République Algérienne Démocratique et Populaire
الجـــمهوريـــــــة الجـــزائريـــــــة الديمـــقراطيـــــــة الشـــــعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التـــعليـــم العالـــــي والبحث العلمـــــــي

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université A/Mira de Bejaia
Faculté des sciences exactes
Département Informatique

Mémoire de projet de fin d'étude

Pour l'obtention du diplôme de Master en informatique

Spécialité : Administration et Sécurité des Réseaux Informatiques

Thème

Mise en place d'un Cluster Hadoop de dix (10) postes avec interface d'exécution de Jobs MapReduce à l'Ecole Nationale Supérieure en Science et Technologie de l'Informatique (ENSTI)

Réalisé par :

- BOUMRAOU KAHINA
- KEDJAR HAKIM

Encadré par :

- Dr SEBAA ABDERRAZAK

Membres de Jury :

- *Président :* Dr BOULAHROUZ Djamila
- *Examinateur :* Mme NOUICER Amina
- *Promoteur :* Dr SEBAA Abderrazak

Promotion : 2019 / 2020

Dédicaces

Nous dédions ce travail,

À nos familles, pour leur soutien et leurs sacrifices durant toutes ces années d'étude. Ils nous ont doté d'une éducation digne, leur dévouement leur amour et parfois leur rigueur, ont fait de nous ce que nous sommes aujourd'hui.

À nos pères et mères particulièrement, en guise de gratitude et de reconnaissance, que ce travail soit le meilleur cadeau que nous puissions leur offrir.

À tous nos frères et sœurs, proches et amis qui nous ont encouragé et aidé de près ou de loin.

Remerciements

Nous nous devons de remercier ALLAH le tout puissant pour toute la volonté et le courage qu'il nous a données pour l'achèvement de ce travail.

Nous tenons à remercier en premier lieu notre encadreur, Monsieur le Docteur SEBAA Abderrazak qui a bien voulu accepter de diriger et d'encadrer notre PFE, pour sa patience, ses sacrifices, ses conseils et l'aide qu'il nous a fourni tout au long de l'année, pour ses qualités humaines et scientifiques qui nous ont toujours motivé à faire plus.

Nous remercions également les membres de jury pour l'intérêt qu'ils ont porté à notre modeste travaille en acceptant de l'examiner et de l'enrichir par leurs précieuses propositions.

Nous tenons aussi à remercier chaleureusement Monsieur MAZARI Oussama Sohaib, et Madame NOUICER Amina deux étudiants doctorants encadrés par le même encadreur, qui ont été à nos coté et qui nous ont aidé par leurs conseils et leurs présences auprès de nous, nous leurs souhaitons le bon courage pour la continuation dans leurs projets.

Nous aimerions adresser un remerciement particulier à tout le personnel et à tous les travailleurs de l'Ecole National Supérieurs en Technologie de l'Informatique (ENSTI), qui ont été toujours à notre service, et qui nous ont aidé pour résoudre les soucis techniques rencontrés.

Enfin, nous remercions l'ensemble du personnel du département d'informatique et tout le corps professoral de l'Université Abderrahmane Mira de Bejaia pour la qualité de leur enseignement et pour les valeurs qu'ils nous ont inculquées et tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail.

Résumé

Depuis plus de cinq (05) décennies l'informatique s'est implanté au cœur de nos entreprises, nos hôpitaux, nos ministères, nos foyers ... etc. Cette forte utilisation de l'informatique a engendré de grands volumes de données qui ne sont pas gérable par les logiciels et matériels classiques. Prenons le cas de grandes entreprises comme Google, Microsoft, Facebook et Yahoo ! qui doivent avoir des milliards de données à stocker, gérer et manipuler. Cette perplexité dans la gestion de ces grands volumes de données a donné naissance au terme Big Data. Les quantités de données potentiellement infinies ainsi que les contraintes qui en dérivent posent de nombreux problèmes de stockage et de traitement de ces données très volumineuses en termes de temps et de calcul en utilisant des plateformes dédiées tel que Hadoop qui constitue l'une des meilleures plateformes du Big Data et qui se repose sur son système de fichier distribué HDFS et son paradigme de traitement et de calcul parallèle MapReduce.

Dans ce travail nous aborderons le monde du big data en définissant ces caractéristiques et son architecture. Ensuite, nous parlerons des technologies de haute performance du Big Data et plus précisément le Framework Hadoop.

Dans la partie applicative de notre projet nous avons mis en place un Cluster Hadoop avec une interface d'exécution de Jobs MapReduce.

Mots clés :

Big Data, Hadoop, HDFS, MapReduce, Cluster, Spark.

Abstract

For more than five (05) decades, IT has been established at the heart of our companies, our hospitals, our ministries, our homes ... etc. This heavy use of IT has created large volumes of data that are not manageable by conventional software and hardware. Take the case of large companies like Google, Microsoft, Facebook and Yahoo! who must have billions of data to store, manage and manipulate. This perplexity in the management of these large volumes of data gave birth to the term Big Data. The potentially infinite amounts of data as well as the constraints that derive from it pose many problems of storage and processing of this very large data in terms of time and calculation using dedicated platforms such as Hadoop which is one of the best platforms. of Big Data and which is based on its HDFS distributed file system and its MapReduce parallel processing and computing paradigm.

In this work we will approach the world of big data by defining these characteristics and its architecture. Then, we will talk about high performance Big Data technologies and more specifically the Hadoop Framework.

In the application part of our project, we set up a Hadoop Cluster with an interface for executing MapReduce Jobs.

Keywords:
Big Data, Hadoop, HDFS, MapReduce, Cluster, Spark.

Table des matières

DEDICACES II

REMERCIEMENTS III

RESUME IV

ABSTRACT V

LISTE DES TABLEAUX IX

LISTE DES FIGURES X

LISTE DES ABREVIATIONS XIII

INTRODUCTION GENERALE 15

CHAPITRE 01 : PROBLEMATIQUE ET OBJECTIF DU PROJET 17

1. Contexte du projet 17

2. Problématique 18

3. La solution proposée 18

4. La démarche à suivre 19

CHAPITRE 02 : BIG DATA ET HADOOP 20

Introduction 20

1. Définition de Big Data 20

2. Caractéristiques de Big Data 21

3. Architecture de Big Data 22

4. Domaines d'utilisation de Big Data 24

5. Les principales technologies de Big Data 25

6. Aperçu sur l'écosystème Hadoop 25

Conclusion 26

CHAPITRE 03 : TECHNOLOGIES DE HAUTE PERFORMANCE 27

Introduction 27

1. Hadoop 27

1.1. HDFS (Le système de fichier distribué de Hadoop) 28

1.2. MapReduce 31

1.3. Les éléments complémentaires de l'écosystème Hadoop 33

2. Apache SPARK 36

2.1. Les composants de SPARK 37

2.2. Pourquoi utiliser Hadoop et Spark ensemble ? 38

Conclusion 39

CHAPITRE 04 : INSTALLATION ET CONFIGURATION 40

Introduction 40

1. Environnements 40

1.1. Matériels utilisés 40

1.2. Environnements logiciels 41

2. Installations des environnements 42

2.1. Installation de Ubuntu 16.04.1 LTS 42

2.2. Installation de Java 44

2.3. Installation de Scala 45

2.4. Installation de spark-2.0.2-bin-hadoop2.6 46

2.5. Installation de Hadoop 2.0.6 48

3. Configuration des environnements 55

3.1. Configuration de Hadoop 55

Conclusion 65

CHAPITRE 05 : INTERFACE D'UTILISATION 66

Introduction 66

1. Outils d'implémentation 66

1.1. HTML 66

1.2. CSS 66

1.3. PHP 67

1.4. JavaScript 69

1.5. Script Bash Ubuntu 69

1.6. Sublime Text 70

1.7. Serveur HTTP Apache 2 70

2. Implémentation de l'interface 73

3. Présentation de l'interface réalisée 75

4. Evaluation de performances en exploitant l'interface 79

4.1. Le Job WordCount 79

4.2. Le Job Pi 82

Conclusion 87

CONCLUSION GENERALE 88

REFERENCES 89

Bibliographie 89

Webographie 90

ANNEXES 92

Annexe A : Problèmes rencontrés lors de l'installation de Hadoop 92
Annexe B : Le fichier "sc.sh" utilisé pour la génération du fichier "prenom.txt" 94
Annexe C : Le Cluster Hadoop avant et après mise en place 95

Liste des tableaux

Tableau 1. Affectation des adresses IP des postes 48
Tableau 2. Résultats d'exécutions du Job WordCount 81
Tableau 3. Résulats d'exécution du Job Pi 86

Liste des figures

Figure 1. Caractéristiques du Big Data (les 6 V) 22
Figure 2. Architecture de Big Data 23
Figure 3. Les principales technologies de Big Data 25
Figure 4. Architecture de HDFS 28
Figure 5. Fonctionnement du Secondary NameNode 30
Figure 6. La lecture dans HDFS 30
Figure 7. L'écriture dans HDFS 31
Figure 8. Exemple d'un programme MapReduce (WordCount) 32
Figure 9. Les versions de Hadoop 33
Figure 10. L'écosystème Hadoop 36
Figure 11. Les composants de Spark 38
Figure 12. Utilisation de Spark & Hadoop 38
Figure 13. Création de clé USB Bootable avec Refus 43
Figure 14. Boot Mode 43
Figure 15. Installation de Ubuntu 44
Figure 16. Redémarrage de la machine 44
Figure 17. Installation de Java 44
Figure 18. Version de Java 45
Figure 19. Installation de Scala 45
Figure 20. Version de Scala 45
Figure 21. Extraction de Spark 46
Figure 22. Déplacement de Spark 46
Figure 23. Variable d'environnement Spark 47
Figure 24. Version de Spark 47
Figure 25. Interface graphique de Spark 48
Figure 26. Configuration des adresses IP (le fichier hosts) 49
Figure 27. Installation de SSH 50
Figure 28. Test de connexion SSH 50
Figure 29. Connexion du maître à un esclave avec SSH 51
Figure 30. Extraction de Hadoop 52
Figure 31. Variable d'environnement de Hadoop 53
Figure 32. Variable d'environnement de Java dans Hadoop 54
Figure 33. Le fichier core-site.xml 55

Figure 34. Le fichier hdfs-site.xml 57
Figure 35. Le fichier yarn-site.xml 58
Figure 36. Le fichier mapred-site.xml 59
Figure 37. Le fichier masters.txt 60
Figure 38. Le fichier slaves.txt 60
Figure 39. Formatage du système de fichiers Hadoop 61
Figure 40. Démarrage de DFS 62
Figure 41. Démarrage de YARN 62
Figure 42. Vérification de démarrage des services sur le maître 62
Figure 43. Vérification de démarrage des services sur un esclave 63
Figure 44. Vérification de démarrage de YARN 63
Figure 45. Interface graphique de Hadoop 64
Figure 46. Installation de PHP 67
Figure 47. Téléchargement des bibliothèques PHP 68
Figure 48. Le fichier ''test.php'' 68
Figure 49. Test d'installation de PHP 69
Figure 50. Installation de Sublime Text 3 70
Figure 51. Installation du Serveur Apache2 71
Figure 52. Page d'accueil Apache2 71
Figure 53. Autorisation des permissions pour ''/var/www/html'' 72
Figure 54. Test du serveur local Apache2 72
Figure 55. Installation de MySQL 72
Figure 56. Installation de phpMyAdmin 73
Figure 57. Les fichiers PHP utilisés 74
Figure 58. Les fichiers script utilisés 75
Figure 59. Interface de gestion de nœuds du Cluster 76
Figure 60. Mise Hors service du Poste 03 76
Figure 61. Actualisation du Cluster 77
Figure 62. Vérification de mise Hors service du Poste 03 78
Figure 63. Informations détaillées sur les Nœuds 78
Figure 64. Exécution du Job WordCount avec neufs (09) nœuds 80
Figure 65. Exécution du Job WordCount avec trois (03) nœuds 81
Figure 66. Exécution du Job Pi avec (09) nœuds (nbr de Map = 4 et nbr échantillons par Map = 10^3) 83
Figure 67. Exécution du Job Pi avec (09) nœuds (nbr de Map = 8 et nbr échantillons par Map = 10^6) 84

Figure 68. Exécution du Job Pi avec (03) nœuds (nbr de Map = 4 et nbr échantillons par Map = 10^3) 85

Figure 69. Exécution du Job Pi avec (03) nœuds (nbr de Map = 8 et nbr échantillons par Map = 106) 86

Figure 70. Problèmes installation de Hadoop 92

Figure 71. Le dossier ''current'' 93

Figure 72. Le fichier sc.sh 94

Figure 73. Le Cluster Hadoop au premier jour du projet 95

Figure 74. Le Cluster Hadoop au dernier jour du projet 95

Liste des abréviations

IT	Technologies de l'Information
IBM	International Business Machines
Hadoop	High-Availability Distributed Object-Oriented Platform
HDFS	Hadoop Distributed File System
LTS	Long Term support
ACM	Association for Computing Machinery
IDC	International Data Corporation
CISCO	Commercial & Industrial Security Corporation
IP	Internet Protocol
SGBDR	Systèmes de Gestion de Bases de Données Relationnelles
YARN	Yet Another Resource Negotiator
OS	Operating System
FsImage	File System Image
SQL	Structured Query Language
CLI	Command Line Interface
RAM	Random Access Memory
API	Application Programming Interface
RDD	Resilient Distributed Dataset
ML	Machine Learning
OpenSSH	Open Secure Shell
JDK	Java Development Kit
JRE	Runtime Environment
RCP	Remote Copy Protocol
FTP	File Transfer Protocol
SFTP	Secured File Transfer Protocol

ISO	International Organization for Standardization
JPS	Java Process Monitoring Tool
URL	Uniform Resource Locator
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
PHP	Hypertexte Preprocessor
Bash	Bourne Again Shell
HTTP	Hypertext Transfer Protocol
WWW	World Wide Web
LAMP	Linux Apache MySQL PHP
CPU	Central Processing Unit

Introduction générale

Introduction générale

Notre monde est actuellement confronté à une explosion importante de données. Google reçoit 40 000 requêtes chaque seconde, Chaque minute, environ 350 000 tweets postés, 200 millions de mails envoyés, 15 millions de SMS, 25 heures de vidéos mises en ligne sur YouTube, 600 000 commentaires écrits sur Facebook, 14 000 nouveaux utilisateurs de Smartphone sont enregistrés ...etc. Des flots d'octets, un océan de données, un déluge d'informations ... !

Le développement et l'accès à ces données a conduit à l'apparition du terme Big Data qui possède ses origines dans le Data Science et le Cloud Computing. Ce phénomène impacte en particulier les entreprises qui sont amenées à manipuler des Téraoctets voir des Pétaoctets de données nécessitant une infrastructure spécifique pour leur création, leur stockage, leur traitement, leur analyse et leur récupération. En d'autres termes, il s'agit du développement en temps réel d'une masse de données volumineuse qui dépasse la capacité des outils de traitement et d'analyse traditionnels (bases de données relationnelles, ... etc.).

Les entreprises sont habituées à utiliser les SGBD traditionnelles pour stocker leurs données structurées sur une seule machine, et lorsqu'elles font face à des problèmes de performance ils achètent du matériel très coûteux comme la mémoire, des processeurs plus performants et cela devient très vite pesant financièrement sur l'entreprise sans une amélioration significative des performances. L'approche SGBDR (Systèmes de Gestion de Bases de Données Relationnelles) ne respecte pas les exigences des entreprises du Web 2.0 telles que Google, Amazon, Yahoo, Facebook, Tweeter et LinkedIn. Ces sociétés gèrent une grande quantité de données. Ces données générées sont non seulement volumineuses, mais ont une vélocité importante, et aussi diversifiées, ce qui a conduit à l'apparition du terme Big Data. Le Big Data impose de nouveaux défis à la manière traditionnelle de traiter les données en utilisant les SGBDR. Un nouveau mouvement de gestion des données est né appelé NoSQL qui est l'abréviation de Not Only SQL. C'est une approche différente des SGBDR qui offre une mise à l'échelle linéaire. Les bases de données NoSQL peuvent traiter différents types de données, qui ne nécessitent pas un schéma strict. Le NoSQL a différente catégorisation pour répondre aux différents besoins de stockage du Big Data.

Ainsi des outils puissants sont requis pour stocker et exploiter ces données en expansion quotidienne dans le but de soumettre un traitement simple et fiable des données récoltées des utilisateurs. Des résultats rapides et de bonne qualité sont attendus. C'est dans ce contexte qu'apparait Apache Hadoop. Hadoop est l'une des solutions technologiques du Big Data qui permet de stocker, traiter et analyser cet océan de données en utilisant le traitement parallèle distribué sur des serveurs économiques standard qui stockent et traitent les données.

On s'intéresse en partie théorique dans ce projet à présenter les Big Data et les différentes solutions technologiques qui sont utilisées pour remédier aux problèmes de stockage et de traitements de grande quantité de données. Et en partie pratique nous allons Mettre en place un Cluster Hadoop

Multi-Nœuds sur des machines physique, et de créer une interface d'exécution de Jobs, enfin nous allons terminer par tester les performances du Cluster. Ce mémoire est structuré comme suit :

- Chapitre 1 : Présentation générale du contexte de notre projet la problématique que nous traiterons ;

- Chapitre 2 : Présentation et définition de Big Data et Hadoop ;

- Chapitre 3 : Présentation de technologies de haute performance utilisée par Big Data ;

- Chapitre 4 : Présentation des différents outils et environnements utilisé pour la mise en œuvre du Cluster Hadoop, ainsi que toutes les étapes d'installation et de configuration ;

- Chapitre 5 : Présentation de la solution proposée pour la réalisation de l'interface d'exécution de Jobs et des tests de performance menée sur le Cluster.

Chapitre 01 :

Problématique et objectif du projet

Chapitre 01 : Problématique et objectif du projet

1. Contexte du projet

Le développement d'internet et la multiplication des objets connectés à travers le monde s'accompagnent d'une croissance exponentielle des données créées sur internet. La multiplication des moyens de communication et d'échange n'y est pas étrangère. En effet, les différents écrans nous suivent partout, tout au long de la journée. En 2011, il y avait un peu moins de 9 milliards de terminaux connectés dans le monde, ce chiffre devrait atteindre les 24 milliards en 2020 selon une étude de Valtech (agence de marketing numérique et technologique) *[1]*. Outre les smartphones, tablettes et télévision connectées, les nouveaux objets connectés, tels que les voitures, les appareils électroménagers ou encore les montres intelligentes qui déferlent sur le marché devraient remonter une quantité phénoménale d'informations dans les années à venir.

Selon certaines études de EMC (Digital Transformation – IT & Workforce Solutions acheté par DELL Technologies actuellement) *[Web1]*, le monde a manipulé en 2012 plus de 2,8 Zettaoctets d'informations soit 2,8 milliards de Gigaoctet. Ce chiffre est colossal, mais le plus intéressant dans cette étude est que seul 0,5% de ces 2,8 Zettaoctets ont été analysés d'une manière ou d'une autre alors que l'étude estime que 25% d'entre elles représentent une valeur potentielle pour des entreprises.

Les informations disponibles sur internet ne sont plus seulement volumineuses, elles sont également très diverses, cela peut passer par des vidéos, de la musique, des photos... etc. Elles ne sont pas structurées où elles ne se présentent pas sous la forme de lignes et de colonnes comme toute les autres données traditionnelles. Ces données doivent donc être structurées avant d'être analysées et exploitées par les technologies actuelles. À titre d'exemple, chaque jour produit plus de vidéos que les cinquante premières années de la télévision *[2]*, cela représente donc une masse d'informations à structurer gigantesque.

À l'heure actuelle, les humains sont les principaux responsables de ce volume d'informations. En effet, toutes leurs interactions avec les nouvelles technologies génèrent des données : téléchargement d'un fichier, consultation d'une vidéo, coup de téléphone, envoi de SMS, utilisation de GPS... etc.

L'ensemble de ces milliards de données, leurs variétés ainsi que la vitesse de leur traitement, c'est ce que on appelle communément les « Big data ». Les premières entreprises à avoir compris leur intérêt sont les géants du web actuel tels que Google, Yahoo, Microsoft, Facebook ou bien Amazon.

La volonté de vouloir gérer une quantité très élevée d'informations, à pousser ces grandes entreprises à apprendre à maitriser ces Big Data, car les outils et méthodes traditionnelles ne leur suffisaient plus, et ont pour cela développé des technologies en interne qui sont désormais, pour la plupart, disponibles dans des versions libres et gratuites.

Ainsi l'objet de ce projet est de cerner le concept du Big Data, dans un premier temps, les mégadonnées sont caractérisées au travers du modèle des 3V étendu ensuite au 6V, la problématique de ces mégadonnées est distinguée de celle de l'informatique décisionnelle.

On s'intéresse à deux grandes problématiques associées aux mégadonnées, d'une part leur stockage, les techniques traditionnelles de stockage de type bases de données relationnelles ne permettant pas de stocker de telles quantités de données, et d'autre part leur exploitation, l'analyse de ces mégadonnées dans des temps raisonnables. En effet, les mégadonnées s'accompagnent principalement du développement d'applications à visée analytique, qui consiste à traiter des données pour en tirer de la connaissance, ces analyses sont généralement appelées Big Analytics, ou Analytique ou encore broyage de données, reposant généralement sur des méthodes de calcul distribué.

2. Problématique

Ces dernières années, nous sommes confrontés à une explosion de données structurées ou non structurées produites massivement par les différentes sources de données numériques. D'une part les applications qui génèrent des données issues des logs, des réseaux de capteurs, des traces de GPS, etc., et d'autre part, les utilisateurs produisent beaucoup de données telles que des photographies, des vidéos et des musiques. Selon IBM (...), chaque heure, 2.5 trillions d'octets de données sont générées. Selon les prévisions faites, d'ici la fin 2020 cette croissance sera supérieure à 40 Zettaoctets, alors qu'un Zettaoctet de données numériques, seulement, ont été générées de 1940 (début de l'informatique) à 2010. Beaucoup de concepts « inséparables » dominent actuellement le marché de l'IT : « Cloud Computing », « Big Data », « NoSQL » ou « MapReduce ».

De rudes contraintes opposent les différents chercheurs dans le domaine, quant au stockage et à l'analyse de ces masses de données. Les prévisions de taux de croissance des volumes de données traitées dépassent les limites des technologies traditionnelles à savoir les bases de données relationnelles ou les Datawarehouses. On parle de Pétaoctet (billiard d'octets 1015), voire d'Exaoctet (trilliard d'octets 1018) et encore le Zettaoctet (1021) ou le Yottaoctet (1024).

Les problématiques traitées dans ce projet sont donc : (1) Comment prendre en charge l'accroissement rapide des volumes de données géographiquement éloignées et comment gérer cette puissante montée en charge. (2) Quel sont les technologies et les modèles de programmation proposées pour pallier ces différents problèmes engendrés par ce déluge de données ?

3. La solution proposée

Cette révolution scientifique qui envahisse le monde de l'information et l'Internet a imposé aux différents chercheurs depuis quelques années, de nouveaux défis les a poussés à concevoir de nouvelles technologies pour contenir, traiter et analyser ces volumes énormes de données. Plusieurs modèles de programmation parallèle et systèmes de gestion de fichiers distribués ont émergé,

principalement, le Framework Hadoop se place comme la solution la plus répandue dans le marché informatique.

Hadoop est un environnement d'exécution distribuée, performant et scalable, il propose un système de stockage distribué via son système de fichier HDFS (Hadoop Distributed File System) et un système d'analyse et traitement de données basé sur le modèle de programmation MapReduce pour réaliser des traitements parallèles et distribués sur des gros volumes de données.

Notre projet de fin d'étude a pour but d'étudier les méthodes et les technologies du Big Data, nous nous intéresserons particulièrement aux technologies Hadoop avec ses composantes HDFS et MapReduce.

La partie applicative, va consister à un déploiement de Hadoop avec tous ses composants dans un environnement distribué, avec l'installation et la configuration d'un cluster de dix (10) machines, et l'exécution de plusieurs jobs MapReduce sur des fichiers HDFS et création d'une interface de gestion et d'exécution de Jobs. L'installation et la configuration de Hadoop a été effectuée sur des machines physique avec Ubuntu 16.04.1 LTS comme système d'exploitation.

La familiarisation avec l'environnement Hadoop et la maitrise de nouvelles technologies, MapReduce et HDFS d'un côté, et la mise en œuvre d'une interface de gestion et d'exécution de Jobs MapReduce d'un autre côté, sont les objectifs principaux ciblés par notre projet.

4. La démarche à suivre

Pour la réalisation de notre projet nous avons commencé par une présentation du contexte du projet, la problématique et la solution adoptée et l'objectif de ce sujet.

Après nous passerons à la définition, la présentation des architectures et les domaines d'utilisation de Big Data et Hadoop.

Ensuite nous allons des technologies de haute performance où on a présenté le Framework Hadoop, ses principaux composants HDFS et sa propre version de MapReduce ainsi que ces éléments complémentaires, on aussi fait une brève présentation d'Apache Spark.

Dans la dernière partie, nous aborderons la partie technique qui consiste à l'installation et la configuration des environnements sur toute les machines, et aux tests d'ensemble des éléments logiciels nécessaires pour la mise en œuvre de la solution.

Chapitre 02 :

Big Data et Hadoop

Chapitre 02 : Big Data et Hadoop

Introduction

De nos jours, l'humanité produit chaque année un volume d'informations numériques de l'ordre de trilliard d'octets. Soit presque autant d'octets qu'il existe d'étoiles dans l'Univers ! En 2010 déjà, Eric Schmidt, le patron de Google, estimait que "tous les deux jours, nous produisons autant d'informations que nous en avons générées depuis l'aube de la civilisation jusqu'en 2003".

Pour faire face à l'explosion du volume de ces données, on parle actuellement de Pétaoctet (billiard d'octets) voir de Zettaoctet (trilliard d'octets) et aussi face à la grande variété des données (image, texte, web, ... etc.) un nouveau domaine technologique a vu le jour : « le Big Data » inventé par les géants du web, au premier rang comme Yahoo, Google et Facebook, qui ont été les tous premiers à déployer ce type de technologie.

Dans l'aspect pratique du Big Data a été implémenter à travers une panoplie de Framework dont « Hadoop » est l'un d'entre eux. Ce concept apporte une architecture distribuée et scalable pour le traitement et le stockage de données. Ce nouveau paradigme a pour principal objectif l'amélioration des performances et l'augmentation de la vitesse d'exécution des requêtes et des traitements.

Dans ce chapitre nous présenterons le terme Big Data, ses caractéristiques, son architecture ainsi que les domaines de son utilisation, enfin on termine par un aperçu sur l'écosystème Hadoop.

1. Définition de Big Data

L'explosion quantitative des données numériques a obligé les chercheurs à trouver de nouvelles manières de voir et d'analyser le monde. Il s'agit de découvrir de nouveaux ordres de grandeur concernant la capture, la recherche, le partage, le stockage, l'analyse et la présentation des données. Ainsi est né le « Big Data ». Il s'agit d'un concept permettant de stocker un nombre indicible d'informations sur une base numérique. Selon les archives de la bibliothèque numérique de l'Association for Computing Machinery (ou ACM) dans des articles scientifiques concernant les défis technologiques à relever pour visualiser les « grands ensembles de données », cette appellation est apparue en octobre 1997 *[Web2]*.

Big data, littérairement les grosses données, est une expression anglophone utilisée pour désigner des ensembles de données qui deviennent tellement volumineux qu'ils 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 très rapidement de capter, traiter et analyser de larges quantités et contenus hétérogènes et changeants, et d'en extraire les informations pertinentes à un coût accessible *[Web3]*.

2. Caractéristiques de Big Data

Le Big Data regroupe une famille d'outils qui répondent à une triple problématique : c'est la règle dite des trois « V ». Il s'agit notamment d'un Volume de données considérable à traiter, une grande Variété d'informations (non-structurées, organisées, Open...), et un certain niveau de Vélocité à atteindre, autrement dit de fréquence de création, collecte et partage de ces données *[Web2]*.

a. Volume : Le Big Data est associé à un volume de données vertigineux, se situant actuellement entre quelques dizaines de Téraoctets et plusieurs Pétaoctets en un seul jeu de données. Les entreprises et tous les secteurs d'activités confondus, devront trouver des moyens pour gérer le volume de données en constante augmentation qui est créé quotidiennement. Les catalogues de plus de 10 millions de produits sont devenus la règle plutôt que l'exception.

Voici quelques chiffres pour illustrer ce phénomène :

- 90% des données actuelles ont été créées dans les deux dernières années seulement ;
- Twitter comme exemple, génère 7 To de données chaque jour.

b. Vitesse : La vitesse décrit la fréquence à laquelle les données sont générées, capturées et partagées. Les entreprises doivent appréhender la vitesse non seulement en termes de création de données, mais aussi sur le plan de leur traitement, de leur analyse et de leur restitution à l'utilisateur en respectant les exigences des applications en temps réel.

c. Variété : La croissance de la variété des données est la conséquence des nouvelles données multi structurelles et de l'expansion des types de données provenant de différentes sources hétérogènes. Aujourd'hui, on trouve des capteurs d'informations aussi bien dans les appareils électroménagers, les trains, les automobiles ou les avions, qui produisent des informations très variées.

Ces nouvelles données dites non-structurées sont variées :

- Des photos ;
- Des mails (avec l'analyse sémantique de leur contenu) ;
- Les données issues des réseaux sociaux (commentaires et avis des internautes sur Facebook ou Twitter par exemple) *[3]*.

Au fur et à mesure que le domaine du big data gagnait en maturité, ces 3 "V" ont été complétés par d'autres, parmi lesquels la Véracité des données qui peuvent être incomplètes, ou erronées, aussi la Valeur des données qui réside dans les conceptions et les connaissances qui découlent des analyses de big data, enfin la Variabilité des données dont la signification change constamment.

d. Véracité : La véracité (ou validité) des données correspond à la confiance que ces données inspirent aux décideurs. Si les utilisateurs de ces données doutent de leur qualité ou de leur pertinence, il devient difficile d'y investir davantage. Le nombre grandissant de fausses informations et de faux avis de consommateurs publiés sur le web est un bon exemple de données externes peu fiables dont les entreprises doivent pouvoir se prémunir.

e. Valeur : Cette dimension s'attache à un aspect étonnant de ces données. Une donnée prise isolément peut ne pas avoir grand intérêt, mais prise dans un ensemble plus large, elle prend tout son sens. C'est le cas par exemple d'un tweet portant sur un événement d'actualité. Le contenu du tweet en lui-même peut ne pas avoir grande importance, en revanche si le tweet est repris et commenté des milliers de fois, c'est le nombre de messages associés à ce tweet qui devient une information de valeur (une opinion partagée par plusieurs, un sujet qui fait consensus ou polémique...).

f. Variabilité : La variabilité fait référence à des données dont la signification change constamment. C'est particulièrement le cas lorsque la collecte de données repose sur le traitement de la langue [*Web4*].

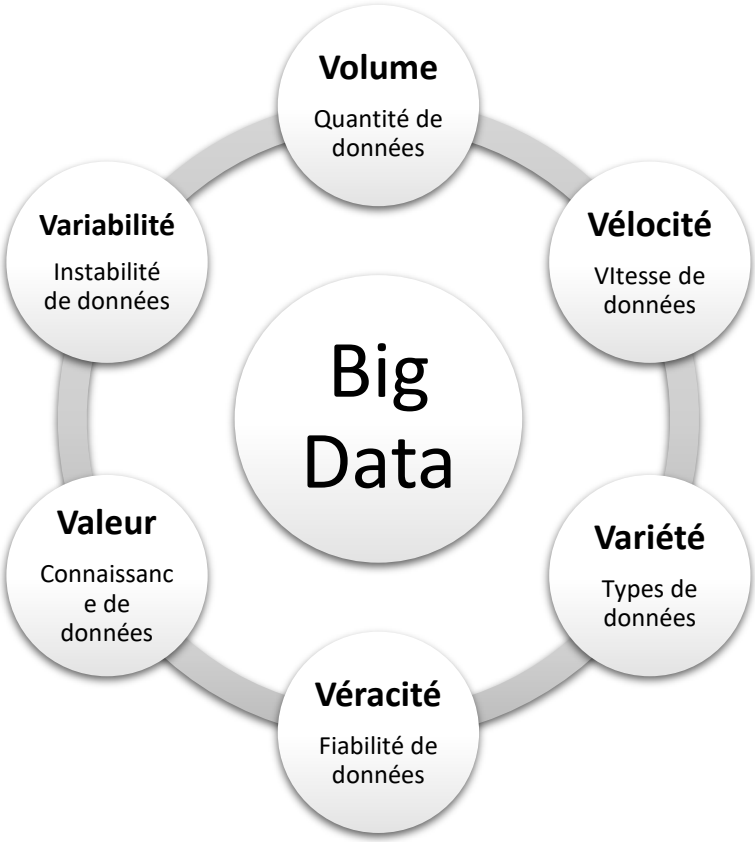

Figure 1. Caractéristiques du Big Data (les 6 V)

Ces caractéristiques illustrées par les six (06) « V », sont les principes définissant le Big Data. Avant tout, il s'agit d'un changement d'orientation sur l'utilisation de la donnée. En somme, le point clé du Big Data est de donner un sens à ces grosses données et pour cela, il faut les analyser *[3]*.

3. Architecture de Big Data

Avant de commencer avec le Big Data, il faut s'assurer que tous les composants essentiels de l'architecture pour analyser tous les aspects d'une grande quantité de données sont en place. Sans cette configuration correcte, on aura du mal à gérer une telle masse de données. Une architecture d'un système Big Data devrait pouvoir consommer une myriade de sources de données de manière rapide et peu coûteuse. Il doit également comprendre les couches suivantes : sources de données, couche d'ingestion, couche de visualisation, couche de gestion de la plateforme Hadoop, couche de stockage Hadoop, couche d'infrastructure Hadoop, couche de sécurité et couche de surveillance *[4]*.

