Ordre...../F.S.S.A/UAMOB/2018

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AKLI MOHAND OULHADJ-BOUIRA

Faculté des Sciences et des Sciences Appliquées
Département d'Informatique

Mémoire de fin d'étude

Présenté par :

Mlle MEDJRI Kahina
Mlle HADDOUCHE Soumia

En vue de l'obtention du diplôme de **Master 02**en **:**

Filière : INFORMATIQUE
Option : Ingénierie des systèmes d'information et du logiciel

Thème :

Mise en place d'un système d'authentification pour hadoop

Devant le jury composé de :

BOUSSAADI Smeil	Grade	UAMOB	Président
BOUDJELABA Hakim	MAA	UAMOB	Encadreur
BENZAOUI Amir	Grade	UAMOB	Examinateur
BRAHIMI Farida	Grade	UAMOB	Examinateur

Année Universitaire 2018/2019

Remerciements

nous tenons à la fin de ce travail de remercier Allah Tout Puissant, de nous avoir donné Le bon sens et la grande volonté pour réaliser ce modeste travail.

Nous voudrions exprimer nos vifs remerciements à notre encadreur ***Mr BOUDJELABA HAKIM*** *pour son aide, sa patience et son encouragement qu'il nous a apportée tout au long de notre préparation de ce mémoire et du fait qu'il a bien voulu diriger ce travail et nous faire profiter de son savoir et de ses précieux conseils.*

Nos remerciements vont également aux membres du jury qui ont acceptés d'évaluer mon travail.

Pour la même occasion, nous remercions tous ceux qui ont participé de près ou de loin, de façon directe ou indirecte, à la réalisation de ce projet, nos enseignants, nos amis, nos collègues et toute la promotion de 2019.

Nos remerciements vont a Mr Hambli qui nous a aider durant ces dernier deux mois, a l'occasion nous voudrions dire merci a toute l'équipe de service informatique a l'université centrale de Bouira.

Enfin, nous espérons que ce travail aura la valeur souhaitée.

Merci à tous.

Dédicaces

Je dédie ce modeste travail à :

A mes parents ***"Mouloud"*** **et** ***"Adidi"*** ,

A mes frères et soeurs, Et mon ange ***"Ouail Abde Raouf"***

A toute la famille,

A mes amies et collègues, et tous ceux qui m'ont aidé .

medjri kahina

Dédicaces

Je tien à dédier ce modeste travail à ma raison de vivre, d'espérer, à ma source de courage, à ceux qu'on a de plus chères, mes parents*" **hamid** et **karima** "*, pour leurs encouragements et leurs sacrifices sans limite, pour que je puisse franchir tout obstacle durant toutes mes années d'étude.

Que le dieu me les gardent en très bonne santé et me les protège et que la réussite soit toujours à ma portée pour que je puisse les combler de bonheur.

A mes très chères soeurs et frère :
Nadia, Sarah, Abderrahmane

A mon petit neveux que j'aime très fore ***Ilyane***

A mes chères Beau-frère : ***Mohammed et hacen***

A tous mes tantes et oncles, mes cousins et cousines.

A ***karim*** mon chére cousin qui m'aide toujours.

A mes enseignants pour leurs patience, leurs soutien, leurs encouragements et à mes amis Pour leur témoigner une amitié et fidélité indéfinies.

A tous les étudiants Master informatique ISIL et GSI pour leur aide morale durant toute la période de préparation.

A tous ceux qui me sens chers et que j'ai omis de citer.

Haddouche Soumia

Résumé

À l'heure actuelle, Hadoop est la principale plate-forme du Big Data, il permet de manipuler des données volumineuses. Par défaut, Hadoop n'authentifie pas les utilisateurs. Il est recommandé à activer l'authentification pour le cluster afin de protéger les données contre les menaces internes et externes au réseau. Pour un niveau de sécurité supérieur, nous avons configuré l'authentification Kerberos qui est un protocole d'authentification réseau largement utilisé et implémenté dans l'écosystème Hadoop. Kerberos fournit un moyen de vérification de l'identité des utilisateurs, la vérification est implémentée via un modèle client / serveur. Lorsque l'authentification Kerberos est activé, les utilisateurs doivent s'authentifier avant d'interagir avec les services Hadoop.

Mots clés : Hadoop, Authentification, Kerberos, Big Data, HDFS, SSL.

