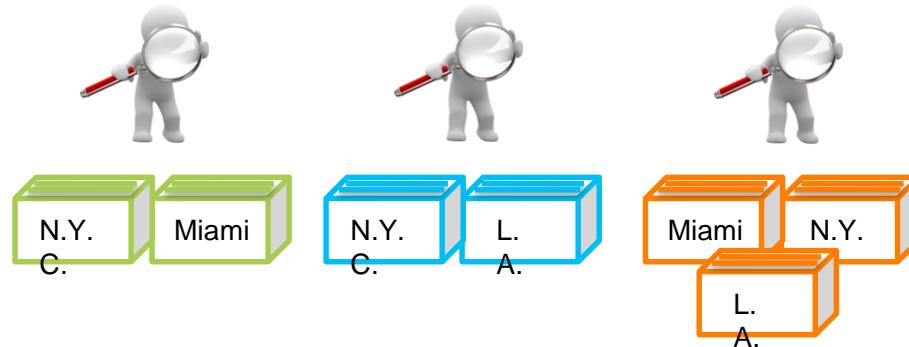
Mappers

- Pour chaque entrée, saisir la ville, et le total des ventes et les enregistrer dans une fiche
- Rassembler les fiches du même magasin dans une même pile

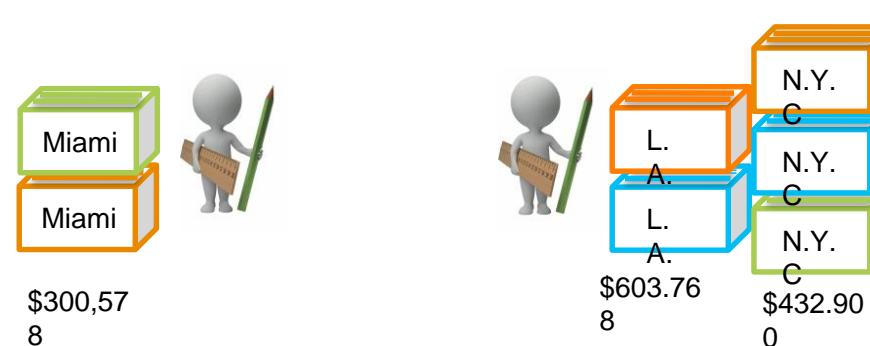
Reducers

- Chaque Reducer sera responsable d'un ensemble de magasins
- Ils collectent les fiches qui leur sont associées des différents Mappers
- Ils regroupent les petites piles d'une même ville en une seule
- Ils parcourrent ensuite chaque pile par ordre alphabétique des villes (L.A avant Miami), et font la somme de l'ensemble des enregistrements

Mappers



Reducers



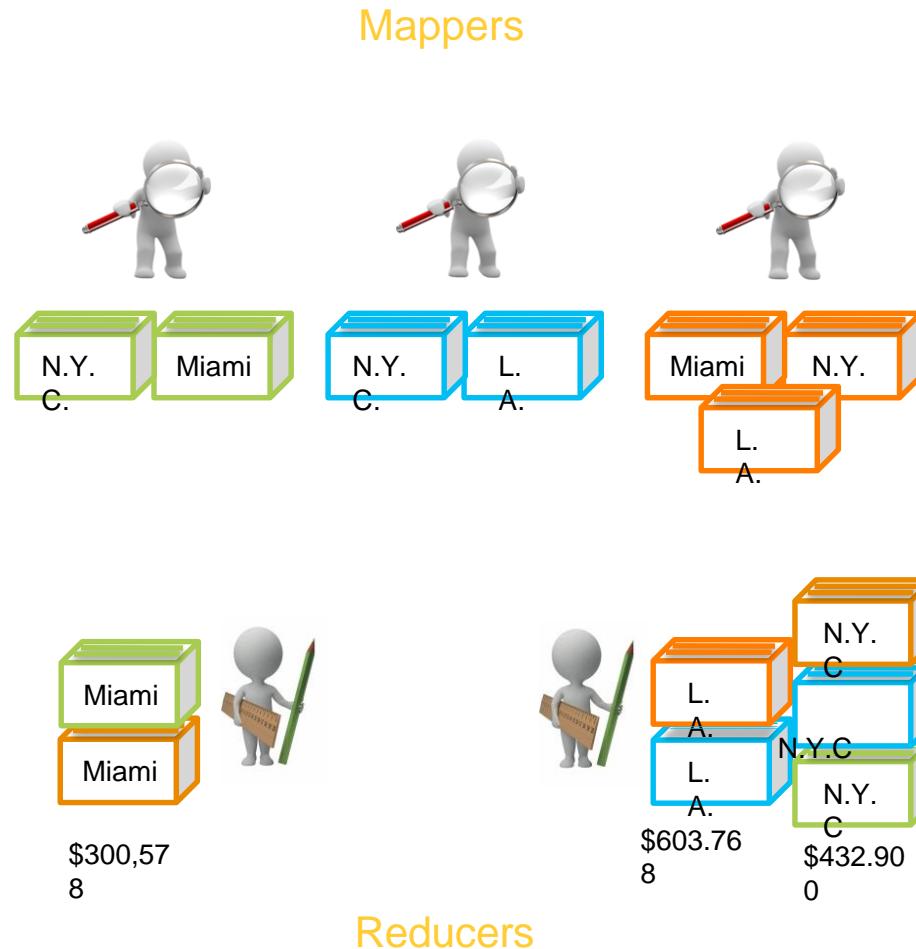
Map-Reduce Exemple

- Le Reducer reçoit des données comme suit :

○ Miami	12.34
○ Miami	99.07
○ Miami	3.14
○ NYC	99.77
○ NYC	88.99

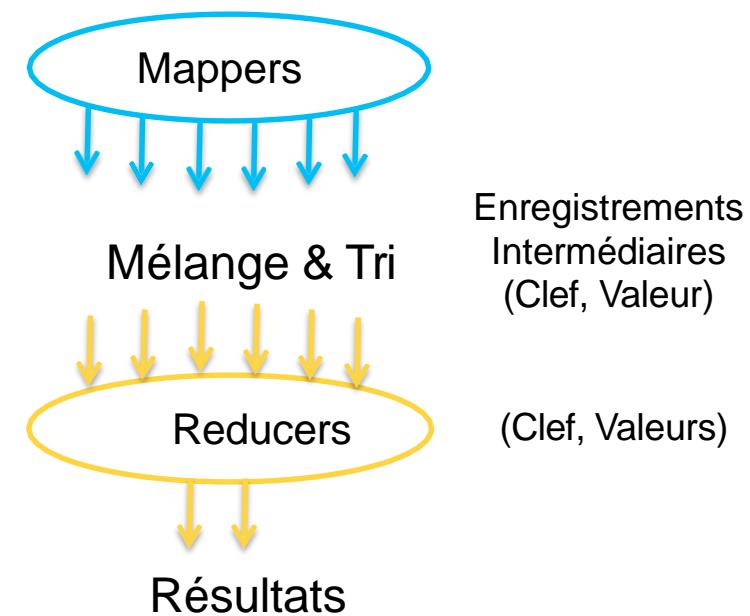
- Pour chaque entrée, de quoi avons-nous besoin pour calculer la totalité des ventes pour chaque magasin?

- Coût précédent
- Coût en cours
- Ventes totales par magasin Magasin précédent
- Magasin en cours



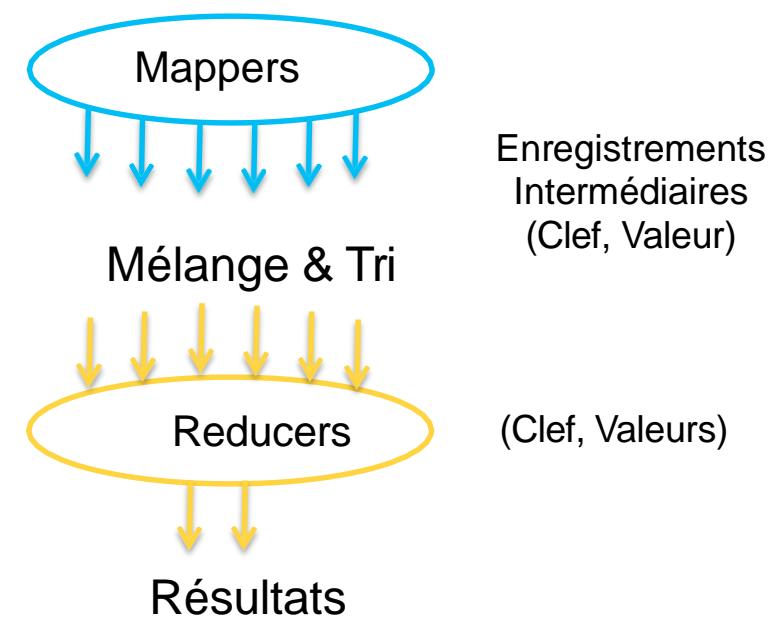
Map-Reduce Fonctionnement

- Les Mappers sont de petits programmes qui commencent par traiter chacun une petite partie des données
- Ils fonctionnent en parallèle
- Leurs sorties représentent les *enregistrements intermédiaires*: sous forme d'un couple (*clef, valeur*)
- Une étape de *Mélange et Tri* s'ensuit
 - *Mélange* : Sélection des piles de fiches à partir des Mappers
 - *Tri* : Rangement des piles par ordre au niveau de chaque Reducer
- Chaque Reducer traite un ensemble d'enregistrements à la fois, pour générer les résultats finaux



Map-Reduce Résultats

- Pour avoir un résultat trié par ordre, on doit:
 - Soit avoir un seul Reducer, mais ça ne se met pas bien à l'échelle
 - Soit ajouter une autre étape permettant de faire le tri final
- Si on a plusieurs Reducers, on ne peut pas savoir lesquels traitent quelles clefs: le partitionnement est aléatoire.



MapReduce : Exemple concret (1/8)

- **Imaginons qu'on nous donne un texte écrit en langue Française. On souhaite déterminer pour un travail de recherche quels sont les mots les plus utilisés au sein de ce texte (exemple Hadoop très répandu).**
- **Ici, nos données d'entrée sont constituées du contenu du texte.**
- **Première étape: déterminer une manière de découper (split) les données d'entrée pour que chacune des machines puisse travailler sur une partie du texte.**
- **Notre problème est ici très simple – on peut par exemple décider de découper les données d'entrée ligne par ligne. Chacune des lignes du texte sera un fragment de nos données d'entrée.**

MapReduce : Exemple concret (2/8)

- Nos données d'entrée (le texte):

Celui qui croyait au
ciel Celui qui n'y
croyait pas [...]
Fou qui fait le délicat
Fou qui songe à ses querelles

(Louis Aragon, *La rose et le Réséda*, 1943, fragment)

- Pour simplifier les choses, on va avant le découpage supprimer toute ponctuation et tous les caractères accentués. On va également passer l'intégralité du texte en minuscules.

MapReduce : Exemple concret (3/8)

- **Nos données d'entrée (le texte):**

celui qui croyait au ciel

celui qui ny croyait pas

fou qui fait le delicat

fou qui songe a ses querelles

- **... on obtient 4 fragments depuis nos données d'entrée.**

MapReduce : Exemple concret (4/8)

- On doit désormais déterminer la clef à utiliser pour notre opération MAP, et écrire le code de l'opération MAP elle-même.
- Puisqu'on s'intéresse aux occurrences des mots dans le texte, et qu'à terme on aura après l'opération REDUCE un résultat pour chacune des clefs distinctes, la clef qui s'impose logiquement dans notre cas est: le mot-lui même.
- Quand à notre opération MAP, elle sera elle aussi très simple: on va simplement parcourir le fragment qui nous est fourni et, pour chacun des mots, générer le couple clef/valeur: (MOT ; 1). La valeur indique ici l'occurrence pour cette clef - puisqu'on a croisé le mot une fois, on donne la valeur « 1 ».

MapReduce : Exemple concret (5/8)

- Le code de notre opération MAP sera donc (ici en pseudo code):

```
POUR MOT dans LIGNE,  
FAIRE : GENERER  
— COUPLE (MOT; 1)
```

- Pour chacun de nos fragments, les couples (clef; valeur) générés seront donc:

celui qui croyait au ciel



(celui;1) (qui;1) (croyait;1) (au;1)
(ciel;1)

celui qui ny croyait pas



(celui;1) (qui;1) (ny;1) (croyait;1)
(pas;1)

fou qui fait le delicat



(fou;1) (qui;1) (fait;1) (le;1) (delicat;1)

fou qui songe a ses
querelles



(fou;1) (qui;1) (songe;1) (a;1)
(ses;1) (querelles;1)

MapReduce : Exemple concret (6/8)

- Une fois notre opération MAP effectuée (de manière distribuée), Hadoop groupera (shuffle) tous les couples par clef commune.
- Cette opération est effectuée automatiquement par Hadoop. Elle est, là aussi, effectuée de manière distribuée en utilisant un algorithme de tri distribué, de manière récursive. Après son exécution, on obtiendra les 15 groupes suivants:

(celui;1) (celui;1)

(qui;1) (qui;1) (qui;1) (qui;1)

(croyait;1) (croyait;1)

(au;1)

(ciel;1)

(ny;1)

(pas;1)

(fou;1) (fou;1)

(delicat;1)

(fait;1)

(songe;1)

(a;1)

(ses;1)

(querelles;1)

MapReduce : Exemple concret (7/8)

- Il nous reste à créer notre opération **REDUCE**, qui sera appelée pour chacun des groupes/clef distincte.
- Dans notre cas, elle va simplement consister à additionner toutes les valeurs liées à la clef spécifiée:

```
TOTAL=0
POUR COUPLE dans GROUPE, FAIRE:
    TOTAL=TOTAL+Valeur
    ENVOYER TOTAL
```

MapReduce : Exemple concret (8/8)

- Une fois l'opération REDUCE effectuée, on obtiendra donc une valeur unique pour chaque clef distincte. En l'occurrence, notre résultat sera:

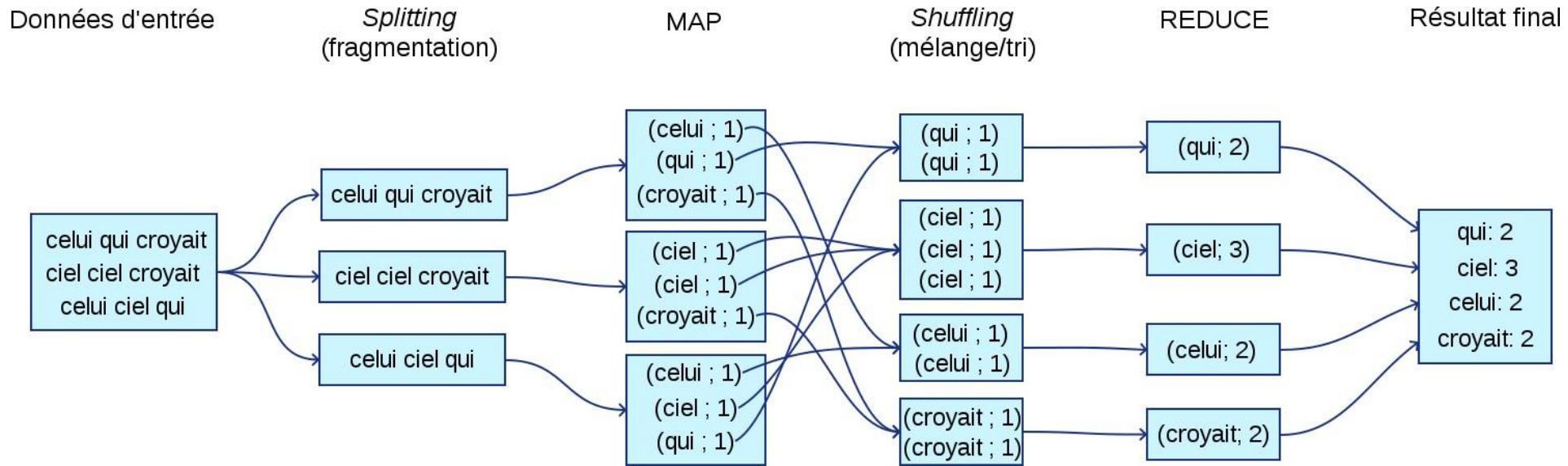
```
qui : 4
celui : 2
croyait : 2
fou : 2
au : 1
ciel : 1
ny : 1
pas : 1
fait : 1
[...]
```

- On constate que le mot le plus utilisé dans notre texte est « qui », avec 4 occurrences, suivi de « celui », « croyait » et « fou », avec 2 occurrences chacun.

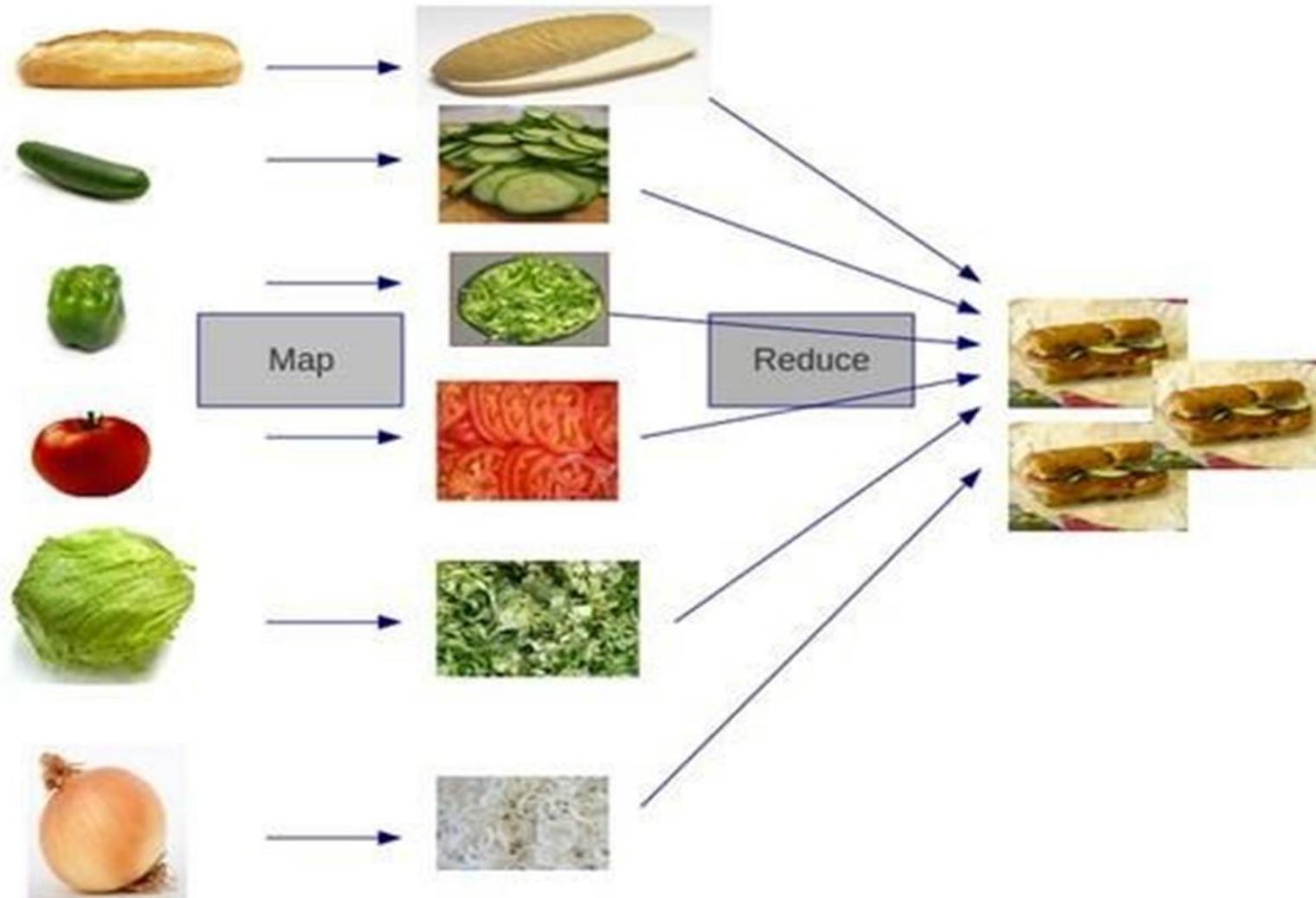
MapReduce : Exemple concret - Conclusion

- Notre exemple est évidemment trivial, et son exécution aurait été instantanée même sur une machine unique, mais il est d'ores et déjà utile: on pourrait tout à fait utiliser les mêmes implémentations de MAP et REDUCE sur l'intégralité des textes d'une bibliothèque Française, et obtenir ainsi un bon échantillon des mots les plus utilisés dans la langue Française.
- L'intérêt du modèle MapReduce est qu'il nous suffit de développer les deux opérations réellement importantes du traitement: MAP et REDUCE, et de bénéficier automatiquement de la possibilité d'effectuer le traitement sur un nombre variable de machines de manière distribuée.

MapReduce : Schéma général



MapReduce



MapReduce : Exemple – Statistiques web

- Un autre exemple: on souhaite compter le nombre de visiteurs sur chacune des pages d'un site Internet. On dispose des fichiers de logs sous la forme suivante:

```
/index.html [19/05/2017:18:45:03 +0200]  
/contact.html [19/05/2017:18:46:15 +0200]  
/news.php?id=5 [24/05/2017:18:13:02 +0200]  
/news.php?id=4 [24/05/2017:18:13:12 +0200]  
/news.php?id=18 [24/05/2017:18:14:31 +0200]  
...etc...
```

- Ici, notre clef sera par exemple l'URL d'accès à la page, et nos opérations MAP et REDUCE seront exactement les mêmes que celles qui viennent d'être présentées: on obtiendra ainsi le nombre de vue pour chaque page distincte du site.

MapReduce : Exemple – Graphe social (1/8)

- Un autre exemple: on administre un réseau social comportant des millions d'utilisateurs.
- Pour chaque utilisateur, on a dans notre base de données la liste des utilisateurs qui sont ses amis sur le réseau (via une requête SQL).
- On souhaite afficher quand un utilisateur va sur la page d'un autre utilisateur une indication « Vous avez N amis en commun ».
- On ne peut pas se permettre d'effectuer une série de requêtes SQL à chaque fois que la page est accédée (trop lourd en traitement). On va donc développer des programmes MAP et REDUCE pour cette opération et exécuter le traitement toutes les nuits sur notre base de données, en stockant le résultat dans une nouvelle table.

MapReduce : Exemple – Graphe social (2/8)

- Ici, nos données d'entrée sous la forme Utilisateur =>Amis:

A => B, C, D

B => A, C,

D, E C =>

A, B, D, E D

=> A, B, C,

E E => B,

C, D

- Puisqu'on est intéressé par l'information « amis en commun entre deux utilisateurs » et qu'on aura à terme une valeur par clef, on va choisir pour clef la concaténation entre deux utilisateurs. Par exemple, la clef « A-B » désignera « les amis en communs des utilisateurs A et B».

- On peut segmenter les données d'entrée là aussi par ligne.

MapReduce : Exemple – Graphe social (3/8)

- Notre opération MAP va se contenter de prendre la liste des amis fournie en entrée, et va générer toutes les clefs distinctes possibles à partir de cette liste. La valeur sera simplement la liste d'amis, telle quelle.
- On fait également en sorte que la clef soit toujours triée par ordre alphabétique (clef « B-A » sera exprimée sous la forme « A-B »).
- Ce traitement peut paraître contre-intuitif, mais il va à terme nous permettre d'obtenir, pour chaque clef distincte, deux couples (clef;valeur): les deux listes d'amis de chacun des utilisateurs qui composent la clef.

MapReduce : Exemple – Graphe social (4/8)

- Le pseudo code de notre opération MAP:

```
UTILISATEUR = [PREMIERE PARTIE DE LA LIGNE]
POUR AMI dans [RESTE DE LA
LIGNE], FAIRE: SI UTILISATEUR <
AMI:
    CLEF = UTILISATEUR+"-"+AMI
SINON:
    CLEF = AMI+"-"+UTILISATEUR
GENERER COUPLE (CLEF; [RESTE DE LA LIGNE])
```

- Par exemple, pour la première ligne: A => B, C, D

- On obtiendra les couples (clef;valeur):

("A-B"; "B C D")

("A-C"; "B C D")

("A-D"; "B C D")

MapReduce : Exemple – Graphe social (5/8)

- Pour la seconde ligne :

$B \Rightarrow A, C, D, E$

- On obtiendra ainsi :

("A-B"; "A C D
E")

("B-C"; "A C D
E")

("B-D"; "A C D
E")

("B-E"; " A C D
E")

- Pour la troisième ligne :

$C \Rightarrow A, B, D, E$

- On aura :

("A-C"; "A B D
E")

("B-C"; "A B D
E")

("C-D"; "A B D
E")

("C-E"; " A B D
E")

- ...et ainsi de suite pour nos 5 lignes d'entrée.

MapReduce : Exemple – Graphe social (6/8)

- Une fois l'opération MAP effectuée, Hadoop va récupérer les couples (clef;valeur) de tous les fragments et les grouper par clef distincte. Le résultat sur la base de nos données d'entrée :

Pour la clef "A-B" : valeurs "A C D E" et "B C D"
Pour la clef "A-C" : valeurs "A B D E" et "B C D"
Pour la clef "A-D" : valeurs "A B C E" et "B C D"
Pour la clef "B-C" : valeurs "A B D E" et "A C D E"
Pour la clef "B-D" : valeurs "A B C E" et "A C D E"
Pour la clef "B-E" : valeurs "A C D E" et "B C D"
Pour la clef "C-D" : valeurs "A B C E" et "A B D E"
Pour la clef "C-E" : valeurs "A B D E" et "B C D"
Pour la clef "D-E" : valeurs "A B C E" et "B C D"

- ... on obtient bien, pour chaque clef « USER1-USER2 », deux listes d'amis: les amis de USER1 et ceux de USER2.

MapReduce : Exemple – Graphe social (7/8)

- Il nous faut enfin écrire notre programme **REDUCE**. Il va recevoir en entrée toutes les valeurs associées à une clef. Son rôle va être très simple: déterminer quels sont les amis qui apparaissent dans les listes (les valeurs) qui nous sont fournies. Pseudo-code:

```
LISTE_AMIS_COMMUNS=[] // Liste vide au départ.  
SI LONGUEUR(VALEURS)!=2, ALORS: // Ne devrait pas se produire.  
    RENVOYER  
ERREUR SINON :  
    POUR AMI DANS VALEURS[0], FAIRE :  
        SI AMI EGALEMENT PRESENT DANS VALEURS[1], ALORS :  
            // Présent dans les deux listes d'amis, on l'ajoute.  
            LISTE_AMIS_COMMUNS+=AMI  
    RENVOYER LISTE_AMIS_COMMUNS
```

MapReduce : Exemple – Graphe social (8/8)

- Après exécution de l'opération REDUCE pour les valeurs de chaque clef unique, on obtiendra donc, pour une clef « A-B », les utilisateurs qui apparaissent dans la liste des amis de A et dans la liste des amis de B. Autrement dit, on obtiendra la liste des amis en commun des utilisateurs A et B. Le résultat:

```
"A-B" : "C, D"  
"A-C" : "B, D"  
"A-D" : "B, C"  
"B-C" : "A, D, E"  
"B-D" : "A, C, E"  
"B-E" : "C, D"  
"C-D" : "A, B, E"  
"C-E" : "B, D"  
"D-E" : "B, C"
```

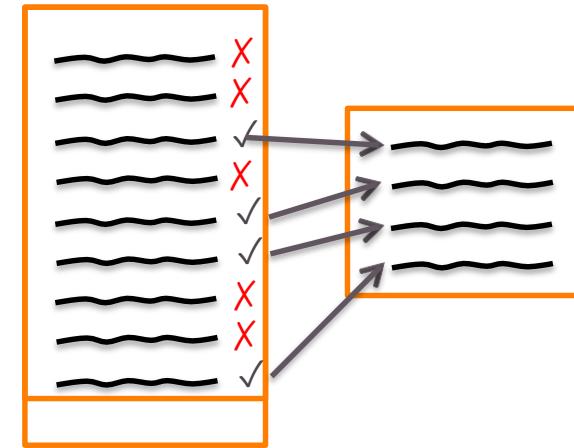
- On sait ainsi que A et B ont pour amis communs les utilisateurs C et D, ou encore que B et C ont pour amis communs les utilisateurs A, D et E.

Map-Reduce Design Patterns

- Les designs patterns (Patrons de Conception) représentent les types de traitements Map-Reduce les plus utilisés avec Hadoop
- Classés en trois catégories:
 - Patrons de Filtrage (*Filtering Patterns*)
 - Echantillonnage de données
 - Listes des top-n
 - Patrons de Récapitulation (*Summarization Patterns*)
 - Comptage des enregistrements
 - Trouver les min et les max
 - Statistiques
 - Indexes
 - Patrons Structurels
 - Combinaison de données relationnelles

Patrons de Filtrage

- Ne modifient pas les données
- Trient, parmi les données présentes, lesquelles garder et lesquelles enlever
- On peut obtenir:
 - Des filtres simples : définition d'une fonction indiquant le critère de filtrage
 - L'échantillonnage (*sampling*): création d'un petit ensemble d'enregistrements à partir d'un grand ensemble, en retenant des échantillons (comme la valeur la plus élevée d'un champs particulier)
 - L'échantillonnage aléatoire : retenir un échantillon représentatif des données initiales
 - Les listes Top-n



Patrons de Filtrage

- Exemple de Filtrage Simple:
 - Cas d'étude : fichier contenant tous les posts des utilisateurs sur un forum
 - Filtre : Retenir les posts les plus courts, contenant une seule phrase
 - Une phrase est un post qui ne contient aucune ponctuation de la forme: .!?, ou alors une seule à la fin.

```
def mapper():
    reader = csv.reader(sys.stdin, delimiter='\t')
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"',
                        quoting=csv.QUOTE_ALL)

    for line in reader:

        for i in line:
            #print('-',i)
            if len(i) == 0:
                continue
            if "!" in i[:-1]:
                continue
            if "." in i[:-1]:
                continue
            if "?" in i[:-1]:
                continue
            else:
                writer.writerow(line)
```

Patrons de Filtrage

- Exemple: Top 10
 - Trouver parmi les différents posts des forums, les 10 posts les plus longs
 - Dans une Base de données Relationnelle:
 - Trier les données
 - Extraire les 10 premières
 - Map-Reduce (de la même manière qu'une sélection sportive)
 - Chaque Mapper génère une liste Top-10
 - Le Reducer trouve les Top 10 globaux

```
def mapper():  
    a = []  
    b = []  
    reader = csv.reader(sys.stdin, delimiter='\t')  
    writer = csv.writer(sys.stdout, delimiter='\t', quotechar='',  
                        quoting=csv.QUOTE_ALL)  
    for line in reader:  
  
        for i in line:  
            if len(i) == 0 :  
                continue  
            else:  
                a.append(line)  
  
    a.sort(key=lambda a: (int)(a[4]), reverse=True)  
  
    for i in range(0,10):  
        b.append(a[i])  
    b.sort(key=lambda b: (int)(b[4]))  
  
    for b1 in b:  
        writer.writerow(b1)
```

Patrons de Récapitulation

- Permettent de vous donner une idée de haut niveau de vos données
- On distingue deux types:
 - Index : Tels que les index à la fin d'un livre, ou les index utilisés par google pour représenter les pages web
 - Récapitulation (ou résumé) numérique : par exemple:
 - Chercher des chiffres, des comptes (combien dispose-t-on d'un certain type d'entrées)
 - Min et Max
 - Premier et dernier
 - Moyenne
 - ...

Patrons de Récapitulation Index

- Les index permettent une recherche plus rapide
- Dans un livre: pour chaque mot donné, indiquer les différentes pages où se trouve ce mot
- Dans le web: on trouve des liens vers des pages web à partir d'un ensemble de mots clefs

Patrons de Récapitulation Index

- *Exemple : Indexation des mots dans les posts d'un forum*
 - Mapper ➡

```
import sys
import csv
import re

firstLine = 1

reader = csv.reader(sys.stdin, delimiter='\t')
writer = csv.writer(sys.stdout, delimiter='\t', quotechar='"', quoting=csv.QUOTE_ALL
)

for line in reader:
    if firstLine == 1:
        #Si on se trouve dans la premiere ligne (celle des titres), sauter c
ette ligne
        firstLine = 0
        continue
    body = line[4]
    node = line[0]
    words = re.findall(r"[\w']+|[.,!?;]",body)
    for word in words:
        if word not in ('.', '!', ',', '?', '#', '$', '[', ']', '/', '\'', '<', '>', '='
, '- ', ': ', '; ', '(' , ') '):
            print "{0}\t{1}".format(word,node)
```

Patrons de Récapitulation Index

- *Exemple :*
Indexation des mots dans les posts d'un forum

- Reducer ➡

```
import sys

nbTotal = 0
oldWord = None
listNodes = []

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        continue

    thisWord, thisNode = data_mapped

    if oldWord and oldWord.lower() != thisWord.lower():
        listNodes.sort(key=lambda listNodes:(int)(listNodes))
        print oldWord, "\t", nbTotal, "\t", listNodes
        oldWord = thisWord.lower();
        nbTotal = 0
        listNodes = []

    oldWord = thisWord.lower()
    nbTotal = nbTotal + 1
    if thisNode not in listNodes:
        listNodes.append(thisNode)

if oldWord != None:
    listNodes.sort(key=lambda listNodes:(int)(listNodes))
```

Patrons de Récapitulations Numériques

- Peuvent être:
 - Le nombre de mots, enregistrements...
 - Souvent: la clef = l'objet à compter, et la valeur = 1 (nombre d'occurrences)
 - Min-Max / Premier-Dernier
 - La moyenne
 - La médiane
 - Écart type
 - ...
- Exemple de question:
 - Y'a-t-il une relation entre le jour de la semaine et la somme dépensée par les clients?
 - Mapper : clef = jour_de_la_semaine; valeur = somme_dépensée
 - Reducer : calcul

Patrons de Récapitulation Mélangeur

- Possibilité d'utiliser un mélangeur (*Combiner*) entre les mappers et les reducers
- Permettent de réaliser la réduction en local dans chacun des DataNodes Mappers AVANT de faire appel au nœud réducteur principal.
- Exemple: pour calculer la moyenne des ventes par jour de la semaine:
 - Sans Combiner
 - Les Mappers parcouruent les données, et affichent le couple (*Jour_de_la_semaine, montant*)
 - Pour chaque jour, le Reducer conserve une somme et un compteur
 - Il divise à la fin la somme par le compteur
 - Avec Combiner
 - Chaque nœud réalise une première réduction où les moyennes locales sont calculées
 - Le Reducer final regroupe ces moyennes et synthétise la moyenne finale
 - Nombre d'enregistrements envoyés au réducteur significativement réduit
 - Temps nécessaire pour la réduction diminue

MapReduce : Conclusion

- En utilisant le modèle MapReduce, on a ainsi pu créer deux programmes très simples (nos programmes MAP et REDUCE) de quelques lignes de code seulement, qui permettent d'effectuer un traitement somme toute assez complexe.
- Mieux encore, notre traitement est parallélisable: même avec des dizaines de millions d'utilisateurs, du moment qu'on a assez de machines au sein du cluster Hadoop, le traitement sera effectué rapidement. Pour aller plus vite, il nous suffit de rajouter plus de machines.
- Pour notre réseau social, il suffira d'effectuer ce traitement toutes les nuits à heure fixe, et de stocker les résultats dans une table. Ainsi, lorsqu'un utilisateur visitera la page d'un autre utilisateur, un seul SELECT dans la base de données suffira pour obtenir la liste des amis en commun – avec un poids en traitement très faible pour le serveur.

Architecture Hadoop: Présentation (1/3)

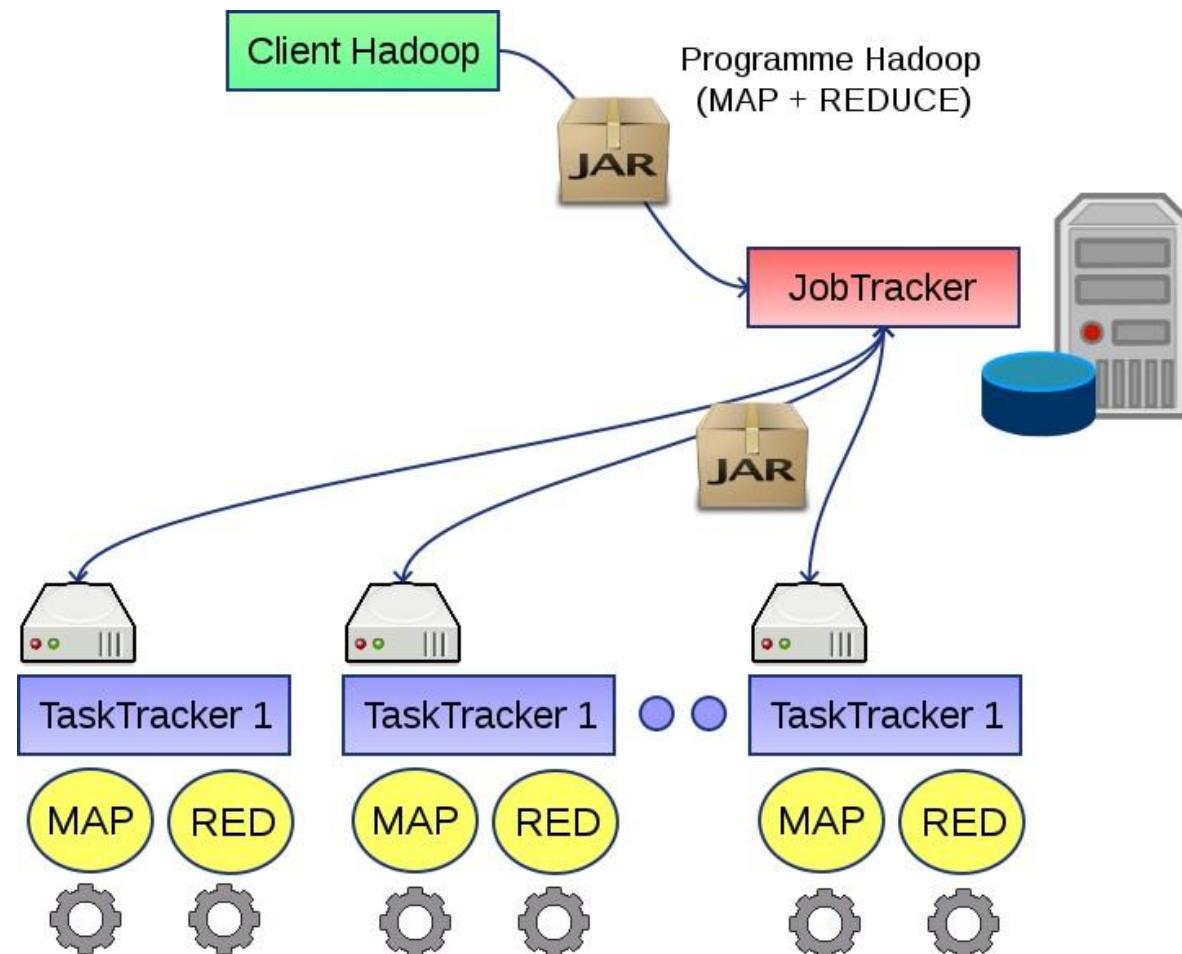
Comme pour HDFS, la gestion des tâches de Hadoop se base sur deux serveurs (des daemons):

- Le **JobTracker**, qui va directement recevoir **la tâche à exécuter** (un .jar Java), ainsi que les **données d'entrées** (nom des fichiers stockés sur HDFS) et **le répertoire où stocker les données de sortie** (toujours sur HDFS). Il y a un **seul JobTracker** sur une seule machine du cluster Hadoop. **Le JobTracker est en communication avec le NameNode de HDFS et sait donc où sont les données.**
- Le **TaskTracker**, qui est en communication constante avec le JobTracker et va recevoir les **opérations simples à effectuer (MAP/REDUCE)** ainsi que les **blocs de données correspondants** (stockés sur HDFS). Il y a un **TaskTracker sur chaque machine du cluster.**

Architecture Hadoop: Présentation (2/3)

- Comme le JobTracker est conscient de la position des données (grâce au NameNode), il peut facilement déterminer les meilleures machines auxquelles attribuer les sous-tâches (celles où les blocs de données correspondants sont stockés).
- Pour effectuer un traitement Hadoop, on va donc stocker nos données d'entrée sur HDFS, créer un répertoire où Hadoop stockera les résultats sur HDFS, et compiler nos programmes MAP et REDUCE au sein d'un .jar Java.
- On soumettra alors le nom des fichiers d'entrée, le nom du répertoire des résultats, et le .jar lui-même au JobTracker: il s'occupera du reste (et notamment de transmettre les programmes MAP et REDUCE aux serveurs TaskTracker des machines du cluster).

Architecture Hadoop: Présentation (3/3)



Architecture Hadoop: Le JobTracker (1/3)

Le déroulement de l'exécution d'une tâche Hadoop suit les étapes suivantes du point de vue du JobTracker :

1. Le client (un outil Hadoop console) va soumettre le travail à effectuer au JobTracker : une archive java .jar implémentant les opérations Map et Reduce. Il va également soumettre le nom des fichiers d'entrée et l'endroit où stocker les résultats.
2. Le JobTracker communique avec le NameNode HDFS pour savoir où se trouvent les blocs correspondant aux noms de fichiers donnés par le client.
3. Le JobTracker, à partir de ces informations, détermine quels sont les noeuds TaskTracker les plus appropriés, c'est à dire ceux qui contiennent les données sur lesquelles travailler sur la même machine, ou le plus proche possible (même rack/rack proche).

Architecture Hadoop: Le JobTracker (2/3)

5. Pour chaque « morceau » des données d'entrée, le JobTracker envoie au TaskTracker sélectionné le travail à effectuer (MAP/REDUCE, code Java) et les blocs de données correspondants.
6. Le JobTracker communique avec les noeuds TaskTracker en train d'exécuter les tâches. Ils envoient régulièrement un « **heartbeat** », un message signalant qu'ils travaillent toujours sur la sous-tâche reçue. Si aucun heartbeat n'est reçu dans une période donnée, le JobTracker considère la tâche comme ayant échouée et donne le même travail à effectuer à un autre TaskTracker.
7. Si par hasard une tâche échoue (erreur java, données incorrectes, etc.), le TaskTracker va signaler au JobTracker que la tâche n'a pas pu être exécutée.

Le JobTracker va alors décider de la conduite à adopter :

- Demander au même TaskTracker de ré-essayer.
- Redonner la sous-tâche à un autre TaskTracker.
- Marquer les données concernées comme invalides, etc.

Il pourra même **blacklister** le TaskTracker concerné comme **non-fiable** dans certains cas

Architecture Hadoop: Le JobTracker (3/3)

7. Une fois que toutes les opérations envoyées aux TaskTracker (MAP + REDUCE) ont été effectuées et confirmées comme effectuées par tous les noeuds, le JobTracker marque la tâche comme « effectuée ». Des informations détaillées sont disponibles (statistiques, TaskTracker ayant posé problème, etc.).

Remarques

- Par ailleurs, on peut également obtenir à tout moment de la part du JobTracker des informations sur les tâches en train d'être effectuées: étape actuelle (MAP, SHUFFLE, REDUCE), pourcentage de complétion, etc.
- La soumission du .jar, l'obtention de ces informations, et d'une manière générale toutes les opérations liées à Hadoop s'effectuent avec le même unique client console vu précédemment: **hadoop** (avec d'autres options que l'option **fs** vu précédemment).

Architecture Hadoop: Le TaskTracker

- **Le TaskTracker dispose d'un nombre de « slots » d'exécution. A chaque « slot » correspond une tâche exécutable** (configurable). Ainsi, une machine ayant par exemple un processeur à 8 cœurs pourrait avoir 16 slots d'opérations configurées.
- Lorsqu'il reçoit une nouvelle tâche à effectuer (MAP, REDUCE, SHUFFLE) depuis le JobTracker, le **TaskTracker va démarrer une nouvelle instance de Java avec le fichier .jar** fourni par le JobTracker, en appelant l'opération correspondante.
- Une fois la tâche démarrée, il enverra **régulièrement** au JobTracker ses **messages heartbeats**. En dehors d'informer le JobTracker qu'il est toujours fonctionnels, ces messages indiquent également le nombre de slots disponibles sur le TaskTracker concerné.
- Lorsqu'une sous-tâche est terminée, le **TaskTracker envoie un message au JobTracker pour l'en informer**, que la tâche se soit bien déroulée ou non (il indique évidemment le résultat au JobTracker).

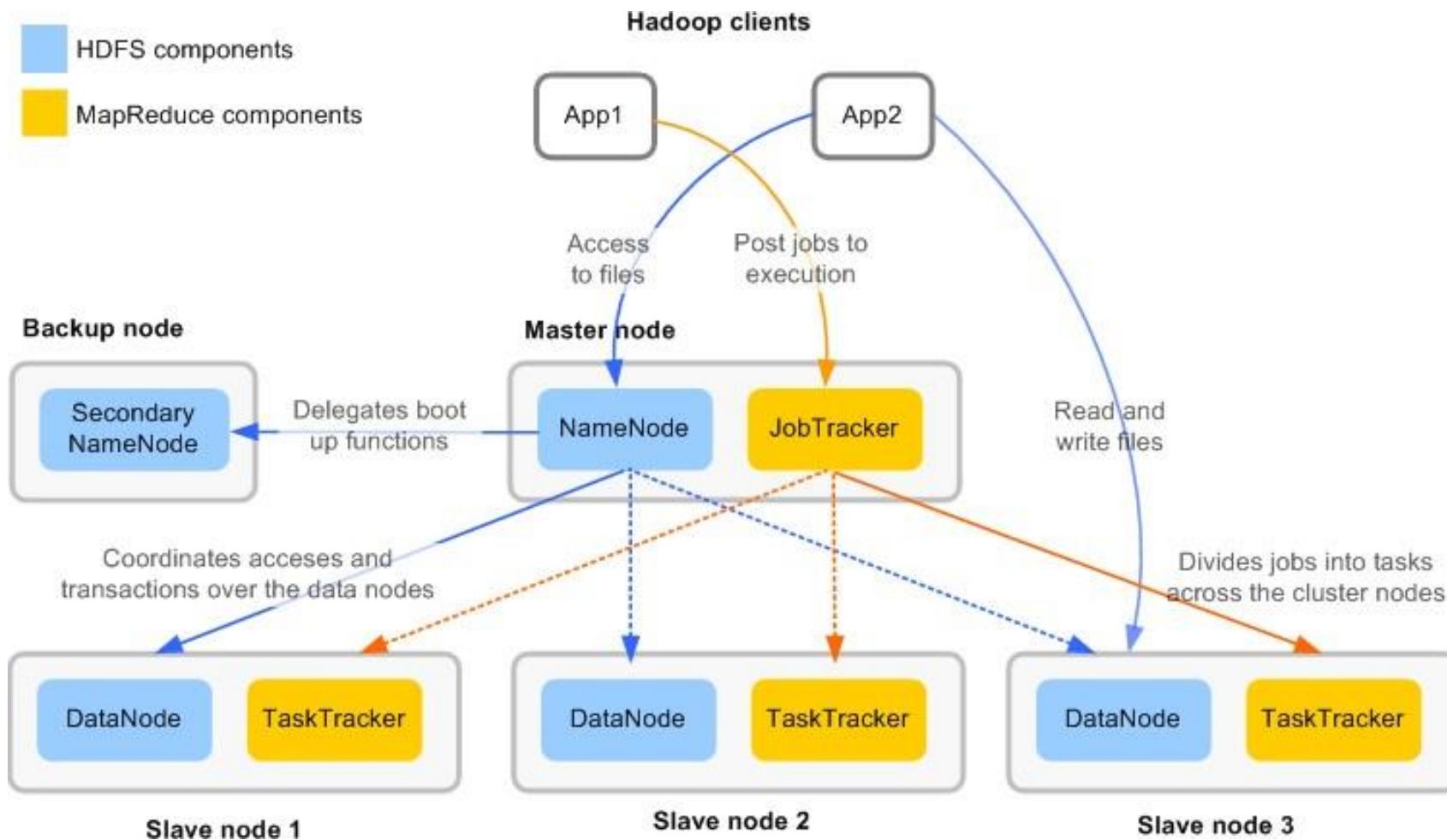
Architecture Hadoop: Remarques (1/2)

- De manière similaire au NameNode de HDFS, **il n'y a qu'un seul JobTracker et s'il tombe en panne, le cluster tout entier ne peut plus effectuer de tâches**. Là aussi, des résolutions aux problèmes sont ajoutées dans la version 2 de Hadoop (explication dans la section suivante).
- **Généralement, on place le JobTracker et le NameNode HDFS sur la même machine** (une machine plus puissante que les autres), sans y placer de TaskTracker/DataNode HDFS pour limiter la charge. Cette machine particulière au sein du cluster (qui contient les deux « gestionnaires », de tâches et de fichiers) est communément appelée le **noeud maître** (« **Master Node** »). Les autres noeuds (contenant TaskTracker + DataNode) sont communément appelés **noeuds esclaves** (« **slave node** »).

Architecture Hadoop: Remarques (2/2)

- Même si le JobTracker est situé sur une seule machine, **le « client » qui envoie la tâche au JobTracker initialement peut être exécuté sur n'importe quelle machine du cluster – comme les TaskTracker sont présents sur la machine, ils indiquent au client comment joindre le JobTracker.**
- La même remarque est valable pour l'accès au système de fichiers: les **DataNodes indiquent au client comment accéder au NameNode.**
- **Enfin, tout changement de configuration Hadoop peut s'effectuer facilement simplement en changeant la configuration sur la machine où sont situés les serveurs NameNode et JobTracker: ils répliquent les changements de configuration sur tout le cluster automatiquement.**

Architecture Hadoop: Architecture générale



YARN (MapReduce 2) : Présentation

- YARN (Yet-Another-Resource-Negotiator) est aussi appelé MRv2 (MapReduce 2). Ce n'est pas une refonte mais une évolution du framework MapReduce.
- YARN répond aux problématiques suivantes du Map Reduce :
 - Problème de limite de “Scalability” notamment par une meilleure séparation de la gestion de l'état du cluster et des ressources.
 - ~ 4000 Noeuds, 40 000 Tâches concourantes.
 - Problème d'allocation des ressources.

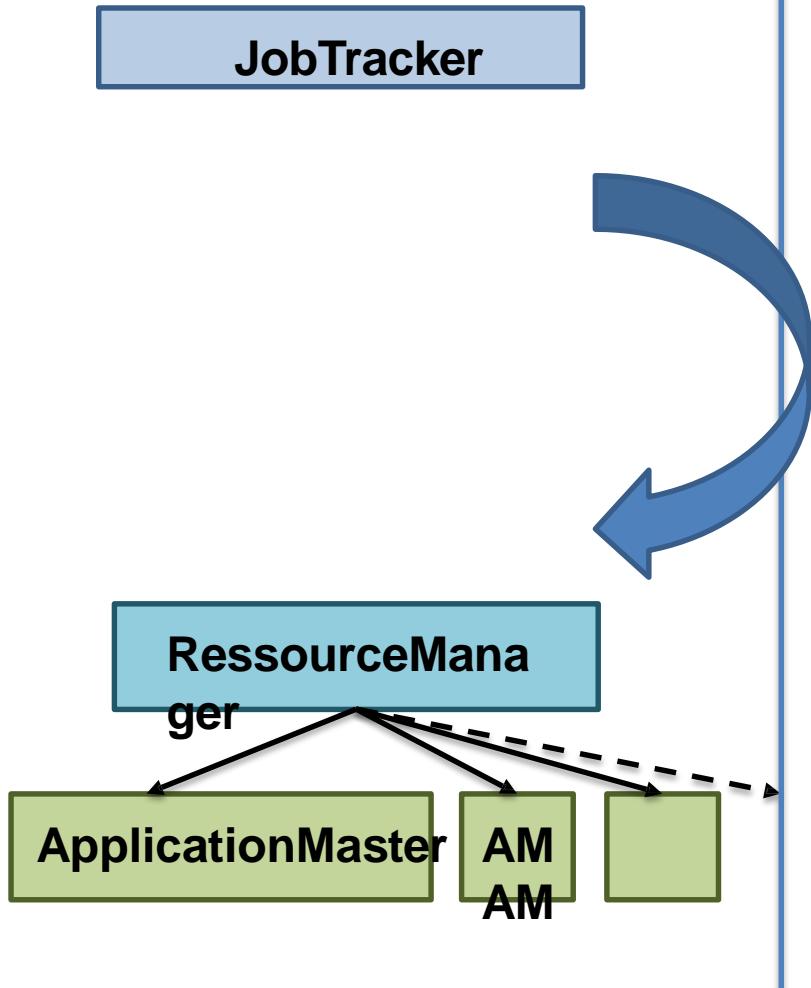
Cluster Summary (Heap Size is 12.58 GB/23.97 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
4512	285	2223	188	4512	285	0	0	4512	3008	40.00

Des tâches Map sont en attentes alors que des slots de tâche reduce sont libres.

Valeurs codées en dûr.

YARN (MapReduce 2) : Architecture (1/7)



Le JobTracker a trop de responsabilités.

- Gérer les ressources du cluster.
- Gérer tous les jobs
 - Allouer les tâches et les ordonner.
 - Monitorer l'exécution des tâches.
 - Gérer le fail-over.

Re-penser l'architecture du Job Tracker.

- Séparer la gestion des ressources du cluster de la coordination des jobs.
- Utiliser les noeuds esclaves pour gérer les jobs.

ResourceManager et ApplicationMaster.

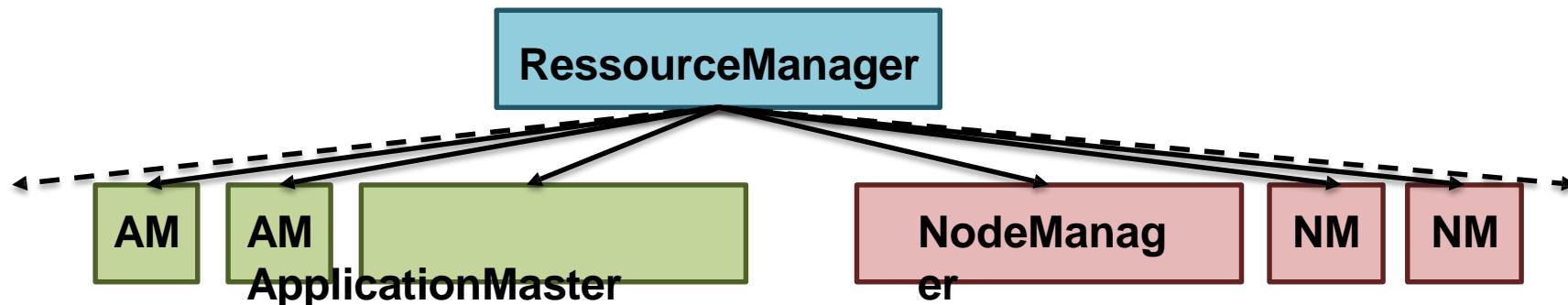
- ResourceManager remplace le JobTracker et ne gère que les ressources du Cluster.
- Une entité ApplicationMaster est allouée par Application pour gérer les tâches.
- ApplicationMaster est déployée sur les noeuds esclaves.

YARN (MapReduce 2) : Architecture (2/7)

- Cette nouvelle version contient aussi un autre composant :

Le NodeManager (NM)

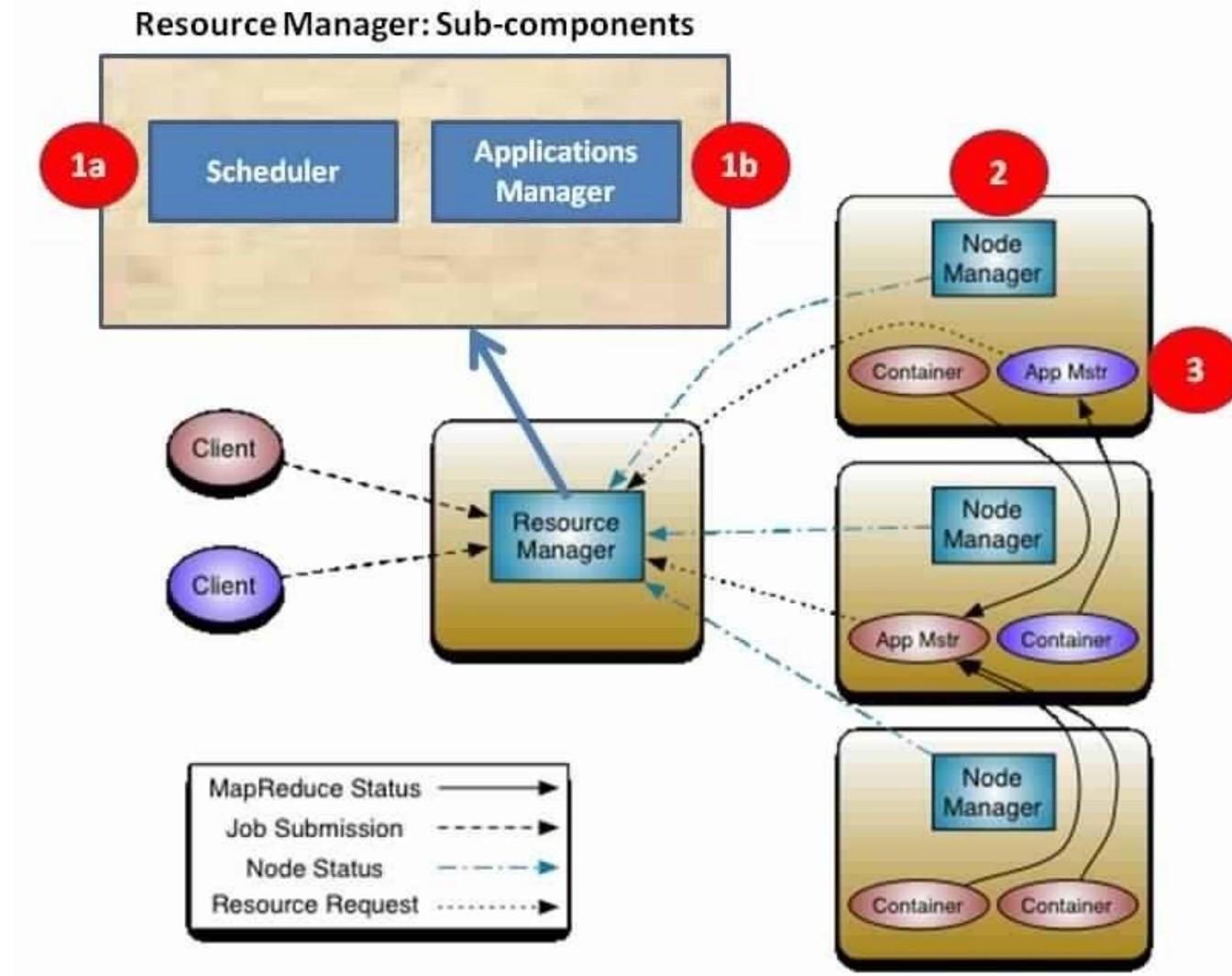
- Permet d'exécuter plus de tâches qui ont du sens pour l'Application Master, pas seulement du Map et du Reduce.
- La taille des ressources est variable (RAM, CPU, network....). Il y aura plus de valeurs codées en dur qui nécessitent un redémarrage.



YARN (MapReduce 2) : Architecture (3/7)

- **Le JobTracker a disparu de l'architecture, ou plus précisément, ses rôles ont été répartis différemment.**
- L'architecture est maintenant organisée autour d'un **ResourceManager** dont le périmètre d'action est global au cluster et à des **ApplicationMaster** locaux dont le périmètre est celui d'un job ou d'un groupe de jobs.
- **En terme de responsabilités**, on peut donc dire que :
JobTracker = ResourceManager + ApplicationMaster.
- La différence, de part le découplage, se trouve dans la multiplicité. En effet, Un ResourceManager gère n ApplicationMaster, lesquels gèrent chacun n jobs.

YARN (MapReduce 2) : Architecture (4/7)



YARN (MapReduce 2) : Architecture (5/7)

1. 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**

a. 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**.

YARN (MapReduce 2) : Architecture (6/7)

❖ Container

- Un Container désigne un regroupement de mémoire, de cpu, d'espace disque, de bande passante réseau, ...

b. ApplicationsManager

- L'ApplicationsManager accepte les soumissions d'applications.
- Une application n'étant pas gérée par le **ResourceManager**, la partie ApplicationsManager ne s'occupe que de négocier le premier **Container** que le **Scheduler** allouera sur un noeud du cluster. La particularité de ce premier **Container** est qu'il contient l'*ApplicationMaster* (*diapo suivant*).

2. NodeManager

- Les NodeManager sont des agents tournant sur chaque nœud et tenant le **Scheduler** au fait de l'évolution des ressources disponibles. Ce dernier peut ainsi prendre ses décisions d'allocation des **Containers** en prenant en compte des demandes de ressources cpu, disque, réseau, mémoire, ...

YARN (MapReduce 2) : Architecture (7/7)

3. ApplicationMaster

- L'ApplicationMaster est le composant spécifique à chaque application, il est en charge des jobs qui y sont associés.
 - Lancer et au besoin relancer des jobs
 - Négocier les Containers nécessaires auprès du Scheduler
 - Superviser l'état et la progression des jobs.
 - Un ApplicationMaster gère donc un ou plusieurs jobs tournant sur un framework donné. Dans le cas de base, c'est donc un ApplicationMaster qui lance un job MapReduce. **De ce point de vue, il remplit un rôle de TaskTracker.**
-
- ❖ **L'ApplicationsManager est l'autorité qui gère les ApplicationMaster du cluster.** A ce titre, c'est donc via l'ApplicationsManager que l'on peut
 - Superviser l'état des ApplicationMaster
 - Relancer des ApplicationMaster

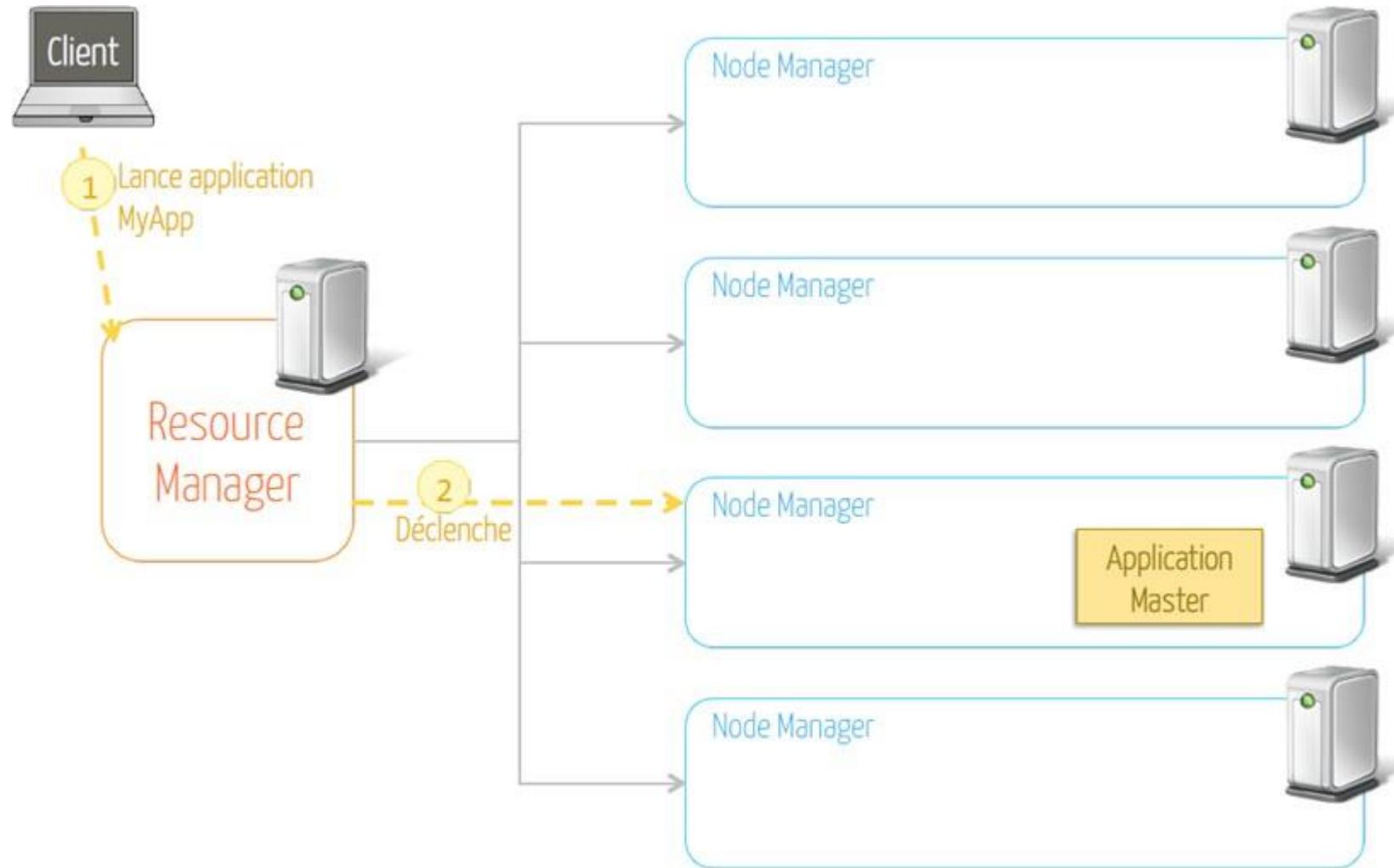
YARN (MapR2): Déroulement de l'exécution

- Le déroulement de l'exécution d'une tâche Hadoop suit les étapes suivantes:
 1. Le client (un outil Hadoop console) va soumettre le travail à effectuer au ResourceManager: une archive java .jar implémentant les opérations Map et Reduce, et également une classe driver (qu'on peut considérer comme le « main » du programme).
 2. Le ResourceManager alloue un container, « Application Master », sur le cluster et y lance la classe « driver » du programme.
 3. Cet « application master » va se lancer et confirmer au ResourceManager qu'il tourne correctement.
 4. Pour chacun des fragments des données d'entrée sur lesquelles travailler, l'Application Master va demander au Resource Manager d'allouer un container en lui indiquant les données sur lesquelles celui-ci va travailler et le code à exécuter.

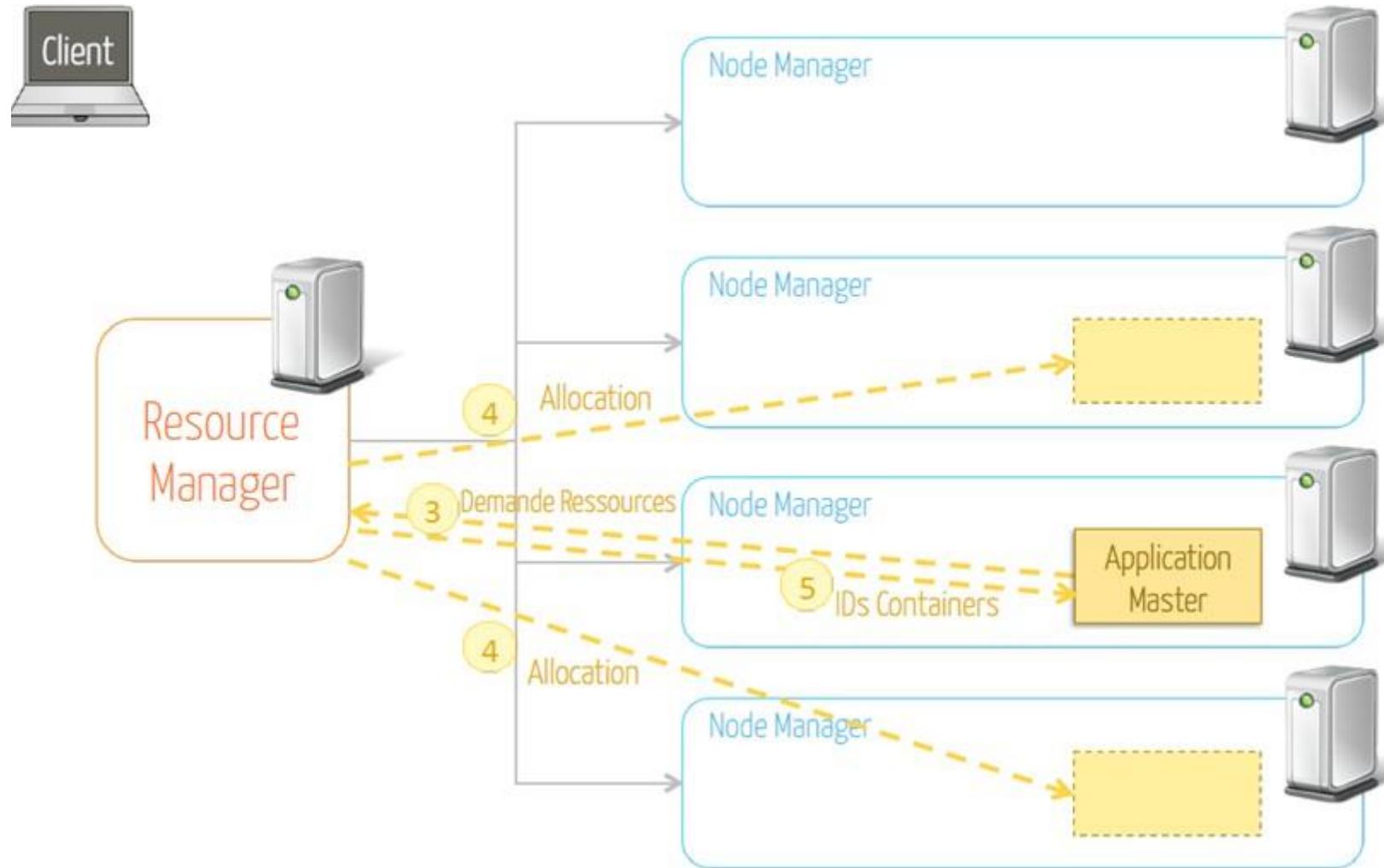
YARN (MapR2): Déroulement de l'exécution

5. L'Application Master va alors lancer le code en question (une classe Java, généralement map ou reduce) sur le container alloué. Il communiquera avec la tâche directement (via un protocole potentiellement propre au programme lancé), sans passer par le ResourceManager.
6. Ces tâches vont régulièrement contacter l'Application Master du programme pour transmettre des informations de progression, de statut, etc. parallèlement, chacun des NodeManager communique en permanence avec le ResourceManager pour lui indiquer son statut en terme de ressources (containers lancés, RAM, CPU), mais sans information spécifique aux tâches exécutées.
7. Pendant l'exécution du programme, le client peut à tout moment contacter le programme en cours d'exécution; pour ce faire, il communique directement avec l'Application Master du programme en contactant le container correspondant sans passer par le ResourceManager du cluster.
8. A l'issue de l'exécution du programme, l'Application Master s'arrête; son container est libéré et est à nouveau disponible pour de futures tâches.