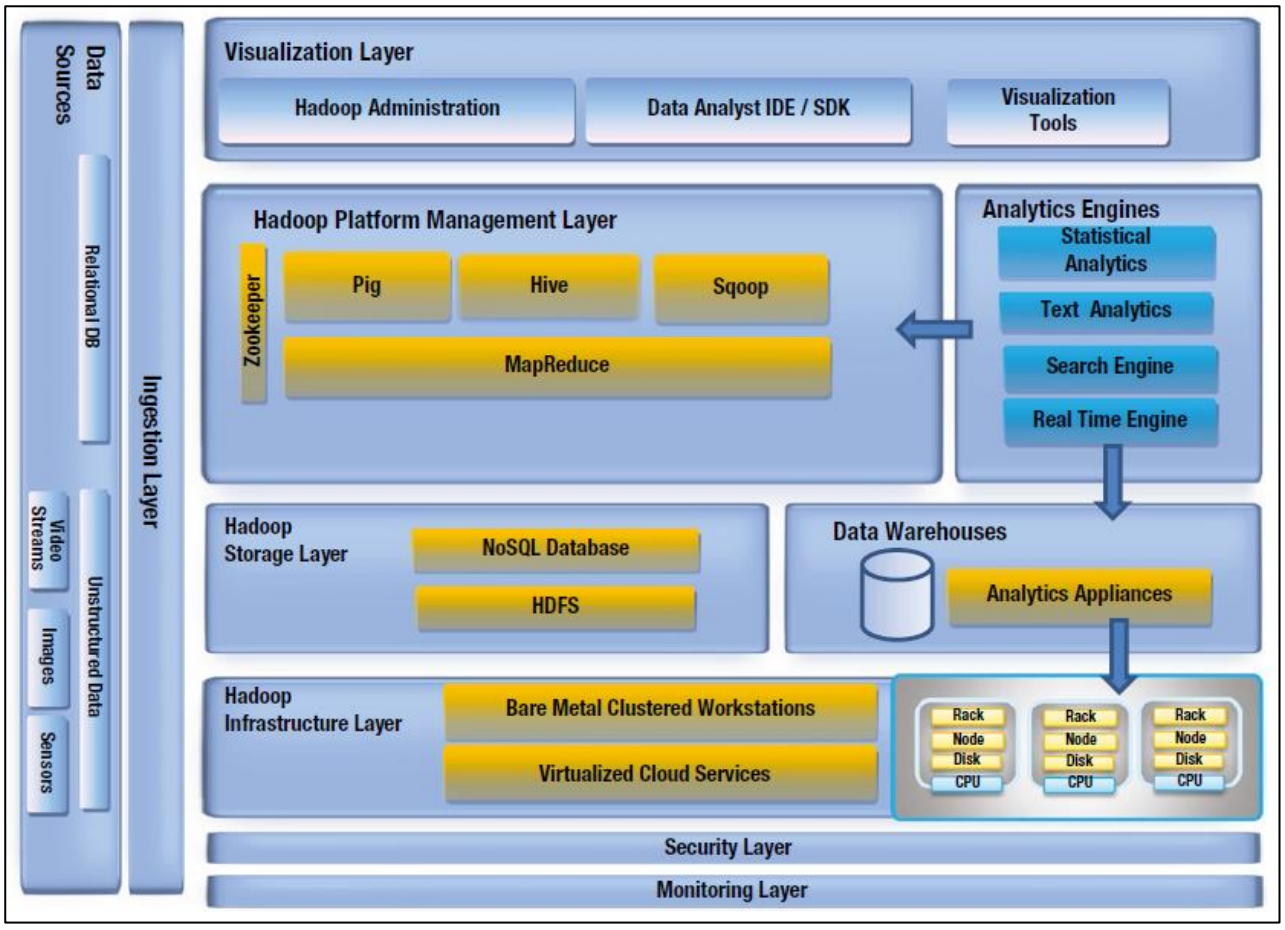

Figure 2. Architecture de Big Data

Cette figure décrit les composants nécessaires de l'architecture qui doivent faire partie d'un système Big Data. Il est nécessaire de choisir des Framework open source ou sous licence pour profiter pleinement de toutes les fonctionnalités des différents composants d'un système Big Data.

L'architecture d'un système Big Data se compose des couches suivantes *[5]* :

• Visualisation (couche visualisation) : cette couche est utile pour les analystes de données et les scientifiques pour mieux comprendre et plus rapidement, et donc être de plus en plus en mesure d'examiner différents aspects des données dans différents modes visuels.

• Hadoop Platform Management (couche de gestion de la plateforme Hadoop) : cette couche fournit les outils nécessaires au traitement de MapReduce ainsi que des langages de requête afin d'accéder aux bases de données NoSQL en utilisant le système de fichiers de stockage distribué HDFS (PIG, HIVE, Sqoop, etc.).

• Hadoop Storage (couche de stockage Hadoop) : cette couche est dédiée au stockage des données en utilisant un stockage et un traitement massivement distribué qui constitue un changement dans la façon dont une entreprise gère les données. Hadoop utilise HDFS, qui est un système de fichiers distribué conçu pour stocker un très grand volume d'informations (téraoctets ou pétaoctets) sur un grand nombre de machines d'un cluster. Il stocke les données de manière fiable, s'exécute sur du matériel de base, utilise des blocs pour stocker un fichier ou une partie de fichier.

• Security (couche de sécurité) : cette couche est conçue pour la protection des données car la sécurité de ces données devient une préoccupation majeure. Les habitudes d'achat des clients, les antécédents médicaux des patients, la démographie des maladies génétiques, tous ces types et de nombreux autres types et utilisations des données doivent être protégés, à la fois pour répondre aux

exigences de conformité et pour protéger la vie privée de l'individu. Ces exigences de sécurité doivent faire partie de tout système Big Data dès le départ.

- Monitoring (couche de surveillance) : avec autant de clusters de stockage de données distribués et de multiples points d'ingestion de source de données, il est important d'avoir une image complète du système Big Data grâce aux systèmes de surveillance. Par conséquent, cette couche définit les concepts utilisés par ces systèmes de surveillance pour augmenter les performances de Hadoop.

- Ingestion (couche d'ingestion) : cette couche permet de séparer le bruit des informations pertinentes. Il doit être capable de gérer l'énorme volume, la vitesse élevée et la variété des données. Il devrait également avoir la possibilité de valider, nettoyer, transformer, réduire et intégrer les données afin que l'écosystème Hadoop puisse les utiliser plus tard.

- Data Sources (sources de données) : cette couche définit les différents types de sources de données à l'intérieur et à l'extérieur de l'entreprise qui doivent être analysées dans une solution Big Data. Les données du Big Data se caractérisent par un volume, une variété, une vitesse et une valeur énormes. Il s'agit donc d'un flux de données complexe qui doit être parfaitement traité dans la couche d'ingestion.

4. Domaines d'utilisation de Big Data

Les secteurs d'activité, qui ont été les premiers à s'intéresser au Big Data, peuvent être séparés en deux groupes : ceux pour qui le Big Data répond à des besoins historiques de leur activité, et ceux pour qu'il ouvre de nouvelles opportunités.

Dans la première catégorie, on retrouve des secteurs qui manipulent quotidiennement des volumes de données très important, avec des problématiques de vitesse associées.

On y trouve :

- Les Banques : la sanctuarisation de données anciennes due à des contraintes réglementaires
- La Télécommunication : l'analyse de l'état du réseau en temps réel ;
- Les Médias Numériques : le ciblage publicitaire et l'analyse de sites web ;
- Les Marchés Financier : l'analyse des transactions pour la gestion des risques et la gestion des fraudes, ainsi que pour l'analyse des clients.

La deuxième catégorie de secteur est plus hétérogène, les besoins, mais aussi l'utilisation qui est faite du Big Data, peuvent être très différents.

On y trouve :

- Les Services Publics : l'analyse des compteurs (gaz, électricité, etc.) et la gestion des équipements ;
- Le Marketing : le ciblage publicitaire et l'analyse de tendance ;
- La Santé : l'analyse des dossiers médicaux et l'analyse génomique.

Le Big Data est déjà amplement utilisé par certains secteurs d'activité, mais d'autres commencent à s'y intéresser, notamment la Recherche, la Police ou encore les Transports. Cette liste d'utilisateurs s'agrandit de plus en plus au fil des années *[3]*.

5. Les principales technologies de Big Data

- Des bases de données NoSQL : (comme MongoDB, Cassandra ou Redis) qui implémentent des systèmes de stockage considérés comme plus performants que le traditionnel SQL pour l'analyse de données en masse (orienté clé/valeur, document, colonne ou graphe).

- Des infrastructures de serveurs pour distribuer les traitements : sur des dizaines, centaines, voire milliers de nœuds. C'est ce qu'on appelle le traitement massivement parallèle. Le Framework Hadoop est sans doute le plus connu d'entre eux. Il combine le système de fichiers distribué HDFS, la base NoSQL HBase et l'algorithme MapReduce. D'autres technologies visant à tendre vers des traitements plus "temps réel" ont émergé dans la foulée (c'est le cas d'Apache Spark).

- Le stockage des données en mémoire : permet d'accélérer les temps de traitement des requêtes.

L'avantage du traitement en mémoire est celui de la vitesse puisque les données sont immédiatement accessibles. En revanche, ces données ne sont pas stockées sur le long terme, ce qui peut poser des problèmes d'historisation *[Web5]*.

Figure 3. Les principales technologies de Big Data

6. Aperçu sur l'écosystème Hadoop

Déjà, en 2012, IDC (International Data Corporation) postulait que de 2005 à 2020, le volume de données croîtrait d'un facteur de 300, soit de 130 exaoctets à 40 000 Exaoctets, ce qui représente plus de 5200 Gigaoctet créés pour chaque personne en 2020. Actuellement, le volume des données en circulation connaît une démultiplication permanente : 5 exaoctets de données sont désormais produits tous les deux jours. En 2014, 90 % de toutes les données jamais générées par l'homme l'ont été au cours des deux dernières années ; Cisco renchérit ce constat lorsqu'il prédit que le trafic IP global annuel serait de 1,3 zettaoctet en 2016. Cet accroissement dans le trafic réseau est attribué à l'accroissement du nombre des smartphones, tablettes et autres appareils connectés à Internet, à la croissance des communautés d'utilisateurs Internet, à la croissance de la bande

passante, à la rapidité offerte par les opérateurs de télécommunication, et à la disponibilité et à la connectivité du Wi-Fi. Ne parlons même pas de la variété d'actifs de données créées !

L'échelle de cette croissance de données surpasse la capacité raisonnable des technologies traditionnelles, précisément les systèmes de gestion de bases de données relationnelles (SGBDR), ou même la configuration matérielle typique supportant les accès à ces données. Plus encore, les données canalisées vers le réseau Internet créent de la pression pour la capture cohérente et rapide de ces données. Les entreprises doivent trouver le moyen de maîtriser et de pouvoir efficacement traiter ces données pour continuer à servir fidèlement leur clientèle et rester compétitives *[Web6]*.

Ainsi résoudre ce problème d'explosion de données ne peut se faire ni avec les approches de centralisation des données dans un seul serveur comme c'est fait dans le passé, et ni avec les approches centralisées de traitement in-memory. Il est dès lors indispensable de penser à d'autres approches, et c'est là où Hadoop entre en scène.

Hadoop est donc une architecture ou une plateforme logicielle de stockage et d'analyse de données dont on peut lister les propriétés suivantes :

- La plateforme possède un système de fichiers distribué très facilement extensible : Hadoop gère seul la distribution et le stockage des données sur ses différents nœuds, et pour augmenter la capacité de stockage il suffit d'ajouter des nœuds de données dans la plateforme.

- Les codes des traitements sont routés jusqu'aux données : Cette stratégie est la plus efficace pour de grosses volumétries de données stockées sur des machines standard reliées par des réseaux standard. Les nœuds de données se transforment donc en nœuds de calculs le temps des traitements, et par conséquent, augmenter le nombre de nœuds de données pour accroître la capacité de stockage augmente aussi la capacité de traitement.

- Des mécanismes de tolérance aux pannes sont intégrés à la plateforme : Hadoop étant conçu pour fonctionner sur du matériel standard, des pannes fréquentes sont supposées inéluctables, et les données sont répliquées sur plusieurs nœuds afin d'être toujours accessibles. Quand un réplica disparaît (suite à une panne), ses copies sont à nouveau répliquées pour maintenir un bon taux de réplication. De même, les tâches de traitements exécutées sur les nœuds de données sont monitorées et relancées sur le nœud d'un autre réplica si une panne survient. L'utilisateur n'a pas à se soucier de la tolérance aux pannes.

- Un paradigme de programmation MapReduce est intégré à la plateforme : Ce paradigme convient à la récupération et au filtrage de données stockées dans l'ensemble des nœuds de données, ainsi qu'à la réalisation de quelques traitements sur les données retenues. Son intégration dans la pile logicielle de Hadoop rend son utilisation très simple.

Conclusion

Le Big Data est un concept vulgarisé ces dernières années pour traduire le fait que les entreprises sont confrontées à de gros volumes de données à traiter progressivement et considérablement. Cette tendance autour de la collecte et de l'analyse du Big Data a donné naissance à de nouvelles solutions qui combinent les technologies classiques, telle que le Framework Hadoop. A ce stade on peut dire que le Big Data est un écosystème large et complexe, il nécessite la maitrise des technologies matérielles et logicielles diverses.

Dans le chapitre qui suit nous présenterons quelques technologies de haute performance utilisées dans les Big Data.

Chapitre 03 :

Technologies de haute performance

Chapitre 03 : Technologies de haute performance

Introduction

Apache Hadoop (High-availability distributed object-oriented platform) est un système distribué qui répond aux problématiques de Big Data. D'une part, il propose un système de fichier distribué HDFS (Hadoop Distributed File System) pour assurer le stockage et l'intégrité des données en dupliquant plusieurs copies d'un même bloc à travers des dizaines, des centaines, voire des milliers de machines différentes. D'autre part, Hadoop fournit un système d'analyse de données appelé MapReduce pour réaliser des traitements sur des gros volumes de données grâce à sa répartition efficace du travail sur différents nœuds de calcul.

Dans ce chapitre nous allons présenter Hadoop avec ses principales composantes HDFS et MapReduce et leurs architectures, ainsi que ses différents éléments complémentaires, et on termine avec la présentation de Spark, ses composants et son architecture.

1. Hadoop

Hadoop est un Framework logiciel open source permettant de stocker des données, et de lancer des applications sur des grappes de machines standards. Cette solution offre un espace de stockage massif pour tous les types de données, une immense puissance de traitement et la possibilité de prendre en charge une quantité de tâches virtuellement illimitée. Basé sur Java, ce Framework fait partie du projet Apache, sponsorisé par Apache Software Foundation.

À l'heure actuelle, Hadoop est la principale plateforme du Big Data. Utilisé pour le stockage et le traitement d'immenses volumes de données, ce Framework logiciel et ses différents composants sont utilisés par de très nombreuses entreprises pour leurs projets Big Data *[Web7]*.

Pendant des années, alors que la puissance de traitement des serveurs d'applications explosait, les bases de données ont pris du retard en raison de leur capacité et de leur vitesse limitée.

Aujourd'hui, alors que de nombreuses applications génèrent des volumes considérables de données à traiter, Hadoop joue un rôle important et très attendu dans la refonte du monde des bases de données *[Web8]*.

L'écosystème Hadoop comporte des éléments de base indispensables au Framework, d'autres sont des éléments complémentaires qui ajoutent des fonctionnalités.

Les éléments de base de Hadoop sont les suivants :

1.1. HDFS (Le système de fichier distribué de Hadoop)

HDFS (Hadoop Distributed File System) est un système de fichier distribué permettant de stocker, de répliquer des données et de récupérer des fichiers en un temps record. Il s'agit de l'un des composants basiques du Framework Hadoop, et plus précisément de son système de stockage.

De par sa capacité massive et sa fiabilité, HDFS est un système de stockage très adapté au Big Data. En combinaison avec YARN, ce système augmente les possibilités de gestion de données du Cluster Hadoop et permet donc de traiter le Big Data efficacement. Parmi ses principales fonctionnalités, on compte la possibilité de stocker des Térabytes, voire des Pétabytes de données *[Web9]*.

- Fonctionnement de HDFS :

HDFS a une architecture maître / esclave. Un cluster HDFS se compose d'un seul NameNode, un serveur maître qui gère l'espace de noms du système de fichiers et régule l'accès aux fichiers par les clients. En outre, il existe un certain nombre de nœuds de données, généralement un par nœud dans le cluster, qui gèrent le stockage attaché aux nœuds sur lesquels ils s'exécutent. HDFS expose un espace de noms de système de fichiers et permet de stocker les données utilisateur dans des fichiers. En interne, un fichier est divisé en un ou plusieurs blocs et ces blocs sont stockés dans un ensemble de DataNodes. Le NameNode exécute les opérations d'espace de noms du système de fichiers comme l'ouverture, la fermeture et le renommage de fichiers et de répertoires. Il détermine également le mappage des blocs aux DataNodes. Les DataNodes sont chargés de répondre aux demandes de lecture et d'écriture des clients du système de fichiers. Les DataNodes effectuent également la création, la suppression et la réplication de blocs sur instruction du NameNode.

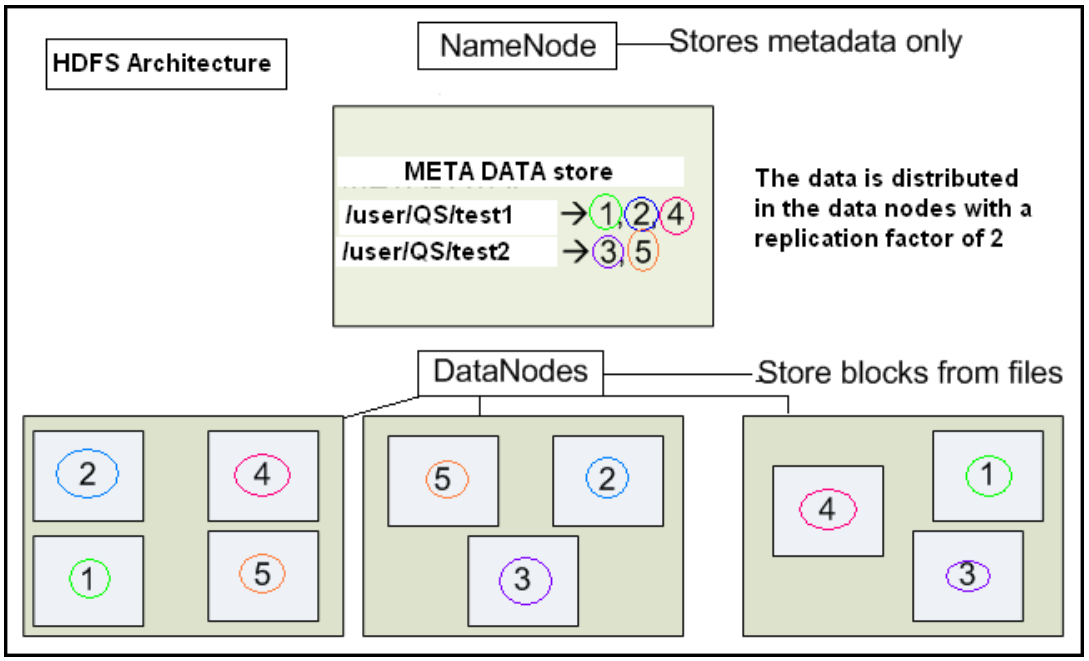

Figure 4. Architecture de HDFS

Le NameNode et le DataNode sont des logiciels conçus pour fonctionner sur des machines standard. Ces machines exécutent généralement un système d'exploitation (OS) GNU / Linux. HDFS est construit en utilisant le langage Java ; toute machine prenant en charge Java peut exécuter NameNode ou le logiciel DataNode. L'utilisation du langage Java hautement portable signifie que HDFS peut être déployé sur une large gamme de machines. Un déploiement typique a une machine dédiée qui exécute uniquement le logiciel NameNode. Chacune des autres machines du cluster exécute une instance du logiciel DataNode. L'architecture n'empêche pas d'exécuter plusieurs DataNodes sur la même machine mais dans un véritable déploiement ce qui est rarement le cas.

L'existence d'un seul NameNode dans un cluster simplifie considérablement l'architecture du système. Le NameNode est l'arbitre et le référentiel de toutes les métadonnées HDFS. Le système est conçu de telle manière que les données utilisateur ne transitent jamais par le NameNode *[Web10]*.

Donc les composantes de base de HDFS sont :

- Le nœud principal ou NameNode : Il se caractérise par :
  - Responsable de la distribution et de la réplication des blocs ;
  - Serveur d'informations du HDFS pour le client HDFS ;
  - Stocke et gère les métadonnées ;
  - Comporte la liste des blocs pour chaque fichier (dans le cas de lecture) ;
  - Contient la liste des DataNodes pour chaque bloc (dans le cas de l'écriture) ;
  - Tenir les attributs des fichiers (ex : nom, date de création, facteur de réplication) ;
  - Logs toute métadonnée et toute transaction sur un support persistant ;
  - Lectures/écritures ;
  - Créations/suppressions ;
  - Démarre à partir d'une image d'HDFS (FsImage).

- Le nœud de données ou DataNode : Il se caractérise par :
  - Stocke des blocs de données dans le système de fichier local ;
  - Maintenir des métadonnées sur les blocs possédés (ex : CRC) ;
  - Serveur de bloc de données et de métadonnées pour le client HDFS ;
  - Heartbeat avec le NameNode : Heartbeat est le système permettant sous Linux la mise en clusters de plusieurs serveurs pour effectuer entre eux un processus de tolérance de panne. Le processus Heartbeat se chargera de passer un message-aller vers le NameNode indiquant : son identité, sa capacité totale, son espace utilisé, son espace restant.

- Noeud maître secondaire ou Secondary NameNode : Le NameNode dans l'architecture Hadoop est un point unique de défaillance. Si ce service est arrêté, il n'y a pas moyen de pouvoir extraire les blocs d'un fichier donné. Pour résoudre ce problème, un NameNode secondaire appelé Secondary NameNode a été mis en place dans l'architecture Hadoop.

  Son rôle consiste à :
  - Télécharger régulièrement les logs sur le NameNode ;
  - Crée une nouvelle image en fusionnant les logs avec l'image HDFS ;
  - Renvoie la nouvelle image au NameNode *[6]*.

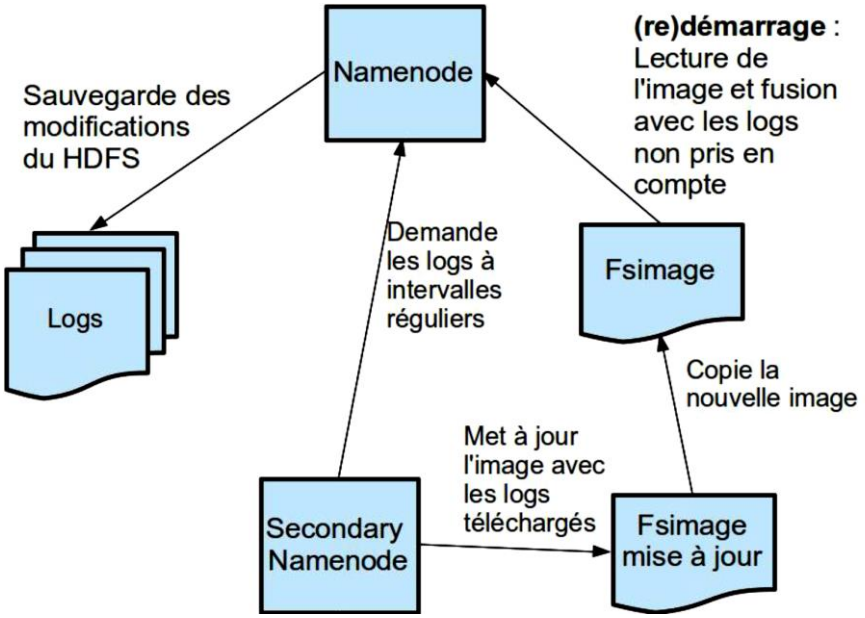

Figure 5. Fonctionnement du Secondary NameNode

- La lecture dans HDFS

Pour lire un fichier au sein de HDFS, il faut suivre les étapes suivantes :

Etape 1 : Le client indique au NameNode qu'il souhaite lire un fichier(ex: Monfichier.txt) ;

Etape 2 : Le NameNode lui indiquera la taille de fichier (nombre de blocs) ainsi que les différents DataNodes hébergeant les n blocs ;

Etape 3 : Le client récupère chacun des blocs à un des DataNodes ;

Etape 4 : En cas d'erreur/non réponse d'un des DataNode, il passe au suivant dans la liste fournie par le NameNode.

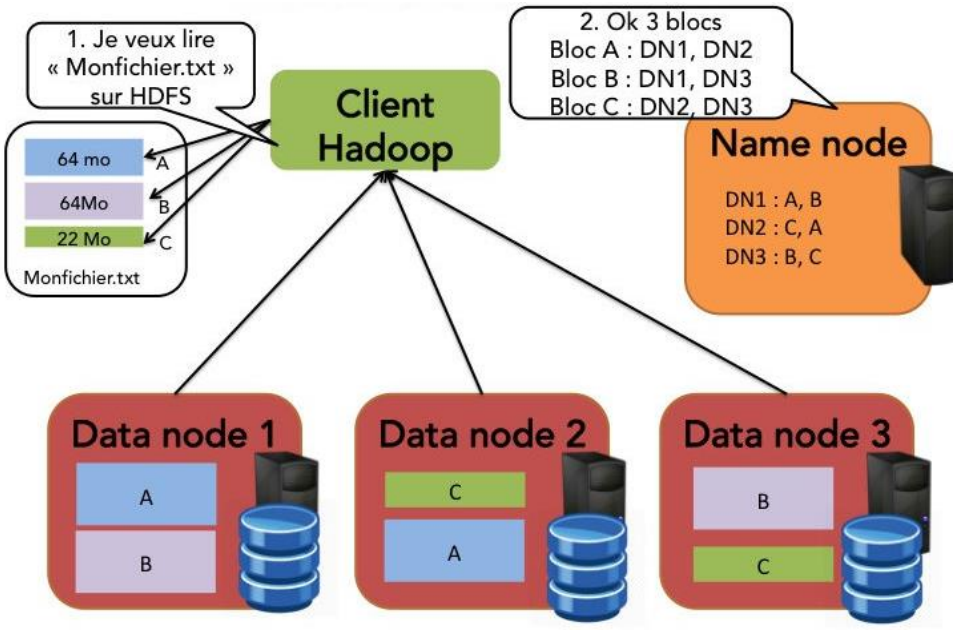

Figure 6. La lecture dans HDFS

- L'écriture dans HDFS

Pour écrire un fichier au sein de HDFS :

Etape 1 : Admettons qu'on souhaite stocker le fichier Monfichier.txt sur HDFS ;

Etape 2 : Le programme va diviser le fichier en blocs de 64KB (ou autre, selon la configuration) ;

Etape 3 : Le NameNode lui indique les DataNodes à contacter ;

Etape 4 : Le client contacte directement le DataNode concerné et lui demande de stocker le bloc ;

Etape 5 : Les DataNodes s'occuperont – en informant le NameNode – de répliquer les données entre eux pour éviter toute perte de données.

Etape 6 : Le cycle se répète pour le bloc suivant *[Web11]*.

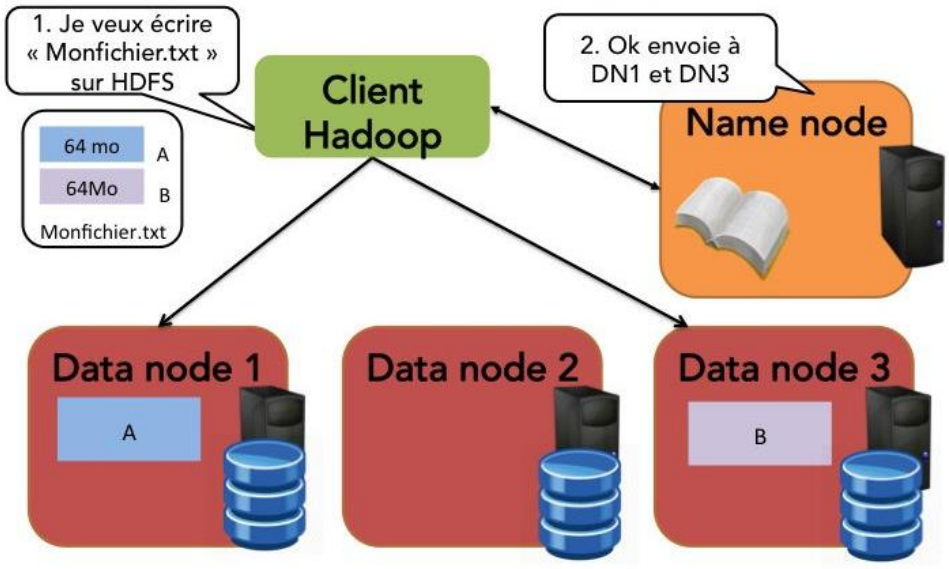

Figure 7. L'écriture dans HDFS

1.2. MapReduce

MapReduce est un modèle (ou structure) de programmation disponible dans les environnements Hadoop qui est utilisé pour accéder aux Big Data stockées dans le HDFS, c'est un élément essentiel et fait partie intégrante du fonctionnement de l'environnement Hadoop.

MapReduce facilite les traitements concurrents en divisant les pétaoctets de données en volumes plus petits « le principe diviser pour régner » et en les traitant en parallèle sur des serveurs standard dédiés à Hadoop. Pour résumer, MapReduce agrège les données de plusieurs serveurs et renvoie un résultat consolidé à l'application.

Avec MapReduce, plutôt que d'envoyer les données à l'endroit où se trouve l'application ou les algorithmes, les algorithmes sont exécutés sur le serveur où se trouvent déjà les données, ce qui a pour effet d'accélérer le traitement. L'accès aux données et le stockage des données se font sur disque, les entrées sont généralement stockées sous forme de fichiers contenant des données structurées, semi-structurées ou non structurées, et la sortie est également stockée dans des fichiers *[Web12]*.

Le modèle de programmation fournit un cadre à un développeur afin d'écrire une fonction Map et une fonction Reduce. Tout l'intérêt de ce modèle de programmation est de simplifier la vie du développeur. Ainsi, ce développeur n'a pas à se soucier du travail de parallélisation et de distribution du travail. MapReduce permet au développeur de ne s'intéresser qu'à la partie algorithmique *[Web13]*.

Un programme MapReduce peut se résumer à deux fonctions Map () et Reduce () :

o MAP : va transformer les données d'entrée en une série de couples clé/valeur. Elle va regrouper les données en les associant à des clés, choisies de telle sorte que les couples clé/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 fonction Map s'écrit de la manière suivante : Map : (K1, V1) → List (V2, V2).

o REDUCE : va appliquer un traitement à toutes les valeurs de chacune des clés distinctes produite par l'opération MAP. Au terme de l'opération REDUCE, on aura un résultat pour chacune des clés distinctes. Ici, on attribuera à chacune des machines du cluster une des clés uniques produites par MAP, en lui donnant la liste des valeurs associées à la clé. Chacune des machines effectuera alors l'opération REDUCE pour cette clé. La fonction Reduce s'écrit de la manière suivante :

Reduce: (K2, List (V2)) → List (K3, V3). **Remarque :* bien souvent K2 = K3

L'exemple classique est celui du WordCount qui permet de compter le nombre d'occurrences d'un mot dans un fichier. En entrée l'algorithme reçoit un fichier texte de trois lignes.

Dans un premier temps on applique un Split pour récupérer le contenu de chaque ligne. Ensuite on applique la fonction Map pour extraire l'information qui nous intéresse, chaque mot de la ligne on lui associe le couple clé/valeur.

Avant de présenter la fonction Reduce, deux opérations intermédiaires doivent être exécutées pour préparer la valeur de son paramètre d'entrée. La première opération appelée shuffle permet de grouper les valeurs dont la clé est commune. La seconde opération appelée sort permet de trier par clé. A la différence des fonctions Map et Reduce, shuffle et sort sont des fonctions fournies par le Framework Hadoop, donc, il n'a pas à les implémenter.

Enfin on applique la fonction Reduce afin d'agrégé et de filtré les informations pour obtenir le résultat illustré par la figure qui suit.

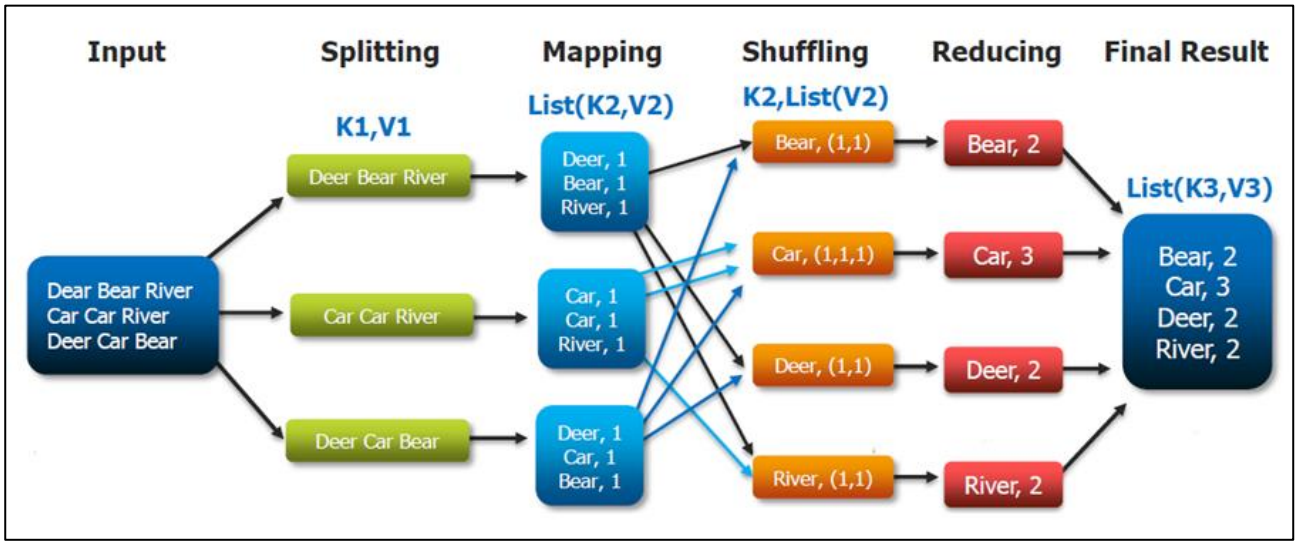

Figure 8. Exemple d'un programme MapReduce (WordCount)

- YARN

La description de Hadoop comme possédant 2 couches (MapReduce et HDFS) est correcte pour la version 1 de Hadoop, depuis la version 2 Hadoop a adopté une troisième couche YARN.

YARN est l'abréviation de « Yet Another Resource Negotiator » (plus simplement, un négociateur de ressources). Cet élément assure la gestion et la planification des ressources (clusters) Hadoop et décide de ce qui doit se passer dans chaque nœud de données, le nœud maître central qui gère toutes les demandes de traitement est le « Resource Manager ». Le Resource Manager interagit avec les différents Node Managers : chaque DataNode esclave possède son propre Node Manager pour l'exécution des tâches *[Web14]*.

YARN provient d'un découpage de la première version de Hadoop MapReduce en deux sous-couches :

- L'une dédiée à la gestion de la puissance de calcul et de la répartition de la charge entre les machines d'un cluster (YARN) ;

- L'autre dédiée à l'implémentation de l'algorithme MapReduce en utilisant cette première couche.

Ce découpage a amené de nombreux autres outils (liés ou indépendants d'Apache Hadoop) à profiter de l'environnement HDFS comme moyen de stocker aisément de grandes quantités de données sans nécessairement MapReduce. Un écosystème d'outils liés à Hadoop a alors émergé et est de nos jours très développé *[Web15]*.