Abstract

At present, Hadoop is the main platform of Big Data, it can handle large data. By default, Hadoop does not authenticate users. It is recommended to enable authentication for the cluster to protect the data from both internal and external threats to the network. For a higher level of security, we have configured Kerberos authentication which is a widely used network authentication protocol implemented in the Hadoop Big Data ecosystem. Kerberos provides a way of verifying the identity of the users, the verification is implemented via a client / server model. When Kerberos authentication is enabled, users must authenticate before interacting with Hadoop services.

Key words : Hadoop, Authentication, Kerberos, Big Data, HDFS, SSL.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

RH	Resource Hummaine
RAID	Redundant Array of Independent Disks
GoogleFS	Google File Systeme
HDFS	Hadoop Distribuated File Systeme
YARN	Yet Another Resource Negotiator
ABAC	Attribute Based Access Control
RBAC	Role Based Access Control
PAP	Password Authenfhentication Protocol
CHAP	Challenge Handshake Authentication Protocol
SSL/TLS	Secure Socket Layer/Transport Layer Security
RADIUS	Remote Authentication Dial-In User Service
LDAP	Lightweight Directory Access Protocol
MIT	Massachusetts Institue of Technology
KDC	Key Distribution Center
AS	Authentification Service
TGS	Ticket Granting Service
TS	Ticket Service
TGT	Ticket Granting Ticket
AP	Access Point
AAA	Authentication Autorization Accounting
RFC	Request For Comments
FAI	Fournisseur d'Accés Internet

NAS	Network Access Server
HTTPS	Hypertext Transfer Protocol Secure
OCSP	Online Certificate Status Protocol
DB	Data Base
JVM	Java Virtual Machine
SSH	Secure SHell
JPS	Java Virtual Machine Process Status Tool
ACL	Access Control List
RPC	Remote Procedure Call
SASL	Simple Authentication and Security Layer
CA	Certificate Authority
NFT	Network File System

Introduction générale

Le Big Data est une nouvelle révolution dans le domaine informatique, elle est relative en taille des données qui deviennent tellement grosses et difficiles à gérer avec des outils classiques de gestion de base de données, ce qui nécessite l'utilisation des plateformes et outils dédiés à la gestion de ces données parmi lesquels la plateforme Hadoop qui est un framework open source, il intègre le stockage et le traitement des données, la gestion du système et un outil d'entreposage de données. Le cœur de Hadoop comprend trois composants essentiels à savoir : HDFS qui est un système de gestion de fichier distribué, MapReduce qui est un nouveau paradigme de programmation sur lequel sont effectués les calculs parallèles et distribués de grandes masses de données et Yarn qui est une technologie qui gère l'utilisation des ressources dans un cluster.

Hadoop a été développé au début sans la moindre mise en place d'un système de sécurité. Ce framework ne possède aucun mode d'authentification des utilisateurs, aucun cloisonnement de données privées et toute personne est autorisée à exécuter un code. Tous les utilisateurs et programmeurs ont le même privilège d'accès sur toutes les données dans n'importe quel cluster où elles se trouvent, n'importe qui dispose du droit de lecture sur celles-ci. Il n'existe aucun chiffrement lors des échanges des informations entre les différents nœuds ou entre un nœud et un client. Il se pourrait qu'un utilisateur malveillant veuille diminuer les priorités des autres processus de Hadoop pour que ses travaux s'effectuent plus rapidement ou bien même supprimer les autres calculs en cours d'exécution.

Notre objectif consiste à prévoir une solution d'authentification permettant de sécuriser l'accès des utilisateurs pour Hadoop.

Pour réaliser notre travail, nous allons suivre le plan suivant :

— **Chapitre 1** donne une vision globale sur l'état de l'art de Big Data, sa technologie hadoop et une présentation générale des différents problèmes de sécurité rencontrés dans ce domaine.

— **Chapitre 2** nous allons présenter les techniques d'authentification d'une manière générale, et particulièrement les techniques les plus utilisées.

— **Chapitre 3** c'est le noyau de notre travail. Ce chapitre défini l'environnement de travail, ainsi les étapes d'installation et configurations que nous avons effectués pour sécuriser Hadoop.

Enfin, notre travail se clôture par une conclusion générale, décrivant les éléments essentiels que nous avons effectués dans ce mémoire, ainsi que quelques perspectives pour ce projet.