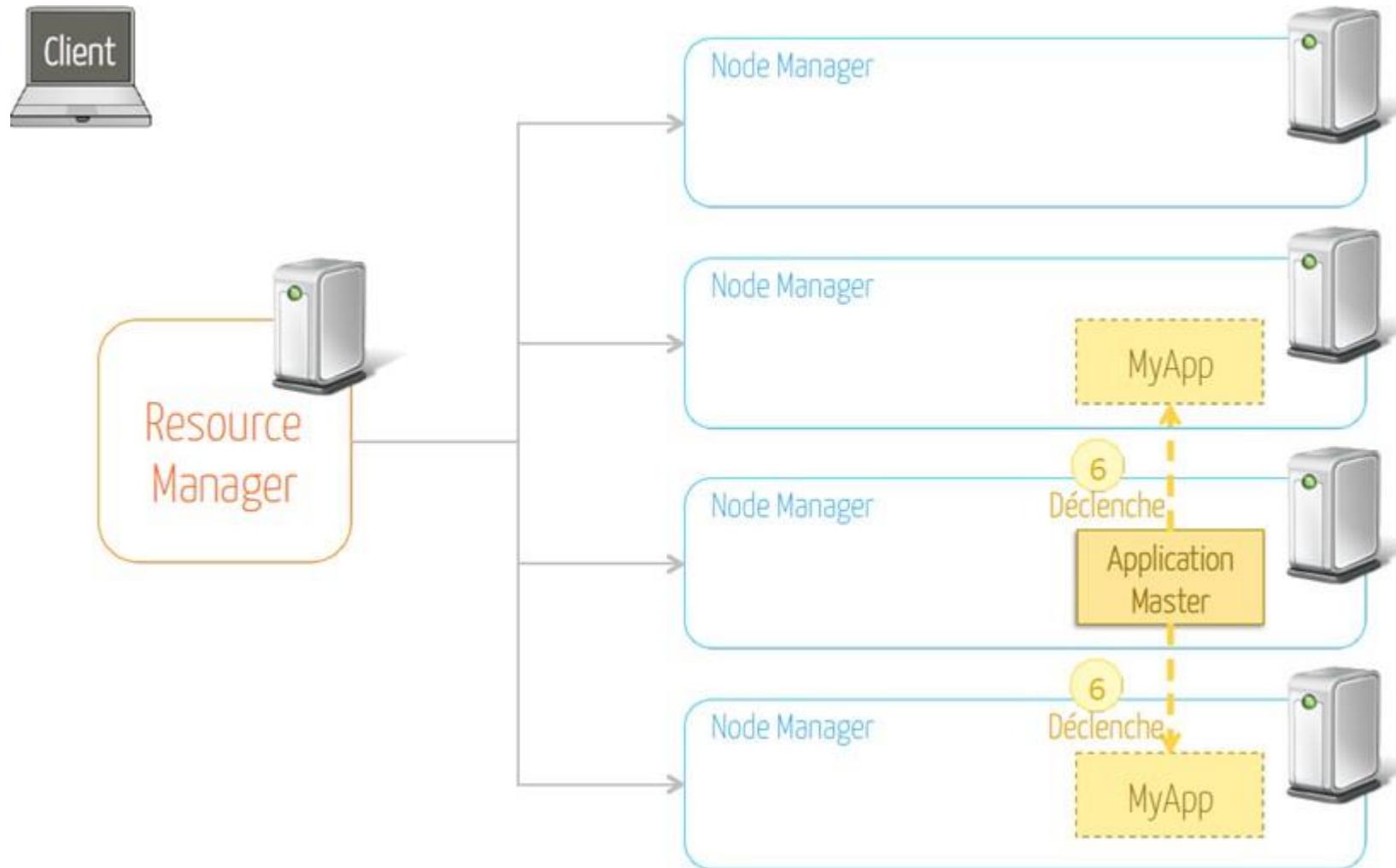
YARN (MapR2) : Lancement d'une Application dans un Cluster Yarn



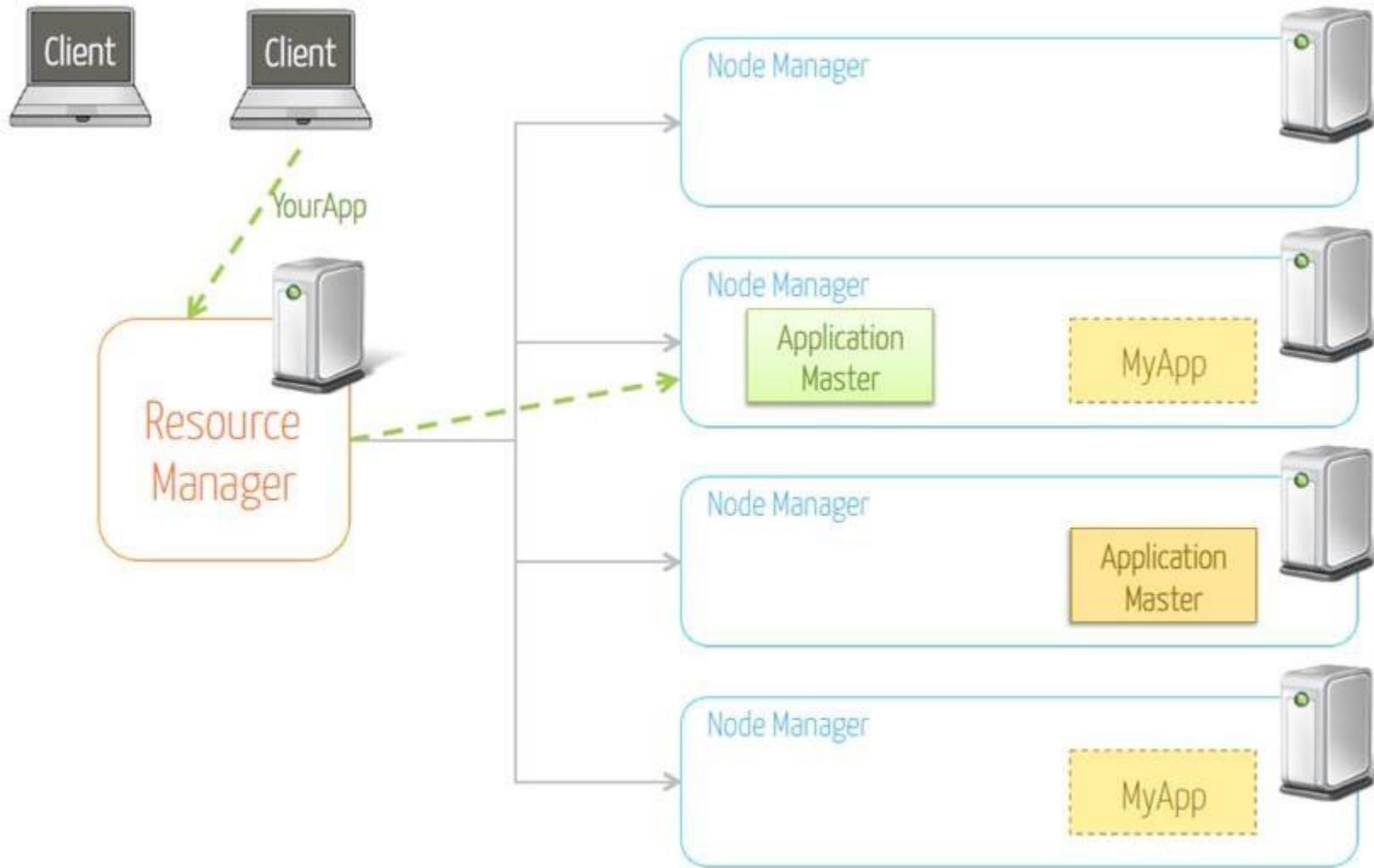
YARN (MapR2) : Lancement d'une Application dans un Cluster Yarn



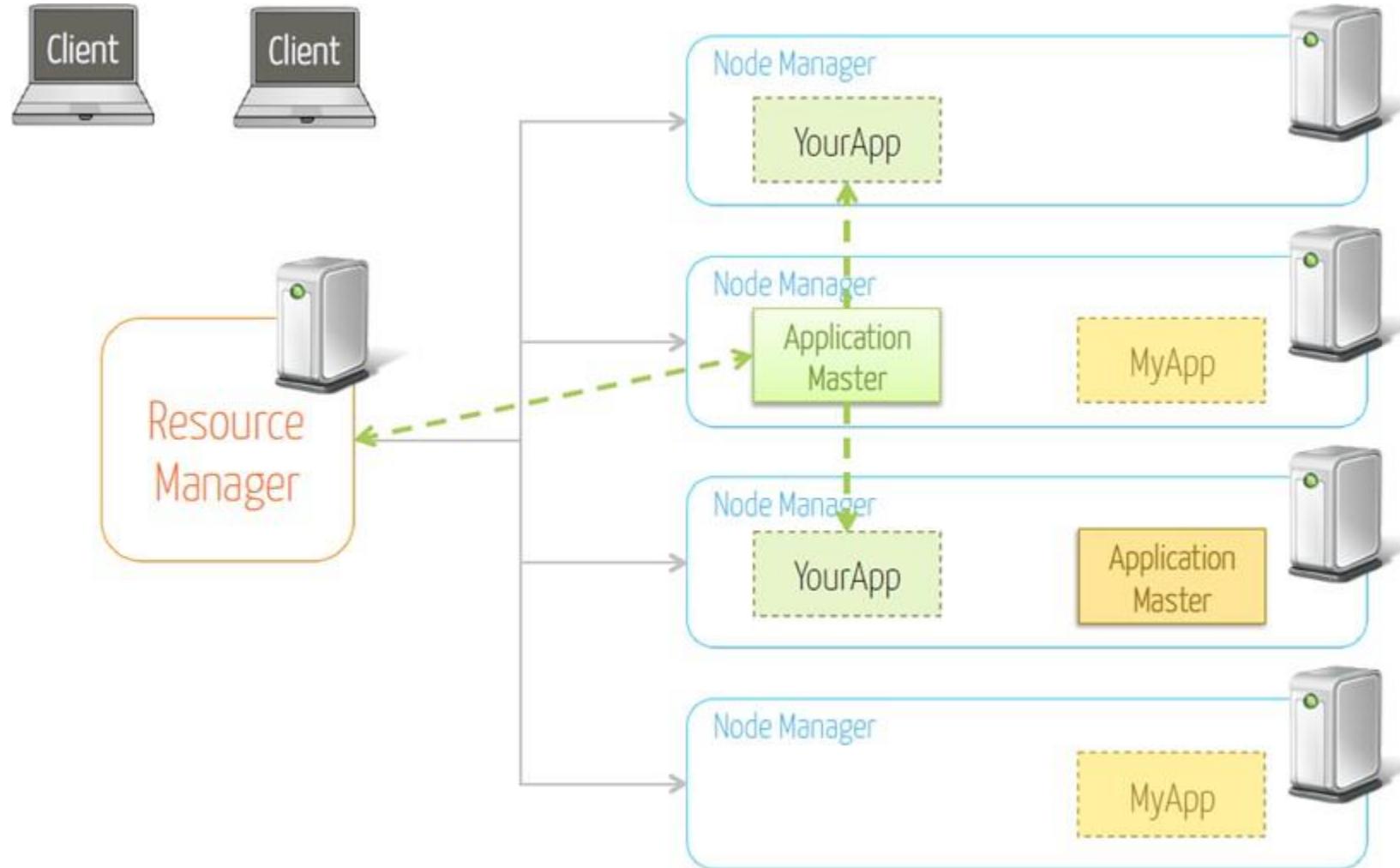
YARN (MapR2) : Lancement d'une Application dans un Cluster Yarn



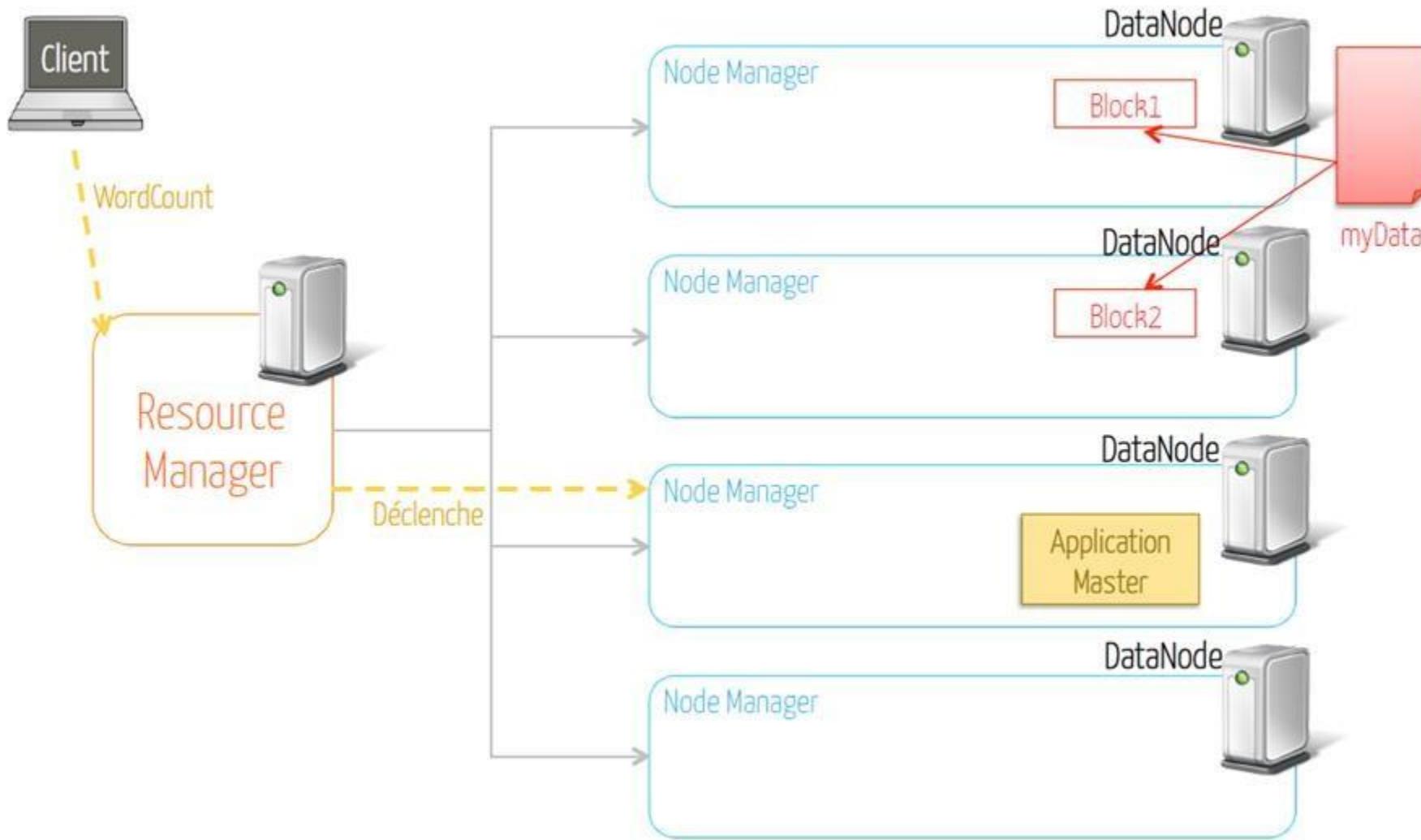
YARN (MapR2) : Lancement d'une Application dans un Cluster Yarn



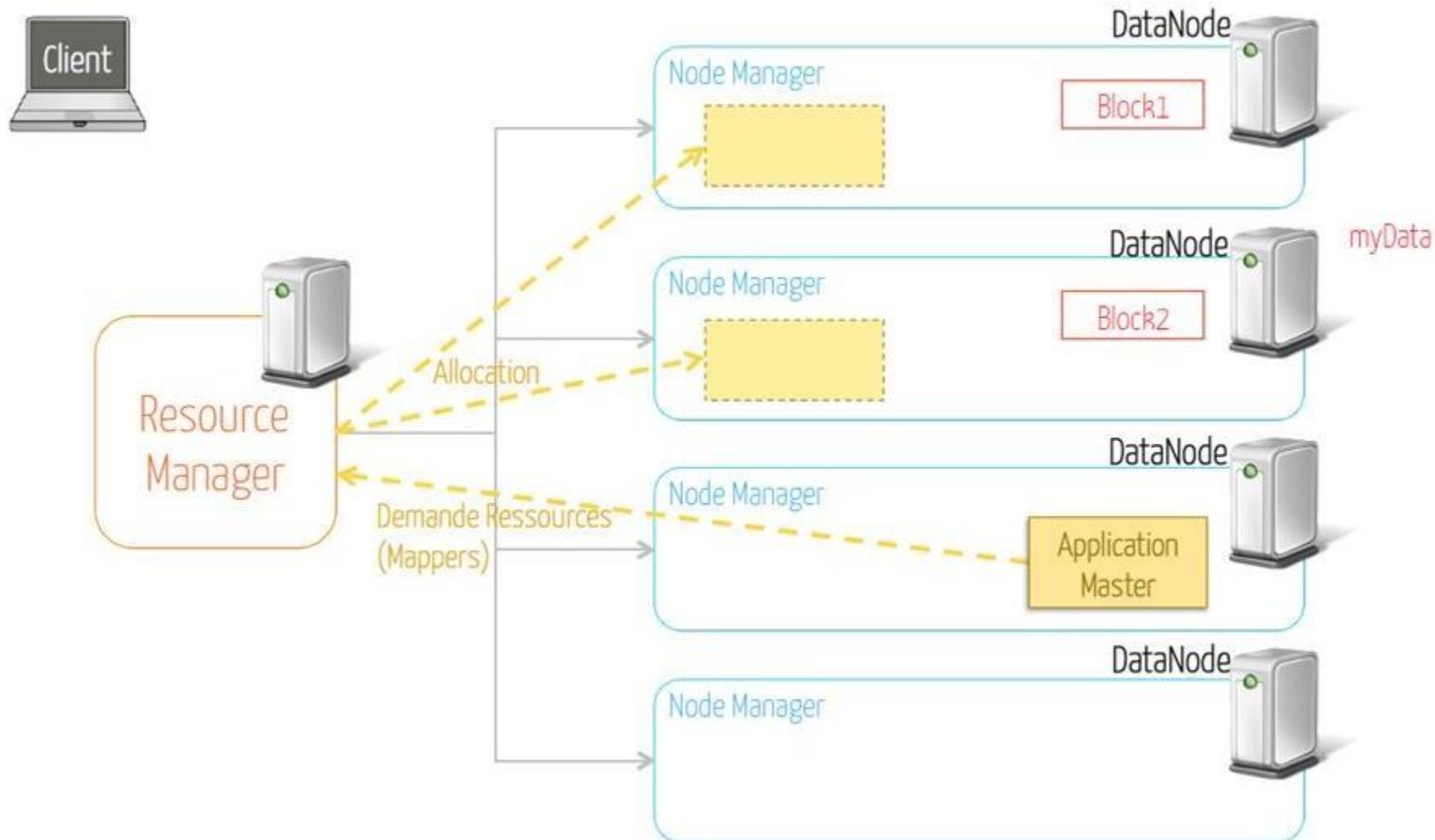
YARN (MapR2) : Lancement d'une Application dans un Cluster Yarn



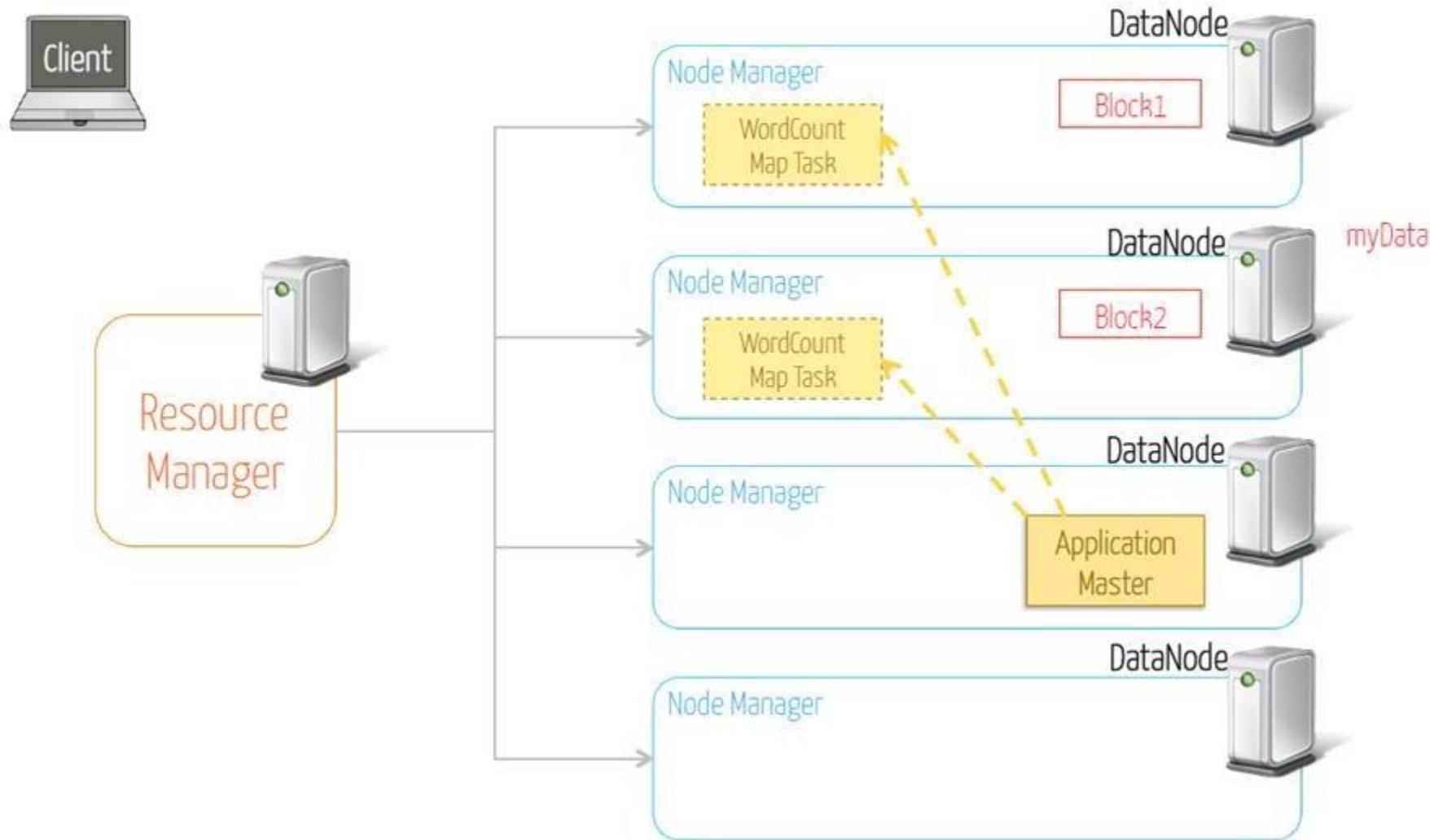
YARN (MapR2) : Exécution d'un Job MR



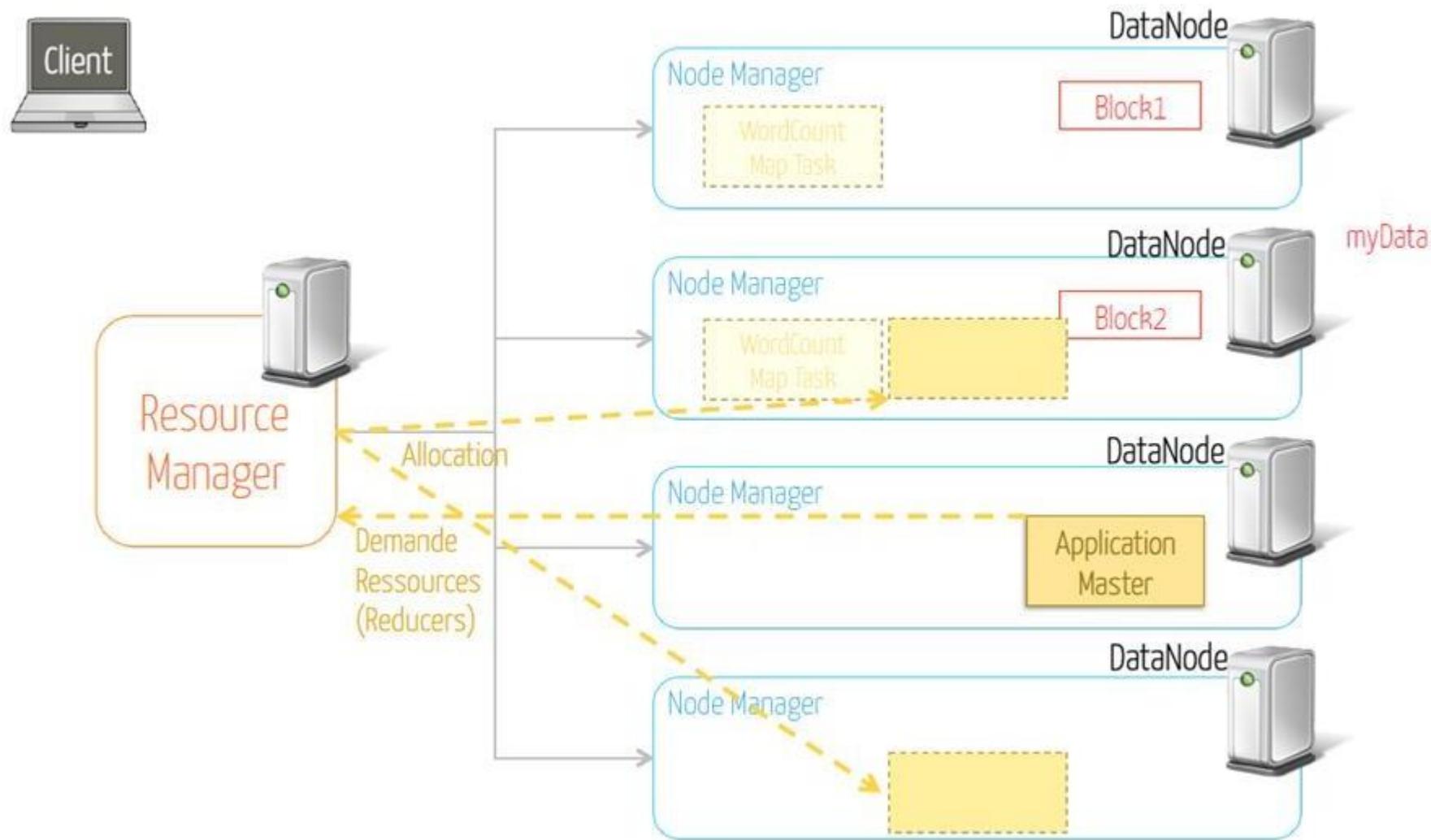
YARN (MapR2) : Exécution d'un Job MR



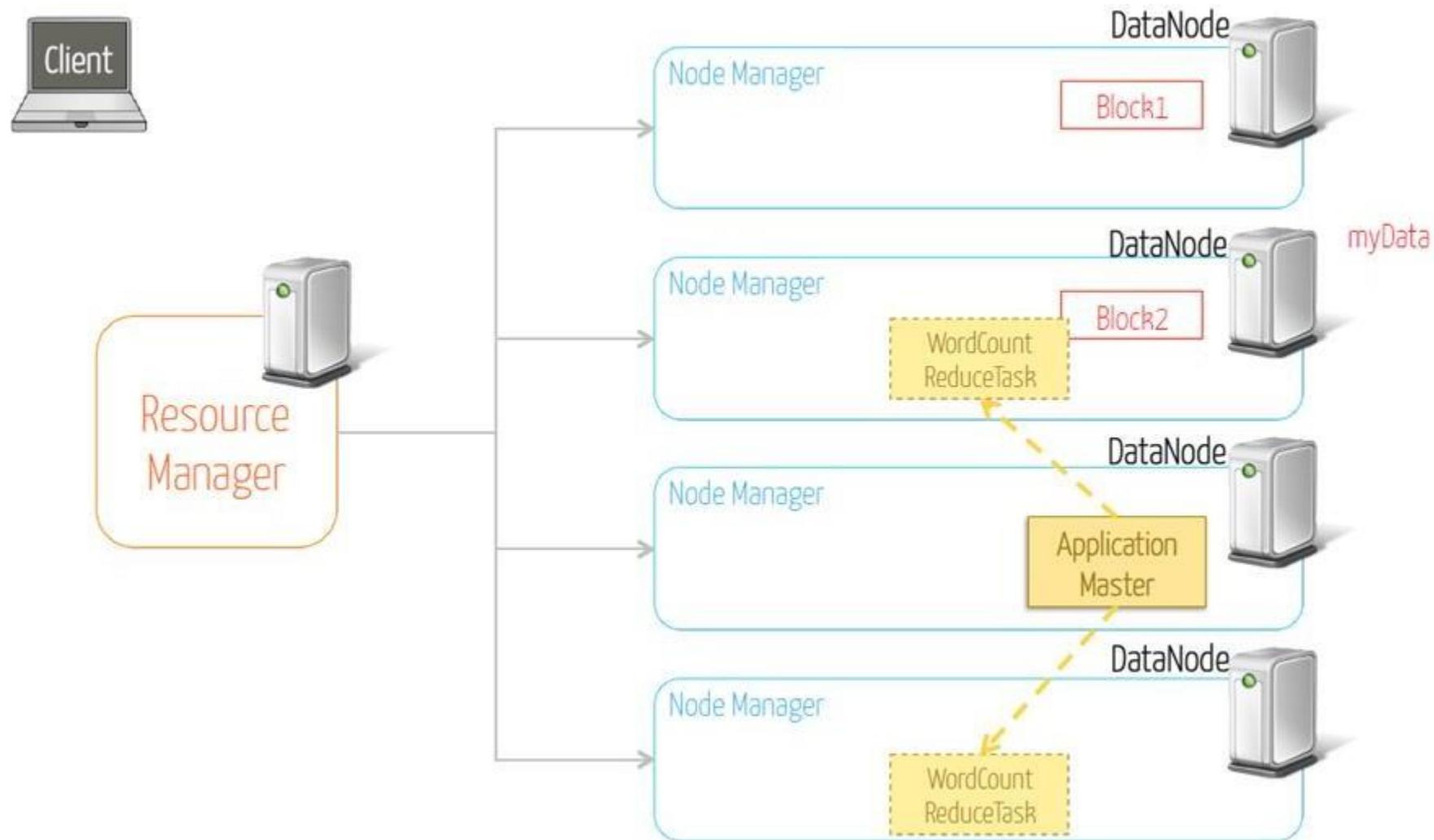
YARN (MapR2) : Exécution d'un Job MR



YARN (MapR2) : Exécution d'un Job MR



YARN (MapR2) : Exécution d'un Job MR



Hadoop 2: HDFS - Remarques

- **De la même façon que l'évolution de MapReduce 2, nous trouvons quelques améliorations de HDFS dans la 2ème version de Hadoop. Ces améliorations ont été déjà expliquées dans la partie consacrée à HDFS.**

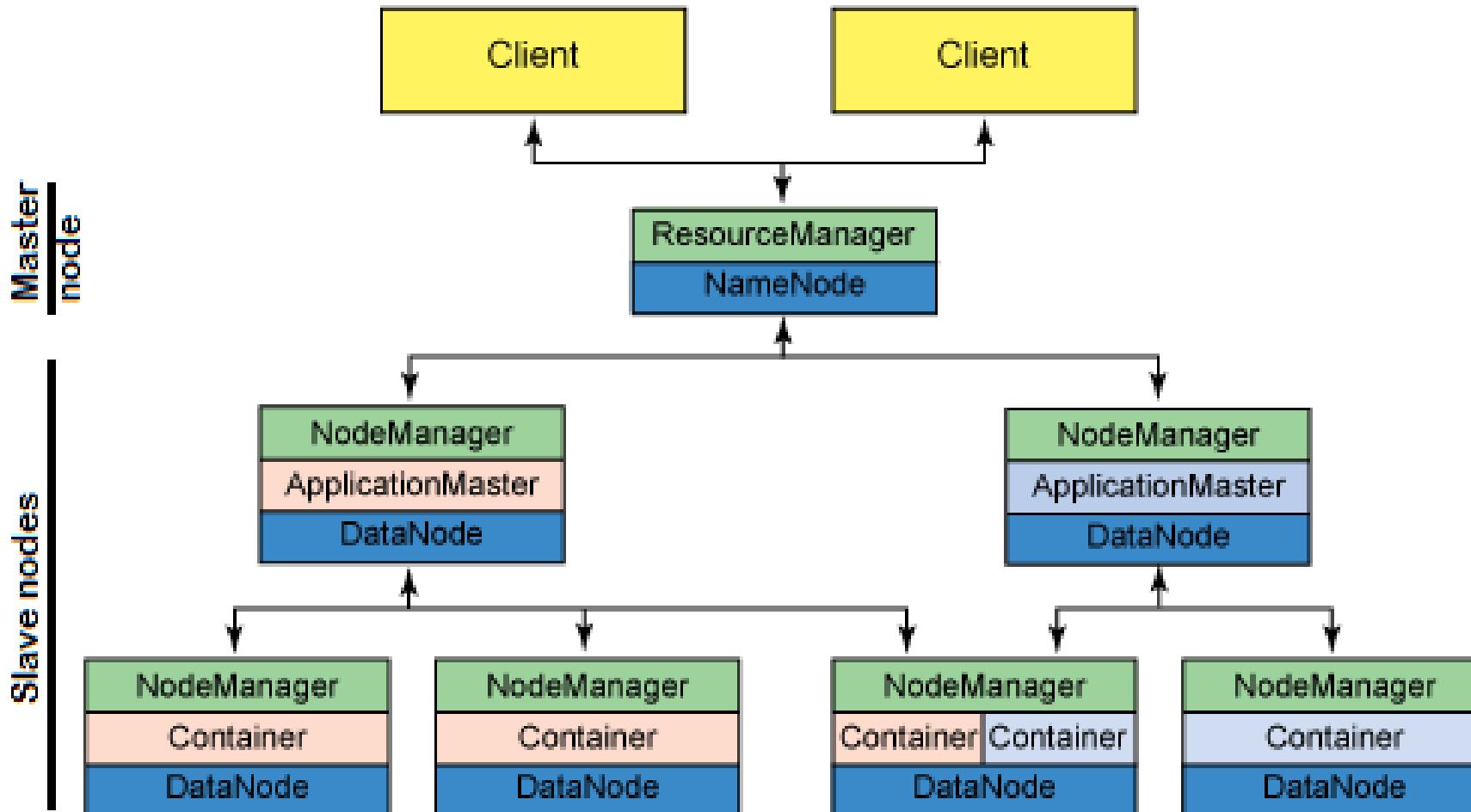
1. Le SPOF du Namenode a disparu

- L'ancienne architecture d'Hadoop imposait l'utilisation d'un seul namenode, composant central contenant les métadonnées du cluster HDFS. Ce namenode était un SPOF (Single Point Of Failure).
- Dans Hadoop 2, on peut mettre deux namenodes en mode actif/attente. Si le Namenode principal est indisponible, le Namenode secondaire prend sa place.

2. La fédération HDFS

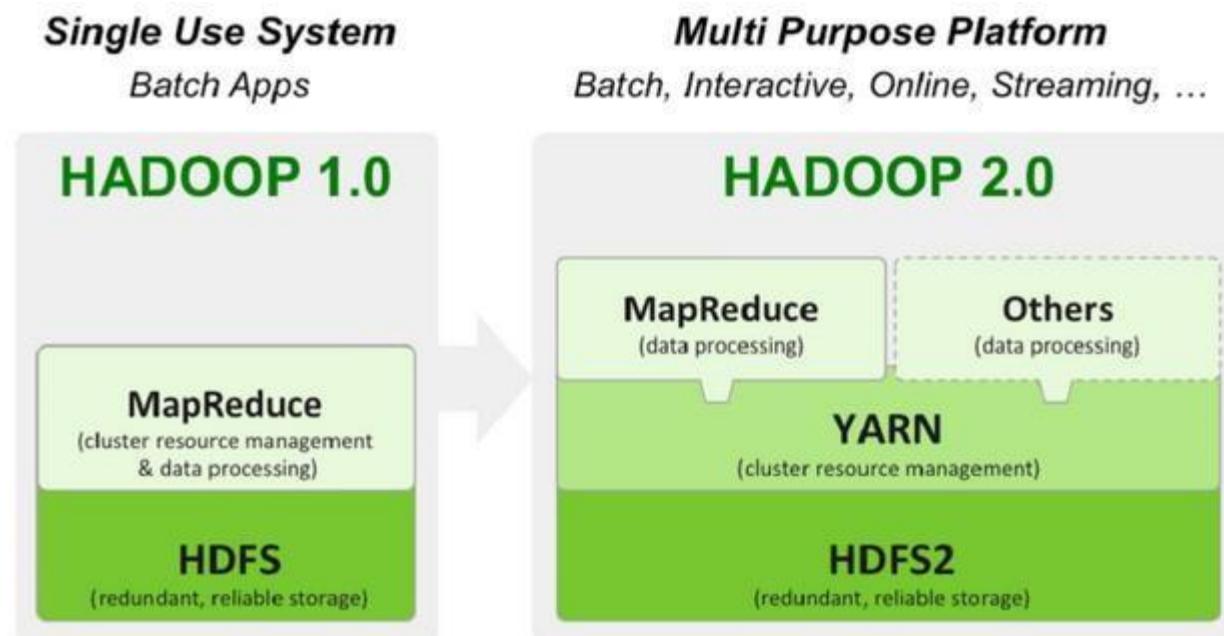
- Dans un cluster HDFS, un namenode correspond à un espace de nommage (namespace). Dans l'ancienne architecture d'Hadoop, on ne pouvait utiliser qu'un namenode par cluster. La fédération HDFS permet de supporter plusieurs namenodes et donc plusieurs namespace sur un même cluster. (Exemple : un NameNode pourrait gérer tous les fichiers sous /user, et un second NameNode peut gérer les fichiers sous /share.)

Hadoop 2: Architecture générale



Hadoop 2: Une évolution architecturale majeure

- Cette remise à plat des rôles a permis aussi de **découpler** Hadoop de MapReduce et, ce faisant, de permettre à des frameworks alternatifs d'être portés directement sur Hadoop et HDFS.
- Cela va donc permettre à Hadoop, outre une meilleure scalabilité, de s'enrichir de nouveaux frameworks couvrant des besoins peu ou pas couverts avec Map Reduce.



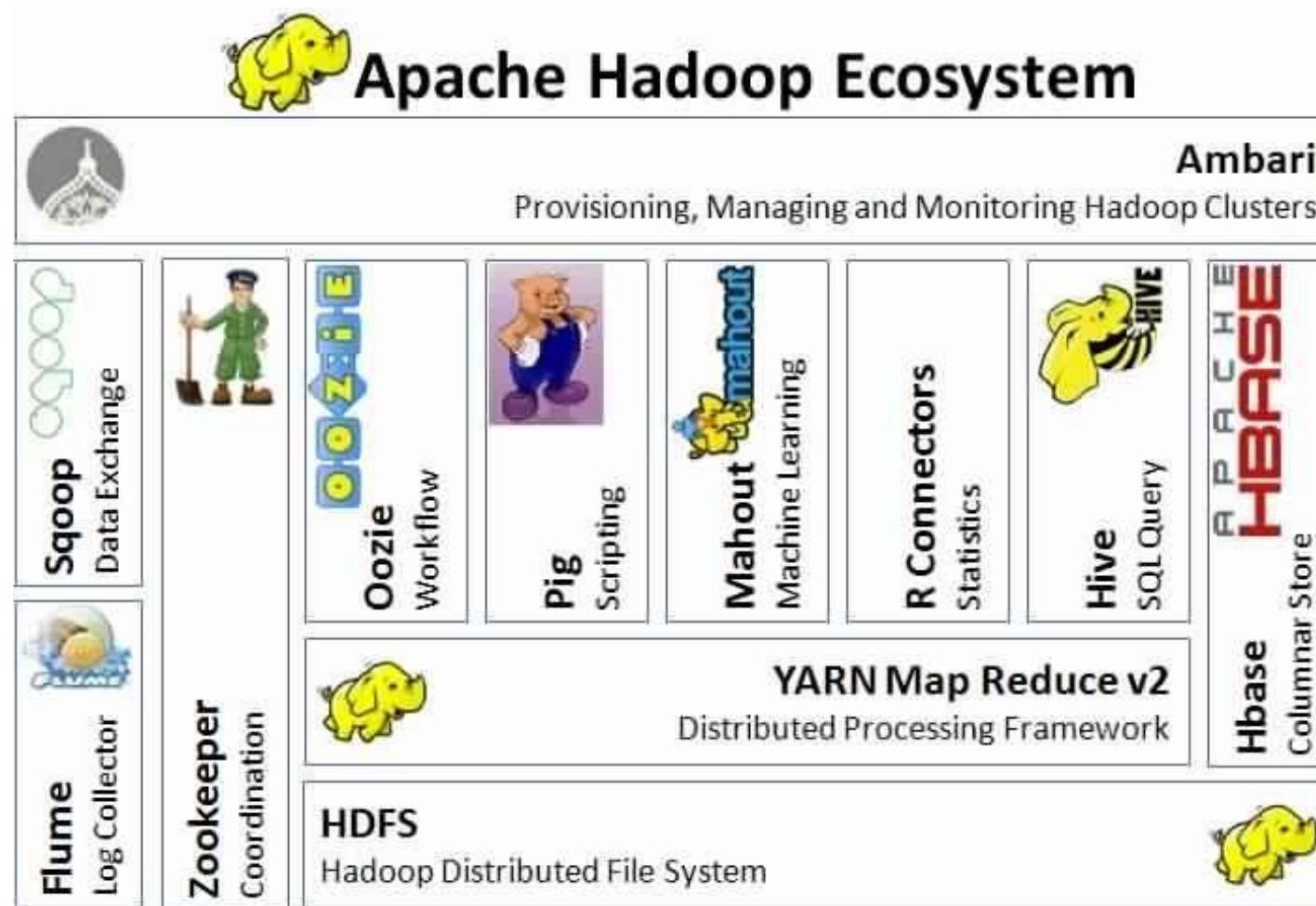
Hadoop 2: Une évolution architecturale majeure

Hadoop se transforme en OS de la donnée !



- Client et cluster peuvent utiliser des versions différentes.
- Des protocoles de communication standardisés et documentés.
- Évolution du framework progressive avec rétro-compatibilité sans destruction des services.

Hadoop 2 : Écosystème (1/2)



Hadoop 2 : Écosystème (2/2)

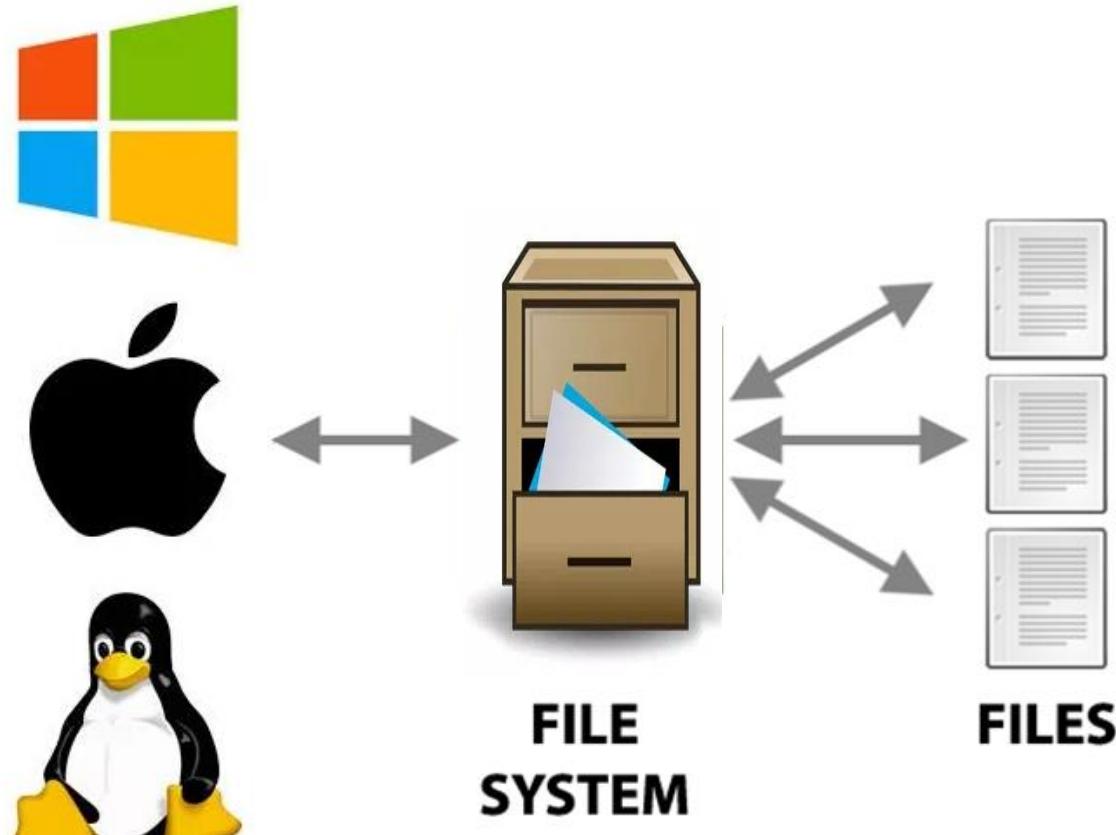
- **Pig** : un langage de haut niveau dédié à l'analyse de gros volumes de données. Il s'adresse aux développeurs habitués à faire des **scripts** via Bash ou Python par exemple.
- **Hive** : un système d'entrepôt de données (Data Warehouse) pour Hadoop qui offre un langage de requête proche de **SQL** pour faciliter les agrégations, le requêtage ad-hoc et l'analyse de gros volumes de données stockés dans des systèmes de fichiers compatibles Hadoop.
- **Hbase** : un système de gestion de base de données **NoSQL** orienté colonnes. Il est distribué, scalable et dédié au big data avec un accès direct et une lecture/écriture temps réel.
- **Sqoop** : un outil conçu pour **transférer** efficacement une masse de données entre Apache Hadoop et un stockage de données **structuré** tel que les bases de données **relationnelles**.
- **Mahout** : un système d'**apprentissage automatique** et d'analyse de données. Il implémente des algorithmes de classification et de regroupement automatique (Machine Learning, DataMining).
- **Flume** : un service distribué, fiable et disponible pour collecter efficacement, agréger et déplacer une grande quantité de **logs**.
- **Zookeeper** : un service centralisé pour maintenir les **configurations**, la **nomenclature**, pour fournir une **synchronisation** distribuée et des services groupés.
- **Oozie** : un système de **flux de travail** (workflow) dont l'objectif est de simplifier la coordination et la séquence de différents traitements (programme Map/Reduce, scripts Pig,...).

Chapitre 3

Utilisation de Hadoop



Hadoop Distributed File System



Hadoop Distributed File System

- ▶ Les données dans Hadoop sont stockées sur un système de fichiers appelé HDFS ou système de fichiers distribué Hadoop.
- ▶ Le HDFS est un système de fichiers distribué développé en Java.
- ▶ Permet de stocker des données volumineuses sur plusieurs nœuds d'un cluster Hadoop.

Hadoop Distributed File System

- ▶ Conçu de manière à ce que chaque machine du cluster apporte son stockage individuel pour stocker tout type de données.
- ▶ Si nous avons 10 machines avec un disque dur de 100 Go sur chaque machine, avec HDFS, nous aurons un service de stockage de 1 To.

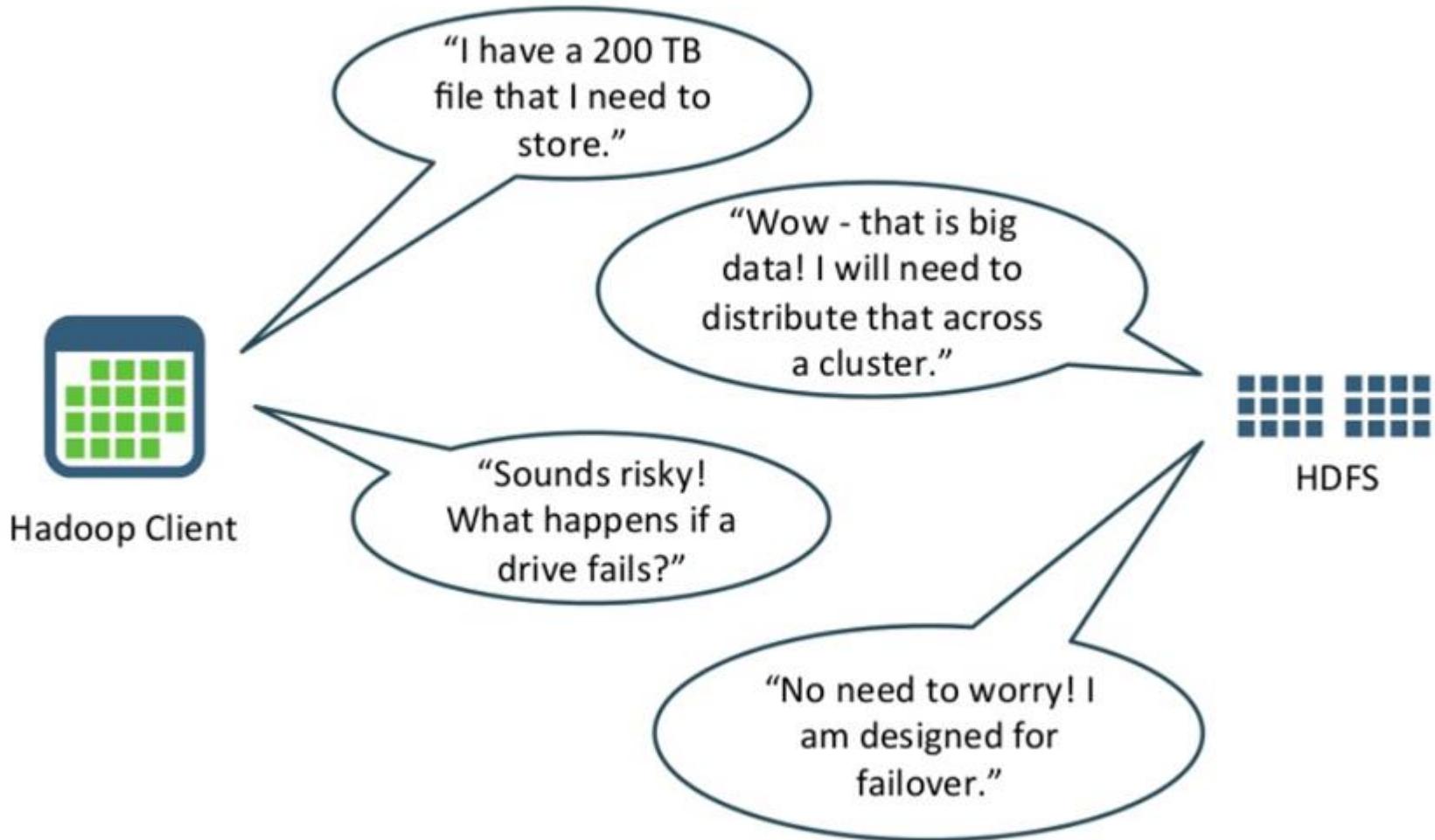
Hadoop Distributed File System

► Du point de vue de l'utilisateur:

- ◆ Similaire au système de fichiers disponible sur une seule machine.
- ◆ Mais les données sont stockées sur plusieurs machines.
- ◆ Plusieurs utilisateurs peuvent accéder aux données simultanément.



Hadoop Distributed File System



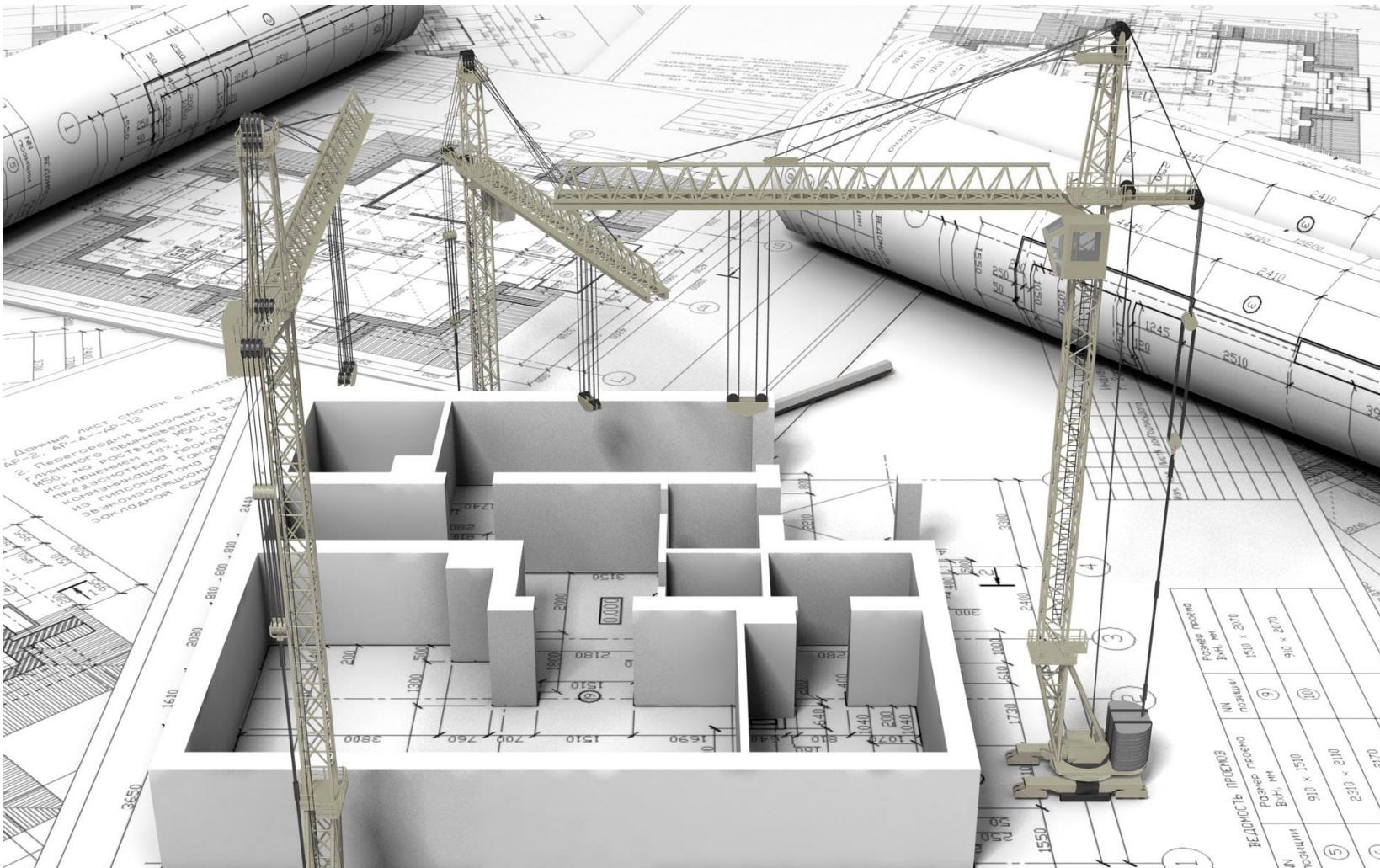
Hadoop Distributed File System

- ▶ Avec HDFS, les données sont divisées en petits morceaux, dits blocs, et distribuées sur un cluster de machines.
- ▶ HDFS présente les caractéristiques suivantes:
 - ◆ Prend en charge une organisation de fichiers hiérarchique traditionnelle.
 - ◆ Conçu pour être déployé sur du matériel peu coûteux.

Hadoop Distributed File System

- ◆ Conçu pour évoluer facilement et efficacement (l'ajout de plusieurs nœuds augmente à la fois l'espace de stockage et le débit de traitement).
- ◆ Fiabilité: les données sont répliquées pour qu'une pane de disque soit non seulement acceptable, mais attendue et gérée de manière transparente.

Architecture du HDFS

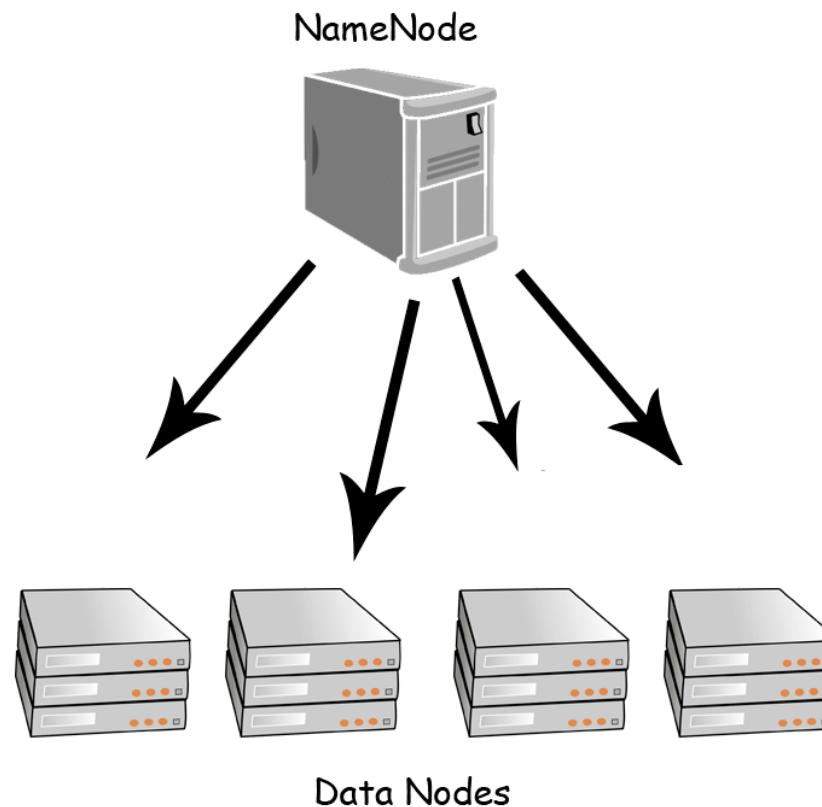


Architecture du HDFS

- ▶ Une instance Hadoop consiste en un cluster de machines HDFS, souvent appelé cluster Hadoop ou cluster HDFS.
- ▶ HDFS a une architecture maître / esclave.
- ▶ Un cluster HDFS comprend deux composants principaux:
 - ◆ NameNode, le nœud «maître» de HDFS.
 - ◆ DataNode, le nœud «esclave» de HDFS.

Architecture du HDFS

► **NameNode et DataNode sont des processus démon exécutés dans le cluster.**



Architecture du HDFS - NameNode

- ▶ Le NameNode gère les données (sans les stocker réellement) en déterminant et en maintenant la façon dont les blocs de données sont répartis dans les DataNodes.
- ▶ Le NameNode est un serveur à très haute disponibilité qui gère l'espace de nommage du système de fichiers.
- ▶ Un NameNode représente un seul espace de noms.

Architecture du HDFS - NameNode

- ▶ Un cluster peut avoir plusieurs NameNodes si plusieurs espaces de noms sont souhaités.
- ▶ Le NameNode contrôle l'accès aux fichiers par les clients.
- ▶ Les données ne résident et ne passent jamais par le NameNode.
- ▶ Le Big Data réside uniquement sur DataNodes.

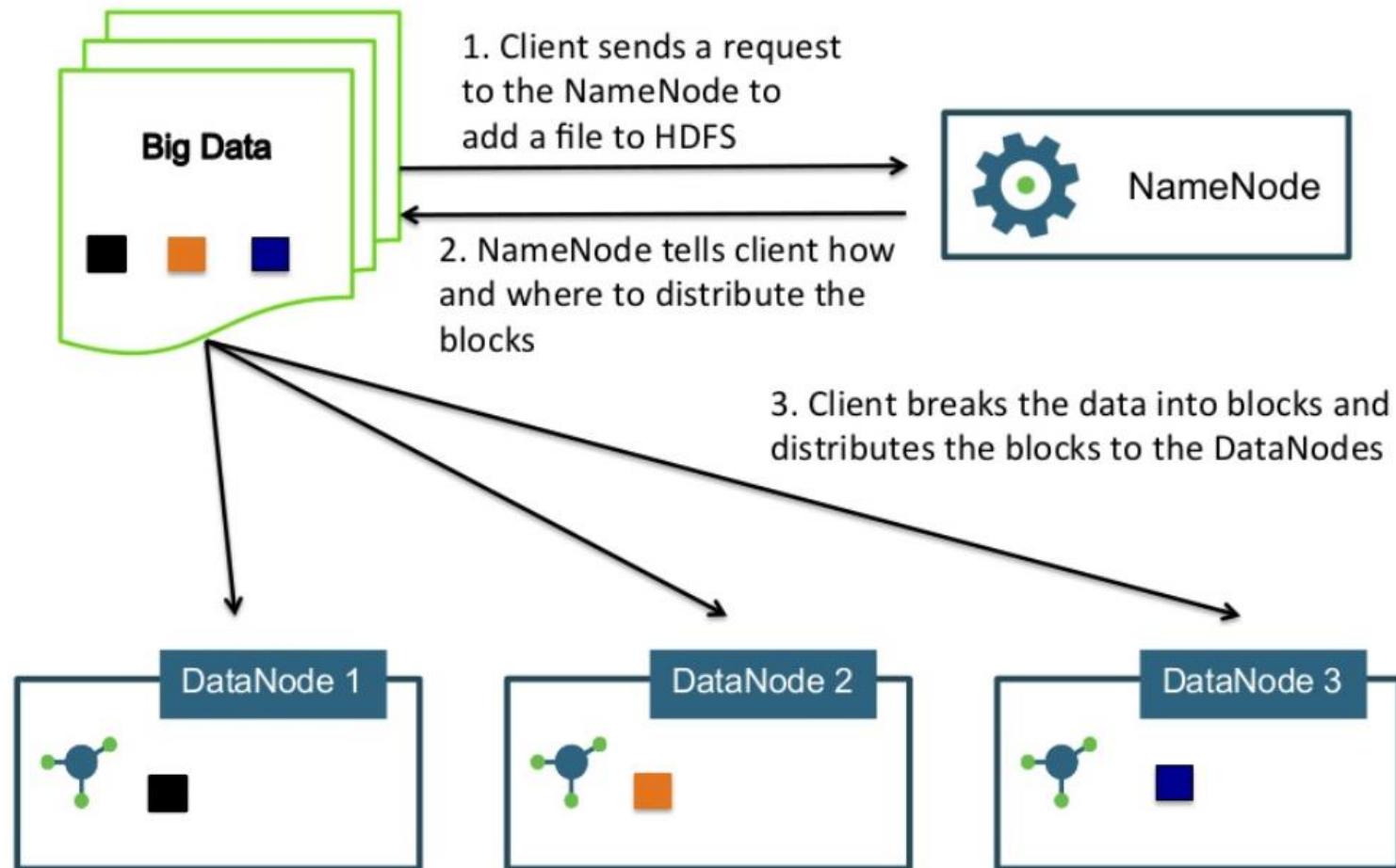
Architecture du HDFS - DataNode

- ▶ **Le DataNode stocke les morceaux de données.**
- ▶ **Le DataNode est responsable de la réPLICATION des morceaux de données sur d'autres DataNodes sur instruction du NameNode.**
- ▶ **Les DataNodes sont appelés démons «esclaves» vis-à-vis du NameNode et communiquent en permanence leur état à ce dernier.**

Architecture du HDFS - DataNode

► Le NameNode enregistre la manière dont les données sont décomposées en fragments sur les DataNodes.

Bloc de stockage



Bloc de stockage

► **L'enregistrement d'un fichier dans le HDFS implique les étapes suivantes:**

- ◆ Une application cliente envoie une demande au NameNode qui spécifie où il voudrait placer le fichier dans le système de fichiers.
- ◆ Le NameNode détermine la manière dont les données sont divisées en blocs et les DataNodes qui seront utilisés pour stocker ces blocs.

Cette information est donnée à l'application cliente.

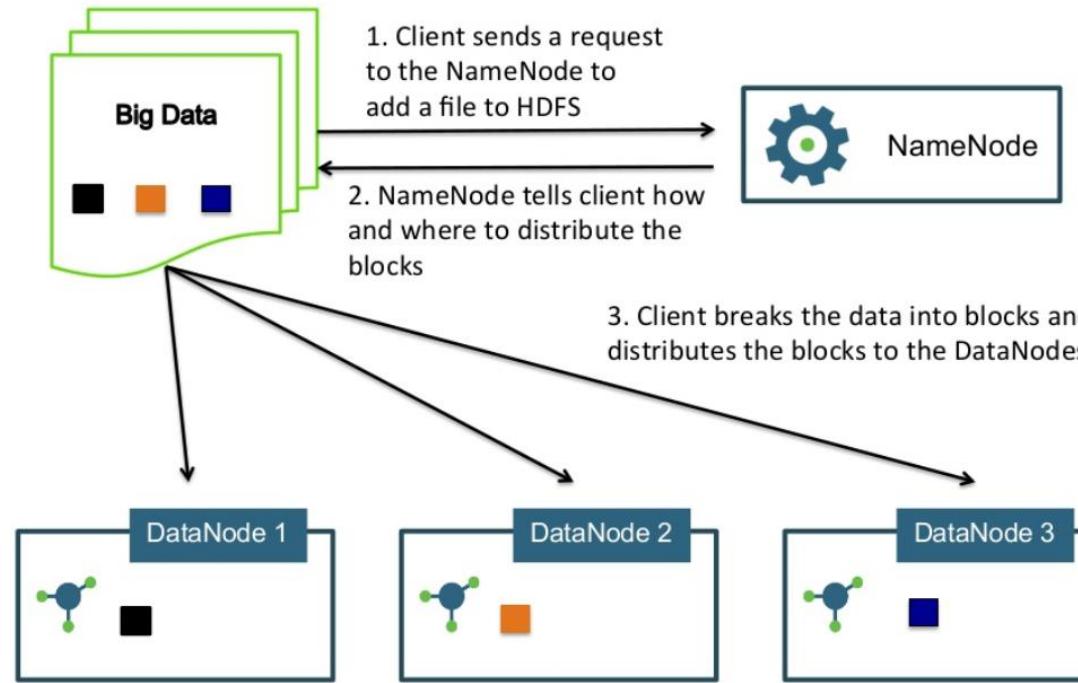
Bloc de stockage

- ◆ L'application cliente communique directement avec chaque DataNode en écrivant les blocs sur les DataNodes.
 - ◆ Les DataNodes répliquent les blocs nouvellement créés en fonction des instructions du NameNode.
- La taille de bloc par défaut dans Hadoop 2 est de 128 Mo.

Bloc de stockage

- ▶ Il est possible de spécifier la taille de bloc pour chaque fichier à l'aide de la propriété `dfs.blocksize`.
- ▶ Si la taille de bloc n'est pas spécifiée au niveau du fichier, la valeur globale de `dfs.blocksize` définie dans le fichier `hdfs-site.xml` est utilisée.
- ▶ La taille des blocs doit être assez grosse pour réduire les métadonnées.

Bloc de stockage



Myfile.txt		
360 Mo		
Block A	Block B	Block C
128 Mo	128 Mo	104 Mo

Bloc de stockage

► Le facteur de réPLICATION par défaut est 3 (valeur également configurable), ce qui signifie que chaque bloc de données est répliqué sur trois DataNodes.

Le NameNode

- ▶ Un cluster HDFS consiste en un seul NameNode, qui est un serveur maître qui gère l'espace de noms du système de fichiers et régule l'accès aux fichiers par les clients.
- ▶ Le NameNode détermine la distribution des blocs sur les DataNodes.
- ▶ Le NameNode est responsable de prendre en compte le facteur de réPLICATION de tous les blocs.

Le NameNode

- ▶ Le NameNode gère les panes des DataNode.
- ▶ Le NameNode exécute les opérations de l'espace de noms du système de fichiers, telles que l'ouverture, la fermeture et le changement de nom de fichiers et de répertoires.

Le NameNode

- ▶ Le NameNode maintient l'espace de noms du système de fichiers en enregistrant les métadonnées de tous les fichiers stockés dans le cluster:
 - ◆ L'emplacement où les blocs sont stockés.
 - ◆ La taille des fichiers.
 - ◆ Les autorisations.
 - ◆ La hiérarchie, etc.

Le NameNode

► Les NameNode effectue ses tâches en maintenant deux fichiers associés aux métadonnées:

- ◆ Fsimage.
- ◆ Edit logs.

► Toute corruption de ces fichiers peut rendre l'instance de cluster HDFS non fonctionnelle.

Le NameNode

- ▶ Le NameNode peut être configuré pour prendre en charge la maintenance de plusieurs copies de Fsimage et Edit logs sur un autre serveur.
- ▶ Le NameNode stocke uniquement les métadonnées du système de fichiers, mais n'est pas responsable du stockage ou du transfert des données.

► Le fichier Fsimage:

- ◆ Contient l'intégralité de l'espace de noms du système de fichiers, y compris le mapping des blocs en fichiers et les propriétés du système de fichiers.
- ◆ Contient l'état complet de l'espace de noms du système de fichiers depuis le démarrage du NameNode.

► Le fichier Edit logs :

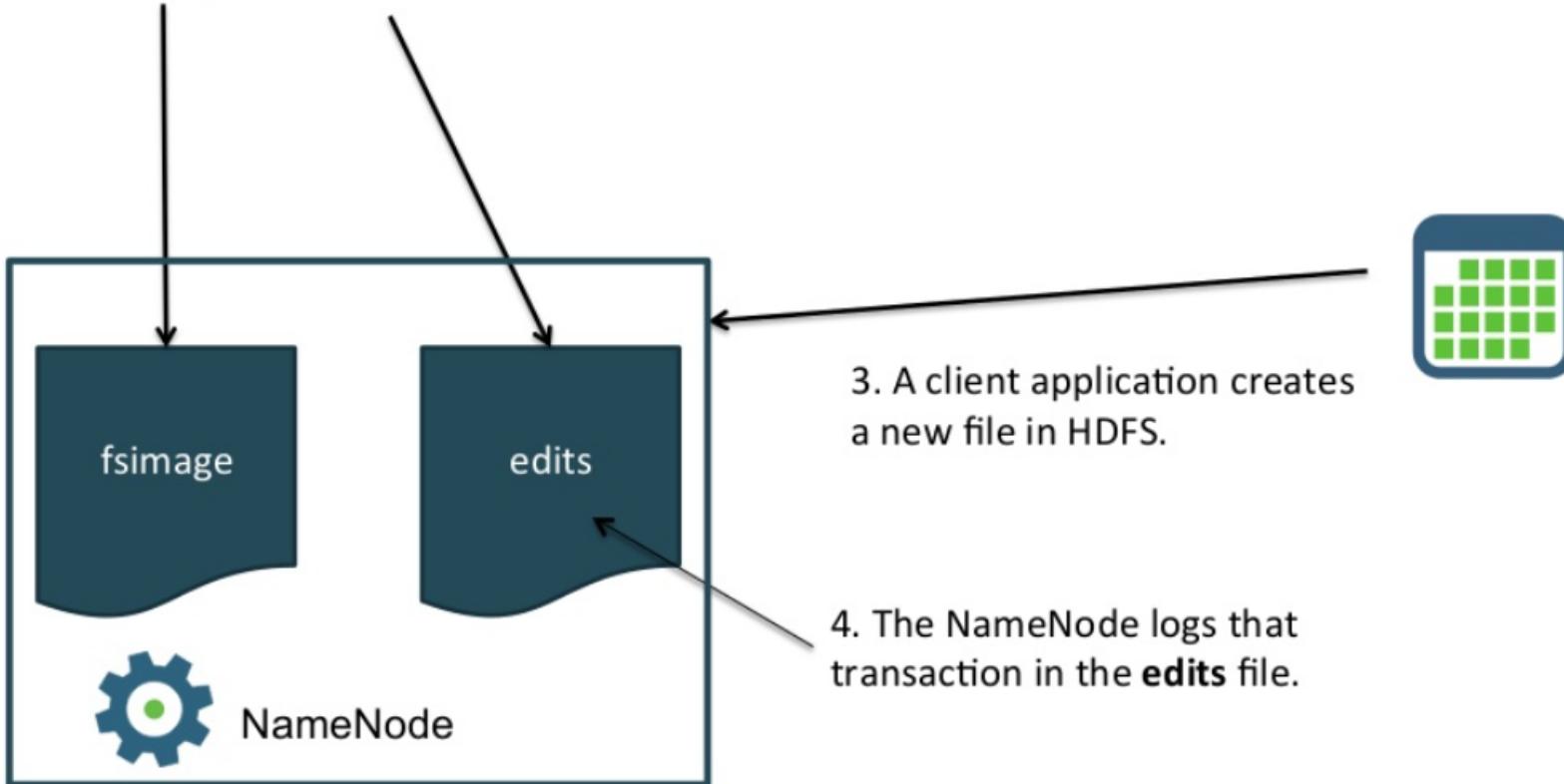
- ◆ Un journal de transactions qui enregistre en permanence toutes les modifications apportées aux métadonnées du système de fichiers.
- ◆ Contient toutes les modifications récentes apportées au système de fichiers par rapport au dernier fichier Fsimage.
- ◆ Enregistre chaque modification apportée aux métadonnées du système de fichiers.

Le NameNode

► Si un fichier est créé, supprimé ou déplacé dans le HDFS, le NameNode l'enregistrera immédiatement dans le fichier Edit logs.

Le NameNode

1. When the NameNode starts, it reads the **fsimage** and **edits** files.
2. The transactions in **edits** are merged with **fsimage**, and **edits** is emptied.



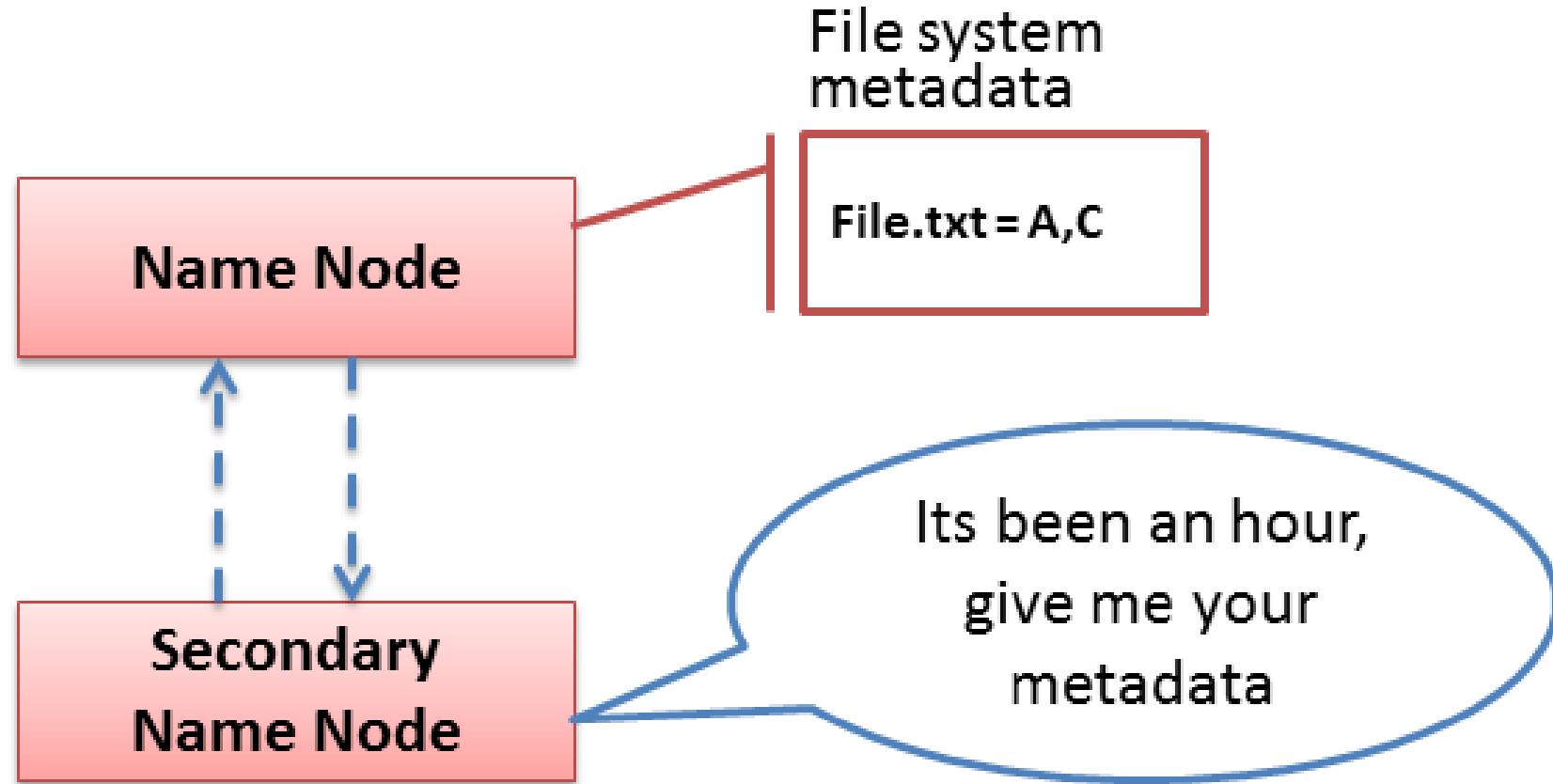
Le NameNode

► Les fichiers Fsimage et Edit logs sont conservés sur la machine NameNode sous le chemin :

- ◆ **/hadoop/hdfs/namenode/current/**

```
#ssh namenode
[namenode ~]# ls -la /hadoop/hdfs/namenode/current/ total 1048
-rw-r--r-- 1 hdfs hdfs1048576 edits_inprogress_00000000000000000001
-rw-r--r-- 1 hdfs hdfs336 fsimage_00000000000000000000
-rw-r--r-- 1 hdfs hdfs62 fsimage_00000000000000000000.md5
```

Le NameNode secondaire



Le NameNode secondaire

- ▶ Le NameNode secondaire fonctionne simultanément avec le NameNode principal en tant que démon d'assistance.
- ▶ La fonction principale du NameNode secondaire est de stocker une copie du fichier Fsimage et de purger le fichier EditLogs.
- ▶ Le NameNode secondaire n'est pas une réplique passive du NameNode.

Le NameNode secondaire

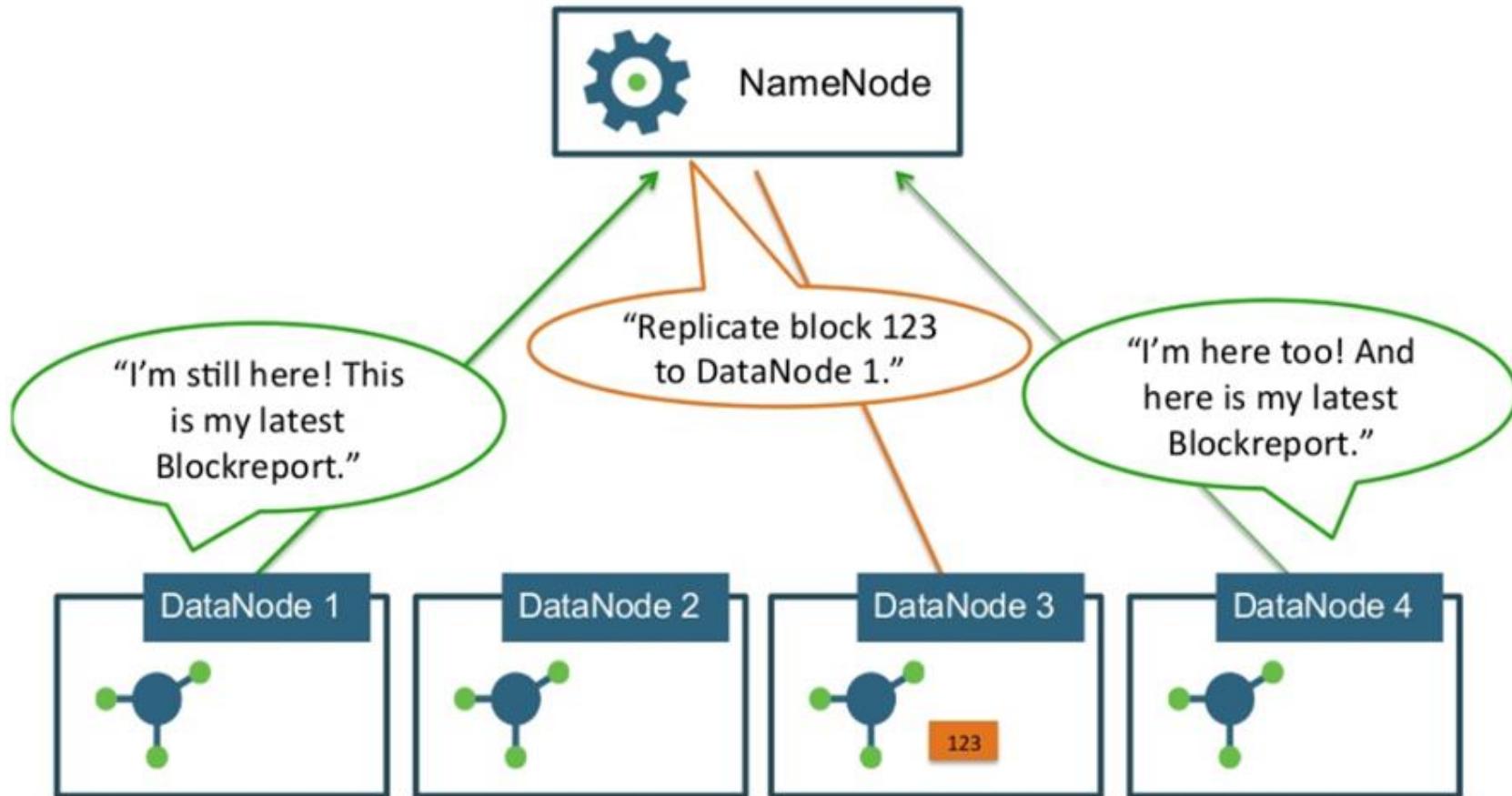
- ▶ Le NameNode secondaire effectue périodiquement une opération de point de contrôle en fusionnant l'espace de nom Fsimage avec le Edit logs et en le réimportant vers le NameNode.
- ▶ Le NameNode secondaire est également responsable de la sauvegarde du NameNode Fsimage (une copie du Fsimage fusionné).
- ▶ Si le NameNode principal échoue, une perte de données peut survenir.

Le NameNode secondaire

- ▶ **Le NameNode secondaire est responsable de:**
 - ◆ Combinaison des Edit logs avec Fsimage à partir du NameNode.
 - ◆ Télécharge les Edit Logs du NameNode à intervalles réguliers et l'applique à Fsimage.
 - ◆ Le nouveau Fsimage est recopié dans le NameNode.
 - ◆ Le nouveau Fsimage est utilisé chaque fois que le NameNode est démarré la prochaine fois.

- ▶ **Les DataNodes sont responsables de:**
 - ◆ La gestion des demandes de lecture et d'écriture des clients.
 - ◆ Exécution de la création, de la suppression et de la réPLICATION de blocs sur instruction du NameNode.
 - ◆ Envoi des heartbeats au NameNode.
 - ◆ Envoi d'un rapport de blocs au NameNode.