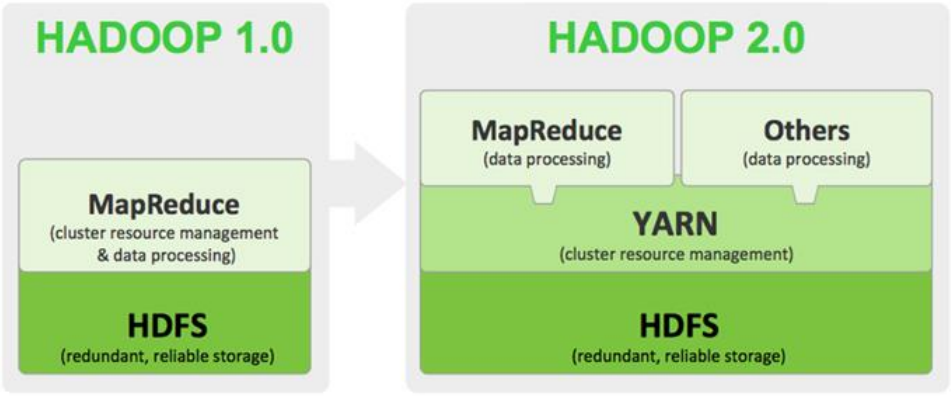

Figure 9. Les versions de Hadoop

1.3. Les éléments complémentaires de l'écosystème Hadoop

1.3.1. Hive – Data warehousing

Hive est à l'origine un projet Facebook qui permet de faire le lien entre le monde SQL et Hadoop, cet élément permet l'exécution de requêtes SQL sur un cluster Hadoop en vue d'analyser et d'agréger les données. Le langage SQL est nommé HiveQL, c'est un langage de visualisation uniquement, c'est pourquoi seules les instructions de type "Select" sont supportées pour la manipulation des données. Dans certains cas, les développeurs doivent faire le Mapping entre les structures de données et Hive *[7]*.

1.3.2. Pig – Réduire les fonctions MapReduce

Pig, développé initialement par Yahoo!, il est semblable à Hive : il élimine la nécessité de créer des fonctions MapReduce pour interroger le HDFS. Comme c'est le cas avec HiveQL, le langage Pig (Pig Latin) est très proche du SQL, c'est une couche de langage pour les flux de données qui repose sur MapReduce.

Pig dispose également d'un environnement d'exécution qui s'interface avec HDFS et les scripts écrits des langages tels que Java ou Python peuvent également être intégrés dans Pig *[Web14]*.

1.3.3. Flume – Importation des big data

Flume est un outil spécialisé dans l'importation des big data et qui agit comme un « messager » entre différentes sources de données et le HDFS, cet élément peut collecter, agréger et transmettre en streaming vers le HDFS des volumes considérables de données de logs ou d'événements générés par des applications telles que des sites de médias sociaux, des applications IoT et des portails d'e-commerce.

Flume présente les caractéristiques suivantes :

- A une architecture distribuée ;
- Assure des transferts de données fiables ;
- Est tolérant aux incidents ;
- Possède la souplesse de collecter les données par lots ou en temps réel ;
- Si nécessaire, peut être mis à l'échelle horizontalement pour gérer plus de trafic.

Les sources de données communiquent avec les agents Flume : chaque agent est associé à une source, un canal et un « sink ». La source collecte les données en provenance d'un expéditeur (sender), le canal stocke les données temporairement et le sink transfère les données à la destination, à savoir un serveur Hadoop *[Web14]*.

1.3.4. Sqoop – Importation de données dans les bases relationnelles

Sqoop est également (comme Flume décrit précédemment) un outil pour l'importation des données. Alors que Flume travaille sur des données non structurées ou semi-structurées, Sqoop est utilisé pour exporter et importer des données à partir de bases de données relationnelles. Comme la

plupart des données d'entreprise sont stockées dans des bases relationnelles, Sqoop est utilisé pour importer ces données dans Hadoop pour que les analystes puissent les examiner.

Pour exporter et importer des données, les administrateurs et développeurs de bases de données disposent d'une interface en ligne de commande (CLI) très simple. Sqoop convertit les commandes au format MapReduce et les envoie au HDFS via YARN.

Tout comme Flume, Sqoop est tolérant aux incidents et peut exécuter des opérations concurrentes *[Web14]*.

1.3.5. ZooKeeper – Coordination des applications distribuées

ZooKeeper est un service de coordination des services d'un cluster Hadoop, en particulier, le rôle de ZooKeeper est de fournir aux composants Hadoop les fonctionnalités de distribution, pour cela il centralise les éléments de configuration du cluster Hadoop, propose des services de clustérisassions et gère la synchronisation des différents éléments (événements).

ZooKeeper est un élément indispensable au bon fonctionnement de Hbase *[7]*.

1.3.6. Kafka – Transferts de données

Kafka est un système distribué de communication inter-processus de type publication-abonnement (publish-subscribe) qui est souvent utilisé avec Hadoop pour accélérer les transferts de données, ce système est constitué d'un groupe de serveurs qui servent d'intermédiaires entre les Producers et les Consumers.

Dans un contexte de big data, un exemple de Producer pourrait être un capteur qui relève des valeurs de température pour les transmettre au serveur. Les Consumers sont en fait les serveurs Hadoop. Les Producers publient un message sur un sujet donné (topic), et les Consumers sélectionnent les messages en « écoutant » leur sujet *[7]*.

1.3.7. HBase – Base de données non relationnelles

HBase est un sous-projet de Hadoop, c'est un système de gestion de base de données non relationnel distribué, écrit en Java, disposant d'un stockage structuré pour les grandes tables. HBase est inspirée des publications de Google sur BigTable. HBase est donc une base de données orientée colonnes tout comme BigTable, elle est souvent utilisée conjointement au système de fichiers distribué HDFS qui permet de faciliter la distribution des données de HBase sur plusieurs

nœuds. Contrairement à HDFS, HBase permet de gérer les accès aléatoires read/write pour des applications de type temps réel [7].

Aujourd'hui il est donc difficile de se retrouver dans la " jungle " de Hadoop parce que c'est une technologie qui encore jeune et que beaucoup de travaux d'acteurs différents qui veulent prendre le train Big Data en marche.

Dans une distribution Hadoop Ils existent de nombreux autres modules, on a donc cité uniquement que certaines composantes. La figure suivante illustre un certain autre nombre de module de l'écosystème Hadoop, à noter que c'est une liste qui reste non-exhaustive.

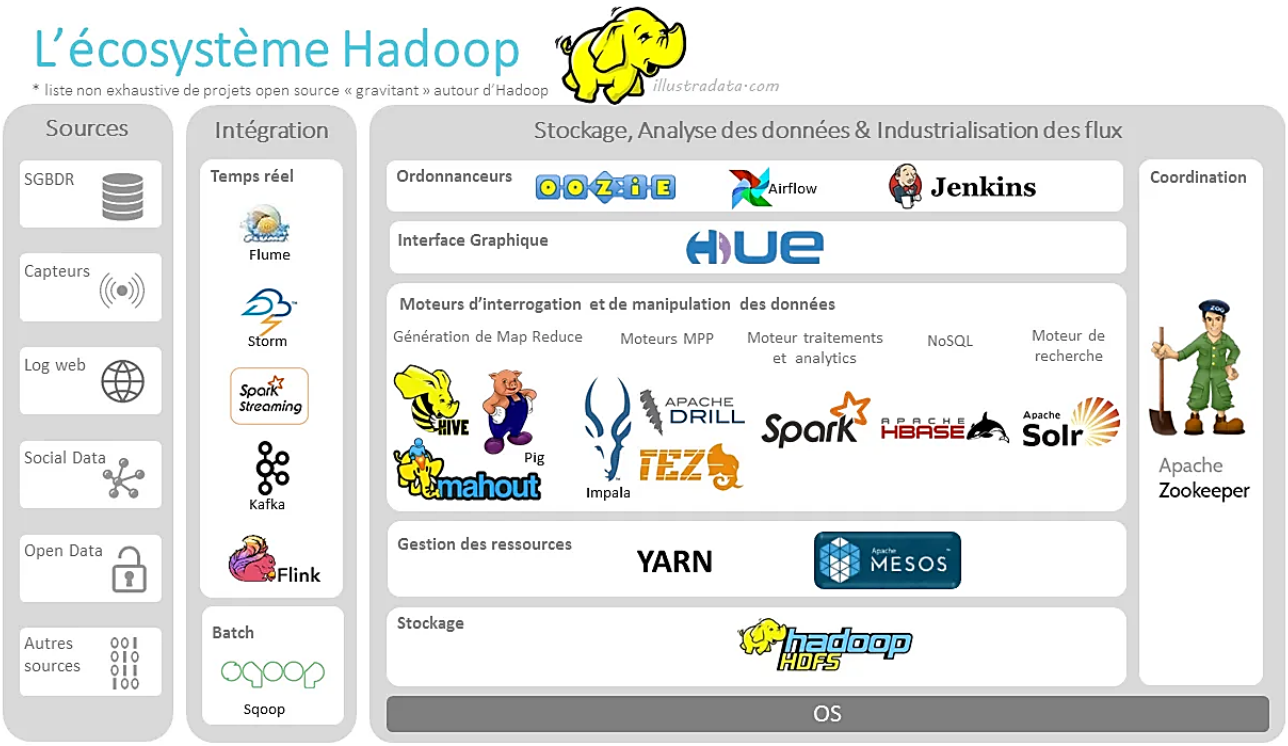

Figure 10. L'écosystème Hadoop

2. Apache SPARK

Apache Spark est un Framework open source de traitement de données, il s'agit d'un projet de la fondation Apache issu de recherches à Berkeley, il est né d'une constatation simple : la technologie MapReduce est très intéressante mais dès que des requêtes complexes (itérées) sont nécessaires et que plus de temps réel entre en jeu, elle atteint ses limites. D'où l'idée de créer un nouveau Framework utilisant la parallélisation massive avec une technologie in-memory.

- La parallélisation massive : il s'agit de distribuer les calculs dans un grand nombre de processeurs ou de machines suivant la taille de l'infrastructure.
- In-memory : il s'agit tout simplement de charger les données en mémoire c'est-à-dire dans la RAM des machines.

Le fait de travailler sur une technologie in memory vient de deux constats :

- La plupart des jeux de données de Big data font quelques gigas et une utilisation de la mémoire RAM devient alors possible ;

- Le coût de la mémoire RAM a très fortement baissé et on peut facilement obtenir des grandes quantités de mémoire.

Grâce à ses principes, Spark permet de diviser par 100 les temps de calcul par rapport à l'utilisation de MapReduce sur Hadoop, de plus, il permet d'appliquer des algorithmes impossibles à mettre en œuvre jusqu'ici avec les opérations de MAP et de REDUCE *[Web16]*.

2.1. Les composants de SPARK

Spark dans son ensemble se compose de diverses bibliothèques, API, bases de données, etc.

Les principaux composants d'Apache Spark sont les suivants :

• Spark Core : est le bloc de construction de base de Spark, qui comprend tous les composants pour la planification des travaux, l'exécution de diverses opérations de mémoire et la tolérance aux pannes, il abrite également l'API composée de RDD (Resilient Distributed Dataset). De plus, Spark Core fournit des API pour créer et manipuler des données dans RDD.

En décortiquant ce nom on peut un peu mieux le comprendre :

o Dataset : Il s'agit d'un jeu de données qui se parcourt comme une collection.

o Distributed : Cette structure est distribuée afin d'être découpée pour être traitée dans les différents nœuds.

o Resilient : Il est résilient, car il pourra être relu en cas de problème.

• Spark SQL : Apache Spark fonctionne avec les données non structurées à l'aide de son outil Spark SQL qui permet d'interroger des données via SQL, ainsi que via la forme SQL d'Apache Hive appelée HiveQL, il prend également en charge les données de diverses sources telles que les tables d'analyse, les fichiers journaux, JSON, etc.

Spark SQL permet aux programmeurs de combiner des requêtes SQL avec des modifications ou des manipulations programmables prises en charge par RDD en Python, Java, Scala et R.

• Spark Streaming : il traite des flux de données en direct ; les données générées par diverses sources sont traitées à l'instant même par Spark Streaming.

Par exemples les données comprennent les fichiers journaux, les messages contenant des mises à jour de statut publiées par les utilisateurs.

• GraphX : c'est la bibliothèque d'Apache Spark pour améliorer les graphiques et permettre le calcul parallèle au graphique, cette bibliothèque comprend un certain nombre d'algorithmes graphiques qui aident les utilisateurs à simplifier l'analyse graphique.

• MLlib : Apache Spark propose une bibliothèque contenant des services Machine Learning (ML) courants appelés MLlib, il fournit différents types d'algorithmes ML, y compris la régression, le clustering et la classification, qui peuvent effectuer diverses opérations sur les données pour en tirer des informations significatives *[Web17]*.

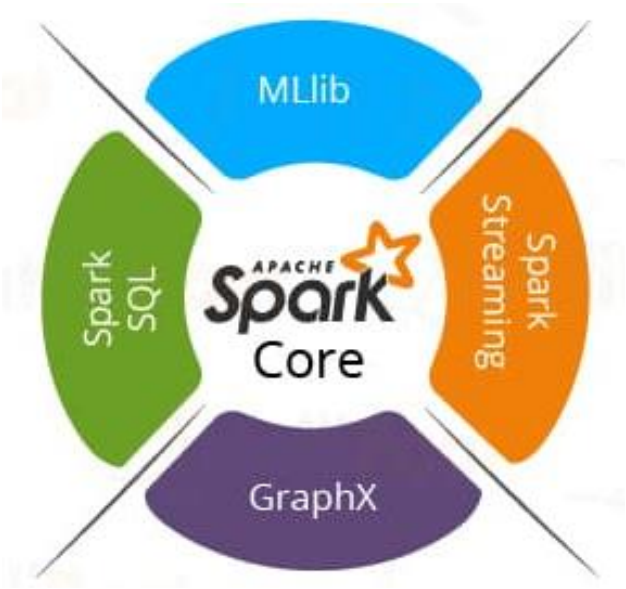

Figure 11. Les composants de Spark

2.2. Pourquoi utiliser Hadoop et Spark ensemble ?

Spark n'est pas un remplacement complet de Hadoop, il existe certains scénarios où Hadoop et Spark vont de pair.

- Les composants MLlib de Spark offrent des fonctionnalités qui ne sont pas facilement accessibles par MapReduce de Hadoop. En utilisant ces composants, les algorithmes d'apprentissage automatique peuvent être exécutés plus rapidement dans la mémoire.

- Spark n'a pas son propre système de fichiers distribué. En combinant Spark avec Hadoop, on peut utiliser diverses capacités Hadoop. Par exemple, on utilise Hadoop pour le stockage et Spark pour le traitement de données.

- Hadoop offre une sécurité renforcée, qui est un composant essentiel pour les charges de travail de production. Les charges de travail Spark peuvent être déployées sur les ressources disponibles n'importe où dans un cluster, sans allouer ni suivre manuellement les tâches individuelles.

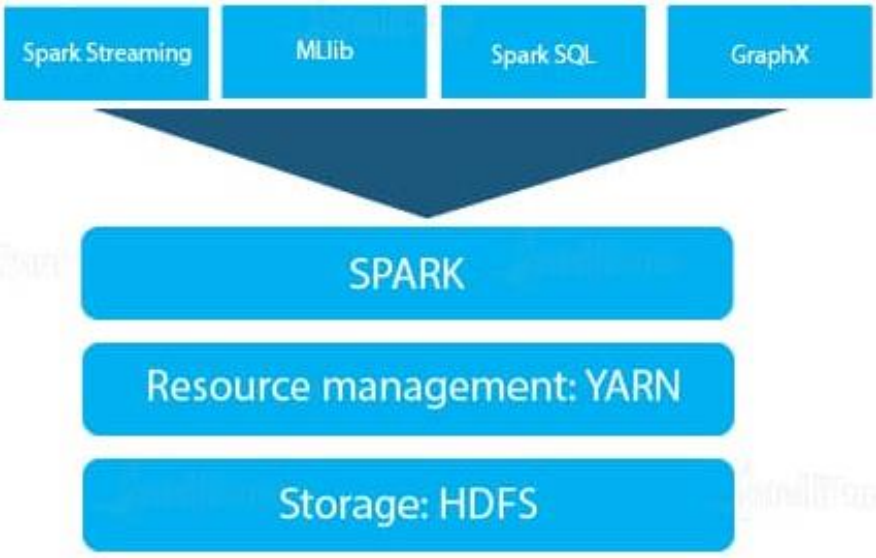

Figure 12. Utilisation de Spark & Hadoop

Spark est donc un Framework permettant de traiter de manière complexe des données de types variés. Il se situe donc plutôt au niveau des étapes Map et Reduce qu'au niveau de l'infrastructure en elle-même.

Spark se combine très bien avec des clusters Hadoop (en HDFS) et permet d'appliquer des algorithmes complexes sur des données issues de ces clusters. En effet, il permet de faire la même chose que MapReduce mais aussi de travailler sur du streaming, faire des requêtes interactives et appliquer des algorithmes d'apprentissage (machine learning ML).

Les capacités de cette technologie sont très importantes, on peut utiliser des clusters avec des milliers de nœuds et traiter des pétaoctets de données, si votre jeu de données est trop grand pour être traité in memory, Spark va automatiquement utiliser uniquement les parties nécessaires pour pouvoir traiter ce jeu.

Conclusion

Nous avons présenté dans ce chapitre Hadoop qui comprend un composant de stockage, connu sous le nom de HDFS, et un outil de traitement appelé MapReduce. Nous avons aussi donné un aperçu sur le Framework Spark, de fait, il n'est pas nécessaire de faire appel à Spark pour traiter ses données Hadoop. Et inversement, il est possible d'utiliser Spark sans faire intervenir Hadoop.

Spark n'a pas de système de gestion de fichiers propre, ce qui veut dire qu'il faut lui associer un système de fichiers - soit HDFS, soit celui d'une autre plate-forme de données. Néanmoins, Spark a été conçu pour Hadoop, et la plupart des chercheurs s'accordent pour dire qu'ils fonctionnent mieux ensemble.

On donc a étalé sur les aspects théoriques pour pouvoir réaliser et implémenter notre cluster Hadoop dans le chapitre qui suit.

Chapitre 04 :

Installation et configuration

Chapitre 04 : Installation et configuration

Introduction

Dans ce chapitre, nous allons aborder la partie pratique de notre projet qui consiste à décrire le matériel et l'ensemble des solutions logicielles utilisées pour la réalisation de notre cluster. Nous décrirons ensuite les étapes d'installation et de configuration de chaque solution.

1. Environnements

1.1. Matériels utilisés

Le système est constitué de 10 postes de haute performance, car les ressources hardware sont de gamme élevée qui permet d'avoir un Cluster avec suffisamment de ressources, nous présenterons donc dans ce qui suit les ressources matérielles utilisés ainsi que leurs caractéristiques.

* Ressources matérielles :

- Carte mère :
  - Fabriquant : Dell Inc.
  - Nom du produit : 040DDP
  - Version : A01
- Processeur :
  - Fabriquant : Intel
  - Famille : Intel Corp.
  - Version : Processeurs Intel® Core™ i5-4590 CPU @ 3.30GHz
  - Vitesse : 3.30GHz
  - Cœur : 4

* Caractéristiques :

    - Compatible 64 bits
    - Multicœurs
    - Hardware thread
    - Virtualisation améliorée

    • Mémoire RAM :

        - Type : DDR3
        - Type de prise : DIMM
        - Mode : Synchrone
        - Cycle d'horloge : 1600 MHz
        - Capacité : 4 Go

    • Stockage:

        - Capacité: 500 Go
        - Marque: Western Digital
        - Type de connexion : ATA

1.2. Environnements logiciels

• Le système d'exploitation Ubuntu 16.04.1 LTS :

Ubuntu est un système d'exploitation Open Source, gratuit, sécurisé et convivial, et dans un jargon plus technique, Ubuntu est une "distribution GNU/Linux. Rappelons qu'un système d'exploitation est ce qui nous permet d'utiliser notre ordinateur pour effectuer diverses tâches, d'interagir avec notre machine. Les systèmes d'exploitation GNU/Linux sont composés d'un noyau (Linux) et d'un ensemble d'outils de base qui agissent comme un relais entre l'utilisateur, les logiciels, et les composants matériels de notre ordinateur (souris, clavier, écran, imprimante, lecteur CD/DVD, etc.).

Dans notre projet on a opté pour l'utilisation de Ubuntu 16.04.1 LTS (long time support, « support à long terme »), étant donné la version la plus optimisée, la plus stables et qui peut être supportées durant cinq ans, afin que notre Cluster fonctionne correctement *[Web18]*.

• Java :

Par défaut, le système d'exploitation Ubuntu n'embarque pas le logiciel Java. Tant mieux car ses versions sont vite obsolètes en raison de correctifs qui comblent des failles de sécurité sans cesse découvertes. Si Java n'est plus indispensable au quotidien pour naviguer sur internet, certaines applications web ou métier imposent d'utiliser le runtime client (JRE) voire le Development kit (JDK). Deux solutions sont possibles pour installer Java sur Linux version Ubuntu : OpenJDK (Java version open source) ou JRE / JDK version propriétaire Oracle. Le tout se fait par ligne de commande dans un simple terminal *[Web19]*.

• Scala :

Scala est un langage de programmation multiparadigmes moderne conçu pour exprimer des schémas de programmation communs de manière concise, élégante et sécurisée. Il intègre de manière transparente les fonctionnalités des langages orientés objet et fonctionnels *[Web20]*.

- Openssh-server :

Abréviation d'Open Secure Shell, OpenSSH est une suite gratuite d'outils qui aident à sécuriser les connexions réseau. OpenSSH crypte tout le trafic (y compris les mots de passe) pour éliminer efficacement l'écoute clandestine, le détournement de connexion et d'autres attaques au niveau du réseau. La suite OpenSSH comprend le programme SSH qui remplace rlogin et telnet, et scp qui remplace RCP et FTP. OpenSSH a également ajouté SFTP et sftp-server qui implémentent une solution plus simple pour le transfert de fichiers *[Web21]*.

- Spark 2.0.2:

Apache Spark est un moteur d'analyse unifié pour le traitement de données à grande échelle. Il fournit des API de haut niveau en Java, Scala, Python et R, et un moteur optimisé qui prend en charge les graphiques d'exécution généraux. Il prend également en charge un riche ensemble d'outils de niveau supérieur, notamment Spark SQL pour SQL et le traitement de données structurées, MLlib pour l'apprentissage automatique, GraphX pour le traitement de graphiques et Streaming structuré pour le calcul incrémentiel et le traitement de flux *[Web22]*.

- Hadoop 2.6.0 :

La bibliothèque de logiciels Apache Hadoop est un cadre qui permet le traitement distribué de grands ensembles de données sur des clusters d'ordinateurs à l'aide de modèles de programmation simples. Il est conçu pour passer de serveurs uniques à des milliers de machines, chacune offrant un calcul et un stockage locaux. Plutôt que de s'appuyer sur du matériel pour fournir une haute disponibilité, la bibliothèque elle-même est conçue pour détecter et gérer les défaillances au niveau de la couche d'application, offrant ainsi un service hautement disponible au-dessus d'un cluster d'ordinateurs, chacun pouvant être sujet à des défaillances *[Web23]*.

2. Installations des environnements

2.1. Installation de Ubuntu 16.04.1 LTS

- Installation de Refus 2.5.7

Si nous souhaitons installer un nouveau système d'exploitation ou tester une distribution Linux depuis un Live CD, il nous faut un périphérique amorçable (bootable). Ce périphérique peut être une simple clé USB.

Si on copie simplement les fichiers contenus dans l'image ISO sur la clé USB, nous ne pourrons pas démarrer depuis cette dernière. En effet, pour qu'un périphérique soit bootable, il faut qu'il contienne des secteurs d'amorçage. Une simple copie ne crée pas ces secteurs. Pour les créer, il faut formater son disque de manière spéciale. De nombreux utilitaires le permettent. Pour y est faire nous avons alors utilisé le logiciel Refus afin de pouvoir créer notre clé USB bootable.

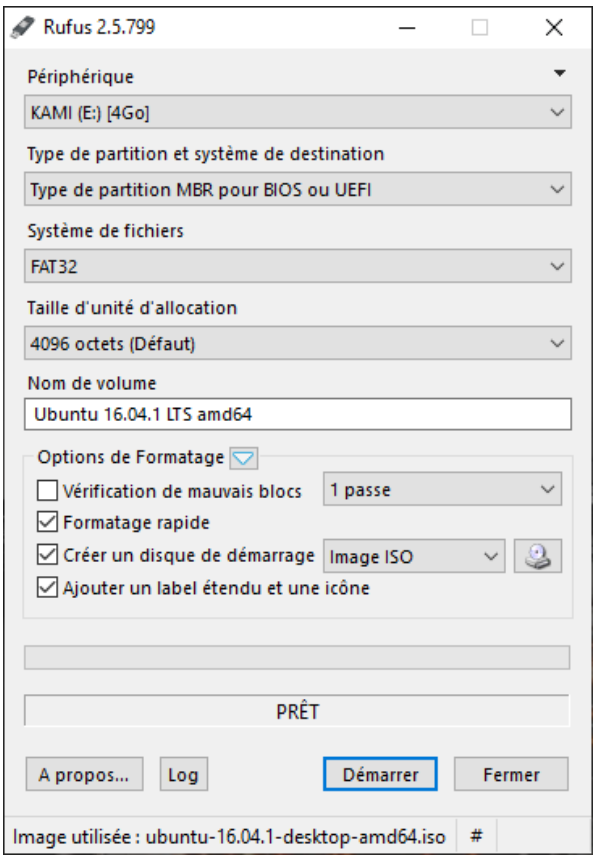

Figure 13. Création de clé USB Bootable avec Refus

- Installation de Ubuntu 16.04.1 LTS

Après avoir créé la clé USB bootable, on passe maintenant à l'installation de Ubuntu.

Tout d'abord il faut insérer la clé dans la prise USB avant le redémarrage du système, après démarrage il faut choisir la méthode dont il faut démarrer notre machine ou tout simplement le « Boot mode », et pour cela on choisit le nom de la clé déjà créé avec Refus, comme le montre la figure suivante :

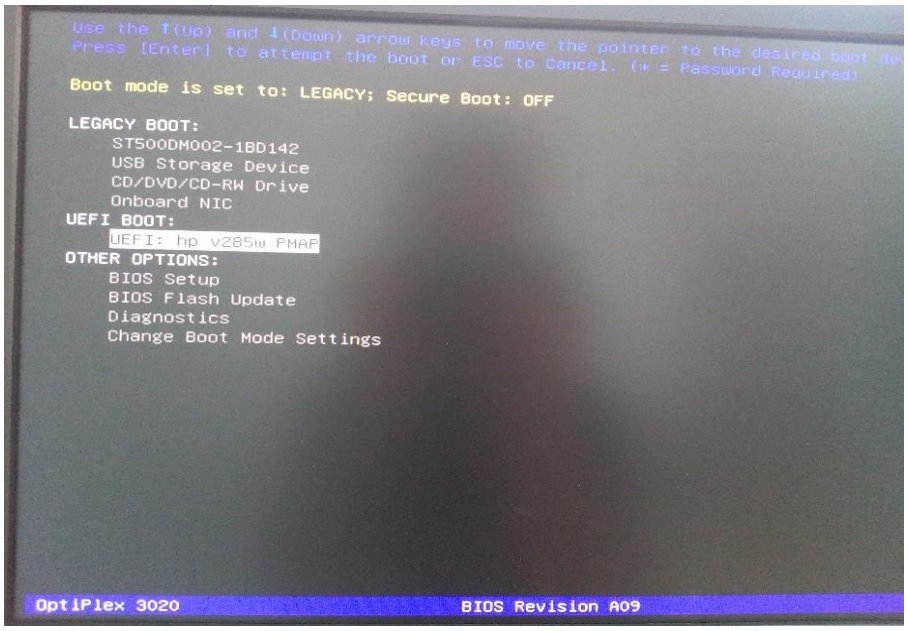

Figure 14. Boot Mode

Ensuite après le démarrage il faut choisir Install Ubuntu et ensuite suivre les configurations nécessaires pour l'installation de ce système d'exploitation.

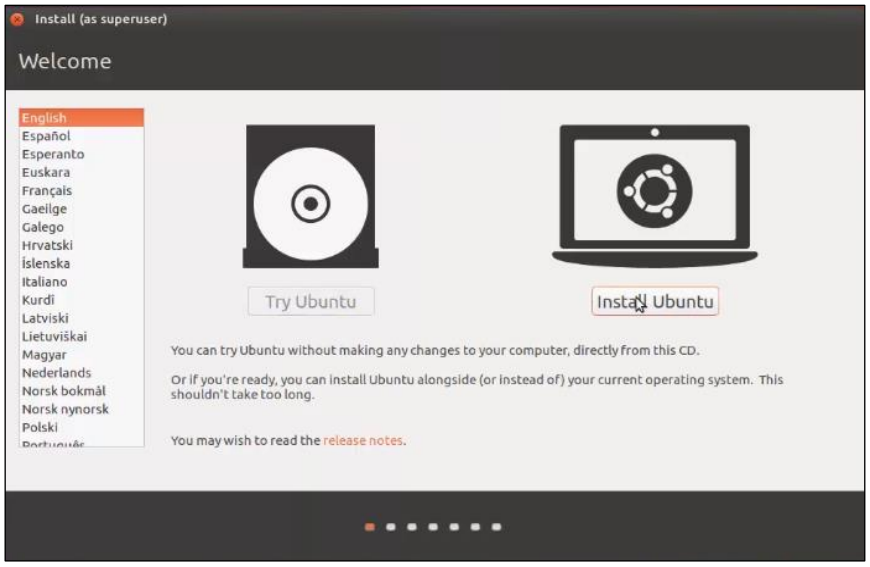

Figure 15. Installation de Ubuntu

L'étape suivante est la configuration nécessaire des répartitions et de stockage spécifié pour ce système. Une fois l'installation est terminée il faut redémarrer la machine.

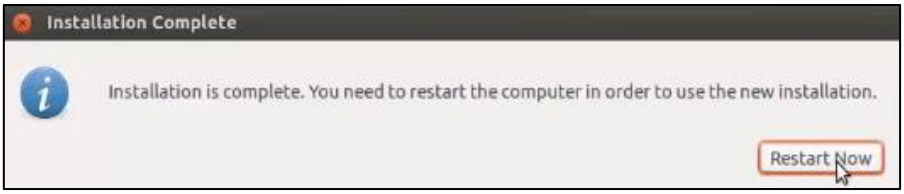

Figure 16. Redémarrage de la machine

Après avoir terminé l'installation de Ubuntu 16.04.1 LTS sur tous les postes on passe maintenant à l'installation des outils nécessaire que nous utiliserons pour la réalisation de notre Cluster Hadoop. Et pour ce faire on utilisera des commandes linux sur le terminal de Ubuntu.