Chapitre 1

Big Data

1.1 Introduction

Le Big Data est un phénomène qui a vu le jour avec l'émergence de données volumineuses qu'on ne pouvait pas traiter avec des techniques traditionnelles.
la notion de Big Data, issue en grande majorité d'appareils connectés *(ordinateurs, tablettes, smartphones,...)*. Grâce aux progrès de la technique, ces données sont enregistrées et des valeurs significatives sont extraites. Ces données ont le potentiel d'être exploitées à des fins d'information.
Le Big Data est devenu une tendance incontournable pour beaucoup d'acteurs industriels du fait de l'apport qu'il offre en qualité de stockage, traitement et d'analyse de données.

Dans ce chapitre, nous allons présenter un état de l'art sur le Big Data et une présentation générale des différents problèmes de sécurité rencontrés dans ce domaine.

1.2 Big data

Le Big Data s'impose comme l'une des évolutions majeures des Systèmes d'information, à la fois sur les plans métiers, fonctionnels et technologiques [11].
à travers cette partie introductive, nous allons présenter les principaux concepts, les caractéristiques, les enjeux et l'usage du big data.

1.2.1 Définition

Le terme de Big data *(parfois appelées « données massives » en français)* désigne une nouvelle discipline qui se situe au croisement de plusieurs domaines : statistiques, technologie, base de données et métiers *(marketing, finance, RH, etc).*
Le Big data a pour objectif d'exploiter des volumes de données qui sont en croissance exponentielle et qui deviennent difficiles à travailler avec des outils classiques de gestion de base de données ou de gestion de l'information. Elle a aussi pour objectif de traiter rapidement des données complexes [2].

Donc, il n'existe pas une définition unique pour le terme Big Data. Voici les quelques définitions les plus utilisées :

- **Littéralement :** ce terme signifie mégadonnées ou grosses données. Ils désignent un ensemble très volumineux de données qu'aucun outil classique de gestion de base de données ou de gestion de l'information ne peut vraiment travailler [12].
- **Théoriquement :** le Big data est le moyen d'étudier de très grande quantité de données afin d'établir des modèles originaux qui nous offriront une vision plus fine de la réalité et nous permettrons de prendre des décisions plus pertinentes[13].
- **Généralement :** le mot big data cache en réalité un ensemble de solutions, technologiques qui permettent d'analyser et de traiter le contenu audio, vidéo et textuel. Ces technologies sont assez récentes et sont pour la plupart issues des grandes sociétés du web américain telles que Google [1], Yahoo [2] ou encore linkedin [3] [14].

1.2.2 Caractéristiques du Big data

Le big data, est un ensemble de données numériques produites par l'utilisation de nouvelles technologies qui sont caractérisées par les 5V et devient difficile à traiter et a analyser en utilisant les outils classiques de gestion de base de données.

Les 5V représentent : le volume, la vélocité, la variété, la véracité et la valeur des données [15].

1. https ://www.google.com/
2. https ://fr.yahoo.com/
3. https ://www.linkedin.com/

FIGURE 1.1 – Les Caractéristiques du Big data[1]

1. Volume : il représente la quantité massive des données générées.
2. Vélocité : signifie la rapidité de la collection et de l'analyse des données.
3. Variété : le type et le format de données générées sont différents.
4. Véracité : correspond à la fiabilité des données. La précision et la qualité des données sont très importantes. Donc il faut s'assurer que les données du big data sont vraies pour pouvoir prendre des décisions.
5. Valeur : le V le plus important, l'objectif est de créer des valeurs pour les entreprises et les clients en transformant toutes les données en valeurs exploitables.

1.2.3 Architecture du Big Data

Le succès du fonctionnement de le Big data dépend de son architecture, son infrastructure correcte et de son utilité que l'on fait « *Data into Information into Value* ».
L'architecture de la Big data est composé de 4 grandes parties : intégration, data processing and stockage, sécurité et opération comme le montre le schéma si dessous [2] :

FIGURE 1.2 – Architecture du Big data [2]

— **Intégration :** consiste à charger le volume de données au sein du stockage.
— **Stockage de données (Data Storage) :** le stockage des données massives et volumineuses dans lequel il faut faire très attention à réduire les coûts de stockage au sein de l'entreprise.
— **Manipulation de données (Data Processing) :** il s'agit de la manipulation et du traitement de données appelé Map Reduce.
— **Sécurité :** sert à l'autorisation, l'authentification et la protection des données.
— **Opérations :** pour la gestion, le monitoring et les taches planifiés.

1.2.4 Les enjeux du Big data

Le Big Data semble aujourd'hui incontournable compte tenu de la place prépondérante du numérique dans le quotidien du consommateur lambda. Cependant, il convient de maîtriser ses enjeux afin de l'apprivoiser et d'en tirer profit [16].