Le DataNode



Le DataNode

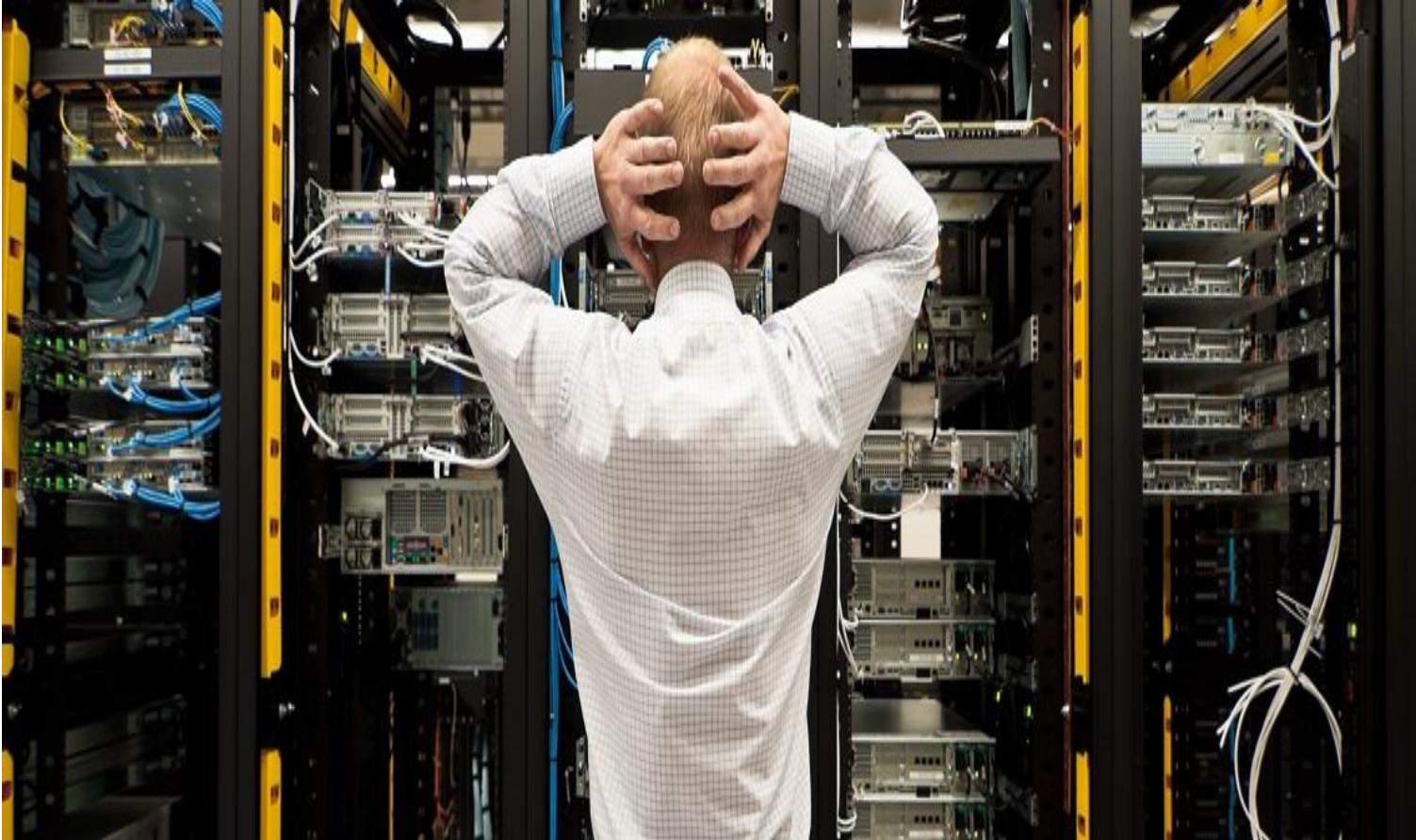
- ▶ Le NameNode reçoit périodiquement un heartbeat et un rapport de blocs de chacun des DataNodes du cluster.
- ▶ La réception d'un heartbeat signifie que le DataNode fonctionne correctement.
- ▶ Un rapport de blocs contient une liste de tous les blocs sur le DataNode.

- ▶ **Quand un DataNode démarre:**
 - ◆ Il analyse son système de fichiers local.
 - ◆ Génère une liste de tous les blocs de données HDFS.
 - ◆ Envoie ces informations au NameNode en tant que rapport de blocs.

Le DataNode

- ▶ Le DataNode n'a aucune connaissance sur les fichiers du HDFS.
- ▶ Le DataNode stocke chaque bloc de données HDFS dans un fichier séparé sur son système de fichiers local.
- ▶ Le DataNode ne crée pas tous les fichiers dans le même répertoire local. Au lieu de cela, il utilise une technique de découverte pour déterminer le nombre optimal de fichiers par répertoire.

La défaillance d'un nœud

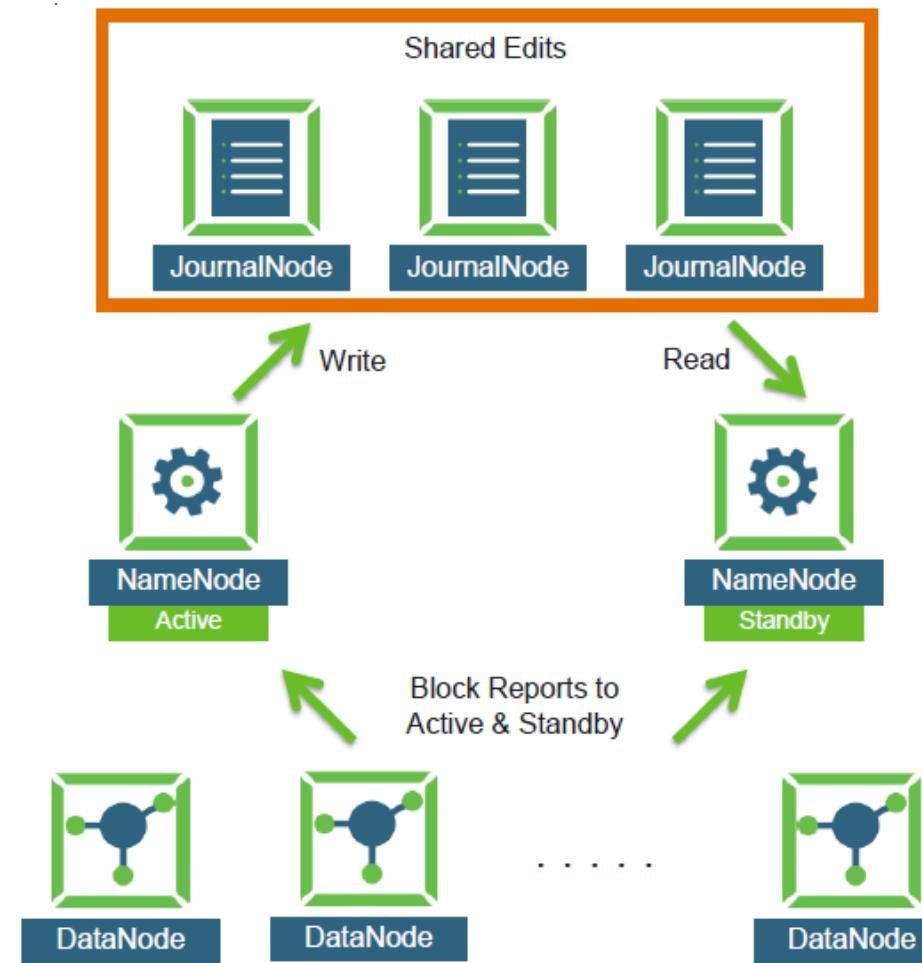


La défaillance du NameNode

► Pour assurer la haute disponibilité du HDFS:

- ◆ Deux NameNodes peuvent être implémentés dans le même cluster Hadoop.
- ◆ Active NameNode: sensible à toutes les opérations client.
- ◆ Standby NameNode conserve un état suffisant pour permettre une bascule rapide.

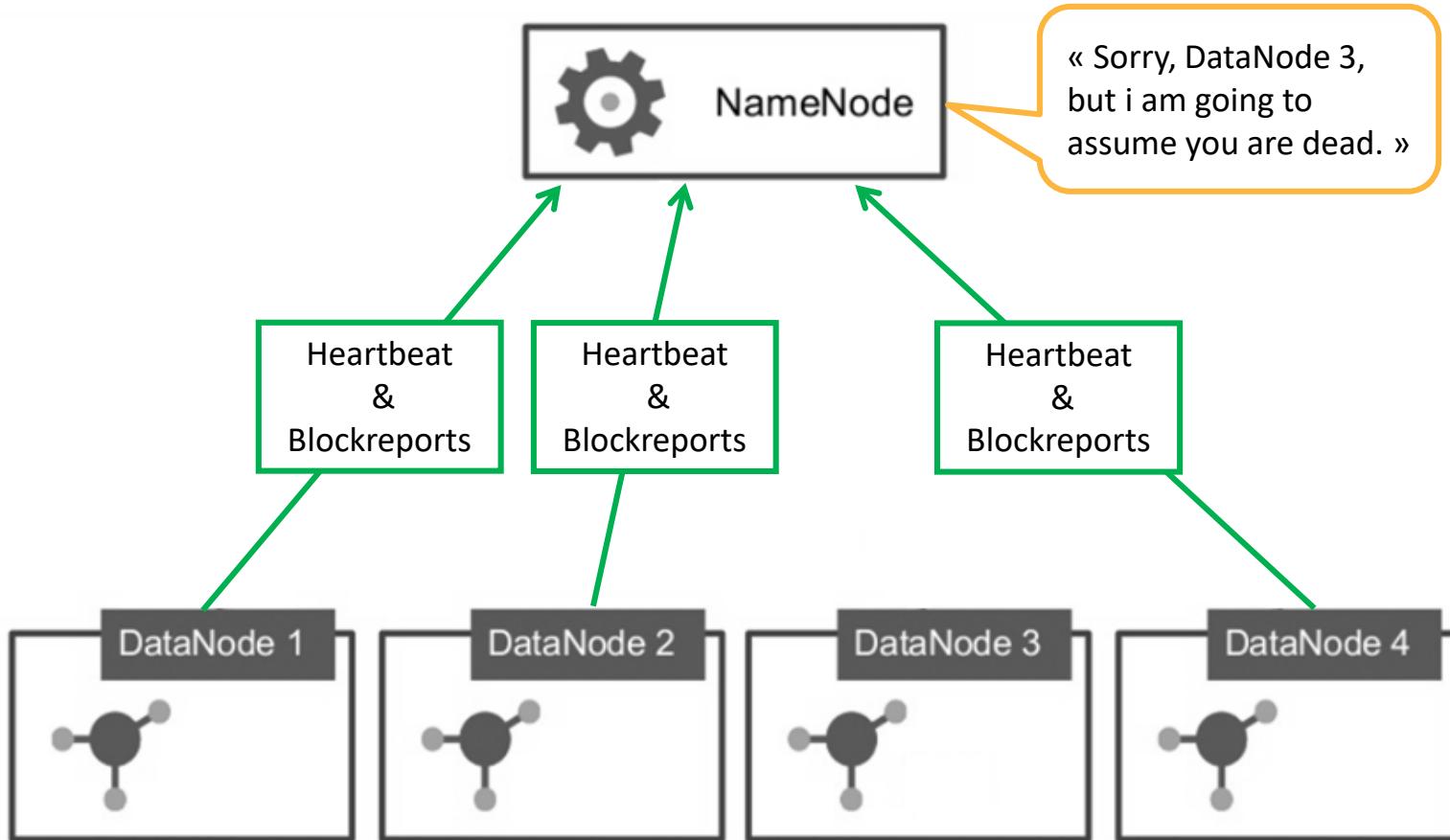
La défaillance du NameNode



La défaillance du NameNode

- ▶ Si un NameNode de secours est utilisé dans un cluster Hadoop, il n'est pas nécessaire d'implémenter un NameNode secondaire:
 - ◆ Le Standby NameNode fera le travail du NameNode secondaire.

La défaillance d'un DataNode



La défaillance d'un DataNode

- ▶ L'objectif principal de HDFS est de stocker les données de manière fiable, même en cas de défaillance.
- ▶ Hadoop est conçu pour permettre une récupération en douceur après une panne de disque ou la défaillance réseau d'un DataNode:
 - ◆ Si un DataNode ne parvient pas à envoyer un heartbeat au NameNode, ce DataNode sera considéré comme mort.

La défaillance d'un DataNode

- ◆ Toutes les données enregistrées dans un DataNode mort ne sont plus disponibles pour le HDFS.
 - ◆ Le NameNode n'enverra plus de nouvelles demandes d'E/S à un DataNode mort et ses blocs seront répliqués dans des DataNodes actifs.
- Une pane de DataNode entraîne généralement le facteur de réPLICATION de certains blocs en dessous de la valeur spécifiée.

La défaillance d'un DataNode

- ▶ Le NameNode surveille en permanence quels blocs doivent être répliqués et lance la réplication chaque fois que cela est nécessaire.
- ▶ Il est possible qu'un bloc de données récupéré à partir d'un DataNode arrive corrompu, à la suite d'une panne de disque ou d'une erreur de réseau.

La corruption de données



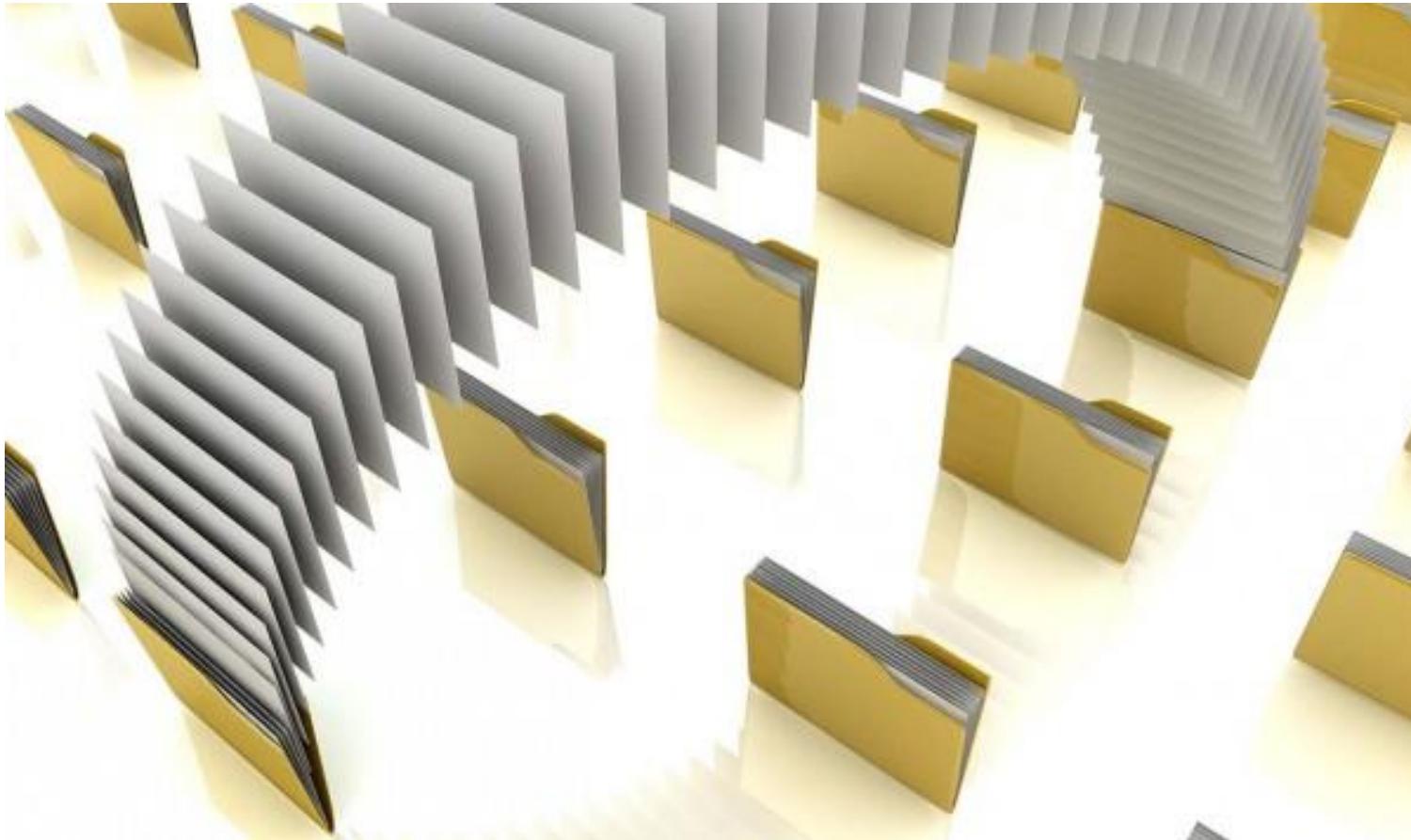
La corruption de données

- ▶ Le HDFS met en œuvre une vérification de la somme de contrôle (checksum) sur le contenu des fichiers HDFS.
- ▶ Lorsqu'un client crée un fichier HDFS, il calcule une somme de contrôle de chaque bloc du fichier et stocke ces sommes de contrôle dans un fichier caché distinct dans le même espace de noms HDFS.

La corruption de données

- ▶ Lorsqu'un client récupère le contenu du fichier, il vérifie que les données qu'il a reçues de chaque DataNode correspondent à la somme de contrôle stockée dans le fichier de somme de contrôle associé.
- ▶ Dans le cas contraire, le client peut choisir de récupérer ce bloc à partir d'un autre DataNode contenant une réplique de ce bloc.

La réPLICATION des blocs



La réPLICATION des blocs

- ▶ Les blocs de données sont répliqués pour assurer la tolérance aux pannes.
- ▶ Le facteur de réPLICATION par défaut est 3, cette propriété est configurable.
- ▶ Une application peut spécifier le nombre de répliques d'un fichier au moment de sa création. Ce nombre peut être modifié à tout moment par la suite.

La réPLICATION des blocs

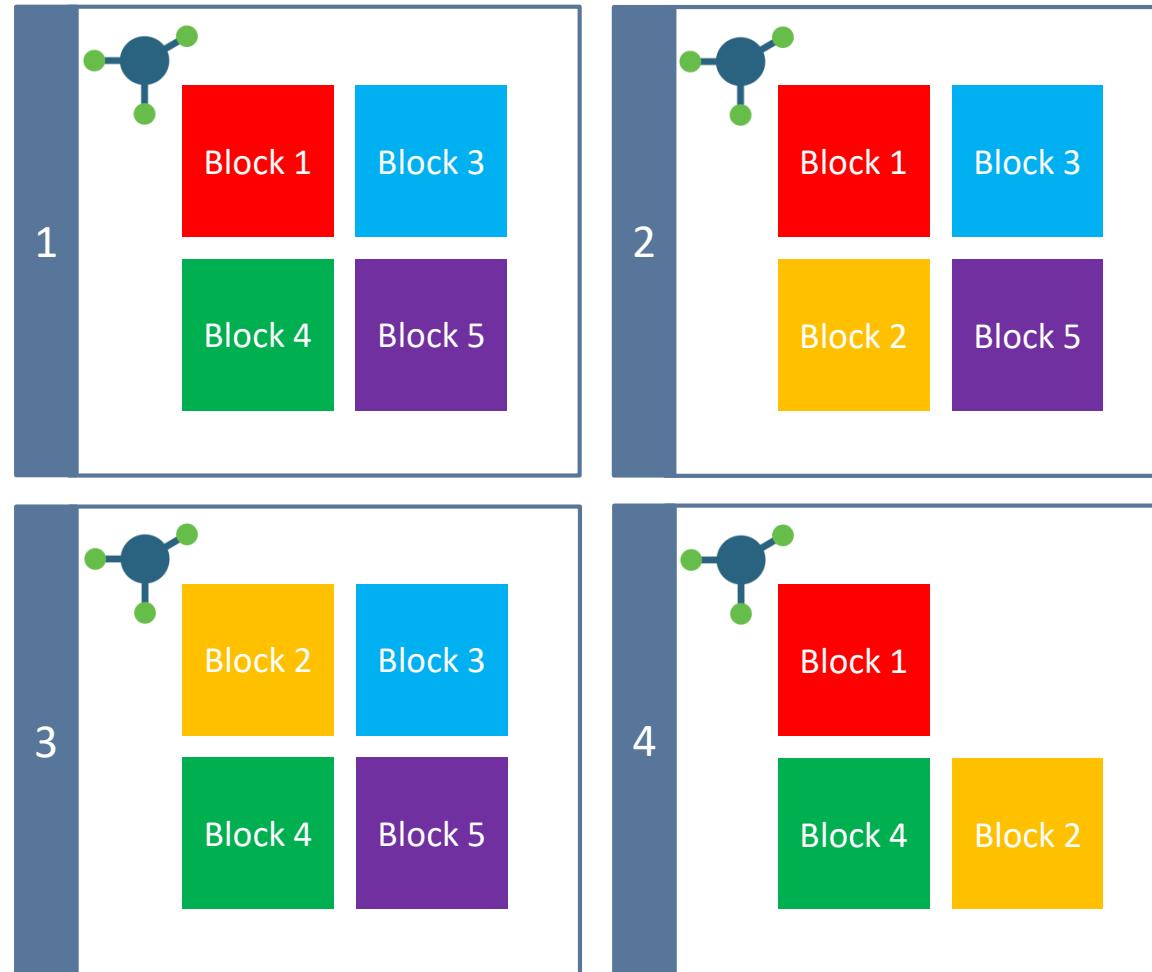
- ▶ Le HDFS utilise un modèle de placement de répliques intelligent pour assurer la fiabilité et garantir les performances.
- ▶ Le NameNode prend toutes les décisions concernant la réPLICATION des blocs.
- ▶ Le NameNode collecte périodiquement le rapport de blocs à partir des DataNodes pour assurer la gestion du facteur de réPLICATION.

La réPLICATION des blocs

- ▶ Chaque fois qu'un bloc est sur-répliqué ou sous-répliqué, le NameNode supprime ou ajoute des répliques selon les besoins.
- ▶ Si la mort d'un DataNode entraîne la baisse du facteur de réPLICATION des blocs de données en dessous de leur valeur minimale, le NameNode initie une réPLICATION supplémentaire pour ramener le facteur de réPLICATION à un état normalisé.

La réPLICATION des blocs

- 4 DataNodes
- Taille du bloc : 128 Mo
- Facteur de réPLICATION : 3
- Taille du fichier : 600 Mo



Le rééquilibrage de blocs



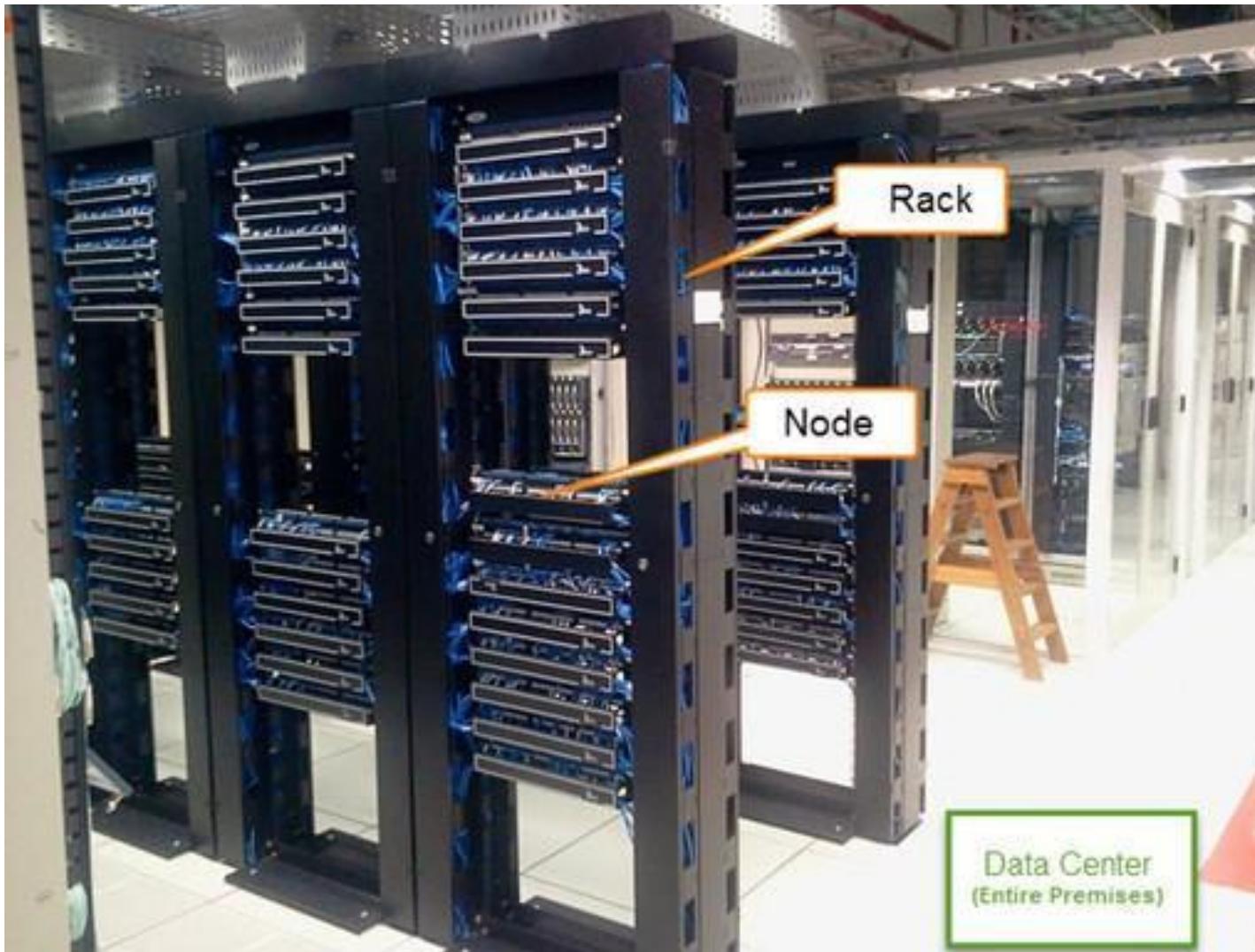
Le rééquilibrage de blocs

- ▶ **L'ajout de nouveaux DataNode à un cluster est une des raisons les plus courantes pour rééquilibrer des blocs.**
- ▶ **Le HDFS prend en charge le rééquilibrage des blocs de données à l'aide de différents modèles:**
 - ◆ Un modèle peut déplacer automatiquement des blocs de données d'un DataNode à un autre si l'espace disponible sur un DataNode devient trop faible.

Le rééquilibrage de blocs

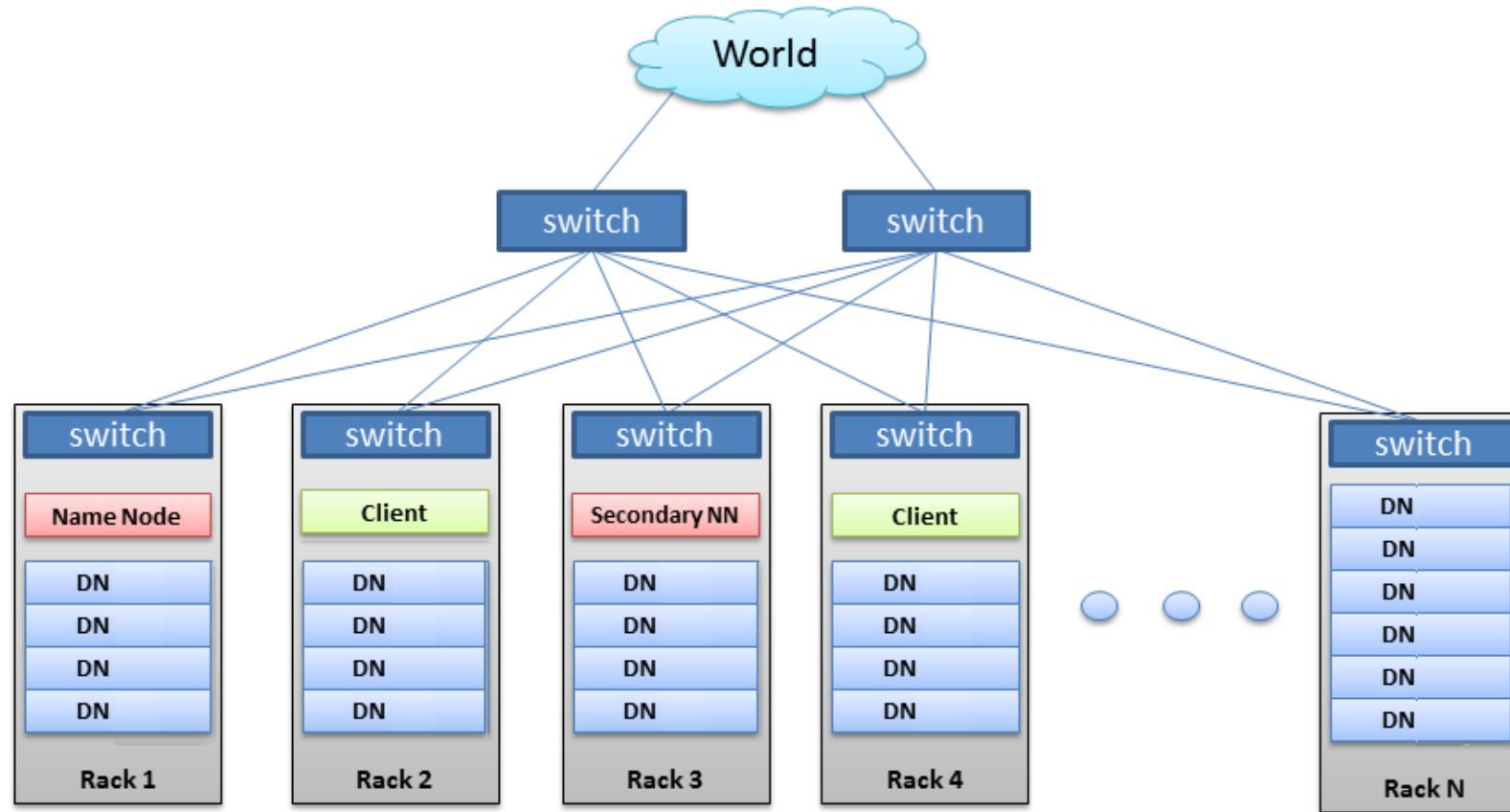
- ◆ Un autre modèle peut créer dynamiquement des répliques supplémentaires et rééquilibrer d'autres blocs de données dans un cluster en cas d'augmentation soudaine de la demande.
- ▶ Le HDFS balancer est un outil permettant d'équilibrer les données sur les périphériques de stockage d'un cluster HDFS.
- ▶ Le HDFS fournit également une commande d'équilibrage pour les tâches de rééquilibrage manuel.

Prise en considération du rack



Prise en considération du rack

Hadoop Cluster



Prise en considération du rack

► La prise en considération du rack consiste à connaître la topologie du cluster ou plus précisément la manière dont les différents DataNodes sont répartis sur les racks d'un cluster Hadoop.

Prise en considération du rack

► L'importance de cette connaissance repose sur l'hypothèse selon laquelle :

- ◆ Les DataNodes co-localisés à l'intérieur d'un rack donné auront plus de bande passante et moins de temps de latence.
- ◆ Deux DataNodes situés dans des racks distincts auront comparativement moins de bande passante et un temps de latence plus élevé.

Prise en considération du rack

- ▶ Le but principal de la prise en considération du rack est :
 - ◆ Augmenter la disponibilité des blocs de données.
 - ◆ Assurer une meilleure performance du cluster.
- ▶ Le NameNode garantit que toutes les répliques ne sont pas stockées sur le même rack ou sur un seul rack.

Prise en considération du rack

- ▶ **L'algorithme de sensibilisation au rack:**
 - ◆ La première réplique d'un bloc sera stockée sur un rack local.
 - ◆ Les deux répliques suivantes seront stockées sur un rack différent, distant, mais sur deux DataNode différents au sein de ce rack.
 - ◆ Le reste des répliques sera placé sur des DataNodes aléatoires à condition que pas plus de deux répliques ne résident sur le même rack, si possible.

Prise en considération du rack

**Pour éviter la perte de données ne
mettez jamais tous vos œufs dans
le même panier!**





|

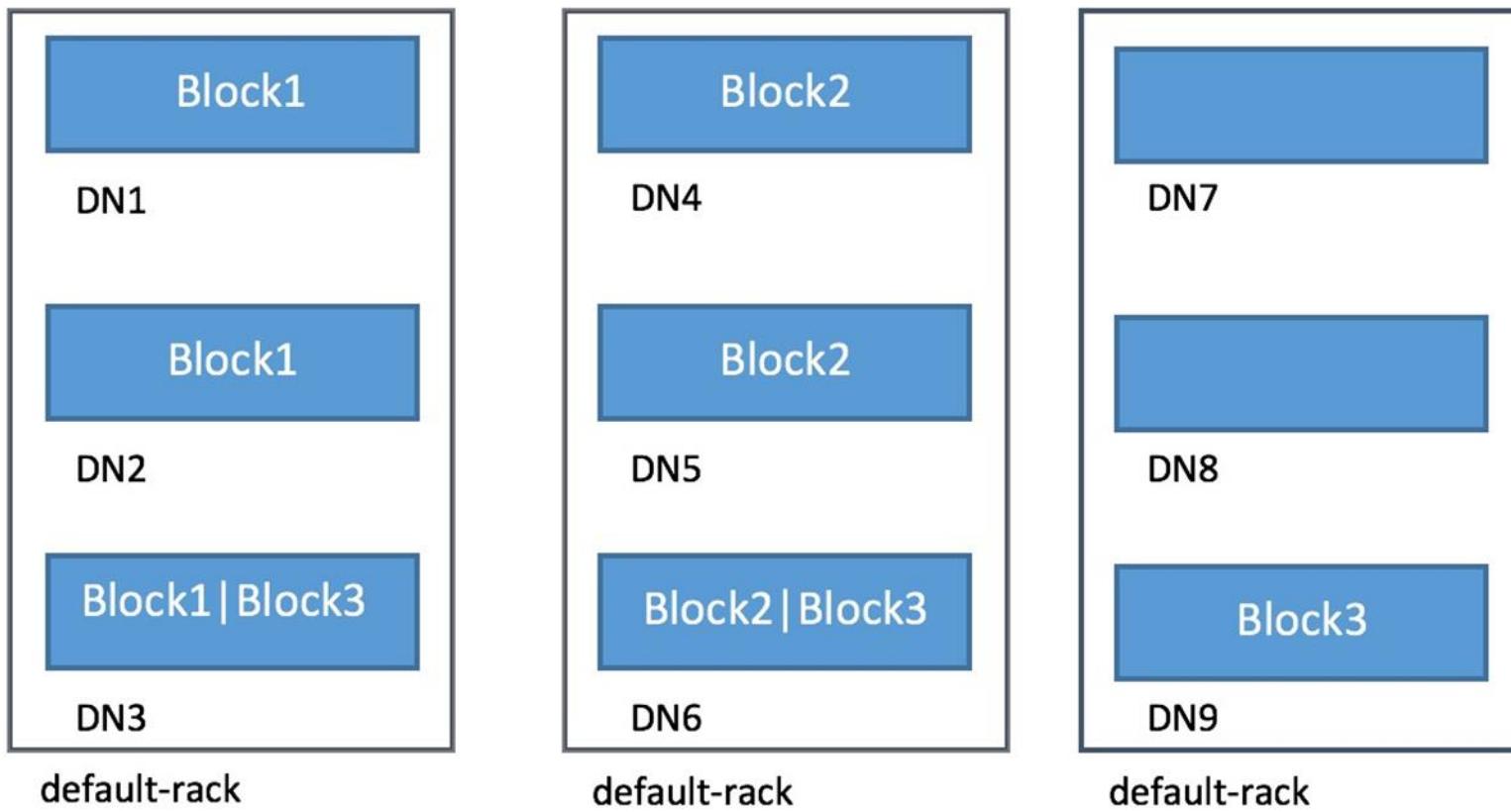
Lab !

Prise en considération du rack

- ▶ Supposons que le cluster dispose de 9 DataNodes avec le facteur de réPLICATION 3.
- ▶ Supposons également qu'il existe 3 racks physiques où les machines sont placées comme suite :
 - ◆ Rack1: DN1;DN2;DN3
 - ◆ Rack2: DN4;DN5;DN6
 - ◆ Rack3: DN7;DN8;DN9

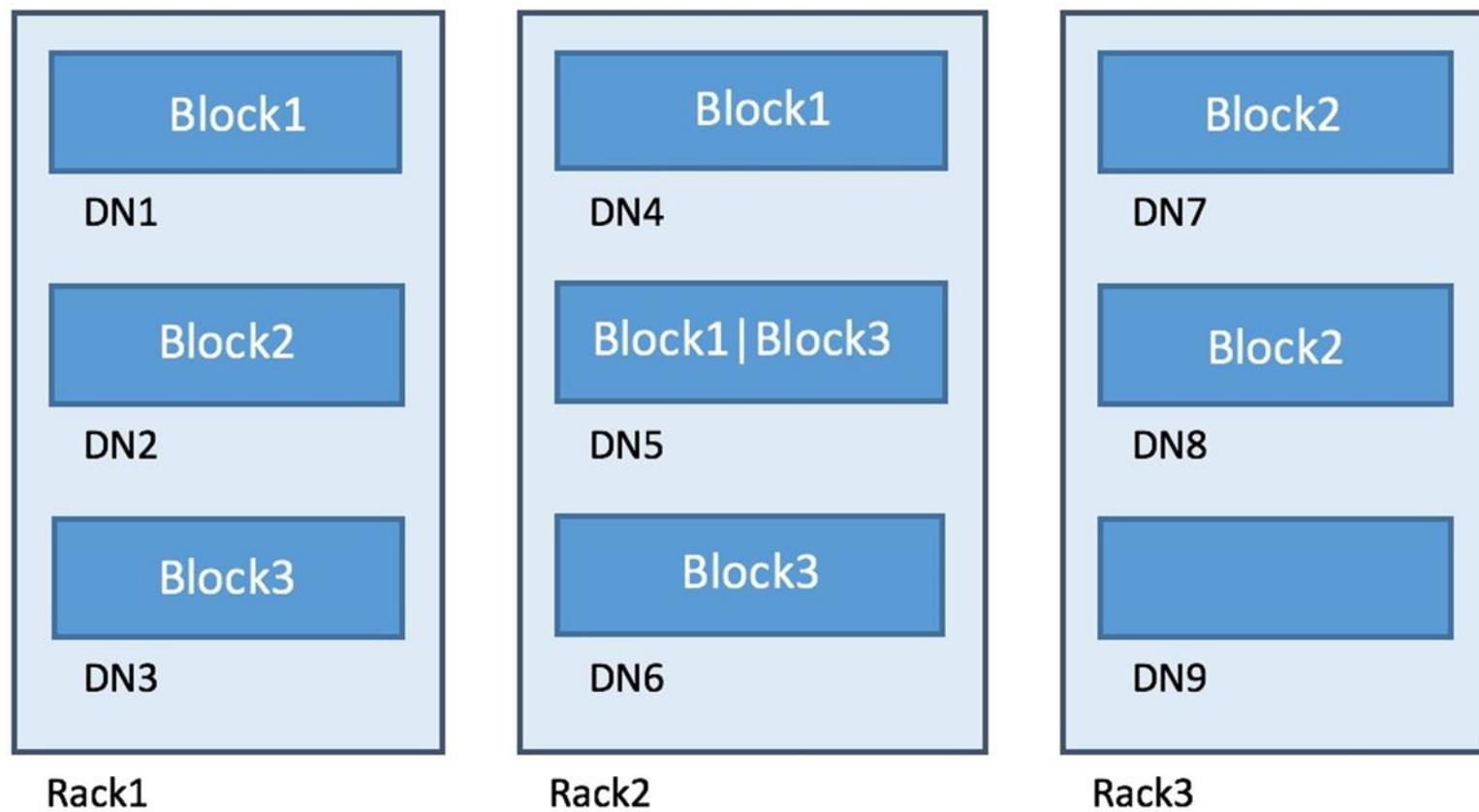
Prise en considération du rack

► Le diagramme suivant illustre un exemple d'emplacement de bloc lorsque HDFS ne prend pas en compte la topologie du rack:

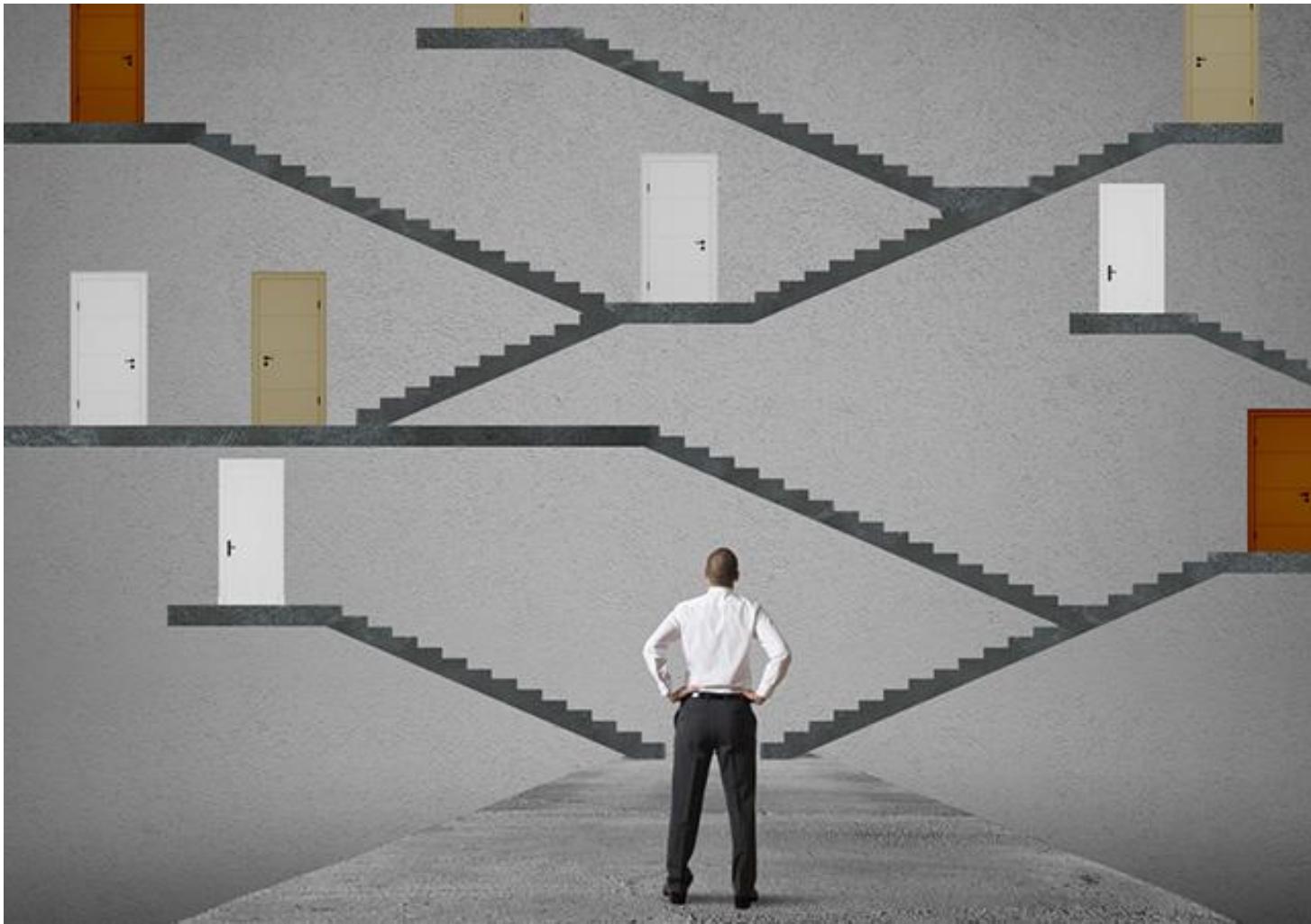


Prise en considération du rack

► Le diagramme suivant illustre un exemple d'emplacement de bloc lorsque HDFS prend en considération la topologie du rack:



La fédération HDFS



La fédération HDFS

- ▶ La fédération améliore une architecture HDFS existante.
- ▶ Robustesse :
 - ◆ Dans l'architecture de base du HDFS, l'intégralité du cluster n'autorise qu'un seul espace de noms.
 - ◆ Dans cette configuration, un seul NameNode gère l'espace de noms.
 - ◆ Si le NameNode tombe en panne, le cluster dans son ensemble serait hors service. Le cluster sera indisponible jusqu'à la réparation du NameNode.

► Scalabilité :

- ◆ L'espace de noms n'est pas évolutif à l'instar de l'espace de stockage.
- ◆ Il est impossible d'ajouter plus d'espace de noms à un cluster existant.
- ◆ La seule possibilité est de redimensionner l'espace de noms verticalement sur un seul NameNode.

► Performance :

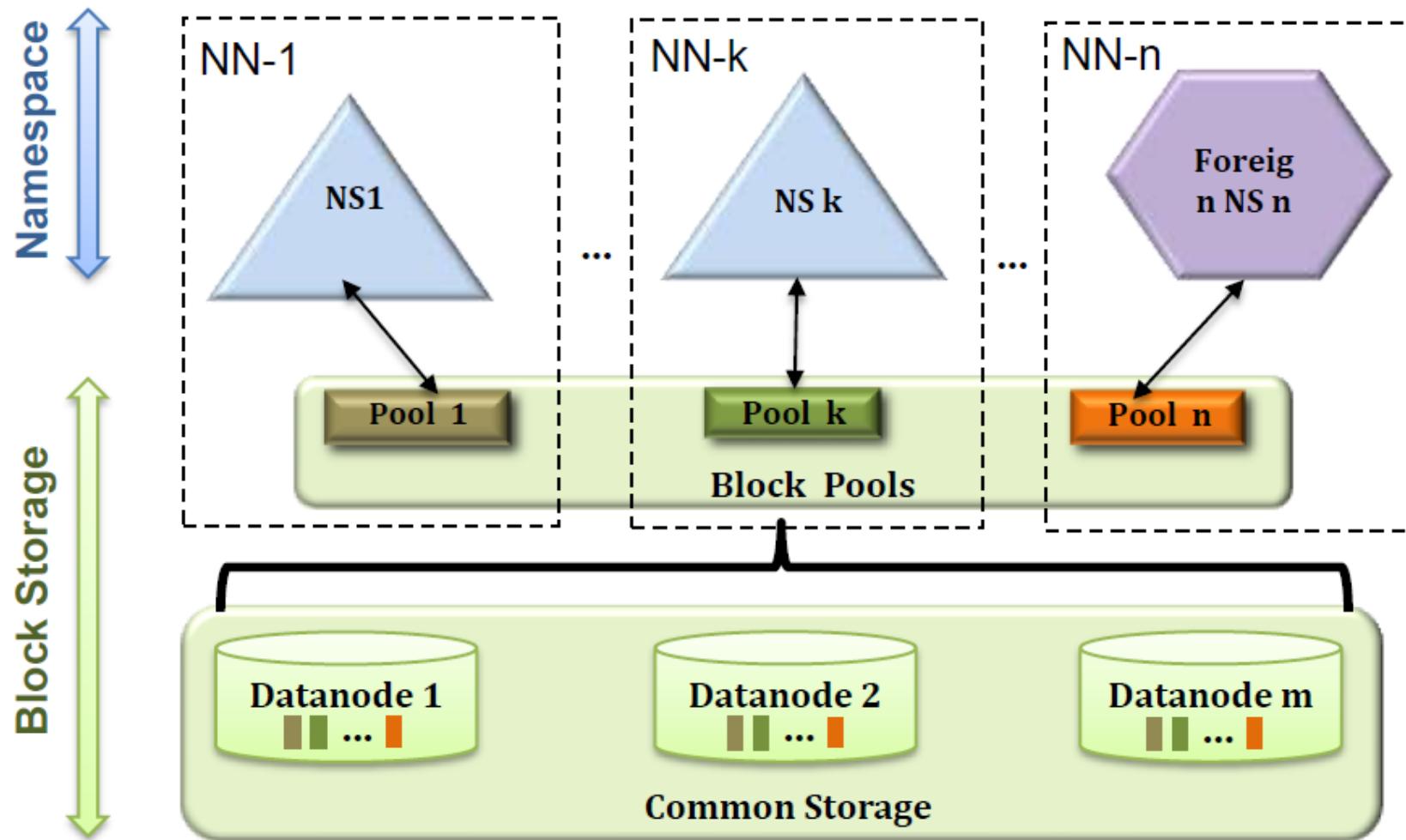
- ◆ La performance du cluster Hadoop dépendent du débit du NameNode.
- ◆ Une opération du système de fichiers dépend du débit d'un seul NameNode.
- ◆ Un NameNode prend en charge jusqu'à 60000 tâches simultanées.

► Isolation :

- ◆ Il n'y a pas de séparation de l'espace de noms.
- ◆ Il n'y a donc pas d'isolation entre les organisations / départements / projets / équipes clientes qui utilisent le cluster Hadoop.

La fédération surmonte cette limitation en ajoutant la prise en charge de plusieurs NameNode / Namespaces au HDFS.

La fédération HDFS



La fédération HDFS

► Dans l'architecture de fédération HDFS:

- ◆ Les DataNodes sont utilisés comme stockage commun pour les blocs gérés par tous les NameNodes.
- ◆ Chaque DataNode s'inscrit avec tous les NameNodes du cluster.
- ◆ Les DataNodes envoient des heartbeats, des rapports de bloc et traitent les commandes des NameNodes.
- ◆ Le heartbeat véhicule l'information sur l'espace total sur le DataNode utilisé par les pools de blocs, ainsi que par des données non HDFS.

La fédération HDFS

- ◆ Plusieurs NameNodes (NN1, NN2, ..., NNn) gèrent respectivement plusieurs espaces de noms (NS1, NS2, ..., NSn).
- ◆ les NameNodes sont indépendants et ne nécessitent pas de coordination les uns avec les autres.
- ◆ Chaque espace de noms a son propre pool de blocs (NS1 possède le pool 1, etc.) et l'ensemble des blocs d'un pool de blocs appartient à un seul espace de noms.
- ◆ Les blocs du pool 1 sont stockés dans les DataNode1, DataNode2, ..., DataNodem.

La gestion des droits sous HDFS

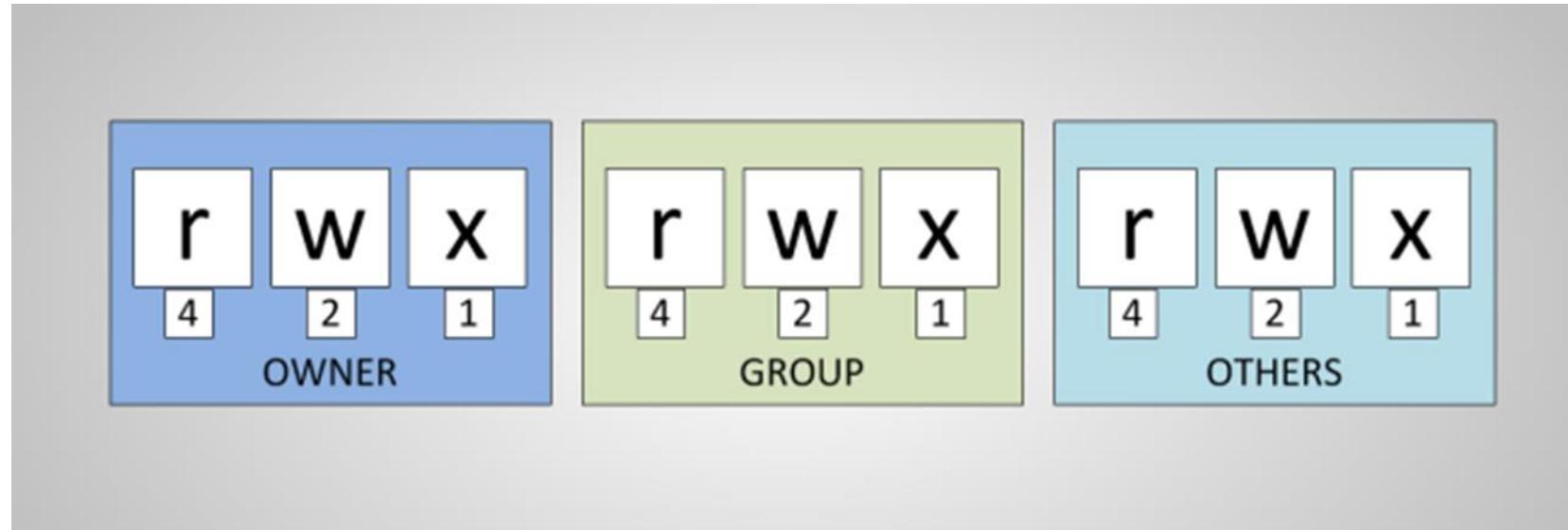


La gestion des droits sous HDFS

- ▶ Le HDFS implémente un modèle d'autorisations pour les fichiers et les répertoires partageant une grande partie du modèle POSIX (Portable Operating System Interface).
- ▶ Chaque fichier et répertoire est associé à un propriétaire et à un groupe.
- ▶ Le modèle d'autorisations HDFS prend en charge:
 - ◆ Lire (r)
 - ◆ Ecrire (w)
 - ◆ Exécuter (x)

La gestion des droits sous HDFS

► Le fichier ou le répertoire dispose d'autorisations distinctes pour l'utilisateur propriétaire, pour les autres utilisateurs membres du groupe et pour tous les autres utilisateurs.



La gestion des droits sous HDFS

► Pour les fichiers:

- ◆ La permission **r** est requise pour lire le fichier.
- ◆ La permission **w** est requise pour écrire ou ajouter du contenu au fichier.

► Pour les répertoires:

- ◆ La permission **r** est requise pour lister le contenu du répertoire.
- ◆ La permission **w** est requise pour créer ou supprimer des fichiers ou des répertoires.
- ◆ La permission **x** est requise pour accéder à un répertoire fils du répertoire.

Les commandes hdfs



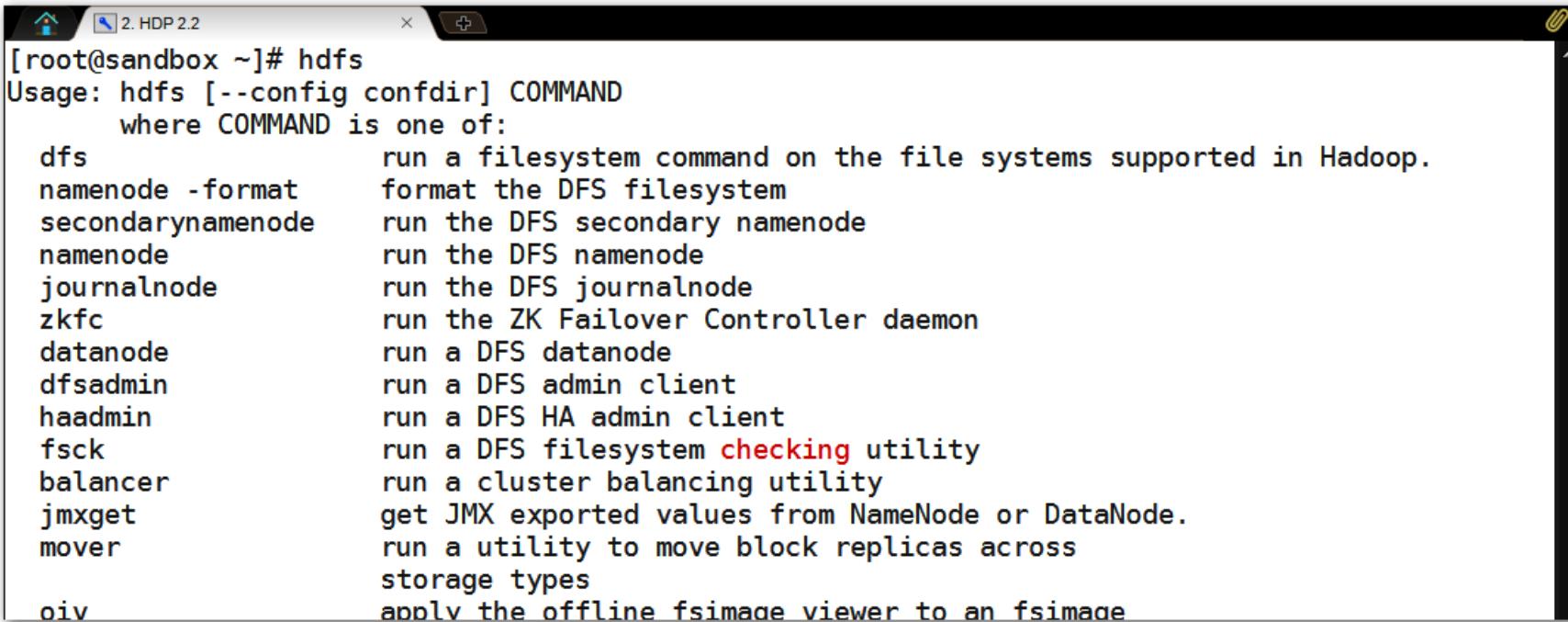
Les commandes hdfs

- ▶ L'application hdfs est une application cliente Hadoop qui permet d'émettre des commandes à HDFS à partir d'une ligne de commande.
- ▶ Toutes les commandes hadoop sont appelées par le script /usr/bin/hdfs.
- ▶ L'exécution du script hdfs sans aucun argument imprime la description de toutes les commandes.

Les commandes hdfs

► La commande hdfs a la syntaxe suivante:

◆ **hdfs [COMMAND] [GENERIC_OPTIONS] [COMMAND_OPTIONS]**

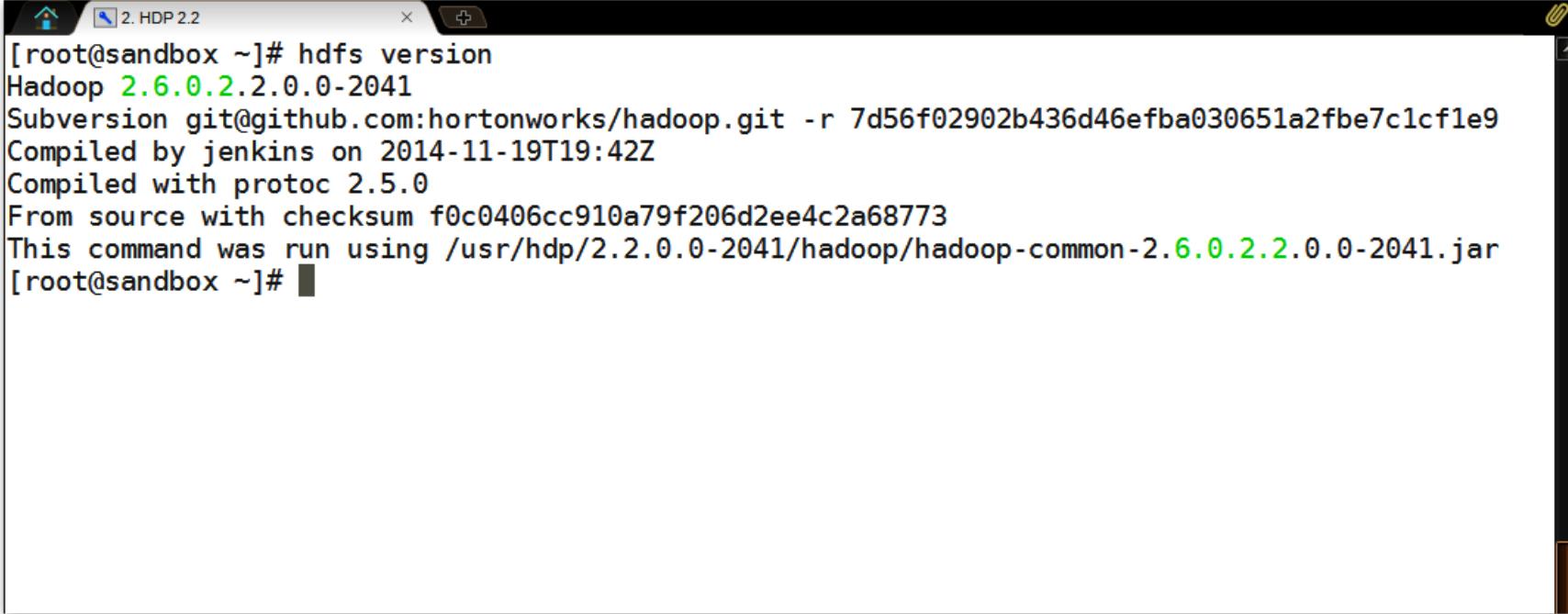


The screenshot shows a terminal window titled "2. HDP 2.2" running on a root shell. The command "hdfs" is entered, and the system displays its usage information. The usage text is as follows:

```
[root@sandbox ~]# hdfs
Usage: hdfs [--config confdir] COMMAND
  where COMMAND is one of:
    dfs          run a filesystem command on the file systems supported in Hadoop.
    namenode -format      format the DFS filesystem
    secondarynamenode   run the DFS secondary namenode
    namenode        run the DFS namenode
    journalnode     run the DFS journalnode
    zkfc           run the ZK Failover Controller daemon
    datanode        run a DFS datanode
    dfsadmin        run a DFS admin client
    haadmin         run a DFS HA admin client
    fsck            run a DFS filesystem checking utility
    balancer        run a cluster balancing utility
    jmxget          get JMX exported values from NameNode or DataNode.
    mover           run a utility to move block replicas across
                    storage types
    oiv             apply the offline fsimage viewer to an fsimage
```

Lab - Les commandes hdfs

- ▶ La commande hdfs version permet d'afficher la version du HDFS.



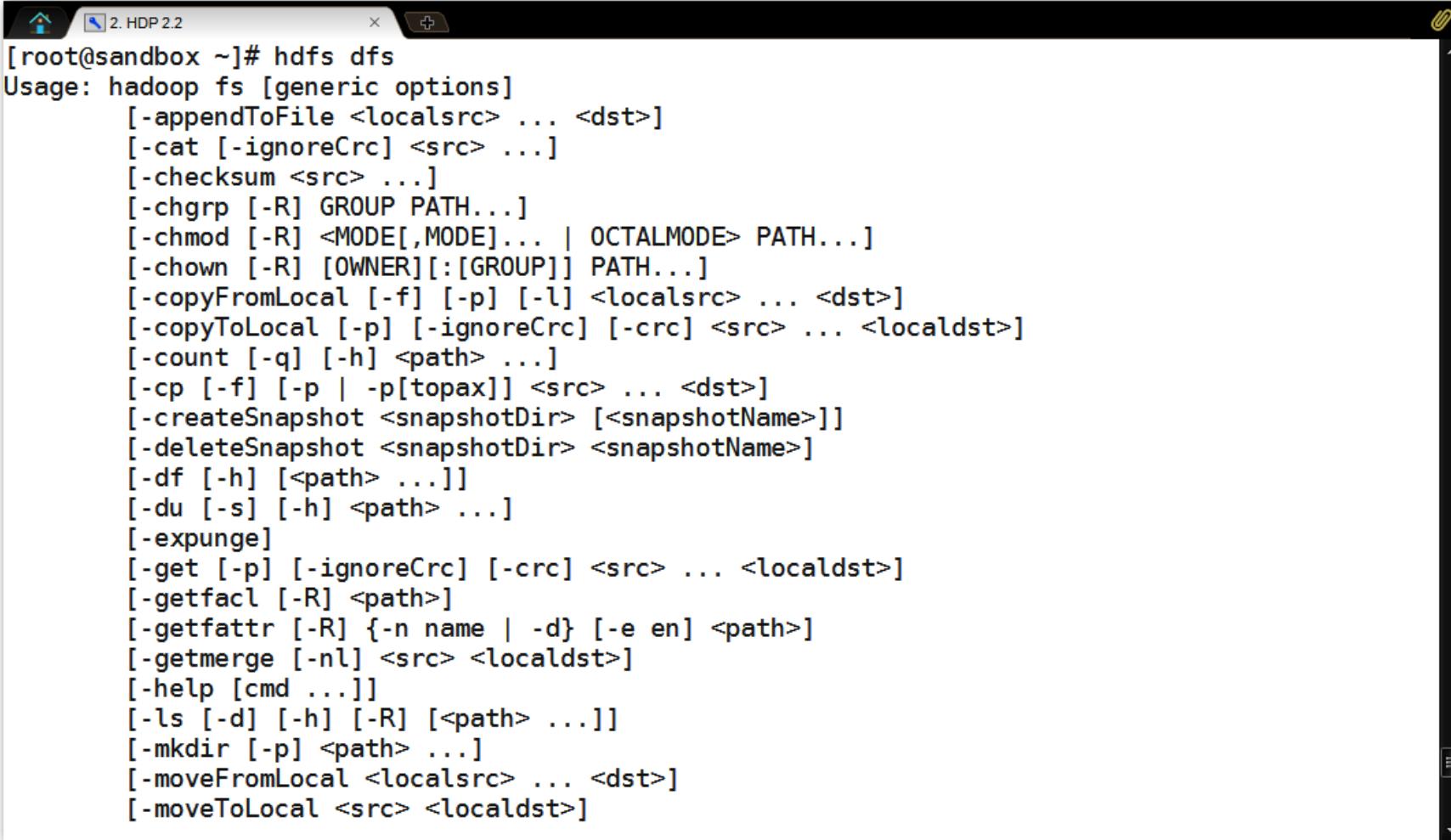
The screenshot shows a terminal window titled "2. HDP 2.2". The command "hdfs version" is run, displaying the following information:

```
[root@sandbox ~]# hdfs version
Hadoop 2.6.0.2.2.0.0-2041
Subversion git@github.com:hortonworks/hadoop.git -r 7d56f02902b436d46efba030651a2fbe7c1cf1e9
Compiled by jenkins on 2014-11-19T19:42Z
Compiled with protoc 2.5.0
From source with checksum f0c0406cc910a79f206d2ee4c2a68773
This command was run using /usr/hdp/2.2.0.0-2041/hadoop/hadoop-common-2.6.0.2.2.0.0-2041.jar
[root@sandbox ~]#
```

Les commandes hdfs

- ▶ La commande `dfs` permet d'exécuter une commande de système de fichiers sur le système de fichiers pris en charge dans Hadoop.
- ▶ La commande `dfs` a la syntaxe suivante:
 - ◆ `hdfs dfs [COMMAND [CMD_OPTIONS]]`
- ▶ L'exécution de la commande `dfs` sans aucun argument imprime la description de toutes les commandes.

Les commandes hdfs



```
[root@sandbox ~]# hdfs dfs
Usage: hadoop fs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-cat [-ignoreCrc] <src> ...]
      [-checksum <src> ...]
      [-chgrp [-R] GROUP PATH...]
      [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
      [-chown [-R] [OWNER][:[GROUP]] PATH...]
      [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
      [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-count [-q] [-h] <path> ...]
      [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [-h] [<path> ...]]
      [-du [-s] [-h] <path> ...]
      [-expunge]
      [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-getfacl [-R] <path>]
      [-getattr [-R] {-n name | -d} [-e en] <path>]
      [-getmerge [-nl] <src> <localdst>]
      [-help [cmd ...]]
      [-ls [-d] [-h] [-R] [<path> ...]]
      [-mkdir [-p] <path> ...]
      [-moveFromLocal <localsrc> ... <dst>]
      [-moveToLocal <src> <localdst>]
```

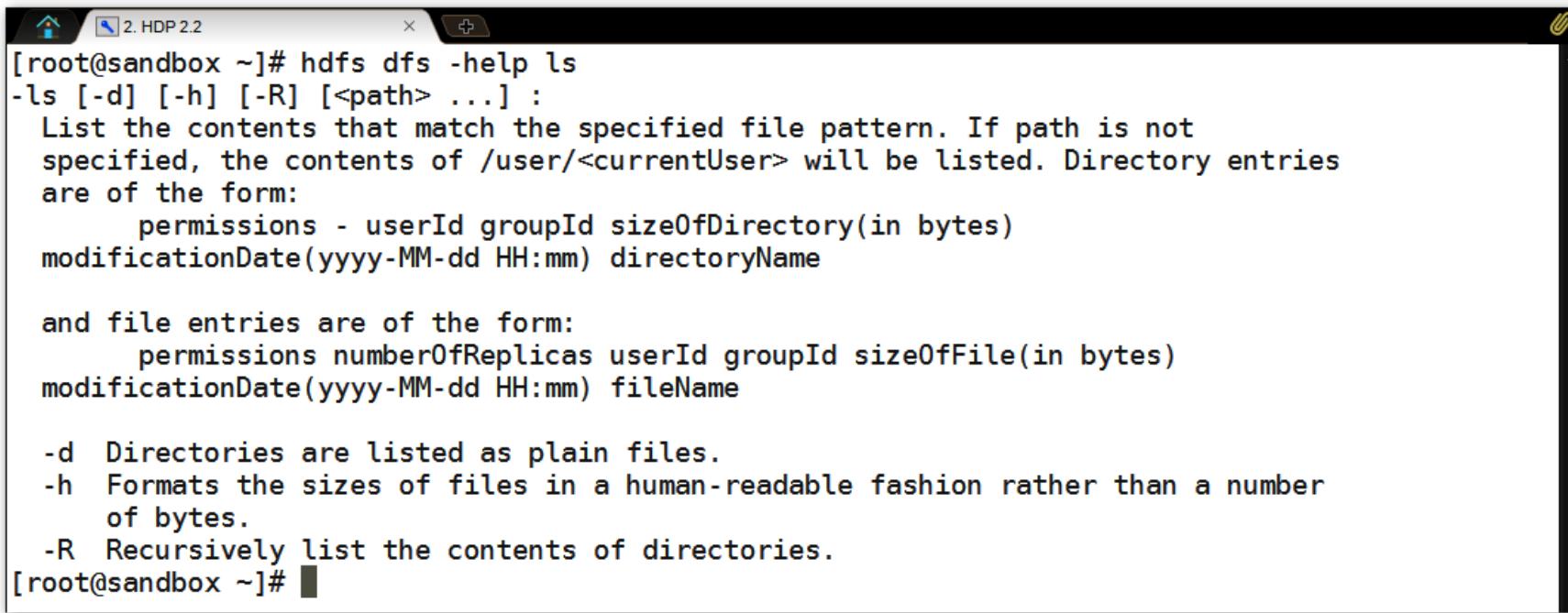
Les commandes hdfs

```
[-mv <src> ... <dst>]
[-put [-f] [-p] [-l] <localsrc> ... <dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r|-R] [-skipTrash] <src> ...]
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>] | [--set <acl_spec> <path>]]
[-setattr {-n name [-v value] | -x name} <path>]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]
[-tail [-f] <file>]
[-test -[defsz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touchz <path> ...]
[-usage [cmd ...]]]

Generic options supported are
-conf <configuration file>      specify an application configuration file
-D <property=value>              use value for given property
-fs <local|namenode:port>        specify a namenode
-jt <local|resourcemanager:port>  specify a ResourceManager
-files <comma separated list of files>  specify comma separated files to be copied to the map
                                         reduce cluster
-libjars <comma separated list of jars>   specify comma separated jar files to include in the
                                         classpath.
-archives <comma separated list of archives>  specify comma separated archives to be unarchiv
                                         ed on the compute machines.
```

Les commandes hdfs

► Utilisez l'option help pour avoir la description d'une commande hdfs dfs.



```
[root@sandbox ~]# hdfs dfs -help ls
-ls [-d] [-h] [-R] [<path> ...] :
  List the contents that match the specified file pattern. If path is not
  specified, the contents of /user/<currentUser> will be listed. Directory entries
  are of the form:
    permissions - userId groupId sizeOfDirectory(in bytes)
    modificationDate(yyyy-MM-dd HH:mm) directoryName

  and file entries are of the form:
    permissions numberOfReplicas userId groupId sizeOfFile(in bytes)
    modificationDate(yyyy-MM-dd HH:mm) fileName

  -d Directories are listed as plain files.
  -h Formats the sizes of files in a human-readable fashion rather than a number
    of bytes.
  -R Recursively list the contents of directories.
[root@sandbox ~]#
```

Les commandes hdfs

► Consultez le guide de référence des commandes Hadoop HDFS pour plus de détails:

- ◆ <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html>



Les commandes utilisateur dfs

►Créer des répertoires ou des fichiers dans le HDFS :

- ◆ ***mkdir <chemin>*** : Crée un répertoire dans le HDFS. Si l'option -p est utilisée crée tous les répertoires parents dans le chemin manquant.
- ◆ ***touch <chemin>*** : Crée un fichier sur le chemin contenant l'heure actuelle sous forme d'horodatage. Échec si un fichier existe déjà dans le chemin d'accès, à moins que le fichier ait déjà la taille 0.

Les commandes utilisateur dfs

► Copier ou déplacer des fichiers ou des répertoires dans le HDFS:

- ◆ `cp <src> <dest>` : Copie le fichier ou le répertoire ou le répertoire identifié par src vers dest, dans le HDFS.
- ◆ `mv <src> <dest>` : Déplace le fichier ou le répertoire indiqué par src vers dest, dans le HDFS.

► Lister des fichiers et des répertoires du HDFS:

◆ ***ls <chemin>*** : Répertorie le contenu du répertoire spécifié par chemin, en indiquant les noms, les autorisations, le propriétaire, la taille et la date de modification de chaque entrée. L'utilisation de l'option R permet d'afficher de manière récursive des entrées dans tous les sous-répertoires du chemin.

Les commandes utilisateur dfs

► Mettre des fichiers dans le HDFS:

- ◆ ***put <localSrc> <dest>*** : Copie le fichier ou le répertoire du système de fichiers local identifié par localSrc dans le HDFS sous dest.
- ◆ ***copyFromLocal <localSrc> <dest>*** : Identique à -put.
- ◆ ***moveFromLocal <localSrc> <dest>*** : Copie le fichier ou le répertoire du système de fichiers local identifié par localSrc dans dest dans le HDFS, puis supprime la copie locale en cas de succès.

Les commandes utilisateur dfs

► Copier des fichiers ou des répertoires sur le système de fichiers local :

- ◆ *get <src> <localDest>* : Copie le fichier ou le répertoire dans le HDFS identifié par src dans le chemin du système de fichiers local identifié par localDest.
- ◆ *copyToLocal <src> <localDest>* : Identique à -get.
- ◆ *getmerge <src> <localDest>* : Récupère tous les fichiers qui correspondent au chemin src dans HDFS et les copie dans un seul fichier fusionné dans le système de fichiers local identifié par localDest.

Les commandes utilisateur dfs

► Afficher le contenu d'un fichier du HDFS au format texte:

- ◆ *cat <nom de fichier>* : Affiche le contenu du fichier sur la sortie standard.
- ◆ *tail [-f] <nom de fichier>* : Affiche le dernier Ko du fichier sur la sortie standard. L'option -f permet d'afficher les données ajoutées à mesure que le fichier grandit.

Les commandes utilisateur dfs

► Ajouter à un fichier:

◆ *appendToFile <localsrc> <dst>* : La commande **appendToFile** est utilisée pour ajouter un ou plusieurs fichiers du système de fichiers local à un fichier HDFS.

Les commandes utilisateur dfs

► Supprimer des fichiers et des répertoires du HDFS:

- ◆ ***rm <chemin>*** : Supprime le fichier ou le répertoire vide identifié par chemin.
- ◆ ***rmr <chemin>*** : Supprime le fichier ou le répertoire identifié par chemin. Supprime récursivement toutes les entrées enfants (c'est-à-dire les fichiers ou les sous-répertoires du chemin).

Les commandes utilisateur dfs

► Compter des fichiers et des répertoires:

- ◆ ***count [-q] <chemin>*** : Compte le nombre de répertoires, fichiers et octets sous le chemin dans le HDFS.
- ◆ ***stat [format] <chemin>*** : Imprime des informations sur le chemin. Le format est une chaîne accepte la taille de fichier en blocs (%b), nom de fichier (%n), taille de bloc (%o), réPLICATION (%r) et date de modification (% y,%Y).

Les commandes utilisateur dfs

► Définir le facteur de réPLICATION de fichiers :

◆ *setrep [-R] [-w] rep <chemin>* : Définit le facteur de réPLICATION cible pour les fichiers identifiés par chemin d'accès à valeur de rep.

Les commandes utilisateur dfs

► Changement du propriétaire, des groupes de fichiers ou répertoires du HDFS:

- ◆ ***chown [-R] [propriétaire] [:[groupe]] <chemin>***: Définit l'utilisateur et / ou les groupes propriétaires des fichiers ou des répertoires identifiés par chemin. Définit le propriétaire de manière récursive si -R est spécifié.
- ◆ ***chgrp [-R] groupe <chemin>*** : Définit le groupe propriétaire des fichiers ou des répertoires identifiés par chemin. Définit le groupe de manière récursive si -R est spécifié.

Les commandes utilisateur dfs

► Changement des permissions des fichiers du HDFS:

◆ *chmod [-R] mode, mode, ... <chemin>* : Modifie les autorisations de fichiers associées à un ou plusieurs objets identifiés par chemin. Effectue les modifications de manière récursive avec R. Le mode est un mode octal à 3 chiffres ou {ugo} +/- {rwxX}.



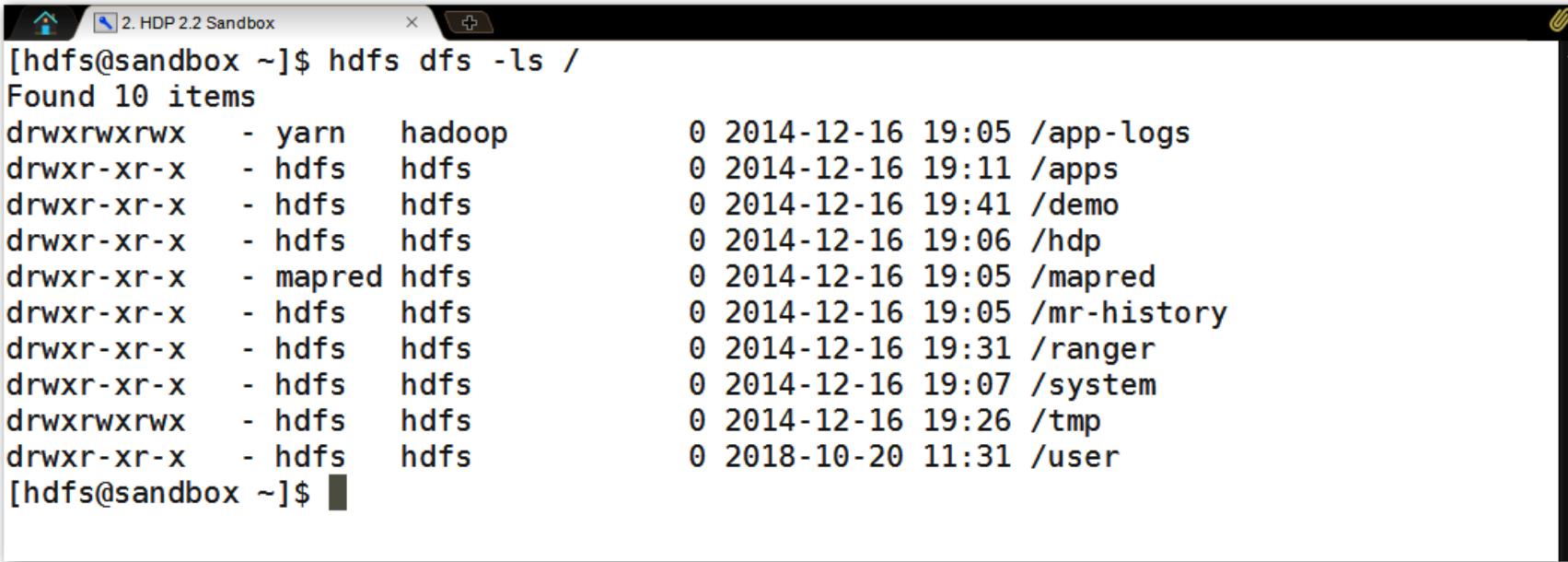
|

Lab !