2.2. Installation de Java

Pour installer Java sur le système d'exploitation Ubuntu ils suffit tout simplement d'utiliser le terminal, et cela avec la commande suivante :

```
$ sudo apt-get install default-jdk
```

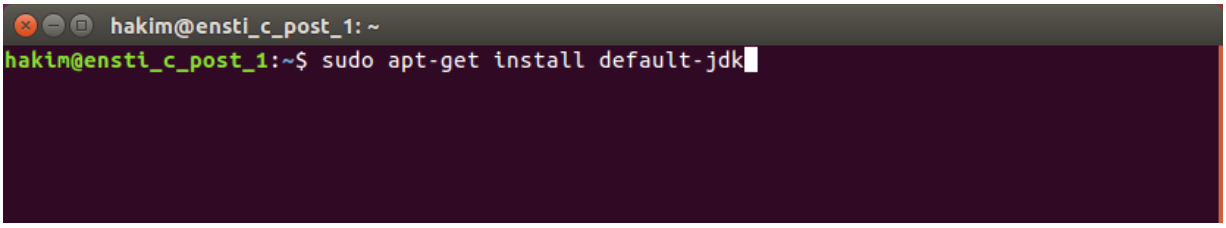

Figure 17. Installation de Java

Et pour vérifier l'installation de Java ainsi que la version installée, on utilise la commande.

```
$ java -version
```

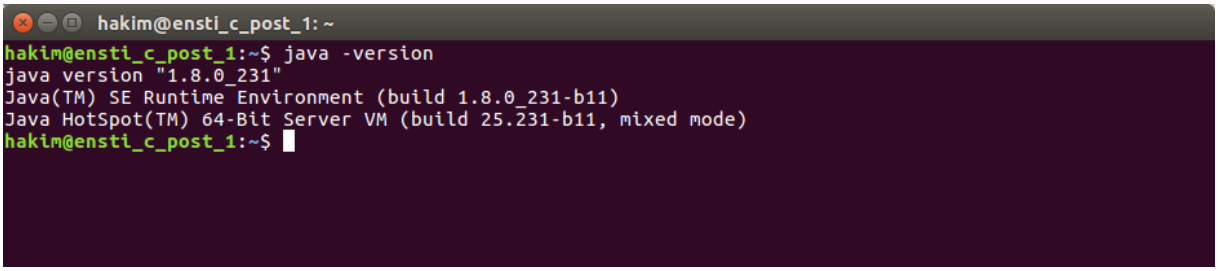

Figure 18. Version de Java

2.3. Installation de Scala

Pour se faire on utilise le terminal avec cette commande :

```
$ sudo apt-get install scala
```

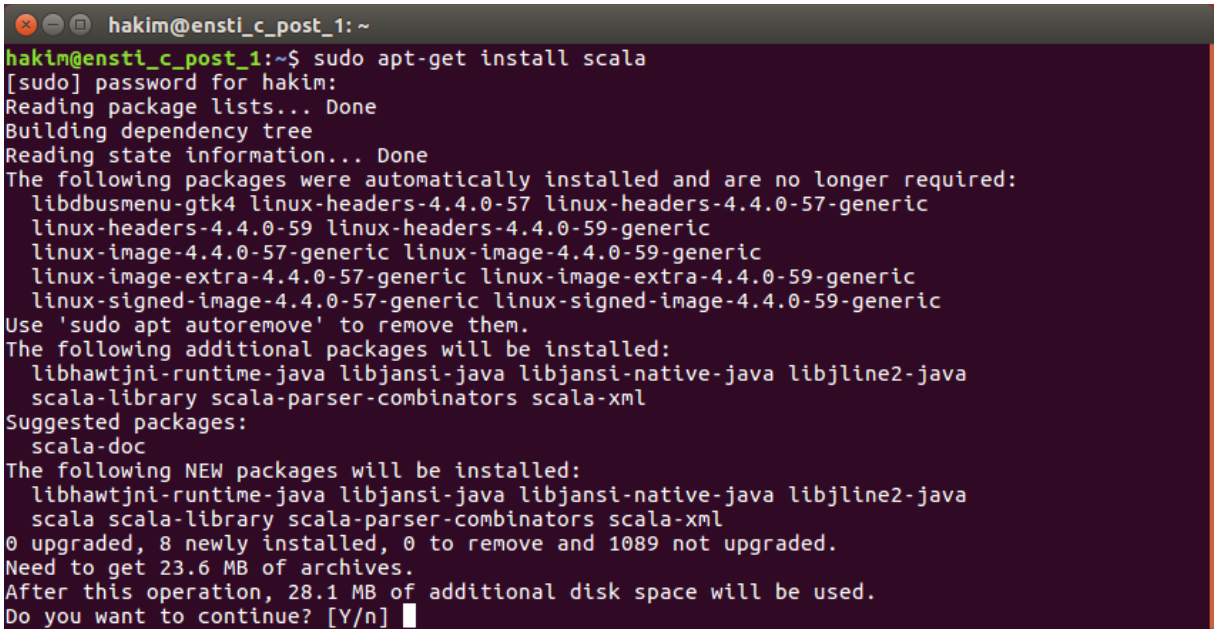

Figure 19. Installation de Scala

C'est aussi la même chose que Java pour vérifier que scala est correctement installé, on utilise la commande :

```
$ scala -version
```

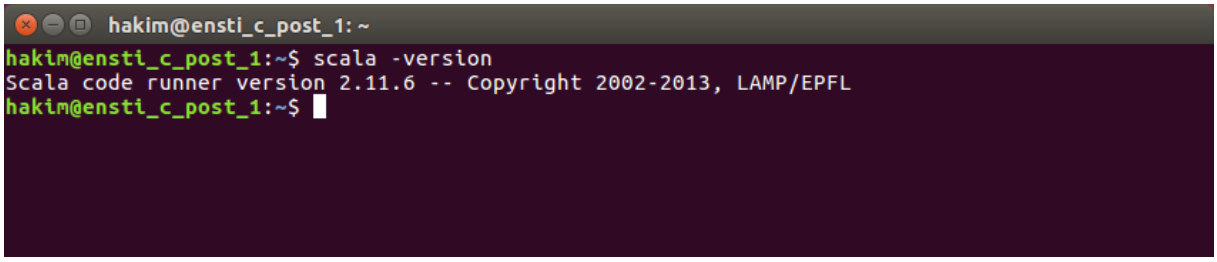

Figure 20. Version de Scala

2.4. Installation de spark-2.0.2-bin-hadoop2.6

On passe maintenant à l'installation de Spark, il faut d'abord télécharger Spark pour Ubuntu sur le site officiel de Spark qui est *''spark.apache.org/downloads''*. Après avoir terminé le téléchargement, il faut extraire le fichier qui est étant compressé avec le terminal, après avoir accéder au répertoire du fichier téléchargé de la manière suivante :

```
$ tar -xvf spark-2.0.2-bin-hadoop2.6.tgz
```

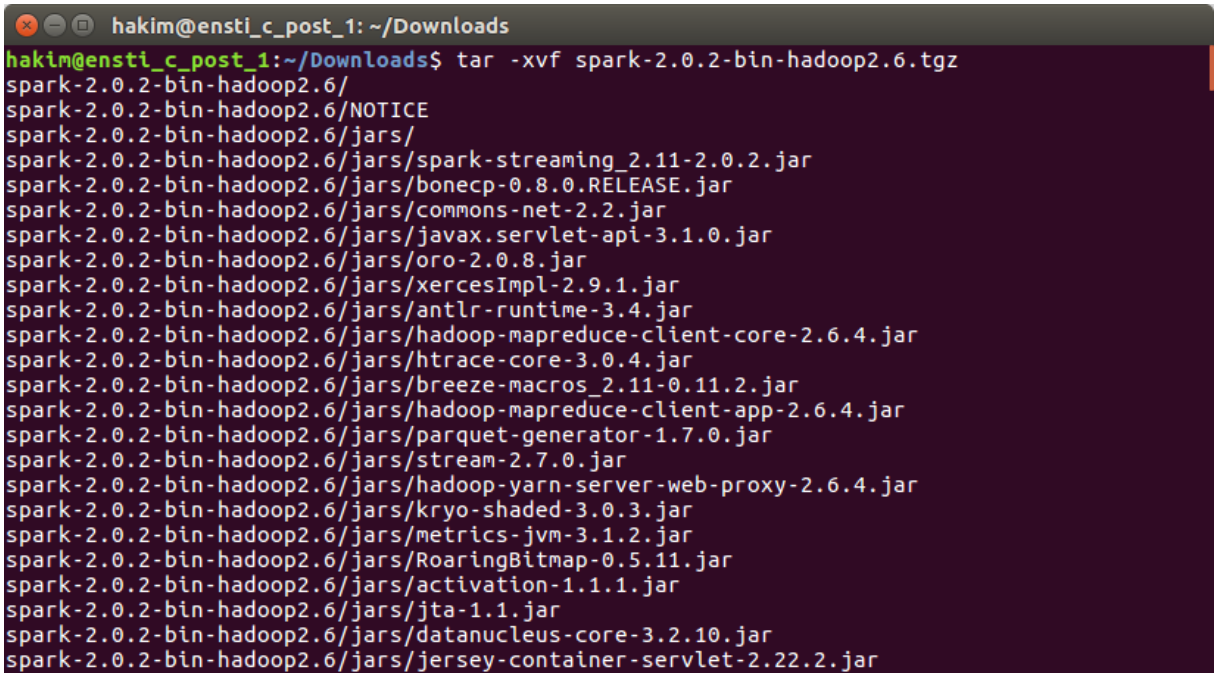

Figure 21. Extraction de Spark

Une fois l'extraction est terminée nous déplacerons le fichier obtenu de l'extraction dans le répertoire */opt/spark*, c'est dans ce répertoire qu'on va installer et configurer les outils nécessaires pour notre Cluster, on utilise toujours le terminal avec la commande :

```
$ sudo mv spark-2.0.2-bin-hadoop2.6 /opt/spark
```

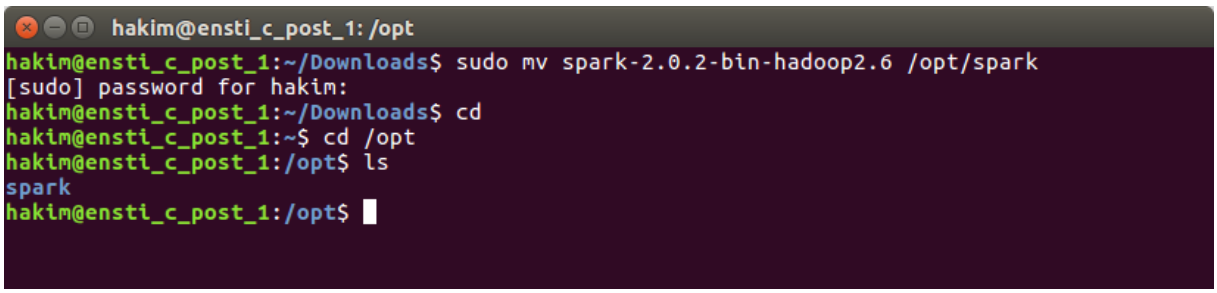

Figure 22. Déplacement de Spark

Ensuite il faut rajouter les variables d'environnement dans le fichier système *''.bashrc''* pour spécifier l'emplacement de Spark.

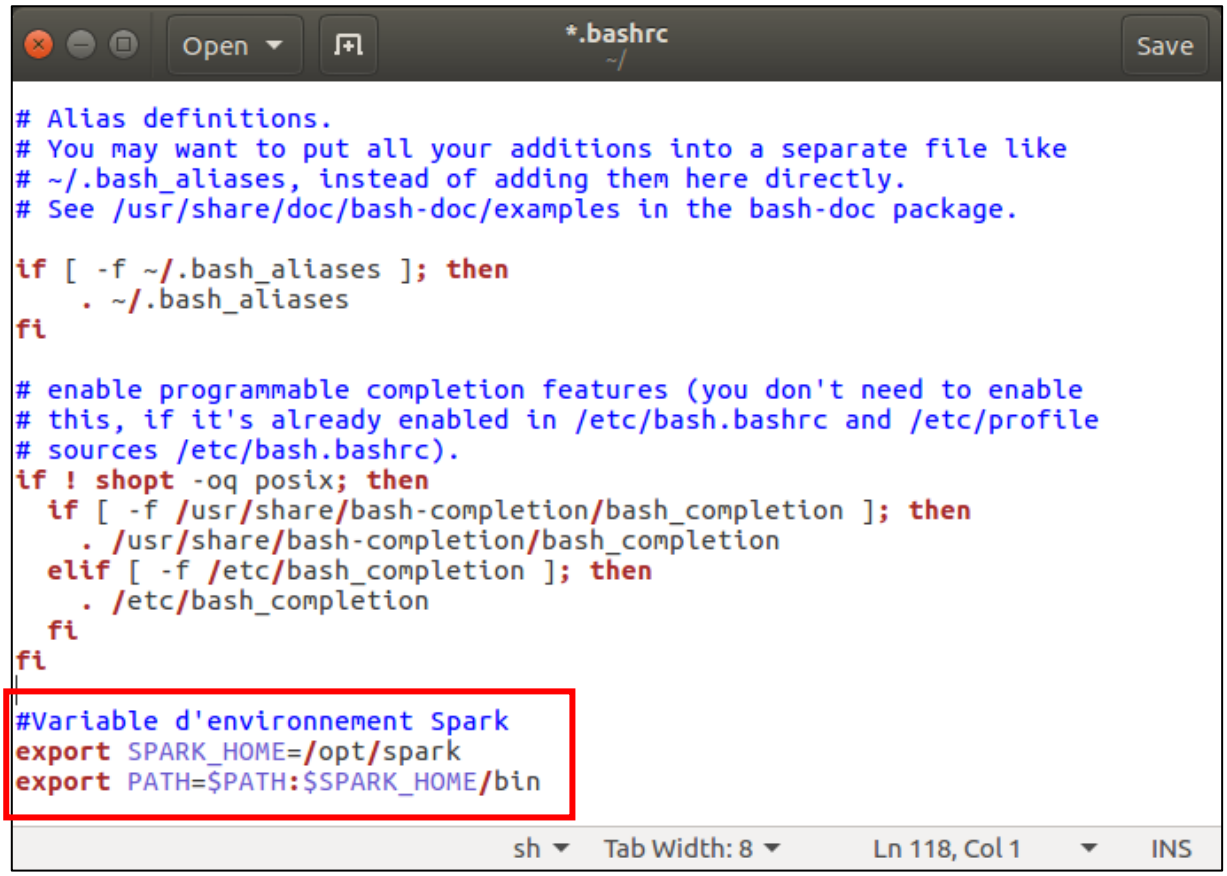

Figure 23. Variable d'environnement Spark

Enfin on vérifie l'installation et le fonctionnement de Spark en le démarrant, après avoir accéder à son répertoire on utilise la commande :

```
$ spark-shell
```

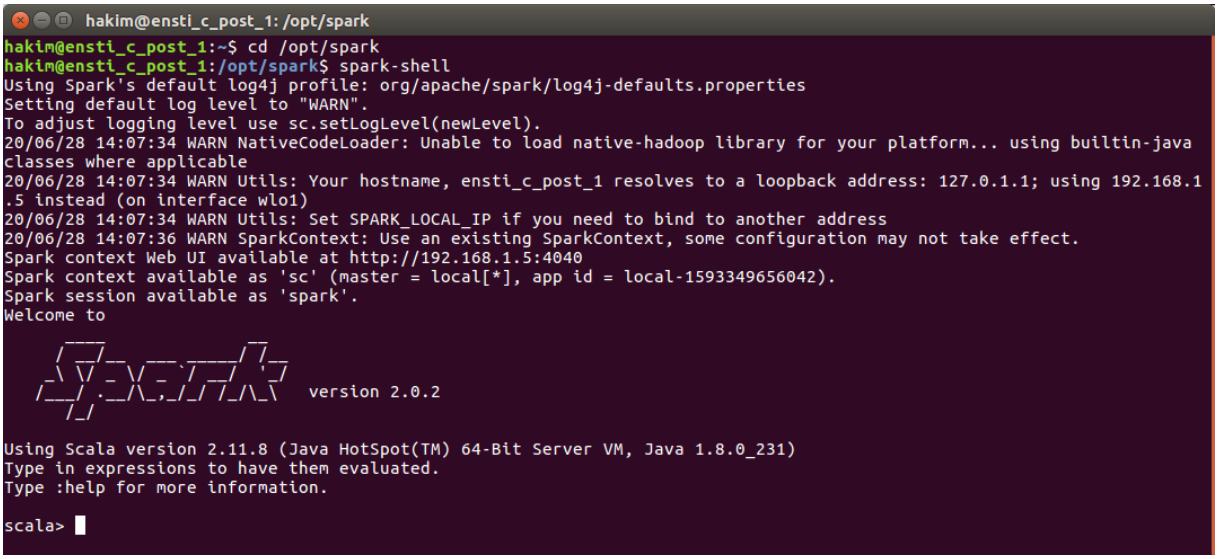

Figure 24. Version de Spark

On peut aussi utiliser l'interface graphique, en utilisant un navigateur web et l'adresse indiqué sur le terminal dans ce cas-là *''http://192.168.1.5:4040''*.

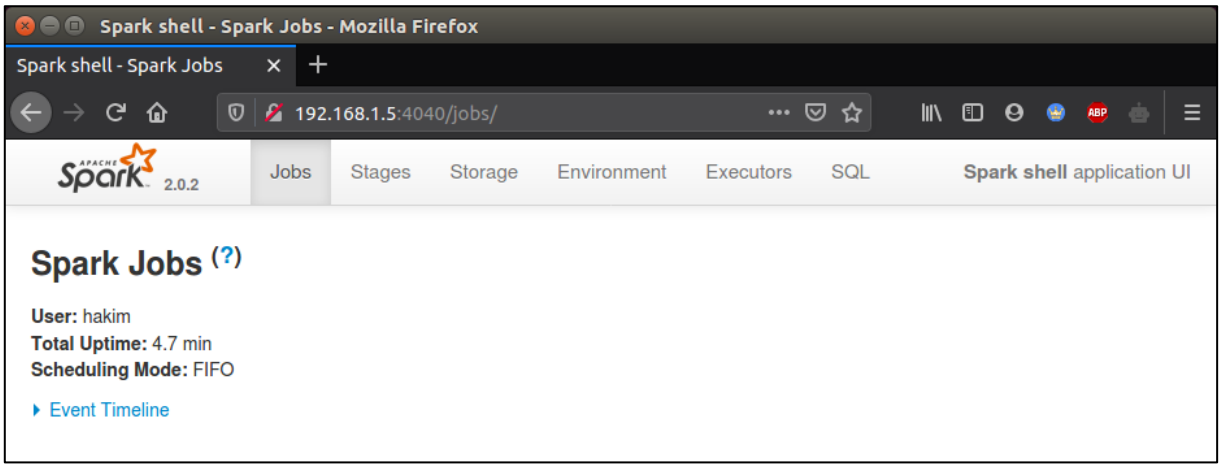

Figure 25. Interface graphique de Spark

2.5. Installation de Hadoop 2.0.6

Avant de commencer l'installation de Hadoop il faut d'abord commencer par la configuration des adresses IP de chaque poste. Pour ce faire on a attribué des adresse IP manuellement pour chaque machine, et on les a attribués de la façon suivante :

Identifiant du poste	Adresse IP du poste
enstipost1	10.2.1.1
enstipost2	10.2.1.2
enstipost3	10.2.1.3
enstipost4	10.2.1.4
enstipost5	10.2.1.5
enstipost6	10.2.1.6
enstipost7	10.2.1.7
enstipost8	10.2.1.8
enstipost9	10.2.1.9
enstipost10	10.2.1.10

Tableau 1. Affectation des adresses IP des postes

Après avoir bien vérifier que le système d'exploitation est bien installé sur toute les machines, et les adressé IP sont bien attribuées, et que tous les postes sont bien relié au même réseau, on passe aux configurations nécessaires.

Maintenant qu'on a un réseau configuré, il faut attribuer des noms d'hôtes (l'identifiant de chaque poste) à chacune des adresses IP. Cela se fait dans le fichier *''/etc/hosts''* de toutes les machine on rajoute les lignes suivante :

```
10.2.1.1   enstipost1
10.2.1.2   enstipost2
10.2.1.3   enstipost3
10.2.1.4   enstipost4
10.2.1.5   enstipost5
10.2.1.6   enstipost6
10.2.1.7   enstipost7
10.2.1.8   enstipost8
10.2.1.9   enstipost9
10.2.1.10  enstipost10
```

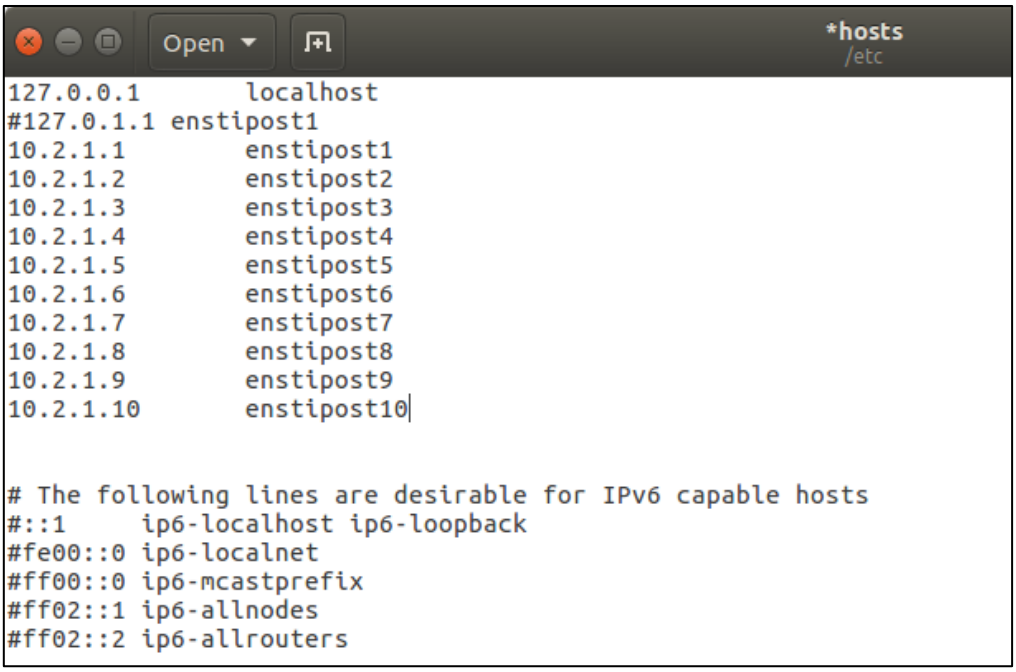

Figure 26. Configuration des adresses IP (le fichier hosts)

Hadoop nécessite un accès SSH pour gérer ses nœuds, nous devons donc installer OpenSSH sur toute les machines, et on procède comme suit :

Avec la commande suivante on installe OpenSSH sur tous les nœuds.

```
$ sudo apt-get install openssh-server
```

Maintenant, nous devons générer une clé SSH sur la machine maître (*enstipost1* dans notre cas). Lorsqu'il demande de saisir un nom de fichier pour enregistrer la clé, il ne faut donner aucun nom, on appuis simplement sur Entrée. On utilise cette commande :

```
$ ssh-keygen -t rsa -P ""
```

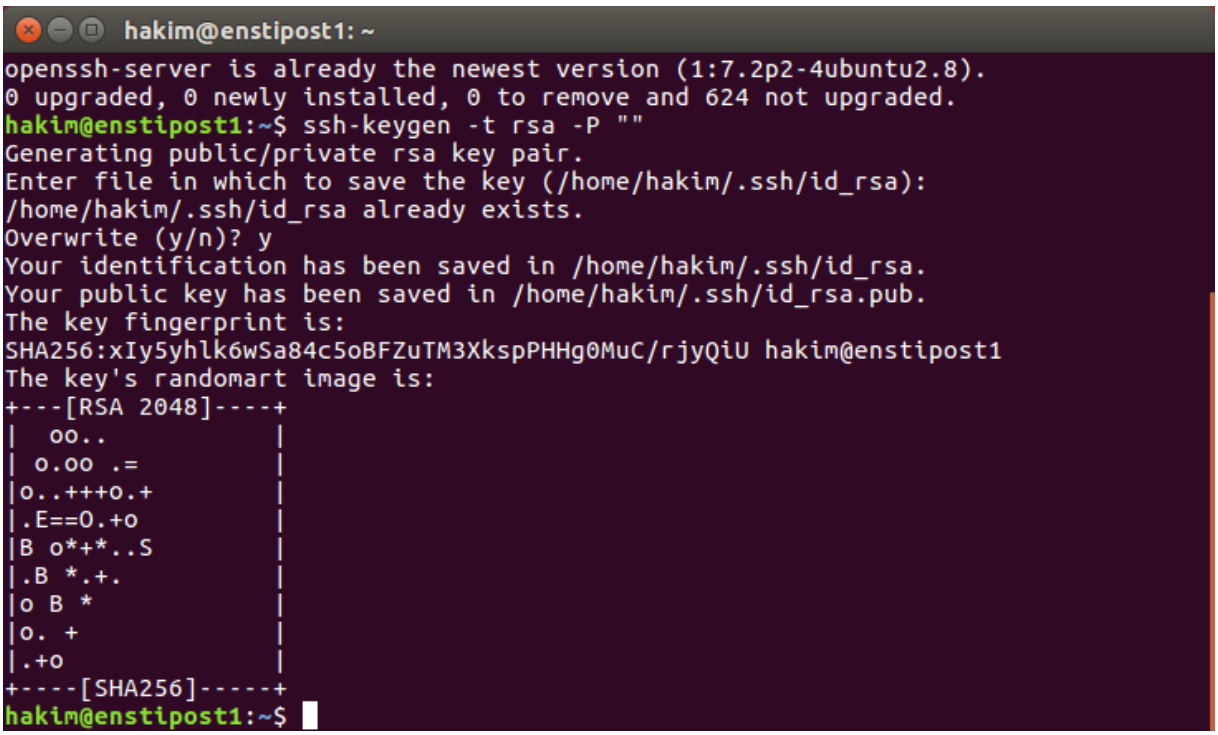

Figure 27. Installation de SSH

Ensuite, nous devons activer l'accès SSH à notre machine maître (*enstipost1*) avec cette clé nouvellement créée.

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

On teste maintenant la configuration SSH en se connectant à notre machine locale.

```
$ ssh localhost
```

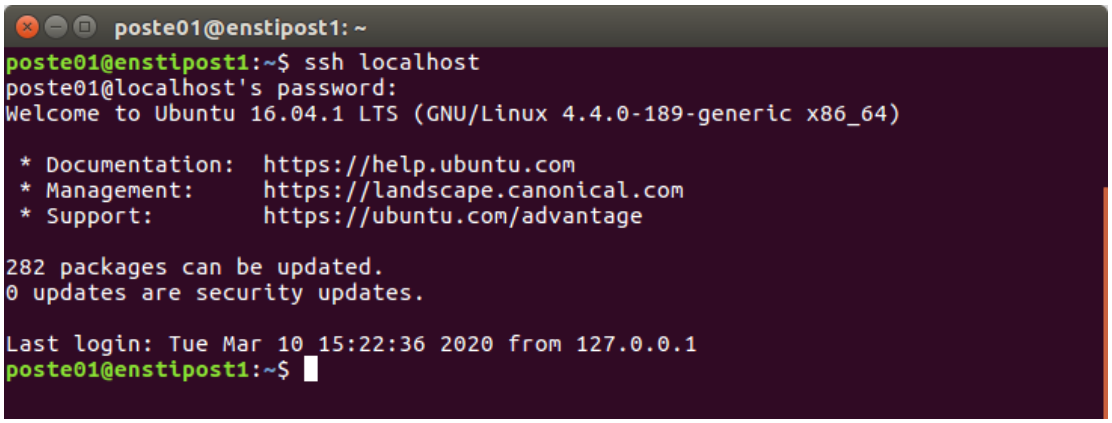

Figure 28. Test de connexion SSH

On exécute maintenant la commande ci-dessous pour envoyer la clé publique générée sur le maître (*enstipost1*) vers tous les esclaves (tous les postes restant).

```
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste02@enstipost2
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste03@enstipost3
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste04@enstipost4
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste05@enstipost5
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste06@enstipost6
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste07@enstipost7
```

```
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste08@enstipost8
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste09@enstipost9
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub poste10@enstipost10
```

Maintenant que le maître et les esclaves ont la clé publique, on peut donc également connecter le maître au maître et le maître à tous les esclaves.

Le maître au maître :

```
$ ssh poste01@enstipost1
```

Un exemple d'esclave :

```
$ ssh poste02@enstipost2
```

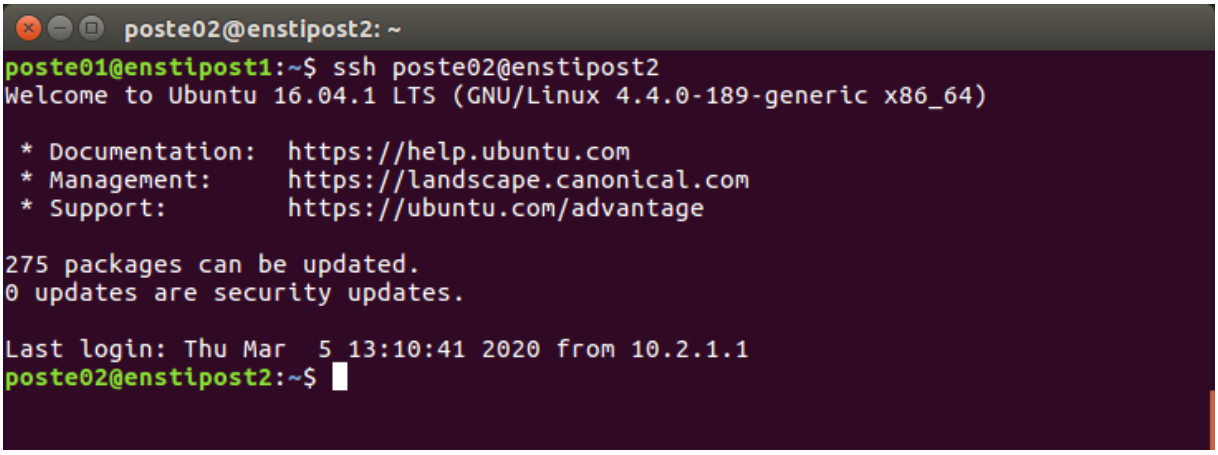

Figure 29. Connexion du maître à un esclave avec SSH

Maintenant de cette manière on pourra se connecter sur n'importe quel nœud de notre cluster en utilisant seulement le nœud maitre (*enstipot1*).

On passe donc maintenant à l'installation de Hadoop, comme première étape nous devant tout d'abord télécharger Hadoop sur le site officiel de Appache Hadoop *''https://hadoop.apache.org/''* en choisissant la version souhaitée, pour notre cas c'est la version *hadoop-2.6.0* comme c'est bien cité auparavant.

Une fois le téléchargement est terminé on recopie le ficher télécharger sur tous les nœuds, pour éviter de le téléchargé sur chaque machine en utilisant simplement la commande :

```
$ scp hadoop-2.6.0.tar.gz poste02@enstipost2:/Downloads
$ scp hadoop-2.6.0.tar.gz poste03@enstipost3:/Downloads
$ scp hadoop-2.6.0.tar.gz poste04@enstipost4:/Downloads
$ scp hadoop-2.6.0.tar.gz poste05@enstipost5:/Downloads
$ scp hadoop-2.6.0.tar.gz poste06@enstipost6:/Downloads
$ scp hadoop-2.6.0.tar.gz poste07@enstipost8:/Downloads
$ scp hadoop-2.6.0.tar.gz poste08@enstipost8:/Downloads
$ scp hadoop-2.6.0.tar.gz poste09@enstipost9:/Downloads
$ scp hadoop-2.6.0.tar.gz poste10@enstipost10:/Downloads
```

Maintenant il faut extraire le fichier télécharger *hadoop-2.6.0.tar.gz*, sur chacun des 10 postes comme suit :

```
$ tar -xvf hadoop-2.6.0.tar.gz
```

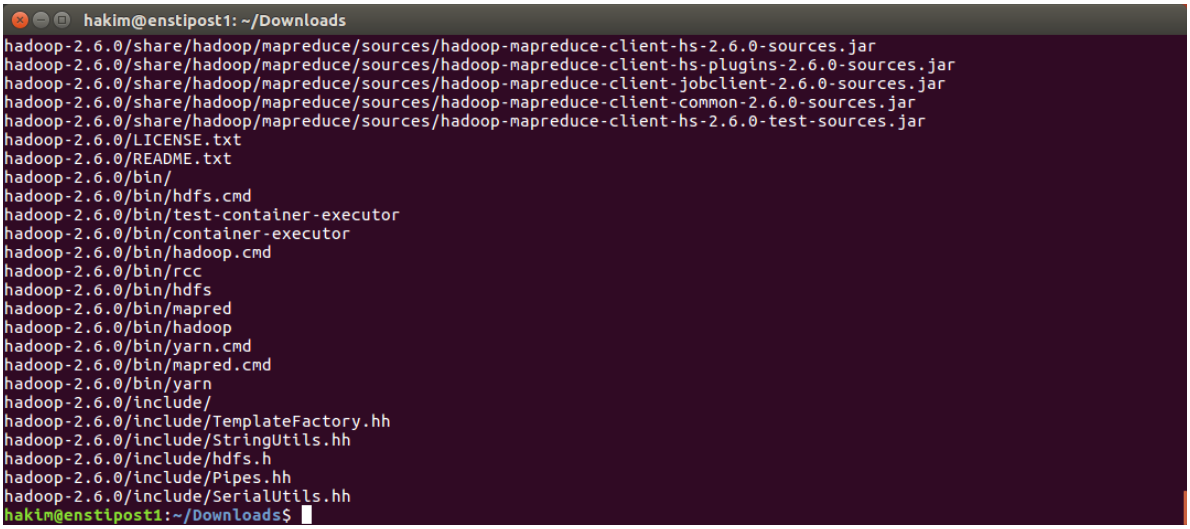

Figure 30. Extraction de Hadoop

Après il suffit de déplacer le contenu extrait dans un répertoire spécifique ou on veut installer Hadoop, pour nous on a choisi le répertoire */opt/hadoop* tout comme Spark.

```
$ mv hadoop-2.6.0 /opt/hadoop
```

Avant de définir les configurations nécessaires pour Hadoop, nous devant d'abord définir les variables d'environnement dans le fichier *''.bashrc''*, en ajoutant les lignes suivantes :

```
#Hadoop
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR=/opt/hadoop
export HADOOP_MAPRED_HOME=/opt/hadoop
export HADOOP_COMMON_HOME=/opt/hadoop
export HADOOP_HDFS_HOME=/opt/hadoop
export YARN_HOME=/opt/hadoop
# Add Hadoop bin/ directory to PATH
export PATH=$PATH:/opt/hadoop/bin
export PATH=$PATH:/opt/hadoop/sbin
```

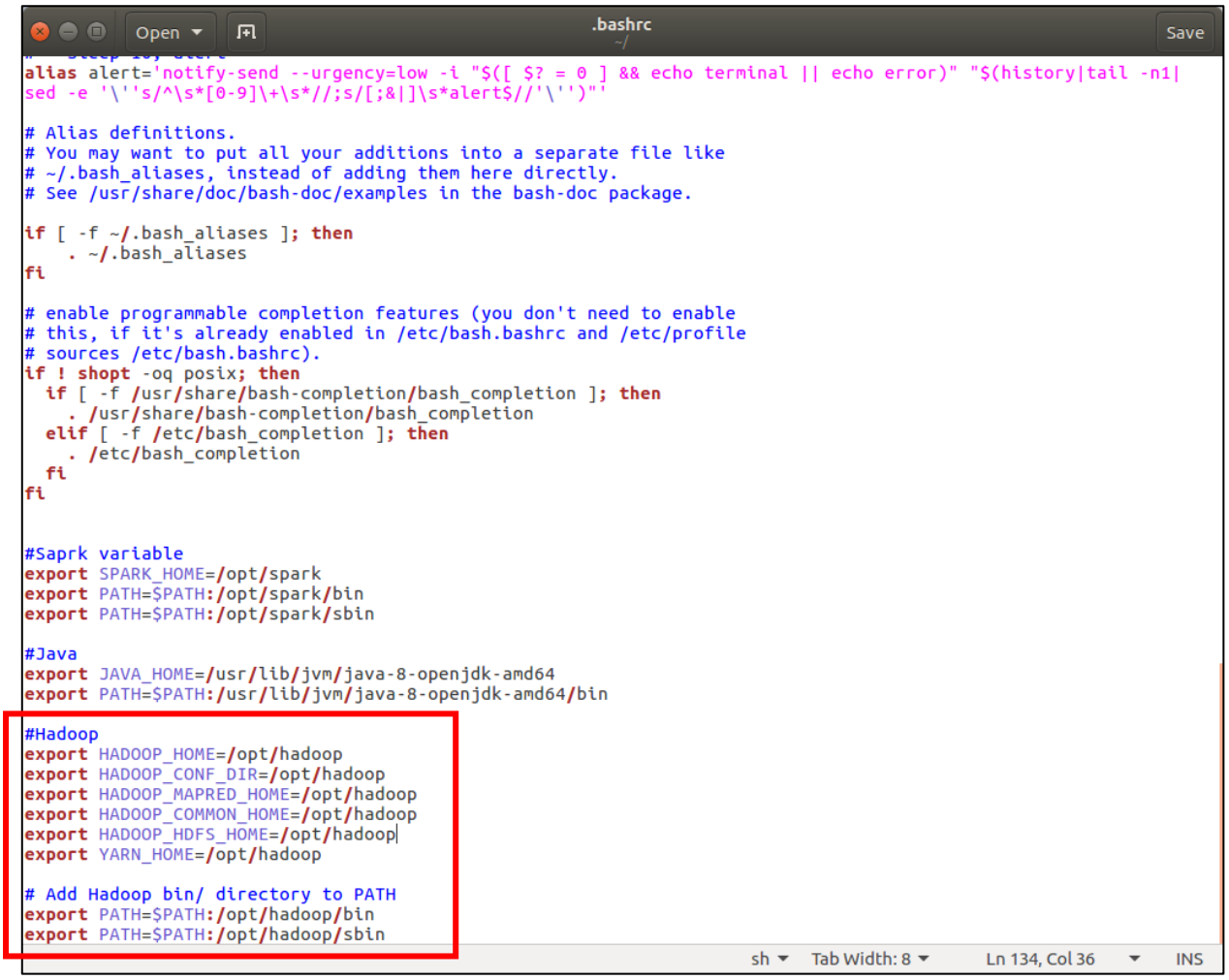

Figure 31. Variable d'environnement de Hadoop

Puis on passe à la configuration de JAVA_HOME dans *''hadoop-env.sh''*. Ce fichier spécifie les variables d'environnement qui affectent le JDK utilisé par les démons Apache Hadoop démarrés par les scripts de démarrage Hadoop, on accède sur le répertoire du ficher ensuite on modifie.

```
$ cd /opt/hadoop/etc/hadoop/

$ sudo gedit hadoop-env.sh
```

Puis on ajoute la ligne suivante :

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

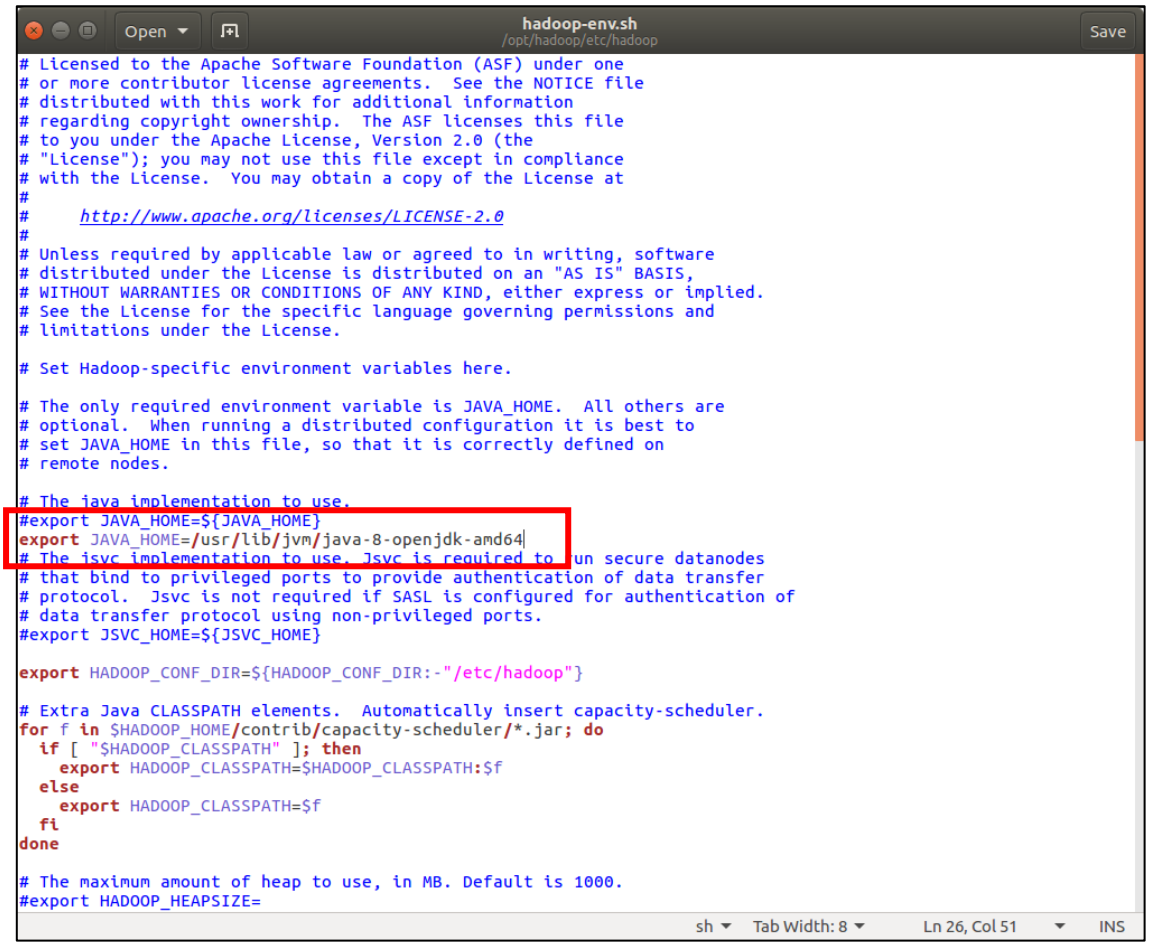

Figure 32. Variable d'environnement de Java dans Hadoop

Enfin pour terminer l'installation de Hadoop et avant de passer à la configuration, il faut ajouter un nouvel utilisateur *ensti_ cluster*, il faut aussi lui donner les autorisations appropriées, *ensti_ cluster* sera l'utilisateur spécifique pour Hadoop, seulement avec cet utilisateur que nous pouvant utiliser Hadoop (sur toutes les machines) :

```
$ adduser ensti_cluster
$ usermod -aG ensti_cluster ensti_cluster
$ chown ensti_cluster:root -R /opt/hadoop
$ chmod g+rwx -R /opt/hadoop
```

Pour finir nous allons créer des répertoires *NameNode* sur le maitre (*enstipost1*) et *DataNode* sur les tous les nœuds esclaves, mais pour y est faire il faut d'abord accéder à l'utilisateur *ensti_ cluster* pour que tous les autres utilisateurs du système d'exploitation n'auront aucun accès sur les fichiers de notre cluster Hadoop :

Sur le nœud maitre (*enstipost1*) :

```
$ su ensti_cluster
$ mkdir -p /opt/hadoop/hadoop2_data/hdfs/namenode
```

Sur tous les nœuds esclaves :

```
$ su ensti_cluster
$ mkdir -p /opt/hadoop/hadoop2_data/hdfs/datanode
```

Maintenant que notre réseau est bien configuré et l'installation de tous les outils est bien faite sur chacun des postes on peut passer aux configurations requises.