1. Garantir la qualité des informations : la qualité des données doit être une priorité pour ne pas fausser les stratégies découlant de leur traitement.
2. Optimiser le traitement des données : pour de nombreux experts, le traitement des données est l'un des enjeux les plus importants du Big Data. En effet, les informations arrivent en masse et se présentent sous divers formats.
3. Mettre en relation tous les métiers : le Big Data implique de faire travailler ensemble différents professionnels dans le but d'atteindre des objectifs précis.
4. Assurer la sécurité : la sécurité a une importance particulière, car elle engage la responsabilité et la réputation de l'enseigne.

5. Humaniser les données : l'intérêt du Big Data est de placer les clients au centre du processus décisionnel, mieux informés.

1.2.5 Classification des Big Data

Identifier les caractéristiques des données est utile pour extraire ses modèles cachés. Les Big Data sont classées en dix catégories en termes de type de données, format de données, source de données, consommateur de données, utilisation de données, analyse de données, stockage de données, fréquence de données, proposition de traitement de données et méthode de traitement de données [17]. (comme illustrée par la FIGURE 1.3)

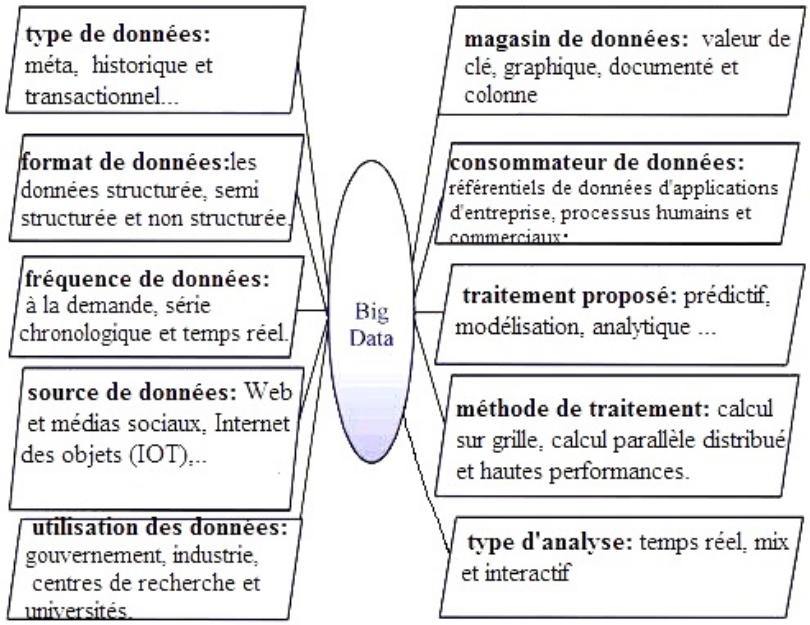


FIGURE 1.3 – Classification des Big Data

1.2.6 Domaines d'utilisations du Big Data

Une grande partie des cas d'usage du Big Data existaient déjà avant son émergence. Les nouvelles techniques permettent cependant d'aller plus vite et de traiter plus de données. Car aujourd'hui, il existe beaucoup plus de données générées automatiquement (issues du web, des appareils mobiles et de capteurs divers). La plupart des contextes d'utilisations actuelles du Big Data se résume en quelques termes : pressentir la naissance d'une ten-

dance, prédire l'évolution d'un phénomène, repérer des corrélations pour optimiser une stratégie, faire des contrôles pour découvrir une fraude, communication virale,...[18]

Le Big Data couvre de nombreux domaines d'applications telles que l'industrie, les banques, l'assurance, le transport, l'éducation, loisirs et le télécom, etc. (comme illustrée par la FIGURE1.4)

FIGURE 1.4 – Les domaines d'usage du big data[3]

1.2.7 Intérêts des Big Data

L'utilisation des Big Data pourrait impacter fortement le monde de l'entreprise et ce de façon méliorative, ainsi les entreprises pourront [19] :

- Améliorer la prise de décision.
- Réduire les coûts d'infrastructures informatiques via l'utilisation des serveurs standards et des logiciels open source.
- Développer la réactivité et l'interactivité à l'égard des clients.
- Améliorer les performances opérationnelles.

1.2.8 Les avantages du Big data

Plusieurs avantages peuvent être associés à une architecture Big Data, citons par exemple :

— **Extensibilité *(scalabilité)* :** le concept Big Data apporte une architecture scalable qui peut prévenir la taille de l'infrastructure et l'espace disque nécessaire.