Lab - Les commandes hdfs dfs

- ▶ **hdfs dfs -ls /** : affiche ce qu'il y a à la racine du HDFS. Vous pouvez descendre inspecter les dossiers que vous voyez.
- ▶ Il n'y a pas de commande équivalente à cd, parce qu'il n'y a pas de notion de dossier courant dans HDFS, donc à chaque fois, il faut remettre le chemin complet.

Lab - Les commandes hdfs dfs

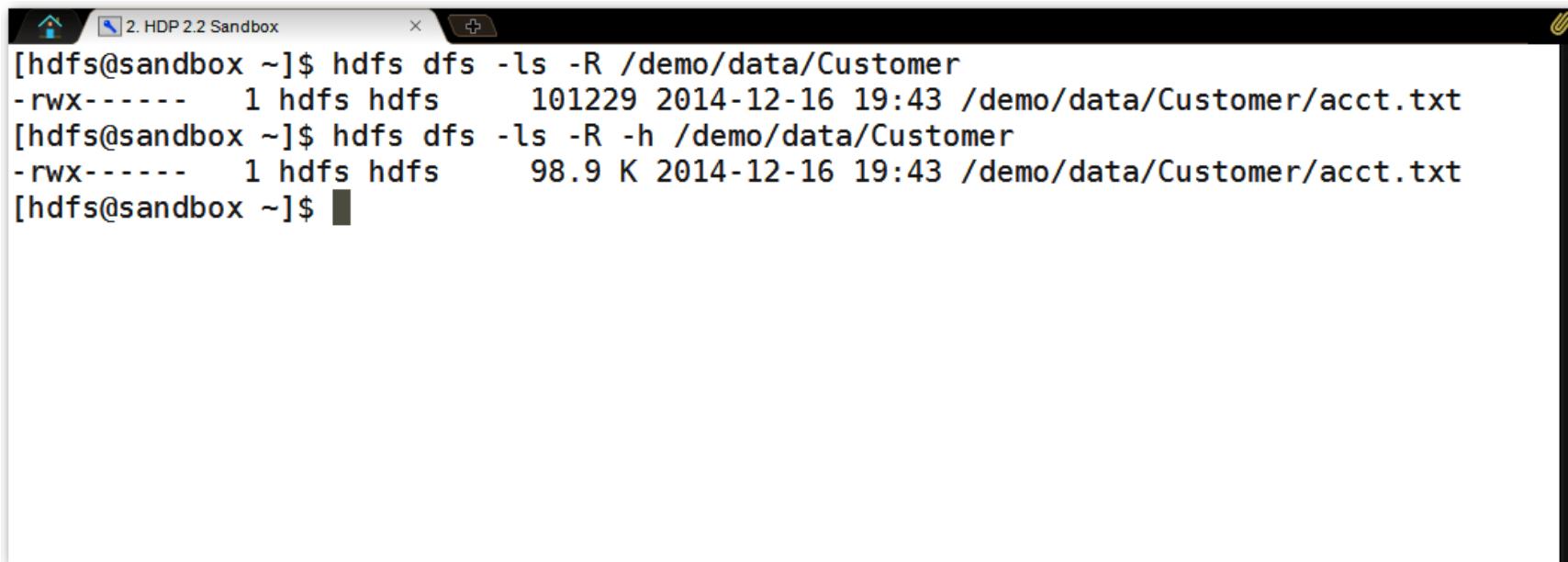


The screenshot shows a terminal window titled "2. HDP 2.2 Sandbox". The command entered is "hdfs dfs -ls /". The output lists 10 items in the root directory:

File Type	Owner	Group	Last Modified	Last Accessed	Path
drwxrwxrwx	-	yarn	hadoop	0	2014-12-16 19:05 /app-logs
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:11 /apps
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:41 /demo
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:06 /hdp
drwxr-xr-x	-	mapred	hdfs	0	2014-12-16 19:05 /mapred
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:05 /mr-history
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:31 /ranger
drwxr-xr-x	-	hdfs	hdfs	0	2014-12-16 19:07 /system
drwxrwxrwx	-	hdfs	hdfs	0	2014-12-16 19:26 /tmp
drwxr-xr-x	-	hdfs	hdfs	0	2018-10-20 11:31 /user

Lab - Les commandes hdfs dfs

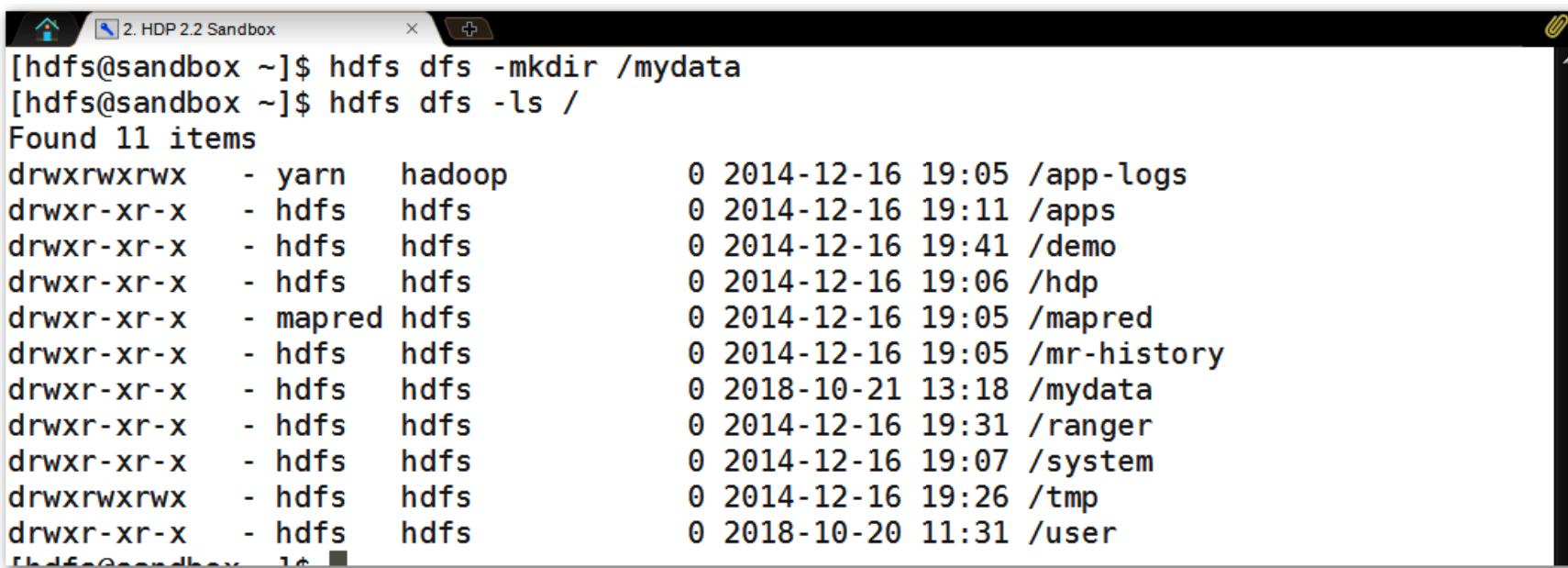
► **hdfs dfs -ls -R -h /var : affiche les fichiers des sous-dossiers, avec une taille arrondie en Ko, Mo ou Go.**



```
[hdfs@sandbox ~]$ hdfs dfs -ls -R /demo/data/Customer
-rwx----- 1 hdfs hdfs 101229 2014-12-16 19:43 /demo/data/Customer/acct.txt
[hdfs@sandbox ~]$ hdfs dfs -ls -R -h /demo/data/Customer
-rwx----- 1 hdfs hdfs 98.9 K 2014-12-16 19:43 /demo/data/Customer/acct.txt
[hdfs@sandbox ~]$
```

Lab - Les commandes hdfs dfs

► **hdfs dfs -mkdir /mydata:** crée un dossier dans le répertoire racine du HDFS. (la taille d'un dossier sera toujours 0).

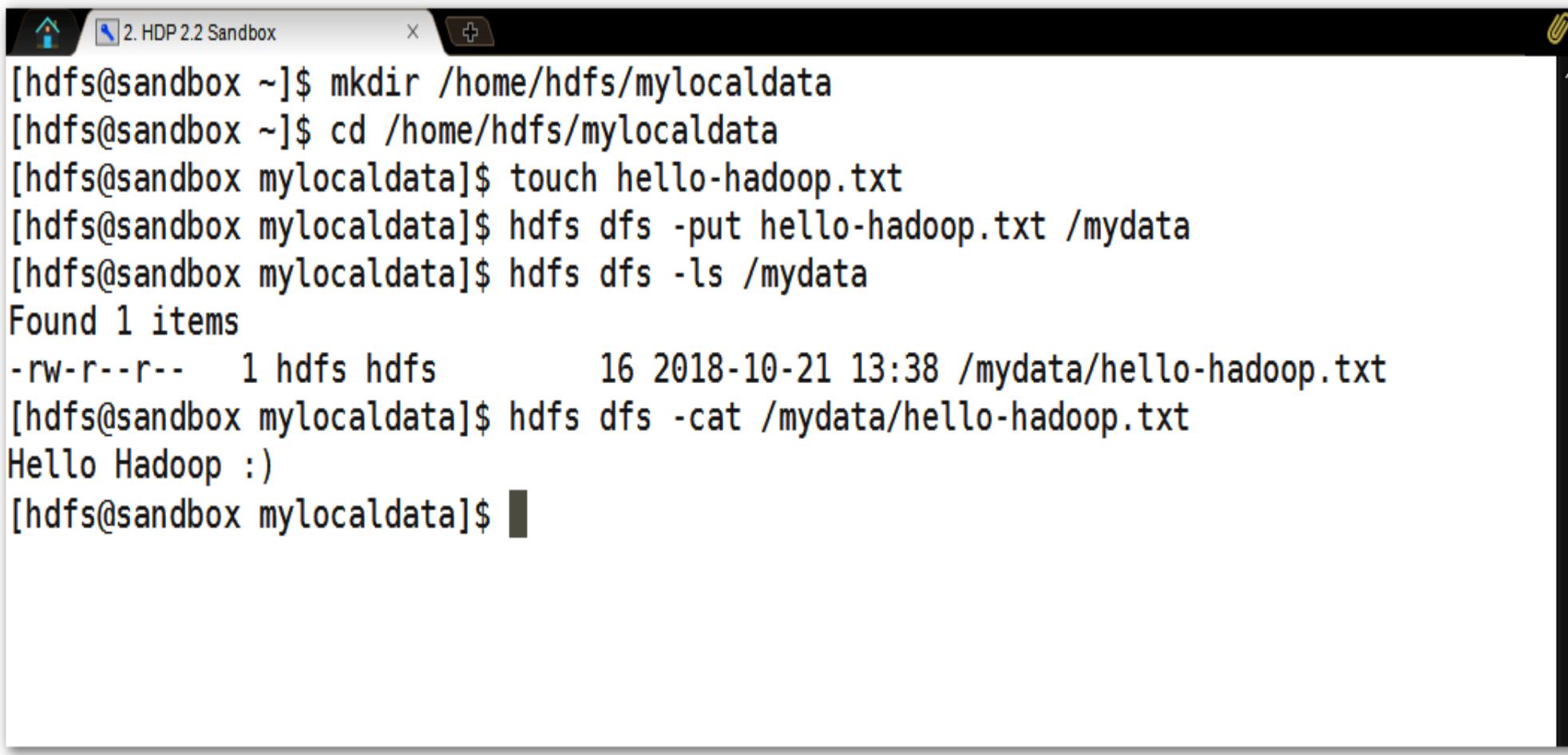


```
[hdfs@sandbox ~]$ hdfs dfs -mkdir /mydata
[hdfs@sandbox ~]$ hdfs dfs -ls /
Found 11 items
drwxrwxrwx  - yarn  hadoop          0 2014-12-16 19:05 /app-logs
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:11 /apps
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:41 /demo
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:06 /hdp
drwxr-xr-x  - mapred hdfs         0 2014-12-16 19:05 /mapred
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:05 /mr-history
drwxr-xr-x  - hdfs  hdfs           0 2018-10-21 13:18 /mydata
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:31 /ranger
drwxr-xr-x  - hdfs  hdfs           0 2014-12-16 19:07 /system
drwxrwxrwx  - hdfs  hdfs           0 2014-12-16 19:26 /tmp
drwxr-xr-x  - hdfs  hdfs           0 2018-10-20 11:31 /user
```

Lab - Les commandes hdfs dfs

- ▶ Créez un fichier appelé `hello-hadoop.txt` dans le compte Linux de l'utilisateur `training` et contenant la phrase « Hello Hadoop ».
- ▶ Copier ce fichier sur HDFS avec la commande `hdfs dfs -put`: `hdfs dfs -put /home/training/hello-hadoop.txt /mydata`.
- ▶ Utilisez la commande `hdfs dfs -ls -R /mydata` pour vérifier le résultat.
- ▶ La commande `hdfs dfs -cat /mydata/hello-hadoop.txt` affiche le contenu du fichier.

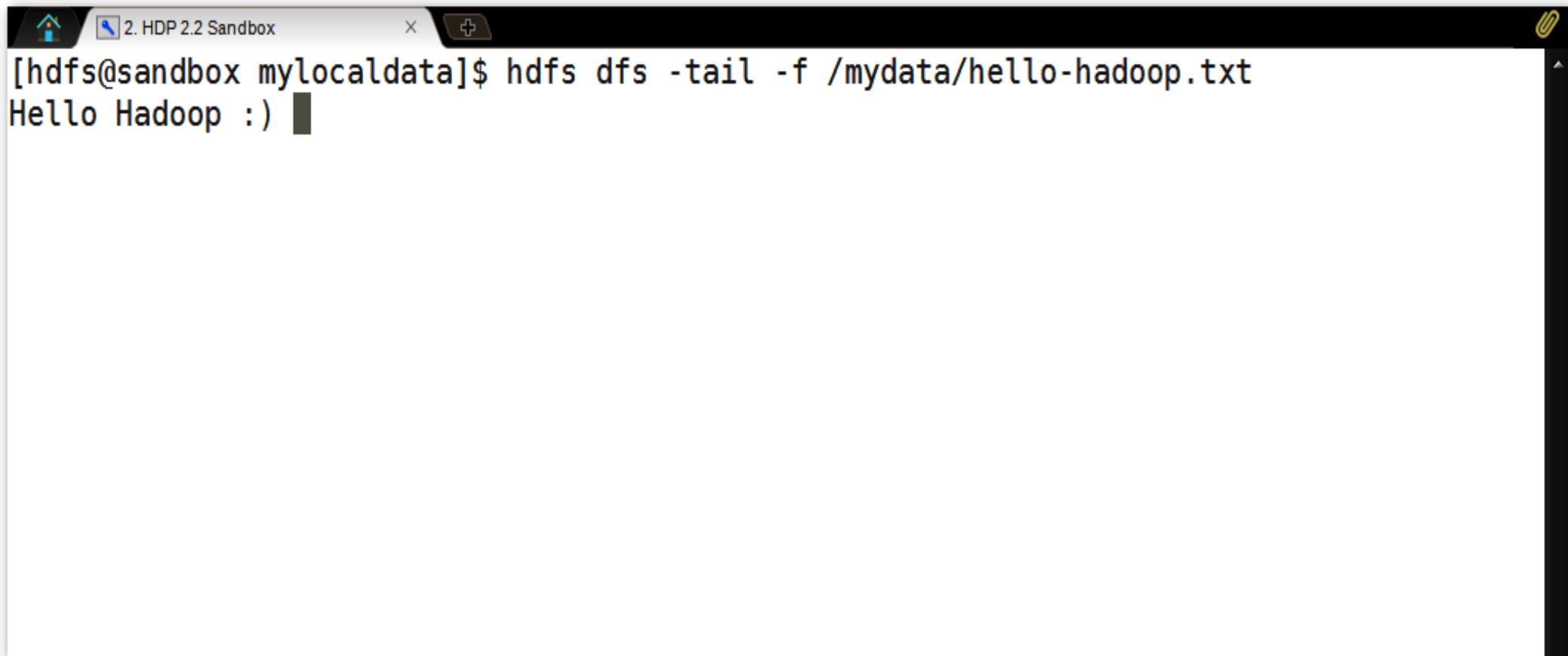
Lab - Les commandes hdfs dfs



```
[hdfs@sandbox ~]$ mkdir /home/hdfs/mylocaldata
[hdfs@sandbox ~]$ cd /home/hdfs/mylocaldata
[hdfs@sandbox mylocaldata]$ touch hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -put hello-hadoop.txt /mydata
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata
Found 1 items
-rw-r--r-- 1 hdfs hdfs      16 2018-10-21 13:38 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -cat /mydata/hello-hadoop.txt
Hello Hadoop :)
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

► La commande **hdfs dfs -tail /mydata/readme.txt** affiche le dernier Ko du fichier.

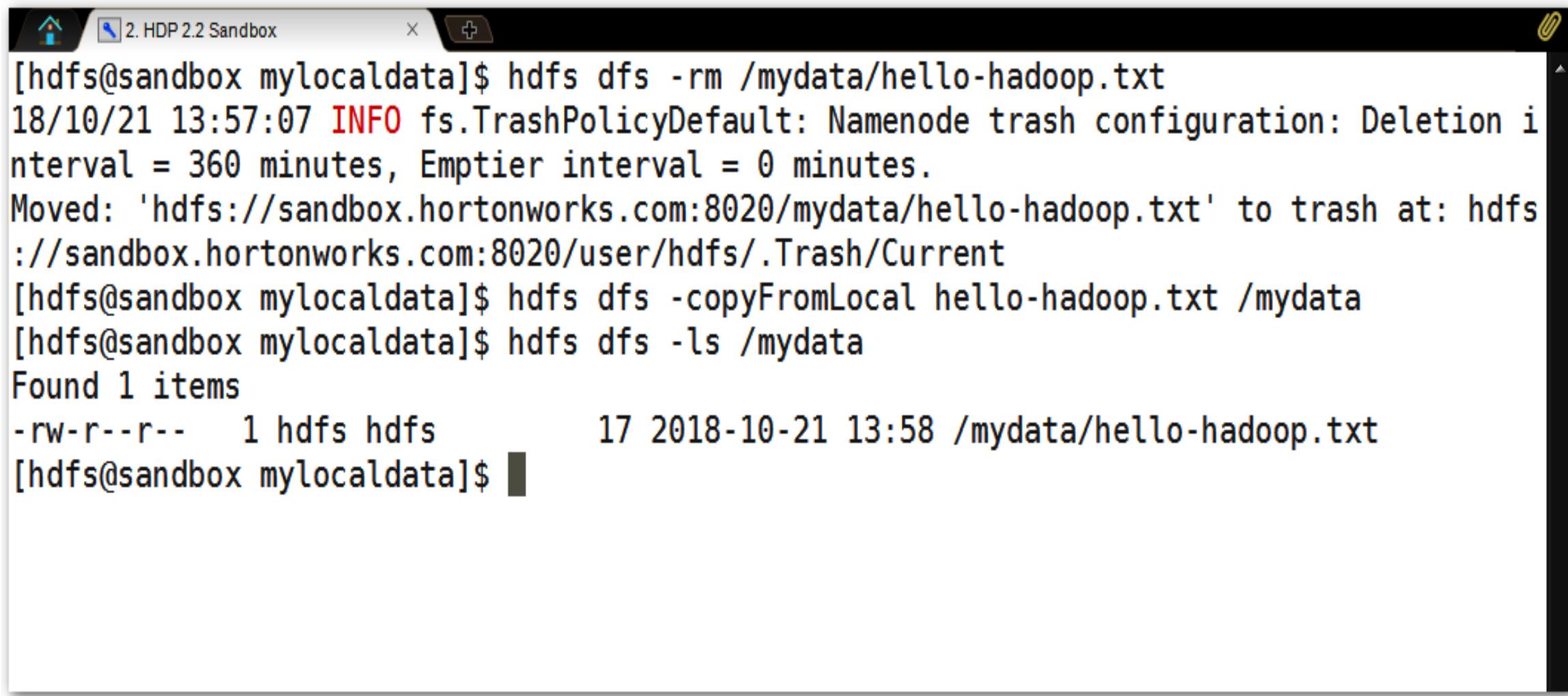


```
[hdfs@sandbox mylocaldata]$ hdfs dfs -tail -f /mydata/hello-hadoop.txt
Hello Hadoop :)
```

Lab - Les commandes hdfs dfs

- ▶ Supprimer ce fichier de HDFS avec la commande `hdfs dfs -rm /mydata/hello-hadoop.txt`.
- ▶ Remettre à nouveau ce fichier avec la commande `hdfs dfs -copyFromLocal /mydata/hello-hadoop.txt` (vérifier le résultat avec `hdfs dfs -ls`).

Lab - Les commandes hdfs dfs

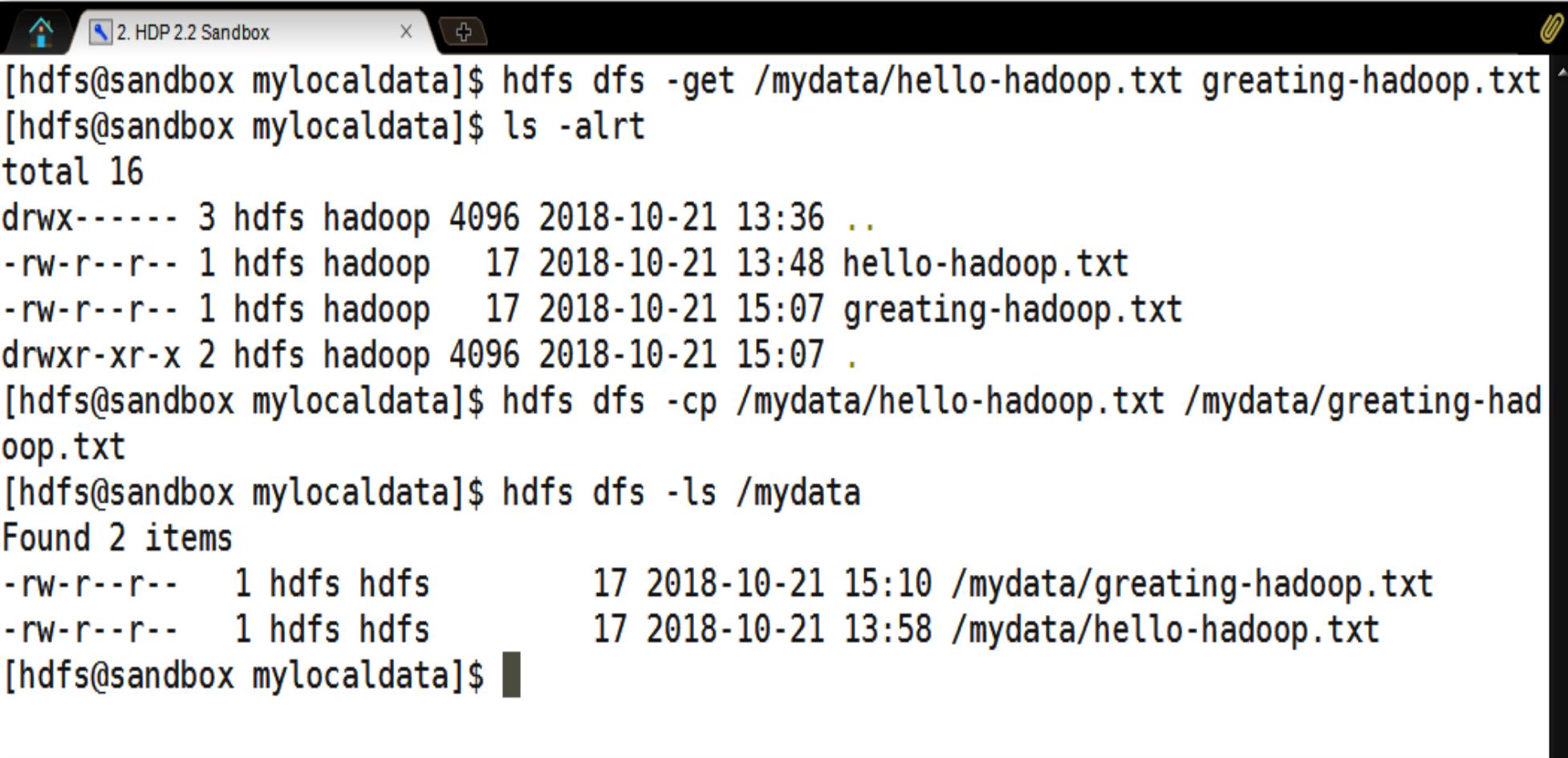


```
[hdfs@sandbox mylocaldata]$ hdfs dfs -rm /mydata/hello-hadoop.txt
18/10/21 13:57:07 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://sandbox.hortonworks.com:8020/mydata/hello-hadoop.txt' to trash at: hdfs://sandbox.hortonworks.com:8020/user/hdfs/.Trash/Current
[hdfs@sandbox mylocaldata]$ hdfs dfs -copyFromLocal hello-hadoop.txt /mydata
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata
Found 1 items
-rw-r--r-- 1 hdfs hdfs 17 2018-10-21 13:58 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

- ▶ Transférer le fichier **hello-hadoop** de HDFS vers le compte Linux en lui changeant son nom avec la commande `hdfs dfs -get /mydata/hello-hadoop.txt greeting-hadoop.txt` .
- ▶ Copier le fichier `hello-hadoop.txt` avec la commande `hdfs dfs -cp /mydata/hello-hadoop.txt /mydata/greeting-hadoop.txt`.

Lab - Les commandes hdfs dfs



```
[hdfs@sandbox mylocaldata]$ hdfs dfs -get /mydata/hello-hadoop.txt greeting-hadoop.txt
[hdfs@sandbox mylocaldata]$ ls -alrt
total 16
drwx----- 3 hdfs hadoop 4096 2018-10-21 13:36 ..
-rw-r--r-- 1 hdfs hadoop 17 2018-10-21 13:48 hello-hadoop.txt
-rw-r--r-- 1 hdfs hadoop 17 2018-10-21 15:07 greeting-hadoop.txt
drwxr-xr-x 2 hdfs hadoop 4096 2018-10-21 15:07 .
[hdfs@sandbox mylocaldata]$ hdfs dfs -cp /mydata/hello-hadoop.txt /mydata/greeting-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata
Found 2 items
-rw-r--r-- 1 hdfs hdfs 17 2018-10-21 15:10 /mydata/greeting-hadoop.txt
-rw-r--r-- 1 hdfs hdfs 17 2018-10-21 13:58 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

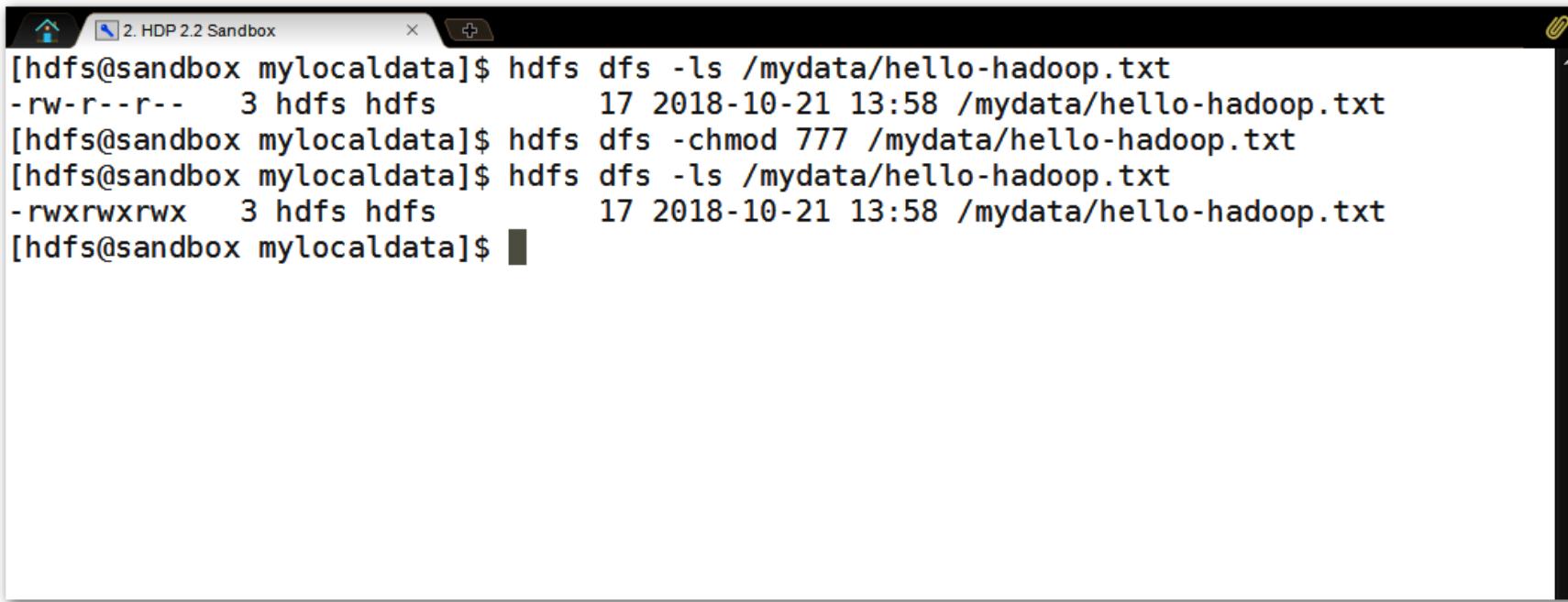
► Positionner le facteur de réPLICATION à 3 pour le fichier **hello-hadoop.txt**. Vérifier avec la commande **stat**.



```
[hdfs@sandbox mylocaldata]$ hdfs dfs -setrep 3 /mydata/hello-hadoop.txt
Replication 3 set: /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -stat %r /mydata/hello-hadoop.txt
3
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

► Changer les permissions sur le fichier hello-hadoop.txt à 777 avec la commande hdfs dfs -chmod 777 /mydata/hello-hadoop.txt

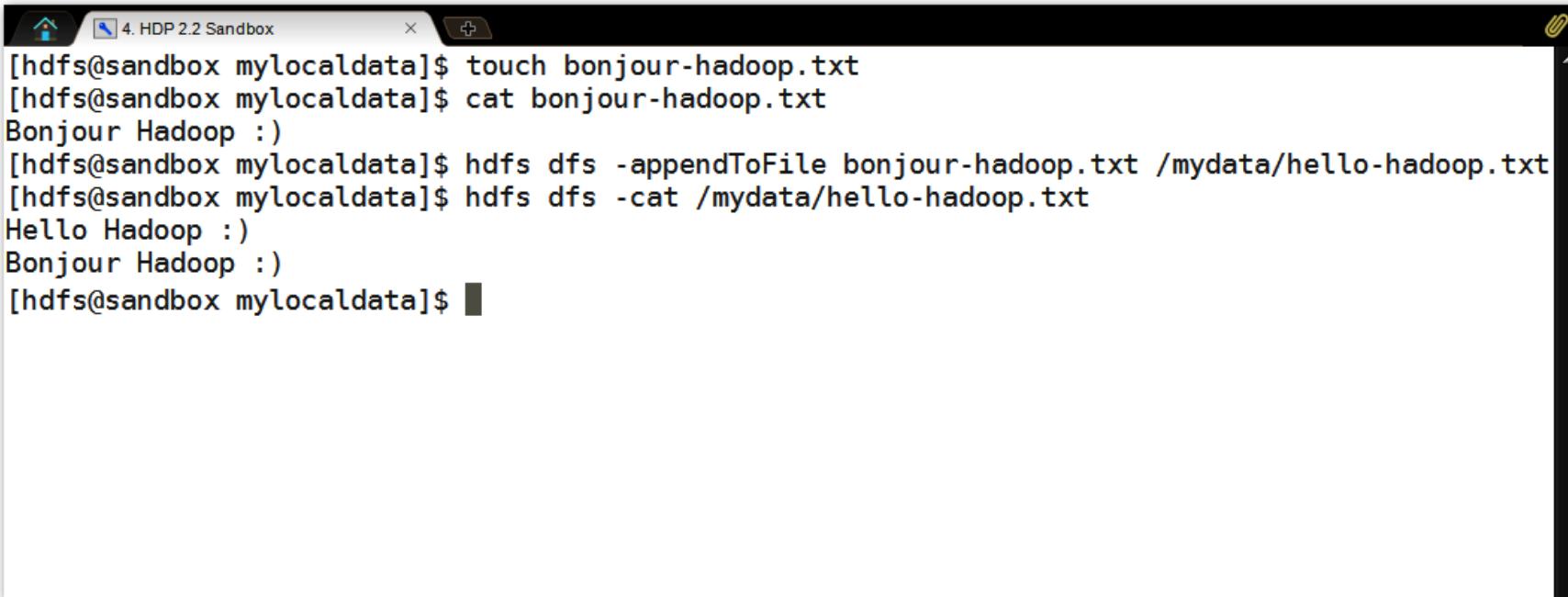


```
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata/hello-hadoop.txt
-rw-r--r-- 3 hdfs hdfs 17 2018-10-21 13:58 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -chmod 777 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata/hello-hadoop.txt
-rwxrwxrwx 3 hdfs hdfs 17 2018-10-21 13:58 /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

► Ajouter le contenu du fichier **bonjour-hadoop.txt** sur le système de fichier local au fichier **hello-hadoop.txt** sur le HDFS avec la commande **hdfs dfs -appendToFile bonjour-hadoop.txt /mydata/hello-hadoop.txt**

Lab - Les commandes hdfs dfs

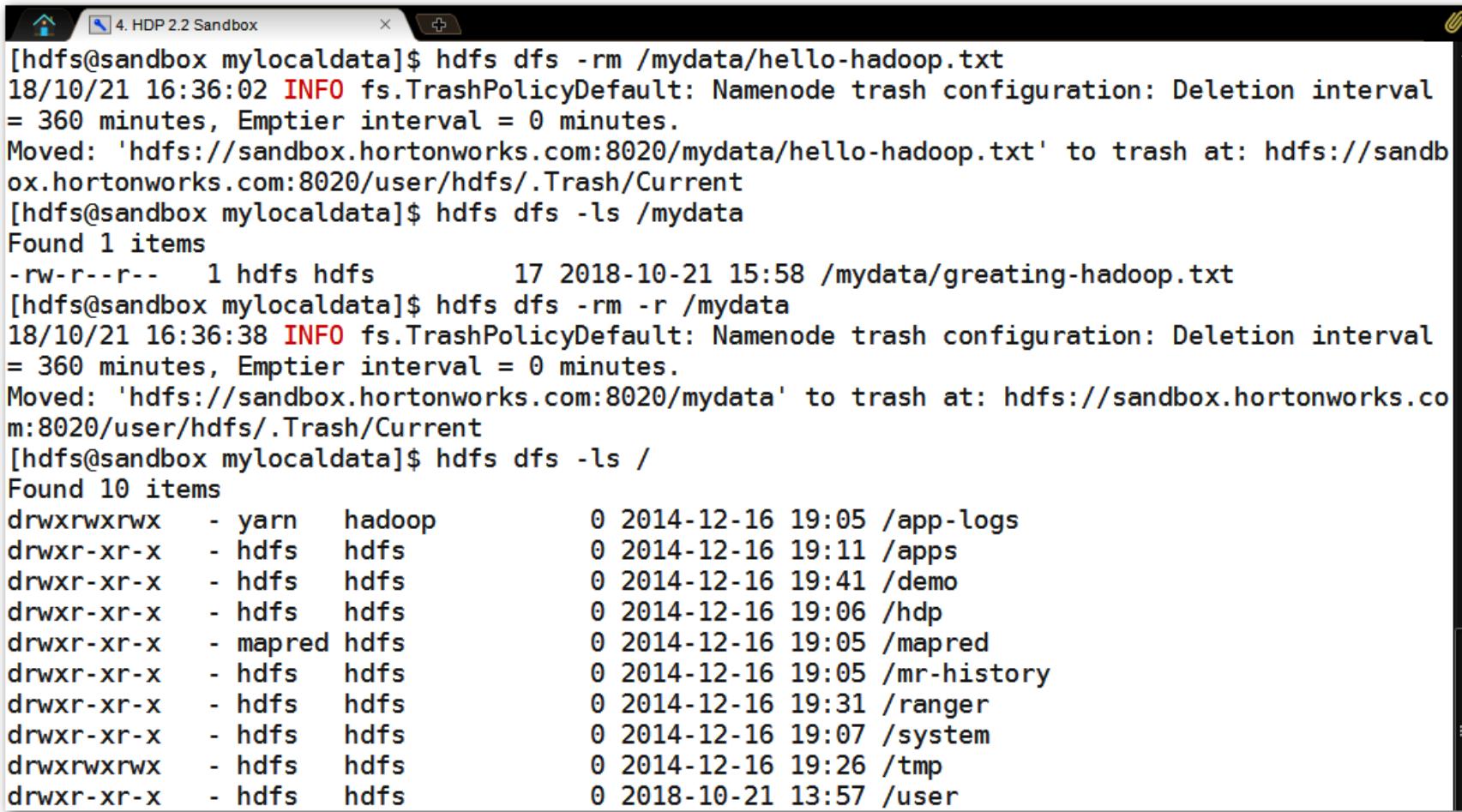


```
[hdfs@sandbox mylocaldata]$ touch bonjour-hadoop.txt
[hdfs@sandbox mylocaldata]$ cat bonjour-hadoop.txt
Bonjour Hadoop :)
[hdfs@sandbox mylocaldata]$ hdfs dfs -appendToFile bonjour-hadoop.txt /mydata/hello-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -cat /mydata/hello-hadoop.txt
Hello Hadoop :)
Bonjour Hadoop :)
[hdfs@sandbox mylocaldata]$
```

Lab - Les commandes hdfs dfs

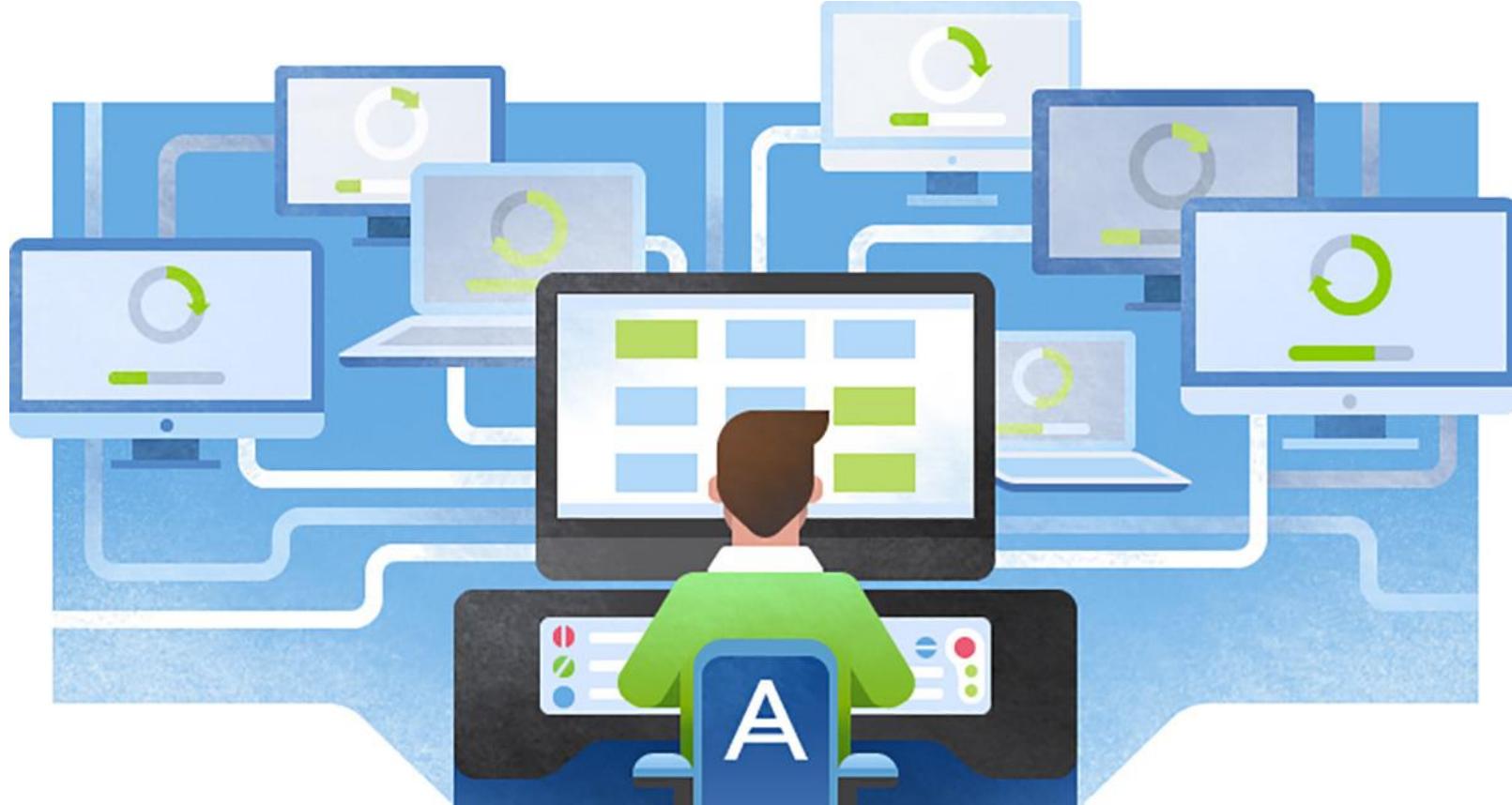
- ▶ Supprimer le fichier `hello-hadoop.txt` avec la commande `hdfs dfs -rm /mydata/hello-hadoop.txt`.
- ▶ Supprimer le répertoire `mydata` avec la commande `hdfs dfs -rm -r /mydata` (vérifier avec la commande `hdfs dfs -ls`).

Lab - Les commandes hdfs dfs



```
[hdfs@sandbox mylocaldata]$ hdfs dfs -rm /mydata/hello-hadoop.txt
18/10/21 16:36:02 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval
= 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://sandbox.hortonworks.com:8020/mydata/hello-hadoop.txt' to trash at: hdfs://sandb
ox.hortonworks.com:8020/user/hdfs/.Trash/Current
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /mydata
Found 1 items
-rw-r--r-- 1 hdfs hdfs 17 2018-10-21 15:58 /mydata/greeting-hadoop.txt
[hdfs@sandbox mylocaldata]$ hdfs dfs -rm -r /mydata
18/10/21 16:36:38 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval
= 360 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://sandbox.hortonworks.com:8020/mydata' to trash at: hdfs://sandbox.hortonworks.co
m:8020/user/hdfs/.Trash/Current
[hdfs@sandbox mylocaldata]$ hdfs dfs -ls /
Found 10 items
drwxrwxrwx  - yarn   hadoop      0 2014-12-16 19:05 /app-logs
drwxr-xr-x  - hdfs   hdfs       0 2014-12-16 19:11 /apps
drwxr-xr-x  - hdfs   hdfs       0 2014-12-16 19:41 /demo
drwxr-xr-x  - hdfs   hdfs       0 2014-12-16 19:06 /hdp
drwxr-xr-x  - mapred hdfs      0 2014-12-16 19:05 /mapred
drwxr-xr-x  - hdfs   hdfs      0 2014-12-16 19:05 /mr-history
drwxr-xr-x  - hdfs   hdfs      0 2014-12-16 19:31 /ranger
drwxr-xr-x  - hdfs   hdfs      0 2014-12-16 19:07 /system
drwxrwxrwx  - hdfs   hdfs      0 2014-12-16 19:26 /tmp
drwxr-xr-x  - hdfs   hdfs      0 2018-10-21 13:57 /user
```

Surveillance et diagnostic du HDFS



Surveillance et diagnostic du HDFS

- ▶ **HDFS fournit une interface Web, Health Web UI, pour surveiller son état de santé :**
 - ◆ Vue d'ensemble du HDFS.
 - ◆ Avancement du démarrage.
 - ◆ Les logs du NameNode.
 - ◆ Le stockage du NameNode.
 - ◆ Des informations sur les DataNodes.

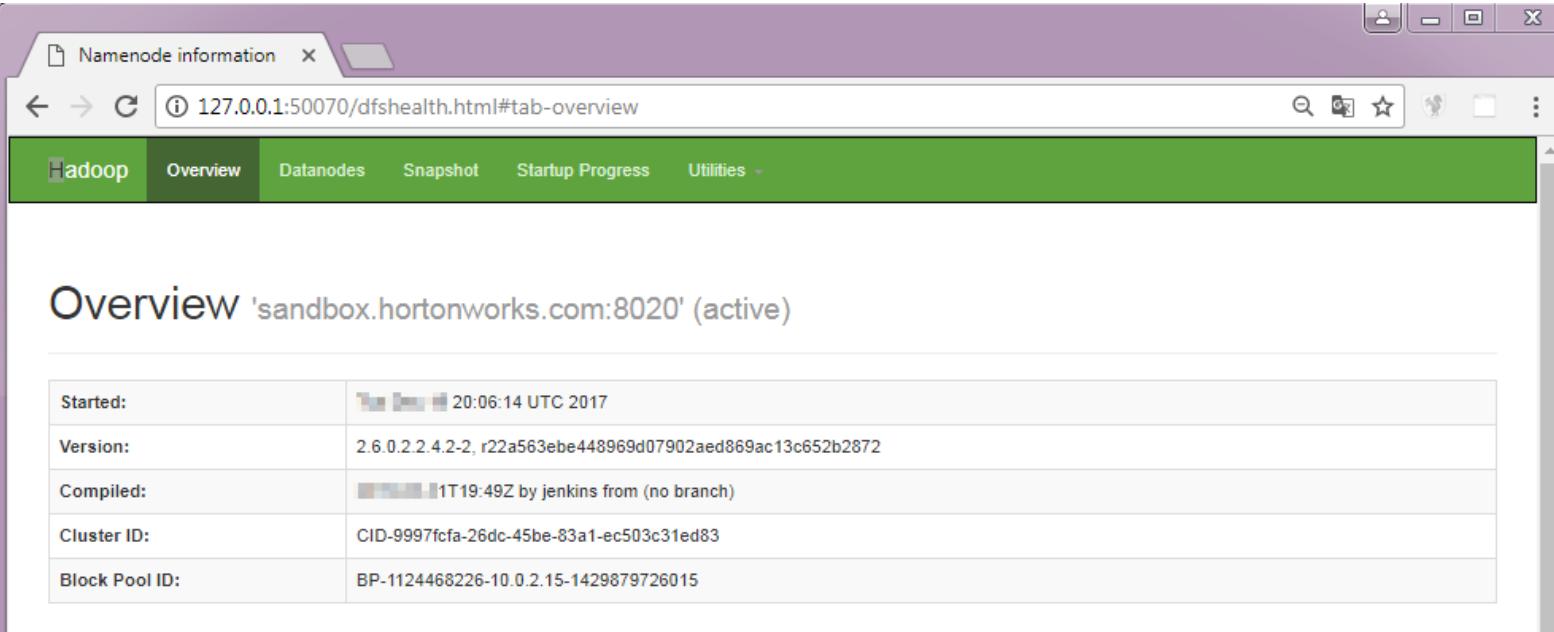
Surveillance et diagnostic du HDFS

► Pour accéder à la Health Web UI du HDFS :

- ◆ <http://host:port/dfshealth.jsp>
- ◆ Par défaut : <http://localhost:50070/dfshealth.jsp>

Surveillance et diagnostic du HDFS

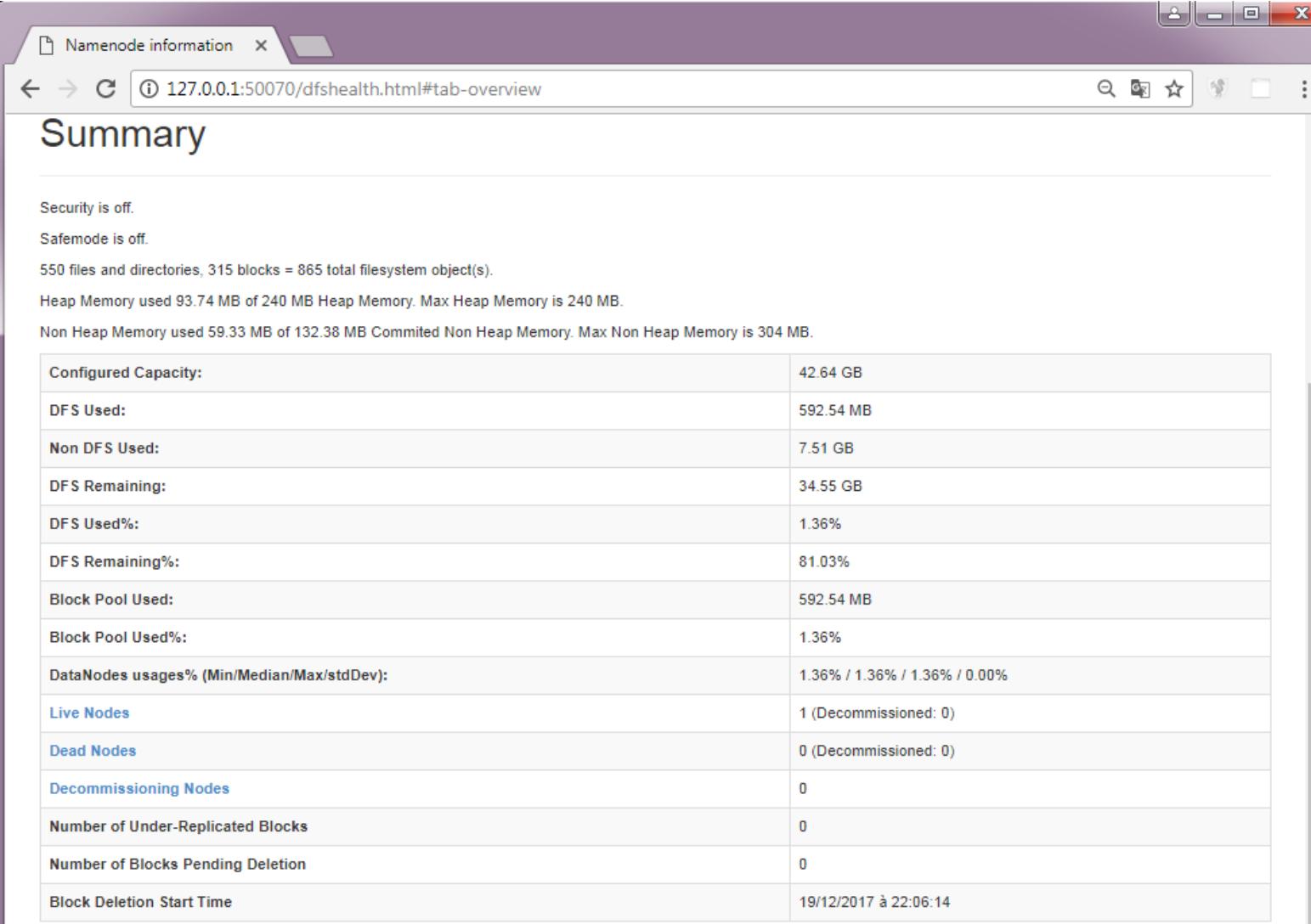
► Le récapitulatif du système de fichiers HDFS figure dans la page de présentation:



The screenshot shows a web browser window titled "Namenode information" with the URL "127.0.0.1:50070/dfshealth.html#tab-overview". The tab bar is green and contains the following tabs: Hadoop, Overview, Datanodes, Snapshot, Startup Progress, and Utilities. The main content area is titled "Overview 'sandbox.hortonworks.com:8020' (active)". Below the title is a table with the following data:

Started:	2017-06-20T06:14 UTC
Version:	2.6.0.2.2.4.2-2, r22a563ebe448969d07902aed869ac13c652b2872
Compiled:	1T19:49Z by jenkins from (no branch)
Cluster ID:	CID-9997fcfa-26dc-45be-83a1-ec503c31ed83
Block Pool ID:	BP-1124468226-10.0.2.15-1429879726015

Surveillance et diagnostic du HDFS



The screenshot shows a web browser window titled "Namenode information" with the URL "127.0.0.1:50070/dfshealth.html#tab-overview". The page displays various HDFS metrics under the "Summary" section. It also includes a table of configuration parameters.

Summary

Security is off.
Safemode is off.
550 files and directories, 315 blocks = 865 total filesystem object(s).
Heap Memory used 93.74 MB of 240 MB Heap Memory. Max Heap Memory is 240 MB.
Non Heap Memory used 59.33 MB of 132.38 MB Committed Non Heap Memory. Max Non Heap Memory is 304 MB.

Configured Capacity:	42.64 GB
DFS Used:	592.54 MB
Non DFS Used:	7.51 GB
DFS Remaining:	34.55 GB
DFS Used%:	1.36%
DFS Remaining%:	81.03%
Block Pool Used:	592.54 MB
Block Pool Used%:	1.36%
DataNodes usages% (Min/Median/Max/stdDev):	1.36% / 1.36% / 1.36% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	19/12/2017 à 22:06:14

Surveillance et diagnostic du HDFS

► Les informations sur le statut et le stockage de NameNode sont également affichées en bas.

NameNode Journal Status

Current transaction ID: 2796

Journal Manager	State
FileJournalManager(root=/hadoop/hdfs/namenode)	EditLogFileOutputStream(/hadoop/hdfs/namenode/current/edits_inprogress_000000000000002772)

NameNode Storage

Storage Directory	Type	State
/hadoop/hdfs/namenode	IMAGE_AND_EDITS	Active

Surveillance et diagnostic du HDFS

► Les informations sur les DataNodes sont affichées dans l'onglet DataNodes, comme indiqué ci-dessous:

The screenshot shows the HDFS DataNodes page with the following sections and data:

- Header:** Hadoop, Overview, Datanodes (highlighted), Snapshot, Startup Progress, Utilities.
- Datanode Information:** A table with columns: Node, Last contact, Admin State, Capacity, Used, Non DFS Used, Remaining, Blocks, Block pool used, Failed Volumes, Version. One row is shown: sandbox.hortonworks.com (10.0.2.15:50010) | 1 | In Service | 42.64 GB | 590.97 MB | 7.42 GB | 34.64 GB | 310 | 590.97 MB (1.35%) | 0 | 2.6.0.2.2.4.2-2
- Decommissioning:** A table with columns: Node, Last contact, Under replicated blocks, Blocks with no live replicas, Under Replicated Blocks In files under construction. One row is shown: (empty)

Surveillance et diagnostic du HDFS

Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Startup Progress

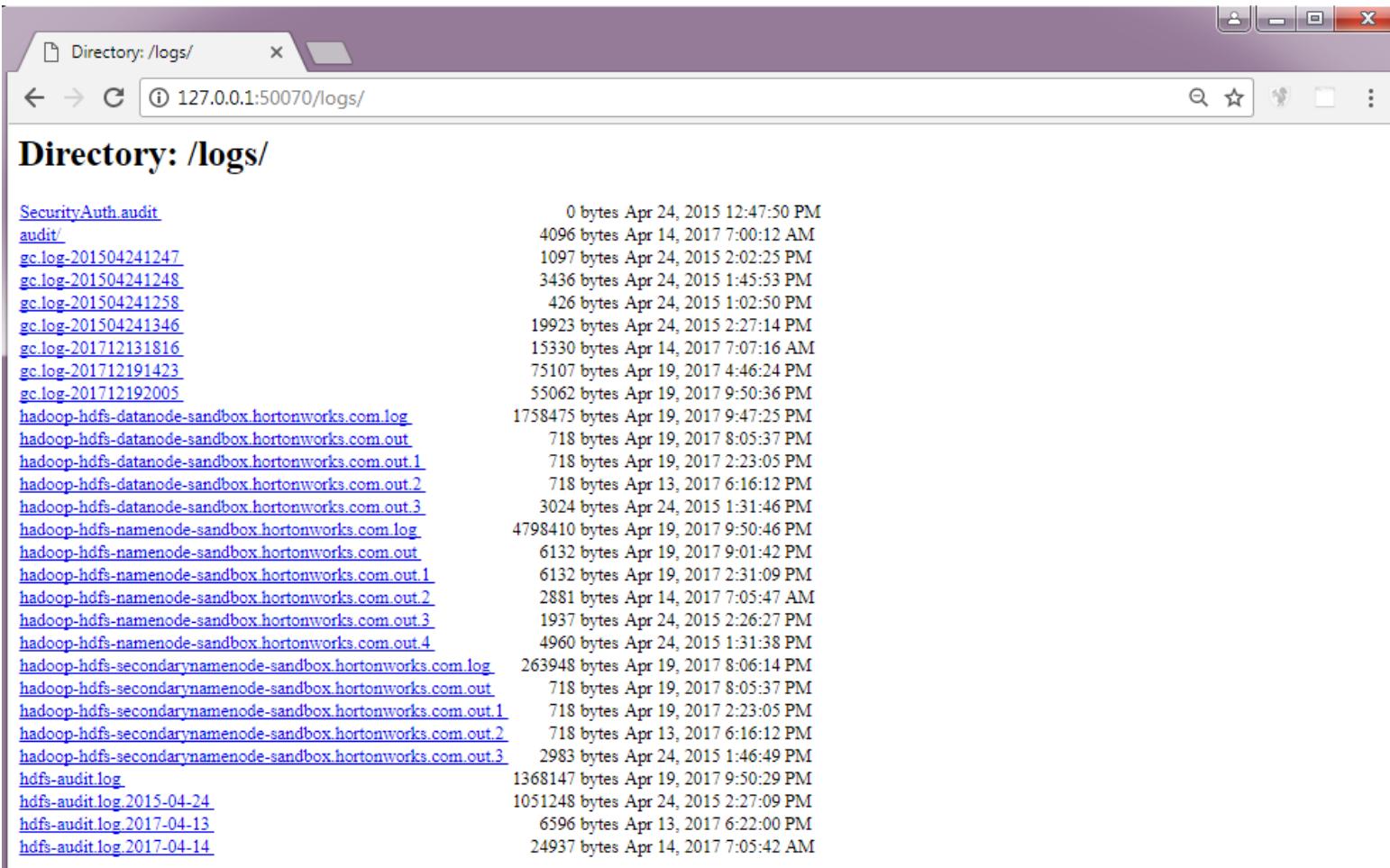
Elapsed Time: 3 mins, 10 sec, Percent Complete: 100%

Phase	Completion	Elapsed Time
Loading fsimage /hadoop/hdfs/namenode/current/fsimage_00000000000000000000 336 B	100%	1 sec
inodes (0/0)	100%	
delegation tokens (0/0)	100%	
cache pools (0/0)	100%	
Loading edits	100%	1 sec
/hadoop/hdfs/namenode/current/edits_0000000000000001-0000000000000222 1 MB (2222/2222)	100%	
/hadoop/hdfs/namenode/current/edits_0000000000000223-00000000000002771 1 MB (549/549)	100%	
Saving checkpoint	100%	0 sec
inodes /hadoop/hdfs/namenode/current/fsimage.ckpt_00000000000002771 (0/0)	100%	
delegation tokens /hadoop/hdfs/namenode/current/fsimage.ckpt_00000000000002771 (0/0)	100%	
cache pools /hadoop/hdfs/namenode/current/fsimage.ckpt_00000000000002771 (0/0)	100%	
Safe mode	100%	3 mins, 3 sec
awaiting reported blocks (0/310)	100%	

Les logs de l'application HDFS

- ▶ Hadoop utilise log4j via la structure de journalisation Apache Commons Logging.
- ▶ Les logs du HDFS sont, par défaut, dans le dossier **/var/log/hadoop/hdfs**.
- ▶ Il est possible de parcourir les fichiers logs générés à partir de divers événements HDFS à l'aide d'un navigateur Web:
 - ◆ <http://hôte:port/logs>
 - ◆ Valeur par défaut: <http://localhost:50070/logs>

Les logs de l'application HDFS



Programmation Hadoop

Comme indiqué précédemment, Hadoop est développé en **Java**. Les tâches MAP/REDUCE sont donc implantables par le biais d'interfaces Java (il existe cependant des **wrappers** très simples permettant d'implémenter ses tâches dans n'importe quel langage). Un programme Hadoop se compile au sein d'un **.jar**.

Pour développer un programme Hadoop, on va créer **trois classes distinctes** :

- **Une classe dite « Driver »** qui contient la **fonction main** du programme. Cette classe se chargera d'informer Hadoop des **types de données clef/valeur** utilisées, des classes se chargeant des opérations **MAP** et **REDUCE**, et des **fichiers HDFS** à utiliser pour les **entrées/sorties**.
- **Une classe MAP** (qui effectuera l'opération MAP).
- **Une classe REDUCE** (qui effectuera l'opération REDUCE).

Programmation Hadoop – Classe Driver (1/12)

La classe Driver contient la fonction « main » de notre programme.

Au sein du main() en question, on va effectuer les opérations suivantes :

- **Créer un objet Configuration de Hadoop**, qui est nécessaire pour permettre à Hadoop d'obtenir la configuration générale du cluster. L'objet en question pourrait aussi nous permettre de récupérer nous-même des options de configuration qui nous intéressent.
- Permettre à Hadoop de récupérer d'éventuels arguments génériques disponibles sur la ligne de commande (par exemple le nom du package de la tâche à exécuter si le .jar en contient plusieurs). On va également récupérer les arguments supplémentaires pour s'en servir; on souhaite que l'utilisateur puisse préciser le nom du fichier d'entrée et le nom du répertoire de sortie HDFS pour nos tâches Hadoop grâce à la ligne de commande.

Programmation Hadoop – Classe Driver (2/12)

- **Créer un nouvel objet Hadoop Job, qui désigne une tâche Hadoop.**
- **Utiliser cet objet Job pour informer Hadoop du nom de nos classes Driver, MAP et REDUCE.**
- **Utiliser le même objet pour informer Hadoop des types de données utilisés dans notre programme pour les couples (clef;valeur) MAP et REDUCE.**
- **Informer Hadoop des fichiers d'entrée/sortie pour notre tâche sur HDFS.**
- **Enfin, utiliser l'objet Job créé précédemment pour déclencher le lancement de la tâche via le cluster Hadoop.**

On reprend ici l'exemple du compteur d'occurrences de mots décrit précédemment.

Programmation Hadoop – Classe Driver (3/12)

- Le prototype de notre fonction main() :

```
public static void main(String[] args) throws Exception
```

- On se sert de `args` pour récupérer les arguments de la ligne de commande. Plusieurs fonctions Hadoop appelées au sein du main sont susceptibles de déclencher des exceptions – on l'indique donc lors de la déclaration.
- Avant toute chose, on créé dans notre main un nouvel objet Configuration Hadoop :

```
// Crée un objet de configuration Hadoop.  
Configuration conf=new Configuration();
```
- Le package à importer est: `org.apache.hadoop.conf.Configuration`

Programmation Hadoop – Classe Driver (4/12)

- Ensuite, on passe à Hadoop les arguments de la ligne de commande pour lui permettre de récupérer ceux qui sont susceptibles de lui être adressés :

```
String[] ourArgs =  
new GenericOptionsParser(conf, args).getRemainingArgs();
```

- On utilise pour ce faire un objet Hadoop `GenericOptionsParser`, dont le package est : `org.apache.hadoop.util.GenericOptionsParser`
- La fonction `getRemainingArgs()` de notre objet nous permet par ailleurs de récupérer les arguments non exploités par Hadoop au sein d'un tableau `ourArgs` – ce qui nous permettra de les utiliser par la suite.
- On peut ainsi rendre paramétrable facilement une tâche Hadoop lors de l'exécution, par le biais des arguments de la ligne de commande.

Programmation Hadoop – Classe Driver (5/12)

- On crée ensuite un nouvel objet Hadoop Job :

```
Job job=Job.getInstance(conf, "Compteur de mots v1.0");
```

- Le package à importer est le suivant:
`org.apache.hadoop.mapreduce.Job`
- On passe au constructeur notre objet Configuration, ainsi qu'une description textuelle courte du programme Hadoop.
- Ensuite, il faut indiquer à Hadoop – par le biais de l'objet Job nouvellement créé – quelles sont les classes Driver, Map et Reduce de notre programme Hadoop.
- Dans notre cas, il s'agira respectivement des classes `WCount`, `WCountMap` et `WCountReduce` du package `enis.hadoop.wordcount`.

Programmation Hadoop – Classe Driver (6/12)

- On utilise pour ce faire les fonctions suivantes:

```
job.setJarByClass(WCount.class);  
job.setMapperClass(WCountMap.class);  
job.setReducerClass(WCountReduce.class);
```

- Il faut ensuite indiquer à Hadoop quels sont les types de données que l'on souhaite utiliser pour les couples (clef;valeur) de nos opérations map et reduce. Dans le cas de notre compteur d'occurrences de mots, on souhaite utiliser des chaînes de caractères pour les clefs (nos mots) et des entiers pour nos occurrences.
- **Remarque:** on ne doit pas utiliser les types classiques Int et String de Java pour désigner nos types, mais des classes qui leur correspondent et qui sont propres à Hadoop. Dans notre cas, les classes IntWritable et Text.

Programmation Hadoop – Classe Driver (7/12)

```
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
```

- Les packages des types en question:
`org.apache.hadoop.io.IntWritable`
et `org.apache.hadoop.io.Text`
- Il en existe beaucoup d'autres dans `org.apache.hadoop.io.*`.
- Ensuite, on doit indiquer où se situent nos données d'entrée et de sortie dans HDFS. On utilise pour ce faire les classes Hadoop FileInputFormat et FileOutputFormat.
- Ces classes sont implémentées suivant un design pattern Singleton – il n'est pas nécessaire de les instancier dans le cas qui nous intéresse (dans des cas plus complexe, on étendra parfois la classe en question dans une nouvelle classe qui nous est propre).

Programmation Hadoop – Classe Driver (8/12)

- On procède de la manière suivante :

```
FileInputFormat.addInputPath(job, new Path(ourArgs[0]));
FileOutputFormat.setOutputPath(job,new Path(ourArgs[1]));
```

- Les packages à utiliser :
 - `org.apache.hadoop.mapreduce.lib.input.FileInputFormat`
 - `org.apache.hadoop.mapreduce.lib.input.FileOutputFormat`
 - `org.apache.hadoop.fs.Path`
- On utilise les arguments restants après ceux qu'a utilisé Hadoop. Le code est ici simplifié – on devrait en théorie vérifier la taille du tableau `ourArgs` pour éviter d'éventuelles erreurs.

Programmation Hadoop – Classe Driver (9/12)

- Enfin, il reste à lancer l'exécution de la tâche par le biais du cluster Hadoop.
On procède ainsi:

```
if(job.waitForCompletion(true))  
    System.exit(0);  
System.exit(-1);
```

- La fonction waitForCompletion de l'objet job va exécuter la tâche et attendre la fin de son exécution. Elle prend un argument: un booléen indiquant à Hadoop si oui ou non il doit donner des indications sur la progression de l'exécution à l'utilisateur sur la sortie standard (stdout) .
- Elle renvoie true en cas de succès; ici, on terminera l'exécution du programme en renvoyant 0 si tout s'est bien passé, et -1 en cas de problème (codes de retour unix standards).

Programmation Hadoop – Classe Driver (10/12)

- Le code complet de notre classe Driver:

```
package hadoop.wordcount;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;

// Notre classe Driver (contient le main du programme Hadoop).
public class WCount
{
```

Programmation Hadoop – Classe Driver (11/12)

```
// Le main du programme.  
public static void main(String[] args) throws Exception  
{  
    // Crée un object de configuration Hadoop.  
    Configuration conf=new Configuration();  
  
    // Permet à Hadoop de lire ses arguments génériques,  
    // récupère les arguments restants dans ourArgs.  
    String[] ourArgs=new GenericOptionsParser(conf,  
                                              args).getRemainingArgs();  
  
    // Obtient un nouvel objet Job: une tâche Hadoop. On  
    // fourni la configuration Hadoop ainsi qu'une description  
    // textuelle de la tâche.  
    Job job=Job.getInstance(conf, "Compteur de mots v1.0");
```

Programmation Hadoop – Classe Driver (12/12)

```
// Défini les classes driver, map et reduce.  
job.setJarByClass(WCount.class);  
job.setMapperClass(WCountMap.class);  
job.setReducerClass(WCountReduce.class);  
  
// Défini types clefs/valeurs de notre programme Hadoop.  
job.setOutputKeyClass(Text.class);  
job.setOutputValueClass(IntWritable.class);  
  
// Défini les fichiers d'entrée du programme et le  
// répertoire des résultats. On se sert du premier et du  
// deuxième argument restants pour permettre à  
// l'utilisateur de les spécifier lors de l'exécution.  
FileInputFormat.addInputPath(job, new Path(ourArgs[0]));  
FileOutputFormat.setOutputPath(job, new Path(ourArgs[1]));  
  
// On lance la tâche Hadoop. Si elle s'est effectuée  
// correctement, on renvoie 0. Sinon, on renvoie -1.  
if(job.waitForCompletion(true))  
    System.exit(0);  
System.exit(-1);  
}
```

Programmation Hadoop – Classe MAP (1/6)

- La classe MAP va être en charge de l'opération MAP de notre programme.
- Elle doit étendre la classe Hadoop org.apache.hadoop.mapreduce.Mapper. Il s'agit d'une classe générique qui se paramétre avec quatre types :
 - Un type **keyin**: le type de clef d'entrée.
 - Un type **valuein**: le type de valeur d'entrée.
 - Un type **keyout**: le type de clef de sortie.
 - Un type **valueout**: le type de valeur de sortie.
- Le type **keyin** est notamment utile lorsqu'on utilise des fonctionnalités plus avancées, comme la possibilité d'effectuer plusieurs opérations MAP les unes à la suite des autres, auquel cas notre opération map recevra en entrée des couples (clef;valeur).
- Dans notre cas, nous n'utiliserons pas cette possibilité; on utilisera donc le type Java **Object** comme type **keyin**.

Programmation Hadoop – Classe MAP (2/6)

Dans notre exemple, notre classe Map sera déclarée ainsi :

```
public class WCountMap extends Mapper<Object, Text, Text,  
IntWritable>
```

On utilise ici comme types:

- **Text** pour le type **valuein**, puisque notre valeur d'entrée à la fonction Map est une chaîne de caractères (une ligne de texte).
- **Text** pour le type **keyout**, puisque notre valeur de clef pour les couples (clef;valeur) de la fonction Map est également une chaîne de caractères (le mot dont on compte les occurrences).
- **IntWritable** pour le type **valueout**, puisque notre valeur pour les couples (clef;valeur) de la fonction Map est un entier (le nombre d'occurrences).

Ici aussi, on utilise les types Hadoop et non les types natifs Java (Int et String).

Programmation Hadoop – Classe MAP (3/6)

- Au sein de la classe **Mapper**, c'est la fonction **map** qui va s'occuper d'effectuer la tâche MAP. C'est la seule qu'on doit absolument implémenter.
- Elle prend trois arguments: la clef d'entrée **keyin** (qu'on ignore dans notre cas), la valeur d'entrée **valuein** (la ligne de texte dont on souhaite compter les mots), et un **Context Java** qui représente un handle Hadoop et nous permettra de retourner les couples (clef;valeur) résultant de notre opération Map.
- Le prototype de notre fonction **map** :

```
protected void map(Object key,Text value,Context context)
throws IOException, InterruptedException
```
- Comme pour la fonction **main**, la fonction **map** appellera des fonctions susceptibles de déclencher des exceptions (notamment concernant l'interruption de l'exécution Hadoop ou des problèmes d'accès HDFS) – on le précise donc dans sa déclaration.

Programmation Hadoop – Classe MAP (4/6)

- Au sein de la méthode `map`, on va donc effectuer la tâche MAP de notre programme MAP/REDUCE.
- Dans le cadre de notre exemple, la fonction devra parcourir la ligne de texte fournie en entrée, et renvoyer un couple (clef;valeur) pour chacun des mots. Ce couple devra avoir pour clef le mot en question, et pour valeur l'entier « 1 ».
- Dans la fonction `map`, afin d'indiquer à Hadoop qu'on souhaite renvoyer un couple (clef;valeur), on utilise la fonction `write` de notre objet `Context`. Elle peut être appelée autant de fois que nécessaire; une fois pour chacun des couples (clef;valeur) qu'on souhaite renvoyer. Par exemple :

```
context.write("ciel", 1);
```

- Il faut évidemment que la clef et la valeur renvoyées ainsi correspondent aux types `keyout` et `valueout` de notre classe `Mapper`.

Programmation Hadoop – Classe MAP (5/6)

- Notre classe Map d'exemple en intégralité :

```
package hadoop.wordcount;

import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import java.util.StringTokenizer;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

// Notre classe MAP.
public class WCountMap extends Mapper<Object, Text, Text, IntWritable>
{
    // IntWritable contant de valeur 1.
    private static final IntWritable ONE=new IntWritable(1);
```

Programmation Hadoop – Classe MAP (6/6)

```
// La fonction MAP elle-même.  
protected void map(Object offset, Text value, Context context)  
    throws IOException, InterruptedException  
{  
    // Un StringTokenizer va nous permettre de parcourir chacun des  
    // mots de la ligne qui est passée à notre opération MAP.  
    StringTokenizer tok=new StringTokenizer(value.toString(), " ");  
  
    while(tok.hasMoreTokens())  
    {  
        Text word=new Text(tok.nextToken());  
        // On renvoie notre couple (clef;valeur): le mot courant suivi  
        // de la valeur 1 (définie dans la constante ONE).  
        context.write(word, ONE);  
    }  
}
```

Programmation Hadoop – Classe REDUCE (1/6)

- La classe **REDUCE** va être en charge de l'opération **REDUCE** de notre programme.
- Elle doit étendre la classe Hadoop `org.apache.hadoop.mapreduce.Reducer`. Il s'agit là aussi d'une classe générique qui se paramétre avec les mêmes quatre types que pour la classe Mapper: `keyin`, `valuein`, `keyout` et `valueout`.
- On rappelle que l'opération **REDUCE** recevra en entrée une clef unique, associée à toutes les valeurs pour la clef en question.
- Dans le cas du compteur d'occurrences de mots, on recevra en entrée une valeur unique pour la clef, par exemple « ciel », suivi de toutes les valeurs qui ont été rencontrées à la sortie de l'opération MAP pour la clef « ciel » (par exemple cinq fois la valeur « 1 » si le mot « ciel » était présent cinq fois dans notre texte d'exemple).

Programmation Hadoop – Classe REDUCE (2/6)

Dans notre exemple, notre classe `Reduce` sera définie comme suit:

```
public class WCountReduce extends Reducer<Text,  
IntWritable, Text, Text>
```

On utilise pour chacun des types paramétrables:

- `Text` pour `keyin`: il s'agit de notre clef unique d'entrée – le mot concerné.
- `IntWritable` pour `valuein`: le type de nos valeurs associées à cette clef (le nombre d'occurrences, un entier).
- `Text` pour `keyout`: le type de clef de sortie. Nous ne modifierons pas la clef, il s'agira toujours du mot unique concerné – on utilise donc `Text`.
- `Text` pour `valueout`: le type de valeur de sortie. On utilise ici `Text` – on renverra le nombre total d'occurrences pour le mot concerné sous la forme d'une chaîne de caractères (on pourrait également utiliser `IntWritable` ici).

Là aussi, on utilise les types de données propres à Hadoop.

Programmation Hadoop – Classe REDUCE (3/6)

- Au sein de la classe **Reducer**, c'est la fonction **reduce** qui va effectuer l'opération REDUCE. C'est la seule qu'on doive implémenter.
- Elle prend trois arguments: la clef concernée, un **Iterable** java (une liste) de toutes les valeurs qui lui sont associées et qui ont été renvoyées par l'opération MAP, et enfin un objet **Context** java similaire à celui de la fonction **map** de la classe **Mapper**, et qui nous permettra de renvoyer notre valeur finale, associée à la clef.
- Dans notre exemple, la déclaration de la fonction **reduce**:

```
public void reduce(Text key, Iterable<IntWritable>
                  values, Context context)
                  throws IOException, InterruptedException
```

- Comme pour map, la fonction fait appel à des fonctions Hadoop susceptibles de provoquer des exceptions – on l'indique ici aussi.

Programmation Hadoop – Classe REDUCE (4/6)

- Au sein de la fonction `reduce`, on pourra renvoyer un couple (clef;valeur) en résultat exactement de la même manière que pour la fonction `map`, par le biais d'un appel à la fonction `write` de notre objet `Context`.
- Par exemple :

```
context.write("ciel", "5 occurrences");
```
- Contrairement à la fonction `map`, en revanche, on cherche à ne produire qu'une et une seule valeur de retour pour la clef concernée. On n'appellera donc la fonction `write` qu'une seule fois.
- Remarque: en théorie et dans des cas plus complexes, on pourrait là aussi appeler la fonction à plusieurs reprises, pour renvoyer plusieurs couples (clef;valeur). En revanche, Hadoop n'appliquera aucun traitement dessus et les considérera simplement comme plusieurs résultats finaux.

Programmation Hadoop – Classe REDUCE (5/6)

- Notre classe Reduce d'exemple en intégralité :

```
package hadoop.wordcount;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Reducer;
import java.util.Iterator;
import java.io.IOException;

// Notre classe REDUCE - paramétrée avec un type Text pour la clef, un
// type de valeur IntWritable, et un type de retour (le retour final de
// la fonction Reduce) Text.
public class WCountReduce extends Reducer<Text, IntWritable, Text, Text>
{
```

Programmation Hadoop – Classe REDUCE (6/6)