3. Configuration des environnements

3.1. Configuration de Hadoop

Hadoop possède de nombreux fichiers de configuration, qui doivent être configurés selon les exigences de notre Cluster Hadoop. Configurons les fichiers de configuration Hadoop un par un qui se trouve dans le répertoire *''/opt/hadoop/etc/hadoop/''*, on fait la configuration sur le nœud maitre (*enstipost1*), et toujours en utilisant le nouvel utilisateur créé *ensti_ cluster*.

```
$ su ensti_cluster
```

Après on commence la configuration des fichiers un par un comme suit :

* Le fichier *core-site.xml*

```
$ sudo gedit core-site.xml
```

On fait la configuration suivante :

```
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://enstipost1:9000</value>
</property>
</configuration>
```

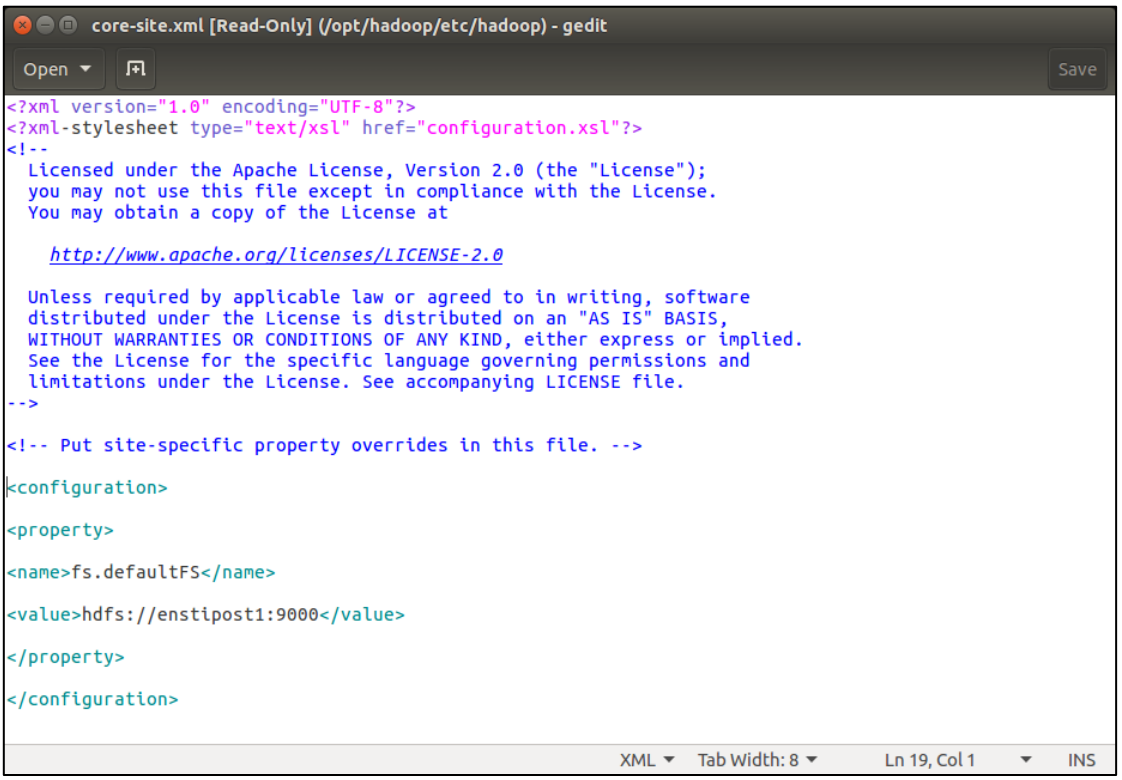

Figure 33. Le fichier core-site.xml

* Le fichier *hdfs-site.xml*

```
$ sudo gedit hdfs-site.xml
```

On ajoute les lignes suivante dans le fichier de configuration :

```
<configuration>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>/opt/hadoop/hadoop2_data/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>/opt/hadoop/hadoop2_data/hdfs/datanode</value>
</property>
</configuration>
```

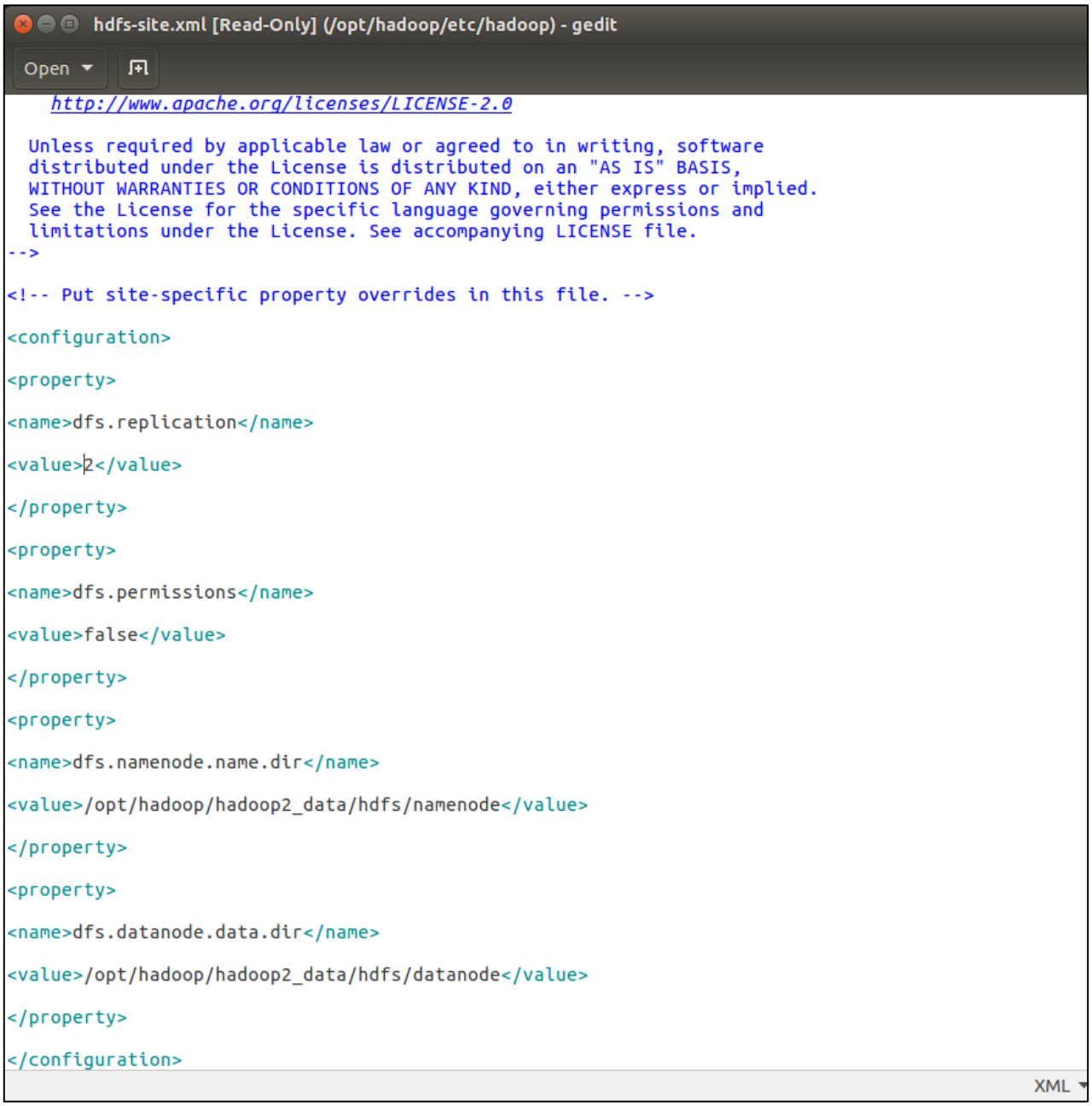

Figure 34. Le fichier hdfs-site.xml

* Le fichier *yarn-site.xml*

```
$ sudo gedit yarn-site.xml
```

Le fichier doit contenir la configuration suivante :

```
<configuration>
<property>
<name>yarn.acl.enable</name>
<value>0</value>
</property>
<property>
<name>yarn.resourcemanager.hostname</name>
```

```
<value>enstipost1</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
</configuration>
```

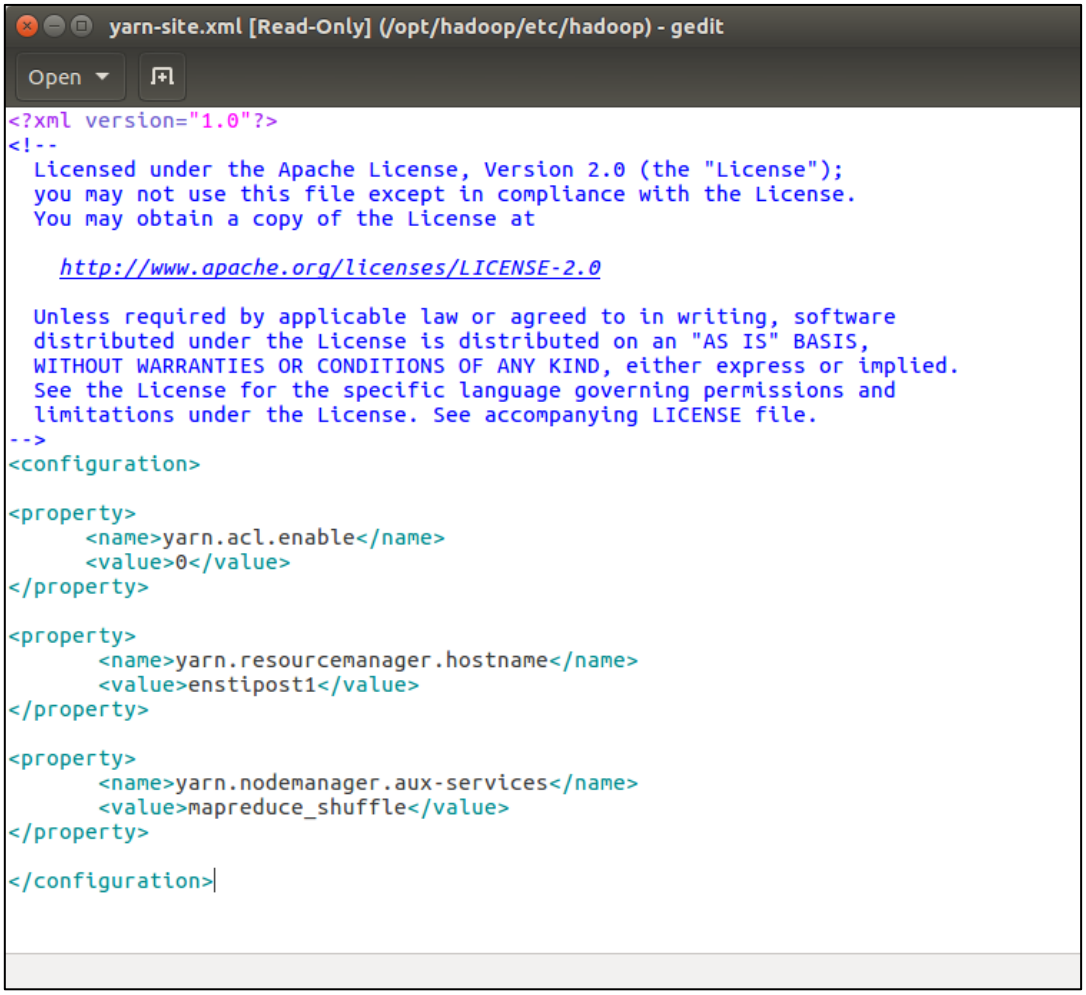

Figure 35. Le fichier yarn-site.xml

* Le fichier *mapred-site.xml*

Pour le fichier *mapred-site.xml* il n'existe pas dans les fichiers de configuration de Hadoop donc on va en créer mais à partir d'un autre fichier *''mapred-site.xml.template''* on utilisant la commande :

```
$ cp mapred-site.xml.template mapred-site.xml
```

Après on pourra ajouter notre configuration dans le fichier créé.

```
$ sudo gedit mapred-site.xml
```

Puis on rajoute les lignes qui suit :

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

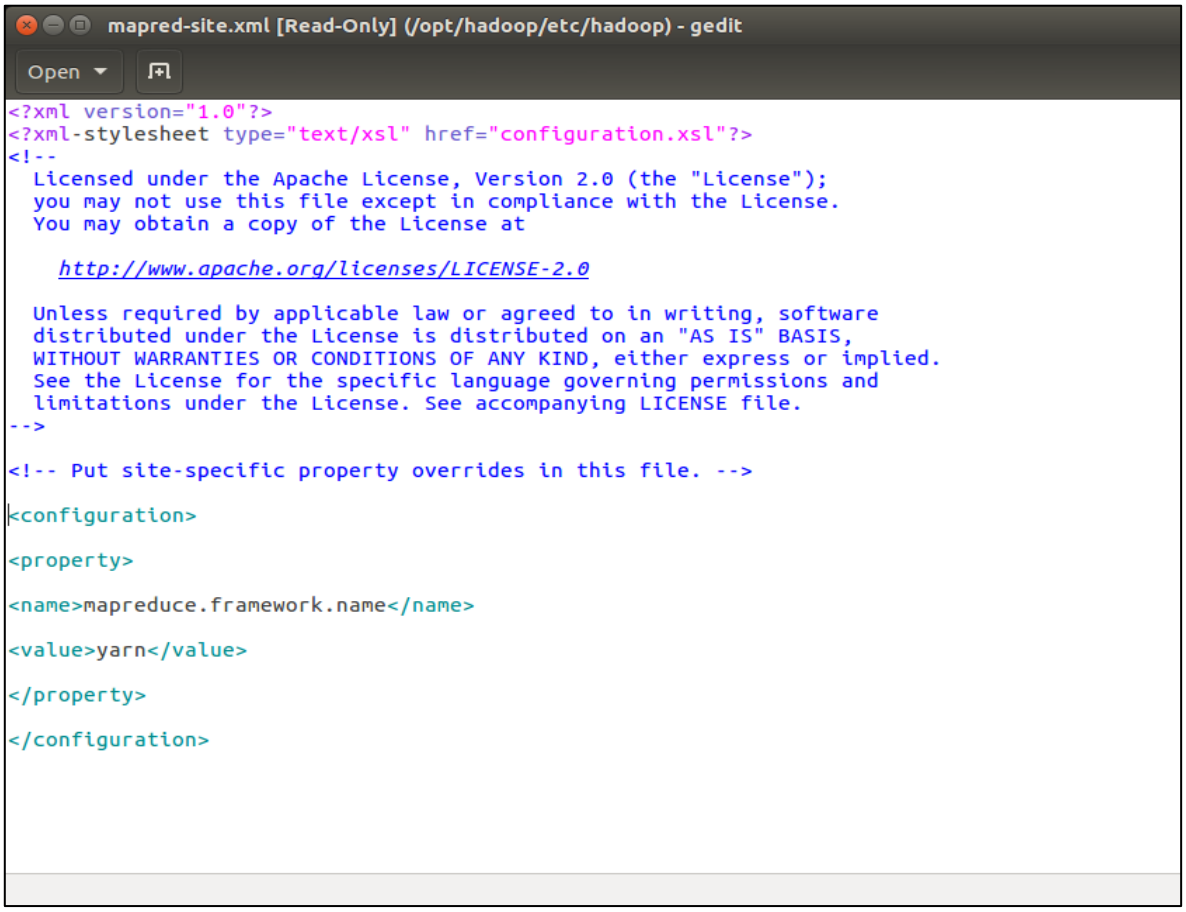

Figure 36. Le fichier mapred-site.xml

Une fois fini avec les fichiers de configuration il faut ajouter deux fichiers *''masters.txt''* et *''slaves.txt''* où nous allons déclarer tous les nœuds maitres sur *''masters.txt''* et tous les nœuds esclaves sur *''slaves.txt''* et ceci dans le même répertoire que les fichiers de configuration *''/opt/hadoop/etc/hadoop/''*.

* Le fichier *masters.txt*

```
$ sudo gedit masters.txt
```

On rajoute l'identifiant du poste maitre, dans notre cas on a un seul nœud maitre qui est le poste 01 :

```
enstipost1
```

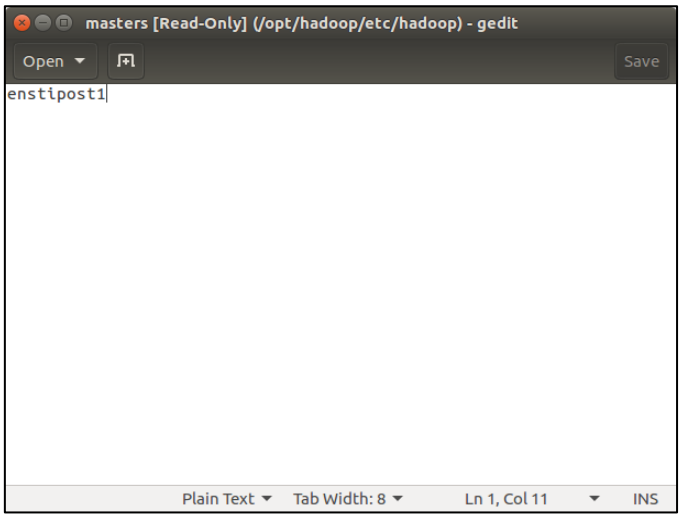

Figure 37. Le fichier masters.txt

* Le fichier *slaves.txt*

```
$ sudo gedit slaves.txt
```

On ajoute l'identifiant de tous les postes esclaves, dans notre cas les neufs (09) postes restants :

```
enstipost2
enstipost3
enstipost4
enstipost5
enstipost6
enstipost7
enstipost8
enstipost9
enstipost10
```

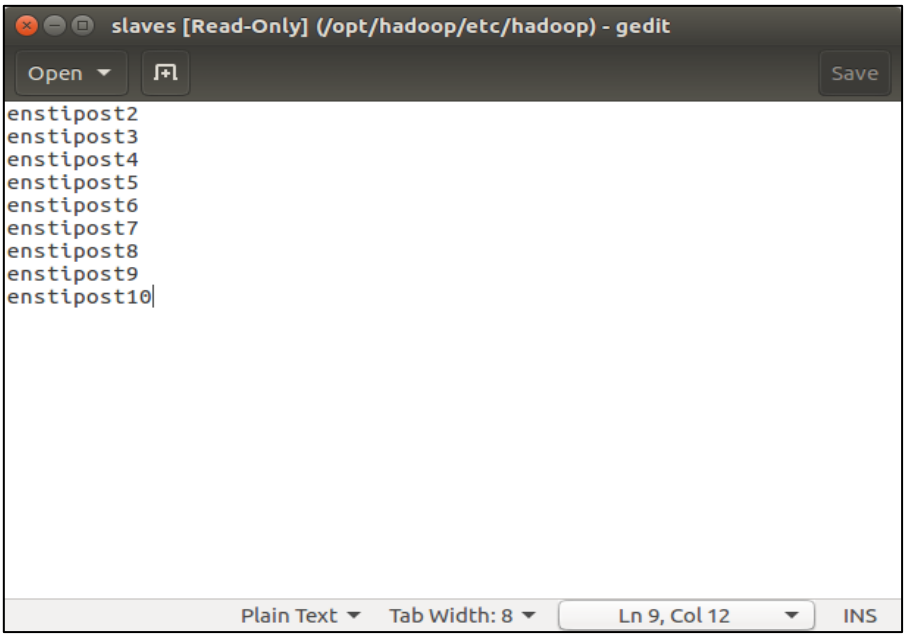

Figure 38. Le fichier slaves.txt

De cette manière notre configuration est terminée sur le nœud master *''enstipost1''*, du coup pour configurer les nouds slaves il suffit tout simplement de copier tous les fichiers de configuration déjà configurés sur les nœuds slaves en utilisant la commande suivante :

```
$ scp /opt/hadoop/etc/hadoop/* enstipost2:/opt/hadoop/etc/hadoop/
```

La même chose pour recopier dans tous autres les postes slaves :

```
$ scp /opt/hadoop/etc/hadoop/* enstipost3:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost4:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost5:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost6:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost7:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost8:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost9:/opt/hadoop/etc/hadoop/
$ scp /opt/hadoop/etc/hadoop/* enstipost10:/opt/hadoop/etc/hadoop/
```

Après avoir installé et configuré Hadoop sur toutes les machines, la première chose à faire pour démarrer notre cluster Hadoop est de formater le système de fichiers Hadoop, qui est implémenté au-dessus des systèmes de fichiers locaux de notre Cluster. Ceci est requis lors de la première installation de Hadoop. Il ne faut pas formater un système de fichiers Hadoop en cours d'exécution, cela effacera toutes les données HDFS.

Bien sûr en utilisant toujours le même utilisateur *''ensti_cluster''*, on accède au répertoire de Hadoop *''/opt/hadoop''* et on exécute la commande suivante :

```
$ hadoop namenode -format
```

```
ensti_cluster@enstipost1: /opt/hadoop
20/09/14 17:41:12 INFO util.GSet: VM type       = 64-bit
20/09/14 17:41:12 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB
20/09/14 17:41:12 INFO util.GSet: capacity      = 2^20 = 1048576 entries
20/09/14 17:41:12 INFO namenode.NameNode: Caching file names occuring more than 10 times
20/09/14 17:41:12 INFO util.GSet: Computing capacity for map cachedBlocks
20/09/14 17:41:12 INFO util.GSet: VM type       = 64-bit
20/09/14 17:41:12 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB
20/09/14 17:41:12 INFO util.GSet: capacity      = 2^18 = 262144 entries
20/09/14 17:41:12 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-pct = 0.9990000128746033
20/09/14 17:41:12 INFO namenode.FSNamesystem: dfs.namenode.safemode.min.datanodes = 0
20/09/14 17:41:12 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension     = 30000
20/09/14 17:41:12 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
20/09/14 17:41:12 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry
 expiry time is 600000 millis
20/09/14 17:41:12 INFO util.GSet: Computing capacity for map NameNodeRetryCache
20/09/14 17:41:12 INFO util.GSet: VM type       = 64-bit
20/09/14 17:41:12 INFO util.GSet: 0.029999999329447746% max memory 889 MB = 273.1 KB
20/09/14 17:41:12 INFO util.GSet: capacity      = 2^15 = 32768 entries
20/09/14 17:41:12 INFO namenode.NNConf: ACLs enabled? false
20/09/14 17:41:12 INFO namenode.NNConf: XAttrs enabled? true
20/09/14 17:41:12 INFO namenode.NNConf: Maximum size of an xattr: 16384
Re-format filesystem in Storage Directory /opt/hadoop/hadoop2_data/hdfs/namenode ? (Y or N) y
20/09/14 17:41:16 INFO namenode.FSImage: Allocated new BlockPoolId: BP-1164381724-10.2.1.1-1600098076921
20/09/14 17:41:17 INFO common.Storage: Storage directory /opt/hadoop/hadoop2_data/hdfs/namenode has been su
ccessfully formatted.
20/09/14 17:41:17 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
20/09/14 17:41:17 INFO util.ExitUtil: Exiting with status 0
20/09/14 17:41:17 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at enstipost1/10.2.1.1
************************************************************/
ensti_cluster@enstipost1:/opt/hadoop$
```

Figure 39. Formatage du système de fichiers Hadoop

Nous sommes donc maintenant prêts à démarrer les démons Hadoop, à savoir NameNode, DataNode, ResourceManager et NodeManager sur notre cluster Apache Hadoop.

On exécute maintenant la commande ci-dessous pour démarrer le NameNode sur la machine master et les DataNodes sur le maître et les esclaves.

```
$ start-dfs.sh
```

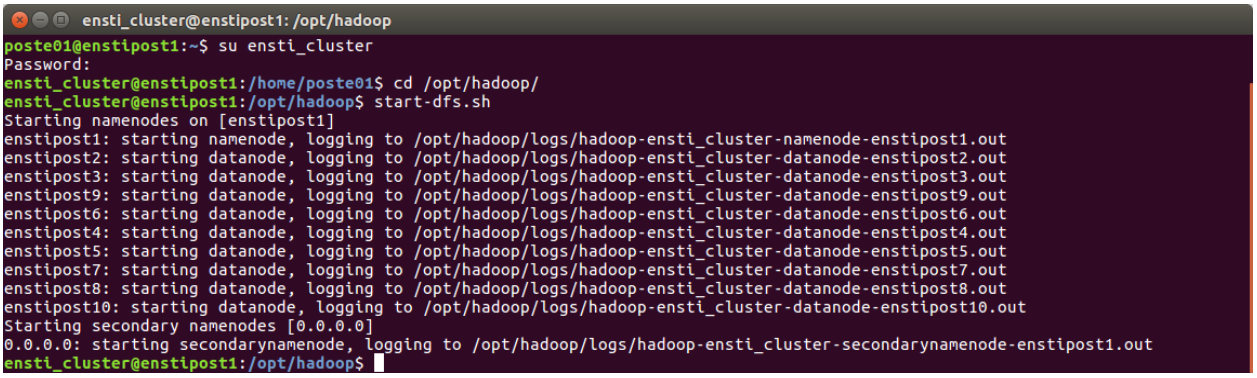

Figure 40. Démarrage de DFS

La commande ci-dessous lancera les démons YARN, ResourceManager fonctionnera sur le maître et NodeManagers fonctionnera sur le maître et l'esclave.

```
$ start-yarn.sh
```

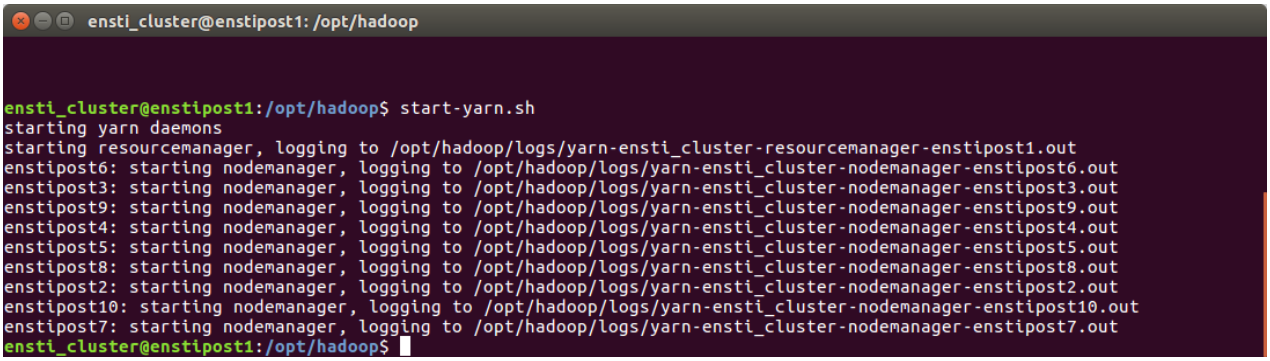

Figure 41. Démarrage de YARN

On vérifie que tous les services ont démarré correctement à l'aide de JPS (Java Process Monitoring Tool), sur le nœud maitre ainsi que tous les nœuds esclaves.

Sur le maitre :

```
$ jps
```

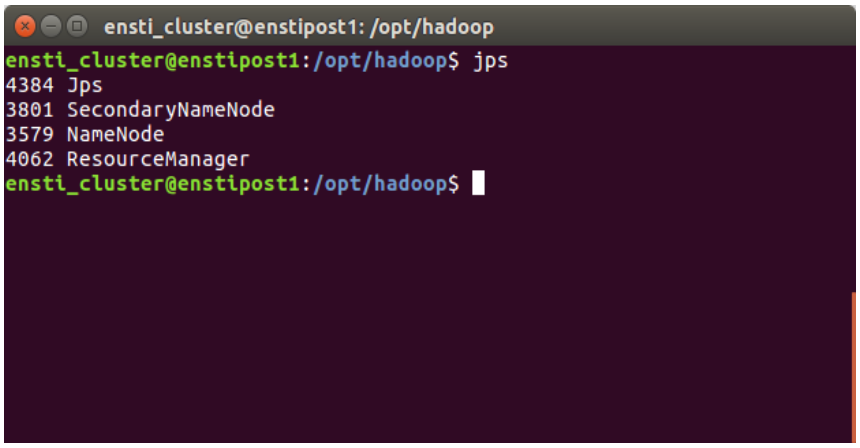

Figure 42. Vérification de démarrage des services sur le maître

Exemple sur un esclave : (le poste 02)

```
$ jps
```

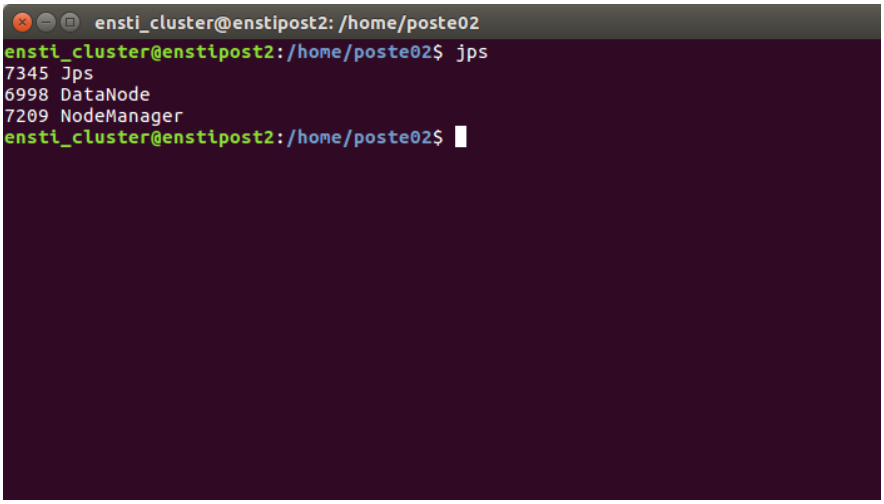

Figure 43. Vérification de démarrage des services sur un esclave

Nous pouvons vérifier les démons YARN ont démarré correctement sur tous les nœuds en exécutant cette commande :

```
$ yarn node -list
```

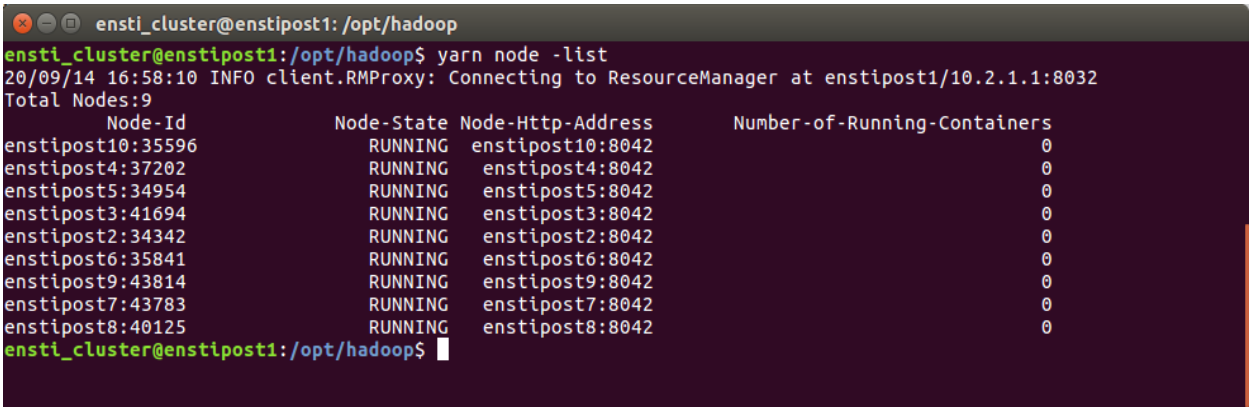

Figure 44. Vérification de démarrage de YARN

On peut voir que le nombre total de nœuds est neufs (09), et il n'y a aucun Job en cours d'exécution car nous n'avons encore démarré aucune tâche.

Maintenant sur notre navigateur Web sur le nœud maitre on peut vérifier que tout est correctement installé et lancé, en accédant l'URL *''enstipost1 :50070''*.

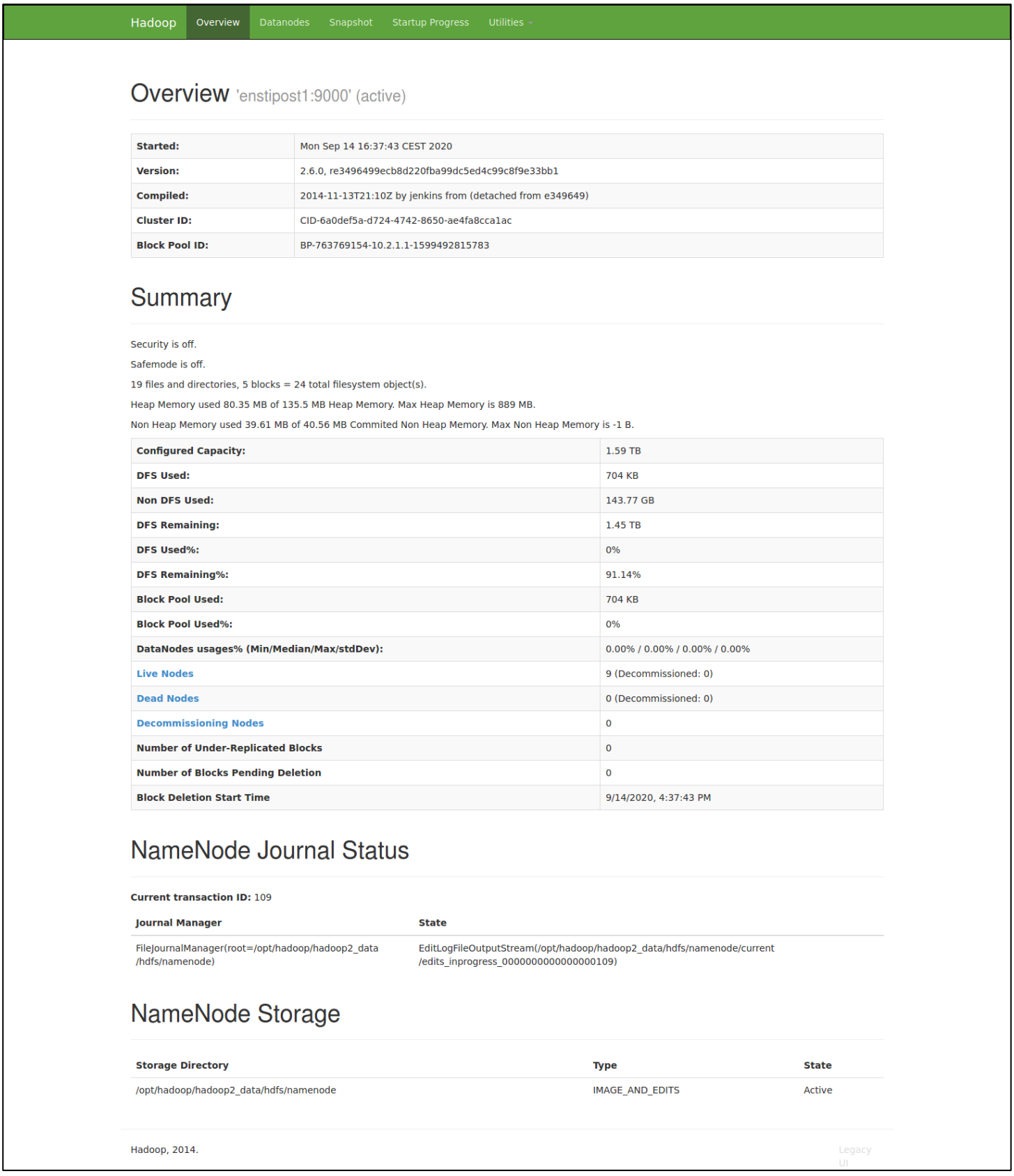

Figure 45. Interface graphique de Hadoop

On peut donc voir que le nombre de nœuds actifs est neufs (09), cela signifie que 9 DataNodes sont opérationnels et que nous avons configuré avec succès notre Cluster Hadoop multi-nœuds.

Conclusion

Dans ce chapitre nous avons présenté les caractéristiques des ressources matérielles utilisées pour la réalisation de notre Cluster, nous avons aussi définis l'ensemble des environnements logiciels exploités pour la mise en place du Cluster, enfin nous avons terminé par la description de toutes les étapes suivies pour l'installation et la configuration complète du Cluster Hadoop.

Ce chapitre nous a permis de mettre en place un Cluster Hadoop, et de se familiariser avec un Cluster Multi-Nœuds, ce qui nous a donné une base solide pour entamer la phase de réalisation de l'interface de gestion de Nœuds du Cluster, qui sera le sujet du prochain chapitre.

Chapitre 05 :

Interface d'utilisation

Chapitre 05 : Interface d'utilisation

Introduction

Dans cette dernière partie, nous allons aborder la réalisation d'une interface permettant à l'utilisateur de gérer les nœuds du Cluster. Elle offrira à l'utilisateur le choix de mettre en service ou hors service de n'importe quel nœud. Ainsi, l'utilisateur pourra choisir l'environnement d'exécution de son programme. Dans ce chapitre, nous définirons les différentes étapes de réalisation, les outils les environnements et la méthodologie suivie pour la conception et l'implémentation de cette interface.

1. Outils d'implémentation

Pour créer l'interface, nous avons donc opté pour la réalisation d'une application web, où on a utilisé les langages de programmation HTML, CSS et JavaScript pour organiser notre interface, et pour les traitements des différentes opérations on a utilisé le langage de programmation PHP ainsi que des Script Bash Ubuntu.

1.1. HTML

HTML (HyperText Markup Language) : Le HTML est un langage qui a été créé en 1991. Les sigles « HTML » sont l'abréviation de « HyperText Markup Language » ou « langage de balisage hypertexte » en français *[WEB 24]*.

Le HTML est donc un langage de balisage, c'est-à-dire un langage qui va nous permettre de définir les différents contenus d'une page, de structurer sémantiquement et logiquement et de mettre en forme des page Web.

1.2. CSS

CSS (Cascading Style Sheets) : Le CSS a été créé en 1996, soit 5 ans après le HTML. Les sigles « CSS » sont l'abréviation de « Cascading Style Sheets » ou « feuilles de styles en cascade » en français.

Les styles permettent de définir des règles appliquées à un ou plusieurs documents HTML. Ces règles portent sur le positionnement des éléments, l'alignement, les polices de caractères, les couleurs, les marges et espacements, les bordures, les images de fond, ... etc.

Le but de CSS est de séparer la structure d'un document HTML et sa présentation. En effet, avec HTML, on peut définir à la fois la structure (le contenu et la hiérarchie entre les différentes parties d'un document) et la présentation. Mais cela pose quelques problèmes. Avec le couple HTML/CSS, on peut créer des pages web où la structure du document se trouve dans le fichier HTML tandis que la présentation se situe dans un fichier CSS *[WEB 25]*.

1.3. PHP

PHP (Hypertexte Preprocessor) : PHP est un langage de script Web intégré au HTML, cela signifie que le code PHP peut être inséré dans le HTML d'une page Web. Lors de l'accès à une page PHP, le code PHP est lu ou « analysé » par le serveur sur lequel la page réside, la sortie des fonctions PHP sur la page est généralement renvoyée sous forme de code HTML, qui peut être lu par le navigateur. Comme le code PHP est transformé en HTML avant le chargement de la page, les utilisateurs ne peuvent pas afficher le code PHP sur une page. Cela rend les pages PHP suffisamment sécurisées pour accéder aux bases de données et à d'autres informations sécurisées *[WEB 27]*.

- Installation de PHP :

Pour y est faire on exécute la commande :

```
$ sudo apt-get install php
```

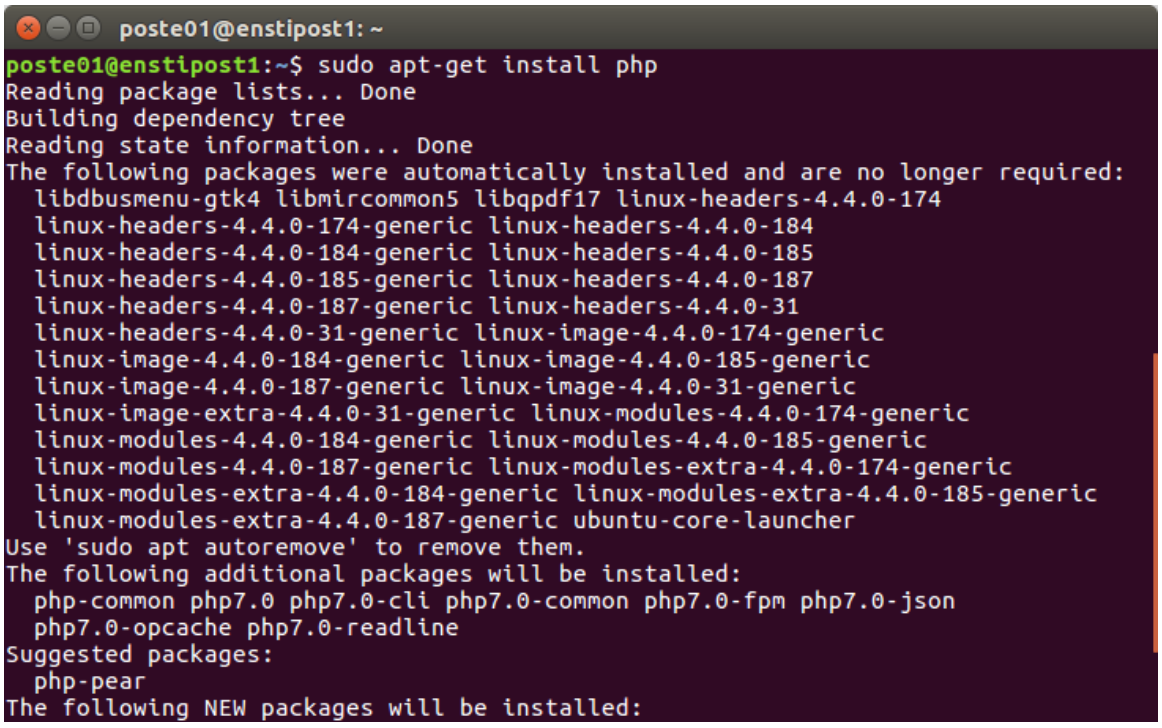

Figure 46. Installation de PHP

Ensuite on doit télécharger toutes les bibliothèques nécessaires utilisées par le langage de programmation PHP.

```
$ sudo apt install libapache2-mod-php
```

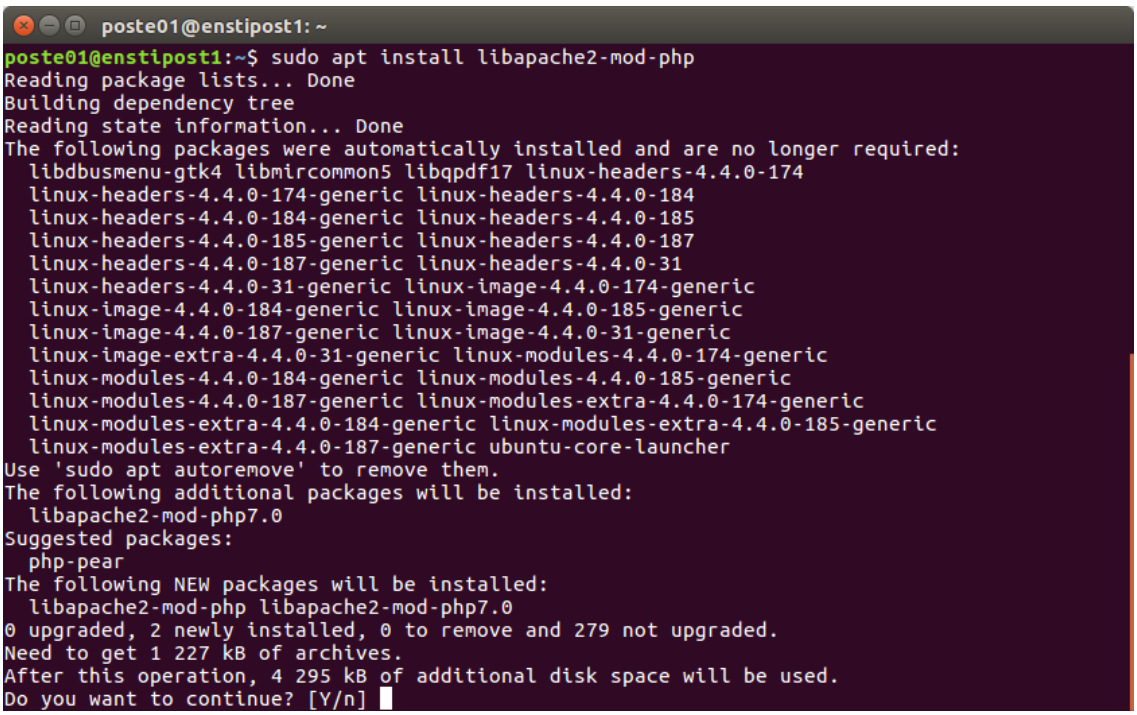

Figure 47. Téléchargement des bibliothèques PHP

Pour la vérification de l'installation on a créé un fichier *''test.php''* ou on a tout simplement appeler la fonction *phpinfo()*; pour afficher les différentes informations concernant la version du langage PHP installé.

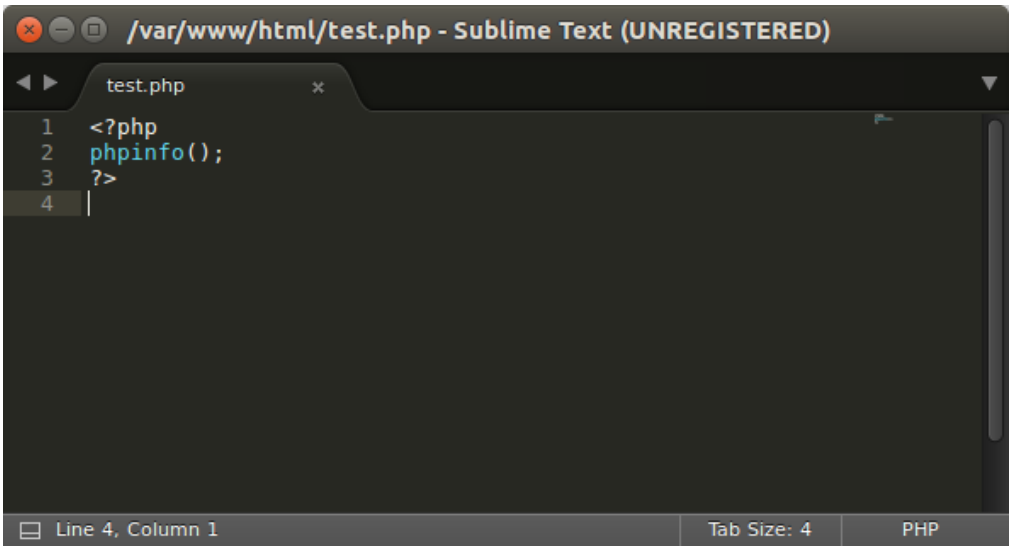

Figure 48. Le fichier ''test.php''

On aura la page suivante après exécution :

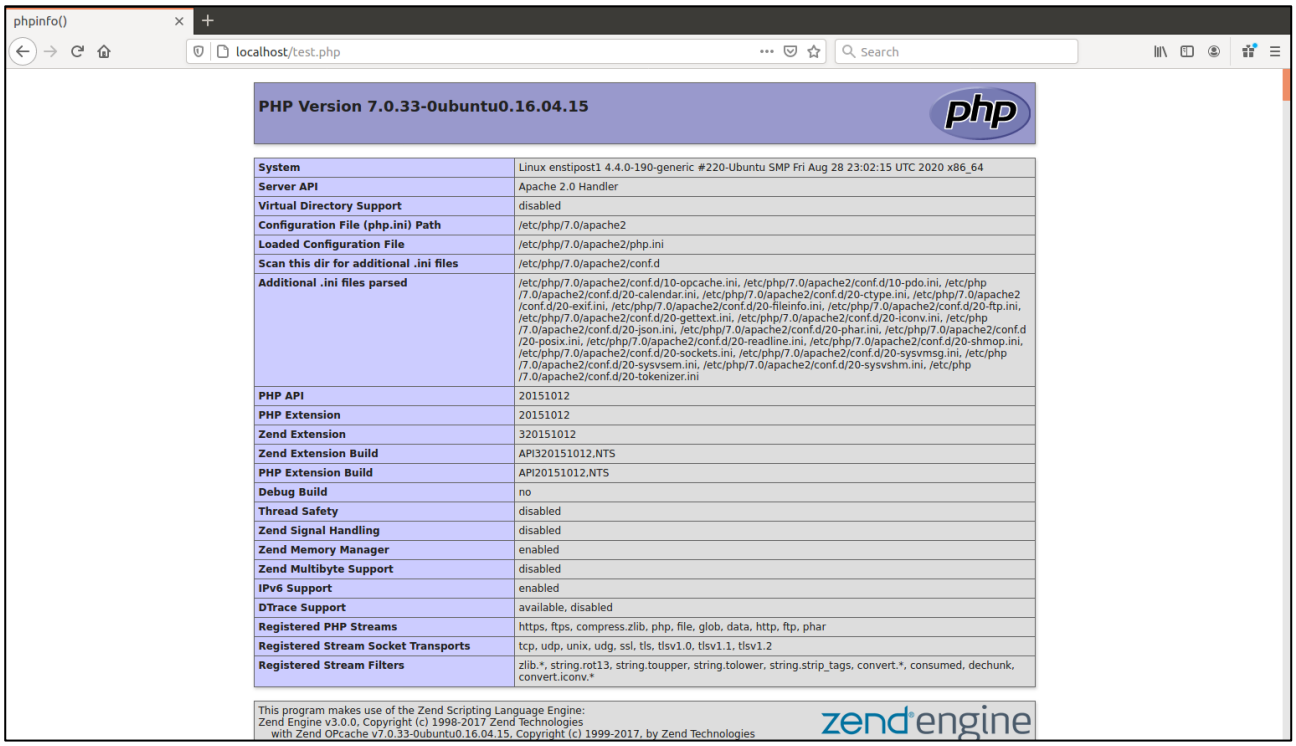

Figure 49. Test d'installation de PHP

1.4. JavaScript

JavaScript est un langage de programmation couramment utilisé dans le développement Web, il a été initialement développé par Netscape pour ajouter des éléments dynamiques et interactifs aux sites Web.

JavaScript est un langage de script côté client, ce qui signifie que le code source est traité par le navigateur Web du client plutôt que sur le serveur Web. Cela signifie que les fonctions JavaScript peuvent s'exécuter après le chargement d'une page Web sans communiquer avec le serveur. Par exemple, une fonction JavaScript peut vérifier un formulaire Web avant qu'il ne soit soumis pour s'assurer que tous les champs obligatoires ont été remplis. Le code JavaScript peut produire un message d'erreur avant que des informations ne soient réellement transmises au serveur *[WEB 26]*.

1.5. Script Bash Ubuntu

Script Bash Ubuntu : Tout d'abord, il faut définir ce qu'est un Shell. Un Shell, c'est un interpréteur de commandes, et en même temps un langage de programmation. Lorsque on tape des commandes sur une console (ou un terminal), on envoie ces commandes au Shell, qui va les interpréter pour renvoyer le résultat.

Le Bash, c'est l'un des nombreux Shell existants dans le monde Unix, c'est en fait un clone du Bourne Shell Bash (Bourne Again Shell), c'est aussi le Shell standard sous Linux. L'une des fonctions les plus intéressantes d'un Shell est que l'on peut écrire des scripts : un script est un fichier texte qui rassemble plusieurs commandes, et qui les exécute les unes après les autres.

1.6. Sublime Text

Sublime Text est un éditeur de texte multiplateforme exclusif conçu pour les personnes qui passent énormément de temps à mélanger du code. Éditeur de programmeur, Sublime Text est une troisième option à l'énigme de longue date "Vi ou Emacs". Au-delà des bases de la coloration syntaxique et du pliage de code, Sublime offre une litanie de fonctionnalités innovantes et uniques, avec la version 3.0 qui approche à grands pas *[WEB 28]*.

- Installation de Sublime Text 3 :

Avant de commencer de créer notre code source, on installe l'éditeur de texte Sublime Text qui compatible avec un la majorité des langages de programmation existant, qui aussi très simple à utiliser.

```
$ sudo add-apt-repository ppa:webupd8team/sublime-text-3
$ sudo apt-get update
$ sudo apt-get install sublime-text-installer
```

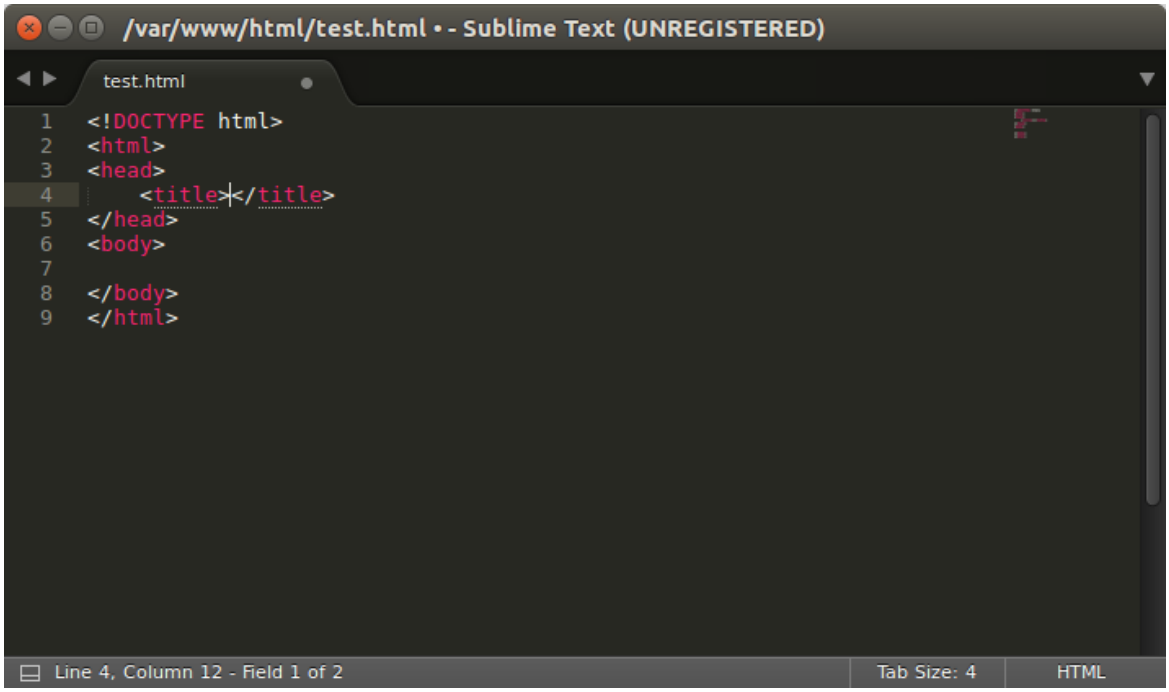

Figure 50. Installation de Sublime Text 3

1.7. Serveur HTTP Apache 2

Serveur HTTP Apache 2 : Un serveur HTTP permet à un site web de communiquer avec un navigateur en utilisant le protocole HTTP(S) et ses extensions (WebDAV, etc.). Apache est probablement le serveur HTTP le plus populaire, c'est donc lui qui met à disposition la plupart des sites internet du WWW.

Ce produit est conçu par Apache Software Foundation, c'est un logiciel libre fourni sous la licence spécifique Apache.

On utilise généralement Apache en conjonction avec d'autres logiciels, permettant d'interpréter du code et d'accéder à des bases de données, le cas le plus courant est celui d'un serveur LAMP (Linux Apache MySQL PHP) *[WEB 29]*.

- Installation du Serveur Apache 2 :

Pour installer le serveur local Apache 2 on utilise la commande suivante :

```
$ sudo apt-get install apache2
```

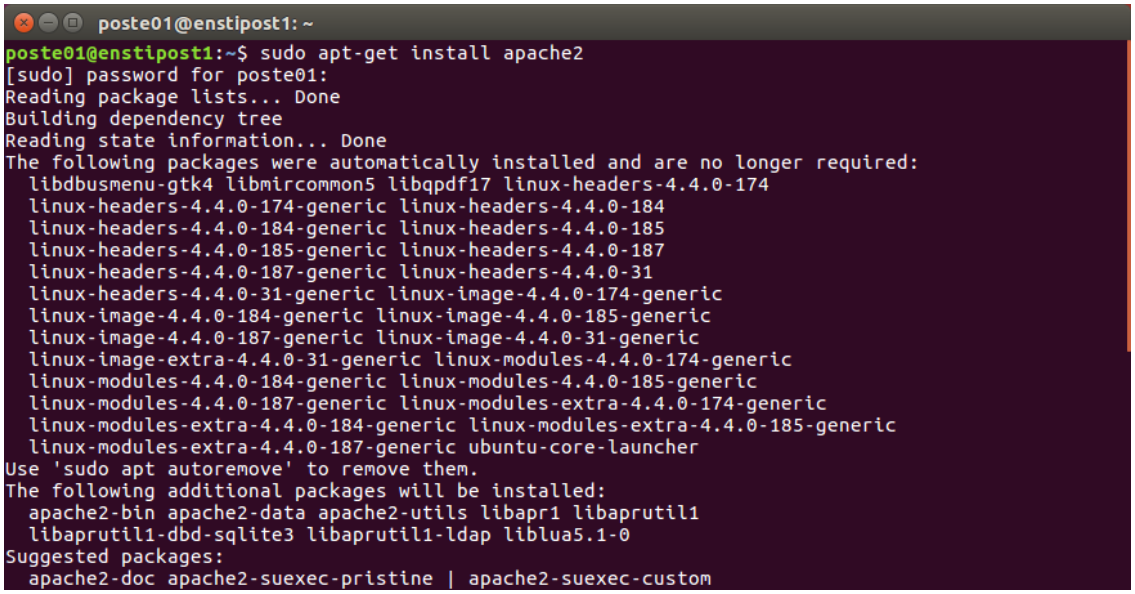

Figure 51. Installation du Serveur Apache2

Afin de vérifier l'installation du serveur, il faut ouvrir un navigateur Web et Taper *''localhost''*, on aura la page suivante :

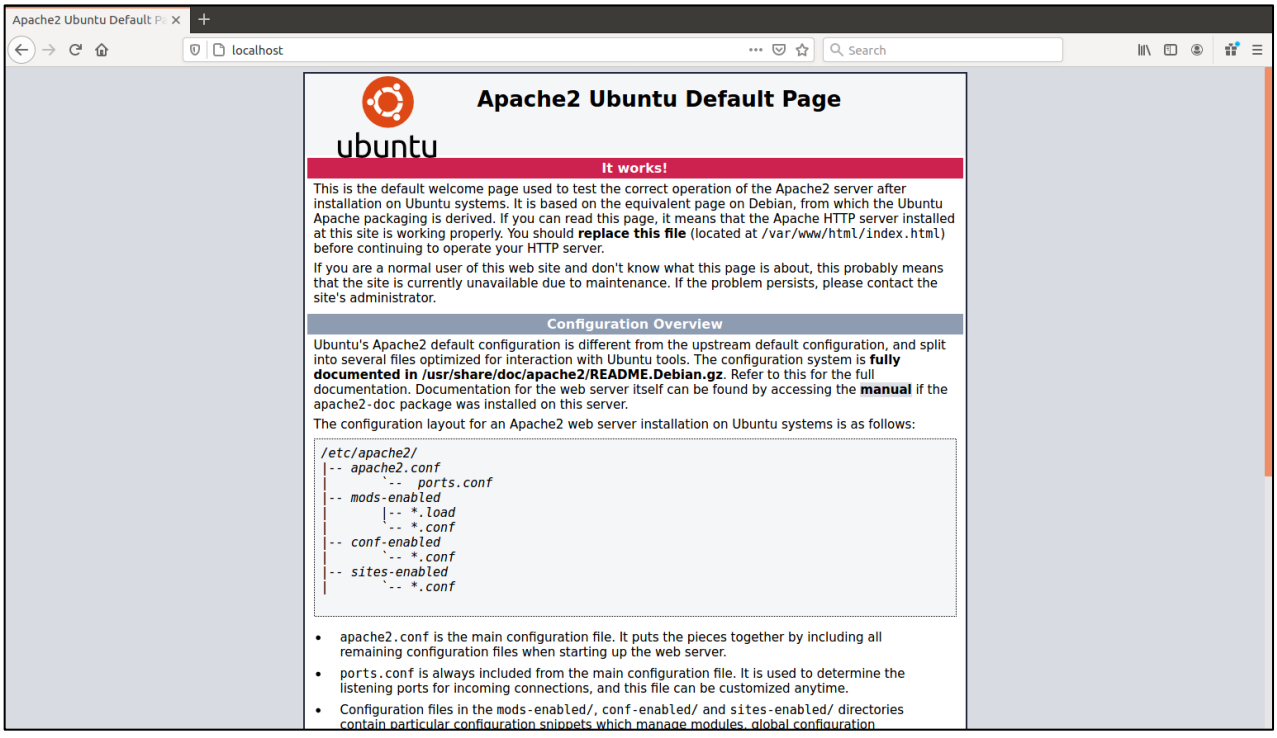

Figure 52. Page d'accueil Apache2

Avant de tester le bon fonctionnement du serveur il faut donner les permissions nécessaires de lecture, d'écriture et d'exécution au répertoire *''/var/www/html''* où nous allons stocker tous nos fichiers contenant le code source de l'interface, on utilise la commande :

```
$ sudo chmod -R 777 /var/www/html/
```

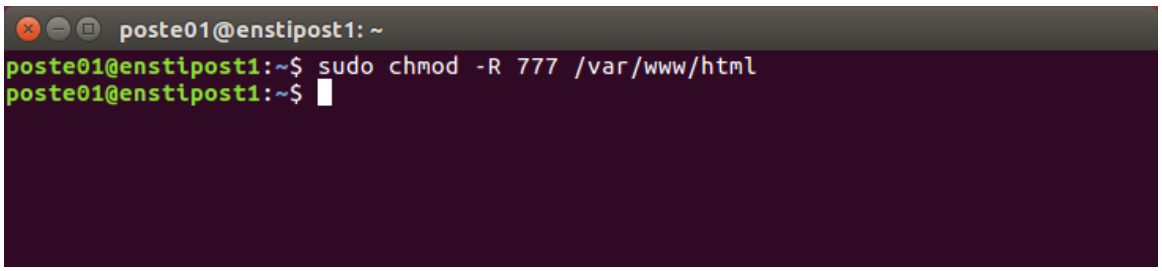

Figure 53. Autorisation des permissions pour ''/var/www/html''

Ensuite nous allons créer un fichier *''test.html''* qu'on utilise uniquement pour tester le bon fonctionnement du serveur local.