```
// La fonction REDUCE elle-même. Les arguments: la clef key, un
// Iterable de toutes les valeurs qui sont associées à la clef en
// question, et le contexte Hadoop (un handle qui nous permet de
// renvoyer le résultat à Hadoop).
public void reduce(Text key, Iterable<IntWritable> values,
                    Context context)
                    throws IOException, InterruptedException
{
    // Pour parcourir toutes les valeurs associées à la clef fournie.
    Iterator<IntWritable> i=values.iterator();
    int count=0; // Notre total pour le mot concerné.
    while(i.hasNext()) // Pour chaque valeur...
        count+=i.next().get(); // ...on l'ajoute au total.
    // On renvoie le couple (clef;valeur) constitué de notre clef key
    // et du total, au format Text.
    context.write(key, new Text(count+" occurrences."));
}
```

Remarques

- **Hadoop est une plate-forme récente, en développement constant: l'API Java change régulièrement.** Il faut toujours s'informer sur les dernières modifications afin de rester à jour des dernières modifications. Par ailleurs, la distribution Hadoop comprend de nombreux exemples de programmes Map/Reduce, mis à jour afin d'utiliser l'API la plus récente.
- Même si l'API change régulièrement, les anciennes manières de procéder restent généralement disponibles pendant plusieurs mois après une mise à jour.
- Enfin, il existe de nombreuses autres possibilités offertes par l'API: extraire des paramètres de configurations globaux du cluster, implémenter sa propre classe `FileInputFormat` pour des traitements plus complexes sur les données d'entrée, etc... d'une manière générale, se référer à la documentation de l'API Hadoop: <https://hadoop.apache.org/docs/>

Compilation

- Pour compiler un programme Hadoop, il suffit d'utiliser le compilateur `javac` comme pour tout autre programme Java.
- Il est cependant nécessaire d'inclure les librairies appropriées. Elles sont disponibles avec les distributions de Hadoop, sous la forme de fichiers `.jar`. Les librairies principales à utiliser :
 - `hadoop-common.jar`
 - `hadoop-mapreduce-client-core.jar`
 - `hadoop-mapreduce-client-common.jar`
 - `commons-cli.jar`

... il en existe d'autres pour des cas d'utilisation plus complexes, à inclure en fonction des besoins.
- Enfin, après compilation, on package le programme Hadoop à l'intérieur d'un fichier `.jar` pour son exécution.

Exécution

- Pour exécuter un programme Hadoop, on utilise là aussi le programme en ligne de commande `hadoop`. La syntaxe est la suivante :

```
hadoop jar [JARFILE] [DRIVERCLASS] [PARAMETERS]
```

- Par exemple pour notre compteur d'occurrences de mots :
(la commande est sur la même ligne) :

```
hadoop jar enis_wcount.jar enis.hadoop.wordcount.WCount  
input/poeme.txt /results
```

- La commande demandera à Hadoop d'exécuter le programme Hadoop de la classe Driver `enis.hadoop.wordcount.WCount` du fichier `enis_wcount.jar`, en lui indiquant qu'il doit lire son texte dans le fichier « `input/poeme.txt` » sur HDFS, et stocker les résultats dans le répertoire « `results` » sur HDFS.

Remarques

- Hadoop stocke les résultats dans une série de fichiers **part-r-xxxx**, où **xxxx** est un compteur incrémental. L'idée est ici de stocker de grandes quantités de résultats dans de nombreux fichiers différents, qui seront en conséquences distribués sur tout le cluster HDFS. Le nom du fichier est paramétrable dans la configuration Hadoop.
- On a un fichier « **part-r** » par opération REDUCE exécutée. Le « **r** » au sein du nom signifie « **Reduce** ». On peut également demander à Hadoop d'effectuer uniquement les opérations MAP (par exemple pour du debug), auquel cas on aura une série de fichiers « **part-m** ».
- Un fichier **_SUCCESS** (vide) est également créé dans le répertoire des résultats en cas de succès. Cela permet de contrôler que tout s'est bien passé rapidement (avec un simple **hadoop fs -ls** sur HDFS).

Autres langages: Streaming (1/7)

- Au delà de Java, Hadoop permet également l'exécution d'un programme Hadoop écrit dans d'autres langages, par exemple en C, en Python ou encore en bash (shell scripting).
- Pour ce faire, un outil est distribué avec Hadoop: streaming. Il s'agit d'un .jar qui est capable de prendre en argument des programmes ou scripts définissant les tâches MAP et REDUCE, ainsi que les fichiers d'entrée et le répertoire de sortie HDFS, et d'exécuter ainsi la tâche spécifiée sur le cluster.
- Ce .jar est disponible dans le répertoire d'installation Hadoop et porte le nom `hadoop-streaming-VERSION.jar`, où `VERSION` est la version de Hadoop concernée.
- Il s'agit en réalité d'un programme Hadoop Java « classique », mais qui appelle les tâches MAP et REDUCE par le biais du système.

Autres langages: Streaming - MAP (2/7)

Lorsqu'on développe un programme MAP Hadoop dans un autre langage pour son utilisation avec l'outil streaming, les données d'entrée doivent être lues sur l'entrée standard (`stdin`) et les données de sorties doivent être envoyées sur la sortie standard (`stdout`).

- En entrée du script ou programme MAP, on aura une série de lignes: nos données d'entrée (par exemple dans le cas du compteur d'occurrences de mots, des lignes de notre texte).
- En sortie du script ou programme MAP, on doit écrire sur `stdout` notre série de couples (clef;valeur) au format:

CLEF [TABULATION] VALEUR

... avec une ligne distincte pour chaque (clef;valeur).

Autres langages: Streaming - Reduce (3/7)

Lorsqu'on développe un programme REDUCE Hadoop dans un autre langage pour son utilisation avec l'outil streaming, les données d'entrée et de sortie doivent être lues/écrites sur `stdin` et `stdout` (respectivement).

- En entrée du script ou programme REDUCE, on aura une série de lignes: des couples (clef;valeur) au format:
CLEF [TABULATION] VALEUR
Les couples seront triés par clef distincte, et la clef répétée à chaque fois. Par ailleurs, on est susceptible d'avoir des clefs differentes au sein d'une seule et même exécution du programme reduce !
- En sortie du script ou programme REDUCE, on doit écrire des couples (clef;valeur), toujours au format **CLEF [TABULATION] VALEUR**.
Remarque: d'ordinaire, on écrira évidemment un seul couple (clef;valeur) par clef distincte.

Autres langages: Streaming – Ex. Python (4/7)

Occurrences de mots version Python – opération MAP:

```
import sys

# Pour chaque ligne d'entrée.
for line in sys.stdin:
    # Supprimer les espaces autour de la ligne.
    line=line.strip()
    # Pour chaque mot de la ligne.
    words=line.split()
    for word in words:
        # Renvoyer couple clef;valeur: le mot comme clef, l'entier "1" comme
        # valeur.
        # On renvoie chaque couple sur une ligne, avec une tabulation entre
        # la clef et la valeur.
        print "%s\t%d" % (word, 1)
```

Autres langages: Streaming – Ex. Python (5/7)

Occurrences de mots version Python – opération REDUCE:

```
import sys

total=0; # Notre total pour le mot courant.
# Contient le dernier mot rencontré.
lastword=None

# Pour chaque ligne d'entrée.
for line in sys.stdin:
    # Supprimer les espaces autour de la ligne.
    line=line.strip()

    # Récupérer la clef et la valeur, convertir la valeur en int.
    word, count=line.split('\t', 1)
    count=int(count)
```

Autres langages: Streaming – Ex. Python (6/7)

```
# On change de mot (test nécessaire parce qu'on est susceptible d'avoir
# en entrée plusieurs clefs distinctes différentes pour une seule et
# même exécution du programme - Hadoop triera les couples par clef
# distincte).
if word!=lastword and lastword!=None:
    print "%s\t%d occurrences" % (lastword, total)
    total=0;
lastword=word

total=total+count # Ajouter la valeur au total

# Ecrire le dernier couple (clef;valeur).
print "%s\t%d occurrences" % (lastword, total)
```

Autres langages: Streaming - Exécution (7/7)

Streaming est un programme Hadoop standard, on l'exécutera donc avec la même commande `hadoop jar` qu'un programme Hadoop Java habituel. C'est dans ses options de ligne de commande qu'on va indiquer les scripts ou programmes Hadoop map et reduce à utiliser. Syntaxe:

```
hadoop jar hadoop-streaming-X.Y.Z.jar -input [HDFS INPUT FILES] \
                                         -output [HDFS OUTPUT FILES] \
                                         -mapper [MAP PROGRAM] \
                                         -reducer [REDUCE PROGRAM]
```

Par exemple:

```
hadoop jar hadoop-streaming-X.Y.Z.jar -input /poeme.txt \
                                         -output /results -mapper ./map.py -reducer ./reduce.py
```

Après quoi la tâche Hadoop s'exécutera exactement comme une tâche standard.

Chapitre 4

L'INTÉGRATION DES DONNÉES DANS LE HDFS

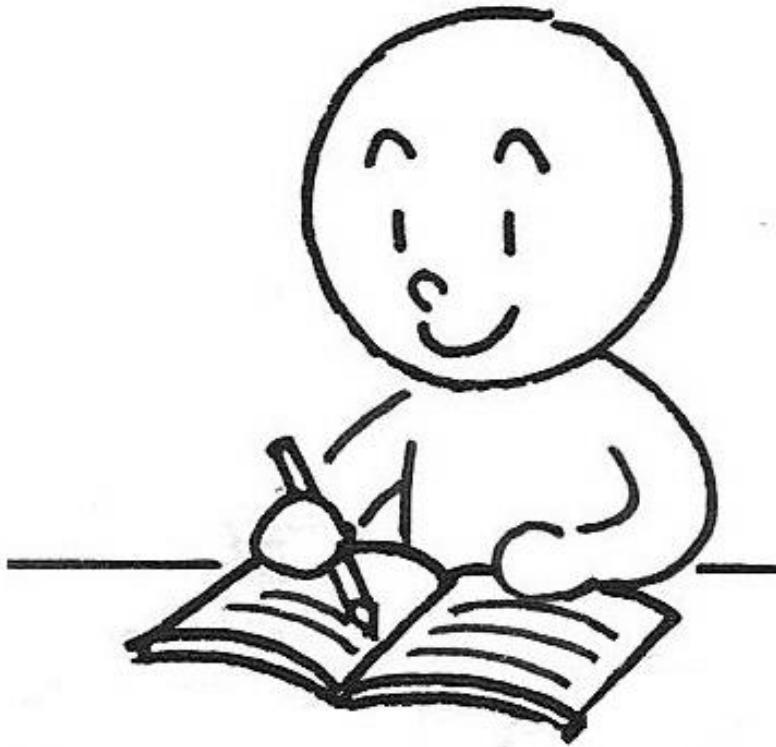


Objectifs



- Cinématique de l'écriture de fichier
- Cinématique de la lecture de fichier
- L'API REST WebHDFS
- Apache Flume
- Apache Sqoop

Cinématique de l'écriture de fichier HDFS



Cinématique de l'écriture de fichier HDFS

► Prenons par exemple le cas de figure suivant :

- ◆ Taille du fichier : 360 Mo
- ◆ Taille du block : 128 Mo
- ◆ Facteur de réPLICATION : 3

Cinématique de l'écriture de fichier HDFS

► **Les étapes suivantes auront lieu pendant l'écriture d'un fichier dans le HDFS :**

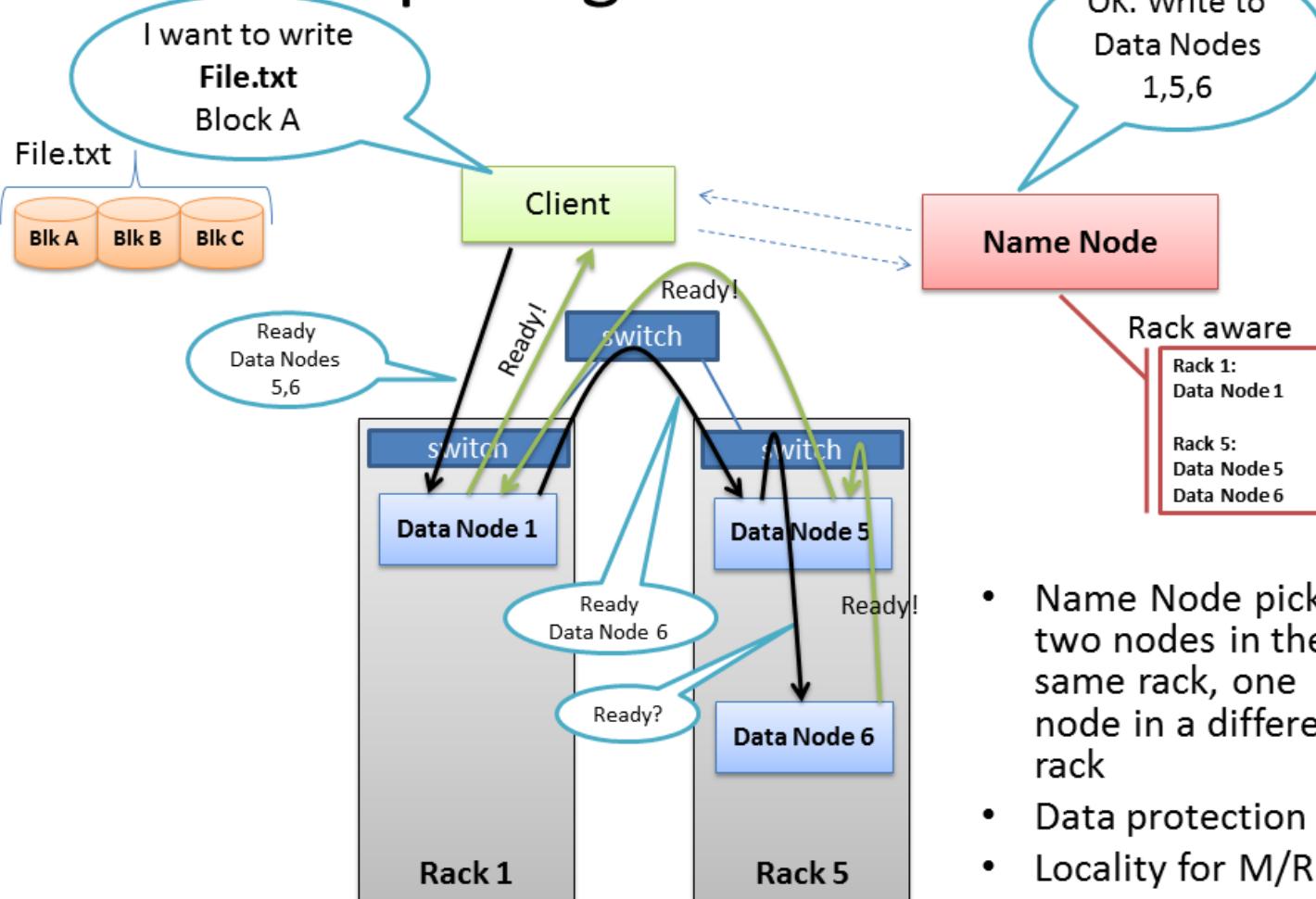
- ◆ Le client HDFS contactera le NameNode pour une demande d'écriture sur les trois blocs, Block 1, Block 2 et Block 3.
- ◆ Le NameNode donnera ensuite au client l'autorisation d'écriture et fournira les adresses IP des DataNodes où les blocs du fichiers seront éventuellement copiés.
- ◆ La sélection des adresses IP de DataNodes est purement aléatoire, en fonction de la disponibilité, du facteur de réPLICATION et de la connaissance du rack.

Cinématique de l'écriture de fichier HDFS

- ◆ Le NameNode fournira au client une liste de 3 adresses IP de DataNodes pour chaque bloc. La liste sera unique pour chaque bloc.
- ◆ Chaque bloc sera copié dans trois DataNodes différents pour maintenir le facteur de réPLICATION cohérent dans tout le cluster.

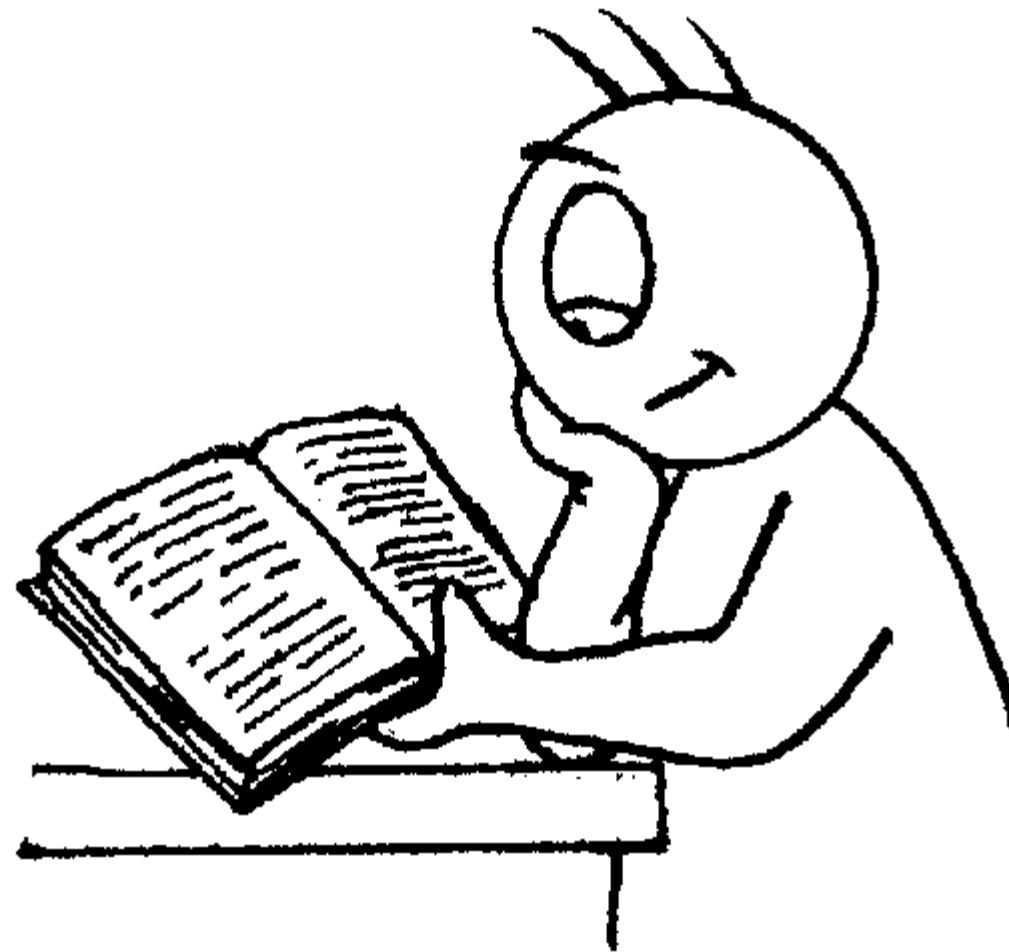
Cinématique de l'écriture de fichier HDFS

Preparing HDFS writes



- Name Node picks two nodes in the same rack, one node in a different rack
- Data protection
- Locality for M/R

Cinématique de la lecture de fichier HDFS



Cinématique de la lecture de fichier HDFS

► **Les étapes suivantes auront lieu pendant la lecture du fichier:**

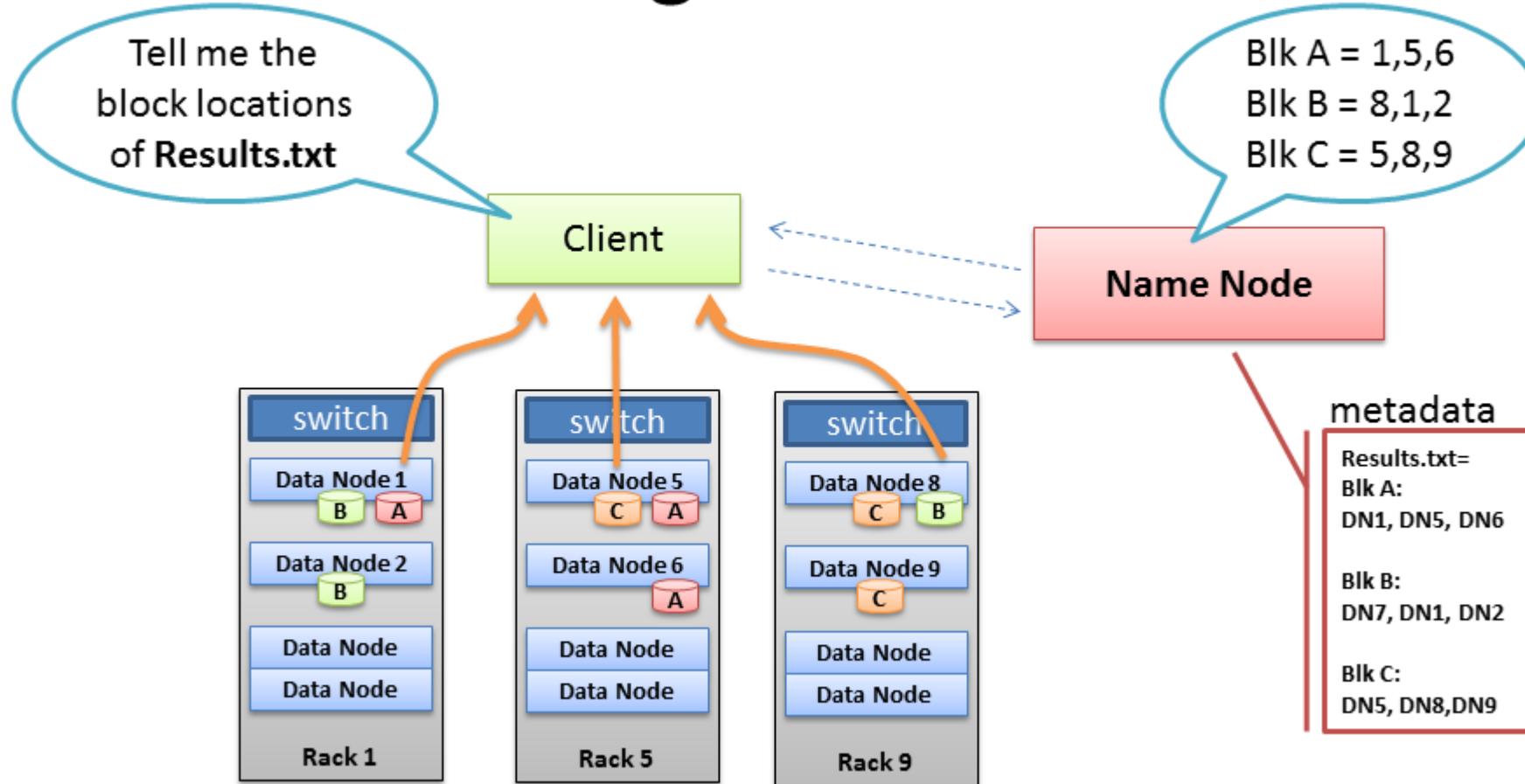
- ◆ Le client contactera le NameNode pour lui demander les métadonnées des blocs du fichier.
- ◆ Le NameNode renverra la liste des DataNodes où chaque bloc (blocs 1, 2 et 3) est stocké.
- ◆ Le client, se connectera aux DataNodes où les blocs sont stockés.

Cinématique de la lecture de fichier HDFS

- ◆ Le client commence à lire les données en parallèle à partir des DataNodes (bloc 1 de DataNode 1 et 2 de DataNode 2, etc).
 - ◆ Une fois que le client obtient tous les blocs du fichiers requis, il combine ces blocs pour former un fichier.
- Lors du traitement de la demande de lecture du client le HDFS sélectionne la réplique la plus proche du client pour réduire la latence de lecture et la consommation de la bande passante.

Cinématique de la lecture de fichier HDFS

Client reading files from HDFS



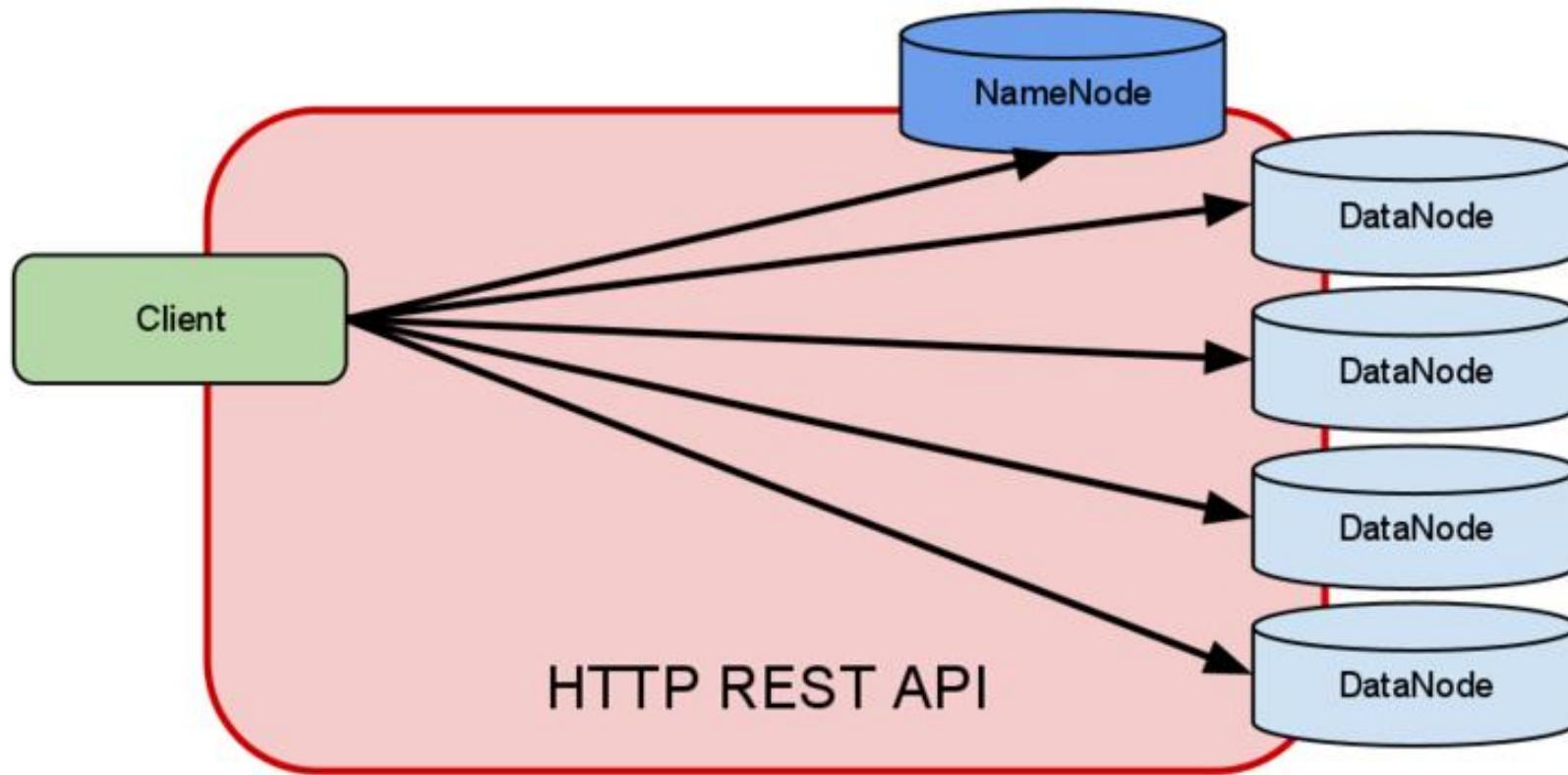
L'API REST WebHDFS



L'API REST WebHDFS

- ▶ **WebHDFS** est une API REST permettant d'accéder à toutes les interfaces du système de fichiers HDFS.
- ▶ Le concept WebHDFS est basé sur des méthodes HTTP telles que GET, PUT, POST et DELETE.

L'API REST WebHDFS



L'API REST WebHDFS

- ▶ **WebHDFS prend en charge toutes les opérations utilisateur HDFS :**
 - ◆ La lecture de fichiers.
 - ◆ L'écrire dans les fichiers.
 - ◆ La création des répertoires.
 - ◆ Le changement des permissions.
 - ◆ Le renomage des fichiers et des répertoires.
- ▶ **Avec WebHDFS, il est possible d'utiliser des outils tels que curl, wget ou tout autre client de services Web, pour accéder aux fichiers d'un cluster Hadoop.**

L'API REST WebHDFS

- ▶ **I'API REST WebHDFS est accessible via l'url :**
 - ◆ <http://host:port/webhdfs/v1/>
 - ◆ Le port par défaut du service est 50070.
- ▶ **L'utilisation de l'API WebHDFS est indépendante du langage de programmation.**
- ▶ **WebHDFS permet aux clients d'accéder à Hadoop à partir de plusieurs langages sans avoir à installer Hadoop.**

L'API REST WebHDFS

- ▶ WebHDFS utilise une authentification sécurisée basée sur les jetons de délégation Kerberos.
- ▶ WebHDFS redirige les appels de lecture et d'écriture de fichier vers les Datanodes correspondants. Il utilise toute la bande passante du cluster Hadoop pour la transmission en continu des données.
- ▶ WebHDFS s'exécute à l'intérieur du NameNode et des DataNodes, il n'y a donc aucun serveur supplémentaire à installer.

L'API REST WebHDFS

- ▶ L'API WebHDFS supporte des opérations avec mes méthodes HTTP GET, POST, PUT et DELETE.
- ▶ HTTP GET :
 - ◆ OPEN : Ouvrir un fichier.
 - ◆ GETFILESTATUS : Renvoie un état de fichier.
 - ◆ LISTSTATUS : Lister les statuts des fichiers ou répertoires
 - ◆ GETFILECHECKSUM : Obtenir la somme de contrôle d'un fichier.

L'API REST WebHDFS

► HTTP PUT :

- ◆ **CREATE** : Créer un fichier.
- ◆ **MKDIRS** : Créer un répertoire.
- ◆ **RENAME** : Renommer un fichier ou répertoire.
- ◆ **SETREPLICATION** : Définir la réplication pour un fichier existant
- ◆ **SETOWNER** : Définir le propriétaire d'un fichier ou répertoire.
- ◆ **SETPERMISSION** : Définir la permission d'un fichier ou répertoire.
- ◆ **SETTIMES** : Définir le temps d'accès d'un fichier.

L'API REST WebHDFS

- ▶ **HTTP POST :**
 - ◆ APPEND : ajouter du contenu dans un fichier existant.
- ▶ **HTTP DELETE :**
 - ◆ DELETE : supprimer un fichier.

L'API REST WebHDFS

▶ Pour plus de détails consulter le guide de référence de l'API WebHDFS :

- ◆ <https://hadoop.apache.org/docs/r2.6.0/hadoop-project-dist/hadoop-hdfs/WebHDFS.html>





|

Lab !

Lab - L'API REST WebHDFS

► Créer un nouveau répertoire dans le HDFS avec la requête :

- ◆ curl -i -X PUT

"<http://127.0.01:50070/webhdfs/v1/mydata?op=MKDIRS&user.name=training&permission=777>"

► Vérifier le statut du répertoire avec la requête :

- ◆ curl -i

"<http://127.0.01:50070/webhdfs/v1/mydata?&op=GETFILESTATUS>"

Lab - L'API REST WebHDFS



```
[21/10/2018 22:20.49] ~
[admin.admin-PC] > curl -i -X PUT "http://hdp.sandbox:50070/webhdfs/v1/mydata?op=MKDIRS&user.name=hdfs&permission=777"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 20:21:19 GMT
Date: Sun, 21 Oct 2018 20:21:19 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 20:21:19 GMT
Date: Sun, 21 Oct 2018 20:21:19 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple&e=1540189279299&s=10914BdxhQpmYdVbzHksR0hA0t4=";
"; Path=/; Expires=Mon, 22-Oct-2018 06:21:19 GMT; HttpOnly
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)

{"boolean":true}
[21/10/2018 22:21.19] ~
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS



```
[21/10/2018 22:28.36] ~
[admin.admin-PC] > curl -i "http://hdp.sandbox:50070/webhdfs/v1/mydata?&op=GETFILESTATUS"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 20:28:56 GMT
Date: Sun, 21 Oct 2018 20:28:56 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 20:28:56 GMT
Date: Sun, 21 Oct 2018 20:28:56 GMT
Pragma: no-cache
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)

{"Filestatus": {"accessTime": 0, "blocksize": 0, "childrenNum": 0, "fileId": 21017, "group": "hdfs", "length": 0, "modificationTime": 1540153279301, "owner": "hdfs", "pathsuffix": "", "permission": "777", "replication": 0, "storagePolicy": 0, "type": "DIRECTORY"}}
[21/10/2018 22:28.56] ~
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS

- ▶ **Créer un fichier dans un répertoire dans le HDFS. La création d'un fichier se déroule en deux étapes :**
 - ◆ Exécuter la requête sur le NameNode, puis suivre la redirection et exécuter l'API WebHDFS sur le DataNode approprié.
 - ◆ Deuxièmement, télécharger le fichier vers le DataNode.

Lab - L'API REST WebHDFS

◆ curl -i -X PUT

"<http://192.168.56.101:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=CREATE&user.name=training&permission=777>"

◆ curl -i -T C:\\mylocaldata\\hello-hadoop.txt "<http://192.168.56.101:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=CREATE&user.name= training&namenoderpcaddress=192.168.56.101:8020&overwrite=false&permission=777>"

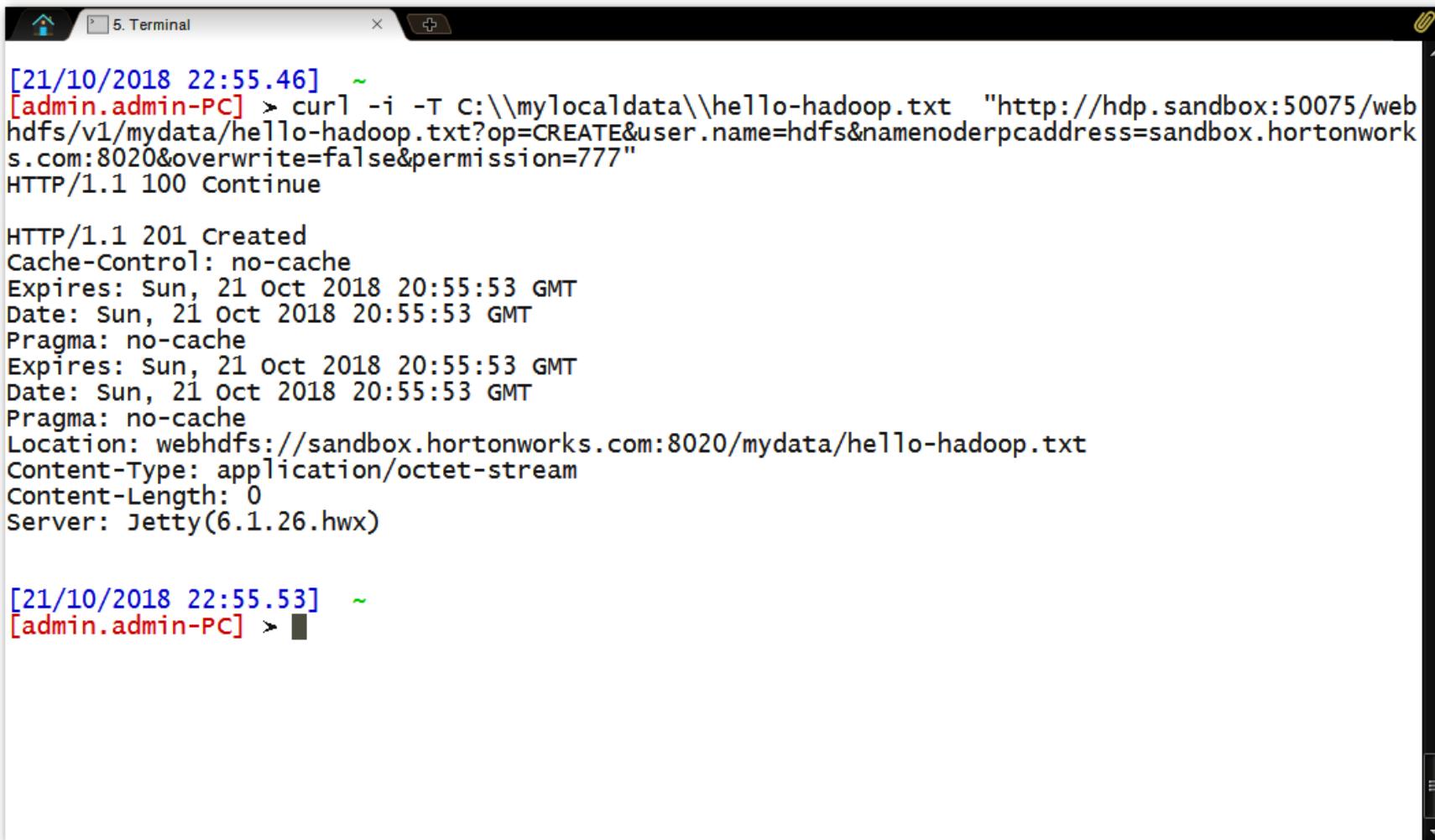
Lab - L'API REST WebHDFS



```
[21/10/2018 22:54.13] ~
[admin.admin-PC] > curl -i -X PUT "http://hdp.sandbox:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=CREATE&user.name=hdfs&permission=777"
HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 20:54:28 GMT
Date: Sun, 21 Oct 2018 20:54:28 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 20:54:28 GMT
Date: Sun, 21 Oct 2018 20:54:28 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple&e=1540191268826&s=hjwgdDut8rLhtT/viHZ6Tznmy2g="; Path=/; Expires=Mon, 22-Oct-2018 06:54:28 GMT; HttpOnly
Location: http://sandbox.hortonworks.com:50075/webhdfs/v1/mydata/hello-hadoop.txt?op=CREATE&user.name=hdfs&namenoderpcaddress=sandbox.hortonworks.com:8020&overwrite=false&permission=777
Content-Type: application/octet-stream
Content-Length: 0
Server: Jetty(6.1.26.hwx)

[21/10/2018 22:54.28] ~
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS



The screenshot shows a terminal window titled "5. Terminal" with a black header bar. The terminal displays the following command and its response:

```
[21/10/2018 22:55.46] ~  
[admin.admin-PC] > curl -i -T C:\\mylocaldata\\hello-hadoop.txt "http://hdp.sandbox:50075/web  
hdfs/v1/mydata/hello-hadoop.txt?op=CREATE&user.name=hdfs&namenoderpcaddress=sandbox.hortonwork  
s.com:8020&overwrite=false&permission=777"  
HTTP/1.1 100 Continue  
  
HTTP/1.1 201 Created  
Cache-Control: no-cache  
Expires: Sun, 21 Oct 2018 20:55:53 GMT  
Date: Sun, 21 Oct 2018 20:55:53 GMT  
Pragma: no-cache  
Expires: Sun, 21 Oct 2018 20:55:53 GMT  
Date: Sun, 21 Oct 2018 20:55:53 GMT  
Pragma: no-cache  
Location: webhdfs://sandbox.hortonworks.com:8020/mydata/hello-hadoop.txt  
Content-Type: application/octet-stream  
Content-Length: 0  
Server: Jetty(6.1.26.hwx)  
  
[21/10/2018 22:55.53] ~  
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS

► Afficher le contenu d'un fichier avec la requête :

- ◆ curl -i -L

"<http://192.168.56.101:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=OPEN&user.name=training>"

Lab - L'API REST WebHDFS

```
[21/10/2018 23:05.52] ~ [admin.admin-PC] > curl -i -L "http://hdp.sandbox:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=OPEN&user.name=hdfs"
HTTP/1.1 307 TEMPORARY_REDIRECT
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 21:07:05 GMT
Date: Sun, 21 Oct 2018 21:07:05 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 21:07:05 GMT
Date: Sun, 21 Oct 2018 21:07:05 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple&e=1540192025707&s=jZwrer8wbcr1usScV/yjeG3fz7A="; Path=/; Expires=Mon, 22-Oct-2018 07:07:05 GMT; HttpOnly
Location: http://sandbox.hortonworks.com:50075/webhdfs/v1/mydata/hello-hadoop.txt?op=OPEN&user.name=hdfs&namenoderpcaddress=sandbox.hortonworks.com:8020&offset=0
Content-Type: application/octet-stream
Content-Length: 0
Server: Jetty(6.1.26.hwx)

HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 21:07:05 GMT
Date: Sun, 21 Oct 2018 21:07:05 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 21:07:05 GMT
Date: Sun, 21 Oct 2018 21:07:05 GMT
Pragma: no-cache
Content-Length: 15
Content-Type: application/octet-stream
Access-Control-Allow-Methods: GET
```

Lab - L'API REST WebHDFS

► Lister le contenu d'un répertoire dans le HDFS avec la requête :

- ◆ curl -i

"<http://127.0.0.1:50070/webhdfs/v1/mydata/?op=LISTSTATUS>"

Lab - L'API REST WebHDFS



The screenshot shows a terminal window titled "5. Terminal" with a black header bar. The terminal displays a curl command executed on an admin PC to query the WebHDFS API for file status.

```
[21/10/2018 23:16.21] ~
[admin.admin-PC] > curl -i "http://hdp.sandbox:50070/webhdfs/v1/mydata/?op=LISTSTATUS"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 21:16:27 GMT
Date: Sun, 21 Oct 2018 21:16:27 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 21:16:27 GMT
Date: Sun, 21 Oct 2018 21:16:27 GMT
Pragma: no-cache
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)

{"Filestatuses": {"Filestatus": [
    {"accessTime": 1540155353157, "blocksize": 134217728, "childrenNum": 0, "fileId": 21156, "group": "hdfs",
     "length": 15, "modificationTime": 1540155353321, "owner": "hdfs", "pathSuffix": "hello-hadoop.txt",
     "permission": "777", "replication": 1, "storagePolicy": 0, "type": "FILE"}]}}

[21/10/2018 23:16.27] ~
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS

► Supprimer un fichier dans le HDFS avec la requête :

- ◆ curl -i -X DELETE

"<http://127.0.0.1:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=DELETE&user.name=training>"

Lab - L'API REST WebHDFS



```
[21/10/2018 23:23.15] ~
[admin.admin-PC] > curl -i -X DELETE "http://hdp.sandbox:50070/webhdfs/v1/mydata/hello-hadoop.txt?op=DELETE&user.name=hdfs"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 21:23:20 GMT
Date: Sun, 21 Oct 2018 21:23:20 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 21:23:20 GMT
Date: Sun, 21 Oct 2018 21:23:20 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple&e=1540193000803&s=i1GsSucnHZi53LPhtDgZOVSrlnU=";
"; Path=/; Expires=Mon, 22-Oct-2018 07:23:20 GMT; HttpOnly
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)

{"boolean":true}
[21/10/2018 23:23.20] ~
[admin.admin-PC] >
```

Lab - L'API REST WebHDFS

► Supprimer un fichier dans le HDFS avec la requête :

- ◆ curl -i -X DELETE

"<http://127.0.0.1:50070/webhdfs/v1/mydata/?op=DELETE&user.name=training&destination=/mydata>"

Lab - L'API REST WebHDFS



```
[21/10/2018 23:25.32] ~
[admin.admin-PC] > curl -i -X DELETE "http://hdp.sandbox:50070/webhdfs/v1/mydata/?op=DELETE&user.name=hdfs&destination=/mydata"
HTTP/1.1 200 OK
Cache-Control: no-cache
Expires: Sun, 21 Oct 2018 21:25:35 GMT
Date: Sun, 21 Oct 2018 21:25:35 GMT
Pragma: no-cache
Expires: Sun, 21 Oct 2018 21:25:35 GMT
Date: Sun, 21 Oct 2018 21:25:35 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple&e=1540193135954&s=xGsbwXJ4yohs0Uxh8h6ibb90Rek=";
"; Path=/; Expires=Mon, 22-Oct-2018 07:25:35 GMT; HttpOnly
Transfer-Encoding: chunked
Server: Jetty(6.1.26.hwx)

{"boolean":true}
[21/10/2018 23:25.35] ~
[admin.admin-PC] >
```

Apache Flume



Apache Flume



Apache Flume

- ▶ Un canal utilise l'eau pour transporter des objets le long du canal.
- ▶ Apache Flume collecte, regroupe et dirige les flux de données dans Hadoop en utilisant les mêmes concepts.
- ▶ Apache Flume est un système permettant de collecter, d'agréger et de transférer efficacement de grandes quantités de données provenant de nombreuses sources différentes vers le HDFS.

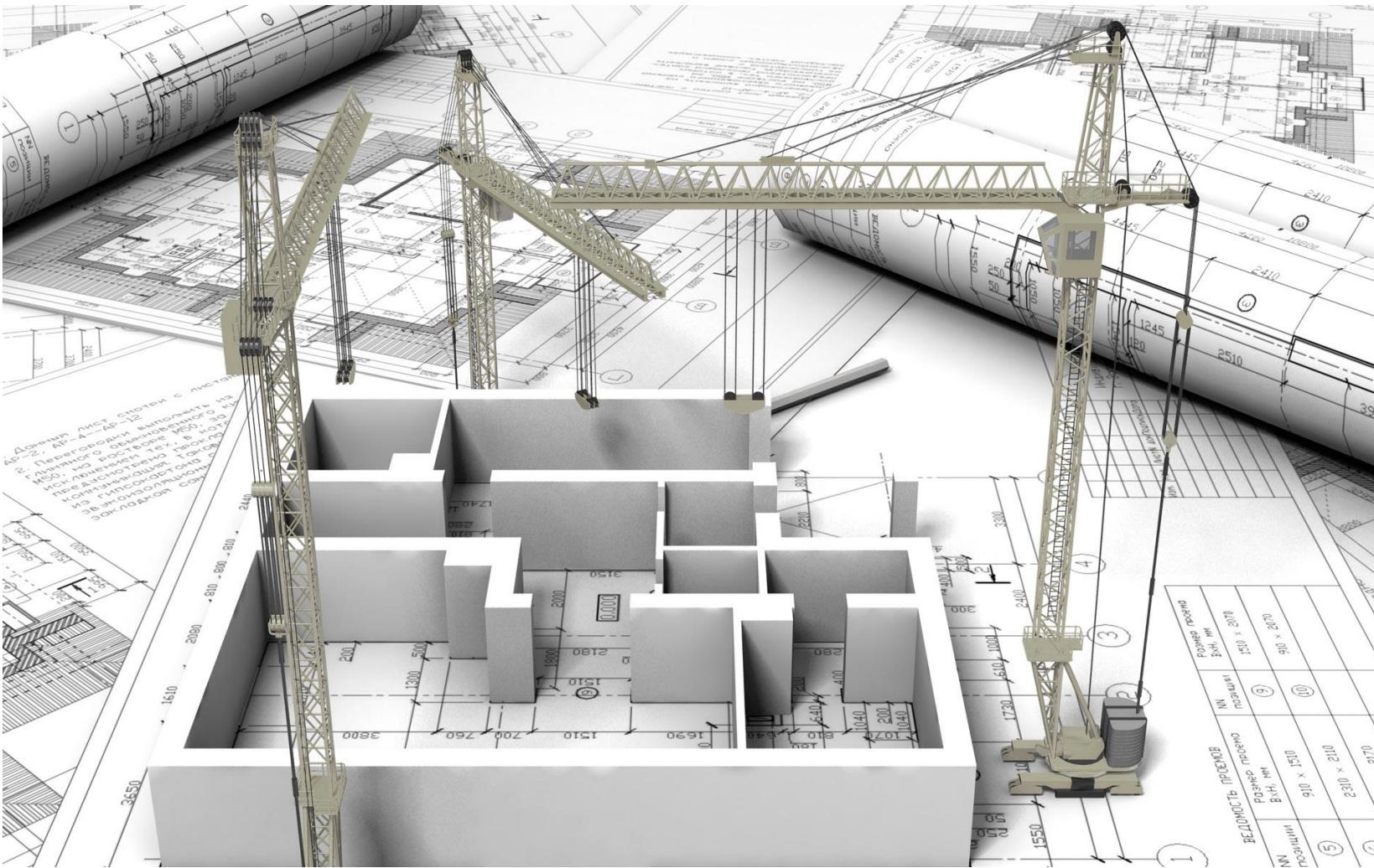
Apache Flume

- ▶ Flume peut s'intégrer avec différentes sources de données pour traiter et envoyer des données à des destinations définies.
- ▶ Flume utilise un modèle producteur-consommateur pour gérer les événements où la source est le producteur et le récepteur est le consommateur des événements.

Flume est conçu pour transmettre des événements en temps réel où le flux de données est continu et son volume est relativement important.



Apache Flume - Architecture



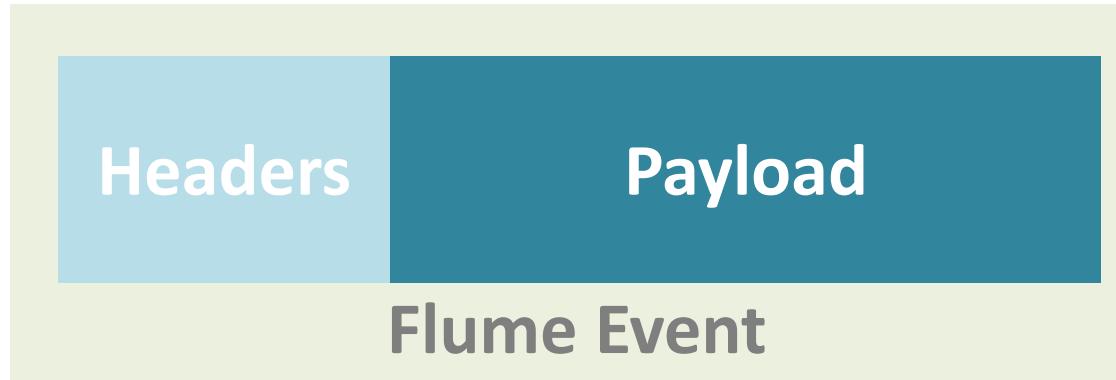
Apache Flume – Événement



Apache Flume – Événement

- ▶ Flume représente les données en tant qu'événements.
- ▶ Les événements sont des structures de données simples, avec un corps et un ensemble d'en-têtes.
- ▶ Le corps de l'événement est un tableau d'octets qui correspond généralement aux données utiles que Flume transporte.

Apache Flume – Événement



Apache Flume – Événement

- ▶ Les en-têtes sont représentés sous forme de Map avec des clés et des valeurs.
- ▶ Les en-têtes sont destinés à des fins de routage et de suivi des priorité, gravité des événements envoyés.
- ▶ Chaque événement doit être essentiellement un enregistrement indépendant.

Apache Flume – Agent



Apache Flume – Agent

- ▶ L'unité de déploiement la plus simple de Flume s'appelle un agent Flume.
- ▶ Un agent est une application Java qui reçoit ou génère des données et les met en mémoire tampon jusqu'à ce qu'elles soient éventuellement écrites sur le prochain agent ou sur un système de stockage ou d'indexation tel que le HDFS, HBase, etc.

Apache Flume – Agent

- ▶ **Un agent Flume est constitué de trois composants principaux:**
 - ◆ Les sources.
 - ◆ Les channels ou canaux.
 - ◆ Les sinks ou récepteurs.
- ▶ **Flume ne limite pas le nombre de sources, de channels et de sinks dans un agent.**

Apache Flume – Agent



Apache Flume – Sources



Apache Flume – Sources

- ▶ Flume accepte les données d'une application ou d'un serveur tel qu'un serveur web.
- ▶ Les sources sont des composants actifs qui reçoivent des données d'une autre application produisant les données.
- ▶ Les sources peuvent être piloté par des événements ou interrogées.
- ▶ Les sources interrogées sont questionnées en boucle par les exéuteurs de source Flume :
 - ◆ Exemple: générateur de séquence.

Apache Flume – Sources

- ▶ **Les sources pilotées par les événements fournissent à la source Flume des événements.**
 - ◆ Exemple: fichier journal d'un serveur Web.
- ▶ **Les sources peuvent écouter un ou plusieurs ports réseau pour recevoir des données ou peuvent lire des données du système de fichiers local.**

Apache Flume – Sources

- ▶ La source Flume envoie les événements qu'elle reçoit à un ou plusieurs canaux.
- ▶ Chaque source doit être connectée à au moins un canal.
- ▶ Une source peut écrire sur plusieurs canaux, en répliquant les événements sur tout ou partie des canaux, en fonction de certains critères.

Apache Flume – Channels



Apache Flume – Channels

- ▶ Un canal est un point de connexion permettant de transférer des événements d'une source à un récepteur.
- ▶ Un canal est un composant passif qui met en mémoire tampon les données reçues par l'agent, mais non encore écrites sur un autre agent ou sur un système de stockage.
- ▶ Comme il est asynchrone, le canal n'est pas obligé d'envoyer des événements au récepteur au même rythme qu'il les reçoit de la source.
- ▶ Les événements sont stockés jusqu'à ce qu'un récepteur les enlève pour un transport ultérieur.

Apache Flume – Channels

Les canaux se comportent comme des files d'attente.



Apache Flume – Channels

- ▶ Flume propose différents niveaux de fiabilité et de performance des canaux :
 - ◆ In-memory : le canal est rapide mais présente un risque de perte de données en cas de défaillance de l'agent.
 - ◆ Persistant : le canal assure un stockage durable et fiable des événements avec des capacités de récupération en cas de défaillance d'un agent.

Apache Flume - Sinks



Apache Flume - Sinks

- ▶ **Les récepteurs interrogent leurs canaux respectifs en continu pour lire et supprimer les événements.**
- ▶ **Les récepteurs repoussent les événements à l'agent suivant ou à la destination finale.**
- ▶ **Une fois que les données sont en toute sécurité arrivées au prochain agent ou à leur destination, les récepteurs informent les canaux, via des validations de transaction, que ces événements peuvent maintenant être supprimés des canaux.**

Apache Flume – Interceptors



Apache Flume – Interceptors

- ▶ Les intercepteurs permettent l'inspection et la transformation des données lors de leur circulation dans le flux.
- ▶ Offre la possibilité de modifier ou de supprimer des événements en fonction d'un critère donné.
- ▶ Flume prend en charge les chaînes d'intercepteurs.

Apache Flume – Channel selectors

- ▶ Les sélecteurs de canaux sont les composants qui décident des canaux sur laquelle chaque événement doit être écrit.
- ▶ Les intercepteurs peuvent être utilisés pour insérer ou supprimer des événements afin que les sélecteurs de canaux puissent appliquer certains critères à ces événements pour décider sur quels canaux les événements doivent être écrits.

Apache Flume – Channel selectors

► Les sélecteurs de canaux peuvent appliquer des critères de filtrage arbitraires aux événements pour décider sur quels canaux chaque événement doit être écrit, et quels canaux sont obligatoires et facultatifs.

Apache Flume – Sink processors

- ▶ Les groupes de sinks permettent aux utilisateurs de regrouper plusieurs sinks dans une même entité.
- ▶ Les sink processors sont utilisés pour appeler un sink particulier à partir d'un groupe de sinks.
- ▶ Le sink processor peut être utilisé pour fournir des capacités d'équilibrage de charge.

Apache Flume – Fichier de configuration

- ▶ La configuration de l'agent Flume est stockée dans un fichier de configuration local.
- ▶ Il s'agit d'un fichier texte qui suit le format du fichier de propriétés Java.
- ▶ Les configurations d'un ou de plusieurs agents peuvent être spécifiées dans le même fichier de configuration.

Apache Flume – Fichier de configuration

- ▶ Le fichier de configuration comprend les propriétés de chaque source, sink et channel dans un agent et comment ils sont câblés ensemble pour former des flux de données.
- ▶ Chaque composant du flux a un nom, un type et un ensemble de propriétés spécifiques au type et à l'instanciation.

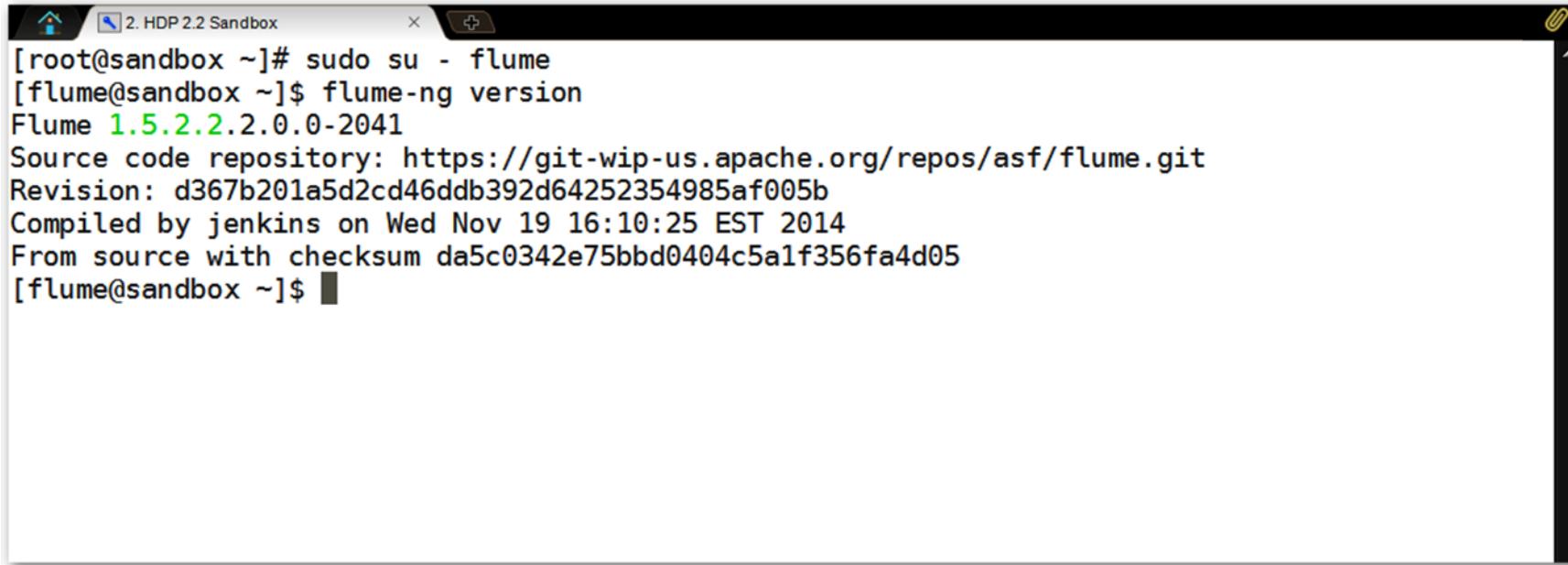


|

Lab !

Lab – Prise en main de Flume

► Taper la commande *flume-ng version* à partir de l'invite de commande pour vérifier que Flume est installé sur votre cluster et récupérer le numéro de version.



The screenshot shows a terminal window titled "2. HDP 2.2 Sandbox". The command [root@sandbox ~]# sudo su - flume was run, followed by [flume@sandbox ~]\$ flume-ng version. The output displays the Flume version (1.5.2.2.2.0.0-2041), source code repository URL (<https://git-wip-us.apache.org/repos/asf/flume.git>), revision (d367b201a5d2cd46ddb392d64252354985af005b), compilation details (compiled by jenkins on Wed Nov 19 16:10:25 EST 2014), and checksum information (From source with checksum da5c0342e75bb0404c5a1f356fa4d05). The prompt [flume@sandbox ~]\$ is visible at the end of the output.

```
[root@sandbox ~]# sudo su - flume
[flume@sandbox ~]$ flume-ng version
Flume 1.5.2.2.2.0.0-2041
Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git
Revision: d367b201a5d2cd46ddb392d64252354985af005b
Compiled by jenkins on Wed Nov 19 16:10:25 EST 2014
From source with checksum da5c0342e75bb0404c5a1f356fa4d05
[flume@sandbox ~]$
```

Lab – Prise en main de Flume

- ▶ La commande `flume-ng help` doit renvoyer les instructions d'utilisation suivantes d'Apache Flume.
- ▶ Pour plus de détails consulter le guide de référence de Flume :
 - ◆ <https://flume.apache.org/releases/content/1.5.2/FlumeUserGuide.pdf>



Lab - Ingestion de données avec Flume

```
[flume@sandbox ~]$ flume-ng help
Usage: /usr/hdp/2.2.0.0-2041/flume/bin/flume-ng.distro <command> [options]...

commands:
  help          display this help text
  agent         run a Flume agent
  avro-client   run an avro Flume client
  password      create a password file for use in flume config
  version       show Flume version info

global options:
  --conf,-c <conf>    use configs in <conf> directory
  --classpath,-C <cp>  append to the classpath
  --dryrun,-d          do not actually start Flume, just print the command
  --plugins-path <dirs> colon-separated list of plugins.d directories. See the
                       plugins.d section in the user guide for more details.
                       Default: $FLUME_HOME/plugins.d
  -Dproperty=value    sets a Java system property value
  -Xproperty=value    sets a Java -X option

agent options:
  --conf-file,-f <file> specify a config file (required)
  --name,-n <name>     the name of this agent (required)
  --help,-h            display help text

avro-client options:
```

Lab - Ingestion de données avec Flume

- ▶ Pour utiliser Flume, il faut démarrer un agent.
- ▶ Un agent a un fichier de configuration associé qui définit ses sources et ses sinks.
- ▶ La commande permettant de démarrer un agent ressemble à ceci :
 - ◆ `flume-ng agent -n agent_name -c conf -f myagent_conf.conf`
 - ◆ `myagent.conf` est le fichier de configuration.

Lab - Ingestion de données avec Flume

► Le fichier de configuration de l'agent suivant illustre la diffusion en continu du fichier journal d'un serveur Apache sur le HDFS en tant que séquence d'événements :



```
[flume@sandbox ~]$ cd /home/flume/
[flume@sandbox ~]$ touch my_apache_log_agent.conf
[flume@sandbox ~]$ chmod 777 my_apache_log_agent.conf
[flume@sandbox ~]$ vi my_apache_log_agent.conf
```

Lab - Ingestion de données avec Flume



The screenshot shows a terminal window with two tabs open: "2. HDP 2.2 Sandbox" and "6. HDP 2.2 Sandbox". The "6. HDP 2.2 Sandbox" tab is active and displays a Flume configuration file named "my_apache_log_agent.conf". The configuration defines a source (apache_log) that reads from a file, a channel (memoryChannel), and a sink (mycluster) that writes to HDFS. The configuration is as follows:

```
my_apache_log_agent.sources = apache_log
my_apache_log_agent.channels = memoryChannel
my_apache_log_agent.sinks = mycluster
##### Sources
my_apache_log_agent.sources.apache_log.type = exec
my_apache_log_agent.sources.apache_log.command = tail -F /home/flume/my_apache_logs.txt
my_apache_log_agent.sources.apache_log.batchSize = 1
my_apache_log_agent.sources.apache_log.channels = memoryChannel
##### Channels
my_apache_log_agent.channels.memoryChannel.type = memory
my_apache_log_agent.channels.memoryChannel.capacity = 100
my_apache_log_agent.channels.memoryChannel.transactionCapacity = 100
##### Sinks
my_apache_log_agent.sinks.mycluster.type = hdfs
my_apache_log_agent.sinks.mycluster.hdfs.path=/mydata/flumedata/my_apache_logs
my_apache_log_agent.sinks.mycluster.channel = memoryChannel
```

At the bottom of the terminal window, the status message "my_apache_log_agent.conf" 16L, 795C is visible.

Lab - Ingestion de données avec Flume

- ▶ Le nom de l'agent est `my_apache_log_agent`.
- ▶ Les noms du sink, de la source et du channel sont arbitraires.
- ▶ Cet agent Flume a une source nommée `apache_log`.
- ▶ La source `apache_log` est de type `exec`, ce qui signifie qu'elle exécute une commande Unix donnée.

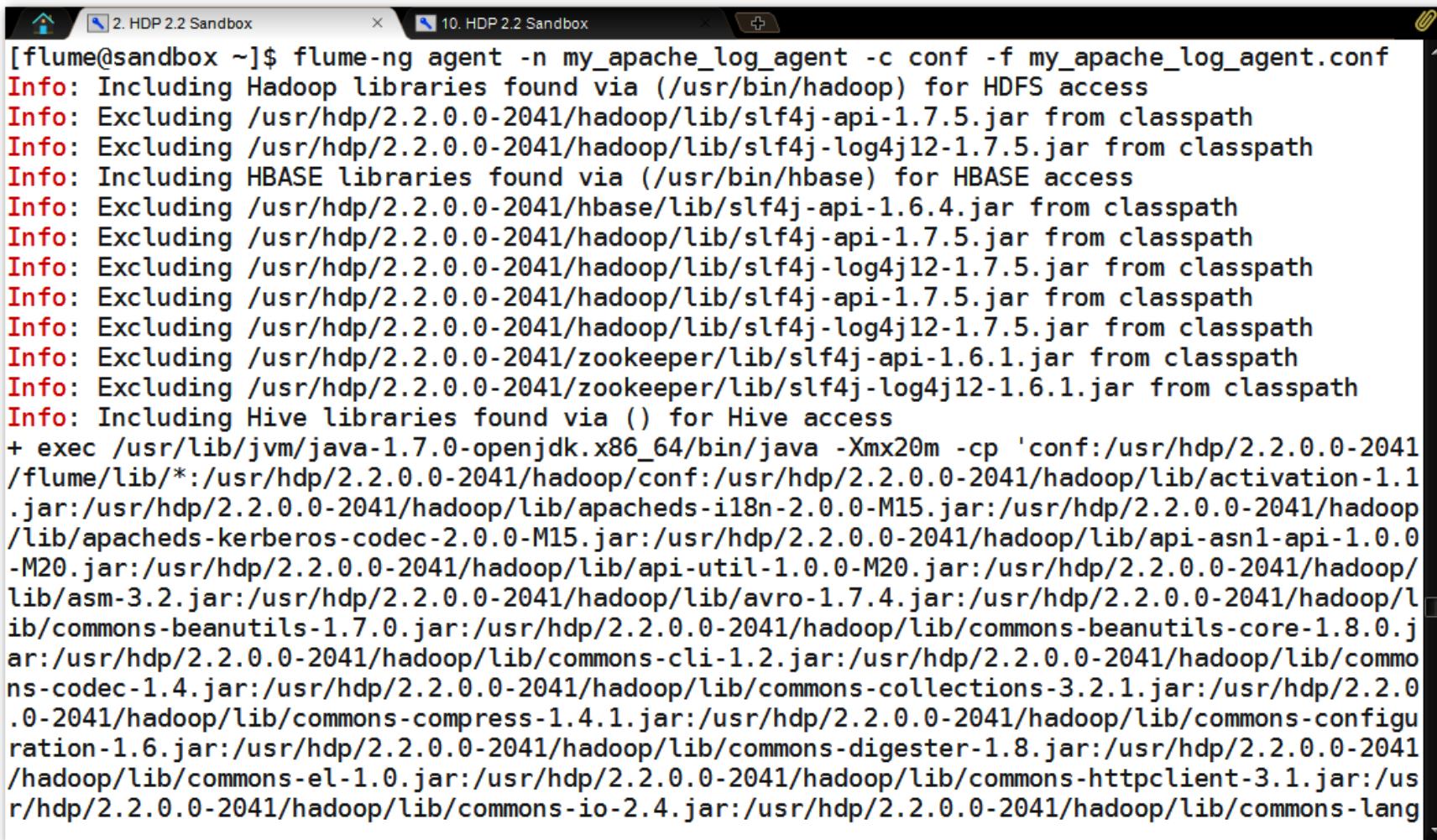
Lab - Ingestion de données avec Flume

- ▶ L'agent possède un sink nommé mycluster, qui envoie les événements à un fichier de séquence situé dans un dossier spécifié dans HDFS.
- ▶ L'agent possède un channel nommé memoryChannel.
- ▶ MemoryChannel est configuré avec un type memory, ce qui signifie qu'il stocke les événements en mémoire.

Lab - Ingestion de données avec Flume

- ▶ Pour démarrer l'agent de transfert `my_apache_log_agent`, exéutez la commande suivante:
 - ◆ `flume-ng agent -n my_apache_log_agent -c conf -f my_apache_log_agent.conf`

Lab - Ingestion de données avec Flume



The screenshot shows a terminal window titled "2. HDP 2.2 Sandbox" with the command `flume-ng agent -n my_apache_log_agent -c conf -f my_apache_log_agent.conf` running. The output is as follows:

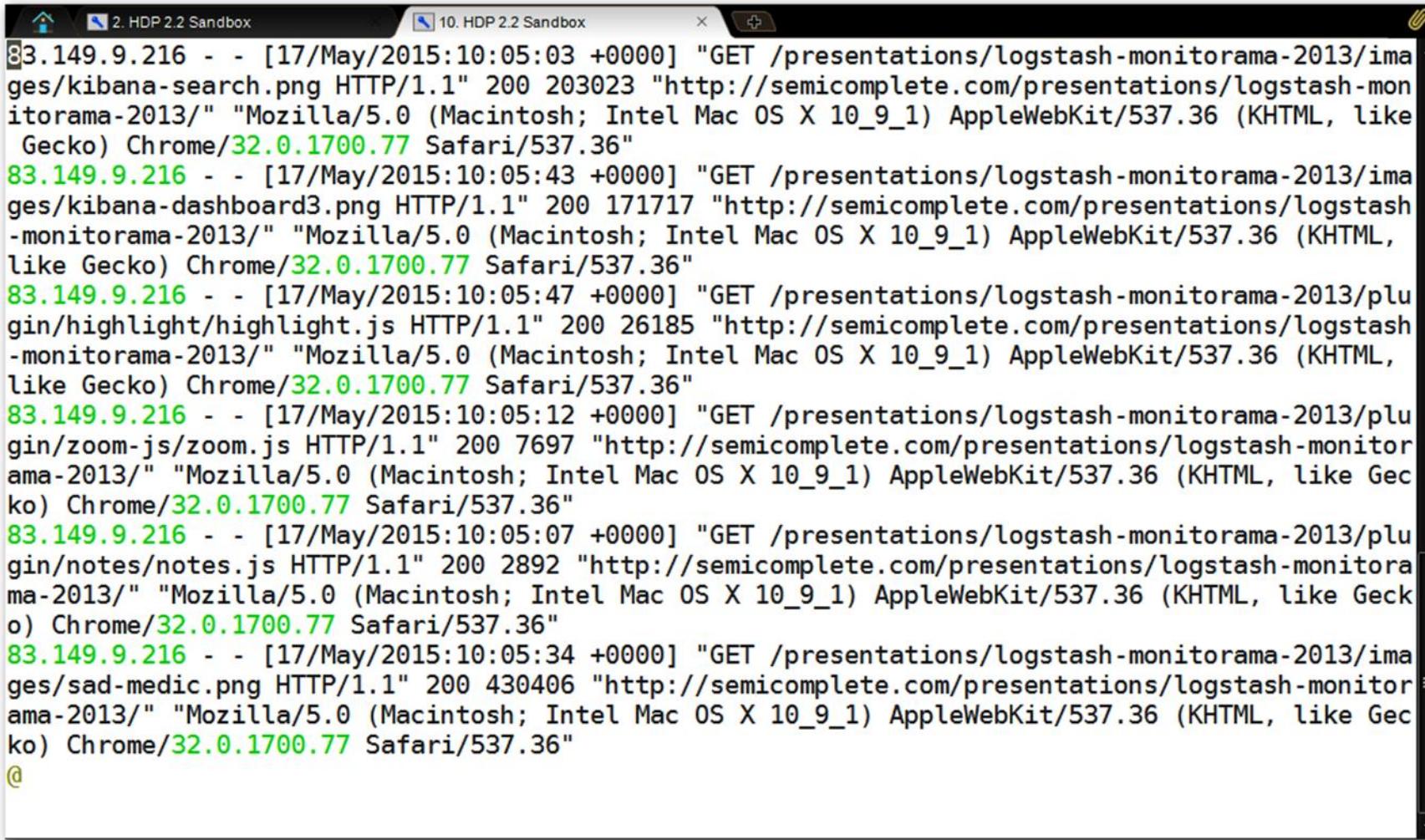
```
[flume@sandbox ~]$ flume-ng agent -n my_apache_log_agent -c conf -f my_apache_log_agent.conf
Info: Including Hadoop libraries found via (/usr/bin/hadoop) for HDFS access
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-api-1.7.5.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar from classpath
Info: Including HBASE libraries found via (/usr/bin/hbase) for HBASE access
Info: Excluding /usr/hdp/2.2.0.0-2041/hbase/lib/slf4j-api-1.6.4.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-api-1.7.5.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-api-1.7.5.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/zookeeper/lib/slf4j-api-1.6.1.jar from classpath
Info: Excluding /usr/hdp/2.2.0.0-2041/zookeeper/lib/slf4j-log4j12-1.6.1.jar from classpath
Info: Including Hive libraries found via () for Hive access
+ exec /usr/lib/jvm/java-1.7.0-openjdk.x86_64/bin/java -Xmx20m -cp 'conf:/usr/hdp/2.2.0.0-2041/flume/lib/*:/usr/hdp/2.2.0.0-2041/hadoop/conf:/usr/hdp/2.2.0.0-2041/hadoop/lib/activation-1.1.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/apacheds-i18n-2.0.0-M15.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/api-asn1-api-1.0.0-M20.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/api-util-1.0.0-M20.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/asm-3.2.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/avro-1.7.4.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-beanutils-1.7.0.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-beanutils-core-1.8.0.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-cli-1.2.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-codec-1.4.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-collections-3.2.1.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-compress-1.4.1.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-configuration-1.6.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-digester-1.8.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-el-1.0.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-httpclient-3.1.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-io-2.4.jar:/usr/hdp/2.2.0.0-2041/hadoop/lib/commons-lang-
```

Lab - Ingestion de données avec Flume

```
fs
18/10/25 20:22:00 INFO hdfs.HDFSEventSink: Hadoop Security enabled: false
18/10/25 20:22:00 INFO node.AbstractConfigurationProvider: Channel memoryChannel connected to [apache_log, mycluster]
18/10/25 20:22:00 INFO node.Application: Starting new configuration:{ sourceRunners:{apache_log=EventDrivenSourceRunner: { source:org.apache.flume.source.ExecSource{name:apache_log,state:IDLE} }} sinkRunners:{mycluster=SinkRunner: { policy:org.apache.flume.sink.DefaultSinkProcessor@9f9fec0 counterGroup:{ name:null counters:{} } }} channels:{memoryChannel=org.apache.flume.channel.MemoryChannel{name: memoryChannel}} }
18/10/25 20:22:00 INFO node.Application: Starting Channel memoryChannel
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type : CHANNEL, name: memoryChannel: Successfully registered new MBean.
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Component type: CHANNEL, name: memoryChannel started
18/10/25 20:22:00 INFO node.Application: Starting Sink mycluster
18/10/25 20:22:00 INFO node.Application: Starting Source apache_log
18/10/25 20:22:00 INFO source.ExecSource: Exec source starting with command:tail -F /home/flume/my_apache_logs.txt
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type : SINK, name: mycluster: Successfully registered new MBean.
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Component type: SINK, name: mycluster started
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Monitored counter group for type : SOURCE, name: apache_log: Successfully registered new MBean.
18/10/25 20:22:00 INFO instrumentation.MonitoredCounterGroup: Component type: SOURCE, name: apache_log started
```

Lab - Ingestion de données avec Flume

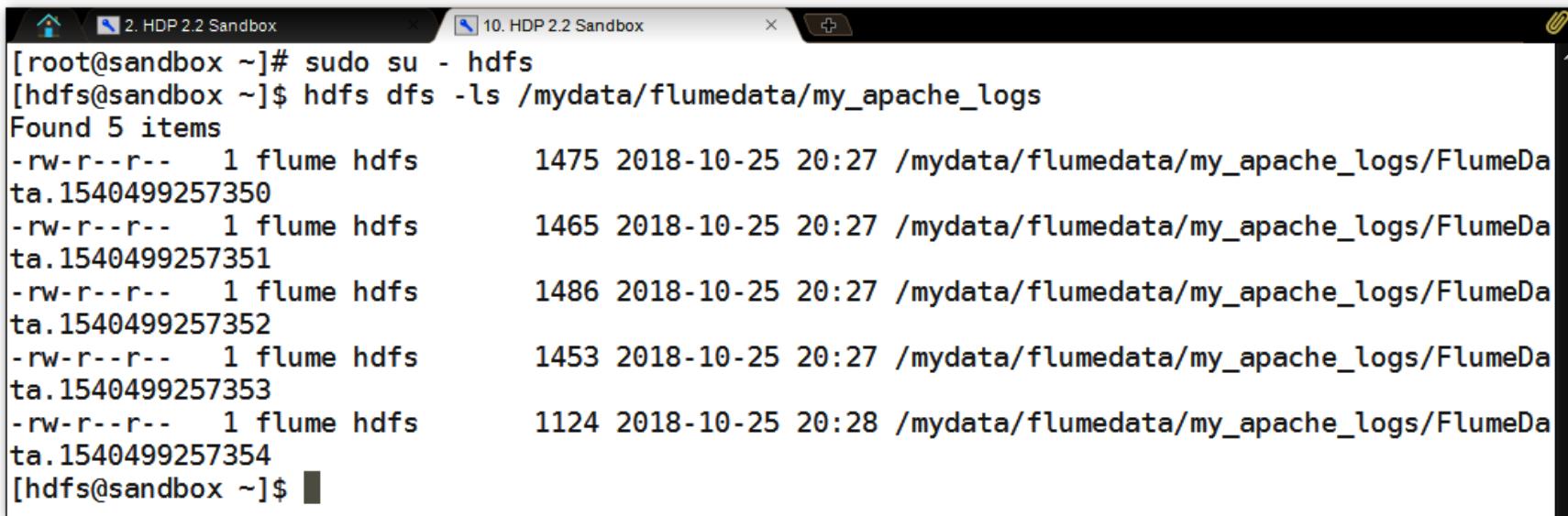
► Pour stimuler le transfert de données avec Flume générer des données dans le fichier `my_apache_logs.txt` scruté en continu par l'agent Flume :



```
2. HDP 2.2 Sandbox 10. HDP 2.2 Sandbox
83.149.9.216 - - [17/May/2015:10:05:03 +0000] "GET /presentations/logstash-monitorama-2013/images/kibana-search.png HTTP/1.1" 200 203023 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [17/May/2015:10:05:43 +0000] "GET /presentations/logstash-monitorama-2013/images/kibana-dashboard3.png HTTP/1.1" 200 171717 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [17/May/2015:10:05:47 +0000] "GET /presentations/logstash-monitorama-2013/plugin/highlight/highlight.js HTTP/1.1" 200 26185 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [17/May/2015:10:05:12 +0000] "GET /presentations/logstash-monitorama-2013/plugin/zoom-js/zoom.js HTTP/1.1" 200 7697 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [17/May/2015:10:05:07 +0000] "GET /presentations/logstash-monitorama-2013/plugin/notes/notes.js HTTP/1.1" 200 2892 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [17/May/2015:10:05:34 +0000] "GET /presentations/logstash-monitorama-2013/images/sad-medic.png HTTP/1.1" 200 430406 "http://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36"
@
```

Lab - Ingestion de données avec Flume

- ▶ Vérifier les fichiers générés dans le HDFS dans avec la commande ls dans le répertoire:
 - ◆ /mydata/flumedata/my_apache_logs

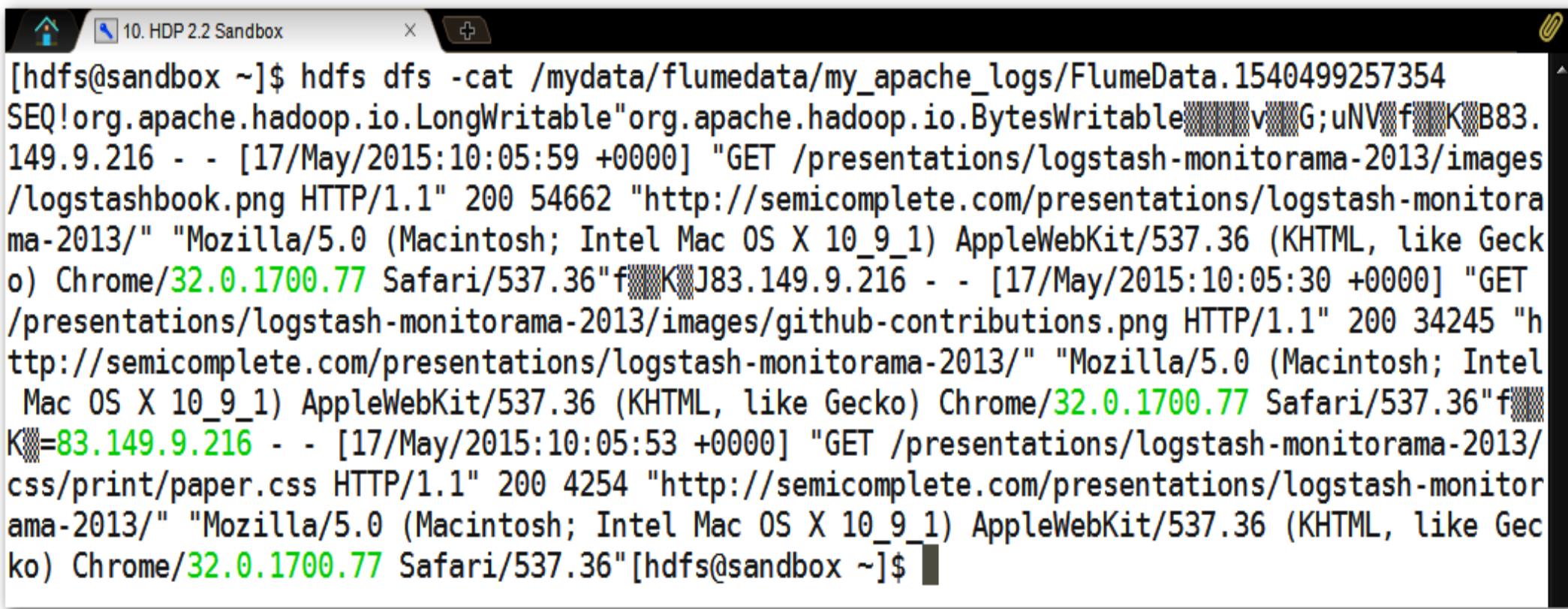


```
[root@sandbox ~]# sudo su - hdfs
[hdfs@sandbox ~]$ hdfs dfs -ls /mydata/flumedata/my_apache_logs
Found 5 items
-rw-r--r-- 1 flume hdfs      1475 2018-10-25 20:27 /mydata/flumedata/my_apache_logs/FlumeDa
ta.1540499257350
-rw-r--r-- 1 flume hdfs      1465 2018-10-25 20:27 /mydata/flumedata/my_apache_logs/FlumeDa
ta.1540499257351
-rw-r--r-- 1 flume hdfs      1486 2018-10-25 20:27 /mydata/flumedata/my_apache_logs/FlumeDa
ta.1540499257352
-rw-r--r-- 1 flume hdfs      1453 2018-10-25 20:27 /mydata/flumedata/my_apache_logs/FlumeDa
ta.1540499257353
-rw-r--r-- 1 flume hdfs     1124 2018-10-25 20:28 /mydata/flumedata/my_apache_logs/FlumeDa
ta.1540499257354
[hdfs@sandbox ~]$
```

Lab - Ingestion de données avec Flume

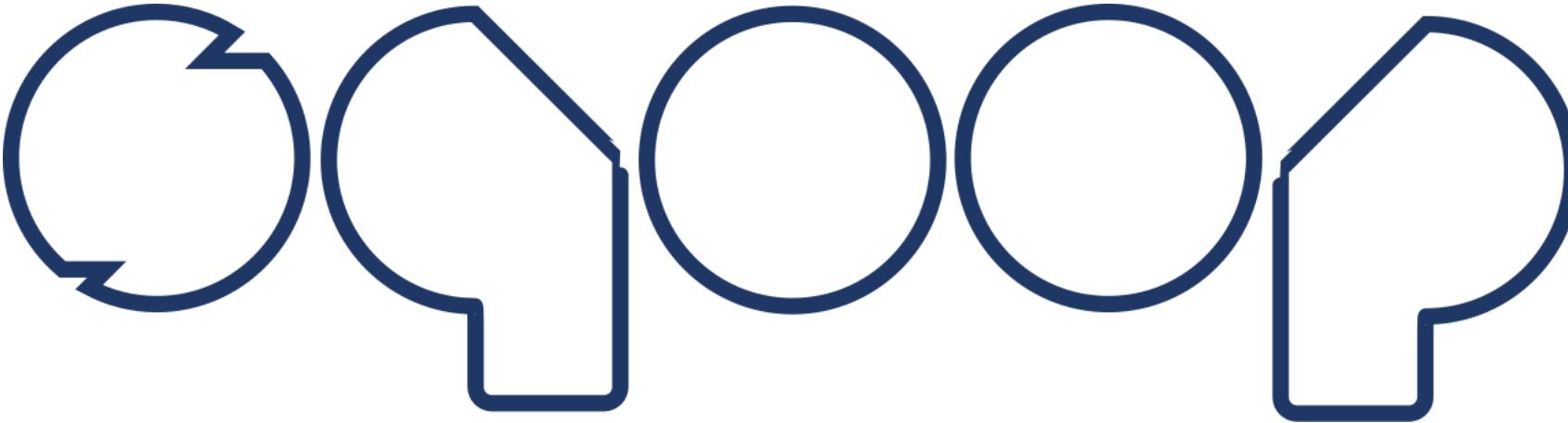
► Visualiser le contenu d'une séquence Flume avec la commande cat :

◆ `/mydata/flumedata/my_apache_logs/FlumeData.xxxxxxx`



```
[hdfs@sandbox ~]$ hdfs dfs -cat /mydata/flumedata/my_apache_logs/FlumeData.1540499257354
SEQ!org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWritable@vG;uNV@f@K@B83.
149.9.216 - - [17/May/2015:10:05:59 +0000] "GET /presentations/logstash-monitorama-2013/images
/logstashbook.png HTTP/1.1" 200 54662 "http://semicomplete.com/presentations/logstash-monitora
ma-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/32.0.1700.77 Safari/537.36" f@K@J83.149.9.216 - - [17/May/2015:10:05:30 +0000] "GET
/presentations/logstash-monitorama-2013/images/github-contributions.png HTTP/1.1" 200 34245 "h
ttp://semicomplete.com/presentations/logstash-monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel
 Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.77 Safari/537.36" f@K@=83.149.9.216 - - [17/May/2015:10:05:53 +0000] "GET /presentations/logstash-monitorama-2013/
css/print/paper.css HTTP/1.1" 200 4254 "http://semicomplete.com/presentations/logstash-monitor
ama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gec
ko) Chrome/32.0.1700.77 Safari/537.36"[hdfs@sandbox ~]$
```

Apache Sqoop



Apache Soop

- ▶ **Soop est un outil conçu pour transférer des données entre Hadoop et des entrepôts de données structurés externes tels que les SGBDR et les Data Warehouses.**
- ▶ **Soop permet un échange de données JDBC entrant et sortant avec Hadoop et son écosystème.**
- ▶ **La source de données fournit le schéma et Soop génère et exécute des instructions SQL à l'aide de JDBC ou d'autres connecteurs.**

Apache Sqoop

- ▶ **Sqoop a la capacité d'import des données depuis des tables ou des bases de données entières dans le HDFS, Hive ou HBase.**
- ▶ **Sqoop a la capacité d'exporter des données depuis des fichiers HDFS, des tables Hive ou des tables HBase.**
- ▶ **Sqoop utilise le framework MapReduce pour importer et exporter les données, ce qui lui permet de paralléliser les traitement et d'être tolérant aux pannes.**

Sqoop agit comme un intermédiaire entre Hadoop et les systèmes de bases de données relationnelles.