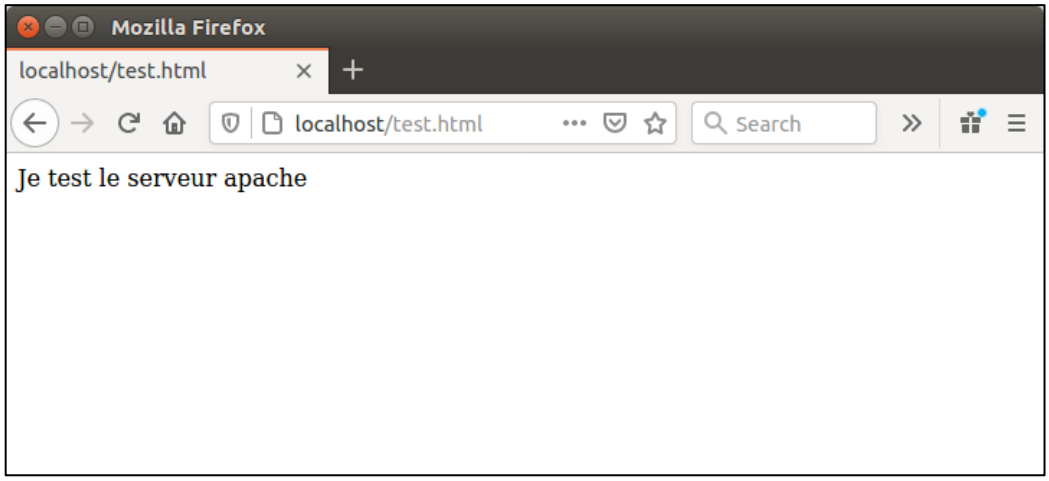

Figure 54. Test du serveur local Apache2

Après nous avons installé MySQL Server et phpMyAdmin pour d'éventuelles future utilisation à l'avenir pour création, manipulation et gestion des base de données relationnelles.

```
$ sudo apt get-install mysql-server mysql-client
```

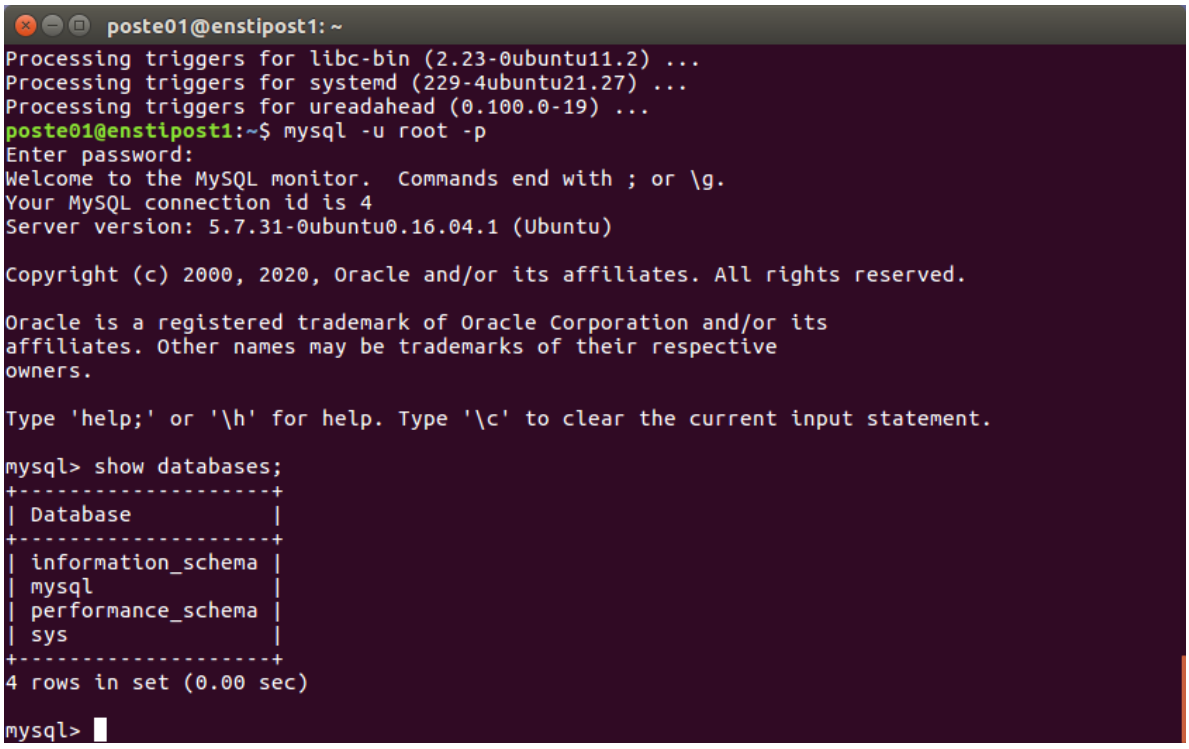

Figure 55. Installation de MySQL

```
$ sudo apt get-install phpmyadmin
```

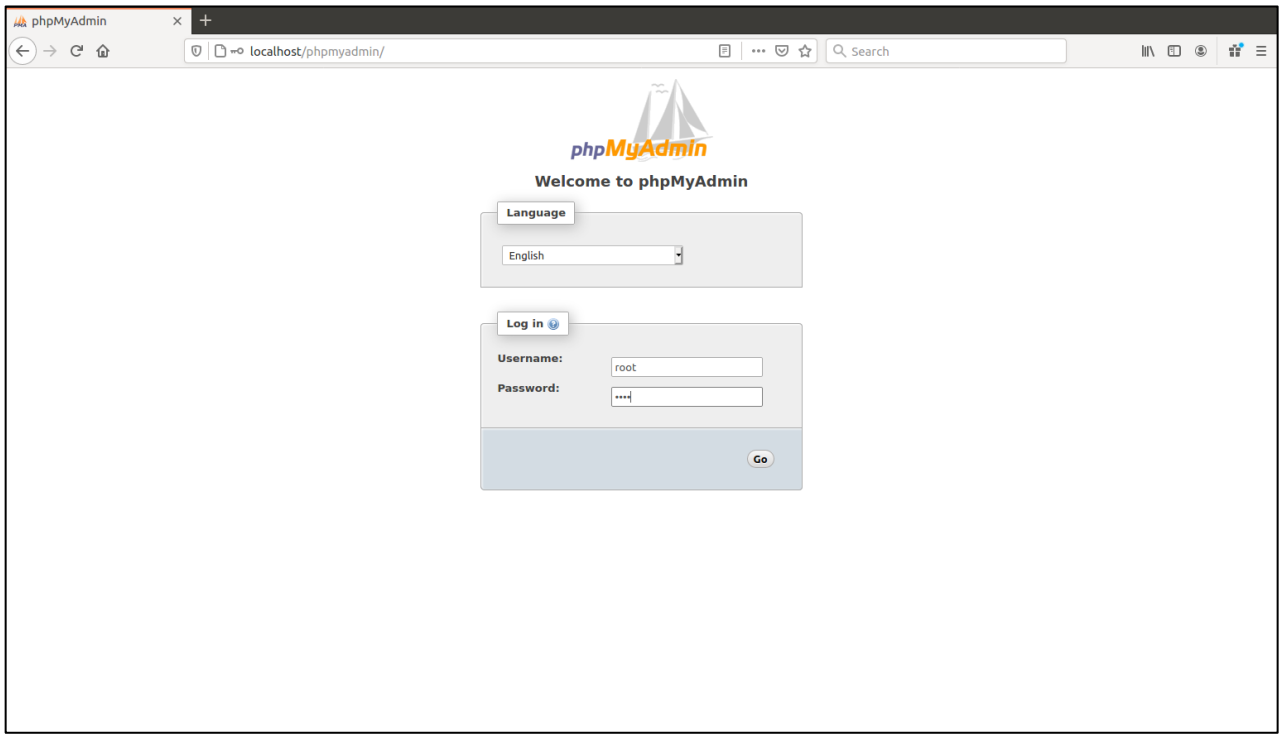

Figure 56. Installation de phpMyAdmin

2. Implémentation de l'interface

La première étape de l'implémentation de l'interface d'utilisation est la création des scripts Shell. Pour chaque poste esclave on a créé deux Script Shell *''add_numPost.sh''* et *''remove_numPost.sh''*. Ces scripts permettent de modifier certains fichiers de configuration du Cluster Hadoop.

D'abord pour rendre le Script exécutable il faut ajouter la permission d'exécution pour chaque Script avec la commande :

```
$ sudo chmod +x add3.sh
```

Et pour autoriser le serveur local apache2 de faire des modifications sur les fichiers de configuration, il faut changer les permissions du répertoire contenant ces fichiers via la commande :

```
$ sudo chown -R www-data:www-data /home/poste01/Desktop/exclude/
```

* Le script *''remove_numPost.sh''* : Permet de mettre Hors service le poste souhaité, et cela en modifiant les fichiers de configuration du cluster, ces fichiers sont *''yarn-site.xml''*, *''hdfs-site.xml''* et la création d'un nouveau fichier nommé *''dfs_exclude''*, ce dernier va contenir les identifiants de tous les nœuds qui sont hors service.

Ce script permet de faire la configuration suivante des fichiers cités auparavant :

- *hdfs-site.xml : Ajouter les lignes suivante à la fin du fichier*

```
<property>
<name>dfs.hosts.exclude</name>
<value>/home/poste01/Desktop/exclude/dfs_exclude</value>
</property>
```

- *yarn-site.xml : Ajouter les lignes suivante à la fin du fichier*

```
<property>
<name>yarn.resourcemanager.node.exclude-path</name>
<value>/home/poste01/Desktop/exclude/dfs_exclude</value>
</property>
```

- *dfs_exclude : (Exemple le poste 03 est Hors Service)*

```
enstipost3
```

* Le script ''add_numPost.sh'' : Permet de mettre En service le poste souhaité (s'il est déjà hors service), et cela en faisant l'opération inverse de *''remove_numPost.sh''*, soit de supprimer la configurer récente pour permettre au nœud d'être à nouveau En service.

Afin d'activer les modifications de ces Script Bash il faut actualiser les Nœuds du Cluster, on a donc créé le Script *''refreshNodes.sh''* qui va nous permettre d'actualiser le Cluster pour qu'il soit prêt à être utilisé.

Enfin pour finir nous avons créé pour chaque poste un fichier PHP *''exec_numPost.php''* qui permet d'exécuter les script Bash déjà créer, et un fichier *''refreshNodes.php''*, pour exécuter le Script d'actualisation du Cluster.

Tous ces fichiers sont stockés dans le répertoire *''var/www/html/interface/''* du Nœud maitre *''enstipost1''*, pour permettre au serveur local d'exécuter les fichier PHP.

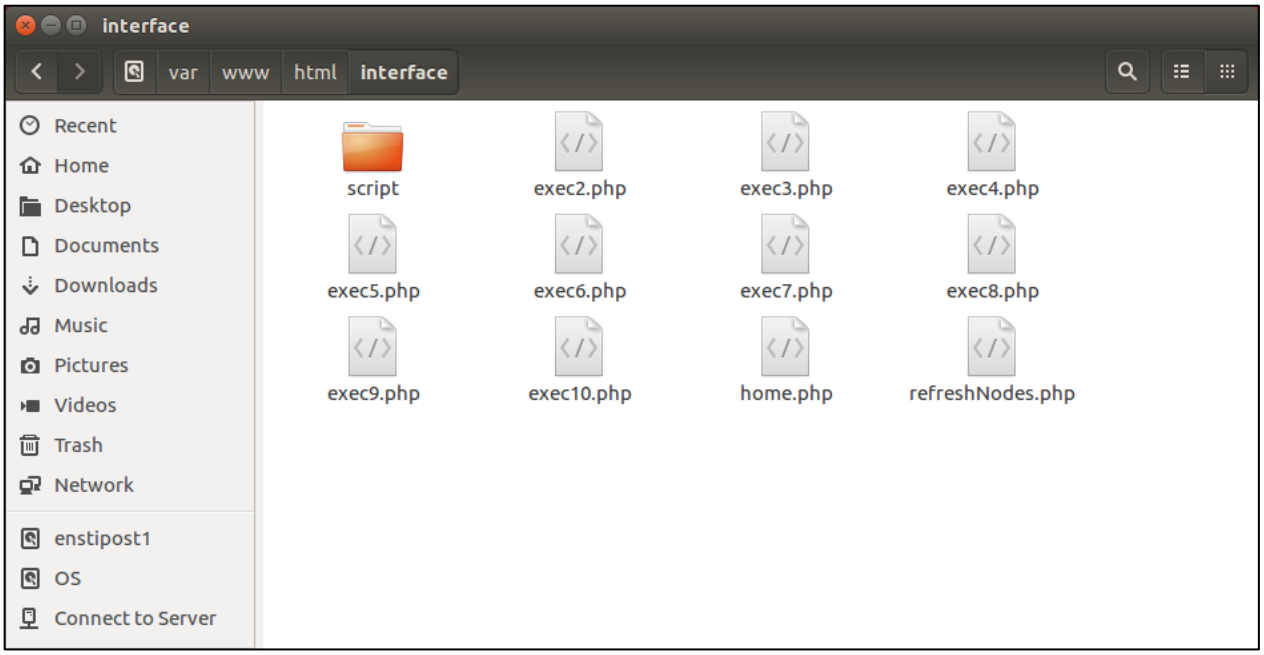

Figure 57. Les fichiers PHP utilisés

Et tous les scripts créés sont stocké dans le répertoire *''var/www/html/interface/script''*.

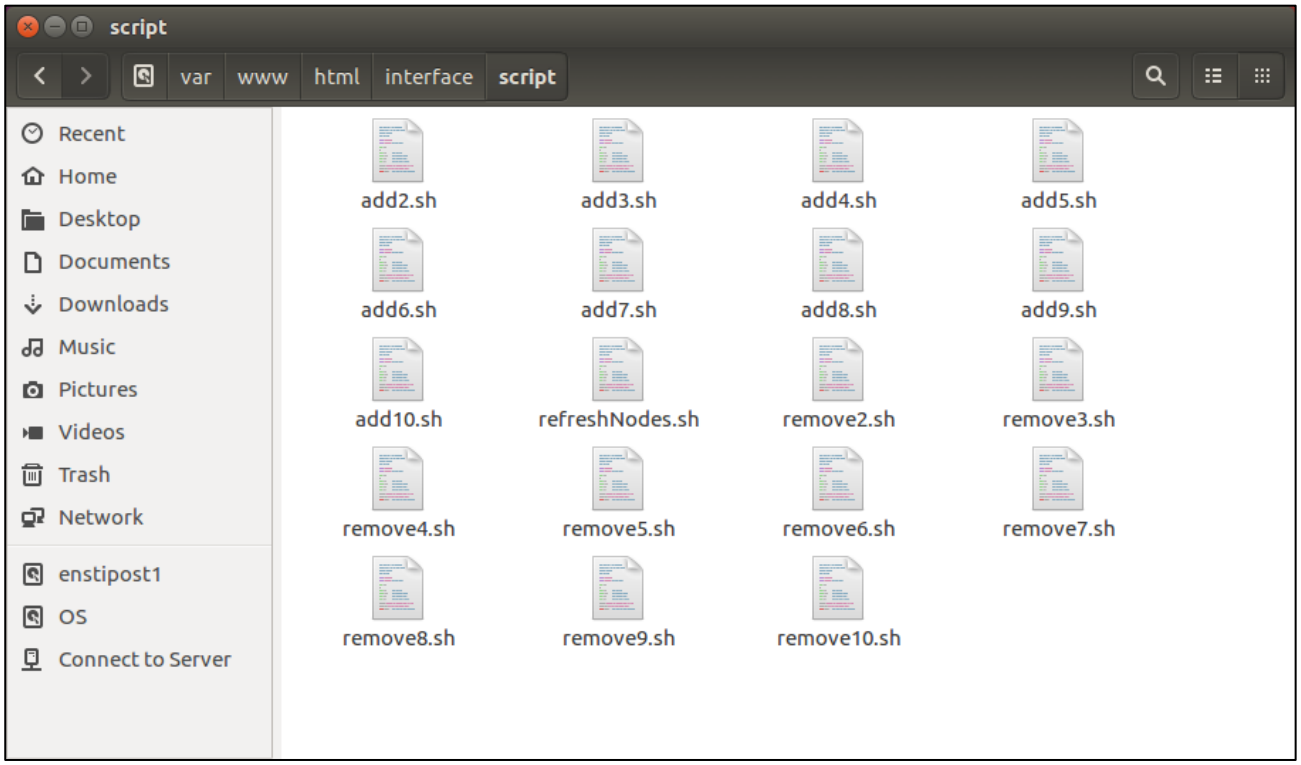

Figure 58. Les fichiers script utilisés

3. Présentation de l'interface réalisée

Maintenant que tous les outils et les environnements nécessaire sont correctement installés on passe à la présentation de l'interface réalisée.

Notre interface nous permet de choisir de mettre En / Hors service les nœuds esclaves de notre Cluster.

La page d'accueil de notre interface est la suivante :

Figure 59. Interface de gestion de nœuds du Cluster

Comme l'interface le montre pour chaque poste (Nœud) on peut choisir soit de le mettre En service ou de le désactiver et de le mettre Hors service.

En cliquant sur le bouton mettre hors service du *poste 03* (par exemple), un message de notification nous informe que le poste est bien mis *Hors service*.

Figure 60. Mise Hors service du Poste 03

Dans la partie traitement en cliquant sur le bouton (*Mettre Hors Service*) le fichier PHP *''exec3.php''* va appeler la fonction qui permet d'exécuter le Script Bash *''remove3.sh''* du poste 03.

Pour activer les modifications de mise hors service du poste 03, il faut actualiser les nœuds du Cluster en cliquant sur le bouton *Actualiser Le Cluster*. Un message indiquant que le cluster est prêt s'affiche.

Figure 61. Actualisation du Cluster

La même chose que précédemment ce bouton permet au fichier PHP *''refreshNodes.php''* d'appelé la fonction qui exécute le script Bash *''refreshNodes.sh''*.

Enfin pour vérifier que l'opération de mise Hors Service du Poste 03 est bien réalisée, on accède à l'URL *''enstipost1:50070''* sur le navigateur Web, qui est l'interface graphique du Cluster Hadoop.

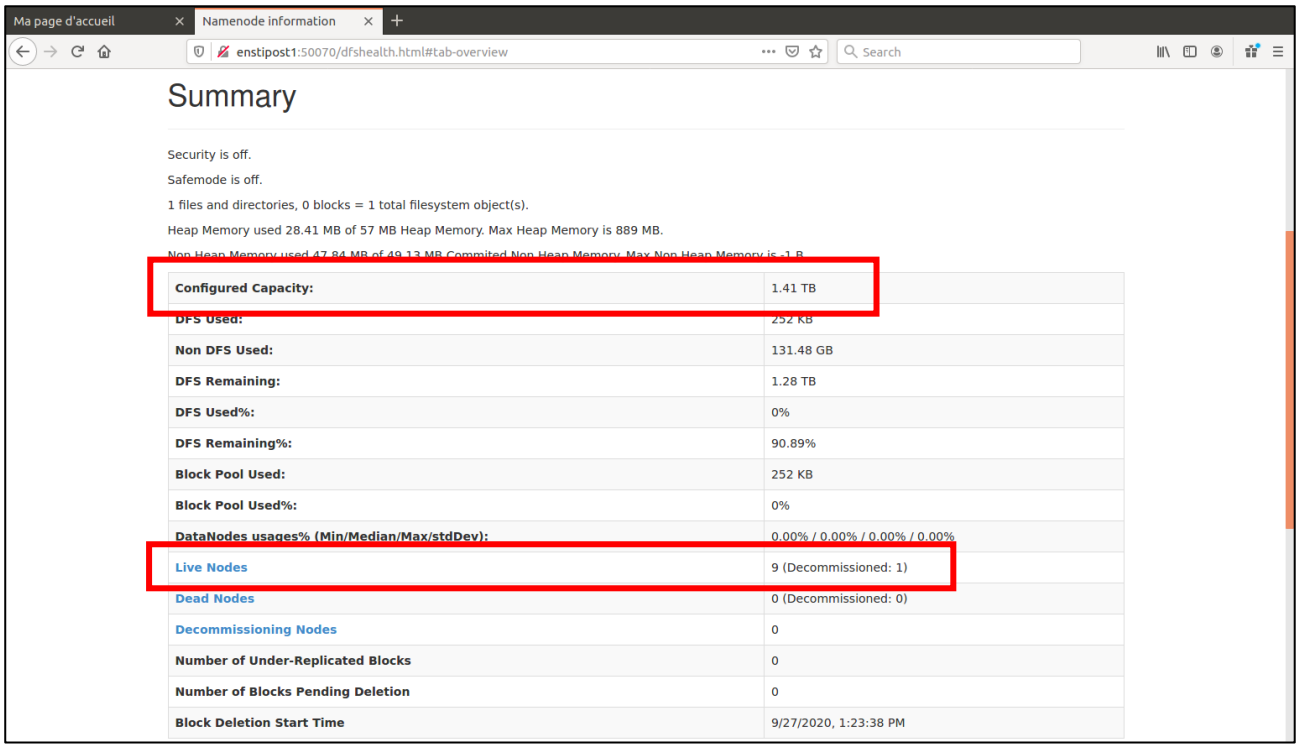

Figure 62. Vérification de mise Hors service du Poste 03

On peut donc remarquer que la capacité configurée a été changer de 1.59 TB initialement à 4.41 TB, on remarque aussi que dans les nœuds actifs, il y a un nœud qui est Hors service.

On clique sur *Live Nodes* pour avoir plus d'information sur les nœuds.

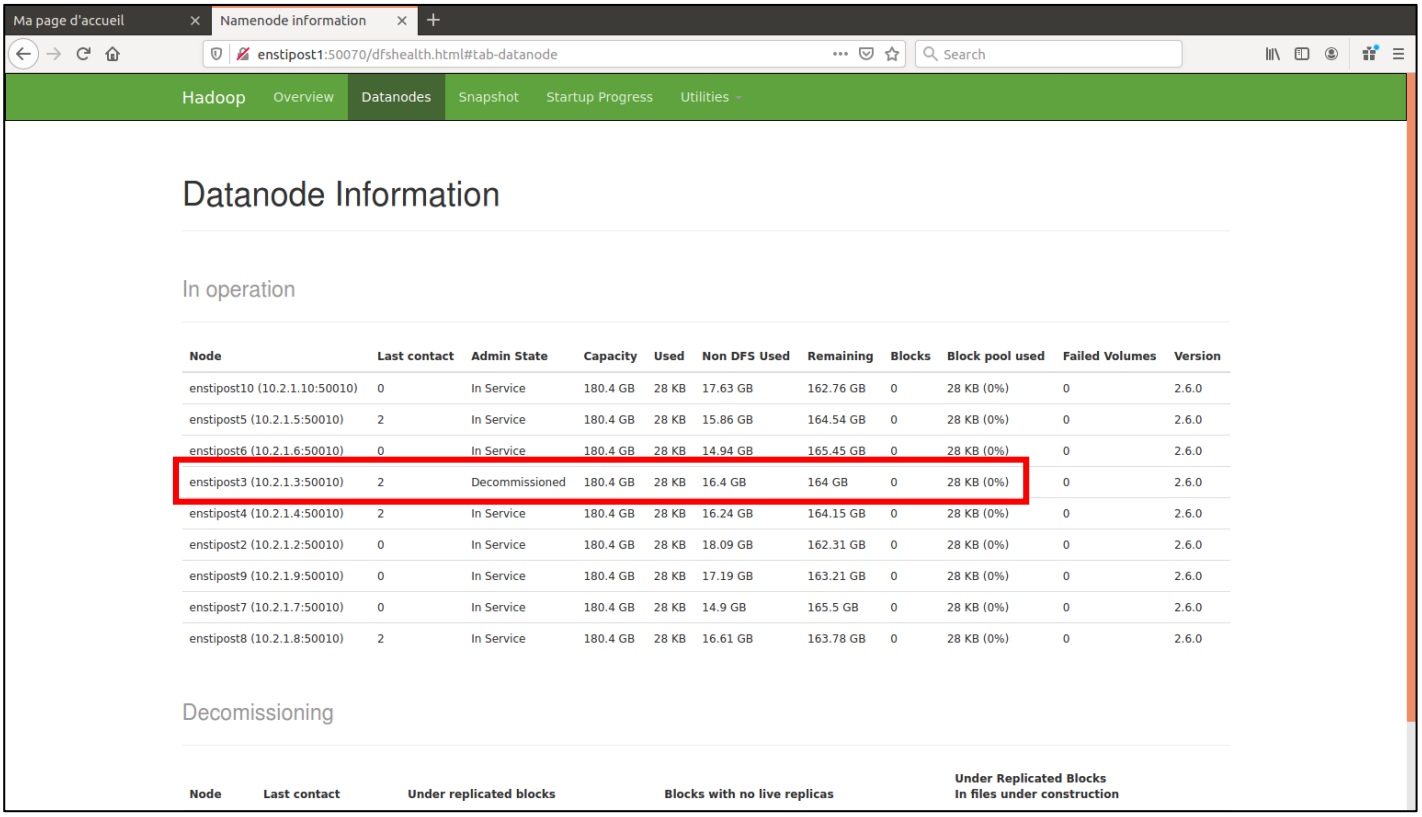

Figure 63. Informations détaillées sur les Nœuds

On peut donc voire clairement que le poste ayant l'identifiant *enstipost3* est bien Hors Service.

4. Evaluation de performances en exploitant l'interface

Pour tester et expérimenter notre Cluster Hadoop nous allons utiliser deux (02) Jobs déjà fournis par MapReduce à savoir le Job WordCount et le Job Pi. Deux types de tests vont être effectués sur les 2 Jobs dans des architectures distribuées à 9 nœuds puis à 3 nœuds.

4.1. Le Job WordCount

WordCount est un programme simple qui compte le nombre d'occurrences de chaque mot dans un ensemble d'entrée donné.

Pour pouvoir tester le Job WordCount, on a généré un fichier qui contient des prénoms avec un script Shell dont la fonction est de prendre en paramètre une source texte (*input.txt*) et d'ajouter son contenu (*n)* fois dans un fichier de sortie (*prenoms.txt*).

La commande utilisée pour générer le fichier *prenoms.txt* est :

```
$ ./sc.sh input.txt prenoms.txt 10000
```

Sur le nœud Master on a créé un volume HDFS par la ligne de commande suivante :

```
$ cd /opt/hadoop
$ hadoop fs -mkdir /user/hdfs/pfe
```

Ensuite on a inséré le fichier généré depuis le système de fichiers local vers le système de fichiers HDFS.

```
$ hadoop fs -put /home/poste01/Desktop/prenoms.txt /user/hdfs/pfe
```

On peut donc maintenant passer à l'exécution du job.

Pour lancer le Job WordCount, il faut donner les paramètres d'entrées *''user/hdfs/pfe''* et les paramètres de sortie *output* à l'aide de la ligne de commande suivante :

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount 'user/hdfs/pfe' output
```

- Exécution du Job WordCount en mode distribué avec neufs (09) nœuds

```
poste01@enstipost1: /opt/hadoop
poste01@enstipost1:/opt/hadoop$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount '/user/hdfs/pfe' output
20/09/30 14:30:23 INFO client.RMProxy: Connecting to ResourceManager at enstipost1/10.2.1.1:8032
20/09/30 14:30:24 INFO input.FileInputFormat: Total input paths to process : 1
20/09/30 14:30:25 INFO mapreduce.JobSubmitter: number of splits:1
20/09/30 14:30:25 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1601464919177_0004
20/09/30 14:30:26 INFO impl.YarnClientImpl: Submitted application application_1601464919177_0004
20/09/30 14:30:26 INFO mapreduce.Job: The url to track the job: http://enstipost1:8088/proxy/application_1601464919177_0004/
20/09/30 14:30:26 INFO mapreduce.Job: Running job: job_1601464919177_0004
20/09/30 14:30:31 INFO mapreduce.Job: Job job_1601464919177_0004 running in uber mode : false
20/09/30 14:30:31 INFO mapreduce.Job:  map 0% reduce 0%
20/09/30 14:30:36 INFO mapreduce.Job:  map 100% reduce 0%
20/09/30 14:30:41 INFO mapreduce.Job:  map 100% reduce 100%
20/09/30 14:30:41 INFO mapreduce.Job: Job job_1601464919177_0004 completed successfully
20/09/30 14:30:41 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=516
                FILE: Number of bytes written=212545
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=3660113
                HDFS: Number of bytes written=510
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=2932
                Total time spent by all reduces in occupied slots (ms)=1761
                Total time spent by all map tasks (ms)=2932
                Total time spent by all reduce tasks (ms)=1761
                Total vcore-seconds taken by all map tasks=2932
                Total vcore-seconds taken by all reduce tasks=1761
                Total megabyte-seconds taken by all map tasks=3002368
                Total megabyte-seconds taken by all reduce tasks=1803264
        Map-Reduce Framework
```

```
poste01@enstipost1: /opt/hadoop
        Map-Reduce Framework
                Map input records=480000
                Map output records=480000
                Map output bytes=5510000
                Map output materialized bytes=516
                Input split bytes=113
                Combine input records=480000
                Combine output records=39
                Reduce input groups=39
                Reduce shuffle bytes=516
                Reduce input records=39
                Reduce output records=39
                Spilled Records=78
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=69
                CPU time spent (ms)=1960
                Physical memory (bytes) snapshot=430166016
                Virtual memory (bytes) snapshot=3828985856
                Total committed heap usage (bytes)=288358400
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=3660000
        File Output Format Counters
                Bytes Written=510
poste01@enstipost1:/opt/hadoop$
```

Figure 64. Exécution du Job WordCount avec neufs (09) nœuds

- Exécution du Job WordCount en mode distribué avec trois (03) nœuds

En utilisant l'interface de gestion de nœuds du Cluster que nous avons créé, nous allons mettre Hors service les nœuds, du nœud numéro 05 jusqu'au nœud numéro 10, pour pouvoir visualiser les différences.

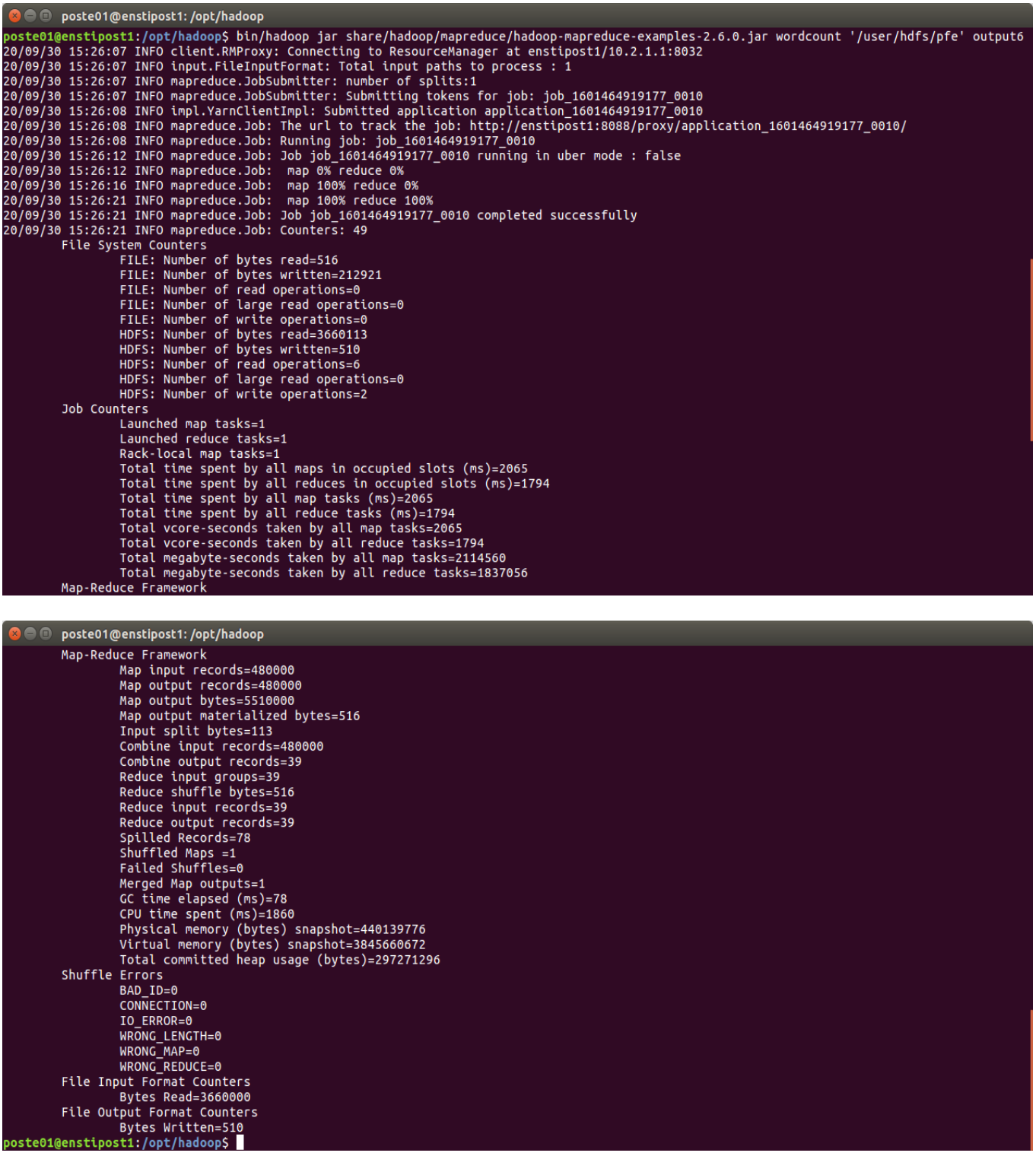

Figure 65. Exécution du Job WordCount avec trois (03) nœuds

- Interprétation des résultats

Après exécution des tests précédents, on peut synthétiser les résultats obtenus comme suit :

	Neuf (09) nœuds	Trois (03) nœuds
Temps d'exécution	14 Sec	18 Sec
CPU	1960 ms	1860 ms

Tableau 2. Résultats d'exécutions du Job WordCount

Après lecture des résultats obtenus, on peut remarquer que le critère de performance principal dans les applications Web à savoir le temps de réponse, est en faveur de l'architecture à neufs (09) nœuds. L'utilisation CPU a été distinguée comme un facteur limitant durant toutes nos expérimentations, ça peut être expliqué par le fait que le système à neufs (09) nœuds applique un fort parallélisme en employant tous les cœurs du CPU, contrairement au système à trois (03) nœuds, puisque c'est un temps cumulé de tous les cœurs des machines.

Un autre renseignement peut être tiré des résultats précédents, est le fait que des résultats obtenus en *output* ont été conservés malgré la mise Hors service des cinq (05) nœuds, ce qui implique une forte tolérance aux pannes, dû principalement au principe de réplication des données, considéré comme un principe de base et un atout des systèmes distribués tel que HDFS.

4.2. Le Job Pi

Pi est un programme qui utilise une méthode statistique (quasi-Monte Carlo) pour estimer la valeur de Pi (π) à partir d'un ensemble de points géométriques, il ne requiert pas de fichiers depuis le système de fichiers HDFS.

Pour lancer le Job Pi il faut lui donner le nombre de Map à traiter (arg.1) et le nombre d'échantillons par Map (arg.2).

On exécute alors dans nos tests le Job Pi, d'abord avec un nombre de Map = (4) et le nombre d'échantillons par Map = (10^3), ensuite avec le nombre de Map = (8) et le nombre d'échantillons par Map = (10^6).

Nous devant d'abord remettre En service tous les nœuds du Cluster avec l'interface de gestion de nœuds. Et les mettre aussi Hors service pour la deuxième exécution.

Nous allons utiliser la commande suivante :

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.6.0.jar pi 4 1000
```