Apache Soop

► **Soop est configurable :**

- ◆ Supporte l'utilisation de requêtes personnalisées pour importer ou exporter des données.
- ◆ Supporte l'importation incrémentales de données.

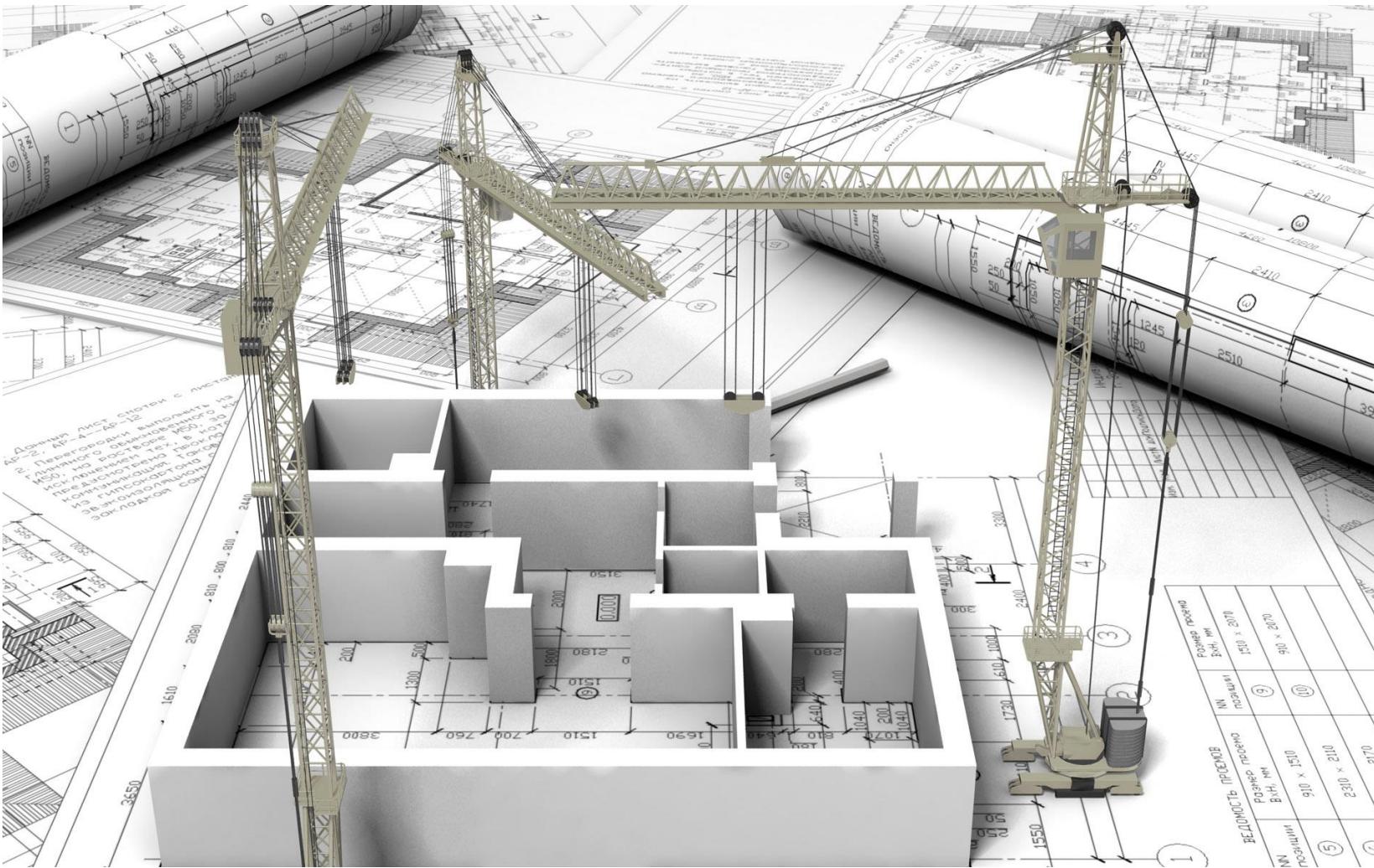
► **Souvent utilisé avec des mises à jour incrémentales à partir de tables.**

- ◆ Le même travail est alors exécuté à plusieurs reprises pour rechercher de nouvelles données.

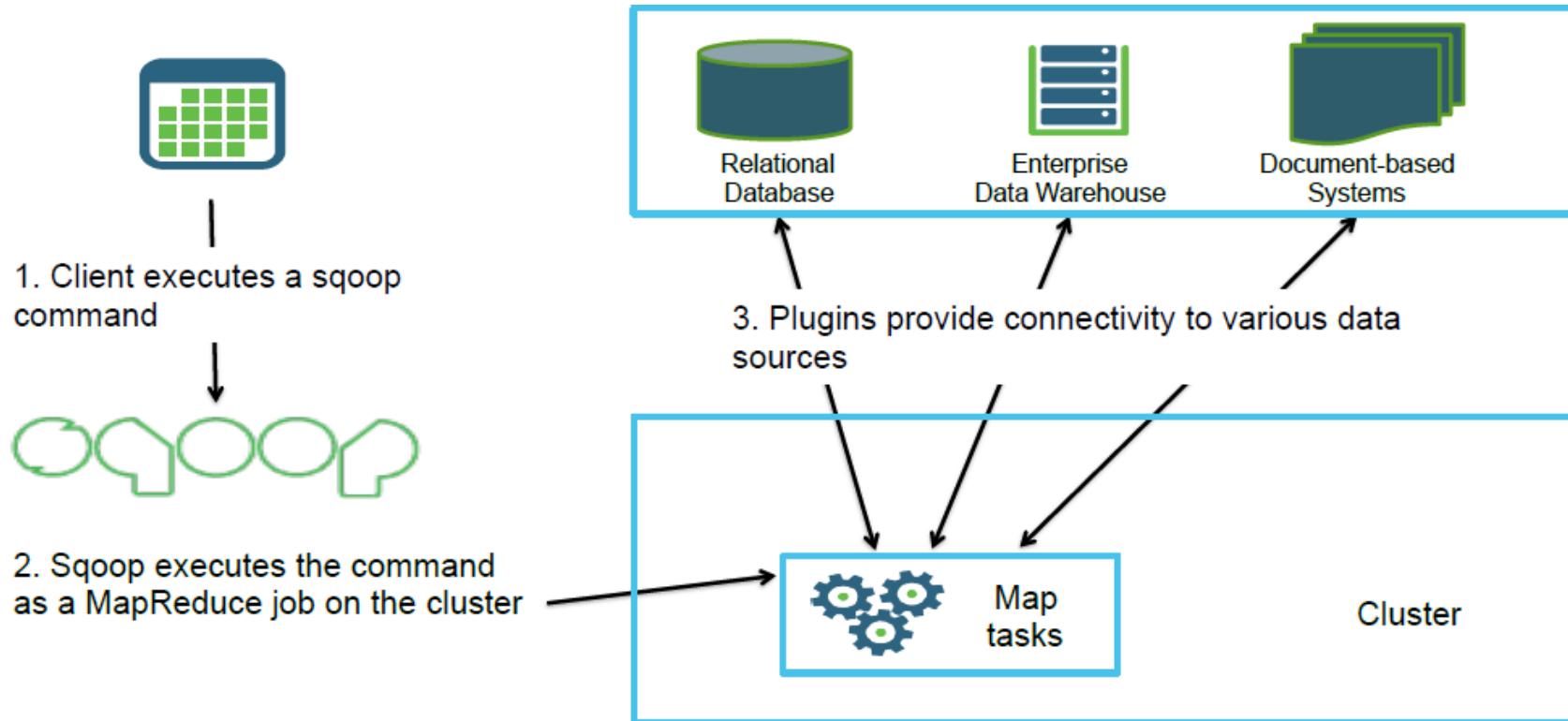
Apache Soop

- ▶ Il existe des connecteurs natifs pour les principaux fournisseurs de bases de données tels que Oracle, Mysql, Postgresql, Teradata, etc.
- ▶ Soop prend en charge des plug-ins fournissant une connectivité à des systèmes externes supplémentaires.

Apache Sqoop - Architecture



Apache Sqoop - Architecture



Apache Soop - Architecture

- ▶ Soop fournit une interface en ligne de commande aux utilisateurs.
- ▶ La commande soumise par l'utilisateur est analysée par Soop et est traduite dans un job MapReduce comportant une tâche Map uniquement pour importer ou exporter des données.
 - ◆ La phase de réduction n'est requise que lorsque des agrégations sont nécessaires.

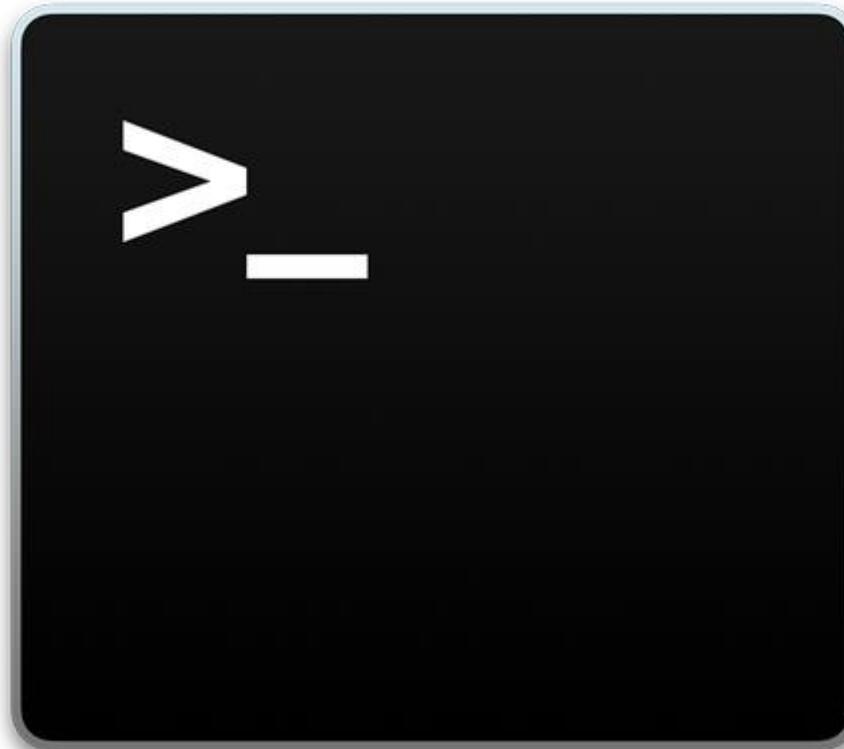
Apache Soop - Architecture

- ▶ La tâche Map lance plusieurs mappeurs en parallèle dépendant du nombre de mappeurs défini par l'utilisateur dans la ligne de commande.
- ▶ Dans le cas d'une importation de données:
 - ◆ chaque mappeur se voit attribuer une partie des données à importer en fonction de la clé définie dans la ligne de commande.

Apache Sqoop - Architecture

- ◆ Sqoop répartit les données d'entrée entre les mappeurs de manière égale pour obtenir des performances élevées.
- ◆ Ensuite, chaque mappeur crée une connexion avec la base de données à l'aide de JDBC et récupère la partie des données affectée par Sqoop et l'écrit dans HDFS, Hive ou HBase en fonction de l'option fournie dans la ligne de commande.

Les commandes Sqoop



Les commandes Sqoop

- ▶ Sqoop fournit une interface en ligne de commande.
- ▶ Dans la commande Sqoop il faut simplement fournir des informations de base telles que :
 - ◆ L'adresse de la source.
 - ◆ Les détails d'authentification de la source.
 - ◆ La destination.
- ▶ Sqoop prendra en charge la partie restante !

La commande import

- ▶ La commande *sqoop import* importe une table d'un SGBDR vers le HDFS.
- ▶ Chaque enregistrement d'une table est considéré comme un enregistrement distinct dans le HDFS.
- ▶ Les enregistrements peuvent être stockés sous forme de fichiers texte ou sous forme d'une représentation binaire tel que les fichiers Avro ou les SequenceFiles.

La commande export

► La syntaxe de la commande **sqoop import** est la suivante :

- ◆ Les arguments generic-args doivent précéder les arguments import-args.
- ◆ Les arguments import-args peuvent être entrés dans n'importe quel ordre.

```
sqoop import (generic-args) (import-args)
```

La commande import

```
sqoop import --connect --username \  
--password --table --target-dir \  
--as-textfile --m
```

- ◆ **--connect** : Prend l'URL JDBC de connexion à la base de données.
- ◆ **--table** : Nom de la table source à importer
- ◆ **--username** : Nom d'utilisateur pour se connecter à la base de données.
- ◆ **--password** : Mot de passe de l'utilisateur qui se connecte à la base de données.
- ◆ **--target-dir** : Importe des données dans le répertoire spécifié.
- ◆ **--as-textfile** : Format du fichier de données, format utilisé par défaut.
- ◆ **--m** : Le nombre de mappeurs à lancer, par défaut 4.

La commande import

► Importer les données sélectionnées d'une table :

```
sqoop import --connect --username \  
--password --table --target-dir \  
--columns --where --as-textfile --m
```

- ◆ --columns : Sélectionne un sous-ensemble de colonnes à importer.
- ◆ --where : Récupère les données qui remplissent la condition.

La commande import

- ▶ Importer des données à partir d'une requête :

```
sqoop import --connect --username \  
--password --target-dir \  
--query --as-textfile --m
```

- ◆ **--query** : Exécute la requête SQL fournie et importe les résultats.

La commande export

- ▶ La commande *sqoop export* exporte un ensemble de fichiers d'un répertoire HDFS vers des tables SGBDR.
- ▶ La table cible devrait déjà exister dans la base de données.
- ▶ La commande *sqoop export* prépare les requêtes INSERT avec un ensemble de données d'entrée, puis les jouent sur la base de données.

La commande export

► La syntaxe de la commande **sqoop export** est la suivante :

sqoop export (generic-args) (export-args)

- ◆ Les arguments generic-args doivent précéder les arguments export-args.
- ◆ Les arguments export-args peuvent être entrés dans n'importe quel ordre.

► Les principaux arguments generic-args sont :

- ◆ --connect : Prend l'URL JDBC de connexion à la base de données.
- ◆ --username : Nom d'utilisateur pour se connecter à la base de données.
- ◆ --password : Mot de passe de l'utilisateur qui se connecte à la base de données.

La commande export

► Les principaux arguments export-args sont :

- ◆ **--columns <"col1, ..., coln">**: La liste des colonnes à exporter vers la table.
- ◆ **--table <nom-table>** : Le nom de la table à peupler.
- ◆ **--export-dir** : Le chemin HDFS des données à exporter.
- ◆ **--m <n>** : Nombre de tâches de mapper à lancer pour exporter les données en parallèle.

La commande export

► Une commande `sqoop export` peut avoir la forme suivante :

```
sqoop export --connect  
--username \  
--password \  
--table \  
--export-dir \  
--m
```

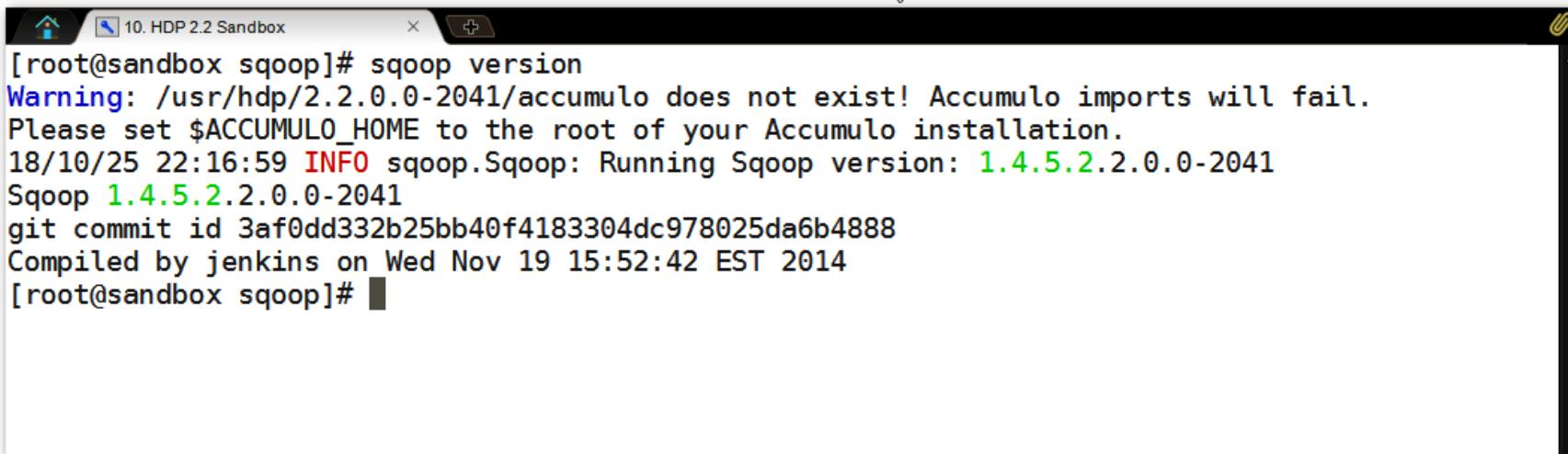


|

Lab !

Lab – Prise en main de Sqoop

► Vérifier la version de Sqoop installée sur le cluster Hadoop en utilisant la commande *sqoop version* :

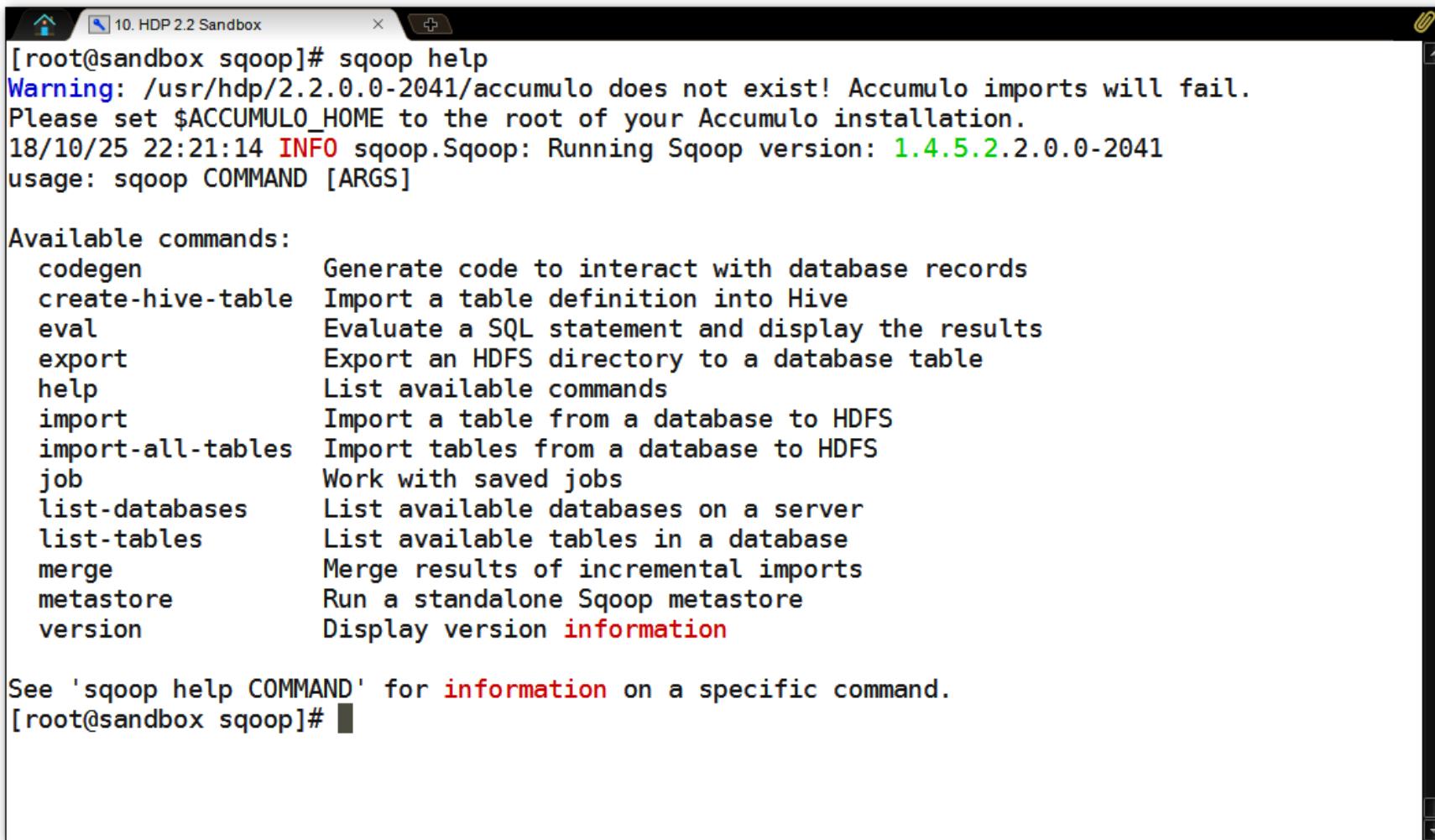


```
[root@sandbox sqoop]# sqoop version
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/25 22:16:59 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
Sqoop 1.4.5.2.2.0.0-2041
git commit id 3af0dd332b25bb40f4183304dc978025da6b4888
Compiled by jenkins on Wed Nov 19 15:52:42 EST 2014
[root@sandbox sqoop]#
```

Lab – Prise en main de Sqoop

- ▶ Utiliser la commande `sqoop help` pour afficher de l'aide par rapport aux commandes supportées par Sqoop.
- ▶ Utiliser la commande `sqoop help <nom de commande>` pour afficher de l'aide par rapport à une commande Sqoop donnée.

Lab – Prise en main de Sqoop

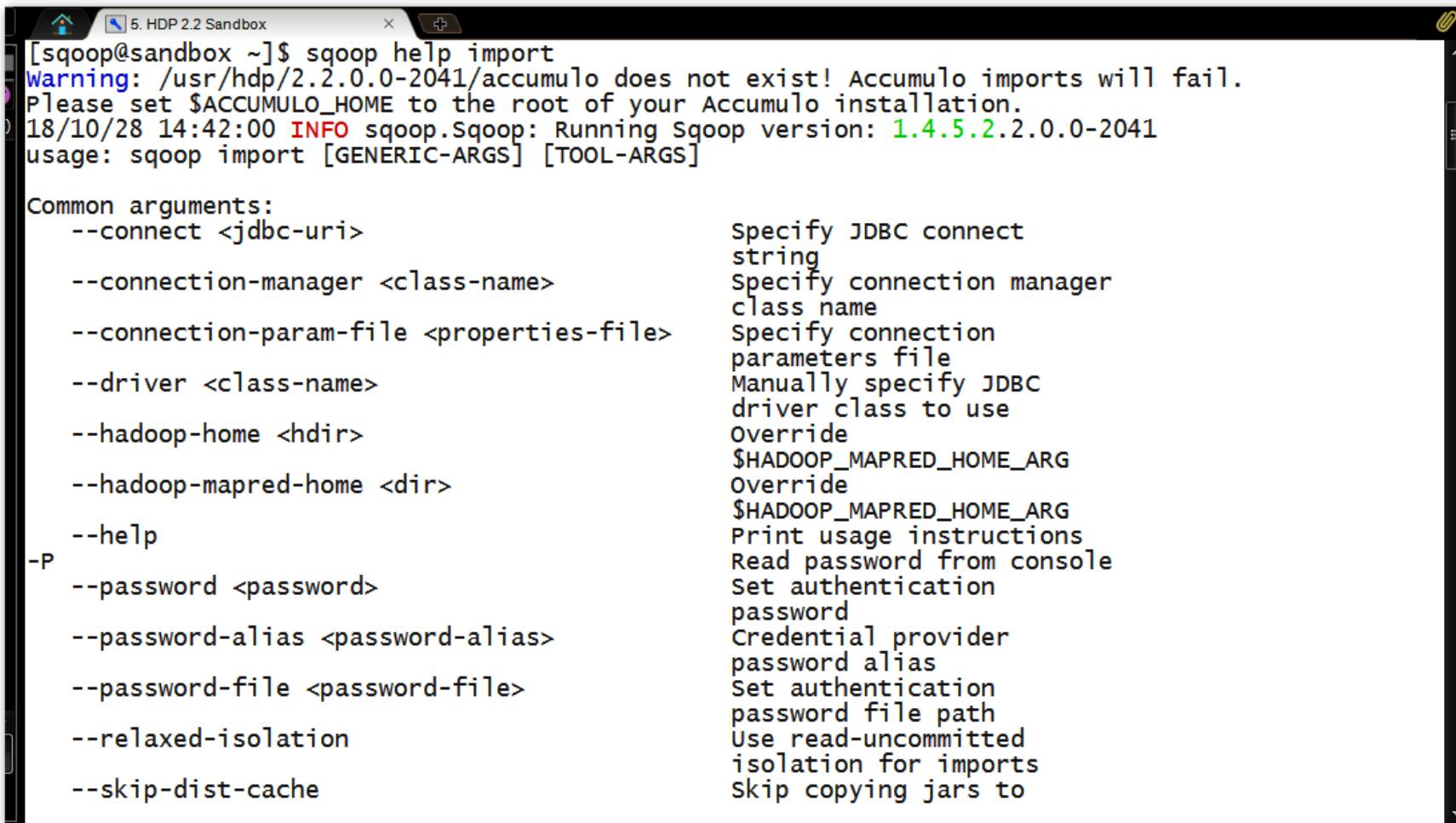
A screenshot of a terminal window titled "10. HDP 2.2 Sandbox". The window contains the output of the command "sqoop help". The output includes a warning about the absence of Accumulo, the version of Sqoop (1.4.5.2.2.0.0-2041), usage instructions, and a detailed list of available commands with their descriptions.

```
[root@sandbox sqoop]# sqoop help
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/25 22:21:14 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
usage: sqoop COMMAND [ARGS]

Available commands:
codegen          Generate code to interact with database records
create-hive-table Import a table definition into Hive
eval             Evaluate a SQL statement and display the results
export            Export an HDFS directory to a database table
help              List available commands
import             Import a table from a database to HDFS
import-all-tables Import tables from a database to HDFS
job               Work with saved jobs
list-databases   List available databases on a server
list-tables      List available tables in a database
merge             Merge results of incremental imports
metastore         Run a standalone Sqoop metastore
version          Display version information

See 'sqoop help COMMAND' for information on a specific command.
[root@sandbox sqoop]#
```

Lab – Prise en main de Sqoop



```
[sqoop@sandbox ~]$ sqoop help import
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/28 14:42:00 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
usage: sqoop import [GENERIC-ARGS] [TOOL-ARGS]

Common arguments:
  --connect <jdbc-uri>                                Specify JDBC connect
                                                       string
  --connection-manager <class-name>                      Specify connection manager
                                                       class name
  --connection-param-file <properties-file>              Specify connection
                                                       parameters file
  --driver <class-name>                                  Manually specify JDBC
                                                       driver class to use
  --hadoop-home <hdir>                                 Override
                                                       $HADOOP_MAPRED_HOME_ARG
  --hadoop-mapred-home <dir>                            Override
                                                       $HADOOP_MAPRED_HOME_ARG
  --help                                                 Print usage instructions
-P                                                     Read password from console
  --password <password>                                Set authentication
                                                       password
  --password-alias <password-alias>                     Credential provider
                                                       password alias
  --password-file <password-file>                      Set authentication
                                                       password file path
  --relaxed-isolation                               Use read-uncommitted
                                                       isolation for imports
  --skip-dist-cache                                Skip copying jars to
```

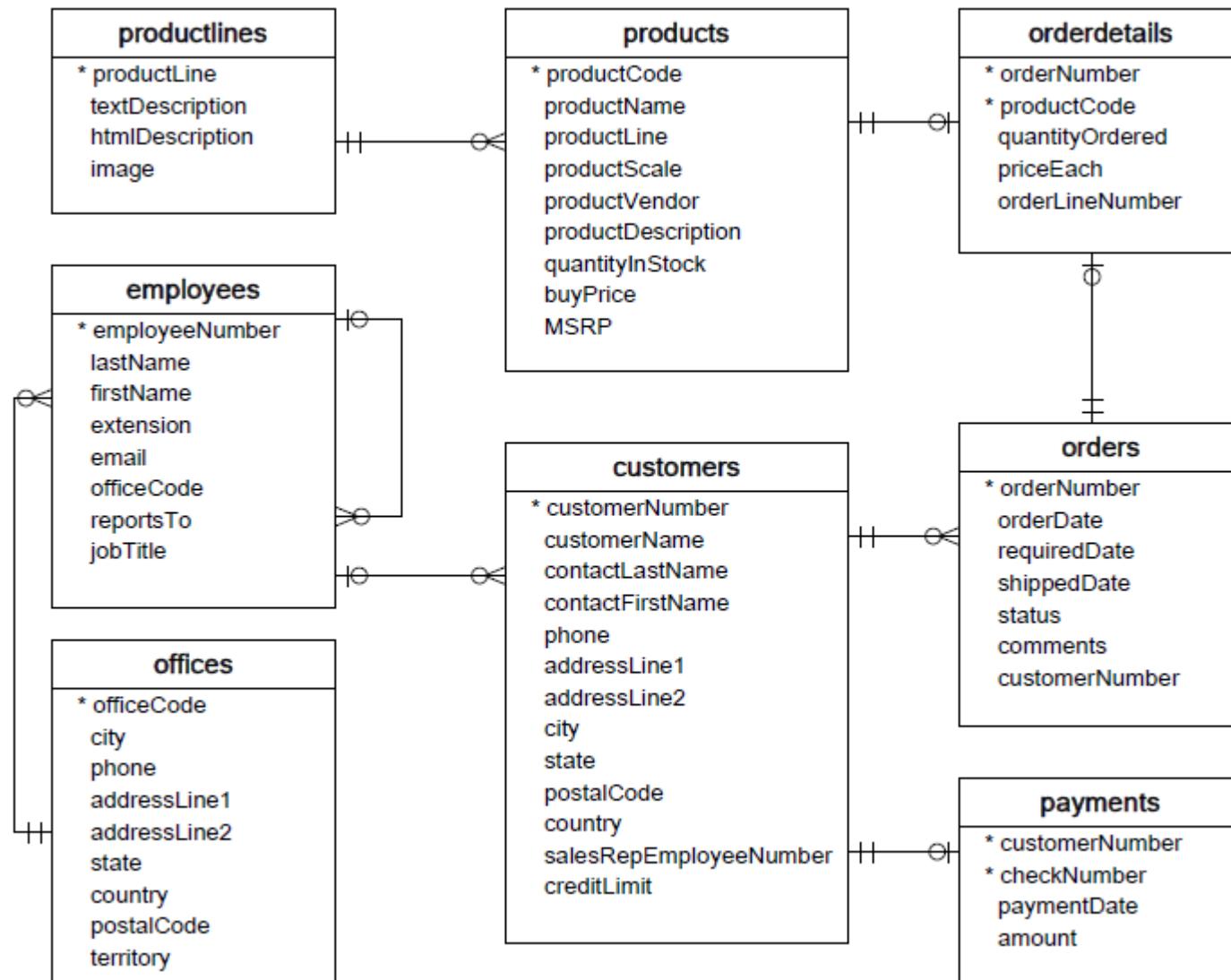
Lab – Prise en main de Sqoop

► Pour plus de détails sur les commandes Sqoop, consulter le guide de référence de Sqoop :

- ◆ <https://sqoop.apache.org/docs/1.4.5/SqoopUserGuide.html>



Lab – Import de tables dans le HDFS

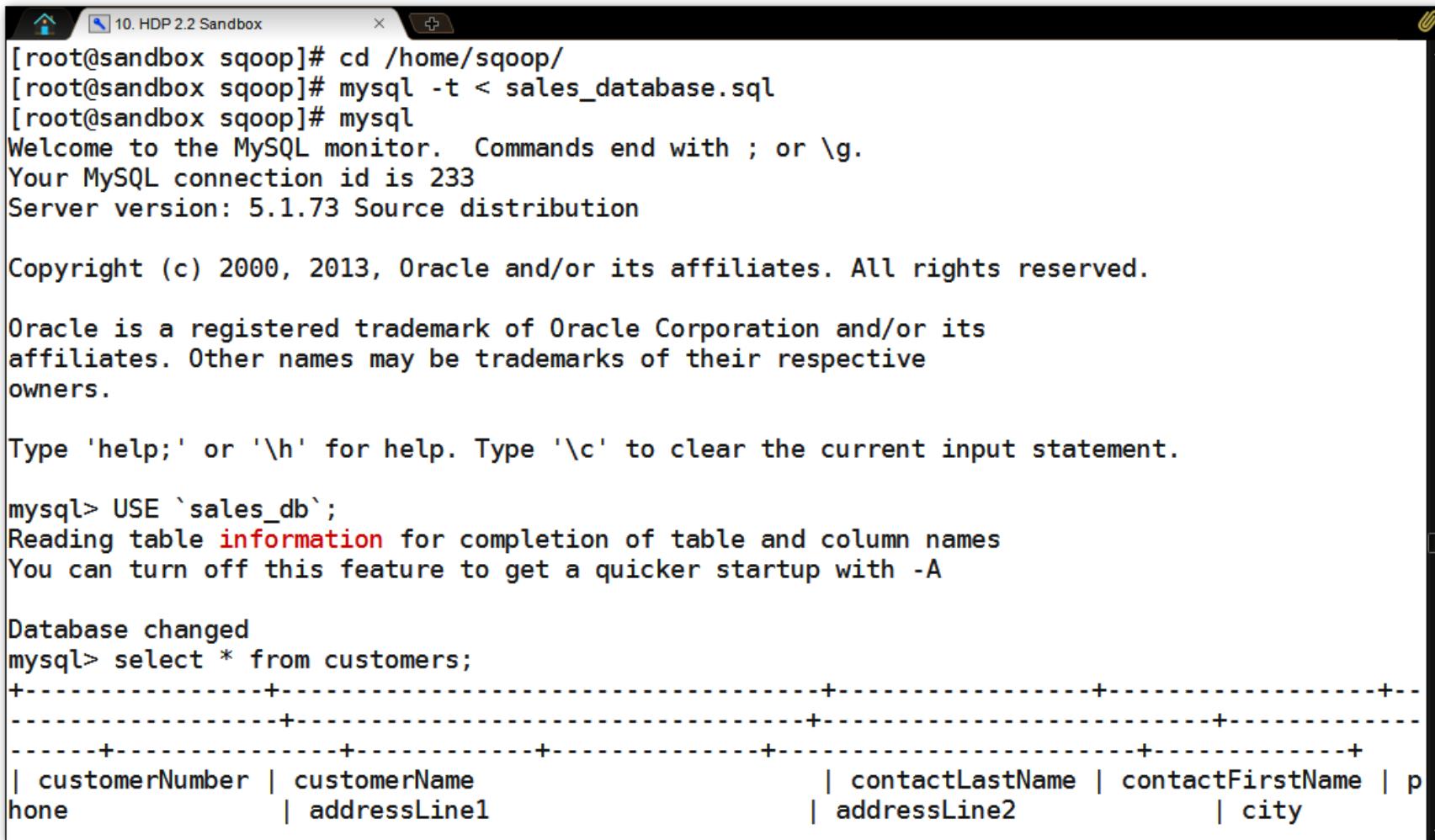


Lab – Import de tables dans le HDFS

- ▶ Créer et charger la base de données sales_db dans le moteur Mysql en utilisant la commande :
 - ◆ Clear
- ▶ Vérifier le bon chargement des données en exécutant les commandes :

```
mysql  
USE classicmodels;  
select * from customers;
```

Lab – Import de tables dans le HDFS



The screenshot shows a terminal window titled "10. HDP 2.2 Sandbox" running on a Linux system. The user is connected to a MySQL database named "sales_db". The MySQL monitor displays standard copyright and trademark information. The user then runs a SQL query to select all columns from the "customers" table. The output shows the column names and their types.

```
[root@sandbox sqoop]# cd /home/sqoop/
[root@sandbox sqoop]# mysql -t < sales_database.sql
[root@sandbox sqoop]# mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 233
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE `sales_db`;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from customers;
+-----+-----+-----+-----+
|-----+-----+-----+-----+
| customerNumber | customerName
hone          | addressLine1
               | contactLastName | contactFirstName | p
               | addressLine2   | city
               |
```

Lab – Sqoop import de table

- ▶ Utiliser la commande suivante pour importer les données de la table customers sous forme de fichiers texte dans le HDFS sous le répertoire :
 - ◆ /mydata/sqoopdata/sales_db/customer_V1

```
sqoop import \
--connect jdbc:mysql://127.0.0.1/ classicmodels?user=root \
--table customers \
--target-dir /mydata/sqoopdata/sales_db/customers_v1
```

Lab – Import de tables dans le HDFS

```
[root@sandbox ~]# sudo su - sqoop
[sqoop@sandbox ~]$ sqoop import \
> --connect jdbc:mysql://127.0.0.1/sales_db?user=root \
> --table customers \
> --target-dir /mydata/sqoopdata/sales_db/customers_v1
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/27 19:55:46 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/27 19:55:46 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/27 19:55:46 INFO tool.CodeGenTool: Beginning code generation
18/10/27 19:55:47 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 19:55:47 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 19:55:47 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-ma
preduce
Note: /tmp/sqoop-sqoop/compile/467ab4a2e336e9de3a955ed4b288b22c/customers.java uses or overrides a d
eprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/27 19:55:52 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/467ab4a2e3
36e9de3a955ed4b288b22c/customers.jar
18/10/27 19:55:52 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/10/27 19:55:52 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/10/27 19:55:52 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/10/27 19:55:52 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/10/27 19:55:52 INFO mapreduce.ImportJobBase: Beginning import of customers
```

Lab – Import de tables dans le HDFS

```
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/zookeeper/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hive/lib/hive-jdbc-0.14.0.2.2.0.0-2041-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
18/10/27 19:55:57 INFO impl.TimelineClientImpl: Timeline service address: http://sandbox.hortonworks.com:8188/ws/v1/timeline/
18/10/27 19:55:57 INFO client.RMProxy: Connecting to ResourceManager at sandbox.hortonworks.com/10.0.2.15:8050
18/10/27 19:55:59 INFO db.DBInputFormat: Using read committed transaction isolation
18/10/27 19:55:59 INFO db.DataDrivenDBInputFormat: BoundingValsQuery: SELECT MIN(`customerNumber`), MAX(`customerNumber`) FROM `customers`
18/10/27 19:56:00 INFO mapreduce.JobSubmitter: number of splits:4
18/10/27 19:56:00 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1540667854836_0003
18/10/27 19:56:01 INFO impl.YarnClientImpl: Submitted application application_1540667854836_0003
18/10/27 19:56:01 INFO mapreduce.Job: The url to track the job: http://sandbox.hortonworks.com:8088/proxy/application_1540667854836_0003/
18/10/27 19:56:01 INFO mapreduce.Job: Running job: job_1540667854836_0003
18/10/27 19:56:14 INFO mapreduce.Job: Job job_1540667854836_0003 running in uber mode : false
18/10/27 19:56:14 INFO mapreduce.Job: map 0% reduce 0%
18/10/27 19:56:47 INFO mapreduce.Job: map 50% reduce 0%
18/10/27 19:56:49 INFO mapreduce.Job: map 75% reduce 0%
18/10/27 19:56:50 INFO mapreduce.Job: map 100% reduce 0%
```

Lab – Import de tables dans le HDFS

```
6. HDP 2.2 Sandbox
18/10/27 19:56:51 INFO mapreduce.Job: Job job_1540667854836_0003 completed successfully
18/10/27 19:56:52 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=496176
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=505
    HDFS: Number of bytes written=14381
    HDFS: Number of read operations=16
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=8
  Job Counters
    Launched map tasks=4
    Other local map tasks=4
    Total time spent by all maps in occupied slots (ms)=121720
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=121720
    Total vcore-seconds taken by all map tasks=121720
    Total megabyte-seconds taken by all map tasks=30430000
  Map-Reduce Framework
    Map input records=122
    Map output records=122
    Input split bytes=505
    Spilled Records=0
```

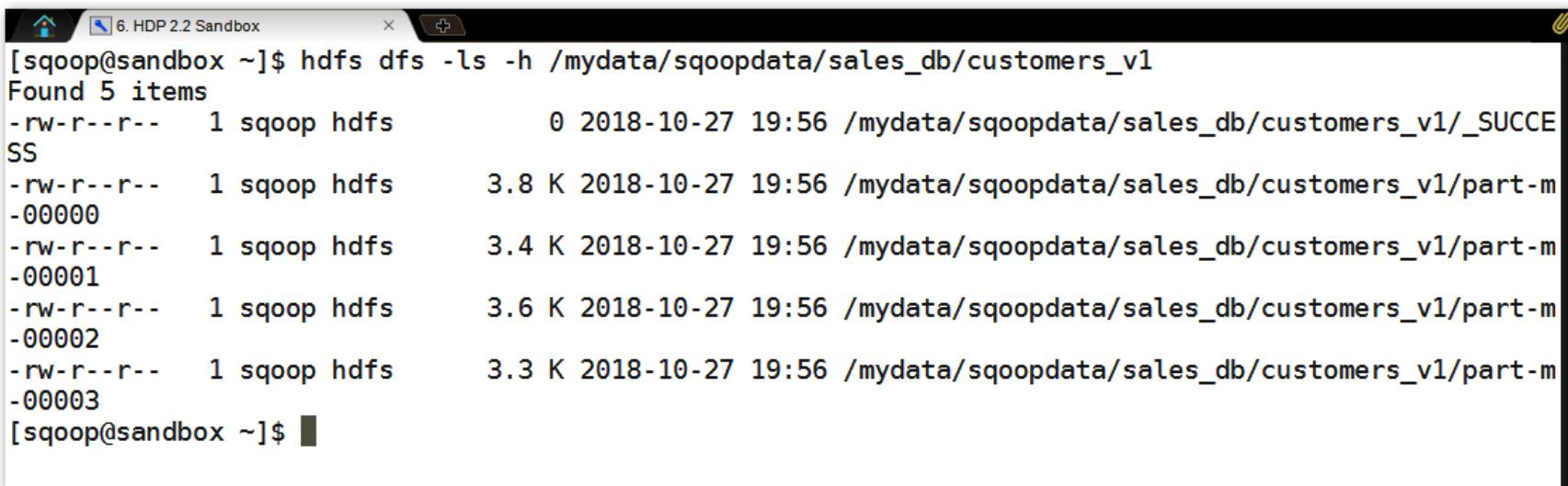
Lab – Import de tables dans le HDFS

```
6. HDP 2.2 Sandbox
Other local map tasks=4
Total time spent by all maps in occupied slots (ms)=121720
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=121720
Total vcore-seconds taken by all map tasks=121720
Total megabyte-seconds taken by all map tasks=30430000
Map-Reduce Framework
  Map input records=122
  Map output records=122
  Input split bytes=505
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=484
  CPU time spent (ms)=5870
  Physical memory (bytes) snapshot=455380992
  Virtual memory (bytes) snapshot=3133325312
  Total committed heap usage (bytes)=227016704
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=14381
18/10/27 19:56:52 INFO mapreduce.ImportJobBase: Transferred 14.0439 KB in 56.7522 seconds (253.3998
bytes/sec)
18/10/27 19:56:52 INFO mapreduce.ImportJobBase: Retrieved 122 records.
[sqoop@sandbox ~]$
```

Lab – Import de tables dans le HDFS

► Vérifier le résultat de l'import de la table `customers` avec les commandes :

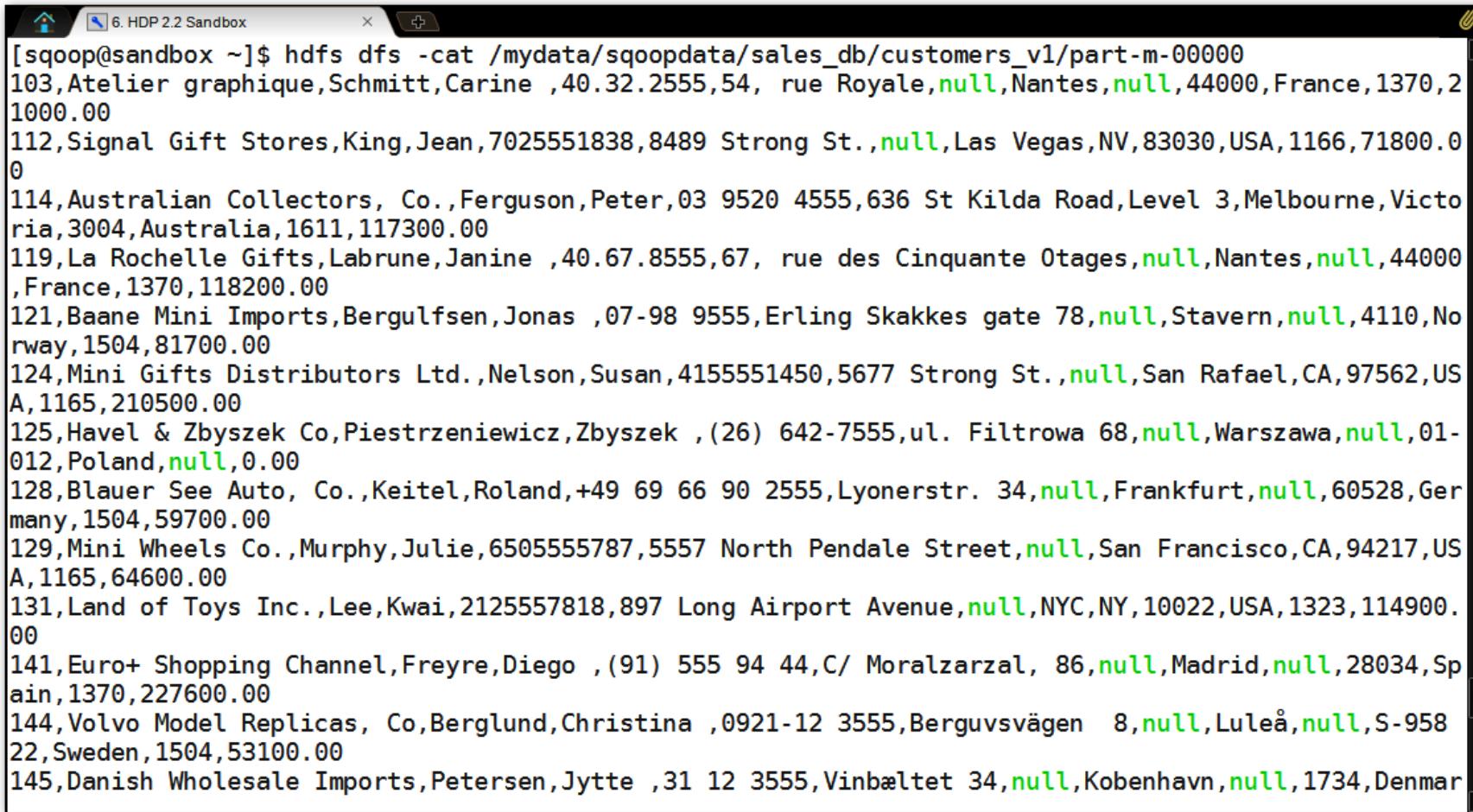
- ◆ `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v1`
- ◆ `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v1/part-m-00000`



The screenshot shows a terminal window titled "6. HDP 2.2 Sandbox". The command entered is `hdfs dfs -ls -h /mydata/sqoopdata/sales_db/customers_v1`. The output shows the following directory structure:

```
[sqoop@sandbox ~]$ hdfs dfs -ls -h /mydata/sqoopdata/sales_db/customers_v1
Found 5 items
-rw-r--r-- 1 sqoop hdfs          0 2018-10-27 19:56 /mydata/sqoopdata/sales_db/customers_v1/_SUCCESS
-rw-r--r-- 1 sqoop hdfs      3.8 K 2018-10-27 19:56 /mydata/sqoopdata/sales_db/customers_v1/part-m-00000
-rw-r--r-- 1 sqoop hdfs      3.4 K 2018-10-27 19:56 /mydata/sqoopdata/sales_db/customers_v1/part-m-00001
-rw-r--r-- 1 sqoop hdfs      3.6 K 2018-10-27 19:56 /mydata/sqoopdata/sales_db/customers_v1/part-m-00002
-rw-r--r-- 1 sqoop hdfs      3.3 K 2018-10-27 19:56 /mydata/sqoopdata/sales_db/customers_v1/part-m-00003
[sqoop@sandbox ~]$
```

Lab – Import de tables dans le HDFS



The screenshot shows a terminal window titled "6. HDP 2.2 Sandbox" displaying the output of an HDFS command. The command is "hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v1/part-m-00000". The output lists 145 rows of customer data from a file named "customers_v1". Each row contains a unique ID, company name, contact person, address, and various coordinates and location details. The data includes entries from countries like France, USA, Australia, Norway, Poland, Germany, and Denmark.

```
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v1/part-m-00000
103,Atelier graphique,Schmitt,Carine ,40.32.2555,54, rue Royale,null,Nantes,null,44000,France,1370,2
1000.00
112,Signal Gift Stores,King,Jean,7025551838,8489 Strong St.,null,Las Vegas,NV,83030,USA,1166,71800.0
0
114,Australian Collectors, Co.,Ferguson,Peter,03 9520 4555,636 St Kilda Road,Level 3,Melbourne,Victo
ria,3004,Australia,1611,117300.00
119,La Rochelle Gifts,Labrune,Janine ,40.67.8555,67, rue des Cinquante Otages,null,Nantes,null,44000
,France,1370,118200.00
121,Baane Mini Imports,Bergulfsen,Jonas ,07-98 9555,Erling Skakkes gate 78,null,Stavern,null,4110,No
rway,1504,81700.00
124,Mini Gifts Distributors Ltd.,Nelson,Susan,4155551450,5677 Strong St.,null,San Rafael,CA,97562,US
A,1165,210500.00
125,Havel & Zbyszek Co,Piestrzeniewicz,Zbyszek ,(26) 642-7555,ul. Filtrowa 68,null,Warszawa,null,01-
012,Poland,null,0.00
128,Blauer See Auto, Co.,Keitel,Roland,+49 69 66 90 2555,Lyonerstr. 34,null,Frankfurt,null,60528,Germany,1504,59700.00
129,Mini Wheels Co.,Murphy,Julie,6505555787,5557 North Pendale Street,null,San Francisco,CA,94217,US
A,1165,64600.00
131,Land of Toys Inc.,Lee,Kwai,2125557818,897 Long Airport Avenue,null,NYC,NY,10022,USA,1323,114900.
00
141,Euro+ Shopping Channel,Freyre,Diego ,(91) 555 94 44,C/ Moralzarjal, 86,null,Madrid,null,28034,Spain,1370,227600.00
144,Volvo Model Replicas, Co,Berglund,Christina ,0921-12 3555,Berguvsvägen 8,null,Luleå,null,S-958
22,Sweden,1504,53100.00
145,Danish Wholesale Imports,Petersen,Jytte ,31 12 3555,Vinbältet 34,null,Kopenhagen,null,1734,Denmar
```

Lab – Sqoop import de table

► Utiliser la commande suivante pour importer les données de la table customers sous forme d'un seul fichier texte dans le HDFS sous le répertoire :

- ◆ /mydata/sqoopdata/sales_db/customer_v2

```
sqoop import \
--connect jdbc:mysql://127.0.0.1/ classicmodels?user=root \
--table customers \
--target-dir /mydata/sqoopdata/sales_db/customers_v2 \
--m 1
```

Lab – Sqoop import de table

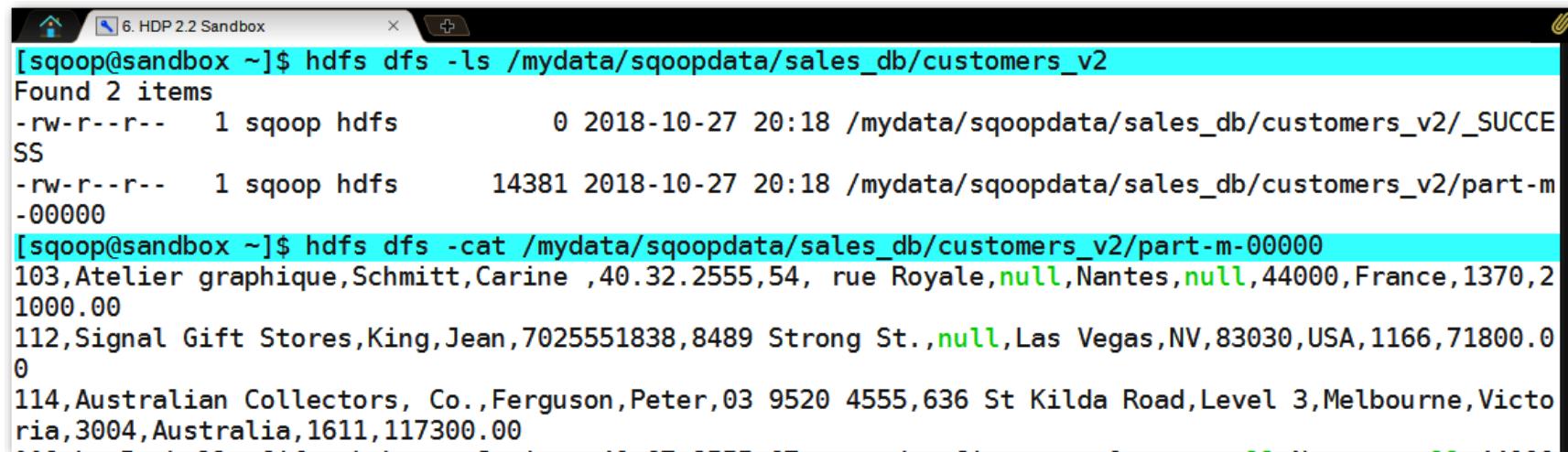
```
[sqoop@sandbox ~]$ sqoop import \
> --connect jdbc:mysql://127.0.0.1/sales_db?user=root \
> --table customers \
> --target-dir /mydata/sqoopdata/sales_db/customers_v2 \
> --m 1
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/27 20:17:41 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/27 20:17:42 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/27 20:17:42 INFO tool.CodeGenTool: Beginning code generation
18/10/27 20:17:43 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 20:17:43 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 20:17:43 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-ma
preduce
Note: /tmp/sqoop-sqoop/compile/f265551e232ccb9a28c19be0c1dcf476/customers.java uses or overrides a d
eprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/27 20:17:50 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/f265551e23
2ccb9a28c19be0c1dcf476/customers.jar
18/10/27 20:17:50 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/10/27 20:17:50 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/10/27 20:17:50 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/10/27 20:17:50 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/10/27 20:17:50 INFO mapreduce.ImportJobBase: Beginning import of customers
```

Lab – Sqoop import de table

```
6. HDP 2.2 Sandbox
Other local map tasks=1
Total time spent by all maps in occupied slots (ms)=18158
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=18158
Total vcore-seconds taken by all map tasks=18158
Total megabyte-seconds taken by all map tasks=4539500
Map-Reduce Framework
  Map input records=122
  Map output records=122
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=133
  CPU time spent (ms)=1400
  Physical memory (bytes) snapshot=115724288
  Virtual memory (bytes) snapshot=774479872
  Total committed heap usage (bytes)=62390272
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=14381
18/10/27 20:18:37 INFO mapreduce.ImportJobBase: Transferred 14.0439 KB in 44.1542 seconds (325.6994
bytes/sec)
18/10/27 20:18:37 INFO mapreduce.ImportJobBase: Retrieved 122 records.
[sqoop@sandbox ~]$
```

Lab – Import de tables dans le HDFS

- ▶ Vérifier le résultat de l'import de la table customers avec les commandes :
 - ◆ `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v2`
 - ◆ `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v2/part-m-00000`



```
[sqoop@sandbox ~]$ hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v2
Found 2 items
-rw-r--r-- 1 sqoop hdfs          0 2018-10-27 20:18 /mydata/sqoopdata/sales_db/customers_v2/_SUCCESS
-rw-r--r-- 1 sqoop hdfs    14381 2018-10-27 20:18 /mydata/sqoopdata/sales_db/customers_v2/part-m-00000
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v2/part-m-00000
103,Atelier graphique,Schmitt,Carine ,40.32.2555,54, rue Royale,null,Nantes,null,44000,France,1370,2
1000.00
112,Signal Gift Stores,King,Jean,7025551838,8489 Strong St.,null,Las Vegas,NV,83030,USA,1166,71800.0
0
114,Australian Collectors, Co.,Ferguson,Peter,03 9520 4555,636 St Kilda Road,Level 3,Melbourne,Victo
ria,3004,Australia,1611,117300.00
```

Lab – Sqoop import de table

- ▶ Utiliser la commande suivante pour importer les données des colonnes `customerNumber`, `customerName`, `contactLastName`, `city` de la table `customers` pour les lignes vérifiant la condition `country='USA'` sous forme d'un seul fichier texte dans le HDFS sous le répertoire :
 - ◆ `/mydata/sqoopdata/sales_db/customer_v3`

Lab – Sqoop import de table

```
sqoop import \
--connect jdbc:mysql://127.0.0.1/ classicmodels?user=root \
--table customers \
--target-dir /mydata/sqoopdata/sales_db/customers_v3 \
--columns "customerNumber, customerName, contactLastName, city" \
--where "country='USA'" \
--m 1
```

Lab – Sqoop import de table

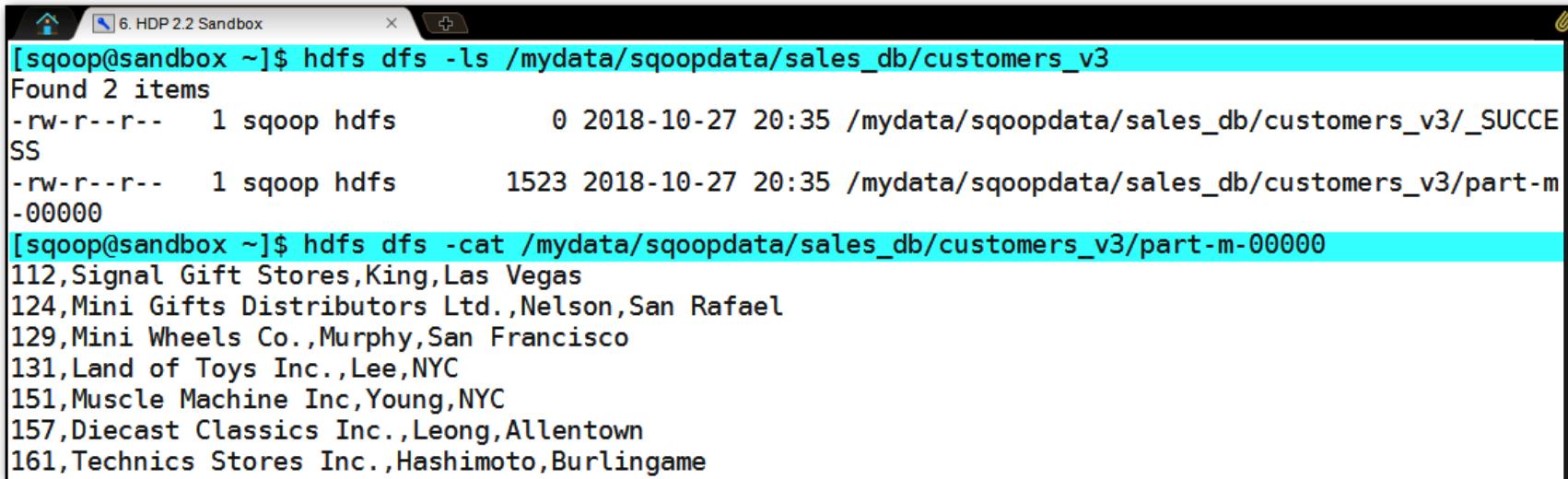
```
[sqoop@sandbox ~]$ sqoop import \
> --connect jdbc:mysql://127.0.0.1/sales_db?user=root \
> --table customers \
> --target-dir /mydata/sqoopdata/sales_db/customers_v3 \
> --columns "customerNumber, customerName, contactLastName, city" \
> --where "country='USA'" \
> --m 1
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/27 20:34:51 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/27 20:34:51 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/27 20:34:51 INFO tool.CodeGenTool: Beginning code generation
18/10/27 20:34:52 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 20:34:52 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t
 LIMIT 1
18/10/27 20:34:52 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-ma
preduce
Note: /tmp/sqoop-sqoop/compile/5ac1a2ff7f10b4fed6b4057d030a6cfb/customers.java uses or overrides a d
eprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/27 20:34:56 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/5ac1a2ff7f
10b4fed6b4057d030a6cfb/customers.jar
18/10/27 20:34:56 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/10/27 20:34:56 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/10/27 20:34:56 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
```

Lab – Sqoop import de table

```
6. HDP 2.2 Sandbox
Other local map tasks=1
Total time spent by all maps in occupied slots (ms)=14270
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=14270
Total vcore-seconds taken by all map tasks=14270
Total megabyte-seconds taken by all map tasks=3567500
Map-Reduce Framework
  Map input records=36
  Map output records=36
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=62
  CPU time spent (ms)=1280
  Physical memory (bytes) snapshot=110923776
  Virtual memory (bytes) snapshot=782016512
  Total committed heap usage (bytes)=63438848
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=1523
18/10/27 20:35:37 INFO mapreduce.ImportJobBase: Transferred 1.4873 KB in 39.1228 seconds (38.9287 bytes/sec)
18/10/27 20:35:37 INFO mapreduce.ImportJobBase: Retrieved 36 records.
[sqoop@sandbox ~]$
```

Lab – Import de tables dans le HDFS

- Vérifier le résultat de l'import de la table `customers` avec les commandes :
- ◆ `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v3`
 - ◆ `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v3/part-m-00000`



The screenshot shows a terminal window titled "6. HDP 2.2 Sandbox". It displays two commands run by the user "sqoop@sandbox ~". The first command, `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v3`, lists two items: a success file named "_SUCCESS" and a part file named "part-m-00000" containing 1523 records. The second command, `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v3/part-m-00000`, outputs the contents of the part file, which are the names and addresses of various companies.

```
[sqoop@sandbox ~]$ hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v3
Found 2 items
-rw-r--r--  1 sqoop hdfs          0 2018-10-27 20:35 /mydata/sqoopdata/sales_db/customers_v3/_SUCCESS
-rw-r--r--  1 sqoop hdfs    1523 2018-10-27 20:35 /mydata/sqoopdata/sales_db/customers_v3/part-m-00000
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v3/part-m-00000
112,Signal Gift Stores,King,Las Vegas
124,Mini Gifts Distributors Ltd.,Nelson,San Rafael
129,Mini Wheels Co.,Murphy,San Francisco
131,Land of Toys Inc.,Lee,NYC
151,Muscle Machine Inc,Young,NYC
157,Diecast Classics Inc.,Leong,Allentown
161,Technics Stores Inc.,Hashimoto,Burlingame
```

Lab – Ssqoop import de table

- ▶ Utiliser la commande suivante pour importer le résultat de la requête SQL suivante sous forme d'un seul fichier texte dans le HDFS sous le répertoire :
 - ◆ `/mydata/sqoopdata/sales_db/customer_v4`

Lab – Sqoop import de table

```
sqoop import \
--connect jdbc:mysql://127.0.0.1/ classicmodels?user=root \
--target-dir /mydata/sqoopdata/sales_db/customers_v4 \
--query "select customerNumber, customerName, contactLastName, city from
customers where country='USA' and \$CONDITIONS" \
--m 1
```

Lab – Sqoop import de table

```
[sqoop@sandbox ~]$ sqoop import \
> --connect jdbc:mysql://127.0.0.1/sales_db?user=root \
> --target-dir /mydata/sqoopdata/sales_db/customers_v4 \
> --query "select customerNumber, customerName, contactLastName, city from customers where country='USA' and \$CONDITIONS" \
> --m 1
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/27 20:52:20 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/27 20:52:21 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/27 20:52:21 INFO tool.CodeGenTool: Beginning code generation
18/10/27 20:52:22 INFO manager.SqlManager: Executing SQL statement: select customerNumber, customerName, contactLastName, city from customers where country='USA' and (1 = 0)
18/10/27 20:52:22 INFO manager.SqlManager: Executing SQL statement: select customerNumber, customerName, contactLastName, city from customers where country='USA' and (1 = 0)
18/10/27 20:52:22 INFO manager.SqlManager: Executing SQL statement: select customerNumber, customerName, contactLastName, city from customers where country='USA' and (1 = 0)
18/10/27 20:52:22 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-mapreduce
Note: /tmp/sqoop-sqoop/compile/8ade4c5f1601741166b25a99c2eac876/QueryResult.java uses or overrides a
      deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/27 20:52:27 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/8ade4c5f1601741166b25a99c2eac876/QueryResult.jar
18/10/27 20:52:27 INFO mapreduce.ImportJobBase: Beginning query import.
SLF4J: Class path contains multiple SLF4J bindings.
```

Lab – Sqoop import de table

```
6. HDP 2.2 Sandbox
Other local map tasks=1
Total time spent by all maps in occupied slots (ms)=9933
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=9933
Total vcore-seconds taken by all map tasks=9933
Total megabyte-seconds taken by all map tasks=2483250
Map-Reduce Framework
  Map input records=36
  Map output records=36
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=69
  CPU time spent (ms)=1270
  Physical memory (bytes) snapshot=127631360
  Virtual memory (bytes) snapshot=781234176
  Total committed heap usage (bytes)=61865984
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=1523
18/10/27 20:53:10 INFO mapreduce.ImportJobBase: Transferred 1.4873 KB in 38.9766 seconds (39.0747 bytes/sec)
18/10/27 20:53:10 INFO mapreduce.ImportJobBase: Retrieved 36 records.
[sqoop@sandbox ~]$
```

Lab – Import de tables dans le HDFS

► Vérifier le résultat de l'import de la requête avec les commandes :

- ◆ `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v4`
- ◆ `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v4/part-m-00000`

```
[sqoop@sandbox ~]$ hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v4
Found 2 items
-rw-r--r-- 1 sqoop hdfs          0 2018-10-27 20:53 /mydata/sqoopdata/sales_db/customers_v4/_SUCCESS
-rw-r--r-- 1 sqoop hdfs    1523 2018-10-27 20:53 /mydata/sqoopdata/sales_db/customers_v4/part-m-00000
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v4/part-m-00000
112,Signal Gift Stores,King,Las Vegas
124,Mini Gifts Distributors Ltd.,Nelson,San Rafael
129,Mini Wheels Co.,Murphy,San Francisco
131,Land of Toys Inc.,Lee,NYC
151,Muscle Machine Inc,Young,NYC
157,Diecast Classics Inc.,Leong,Allentown
161,Technics Stores Inc.,Hashimoto,Burlingame
168,American Souvenirs Inc,Franco,New Haven
```

Lab – Sqoop import de table

► Utiliser la commande suivante pour importer les données de la table customers sous forme d'un fichier texte dans le HDFS en utilisant le caractère "|" comme séparateur de colonne :

```
sqoop import \
--connect jdbc:mysql://127.0.0.1/ classicmodels?user=root \
--table customers \
--fields-terminated-by "|" \
--target-dir /mydata/sqoopdata/sales_db/customers_v5 \
--m 1
```

Lab – Import de tables dans le HDFS

```
[sqoop@sandbox ~]$ sqoop import \
> --connect jdbc:mysql://127.0.0.1/sales_db?user=root \
> --table customers \
> --fields-terminated-by "|" \
> --target-dir /mydata/sqoopdata/sales_db/customers_v5 \
> --m 1
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/28 18:47:04 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/28 18:47:05 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/28 18:47:05 INFO tool.CodeGenTool: Beginning code generation
18/10/28 18:47:06 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t LIMIT 1
18/10/28 18:47:06 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t LIMIT 1
18/10/28 18:47:06 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-mapreduce
Note: /tmp/sqoop-sqoop/compile/6cadfe929c3f94a11f05a7045290ea53/customers.java uses or overrides
a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/28 18:47:11 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/6cadfe929c3f94a11f05a7045290ea53/customers.jar
18/10/28 18:47:11 WARN manager.MySQLManager: It looks like you are importing from mysql.
18/10/28 18:47:11 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
18/10/28 18:47:11 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
18/10/28 18:47:11 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
18/10/28 18:47:11 INFO mapreduce.ImportJobBase: Beginning import of customers
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar!/org/
```

Lab – Import de tables dans le HDFS

► Vérifier le résultat de l'import de la requête avec les commandes :

- ◆ `hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v5`
- ◆ `hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v5/part-m-00000`

Lab – Import de tables dans le HDFS

```
[sqoop@sandbox ~]$ hdfs dfs -ls /mydata/sqoopdata/sales_db/customers_v5
Found 2 items
-rw-r--r-- 1 sqoop hdfs          0 2018-10-28 18:47 /mydata/sqoopdata/sales_db/customers_v5/_s
UCCESS
-rw-r--r-- 1 sqoop hdfs    14381 2018-10-28 18:47 /mydata/sqoopdata/sales_db/customers_v5/part-m-00000
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_db/customers_v5/part-m-00000
103|Atelier graphique|Schmitt|Carine |40.32.2555|54, rue Royale|null|Nantes|null|44000|France|13
70|21000.00
112|Signal Gift Stores|King|Jean|7025551838|8489 Strong St.|null|Las Vegas|NV|83030|USA|1166|718
00.00
114|Australian Collectors, Co.|Ferguson|Peter|03 9520 4555|636 st Kilda Road|Level 3|Melbourne|V
ictoria|3004|Australia|1611|117300.00
119|La Rochelle Gifts|Labrunie|Janine |40.67.8555|67, rue des Cinquante Otages|null|Nantes|null|4
4000|France|1370|118200.00
121|Baane Mini Imports|Bergulfson|Jonas |07-98 9555|Erling Skakkes gate 78|null|Stavern|null|411
0|Norway|1504|81700.00
124|Mini Gifts Distributors Ltd.|Nelson|Susan|4155551450|5677 Strong St.|null|San Rafael|CA|9756
2|USA|1165|210500.00
125|Havel & Zbyszek Co|Piestrzewicz|Zbyszek |(26) 642-7555|ul. Filtrowa 68|null|Warszawa|null
|01-012|Poland|null|0.00
128|Blauer See Auto, Co.|Keitel|Roland|+49 69 66 90 2555|Lyonerstr. 34|null|Frankfurt|null|60528
|Germany|1504|59700.00
129|Mini Wheels Co.|Murphy|Julie|6505555787|5557 North Pendale Street|null|San Francisco|CA|9421
7|USA|1165|64600.00
131|Land of Toys Inc.|Lee|Kwai|2125557818|897 Long Airport Avenue|null|NYC|NY|10022|USA|1323|114
900.00
141|Euro+ Shopping Channel|Freyre|Diego |(91) 555 94 44|C/ Moralzarjal, 86|null|Madrid|null|2803
4|Spain|1370|227600.00
144|Volvo Model Replicas, Co|Berglund|Christina |0921-12 3555|Berguvsvagen 8|null|Luleå|null|s-
```

Lab – Export de fichiers dans une table

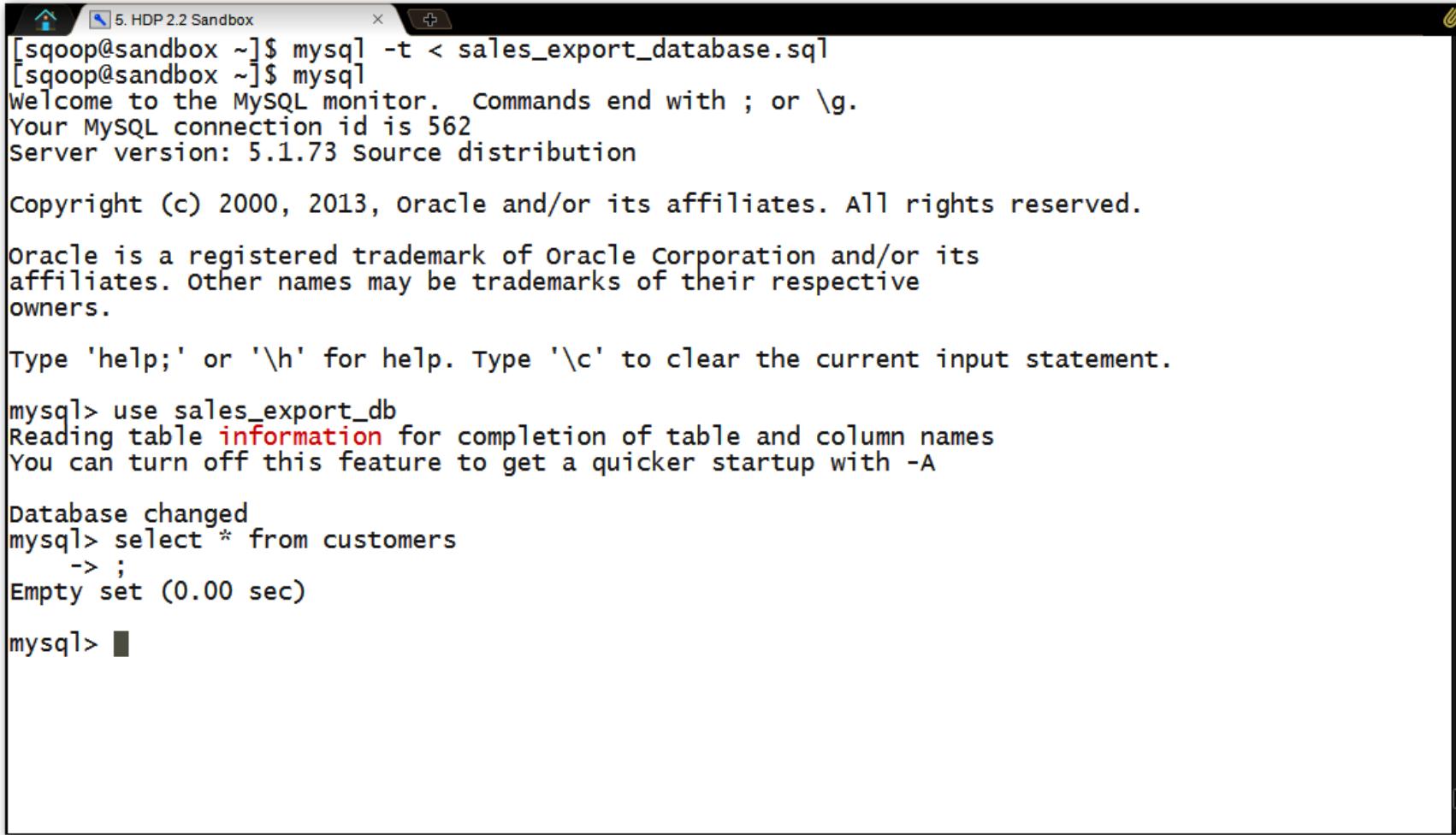
►Créer la base de données sales_export_db dans le moteur Mysql en utilisant la commande :

- ◆mysql -t < sales_export_database.sql

►Vérifier la création de la base de données en exécutant les commandes :

```
mysql  
USE sales_export_db;  
select * from customers;
```

Lab – Export de fichiers dans une table



```
[sqoop@sandbox ~]$ mysql -t < sales_export_database.sql
[sqoop@sandbox ~]$ mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 562
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sales_export_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from customers
-> ;
Empty set (0.00 sec)

mysql> █
```

Lab – Sqoop export de table

► Utiliser la commande suivante pour exporter des données depuis le HDFS vers la table customers de la base de données sales_export_db :

```
sqoop export \
--connect jdbc:mysql://127.0.0.1/sales_export_db?user=root \
--table customers \
--input-fields-terminated-by "|" \
--export-dir \
/mydata/sqoopdata/sales_db/customers_v5/ part-m-00000
```

Lab – Export de fichiers dans une table

```
[sqoop@sandbox ~]$ hdfs dfs -cat /mydata/sqoopdata/sales_export/customers_v1/customers.csv  
103|Atelier graphique|Schmitt|Carine |40.32.2555|54, rue Royale|null|Nantes|null|44000|France|13  
70|21000.00  
112|Signal Gift Stores|King|Jean|7025551838|8489 Strong St.|null|Las Vegas|NV|83030|USA|1166|718  
00.00  
114|Australian Collectors, Co.|Ferguson|Peter|03 9520 4555|636 st Kilda Road|Level 3|Melbourne|V  
ictoria|3004|Australia|1611|117300.00  
119|La Rochelle Gifts|Labrune|Janine |40.67.8555|67, rue des Cinquante Otages|null|Nantes|null|4  
4000|France|1370|118200.00  
121|Baane Mini Imports|Bergulfsen|Jonas |07-98 9555|Erling skakkes gate 78|null|Stavern|null|411  
0|Norway|1504|81700.00  
124|Mini Gifts Distributors Ltd.|Nelson|Susan|4155551450|5677 Strong St.|null|San Rafael|CA|9756  
2|USA|1165|210500.00  
125|Havel & Zbyszek Co|Piestrzewicz|Zbyszek |(26) 642-7555|ul. Filtrowa 68|null|Warszawa|null  
|01-012|Poland|null|0.00  
128|Blauer See Auto, Co.|Keitel|Roland|+49 69 66 90 2555|Lyonerstr. 34|null|Frankfurt|null|60528  
|Germany|1504|59700.00  
129|Mini Wheels Co.|Murphy|Julie|6505555787|5557 North Pendale Street|null|San Francisco|CA|9421  
7|USA|1165|64600.00  
131|Land of Toys Inc.|Lee|Kwai|2125557818|897 Long Airport Avenue|null|NYC|NY|10022|USA|1323|114  
900.00  
141|Euro+ Shopping Channel|Freyre|Diego |(91) 555 94 44|c/ Moralzarza 1, 86|null|Madrid|null|2803  
4|Spain|1370|227600.00  
144|Volvo Model Replicas, Co|Berglund|Christina |0921-12 3555|Berguvsvägen 8|null|Luleå|null|S-  
958 22|Sweden|1504|53100.00  
145|Danish Wholesale Imports|Petersen|Jytte |31 12 3555|Vinbæltet 34|null|København|null|1734|De  
nmark|1401|83400.00  
146|Saveley & Henriot, Co.|Saveley|Mary |78.32.5555|2, rue du Commerce|null|Lyon|null|69004|Fran  
ce|1337|123900.00  
148|Dragon Souveniers, Ltd.|Natividad|Eric|+65 221 7555|Bronz Sok. |Bronz Apt. 3/6 Tesvikiye|Sing
```

Lab – Export de fichiers dans une table

```
[sqoop@sandbox ~]$ sqoop export \
> --connect jdbc:mysql://127.0.0.1/sales_export_db?user=root \
> --table customers \
> --input-fields-terminated-by "|" \
> --export-dir /mydata/sqoopdata/sales_export/customers_v1/customers.dsv
Warning: /usr/hdp/2.2.0.0-2041/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
18/10/28 19:02:29 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5.2.2.0.0-2041
18/10/28 19:02:29 INFO manager.SqlManager: Using default fetchSize of 1000
18/10/28 19:02:29 INFO tool.CodeGenTool: Beginning code generation
18/10/28 19:02:30 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t LIMIT 1
18/10/28 19:02:30 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `customers` AS t LIMIT 1
18/10/28 19:02:30 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.2.0.0-2041/hadoop-mapreduce
Note: /tmp/sqoop-sqoop/compile/75513cf7ea39b4fdb0553af8a0fd2638/customers.java uses or overrides
a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
18/10/28 19:02:34 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-sqoop/compile/75513cf7ea39b4fdb0553af8a0fd2638/customers.jar
18/10/28 19:02:34 INFO mapreduce.ExportJobBase: Beginning export of customers
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hadoop/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/zookeeper/lib/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/2.2.0.0-2041/hive/lib/hive-jdbc-0.14.0.2.2.0.0-2041-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

Lab – Export de fichiers dans une table

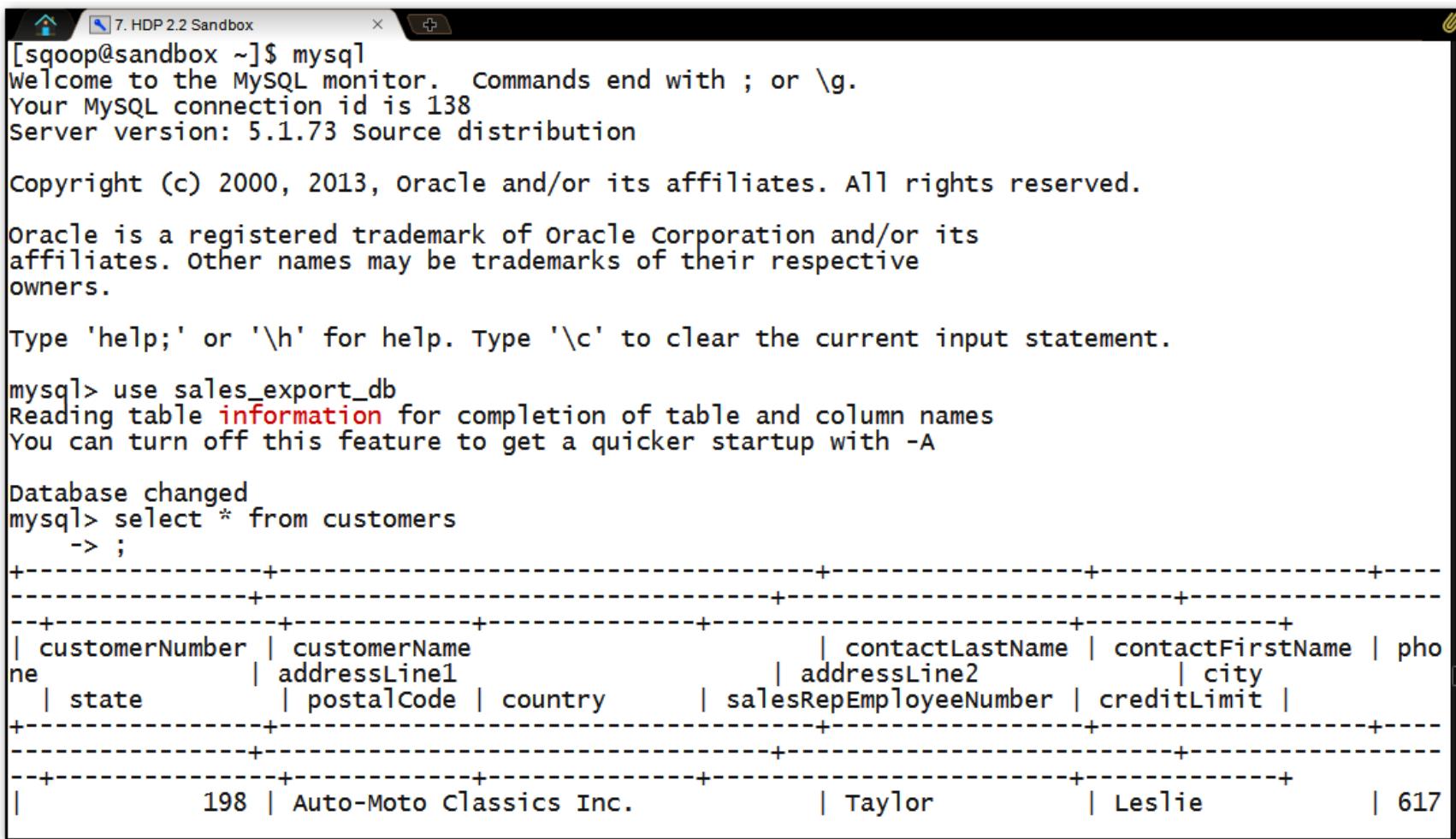
```
7. HDP 2.2 Sandbox
HDFS: Number of large read operations=0
HDFS: Number of write operations=0
Job Counters
    Launched map tasks=4
    Data-local map tasks=4
    Total time spent by all maps in occupied slots (ms)=124240
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=124240
    Total vcore-seconds taken by all map tasks=124240
    Total megabyte-seconds taken by all map tasks=31060000
Map-Reduce Framework
    Map input records=122
    Map output records=122
    Input split bytes=801
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=376
    CPU time spent (ms)=5080
    Physical memory (bytes) snapshot=427610112
    Virtual memory (bytes) snapshot=3124699136
    Total committed heap usage (bytes)=216530944
File Input Format Counters
    Bytes Read=0
File Output Format Counters
    Bytes Written=0
18/10/28 19:03:33 INFO mapreduce.ExportJobBase: Transferred 37.6641 KB in 55.4895 seconds (695.0
508 bytes/sec)
18/10/28 19:03:33 INFO mapreduce.ExportJobBase: Exported 122 records.
[sqoop@sandbox ~]$
```

Lab – Export de fichiers dans une table

► Vérifier le résultat de l'export de données avec les commandes :

```
mysql  
Use sales_export_db  
Select * from customers
```

Lab – Export de fichiers dans une table



```
[sqoop@sandbox ~]$ mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 138
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sales_export_db
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from customers
-> ;
+-----+-----+-----+-----+
| customerNumber | customerName           | contactLastName | contactFirstName | phone
ne          | addressLine1             | addressLine2   | city
          | state                  | postalCode    | country        | salesRepEmployeeNumber | creditLimit |
+-----+-----+-----+-----+
|         198 | Auto-Moto Classics Inc. | Taylor        | Leslie         | 617
+-----+-----+-----+-----+
```

Chapitre 5

L'ECOSYSTEME HADOOP



The Hadoop UI



The Hadoop UI

Qu'est-ce que Hue

- Interface Web pour rendre Hadoop plus facile à utiliser
- Série d'applications pour l'exécution de requêtes, la copie de fichiers, la création de workflows...

The screenshot shows the Hue File Browser interface. At the top, there's a toolbar with various icons for file operations like search, rename, move, copy, change permissions, download, and delete. Below the toolbar is a breadcrumb navigation bar showing the path: Home / user / flume / tweets / 2013 / 02 / 25 / 17. To the right of the breadcrumb is a trash icon. The main area is a table listing files. The columns are: Type, Name, Size, User, Group, Permissions, and Date. The table contains 105 items, all of which are directories named 'FlumeData.1361840400169' through 'FlumeData.1361840400180'. The permissions for all these files are listed as '-rwxr-xr-x'. The date column shows various times from February 25, 2013, at 05:00 pm to 06:00 pm. At the bottom of the table, there are pagination controls: 'Show 45 items per page. Showing 1 to 45 of 105 items, page 1 of 3.', 'Next Page', and 'End of List'.

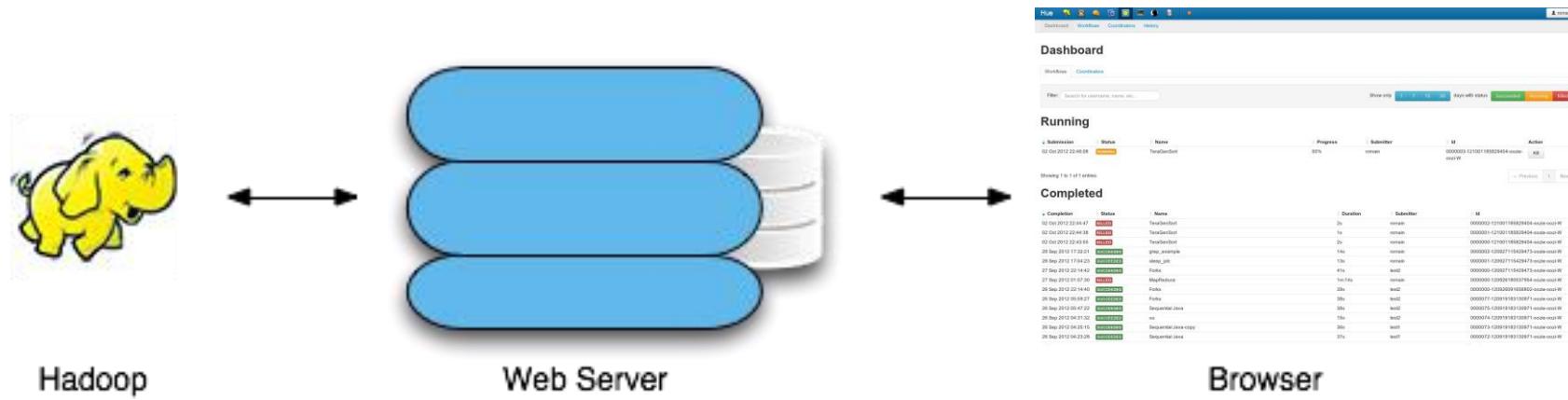
Type	Name	Size	User	Group	Permissions	Date
..	.	52.7 KB	flume	flume	drwxr-xr-x	February 25, 2013 06:00 pm
..	FlumeData.1361840400169	38.3 KB	flume	flume	-rwxr-xr-x	February 28, 2013 10:32 am
..	FlumeData.1361840400170	38.3 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:00 pm
..	FlumeData.1361840400171	38.3 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:01 pm
..	FlumeData.1361840400172	31.6 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:01 pm
..	FlumeData.1361840400173	36.0 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:02 pm
..	FlumeData.1361840400174	37.8 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:03 pm
..	FlumeData.1361840400175	12.9 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:03 pm
..	FlumeData.1361840400176	31.4 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:04 pm
..	FlumeData.1361840400177	25.5 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:04 pm
..	FlumeData.1361840400178	44.1 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:05 pm
..	FlumeData.1361840400179	33.0 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:05 pm
..	FlumeData.1361840400180	35.0 KB	flume	flume	-rwxr-xr-x	February 25, 2013 05:06 pm

La cible de Hue

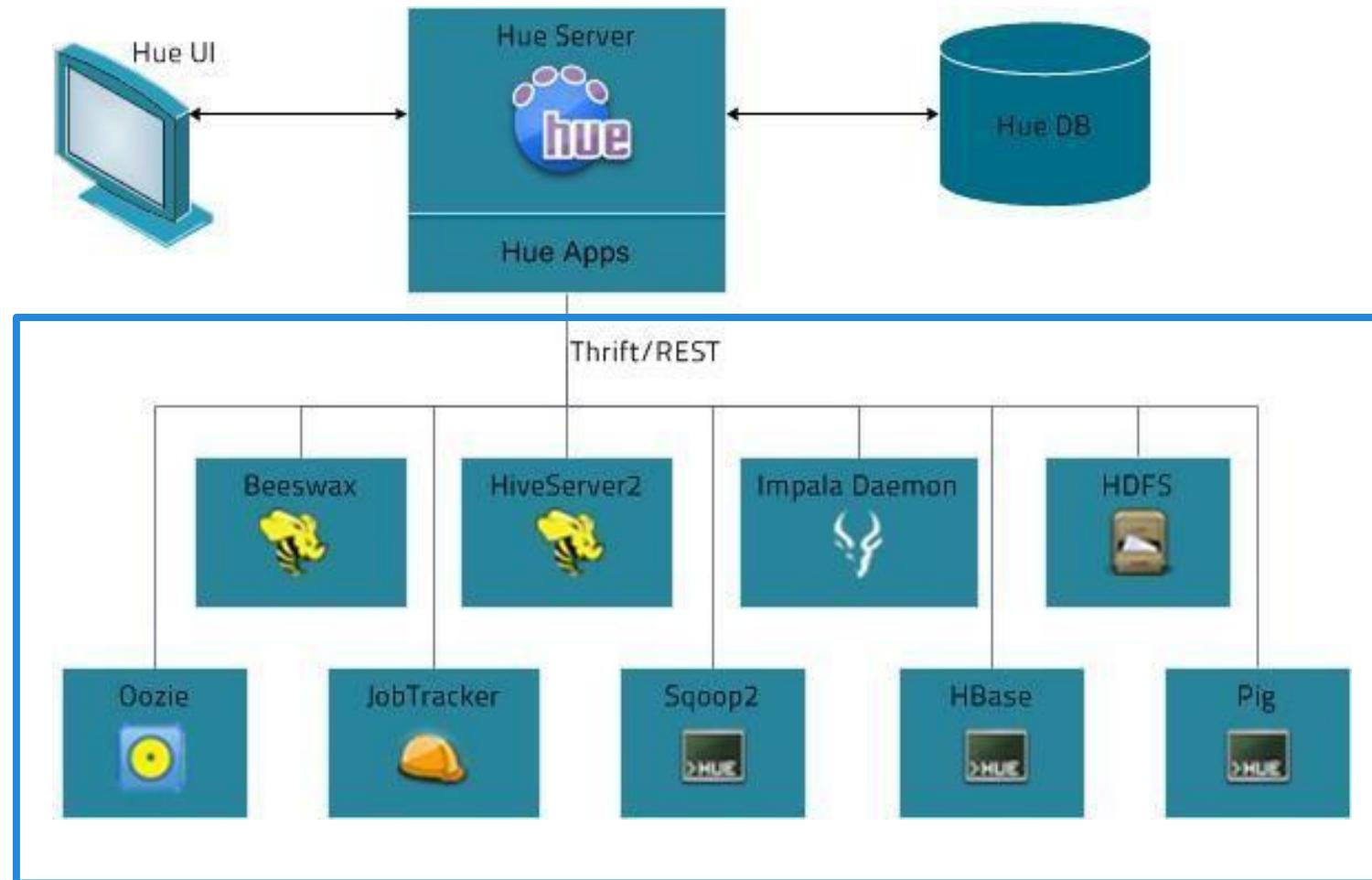
- Démarrer avec Hadoop
- Se familiariser avec les différents angles de la plate-forme et les explorer
- Permet aux utilisateurs de se concentrer sur le traitement du Big Data



Vue de 30 000 pieds



Écosystème



Télécharger des données via le navigateur de fichiers

The screenshot shows the Hue File Browser interface. The top navigation bar includes icons for Home, Hue, and various system services like HDFS, MapReduce, and YARN. A user dropdown shows 'romain'. Below the header is a toolbar with buttons for Search, Rename, Move, Copy, Change Permissions, Download, Delete, New, and Upload.

The main area displays a file listing under the path '/user/flume/tweets/2013/02/25/17'. The listing includes columns for Type, Name, Size, User, Group, Permissions, and Date. The files are named 'FlumeData.1361840400169' through 'FlumeData.1361840400180'. All files are owned by 'flume' and have '-rw-r--r--' permissions. The sizes range from 12.9 KB to 52.7 KB. The dates are mostly February 25, 2013, with one entry on February 28, 2013.

Type	Name	Size	User	Group	Permissions	Date
..	.		flume	flume	drwxr-xr-x	February 25, 2013 06:00 pm
..	..		flume	flume	drwxr-xr-x	February 28, 2013 10:32 am
File	FlumeData.1361840400169	52.7 KB	flume	flume	-rw-r--r--	February 25, 2013 05:00 pm
File	FlumeData.1361840400170	38.3 KB	flume	flume	-rw-r--r--	February 25, 2013 05:01 pm
File	FlumeData.1361840400171	38.3 KB	flume	flume	-rw-r--r--	February 25, 2013 05:01 pm
File	FlumeData.1361840400172	31.6 KB	flume	flume	-rw-r--r--	February 25, 2013 05:02 pm
File	FlumeData.1361840400173	36.0 KB	flume	flume	-rw-r--r--	February 25, 2013 05:02 pm
File	FlumeData.1361840400174	37.8 KB	flume	flume	-rw-r--r--	February 25, 2013 05:03 pm
File	FlumeData.1361840400175	12.9 KB	flume	flume	-rw-r--r--	February 25, 2013 05:03 pm
File	FlumeData.1361840400176	31.4 KB	flume	flume	-rw-r--r--	February 25, 2013 05:04 pm
File	FlumeData.1361840400177	25.5 KB	flume	flume	-rw-r--r--	February 25, 2013 05:04 pm
File	FlumeData.1361840400178	44.1 KB	flume	flume	-rw-r--r--	February 25, 2013 05:05 pm
File	FlumeData.1361840400179	33.0 KB	flume	flume	-rw-r--r--	February 25, 2013 05:05 pm
File	FlumeData.1361840400180	35.0 KB	flume	flume	-rw-r--r--	February 25, 2013 05:06 pm

At the bottom, there are buttons for 'Show 45 items per page. Showing 1 to 45 of 105 items, page 1 of 3.', 'Next Page', and 'End of List'.

Afficher le contenu des fichiers

Actions		Viewing Bytes: 1 - 4096 of 32368 (4096 B block size)			
		First Block	Previous Block	Next Block	Last Block
View As Binary					
Edit File					
Download					
View File Location					
Refresh					
INFO					
Last Modified	Feb. 25, 2013 5:02 p.m.				
User	flume				
Group	flume				
Size	31.6 KB				
Mode	100644				
<pre>{"filter_level": "medium", "contributors": null, "text": "RT @higher_rhythm Big thanks to Pete @rightstreamit. Our server literally exploded. Repaired + data restored thanks to his back-up system...", "geo": null, "retweeted": false, "in_reply_to_screen_name": null, "truncated": false, "entities": {"urls": [], "hashtags": [], "user_mentions": [{"id": 173515075, "name": "Steve Mundin (Boss)", "indices": [3, 17]}, {"screen_name": "higher_rhythm", "id_str": "173515075"}, {"id": 239577603, "name": "Pete Gibson", "indices": [37, 51]}, {"screen_name": "RightstreamIT", "id_str": "239577603"}]}, "in_reply_to_status_id": null, "retweet_count": 0, "created_at": "Tue Feb 26 01:01:33 +0000 2013", "in_reply_to_user_id": null, "id_str": "306207339013369857", "place": null, "user": {"location": "Doncaster", "default_profile": false, "statuses_count": 61412, "profile_background_tile": false, "lang": "en", "profile_link_color": "#009999", "profile_banner_url": "https://si0.twimg.com/profile_banners/228300826/1348327337", "id": 228300826, "following": null, "favourites_count": 2, "protected": false, "profile_text_color": "#333333", "description": "Roofing Services in and around the Doncaster area will RT most Tweets with #doncasterisgreat OR #KPRS in it ", "verified": false, "contributors_enabled": false, "profile_sidebar_border_color": "#EEEEEE", "name": "K Pearson Roofing", "profile_background_color": "#131516", "created_at": "Sun Dec 19 09:30:50 +0000 2010", "default_profile_image": false, "followers_count": 28619, "profile_image_url_https": "https://si0.twimg.com/profile_images/2511884349/w8riw67sfq9vz96qv0ux_normal.jpeg", "geo_enabled": true, "profile_background_image_url": "http://a0.twimg.com/profile_background_images/599193420/466ds0qa9hrhrlf7mjki.jpeg", "profile_background_image_url_https": "https://si0.twimg.com/profile_background_images/599193420/466ds0qa9hrhrlf7mjki.jpeg", "follow_request_sent": null, "url": "http://kpearsonroofing.com", "utc_offset": "0", "time_zone": "London", "notifications": null, "profile_use_background_image": true, "friends_count": 22216, "profile_sidebar_fill_color": "#FFFFFF", "screen_name": "kpearsonroofing", "id_str": "228300826", "profile_image_url": "http://a0.twimg.com/profile_images/2511884349/w8riw67sfq9vz96qv0ux_normal.jpeg", "listed_count": 84, "is_translator": false, "coordinates": null}, "filter_level": "medium", "contributors": null, "text": "Big Data's Value Lies in Self-Regulation: John C. Havens is the founder of The H(app)athon P... http://t.co/xWihORADeF RT @break_w_katie", "geo": null, "retweeted": false, "in_reply_to_screen_name": null, "possibly_sensitive": false, "truncated": false, "entities": {"urls": [{"expanded_url": "http://bit.ly/V2TBns", "indices": [96, 118]}, {"play_url": "bit.ly/V2TBns", "url": "http://t.co/xWihORADeF"}], "hashtags": [], "user_mentions": [{"id": 471439414, "name": "Katrina Cole", "indices": [122, 136]}, {"screen_name": "Break_w_Katie", "id_str": "471439414"}]}, "in_reply_to_status_id": null, "id": 306207362656649216, "source": "twitterfeed", "in_reply_to_user_id": null, "favorited": false, "in_reply_to_status_id": null, "retweet_count": 0, "created_at": "Tue Feb 26 01:01:38 +0000 2013", "in_reply_to_user_id": null, "id_str": "306207362656649216", "place": null, "user": {"location": "US", "default_profile": false, "statuses_count": 5396, "profile_background_tile": false, "lang": "en", "profile_link_color": "#990000", "id": 471439414, "following": null, "favourites_count": 33, "protected": false, "profile_text_color": "#333333", "description": "I am an admitted workaholic, love consulting and helping others grow their business, marketing, social media and visibility in their unique market places.", "verified": false, "contributors_enabled": false, "profile_sidebar_border_color": "#990000", "name": "Katrina Cole", "profile_background_color": "#EBEBEB", "created_at": "Sun Jan 22 21:16:35 +0000 2012", "default_profile_image": false, "followers_count": 64, "profile_image_url_https": "https://si0.twimg.com/profile_images/1773931611/brightidea_normal.jpg", "geo_enabled": true}, "ue", }</pre>					

Créer une table dans hive ui

The screenshot shows the Hue Query Editor interface. The top navigation bar includes icons for Home, Hue, Help, and a user dropdown set to 'flume'. Below the bar, a menu bar has tabs for Query Editor, My Queries, Saved Queries, History, Tables, and Settings.

The main area is divided into two sections: 'DATABASE' on the left and 'Query Editor' on the right.

DATABASE: A dropdown menu is set to 'default'. Other options include 'bigdata' and 'flume'.

FILE RESOURCES: A 'Type' dropdown is set to 'jar', and the 'Path' field contains '/user/flume/hive-serdes-1.0-SNAPSHOT.jar'. There is also an 'Add' button for file resources.

USER-DEFINED FUNCTIONS: An 'Add' button is present.

PARAMETERIZATION: A checked checkbox for 'Enable Parameterization'.

EMAIL NOTIFICATION: An unchecked checkbox for 'Email me on completion'.

Query Editor: The title is 'Query Editor'. The code area contains the following HiveQL script:

```
1 CREATE EXTERNAL TABLE tweets (
2   id BIGINT,
3   created_at STRING,
4   source STRING,
5   favorited BOOLEAN,
6   retweet_count INT,
7  retweeted status STRUCT<
8     text:STRING,
9     user:STRUCT<screen_name:STRING,name:STRING>,
10    entities STRUCT<
11      urls:ARRAY<STRUCT<expanded_url:STRING>>,
12      user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
13      hashtags:ARRAY<STRUCT<text:STRING>>,
14      text STRING,
15      user STRUCT<
16        screen_name:STRING,
17        name:STRING,
18        friends_count:INT,
19        followers_count:INT,
20        statuses_count:INT,
21        verified:BOOLEAN,
22        utc_offset:INT,
23        time_zone:STRING>,
24      in_reply_to_screen_name STRING
25    )
26  PARTITIONED BY (datehour INT)
27  ROW FORMAT SERDE 'com.cloudera.hive.serde.JSONSerDe'
28  LOCATION '/user/flume/tweets'
29
30
31
```

At the bottom of the code area are buttons for 'Execute', 'Save as...', 'Explain', 'or create a', and 'New query'.

Liste des tables dans l'application Catalogue

The screenshot shows the Hue Catalogue interface. At the top, there is a blue header bar with various icons and a dropdown menu labeled "hdfs". Below the header, the URL "Home default" is displayed. The main area is titled "Tables". On the left, there is a sidebar with a "DATABASE" dropdown set to "default" and several "ACTIONS" options: "Install samples", "Create a new table from a file", and "Create a new table manually". On the right, there is a search bar with placeholder text "Search...", and three buttons: "View", "Browse Data", and "Drop". Below the search bar, there is a table with a single column labeled "Table Name". The table contains six entries: "page_view", "sample_07", "sample_08", "test", and "tweets". Each entry has a small checkbox icon to its left.

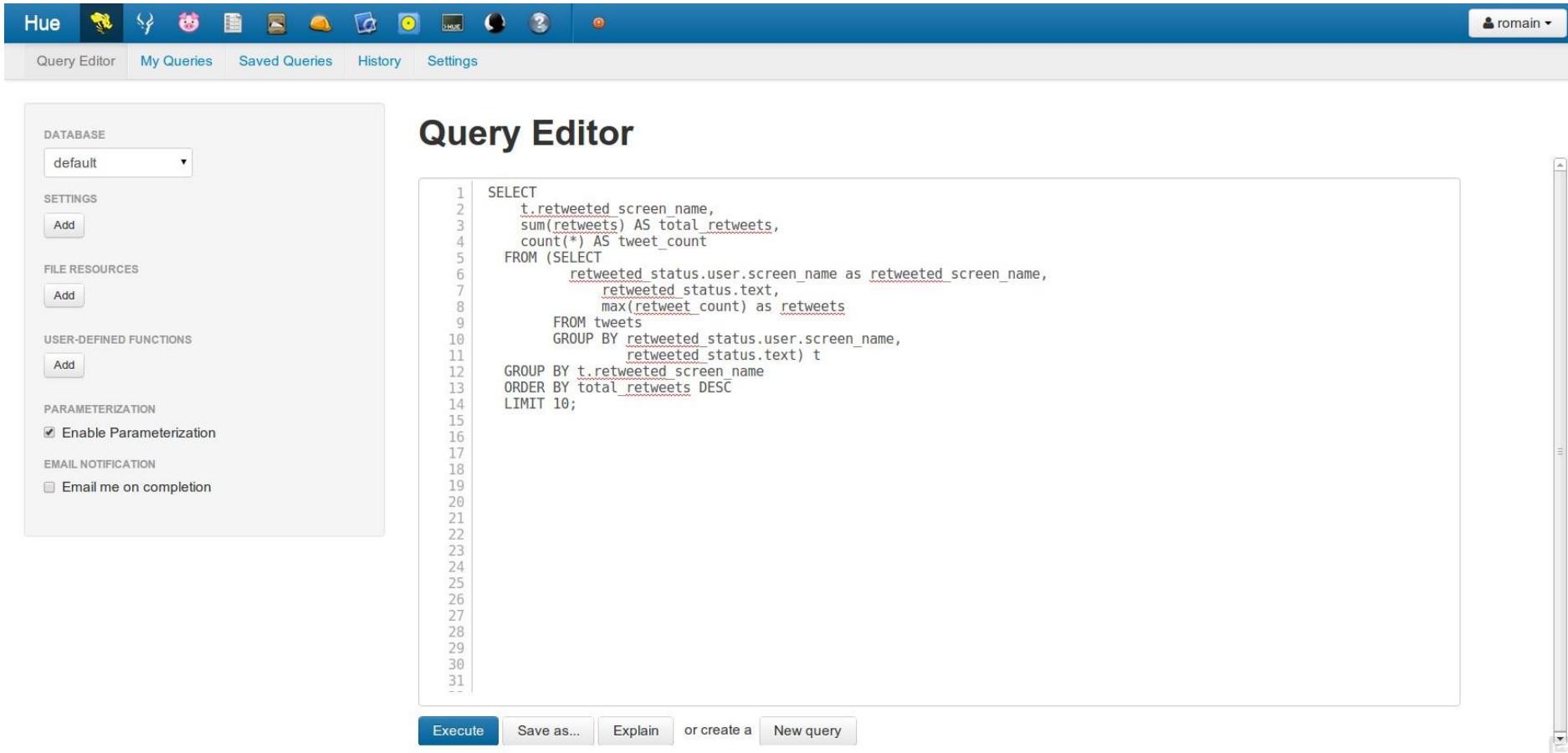
Table Name
page_view
sample_07
sample_08
test
tweets

Assistants et navigateur de métadonnées

The screenshot shows the Hue interface for managing data in Hadoop. The top navigation bar includes icons for Home, Browse, Import, Export, and various system status indicators. A dropdown menu for 'hdfs' is visible. Below the header, the breadcrumb navigation shows 'Home / default / tweets'. The main title is 'Table Metadata: tweets'. On the left, a sidebar titled 'ACTIONS' lists 'Import Data', 'Browse Data', 'Drop Table', and 'View File Location'. The central area displays the table schema with tabs for 'Columns', 'Partition Columns', and 'Sample'. The 'Columns' tab is selected, showing the following schema:

Name	Type
created_at	string
entities	struct<urls:array<struct<expanded_url:string>>,user_mentions:array<struct<screen_name:string,name:string>>,hashtags:array<struct<text:string>>>
favorited	boolean
id	bigint
in_reply_to_screen_name	string
retweet_count	int
retweeted_status	struct<text:string,user:struct<screen_name:string,name:string>>
source	string
text	string
user	struct<screen_name:string,name:string,friends_count:int,followers_count:int,statuses_count:int,verified:boolean,utc_offset:int,time_zone:string>

Exécuter la requête avec hive



The screenshot shows the Hue Query Editor interface. The top navigation bar includes icons for Home, Hue, Help, and a user dropdown labeled "romain". Below the navigation is a menu bar with tabs: "Query Editor" (selected), "My Queries", "Saved Queries", "History", and "Settings". The main area is titled "Query Editor" and contains a sidebar with configuration options: DATABASE (set to "default"), SETTINGS (with an "Add" button), FILE RESOURCES (with an "Add" button), USER-DEFINED FUNCTIONS (with an "Add" button), PARAMETERIZATION (checkbox "Enable Parameterization" checked), and EMAIL NOTIFICATION (checkbox "Email me on completion" unchecked). The central code editor displays a Hive query:

```
1 SELECT
2     t.retweeted_screen_name,
3     sum(retweets) AS total_retweets,
4     count(*) AS tweet_count
5 FROM (SELECT
6     retweeted_status.user.screen_name as retweeted_screen_name,
7     retweeted_status.text,
8     max(retweet_count) as retweets
9     FROM tweets
10    GROUP BY retweeted_status.user.screen_name,
11                      retweeted_status.text) t
12 GROUP BY t.retweeted_screen_name
13 ORDER BY total_retweets DESC
14 LIMIT 10;
```

At the bottom of the editor are buttons for "Execute", "Save as...", "Explain", "or create a", and "New query".

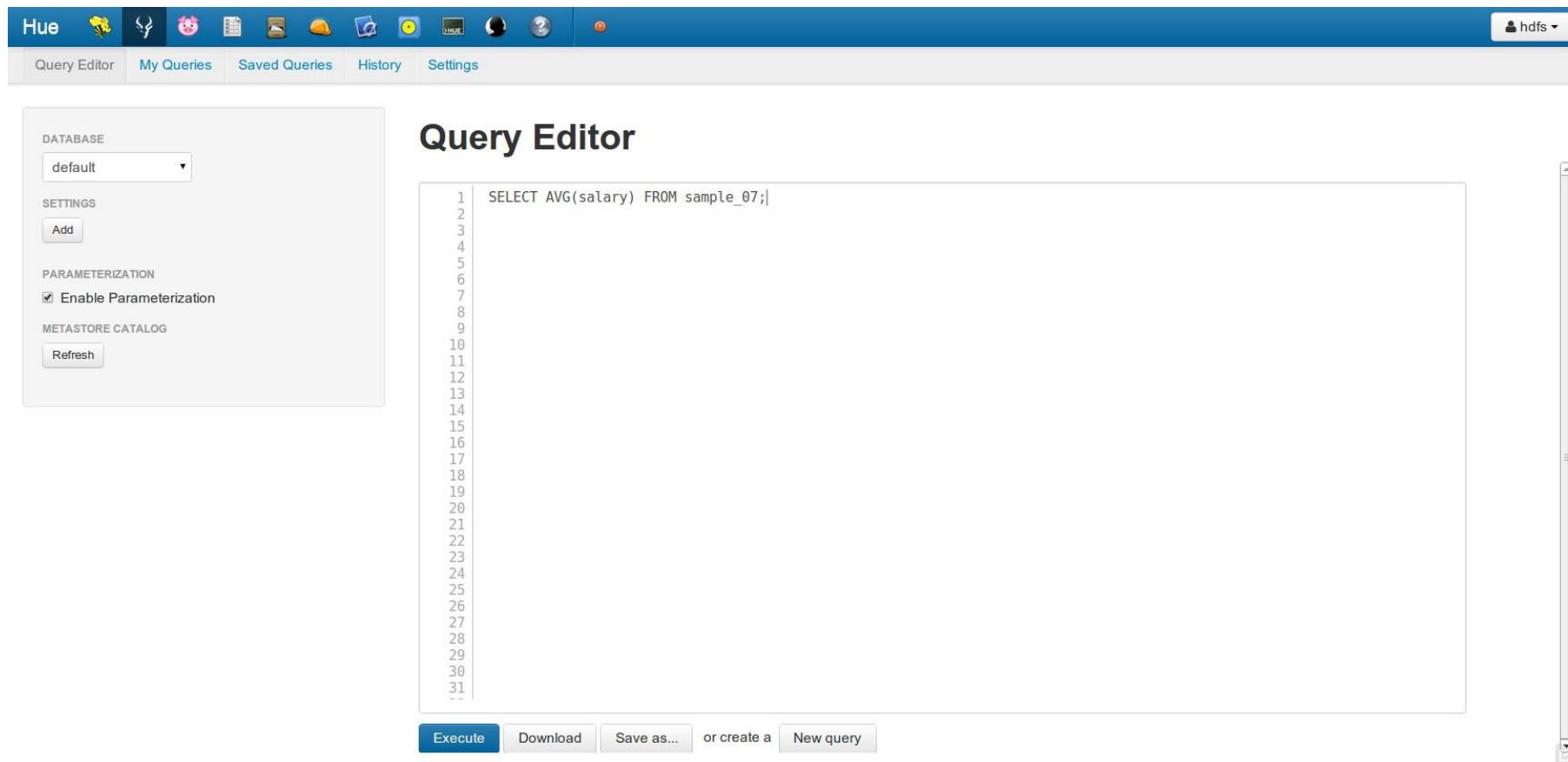
Liens directs vers des tâches MR ou à partir de Job Browser

The screenshot shows the Hue Job Browser interface. At the top, there's a blue header bar with the Hue logo and various icons. On the right of the header is a user dropdown labeled "romain". Below the header, the title "Job Browser" is centered. Underneath the title are search and filter fields: "Job status: All States" with a dropdown arrow, a checkbox for "Show retired jobs", "Username: romain", and a "Text: Text Filter" input field.

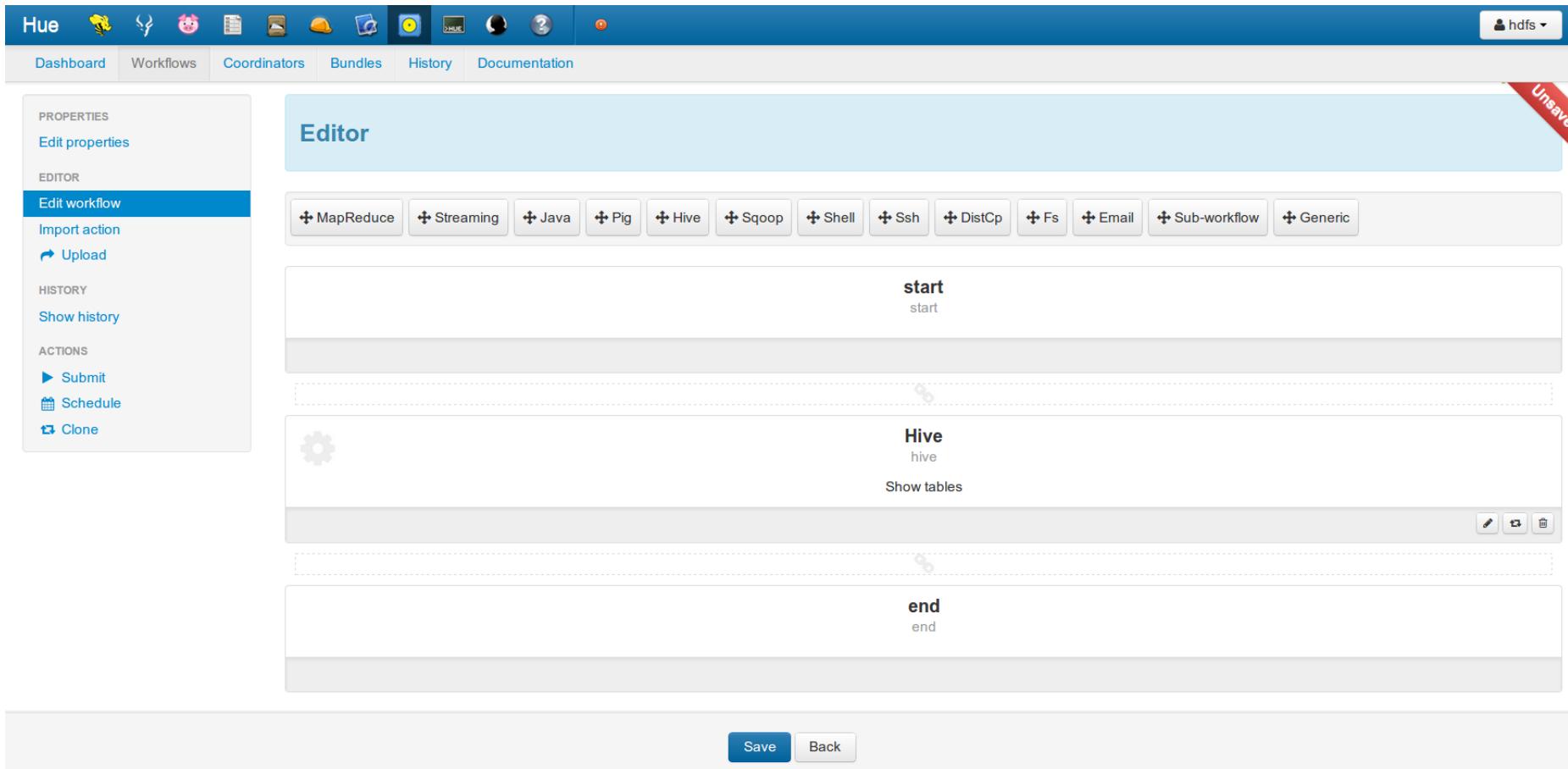
The main area is a table listing 12 completed Hadoop jobs. The columns are: Logs (with a dropdown arrow), ID, Name, Status, User, Maps, Reduces, Queue, Priority, Duration, and Date. Each job entry includes a small green icon to the left of the ID, the job name, its status (all are "succeeded"), the user (all are "romain"), and the number of maps and reduces (all are 1/1). The "Maps" and "Reduces" columns contain green progress bars. The "Queue" column shows "default" for all jobs. The "Priority" column shows "normal". The "Duration" column shows times like "6s", "7s", and "6s". The "Date" column shows dates like "03/27/13 12:16:54", "03/27/13 12:04:10", and "03/27/13 11:30:13".

Logs	ID	Name	Status	User	Maps	Reduces	Queue	Priority	Duration	Date
	201303181417_0139	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000064-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 12:16:54
	201303181417_0138	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000063-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	7s	03/27/13 12:04:10
	201303181417_0137	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000062-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	7s	03/27/13 12:01:50
	201303181417_0136	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000061-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 11:30:13
	201303181417_0135	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000060-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 11:29:38
	201303181417_0134	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000059-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 11:28:41
	201303181417_0133	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000058-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 11:27:30
	201303181417_0132	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000057-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	7s	03/27/13 11:17:14
	201303181417_0131	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000056-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 11:03:45
	201303181417_0130	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000055-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 10:59:48
	201303181417_0129	oozie:launcher:T=pig:W=pig-app-hue-script:A=pig:ID=0000054-130318141800797-oozie-oozi-W	succeeded	romain	1 / 1	0 / 0	default	normal	6s	03/27/13 10:50:01

SQL rapide avec Impala



Insertion de requêtes Hive dans un workflow



Exécution du flux de travail

The screenshot shows the Hue interface for managing Hadoop workflows. The top navigation bar includes links for Dashboard, Workflows, Coordinators, Bundles, History, and Documentation. The current view is the 'Dashboard' under the 'Workflows' tab.

Workflow Details:

- WORKFLOW:** LoadTwitterData - hdfs
- SUBMITTER:** hdfs
- STATUS:** RUNNING
- PROGRESS:** 50%
- ID:** 0000065-130318141800797-oozie-oozi-W
- VARIABLES:** Manage buttons for Kill or Suspend

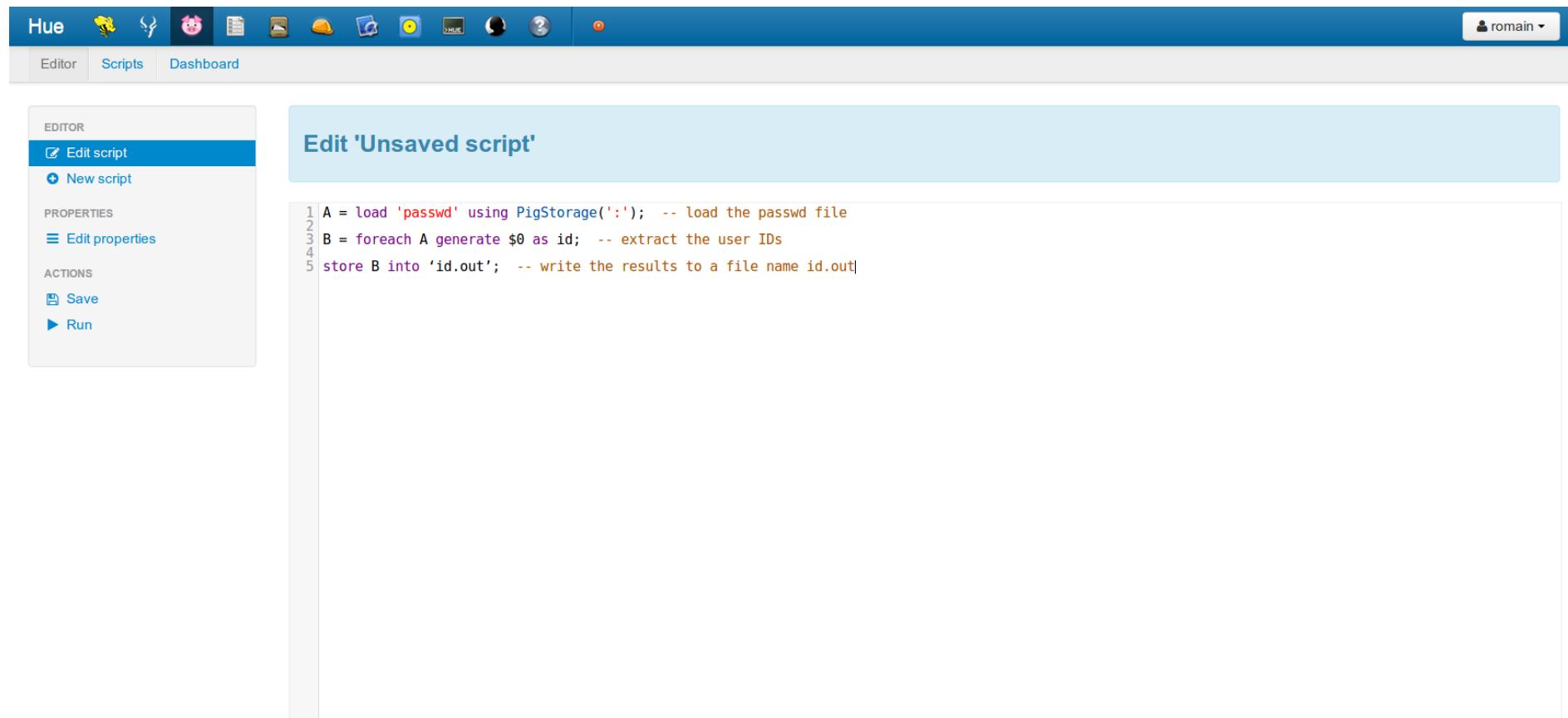
Workflow Graph:

- start:** Status OK, RUNNING
- Hive:** Status RUNNING
- end:** Placeholder step

Actions: Graph, Actions, Details, Configuration, Log, Definition

Log: Show tables

Éditeur pig



Apache Hive



D'où vient Hive ?

- Chez Facebook manipulation d'une quantité monstre des données chaque jour
- Avoir des alternatives à
 - MapReduce
 - Pig
- Rentabiliser les ingénieurs qui connaissent et maîtrisent SQL !

C'est quoi Hive ?

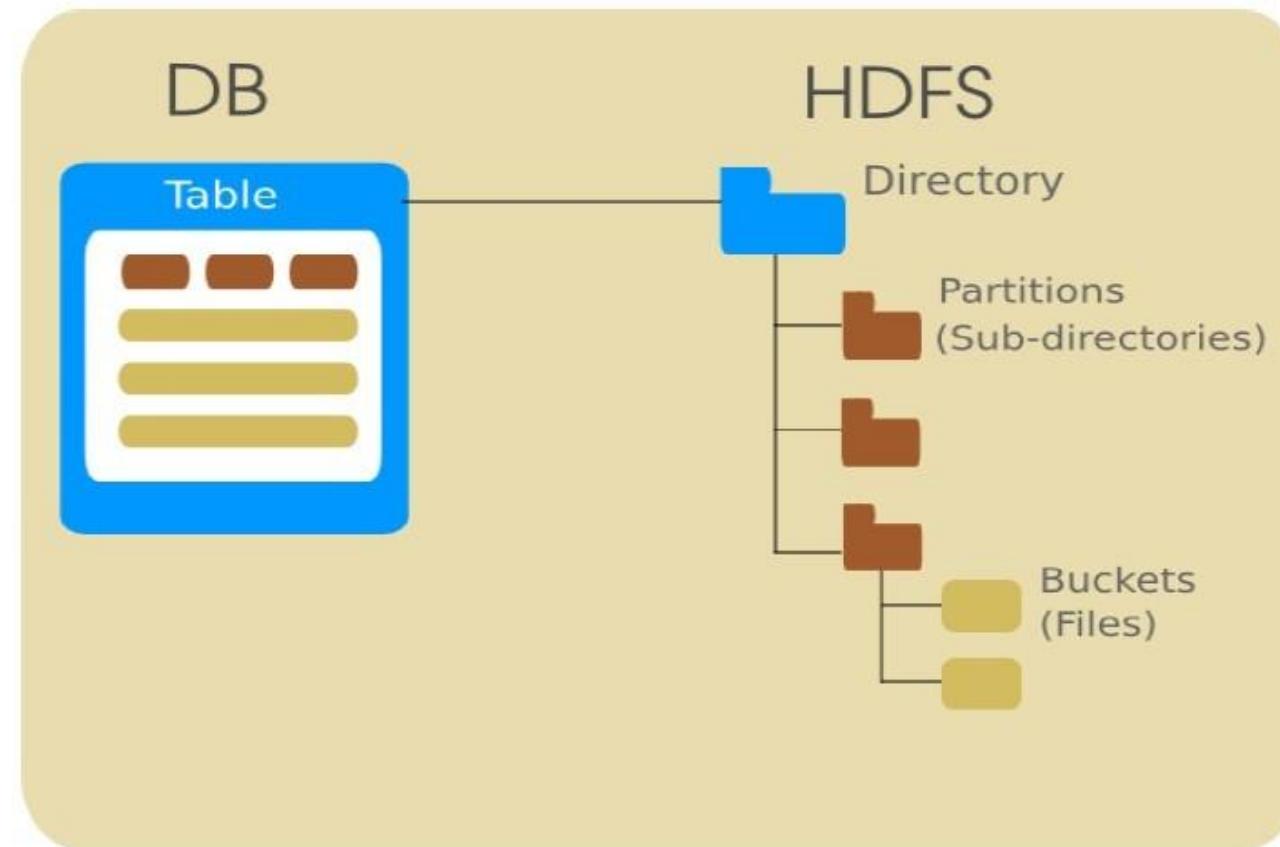
Une infrastructure pour Entrepôt de Données

**on peut tout simplement le considérer comme
un Data Warehouse !**

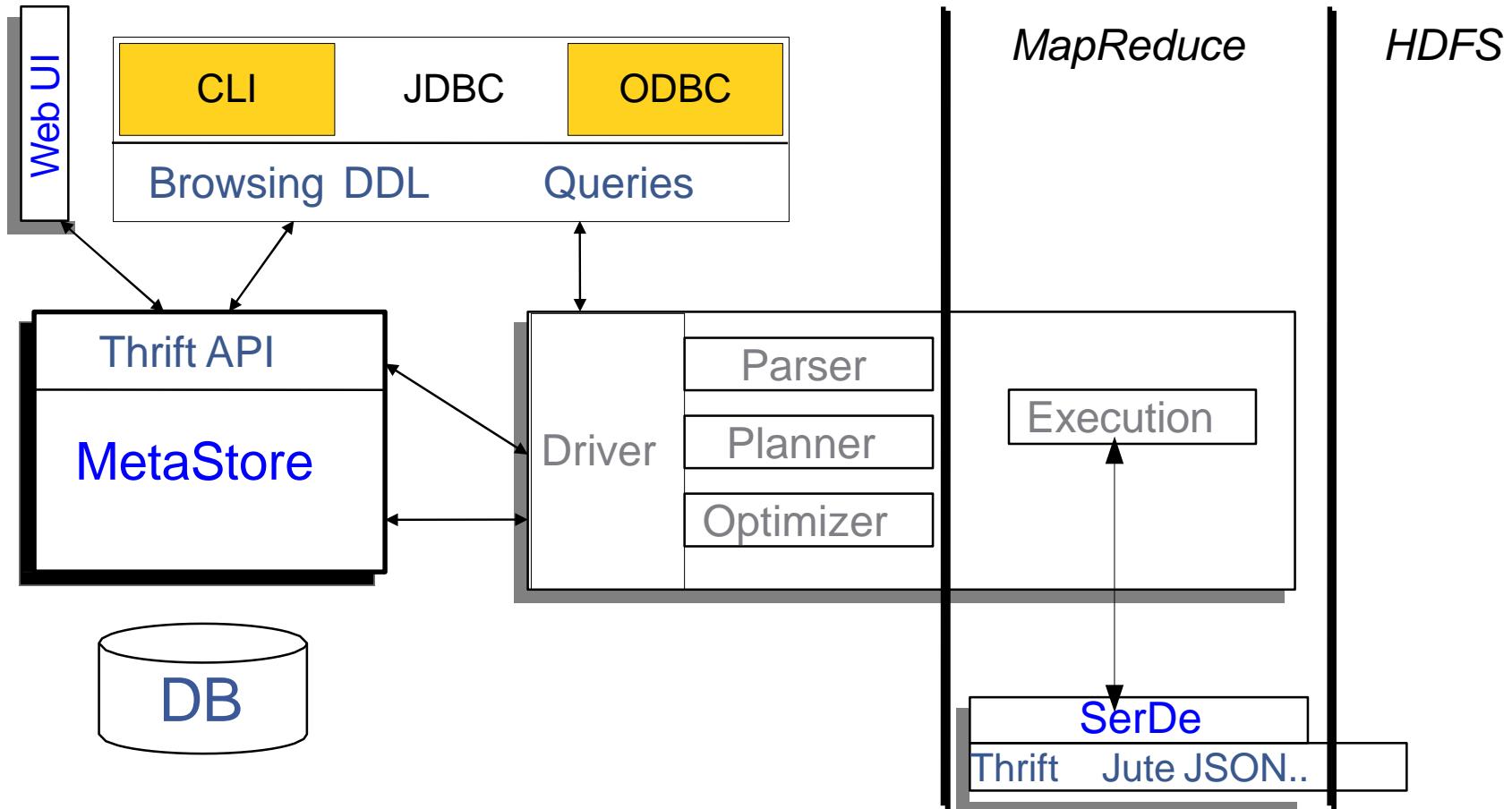
Comment ça marche ?

- Hive a pour fondation Hadoop
- Hive stocke ses données dans HDFS
- Hive compile des requêtes SQL en jobs MapReduce et les exécute sur le cluster Hadoop

Hive structure la donnée dans un modèle bien connu : Tables, colonnes, lignes...



Hive architecture, composants



Hive Architecture, détail composants (1)

- **Interface Externe**
 - fournit deux sortes d'interfaces
 - utilisateur
 - commande line (CLI)
 - WEB
 - programmation (API)
 - JDBC, ODBC
 - **Thrift Server** expose une API très simple pour exécuter les requêtes HiveQL
- **Metastore** est un catalogue.
 - Tous les autres composants de Hive interagissent avec le Metastore.

Hive Architecture, détail composants (2)

- **Driver**

Gère le cycle de vie des requêtes HiveQL durant leur compilation, leur optimisation, et leur exécution

- **Compiler**

Transforme les requêtes en un plan qui constitue une suite de DAG de job MapReduce

Le Driver soumet les jobs mapReduce individuels depuis le DAG vers le **moteur d'exécution** selon une séquence

Hive Architecture, détail composants (3)

- Le **Metastore** est un catalogue qui contient les métadonnées des tables stockées dans Hive
- **Database** : espace de nom pour les tables
- **Table** : Métadonnées des tables qui contiennent la liste des colonnes, leur types, leurs propriétaires, leur emplacements de stockages et les informations de sérialisation
- **Partition** : chaque partition peut avoir ses propres colonnes, son stockage et sa sérialisation
- **Buckets** : découpage des partitions (optimisation pour les jointures)

Cycle de vie d'une requête

Parser transforme la requête en arbre

Semantic Analyzer transforme l'arbre en une représentation interne de la requête par bloc

Logical Plan Generator convertit la représentation interne en un plan logique, c'est-à-dire un arbre d'opérations logiques

Optimizer parcourt plusieurs fois le plan logique d'opérations pour le rendre plus performant

Physical Plan Generator convertit le plan logique en un plan physique de DAG de job mapReduce

HiveQL, SQL pour Hive

- HiveQL supporte
 - DDL (Create, Alter, Drop)
 - DML (load, Insert, Select)
 - Fonction utilisateurs
 - Appel à des programmes externe MapReduce

Création de tables

SHOW TABLES;

CREATE TABLE shakespeare (freq **INT**, word **STRING**)
ROW FORMAT

DELIMITED FIELDS TERMINATED **BY** '\t'

STORED **AS** TEXTFILE;

DESCRIBE shakespeare;

Chargement de données depuis HDFS

```
LOAD DATA LOCAL INPATH '/logs/status_updates' INTO  
TABLE status_updates PARTITION (ds='2013-05- 28');
```

Sélection / Insertion résultat

```
FROM (
    SELECT a.status, b.school, b.gender
    FROM status_updates a
    JOIN profiles b ON (a.userid = b.userid and a.ds='2013-05-28')
) subq1
INSERT OVERWRITE TABLE gender_summary PARTITION(ds='2013-05-28')

SELEC subq1.gender, COUNT(1) GROUP BY subq1.gender
T
OVERWRITE TABLE school_summary PARTITION(ds='2009-05-28')
INSERT

SELECT subq1.school, COUNT(1) GROUP BY subq1.school
```

Apache Pig



C'est quoi Pig ?

- Crée chez Yahoo!
- Une plate-forme **très simple pour traiter les Big Data**
- **PigLatin** : langage dont le traitement est en flux, simple, proche du scripting, 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 ?

- 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 (filters, joins, ordering) , des types primitifs, des types complexe (tuples, bags, maps)

Bien plus simple à comprendre pour un analyste que du MapReduce

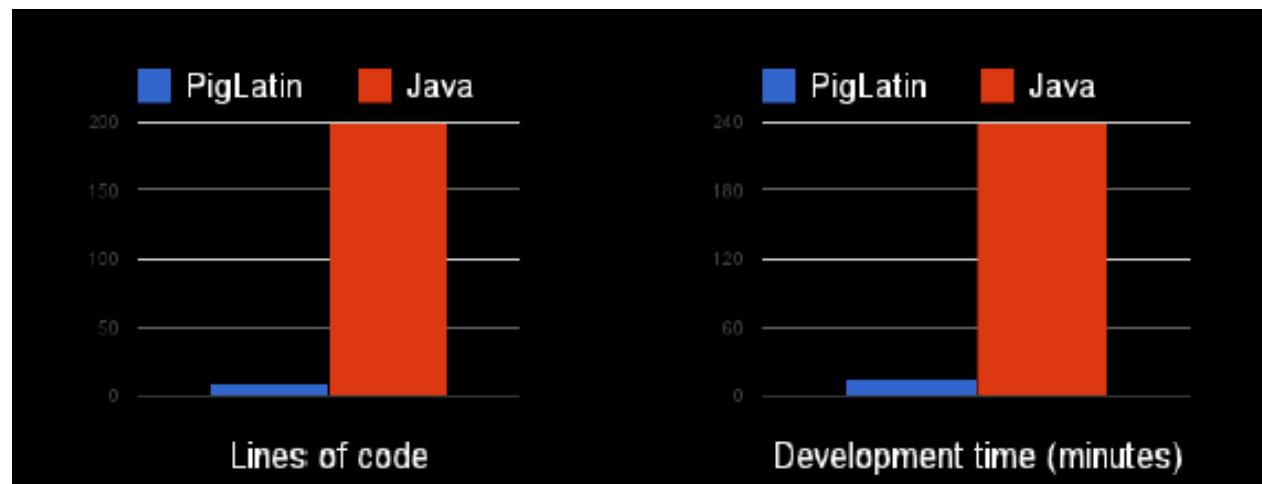
Il ouvre Hadoop au non-programmeur-java

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 count;
-- 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
Srtd = ORDER Smmd BY clicks DESC;
-- Get top 5 pages
Lmt = LIMIT Srtd 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

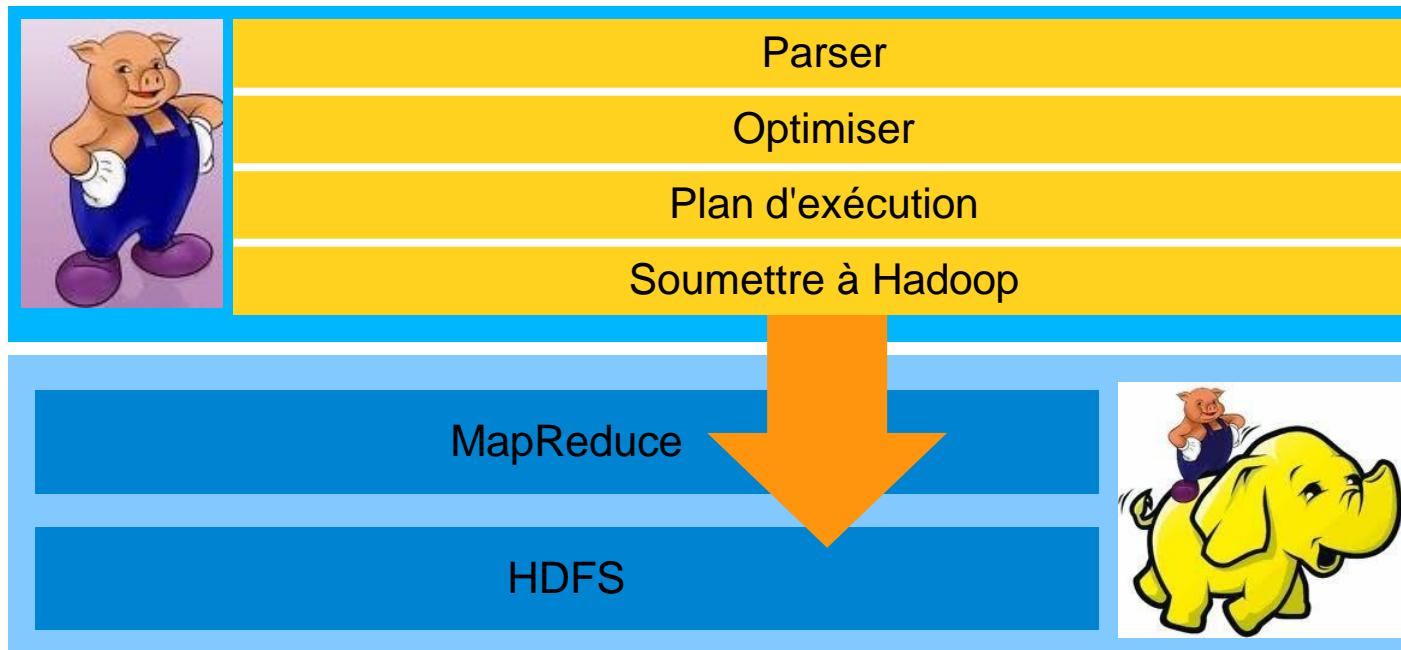
Facile à étendre

- Possibilité d'embarqué
 - Python
 - JavaScript
- Intégré à HBase

Communauté très active

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 !

- Ecplise
 - 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;
```

```
grunt> DUMP Unique;
```



Question ?



Merci !