- Exécution du Job Pi en mode distribué avec neufs (09) nœuds

** Le cas nombre de Map = (4) et le nombre d'échantillons par Map = (10^3)*

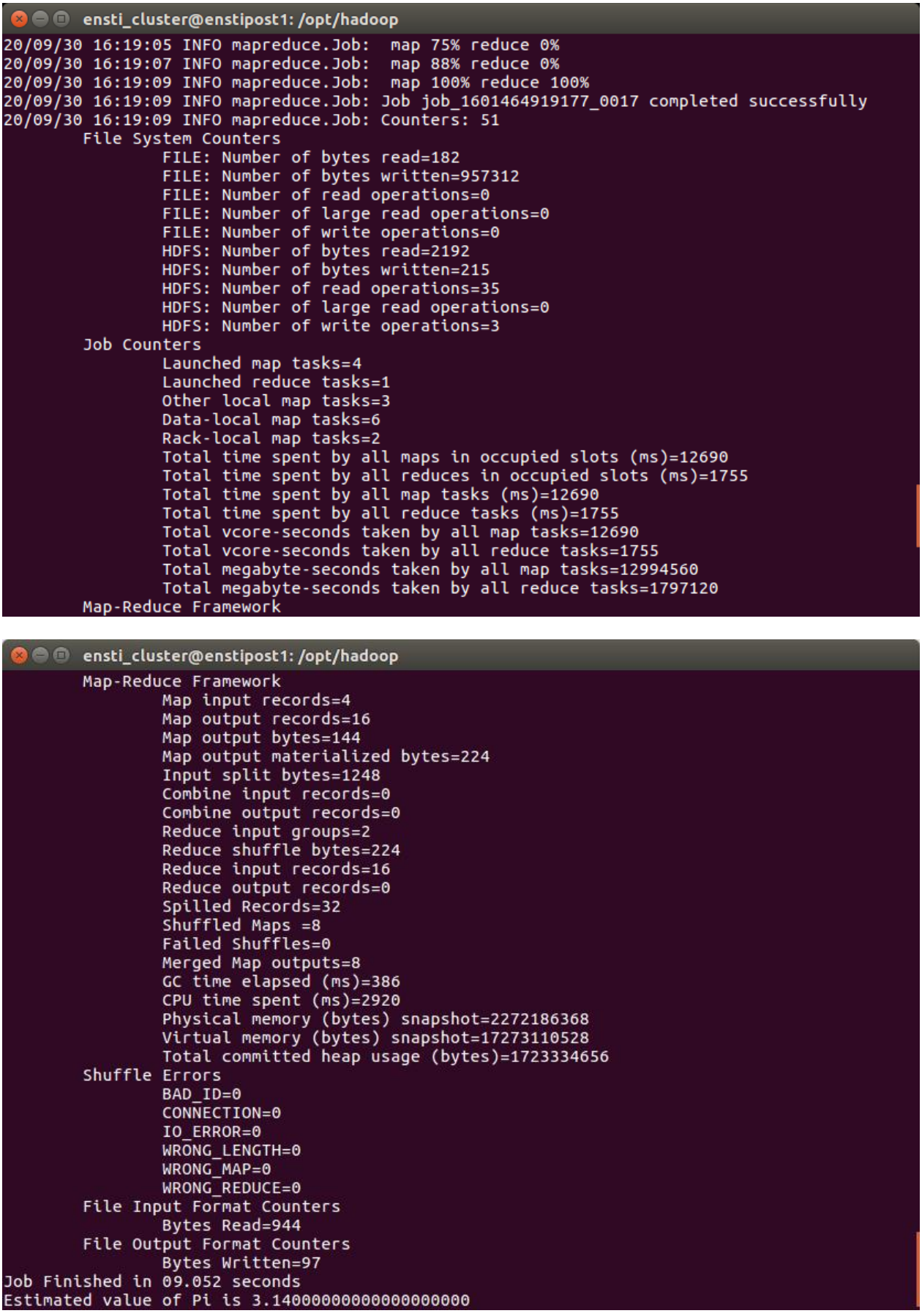

Figure 66. Exécution du Job Pi avec (09) nœuds (nbr de Map = 4 et nbr échantillons par Map = 10^3)

** Le cas nombre de Map = (8) et le nombre d'échantillons par Map = (10^6)*

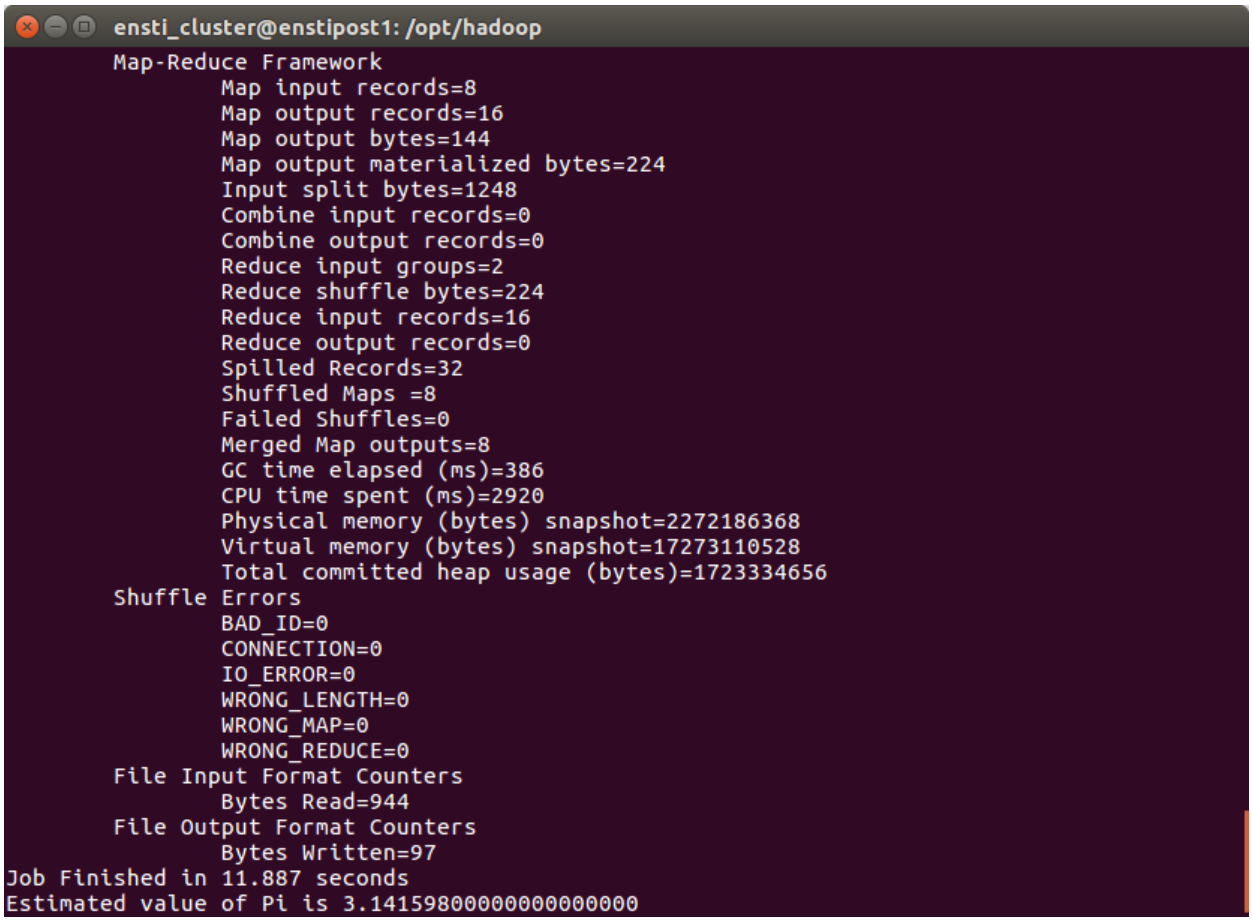

Figure 67. Exécution du Job Pi avec (09) nœuds (nbr de Map = 8 et nbr échantillons par Map = 10^6)

- Exécution du Job Pi en mode distribué avec trois (03) nœuds

** Le cas nombre de Map = (4) et le nombre d'échantillons par Map = (10^3)*

```
ensti_cluster@enstipost1: /opt/hadoop
20/09/30 16:17:55 INFO mapreduce.Job:  map 88% reduce 0%
20/09/30 16:17:58 INFO mapreduce.Job:  map 100% reduce 0%
20/09/30 16:17:59 INFO mapreduce.Job:  map 100% reduce 100%
20/09/30 16:18:00 INFO mapreduce.Job: Job job_1601464919177_0016 completed successfully
20/09/30 16:18:00 INFO mapreduce.Job: Counters: 51
        File System Counters
                FILE: Number of bytes read=182
                FILE: Number of bytes written=957321
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2184
                HDFS: Number of bytes written=215
                HDFS: Number of read operations=35
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=3
        Job Counters
                Launched map tasks=4
                Launched reduce tasks=1
                Other local map tasks=1
                Data-local map tasks=1
                Rack-local map tasks=7
                Total time spent by all maps in occupied slots (ms)=16832
                Total time spent by all reduces in occupied slots (ms)=3661
                Total time spent by all map tasks (ms)=16832
                Total time spent by all reduce tasks (ms)=3661
                Total vcore-seconds taken by all map tasks=16832
                Total vcore-seconds taken by all reduce tasks=3661
                Total megabyte-seconds taken by all map tasks=17235968
                Total megabyte-seconds taken by all reduce tasks=3748864
        Map-Reduce Framework
```

```
ensti_cluster@enstipost1: /opt/hadoop
        Map-Reduce Framework
                Map input records=4
                Map output records=16
                Map output bytes=144
                Map output materialized bytes=224
                Input split bytes=1248
                Combine input records=0
                Combine output records=0
                Reduce input groups=2
                Reduce shuffle bytes=224
                Reduce input records=16
                Reduce output records=0
                Spilled Records=32
                Shuffled Maps =8
                Failed Shuffles=0
                Merged Map outputs=8
                GC time elapsed (ms)=386
                CPU time spent (ms)=2920
                Physical memory (bytes) snapshot=2272186368
                Virtual memory (bytes) snapshot=17273110528
                Total committed heap usage (bytes)=1723334656
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=944
        File Output Format Counters
                Bytes Written=97
Job Finished in 10.920 seconds
Estimated value of Pi is 3.14000000000000000000
```

Figure 68. Exécution du Job Pi avec (03) nœuds (nbr de Map = 4 et nbr échantillons par Map = 10^3)

** Le cas nombre de Map = (8) et le nombre d'échantillons par Map = (10^6)*

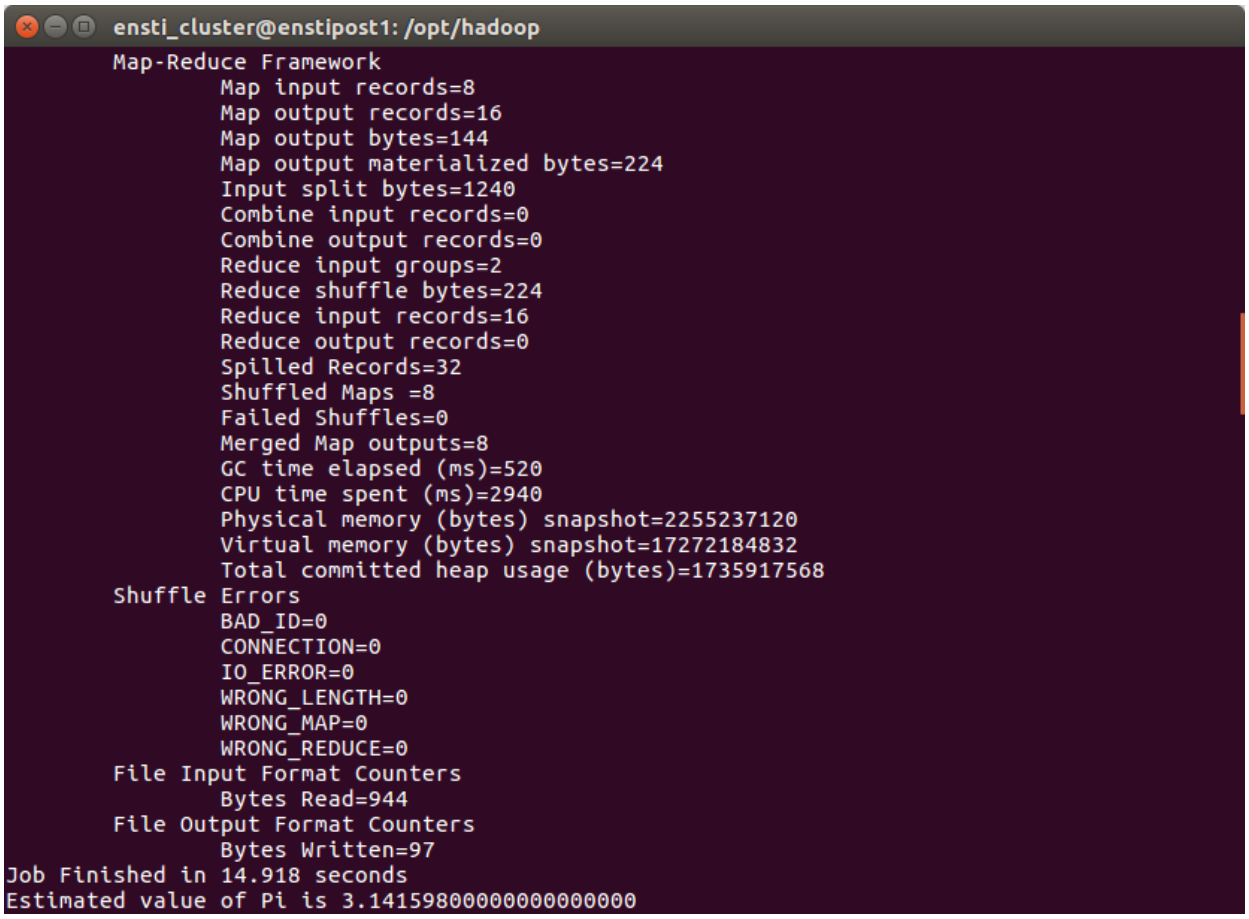

Figure 69. Exécution du Job Pi avec (03) nœuds (nbr de Map = 8 et nbr échantillons par Map = 10^6)

- Interprétation des résultats

On peut résumer les résultats obtenus à partir de cette exécution comme suit :

	Nbr Map = (4) Nbr échantillons = (10^3)		Nbr Map = (8) Nbr échantillons = (10^6)	
Nombre de Nœuds	(09) Nœuds	(03) Nœuds	(09) Nœuds	(03) Nœuds
Temps d'exécution	09.052 Sec	10.920 Sec	11.887 Sec	14.918 Sec
Estimation de Pi (π)	3.14	3.14	3.141598	3.141598

Tableau 3. Résulats d'exécution du Job Pi

On peut affirmer que la complexité des données influe directement sur le temps d'exécution. La différence entre les temps d'exécutions des deux architectures augmente proportionnellement par rapport à la taille des échantillons. La différence continue à accroître au fur et à mesure lorsqu'on augmente le nombre d'échantillons. Cela implique l'influence de la parallélisation des traitements des grands volumes de données sur les performances du système.

La même déduction faite lors du Job précédent WordCount, est observée ici aussi : la précision et la qualité du résultat obtenue de Pi est préservée malgré la mise hors service d'un certain nombre de nœuds, ce qui confirme la forte tolérance aux pannes de HDFS.

Conclusion

Dans ce dernier chapitre nous avons présenté le programme d'implémentation de notre interface, où nous avons expliqué la solution proposée pour la réalisation de l'interface, ensuite nous avons définit l'ensemble des outils et des environnements utilisés ainsi que les étapes d'installation de chacun, après nous somme passer à la présentation de l'interface, enfin on a terminé par quelques tests de performance du Cluster Hadoop mis en place.

Les tests et les expérimentations effectués dans ce chapitre ont amené à la conclusion des points suivants :

- L'augmentation du nombre de nœuds influe sur l'exécution des programmes MapReduce, réduit son temps d'exécution et en conséquence la latence est minimisée.
- La réplication des données garantit un niveau de fiabilité pour le système Hadoop en cas de défaillance d'un de ses nœuds et en conséquence assure la disponibilité du système.
- La flexibilité d'une telle architecture distribuée en augmentant le nombre de nœuds des Clusters reflète le passage à l'échelle et assure l'extensibilité.

Conclusion générale

Conclusion générale

Notre projet a fait l'objet d'une étude des Big Data et de ses nouvelles technologies, nous nous sommes intéressés au Framework Hadoop, qui est l'infrastructure la plus émergente utilisée par la plupart des grandes entreprises telles que Facebook, Microsoft, IBM, Yahoo, Amazon et bien d'autres encore.

La partie applicative consistait à un déploiement de Hadoop avec ses composants MapReduce et HDFS dans un environnement distribué à dix (10) machines physiques.

Les différents systèmes de gestion des données conçus ces dernières années, se distinguent à base de trois facteurs principaux : La performance, caractérisée par la latence et le débit, l'extensibilité (élasticité ou scalabilité) et la disponibilité.

La mise en œuvre d'un Cluster Hadoop avec ses composants, la création de l'interface de gestion de Nœuds du Cluster, et l'exécution de plusieurs Jobs MapReduce sur des fichiers HDFS dans des architectures à plusieurs nœuds et l'évaluation des différents statistiques des opérations effectuées nous ont permis d'approuver l'impact de la distribution et le parallélisme sur l'efficacité des systèmes gérant de grands volumes de données en réduisant la latence et en garantissant et une meilleure extensibilité.

D'un autre côté, on a pu constater, que la réplication des données sur les différents nœuds du réseau garanti un certain niveau de disponibilité du système même en cas de défaillance d'un ou plusieurs nœuds du réseau.

Enfin pour conclure le bilan de ce projet est dans l'ensemble positif, les principaux buts du projet étant accomplis et la plus grande partie du travail a été réalisée. Finalement le projet nous a été d'un apport indéniable en matière de connaissances acquises sur les technologies Big Data et la familiarisation avec le Framework Hadoop.

Références

Références

Bibliographie

[1] VALTECH, Livre Blanc, (2013). Cross Data Marketing, page 16.

[2] Thomas Davenport (2014). Stratégie Big Data: Edition Pearson.

[3] M. CORINUS, T. DEREY, J. MARGUERIE, W. TECHER, N.VIC, (2012). Rapport d'étude sur le Big Data: SRS Day, page 54.

[4] Niting Sawant, Himanshu Shah, (2013). Big data application and architecture Q&A: Apress.

[5] Allae Erraissi, Abdessamad Belangour, Abderrahim Tragha, (2017). International Journal of Computer Trends and Technology (IJCTT) – Volume 48: A Big Data Hadoop building blocks comparative study.

[6] Jonathan Lejeune, (2015). Hadoop : une plate-forme d'exécution de programme Map-reduce, École des Mines de Nantes, page 83.

[7] Mekideche Mounir, (2015). Conception et implémentation d'un moteur de recherche à base d'une architecture Hadoop (Big Data).

Webographie

[WEB 1] https://www.emc.com/leadership/digital-universe/2012iview/big-data-2020.htm (Consulté le 28/09/2020)

[WEB 2] https://www.lebigdata.fr/definition-big-data (Consulté le 18/03/2020)

[WEB 3] https://www.redsen-consulting.com/fr/inspired/data-analyse/big-data (Consulté le 18/03/2020)

[WEB 4] https://lebigdata.com/les-3-v-du-big-data/ (Consulté le 12/04/2020)

[WEB 5] https://www.journaldunet.com/solutions/analytics/big-data/ (Consulté le 18/03/2020)

[WEB 6] https://juvenal-chokogoue.developpez.com/tutoriels/hadoop-fonctionnement/ (Consulté le 17/04/2020)

[WEB 7] https://www.lebigdata.fr/hadoop (Consulté le 20/03/2020)

[WEB 8] https://fr.talend.com/resources/what-is-hadoop/ (Consulté le 18/03/2020)

[WEB 9] https://www.lebigdata.fr/hdfs-fonctionnement-avantages (Consulté le 19/03/2020)

[WEB 10] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (Consulté le 19/03/2020)

[WEB 11] https://openclassrooms.com/fr/courses/4467481-creez-votre-data-lake/4509426-decouvrez-le-systeme-de-fichiers-distribue-hdfs (Consulté le 22/04/2020)

[WEB 12] https://fr.talend.com/resources/what-is-mapreduce/ (Consulté le 20/03/2020)

[WEB 13] https://mbaron.developpez.com/tutoriels/bigdata/hadoop/introduction-hdfs-map-reduce/ (Consulté le 20/03/2020)

[WEB 14] https://fr.talend.com/resources/what-is-hadoop/ (Consulté le 20/03/2020)

[WEB 15] https://stph.scenaricommunity.org/contribs/nos/Hadoop1/co/a_YARN_ecosysteme.html (Consulté le 20/03/2020)

[WEB 16] https://www.stat4decision.com/fr/spark-boite-a-outils-du-big-data/ (Consulté le 01/05/2020)

[WEB 17] https://intellipaat.com/blog/what-is-apache-spark/ (Consulté le 01/05/2020)

[WEB 18] https://doc.ubuntu-fr.org/ubuntu_distribution (Consulté le 21/06/2020)

[WEB 19] https://www.windows8facile.fr/installer-java-ubuntu-jdk-jre/ (Consulté le 21/06/2020)

[WEB 20] https://docs.scala-lang.org/tour/tour-of-scala.html (Consulté le 22/06/2020)

[WEB 21] https://www.webopedia.com/TERM/O/OpenSSH.html (Consulté le 22/06/2020)

[WEB 22] https://spark.apache.org/docs/latest/ (Consulté le 22/06/2020)

[WEB 23] https://hadoop.apache.org/ (Consulté le 22/06/2020)

[WEB 24] https://www.pierre-giraud.com/html-css-apprendre-coder-cours/definition-utilisation/ (Consulté le 26/09/2020)

[WEB 25] https://www.futura-sciences.com/tech/definitions/internet-css-4050/ (Consulté le 26/09/2020)

[WEB 26] https://techterms.com/definition/javascript (Consulté le 26/09/2020)

[WEB 27] https://techterms.com/definition/php (Consulté le 26/09/2020)

[WEB 28] https://www.linuxjournal.com/content/sublime-text-one-editor-rule-them-all (Consulté le 26/09/2020)

[WEB 29] https://doc.ubuntu-fr.org/apache2 (Consulté le 26/09/2020)

Annexes

Annexes

Annexe A : Problèmes rencontrés lors de l'installation de Hadoop

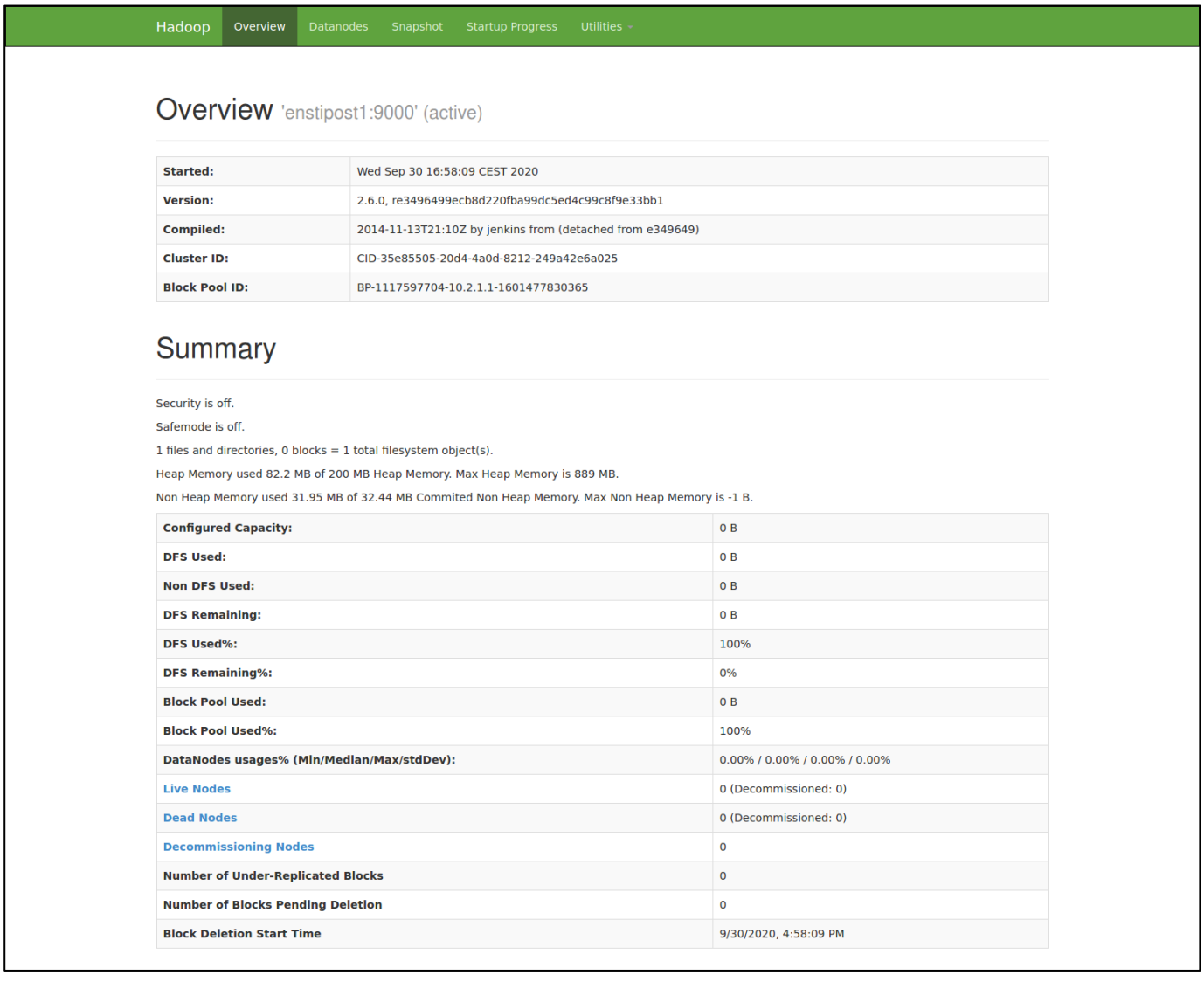

Figure 70. Problèmes installation de Hadoop

L'un des plus gros problèmes que nous avons rencontré lors de l'installation et la configuration de Hadoop et que toutes les informations sont à zéro après avoir accéder à l'interface graphique de Hadoop comme le montre la figure ci-dessus, malgré que toutes les configurations sont correctes. Après de longues recherches et un tas de tests effectués nous avons trouvé la solution pour régler ce problème. La solution se résume tout simplement à la suppression d'un dossier nommé *''current''* situé dans le répertoire *''opt/hadoop/hadoo2_ data/hdfs/datanode''*, il faut le supprimer dans tous les nœuds slaves.

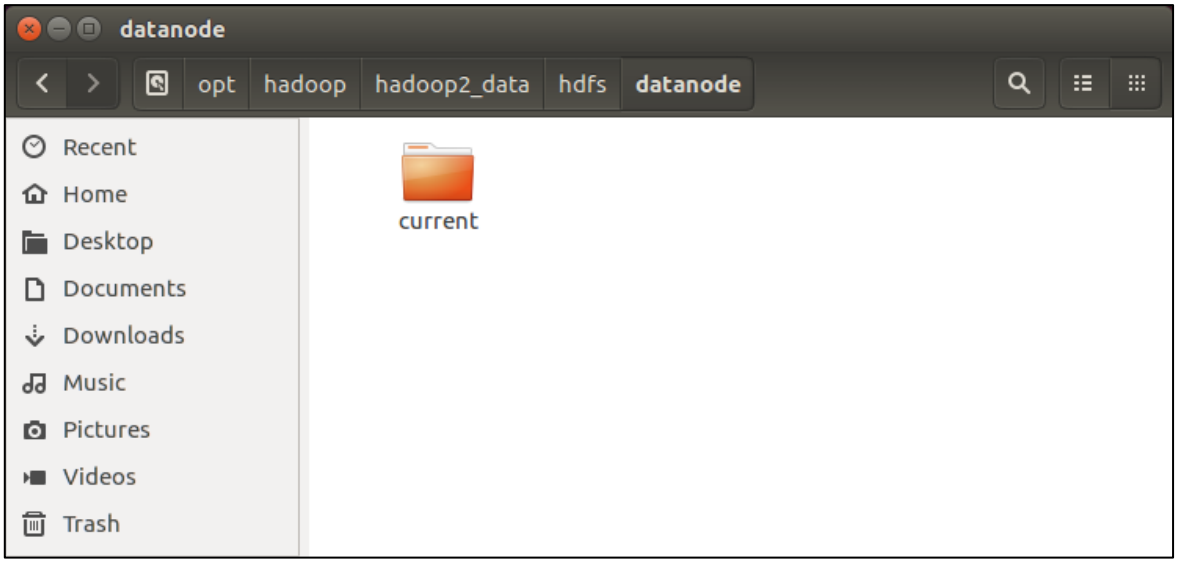

Figure 71. Le dossier ''current''

Après la suppression du dossier dans tous les nœuds esclaves, on formate à nouveau le système de fichier Hadoop en utilisant la commande.

```
$ hadoop namenode -format
```

On actualise l'interface graphique de Hadoop, le problème sera résolu.

Important :

- Lors de l'installation de Ubuntu qu'on attribue les identifiants des postes, il faut éviter de donner des identifiants qui contient des caractères spéciaux (#, !, *, ... etc.), et éviter aussi de donner des identifiants qui contient "-" ou "_", uniquement des lettres et des chiffres.

- Il faut désactiver l'IPv6, et l'attribution automatique des adresses IP, et attribuer des adresses IP au machines d'une manière manuelle.

- Vérifier l'interconnexion de toutes les machines en accédant du nœud Master vers tous les autres nœuds Slaves en utilisant SSH.

- Il faut toujours utiliser l'utilisateur *"ensti_ cluster"* pour utiliser le Cluster.

Annexe B : Le fichier "sc.sh" utilisé pour la génération du fichier "prenom.txt"

Le fichier *sc.sh* est le Script Shell que nous avons utilisé pour générer le fichier *"prenom.txt"*, pour que le fichier soit volumineux pour pouvoir voir les différences lors de l'exécution du Job WordCount. Le code source écrit en Shell Script est le suivant :

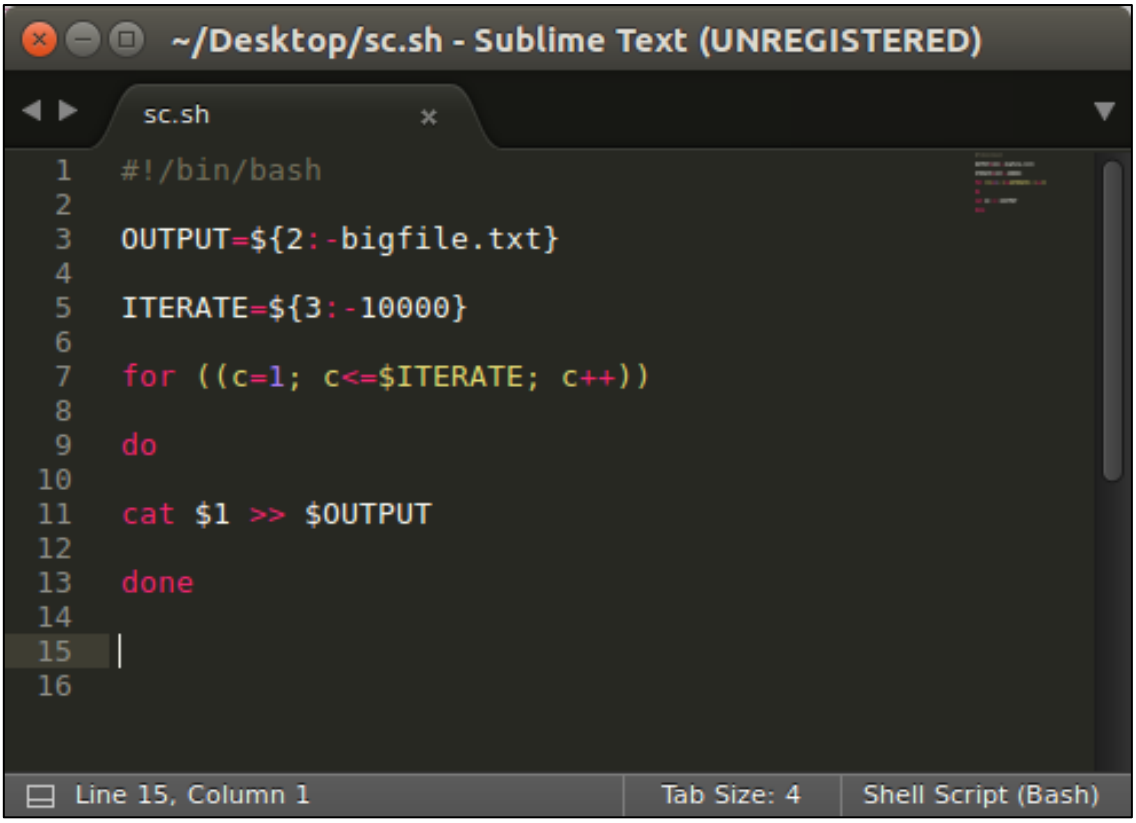

Figure 72. Le fichier sc.sh

Annexe C : Le Cluster Hadoop avant et après mise en place

Figure 73. Le Cluster Hadoop au premier jour du projet

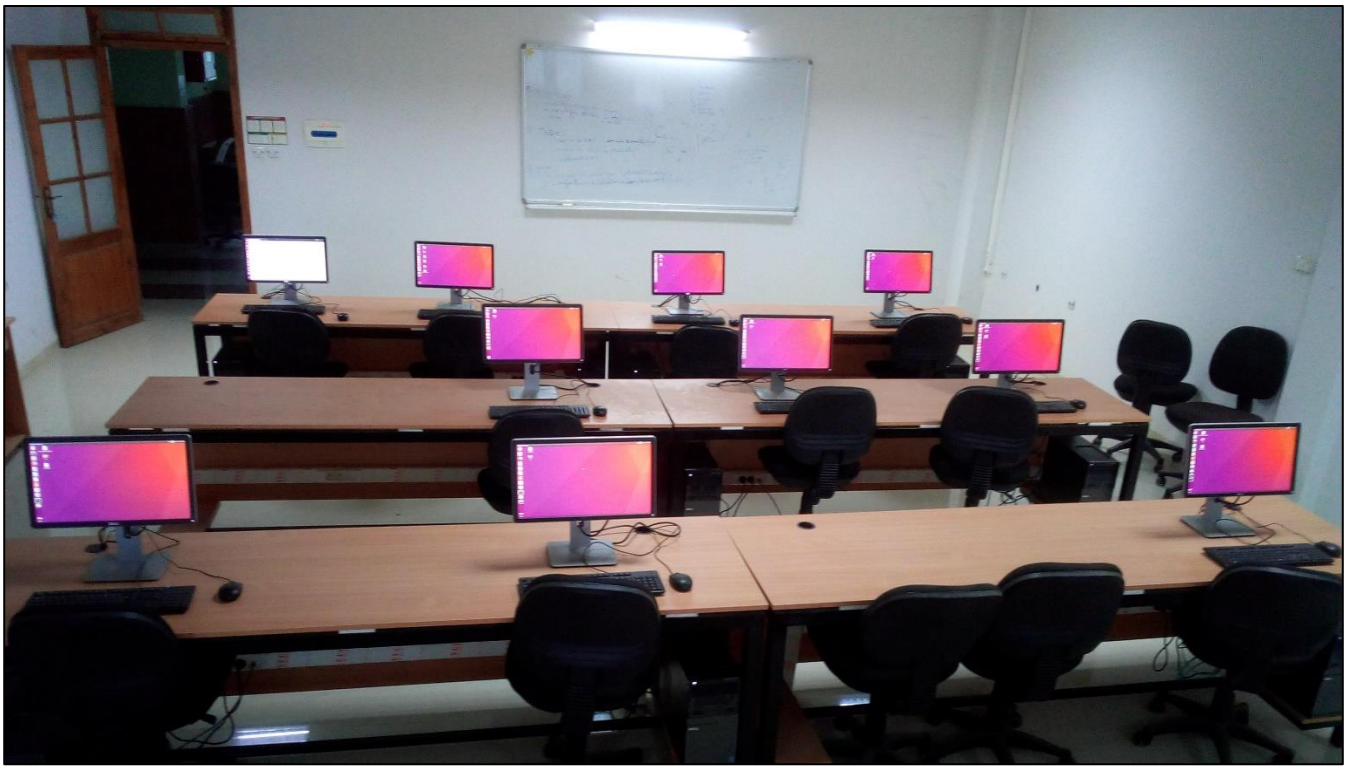

Figure 74. Le Cluster Hadoop au dernier jour du projet