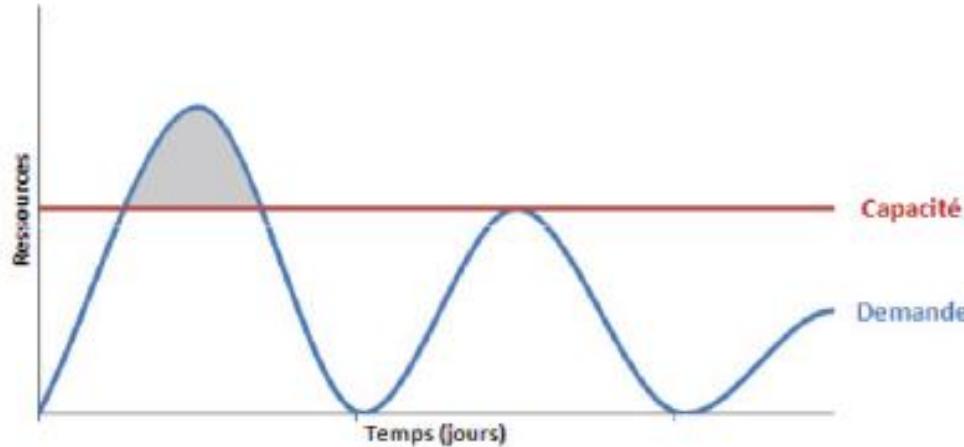
Cloud Computing

Amel HAJI

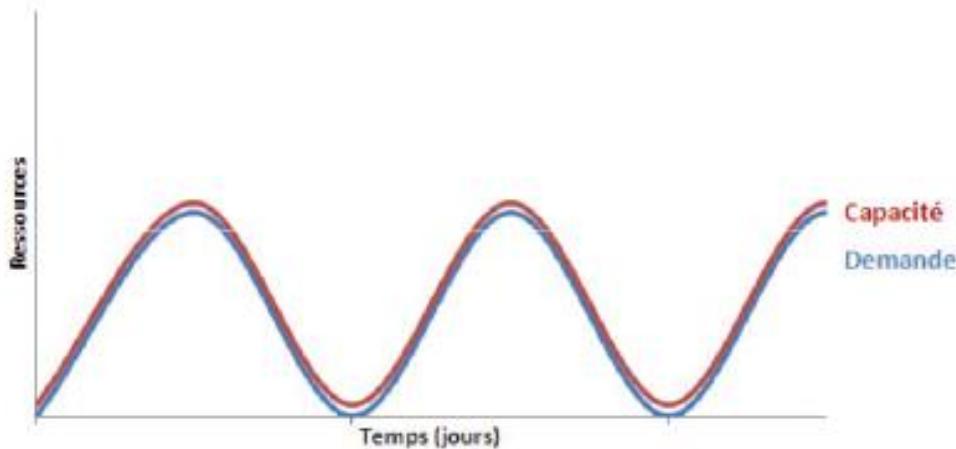
Cloud Computing

1. Contexte
2. Définition
3. Modèles de service du Cloud Computing
4. Modèles de déploiement du Cloud Computing
5. Avantages du Cloud Computing
6. Limites du Cloud Computing

1. Contexte: Pourquoi le Cloud Computing ?

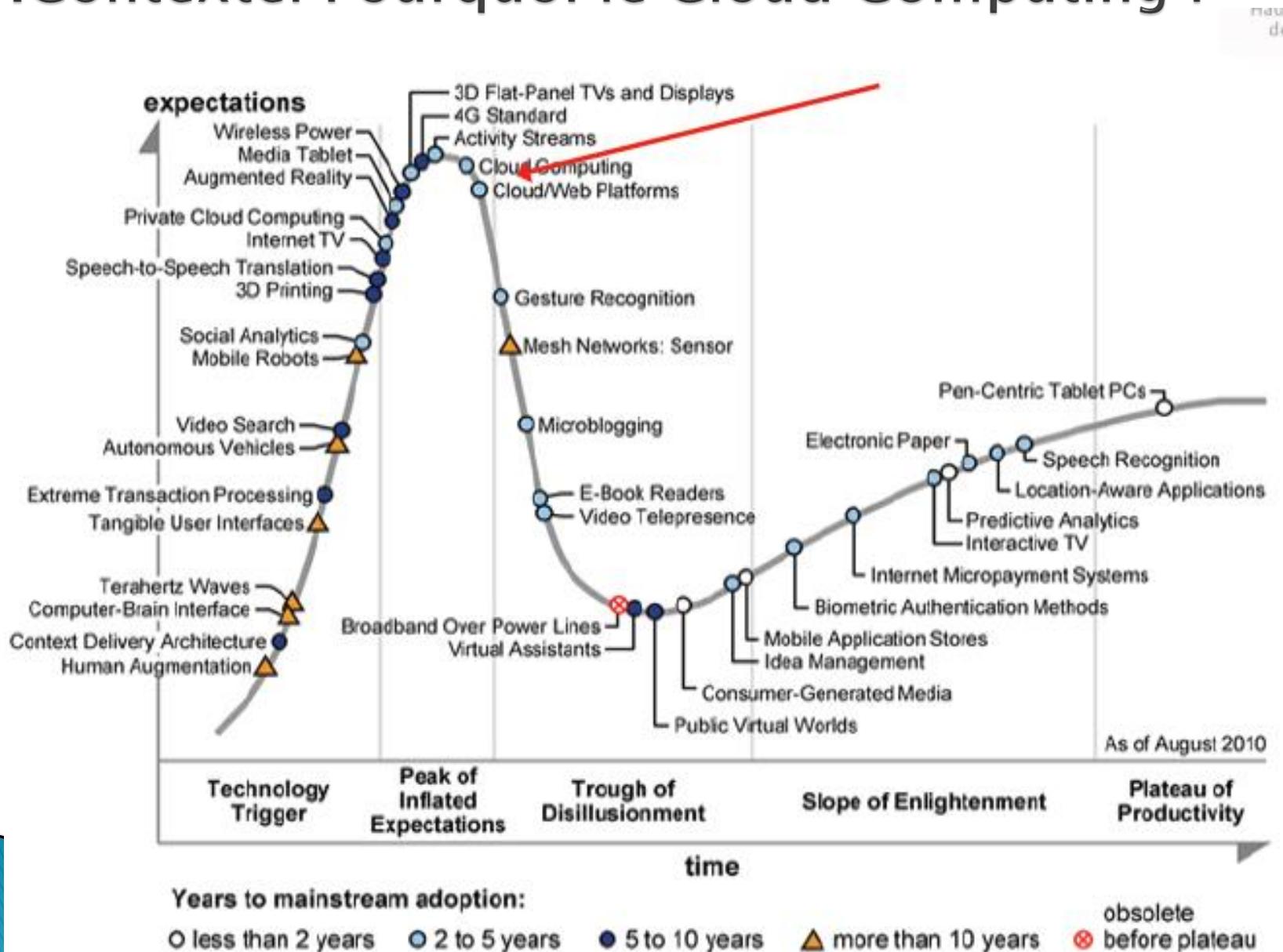


On-premise IT

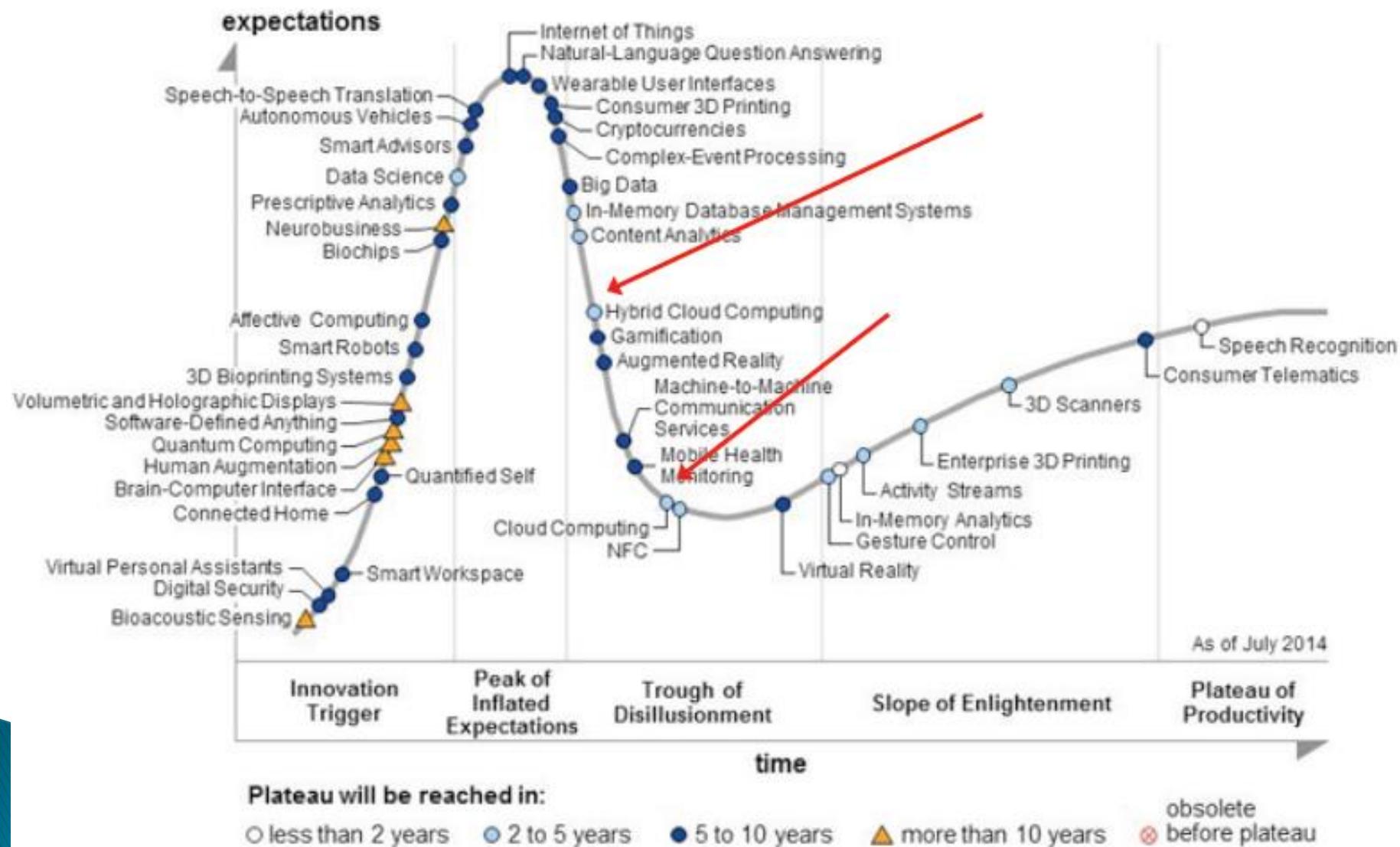


Cloud computing

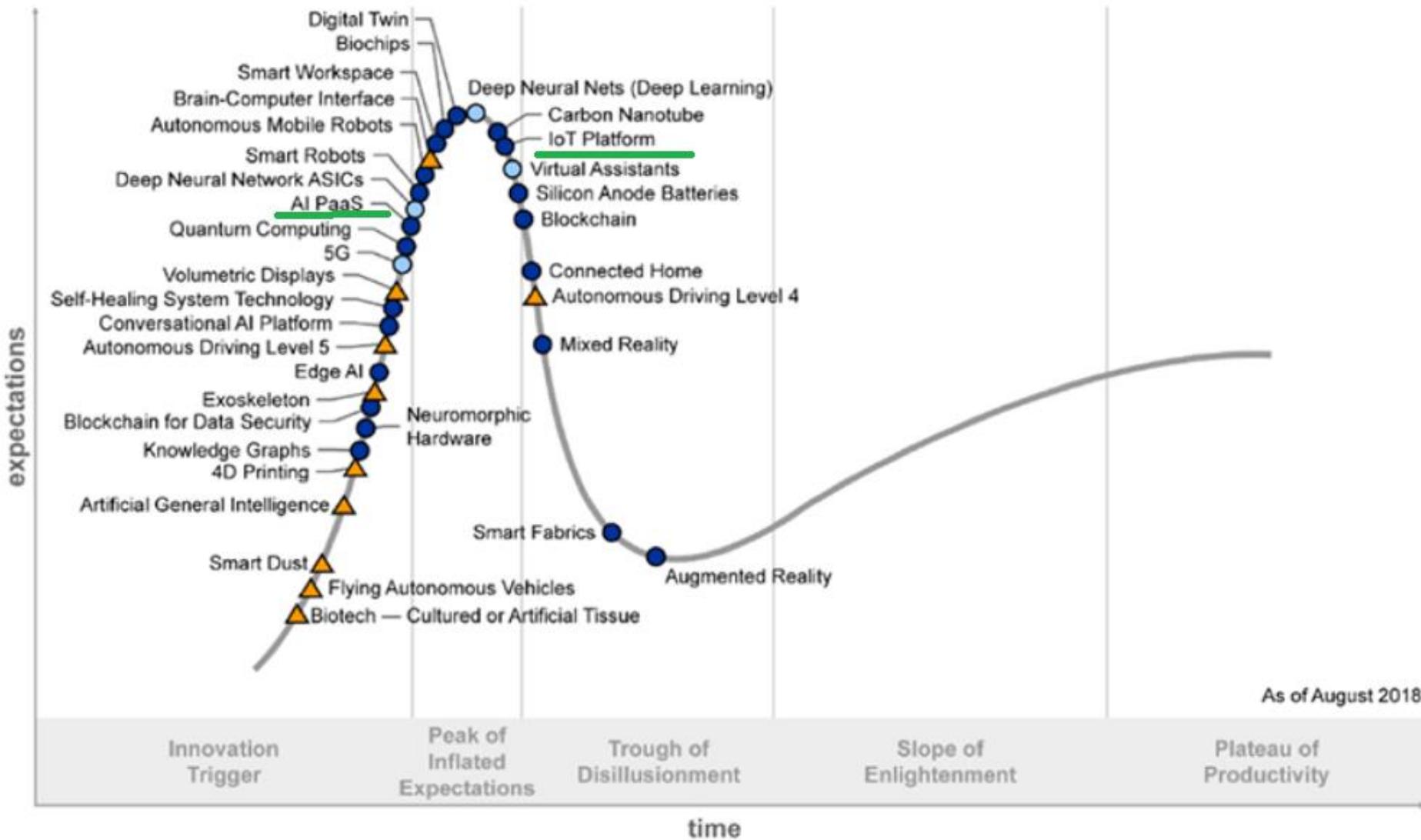
1. Contexte: Pourquoi le Cloud Computing ?



1. Contexte: Pourquoi le Cloud Computing ?



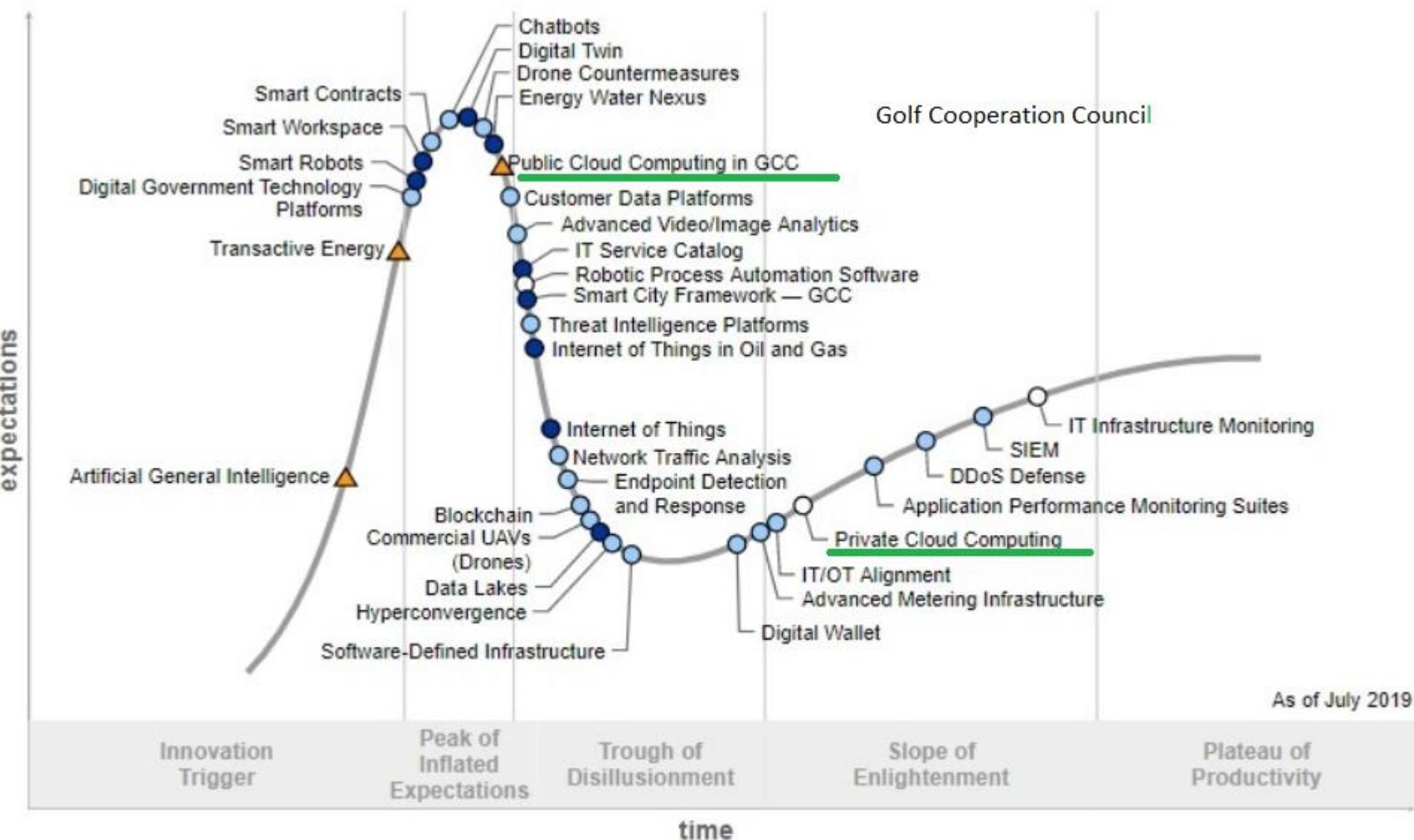
1. Contexte: Pourquoi le Cloud Computing ?



Plateau will be reached:

- less than 2 years
- 2 to 5 years
- 5 to 10 years
- ▲ more than 10 years
- ✗ obsolete before plateau

1. Contexte: Pourquoi le Cloud Computing ?



Plateau will be reached:

○ less than 2 years ○ 2 to 5 years ● 5 to 10 years ▲ more than 10 years ✖ chocolate before plateau

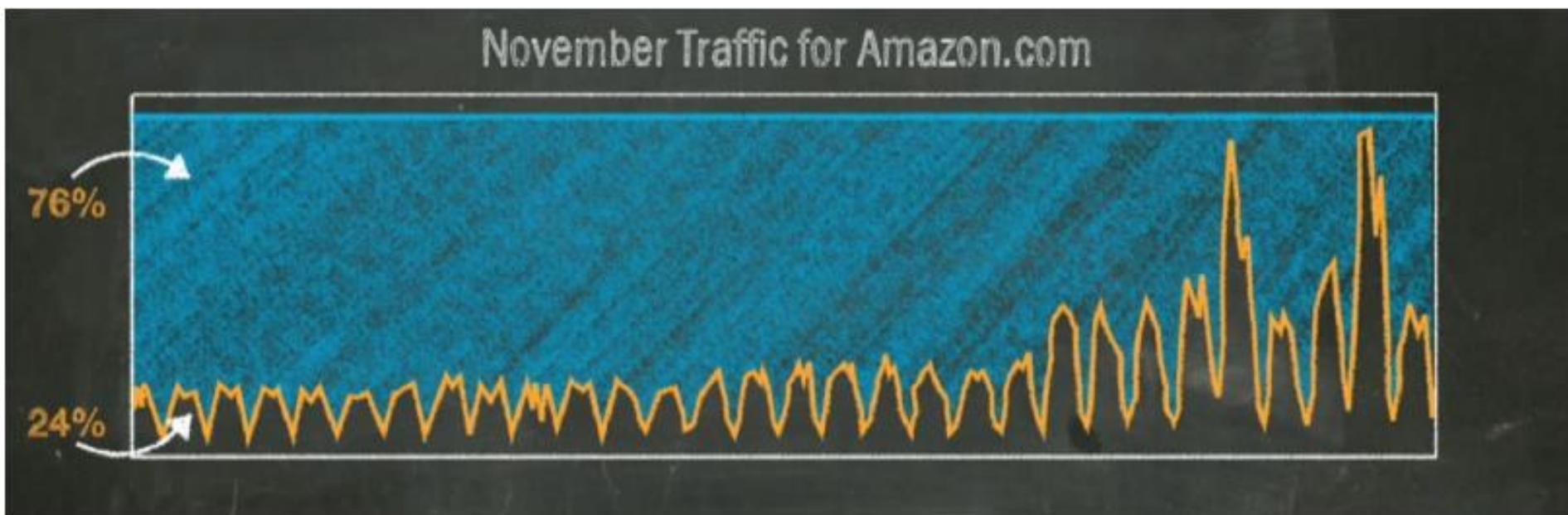
1. Contexte: Pourquoi le Cloud Computing ?

- ▶ Trafic hebdomadaire typique sur le site Web de commerce électronique d'Amazon en 2007



1. Contexte: Pourquoi le Cloud Computing ?

- ▶ Trafic au mois de novembre 2007



2. Cloud Computing:Définitions

- ▶ Il existe plusieurs définitions du terme Cloud Computing
- ▶ Dépend de l'usage qu'en font les utilisateurs
- ▶ Différents points de vue.
 - cabinets d'analyse
 - universitaires
 - l'industrie
 - entreprises informatiques

2. Cloud Computing:Définitions

Selon les cabinets d'analyses

- ▶ Style d'informatique, dans lequel des **fonctionnalités informatiques extrêmement évolutives** sont fournies «en tant que service pour **plusieurs clients externes** en utilisant les technologies Internet (Gartner 2008)
- ▶ Un modèle IT de développement émergent, de déploiement et de fourniture, permettant la livraison en temps réel de **produits, services et solutions sur Internet** (IDC 2008)
- ▶ Solution de stockage d'information
 - Se concentre sur la donnée indépendamment du support
 - est capable de la restituer indépendamment de sa localisation (**Cigref**)

2. Cloud Computing:Définitions

- ▶ Ces définitions ont des caractéristiques communes:
 - Ils définissent le Cloud Computing du point de vue des utilisateurs finaux
 - La principale caractéristique du Cloud est la configuration de l'infrastructure et des applications informatiques en tant que **service de manière évulsive**

2. Cloud Computing:Définitions

Selon NIST (National Institute of Standards and Technology)

- ▶ Le Cloud Computing est un modèle qui permet
 - un **accès omniprésent**,
 - pratique et **à la demande** à un réseau **partagé** et à un ensemble de **ressources informatiques configurables** (réseaux, serveurs, stockage, des applications et des services) qui peuvent être **rapidement provisionnées et libérées** avec un **minimum d'administration**.

2. Cloud Computing:Définitions

Selon NIST (National Institute of Standards and Technology)

5 caractéristiques essentielles:

- **Libre-service à la demande**

- ❖ (approvisionnement automatique sans intervention humaine)

- **Accès réseau étendu**

- ❖ (accès via des protocoles standardisés)

- **Mise en commun des ressources**

- ❖ (servir plusieurs clients)

- **Élasticité rapide**

- ❖ (provisionnement / déprovisioning rapide pour l'intégration, capacité illimitée)

- **Service mesuré**

- ❖ (l'utilisation est surveillée et contrôlée, offrant une transparence dans la facturation)

2. Cloud Computing: Définitions

Selon NIST (National Institute of Standards and Technology)

Caractéristiques principales

Réseaux à bande
passante élevée

Elasticité
Souplesse

Services
mesurés

Services à la
carte

Partage des ressources

Modèles de services

SaaS

Software as a Service

PaaS

Platform as a Service

IaaS

Infrastructure as a Service

Modèles de déploiement

Public

Privé

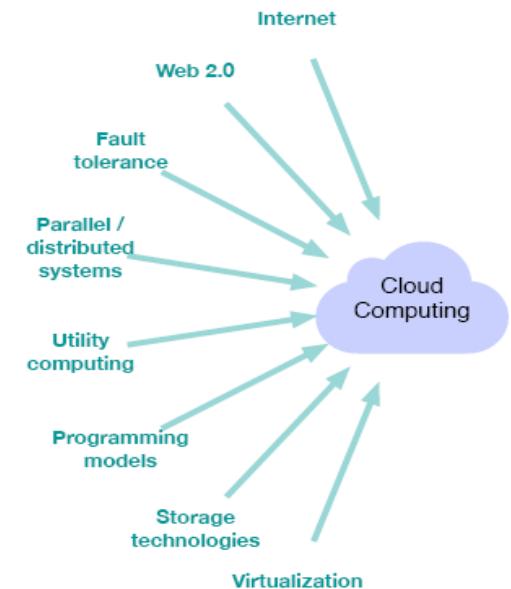
Hybride

Communautaire

2. Cloud Computing:Définitions

Est-ce que le Cloud Computing est une technologie?

- ▶ **Non...** C'est une combinaison de technologies pré-existantes
- ▶ ces technologies ont évolué à des rythmes différents, et n'ont pas été conçus comme un tout cohérent
- ▶ Ils se sont réunis pour créer un écosystème pour le Cloud Computing
- ▶ Ces technologies sont:
 - Navigateurs et clients légers
 - Connexion Internet à haut débit
 - Technologie de virtualisation



2. Cloud Computing:Définitions (en résumé)

- ▶ **Point de vue non technique**
- ▶ Ce n'est qu'un moyen de déployer l'architecture de votre entreprise d'une manière qui a le potentiel d'être **plus productif et à moindre cout**.
- ▶ En substance, il s'agit d'un **outil**. Ce n'est pas nouveau, mais si on s'en approche correctement, il pourrait être un chemin vers l'efficacité

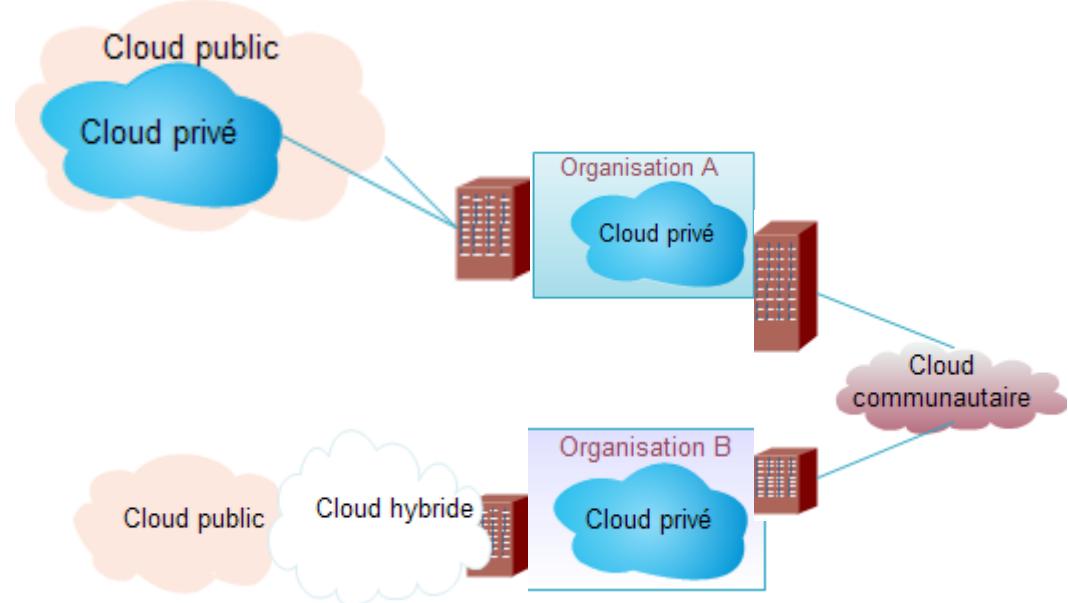
2. Cloud Computing:Définitions (en résumé)

Point de vue technique

- ▶ CC est basé sur des business modèles du type « **Pay as you use** »
- ▶ Les principales fonctionnalités de CC sont basées sur la **virtualisation et l'évolutivité dynamique à la demande**
- ▶ Les services dans le Cloud sont consommés via le navigateur Web ou API définie
- ▶ CC est **élastique** et massivement **évolutif**
- ▶ **Auto approvisionnement** des ressources

3. Cloud Computing: Modèles de déploiement

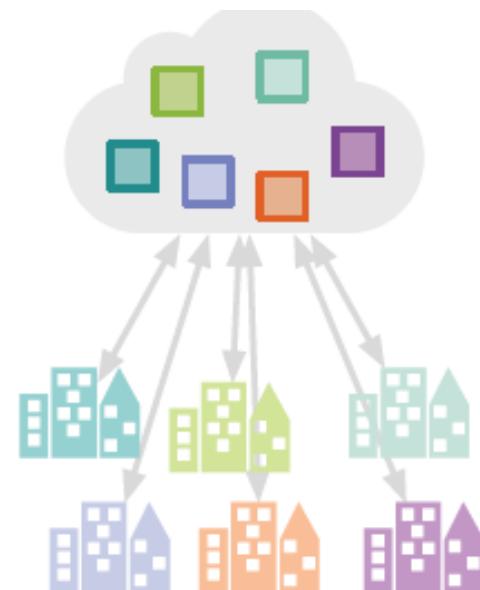
- Cloud Public**
- Cloud privé**
- Cloud Hybride**
- Cloud Communautaire**



3. Cloud Computing: Modèles de déploiement

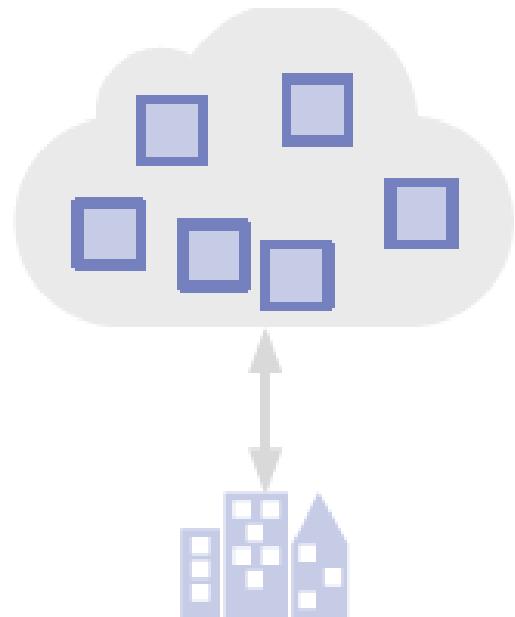
Cloud Public: Les services sont livrés au client, via Internet, à partir d'un fournisseur de services tiers

Fournisseur possède une infrastructure dont il loue les services à plusieurs entreprises (**Externe-ouvert**)



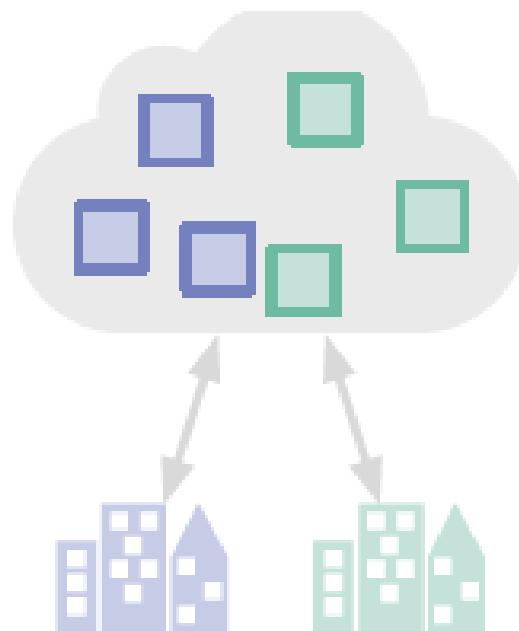
3. Cloud Computing: Modèles de déploiement

- ▶ **Cloud Privé:** Ses services sont gérés et fournis par l'organisation (1 Seul client)
 - * Il y a moins de restriction sur la bande passante réseau et moins de risques concernant la sécurité
- Une seule organisation possède son Infrastructure et la gère uniquement pour ses besoins
Pas de partage avec d'autres organisations
(Interne-privé)



3. Cloud Computing: Modèles de déploiement

- ▶ **Cloud communautaire:** C'est la combinaison entre les services de Cloud privé. Le Cloud est hébergé par un fournisseur de service pour un nombre restreint d'utilisateurs
- ▶ infrastructure partagée par plusieurs Organisations et concerne une communauté spécifique qui partage des intérêts communs (Externe-privé)



3. Cloud Computing: Modèles de déploiement

- ▶ **Cloud hybride:** C'est la combinaison des services fournis par le Cloud public et le Cloud privé
(Interne-ouvert)

Les principaux fournisseurs du Cloud Computing ?



Microsoft Azure



IBM Cloud



IBM Bluemix™



Google Compute Engine

Google
App Engine™



heroku

af appfog

force.com™
platform as a service

Google docs



Office
online



OneDrive

salesforce

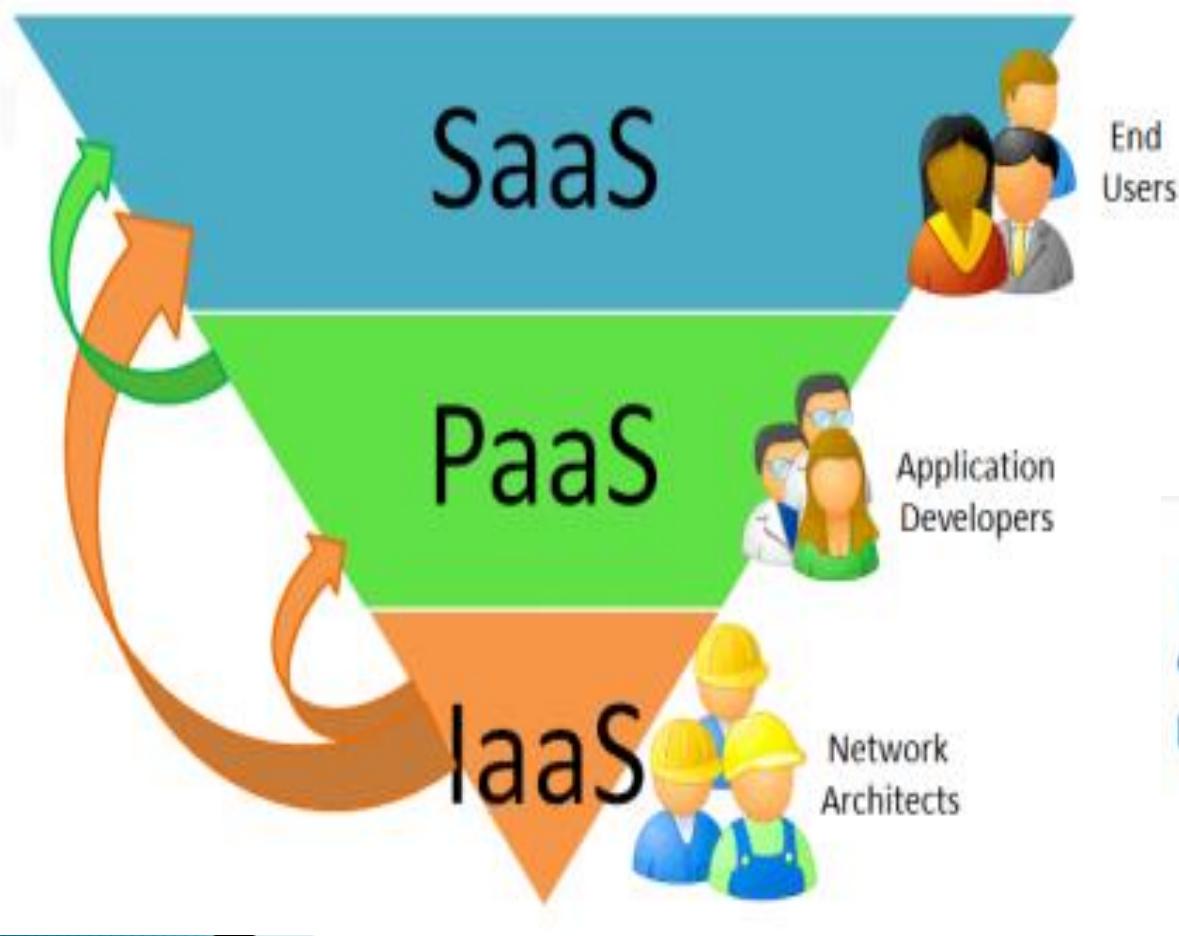


workday.

NETSUITE



4. Cloud Computing: Modèles de service



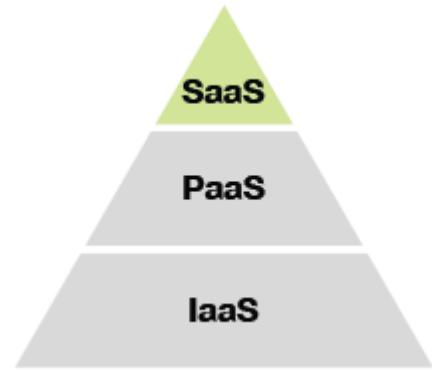
Le Cloud Computing se résume en :

- Services disponibles et accessibles à tous, instantanément,
- sans engagement
- À la demande.

Utiliser ➔ SaaS
Construire ➔ PaaS
Héberger ➔ IaaS

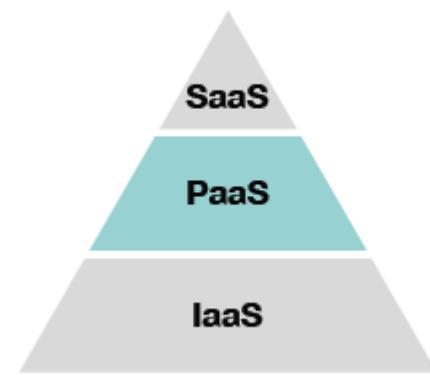


4.1 Modèles de service: SaaS Software as a Service



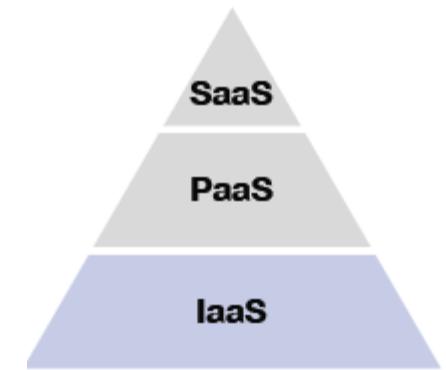
- ▶ Les applications ou logiciels hébergés sont consommés **directement par les utilisateurs**.
- ▶ L'application est déjà construite et opérationnelle, pas de développement mais plutôt du paramétrage
- ▶ les **fournisseurs de services** se chargent
 - de **maintenir et de gérer** les logiciels, les données et l'infrastructure sous-jacente.
 - Exemples: Messagerie, Google (Gmail, docs,...), Google Drive, Lotus Live, NetSuite, Online Office, Salesforce CRM, DropBox.

4.2 Modèles de service: PaaS Platform as a Service PaaS



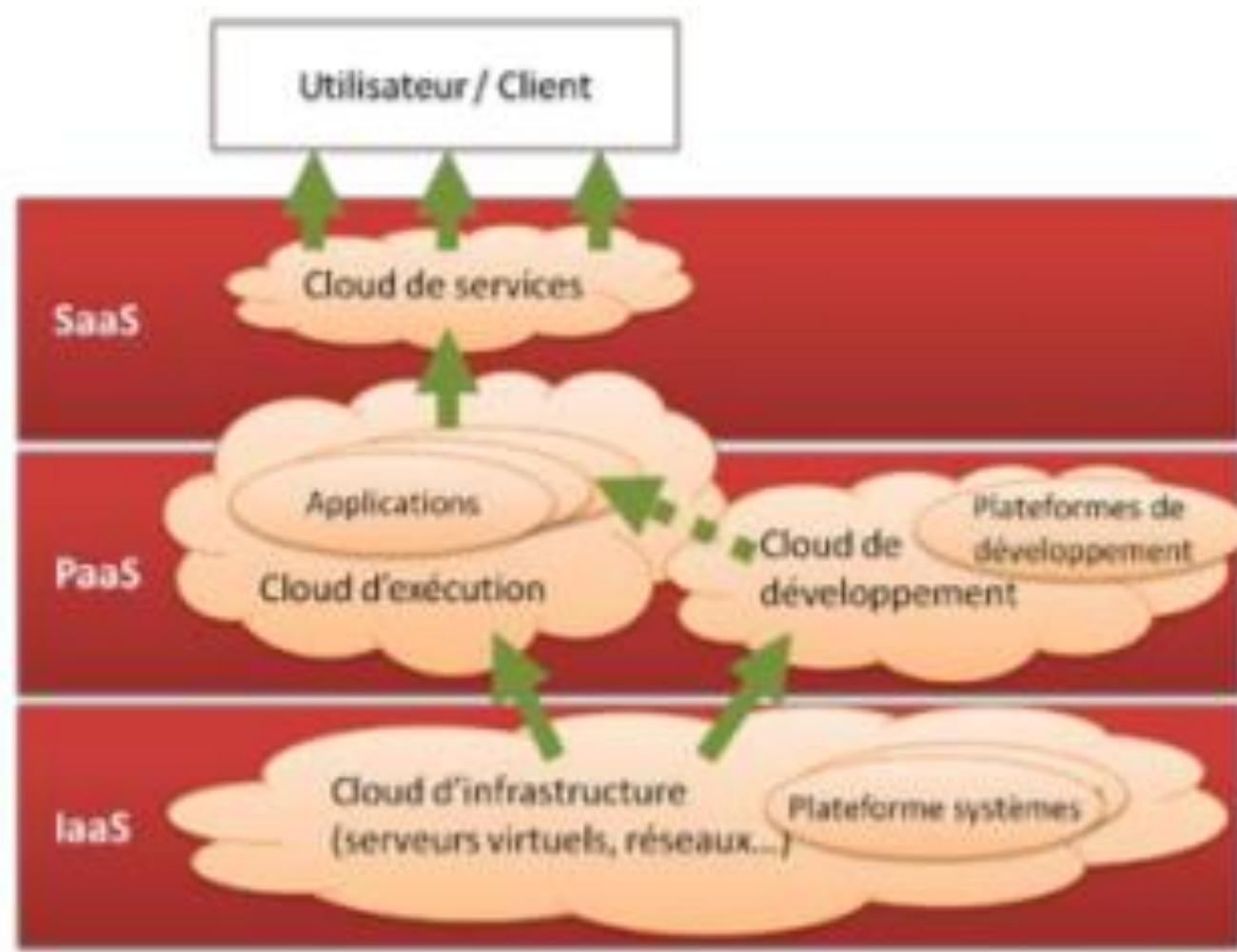
- ▶ l'environnement de développement est hébergé dans le Cloud et est accessible via un navigateur
- ▶ **l'environnement de développement est offert en tant que service**
- ▶ Le **fournisseur de Cloud** propose **un certain nombre d'outils (la plate-forme)** qui permettent à un client (développeur) de créer des applications.
 - L'application fonctionne sur l'infrastructure du fournisseur du Cloud.
 - Le fournisseur gère les outils et le matériel sous-jacent.
- ▶ Exemples : **Openshift de Redhat**, Bluemix d'IBM, Heroku de SalesForce, Google App Engine, Cloud fundry, Caspio,

4.3 Modèles de service: IaaS Infrastructure as a Service



- ▶ Le prestataire de services loue à ses clients
 - les machines (CPU), Les disques, connexions réseau
 - en utilisant la technologie de virtualisation.
- ▶ Sur une machine virtuelle, l'utilisateur accède à un environnement de système d'exploitation standard
 - Il peut installer et configurer dessus toutes les couches dont il a besoin
- ▶ **Le consommateur n'a pas à gérer ou contrôler l'infrastructure du Cloud sous-jacente, mais il a le contrôle sur les systèmes d'exploitation, le stockage,.....**
- ▶ Exemples: **Amazon EC2**, Microsoft Azure, Rackspace, GoGrid, Right Scale, Eucalyptus, OpenStack, OpenNebula, Nimbus, CloudStack,

4. Cloud Computing: Modèles de service



4. Cloud Computing: Modèles de service

SaaS	PaaS	IaaS
CRM		Stockage & Sauvegarde S3
ERP		
SCM		Calcul EC2
CCC	 force.com™ platform as a service 	 Cloud Privés
DCC		
Int.aaS		

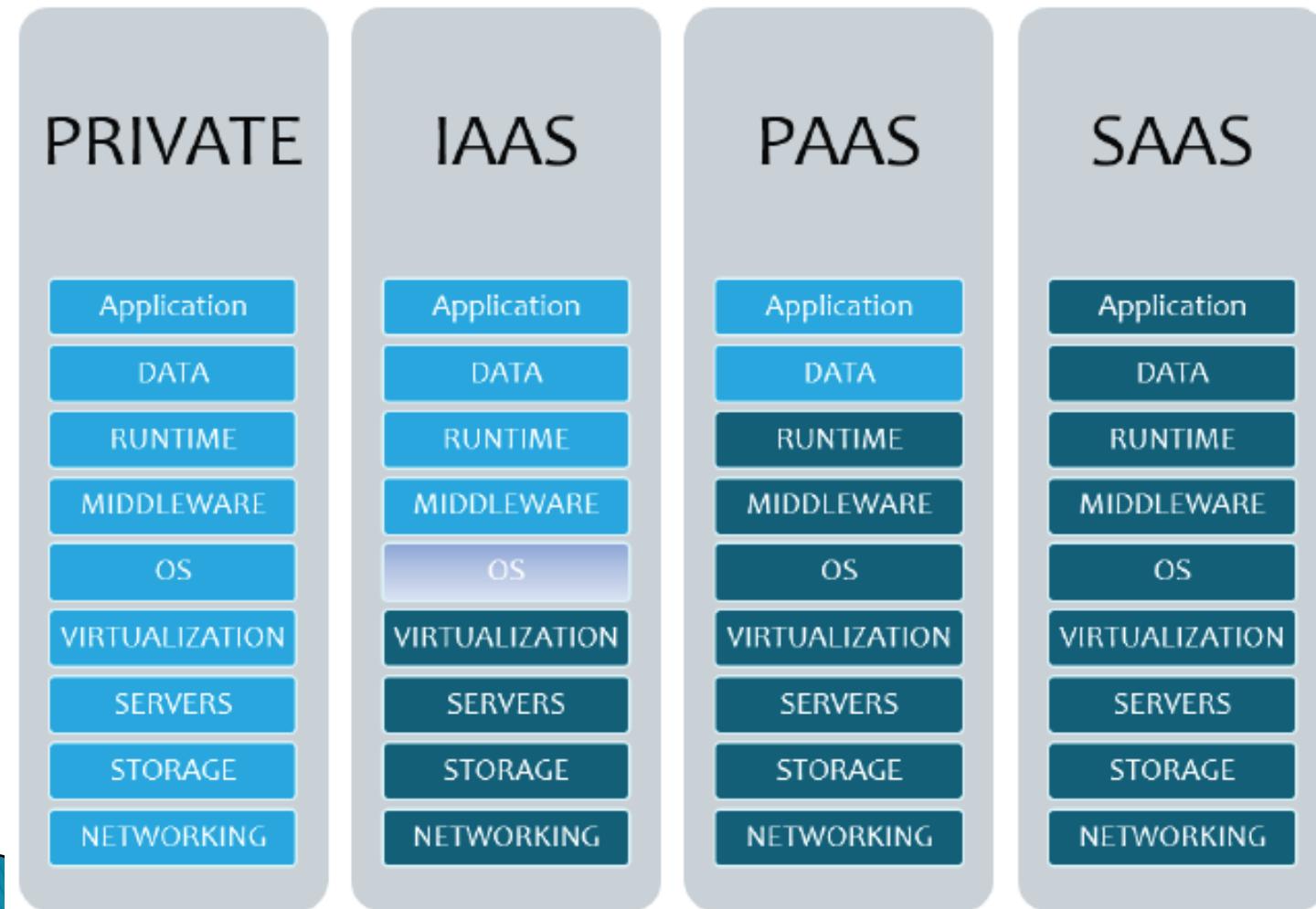
CRM : Customer Relationship Management
ERP : Enterprise Resource Planning
SCM : Supply Chain Management

CCC : Content Communication and Collaboration
DCC : Digital Content Creation
Int .aaS : Integration-as-a-Service

4.4 XaaS: Everything as a Service

CaaS	Communication as a Service
DaaS	Data as a Service Desktop as a Service
DBaaS	DataBase as a Service
DCaaS	DataCenter as a Service
HaaS	Hardware a as Service
MaaS	Management as a Service Monitoring as a Service
NaaS	Network as a Service
RaaS	Resource as a Service
SaaS	Storage as a Service

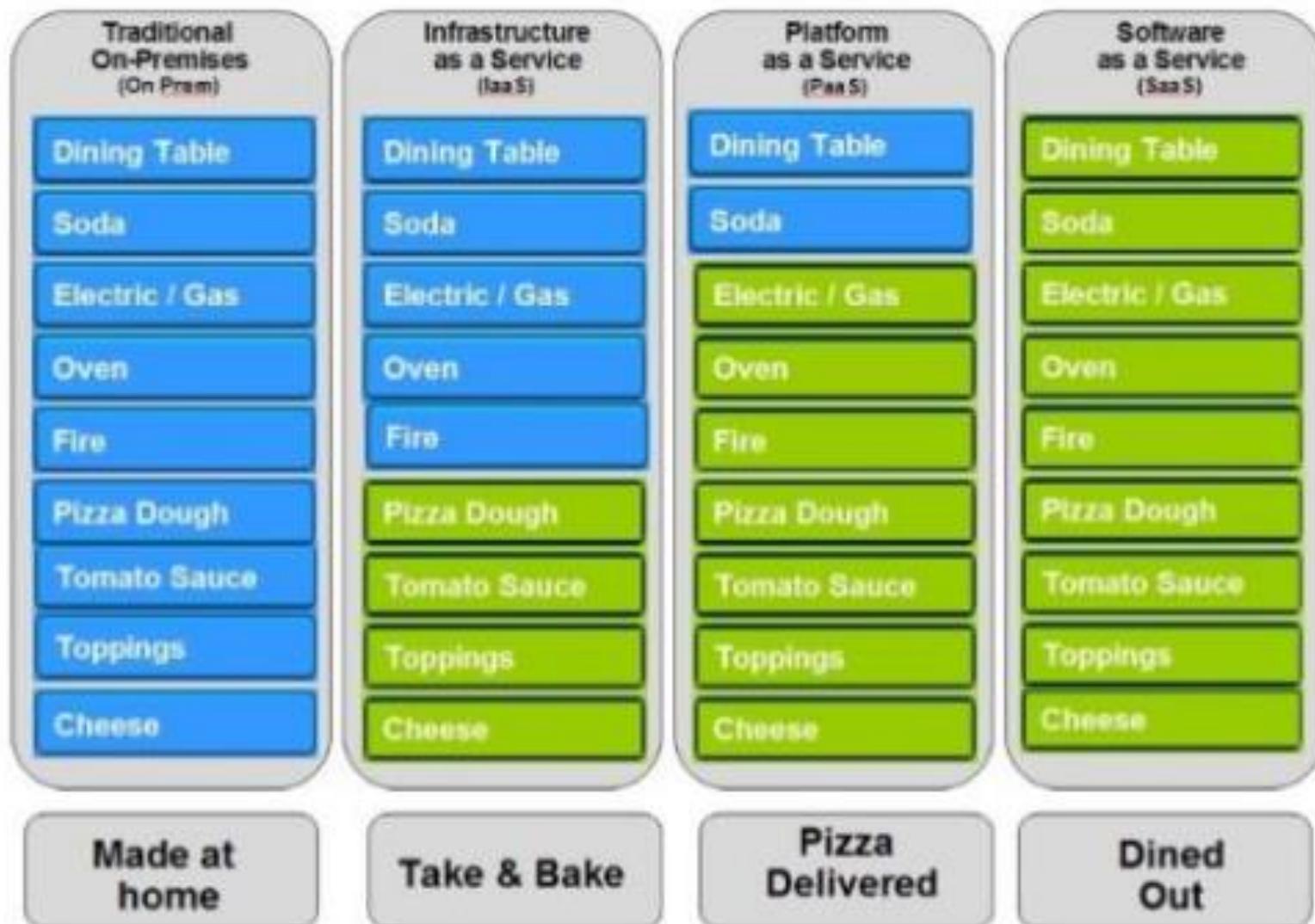
Gouvernance dans le Cloud



By the User

By the Provider

Pizza as a Service



5. Avantages du Cloud Computing

- ▶ **Agilité, flexibilité** : Capacité à adapter ses ressources à la demande, en fonction de l'évolution de son activité
- ▶ **Réduction des coûts** : Mutualisation des ressources physiques, réduction des coûts de possession et d'exploitation
- ▶ **Productivité** : Déploiement plus rapide de nouvelles solutions riches en fonctionnalités
- ▶ **Mobilité** : Capacité à travailler et à collaborer de n'importe où dans les mêmes conditions qu'au sein de l'entreprise. Accès à ses ressources via tout type d'équipement.

5. Avantages du Cloud Computing

- ▶ Petits investissements initiaux et faibles coûts récurrents
 - Pas de matériel, logiciel, périphériques réseau à acheter
- ▶ Normes ouvertes et exigences de conformité
 - La plupart des CC sont basés sur des standards ouverts
 - Les normes ouvertes sont essentielles pour permettre l'évolution continue dans le Cloud (ssh, http,...)
 - MAIS... jusqu'à présent, il n'y a pas de normes adoptées concernant les API Cloud («accès» aux ressources du Cloud)

5. Avantages du Cloud Computing

► Durabilité

- Traditionnellement, les entreprises investissent périodiquement afin de maintenir leurs services informatiques à jour.
- l'objectif est :
 - d'éviter l'échec
 - suivre le rythme des changements commerciaux
- Avec le Cloud Computing, les entreprises comptent sur leurs fournisseurs de services du Cloud pour minimiser les pannes

6.Limites du Cloud Computing

Défis techniques

- ▶ Sécurité
- ▶ Les outils évoluent continuellement
- ▶ Déplacer des données volumineuses coûte cher
- ▶ Qualité de service
- ▶ Dépendance à Internet

6.Limites du Cloud Computing

Défis non-techniques

- ▶ vendor lock-in
 - Les entreprises ne veulent pas devenir dépendantes d'un seul fournisseur de Cloud/ utilisation des normes du Cloud
- ▶ Risques de sécurité
 - Les données transitent sur Internet / les applications fonctionnent en infrastructure partagée/ isolation fournie par la virtualisation
 - Les données et les applications peuvent être accessible par les employés du fournisseur de Cloud
 - le cryptage des données en transit, insister sur la certification de sécurité du fournisseur de Cloud

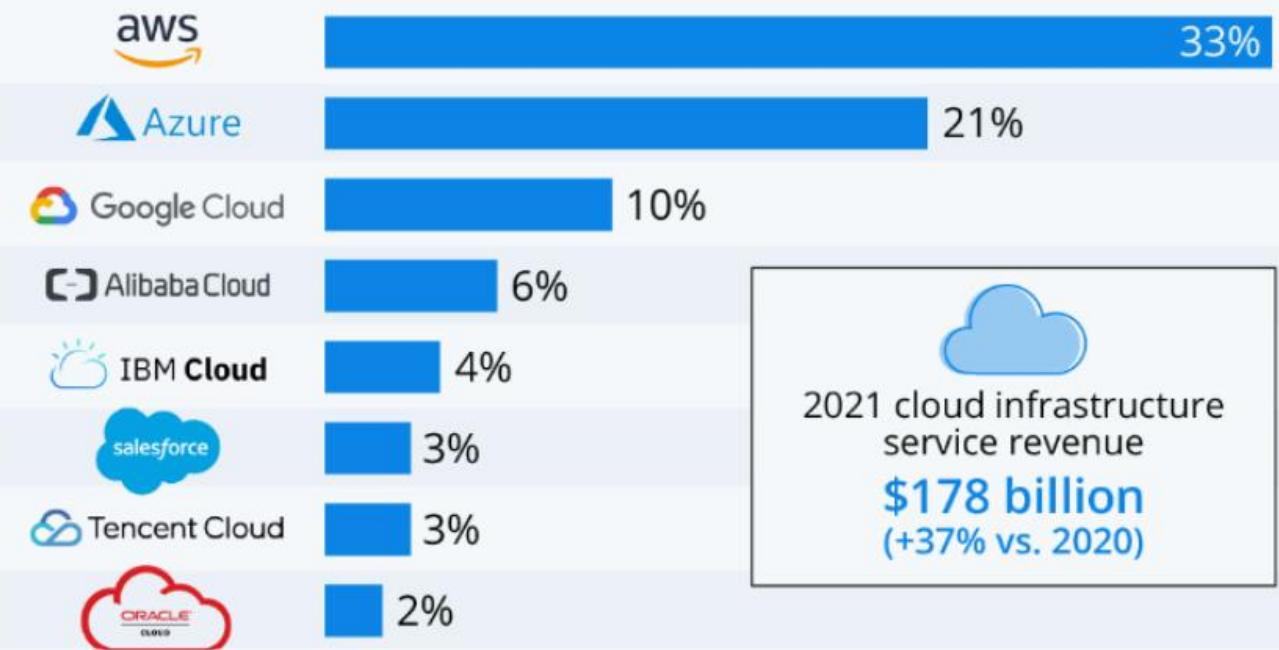
6.Limites du Cloud Computing

- ▶ Accords de Niveau de Service SLA
 - Responsabilité du fournisseur de Cloud si le SLA n'est pas respecté
- ▶ Légal
 - La société doit obéir aux lois nationales sur la protection de la vie privée
 - interdisent souvent que des données quittent le pays
 - les fournisseurs de cloud américains sont soumis au Patriot Act:
 - Le gouvernement américain peut accéder aux données même si elles sont situées en dehors des États-Unis
 - Évitez en choisissant un fournisseur de cloud local

7. Cloud Computing: Marché

Amazon Leads \$180-Billion Cloud Market

Worldwide market share of leading cloud infrastructure service providers in Q4 2021*



* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

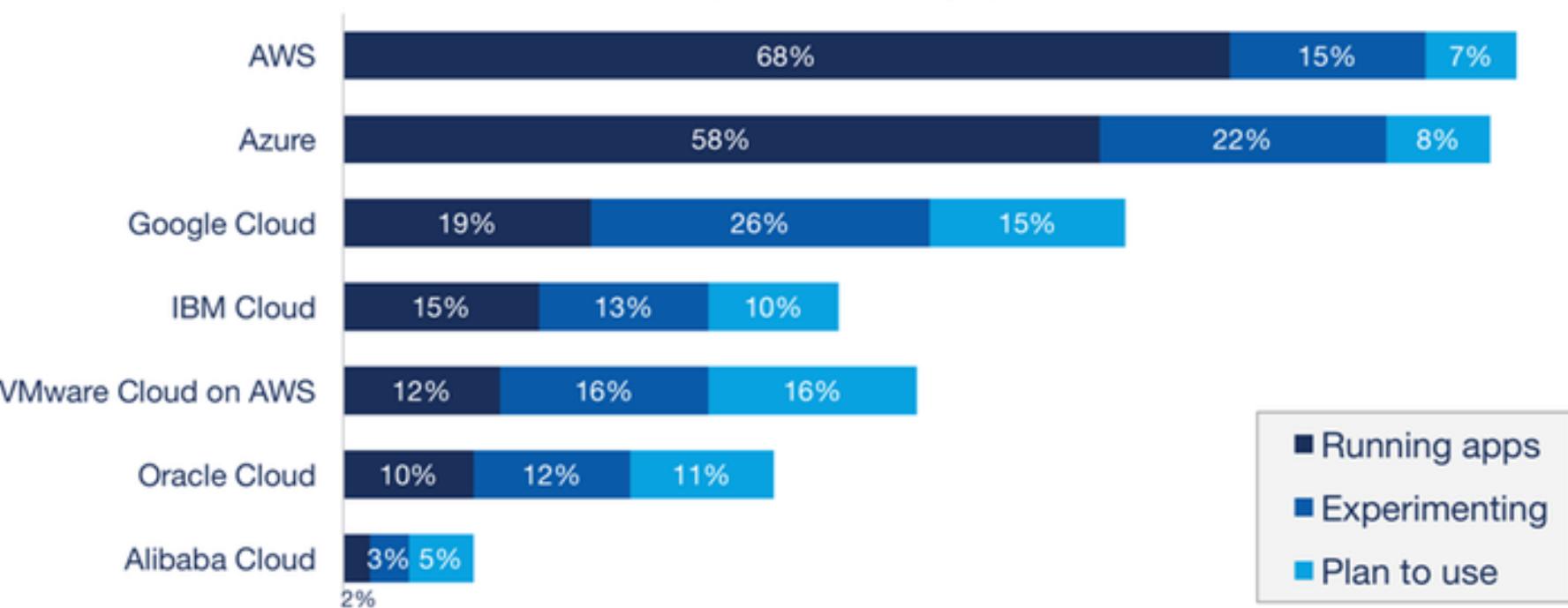
Source: Synergy Research Group



Les fournisseurs cloud du IaaS (Infrastructure as a Service) et du PaaS (Platform as a Service) :

7. Cloud Computing: Marché

Enterprise Public Cloud Adoption
% of Respondents Running Applications



Source: RightScale 2018 State of the Cloud Report

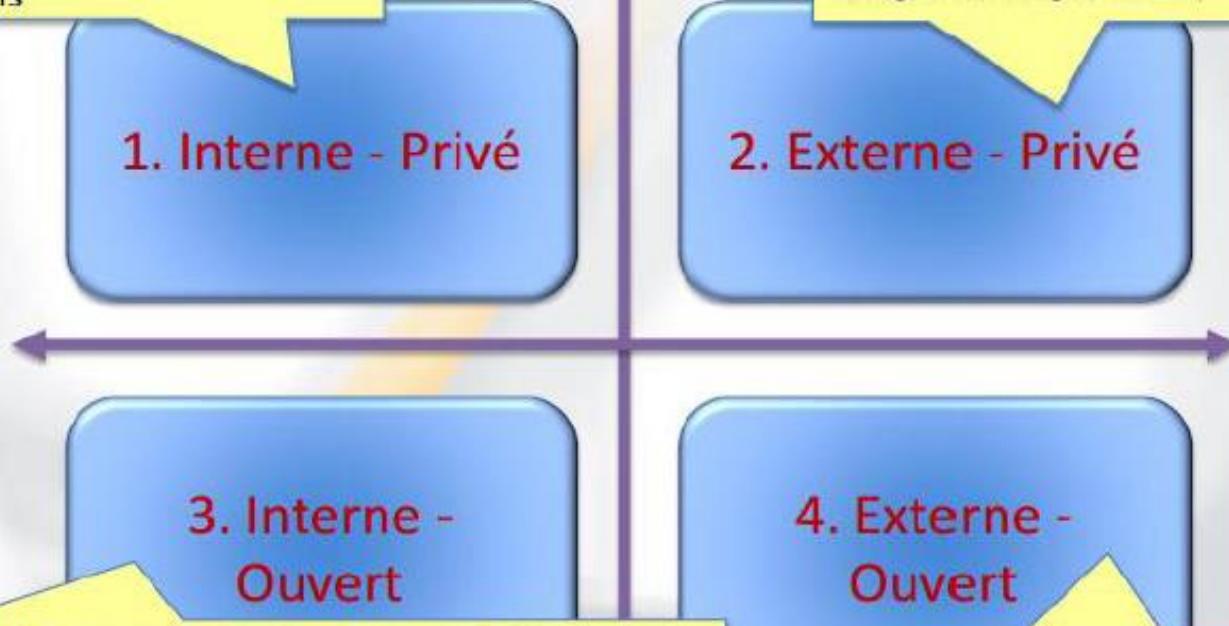
Références Bibliographiques

- ▶ Nabil Abdennadher (Haute école du paysage, d'ingénierie et d'architecture de Genève), Majd Sakr (Carnegie Mellon University) « Introduction to Cloud Computing », 2017
- ▶ « Fondamentaux du Cloud Computing », Cigref (Réseaux de grandes entreprises), 2013

Cloud Privé : une seule et unique organisation loue ou possède son Infrastructure, et la gère uniquement pour ses besoins. Elle ne la partage pas avec d'autres organisations

Cloud communautaire : l'infrastructure est partagée par plusieurs organisations et concerne une communauté spécifique qui partage des intérêts communs (une mission, des exigences de sécurité, des obligations légales, etc)

Gestion interne à l'entreprise



Cloud hybride : composition de deux ou plusieurs nuages internes, communautaires ou publics. Cette plate-forme constitue une entité unique. Les échanges qui s'y produisent s'effectuent grâce à une technologie standard ou propriétaire qui permet la portabilité des données et des applications

à l'entreprise

Cloud Public : un fournisseur possède une infrastructure dont il loue les services à plusieurs entreprises ou groupes industriels.

Amazon Web Service

Amel HAJI

AWS: Amazon Web Service

- ▶ Amazon Web Services (AWS) est un ensemble de services d'infrastructure distants principalement dans les domaines suivants:
- ▶ Catégorie d'infrastructure en tant que service (IaaS), avec certains services dans la plate-forme en tant que service (PaaS)
- ▶ Introduit en 2006/2007, il est considéré comme la première offre réelle de cloud computing.

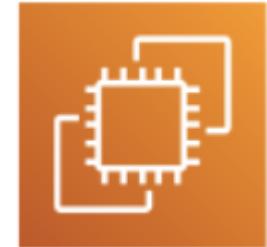
AWS: Amazon Web Service

- ▶ En IaaS, les principaux services offerts par AWS sont
- ▶ Compute: Elastic Compute Cloud EC2
- ▶ Stockage: Simple Storage Service S3
- ▶ Base de données: service de base de données relationnelle RDS
- ▶ Réseau: Virtual Private Cloud VPC

AWS: Amazon Web Service

EC2 : Elastic Compute Cloud

- ▶ Ce service permet de gérer des serveurs sous forme de machines virtuelles dans le cloud
- ▶ Vous avez accès à la ligne de commande,
- ▶ vous pouvez les piloter à distance.
- ▶ Ce service permet de gérer des bases de données managées dans le cloud.



RDS: Relational DataBase

- ▶ On met à votre disposition un serveur de base de données préconfiguré.
- ▶ (pas accès à la ligne de commande//Amazon s'occupe de toute la gestion du serveur)

AWS: Amazon Web Service

S3 : Elastic Compute Cloud

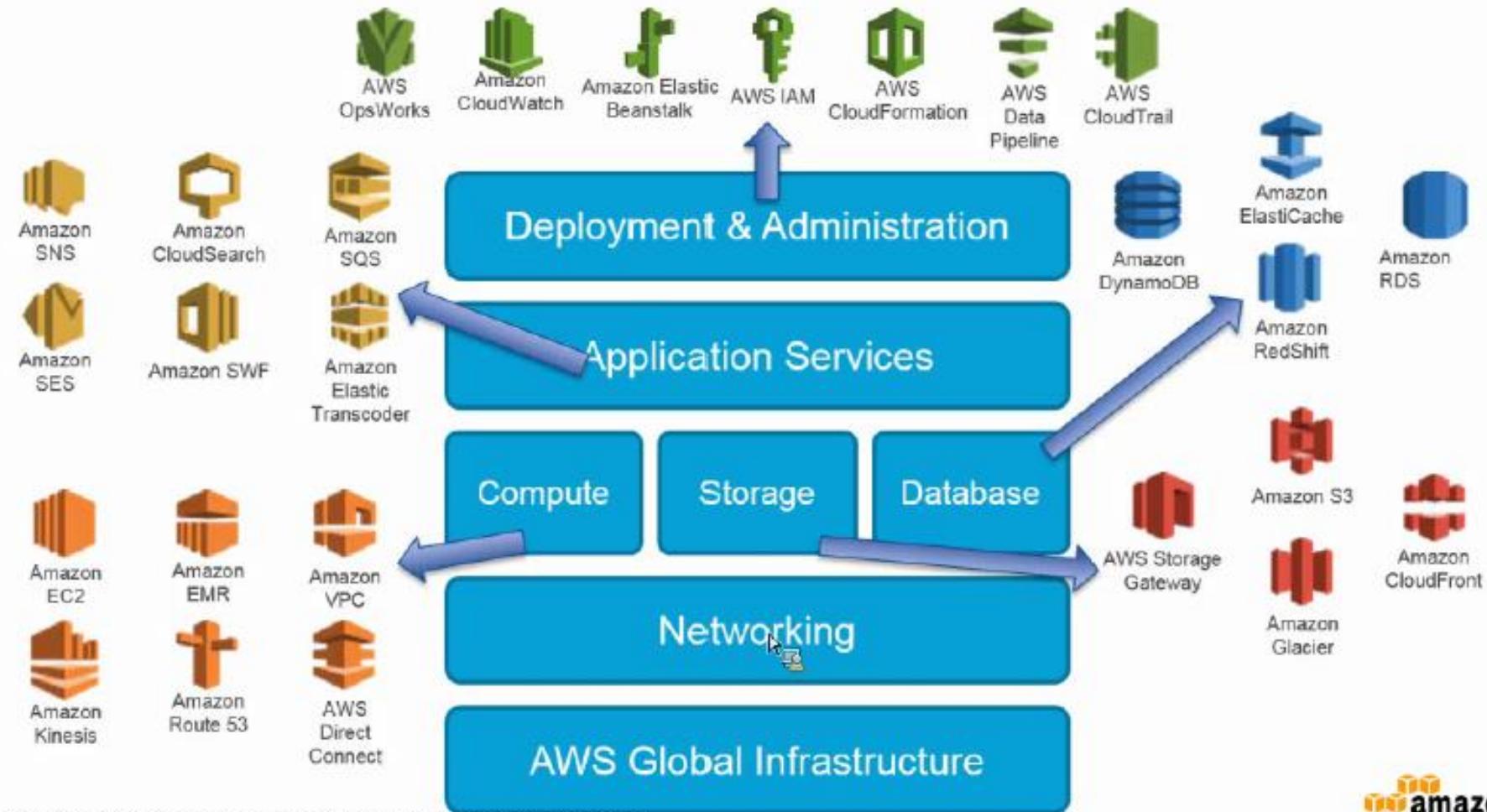
- ▶ C'est un service de stockage et de distribution de fichiers
- ▶ Une sorte d'entrepôt de fichiers à très bas coût qui garantit de ne jamais perdre vos données.
 - Faire télécharger des fichiers sur votre site, y stocker des images
 - permet de gérer des serveurs sous forme de machines virtuelles dans le cloud



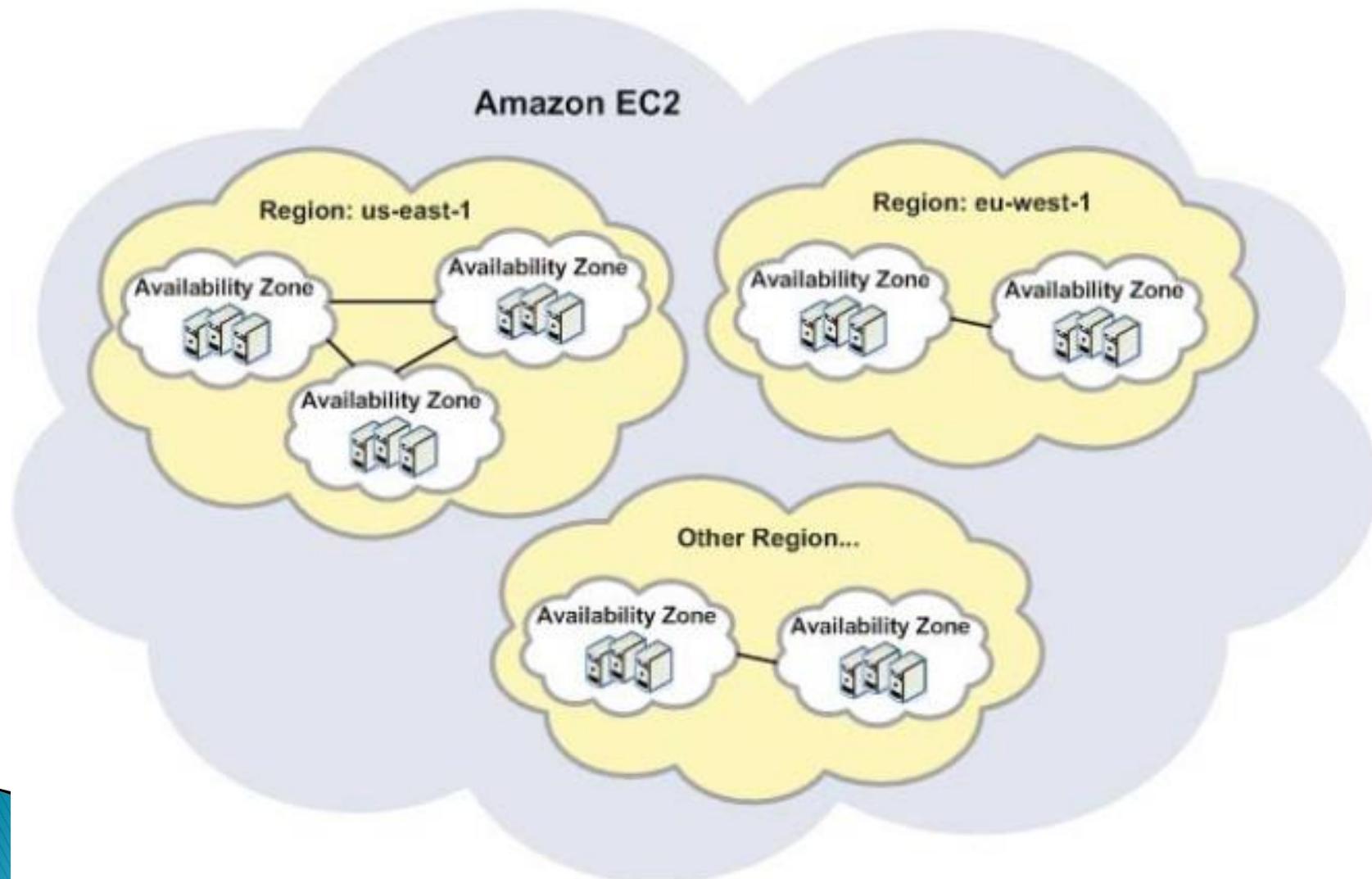
IAM : Gestion des identités

- ▶ AWS IAM (Identity and Access Management): service de sécurité.
 - Définition des règles d'accès des utilisateurs aux services de AWS.

AWS: Amazon Web Service



AWS: Régions et zone de disponibilité



AWS: Régions et zone de disponibilité

- ▶ *Régions et zones de disponibilité « Availability Zone »*
- ▶ Lors de l'allocation d'une ressource cloud (ex: VM)
- ▶ le client AWS
 - peut choisir la région
 - la zone de disponibilité (dans la région)
- ▶ Les **régions** sont réparties dans le monde entier et permettent à un développeur de placer ses application et / ou données dans un pays / une région en particulier...
 - être plus proche de ses clients,
 - afin que les données résident dans une juridiction particulière pour être en conformité avec la réglementations (par exemple, lois sur la confidentialité des données).

AWS: Régions et zone de disponibilité

- ▶ Les **zones de disponibilité** sont des centres de données distincts dans une région.
- ▶ Chaque centre de données possède sa propre infrastructure indépendante pour l'alimentation, refroidissement,...
 - Lorsque une catastrophe se produit, ses effets peuvent se limiter à une seule zone de disponibilité.
- ▶ Un client AWS peut distribuer des copies redondantes de son application sur plusieurs zones de disponibilité
 - protéger contre la défaillance d'une Zone de disponibilité

AWS: Elastic Compute Cloud EC2

- ▶ Amazon Elastic Compute Cloud (EC2) est un service Web qui fournit des calculs redimensionnables et une grande capacité de stockage dans le cloud.
- ▶ Est considéré comme le premier véritable produit de cloud computing
- ▶ Les développeurs peuvent louer des machines virtuelles (appelées Instances EC2).
- ▶ De nombreux types d'instances sont disponibles.

AWS: Elastic Compute Cloud EC2

- ▶ Amazon EC2 présente un environnement informatique virtuel qui permet:
- ▶ d'utiliser une interface Web pour lancer des instances avec une variété de systèmes d'exploitation (images)
 - Ces images d'OS sont regroupées dans Amazon Machine Images (AMI).
- ▶ de charger des instances avec un environnement d'application personnalisé.
- ▶ de gérer les autorisations d'accès au réseau (security group).
- ▶ de réduire le temps requis pour obtenir et démarrer de nouvelles instances de serveur en minutes.
 - permet d'échelonner rapidement la capacité selon les besoins

AWS: Elastic Compute Cloud EC2

Modèles d'instances EC2

- ▶ Amazon propose trois modèles d'instance différents dans leur disponibilité et leur structure de prix

1. Instances à la demande

- ▶ Paiement à l'heure / Commencez et arrêtez à volonté

2. Instances réservées

- ▶ Payer une cotisation initiale annuelle et recevoir une réduction sur la charge horaire / Commencez et arrêtez à volonté

3. Instances ponctuelles

- ▶ Offre pour une capacité EC2 non utilisée
- ▶ Proposer un prix , si le cours du marché est moins cher, obtenir l'instance
- ▶ L'instance se termine automatiquement si le prix proposé devient inférieur au taux du marché actuel

AWS: Elastic Compute Cloud EC2

Types d'instances EC2

Instance Name	Mem (GB)	CPU Capacity	Disk (GB)	Platform	On-Demand Pricing / hour (Linux)
Micro	0.59	Upto 2 ECUs	--	32/64	\$0.02
Small	1.7	1 core - 1 ECU	160	32	\$0.085
Large	7.5	2 cores, 2 ECUs each	850	64	\$0.34
Extra Large	15	4 cores, 2 ECUs each	1690	64	\$0.68
High-Mem Extra Large	17.1	2 cores, 3.25 ECUs each	420	64	\$0.50
High-Mem Double Extra Large	34.2	4 cores, 3.25 ECUs each	850	64	\$1.00
High-Mem Quad Extra Large	68.4	8 cores, 3.25 ECUs each	1690	64	\$2.00
High CPU Medium	1.7	2 cores, 2.5 ECUs each	350	32	\$0.17
High CPU Extra Large	7	8 cores, 2.5 ECUs each	1690	64	\$0.68
Cluster Compute Quad XL	23	33.5 ECUs	1690	64	\$1.30
Cluster GPU Quad XL	22	33.5 ECUS + 2x GPUs	1690	64	\$2.10

AWS: Elastic Compute Cloud EC2 AMI

- ▶ Lors de la création d'une VM, le système d'exploitation est déjà installé.
- ▶ AWS copie une image de la VM (Amazon Machine Image, AMI)
- ▶ Des milliers d'AMI disponibles,
 - certains fournis par Amazon,
 - d'autres par la communauté.
- ▶ Tous les principaux systèmes d'exploitation sont disponible.
- ▶ Une AMI peut également contenir une pile logicielle complète avec middleware et applications.
- ▶ L'utilisateur est capable de créer sa propre AMI.

AWS: Elastic Compute Cloud EC2 AMI

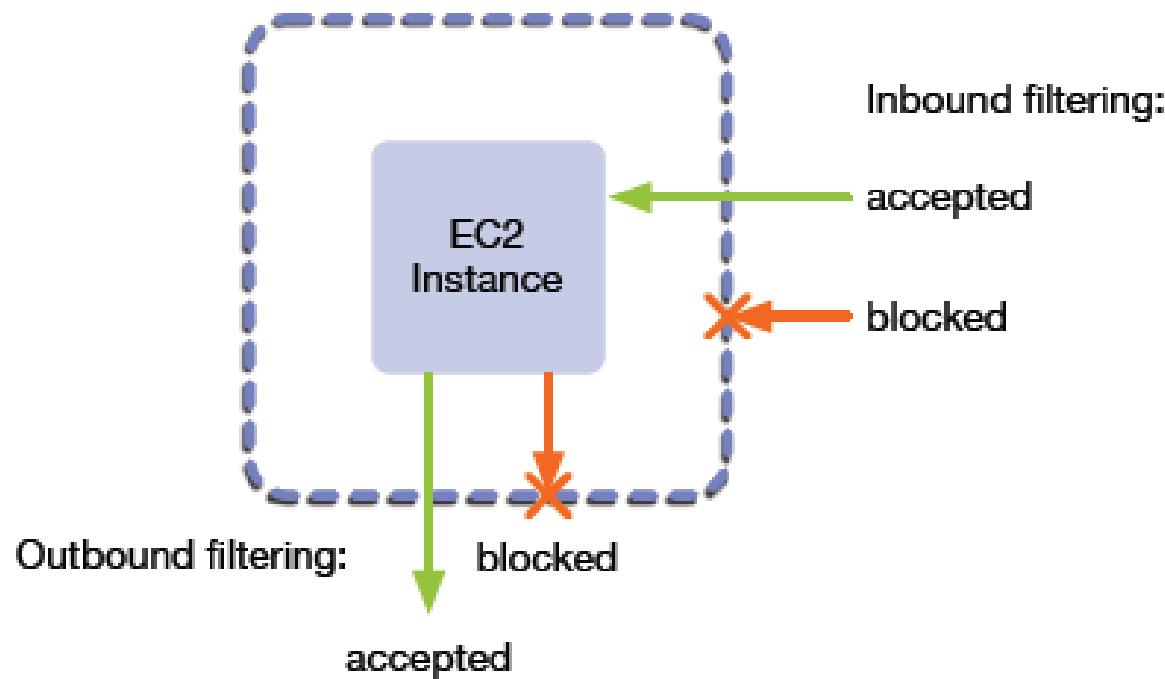
 Red Hat	Red Hat Enterprise Linux 7.1 (HVM), SSD Volume Type - ami-12663b7a	Select	64-bit
	Red Hat Enterprise Linux version 7.1 (HVM), EBS General Purpose (SSD) Volume Type		
	Root device type: ebs Virtualization type: hvm		
 SUSE Linux	SUSE Linux Enterprise Server 12 (HVM), SSD Volume Type - ami-aeb532c6	Select	64-bit
	SUSE Linux Enterprise Server 12 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.		
	Root device type: ebs Virtualization type: hvm		
 Ubuntu	Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8	Select	64-bit
	Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).		
	Root device type: ebs Virtualization type: hvm		
 Windows	Microsoft Windows Server 2012 R2 Base - ami-c9cea0ac	Select	64-bit
	Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]		
	Root device type: ebs Virtualization type: hvm		

AWS: Elastic Compute Cloud EC2

Security Group

- ▶ Une instance EC2 a une adresse IP publique et est donc accessible depuis Internet.
 - Ceci est un risque de sécurité.
- ▶ Chaque instance EC2 est livrée avec un Firewall virtuel.
- ▶ La configuration d'un firewall s'appelle ‘Groupe de Sécurité’ ‘Security Group’.
- ▶ Le Firewall effectue les transferts entrants et sortants de l’instance EC2
 - basé sur les protocoles / numéros de port
 - basé sur les adresses IP
 - Exemple: Pour pouvoir vous connecter à une instance Linux, le port 22(SSH) doit être ouvert.

AWS: Elastic Compute Cloud EC2 Security Group



AWS: Elastic Compute Cloud EC2

Comparaison de Coût Do-it-yourself/AWS

Annual Cost Comparison (100% utilization)				
	Do-It-Yourself	EC2 On-Demand	EC2 Reserved (1 Year Term)	EC2 Reserved (3 Year Term)
Usage Costs	-	\$ 157,680	\$ 75,411	\$ 48,123
Server Hardware	\$ 20,129	-	-	-
Network Hardware	\$ 4,026	-	-	-
Hardware Maintenance	\$ 28,986	-	-	-
Operating System	\$ -	-	-	-
Facility Expense	\$ 131,382	-	-	-
Remote Hands Support	\$ 1,014	-	-	-
Data Transfer Costs	\$ 10,071	\$ 6,138	\$ 6,138	\$ 6,138
TOTAL COST	\$ 195,608.00	\$ 163,818.00	\$ 81,550.00	\$ 54,263.00

Terminologie AWS

AWS term	Generic term	AWS term	Generic term
EC2	IaaS offering	Elastic Block Store (EBS) volume	Virtual disk on a SAN
EC2 Instance	Virtual machine	Instance store volume	Virtual disk co-located with virtual machine
Amazon Machine Image (AMI)	Virtual machine image	S3	Object storage service
Security Group	Firewall configuration	Cloud Watch	Monitoring service
Elastic IP address	Static external IP address		