— **Performance :** grâce au traitement parallèle des données et à son système de fichiers distribué, le concept Big Data est hautement performant en diminuant la latence des requêtes.

— **Coût faible :** le principal outil Big Data à savoir Hadoop est en Open Source, en plus on n'aura plus besoin de centraliser les données dans des baies de stockage souvent excessivement chère, avec le Big Data et grâce au système de fichiers distribués les disques internes des serveurs suffiront.

— **Disponibilité :** on a plus besoin des RAID disques, souvent coûteux.
L'architecture Big Data apporte ses propres mécanismes de haute disponibilité [20].

1.2.9 Les inconvénients de Big data

— Néanmoins le Big DATA présente de nombreux risques d'atteinte à la vie privée et aux droits fondamentaux Le respect de la vie privée est encadré, de façon toute relative, par la loi «informatique et libertés».

— Déshumanisation : dans ce que Bruce Schneier dénomme «l âge d'or de la surveillance», la plupart des individus peuvent se sentir déshumanisés et ils ne peuvent plus protéger les données personnelles ou non qui les concernent, et qui sont collectées, analysées et vendues à leur insu [21].

1.2.10 Technologies du Big Data

Il existe plusieurs technologies du Big Data répondant à ces besoins comme illustré dans la FIGURE 1.5. Mais la plus répandue des solutions est bien évidemment Apache Hadoop, un framework largement utilisé aujourd'hui pour traiter de très gros volumes de données [22].

FIGURE 1.5 – Les technologies du Big Data[4]

Hadoop

Hadoop est un Framework libre et open source écrit en Java destiné à faciliter la création d'applications distribuées *(au niveau du stockage des données et de leur traitement)* et échelonnables *(scalables)* permettant aux applications de travailler avec des milliers de nœuds et des pétaoctets de données. Ainsi chaque nœud est constitué de machines standard regroupées en grappe [15].

Hadoop a été créé en 2005, basé sur des travaux de Google publié en 2004 sur le Map/Reduce et sur GoogleFS, un système de fichier distribué. C'est Doug Cutting qui l'a créé et qui a choisi le nom et le logo grâce à la peluche de son fils, un éléphant jaune qu'il appellait Hadoop[23].

Hadoop est composé de plusieurs éléments : un système de stockage *(HDFS)*, un système de planification des traitements *(YARN)* et le framework de traitement *(MapReduce)* [22].

Apache HDFS : est un système de fichier distribué conçu pour le stockage de gros volumes de données et peut fonctionner sur plusieurs machines de faible coût. Il offre aux applications un accès rapide aux données. HDFS est adapté pour les applications qui traitent une grande quantité de données. Il est largement inspiré de GoogleFS [24].

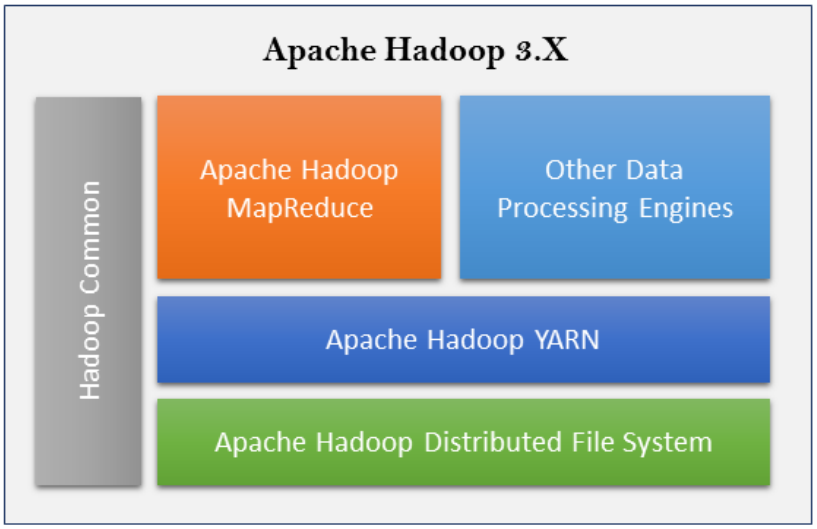


FIGURE 1.6 – Architecture Hadoop

L'architecture est faite de la façon suivante. HDFS se compose d'un nœud principal, appelé le NameNode. Ce nœud est très important car c'est lui qui va gérer l'emplacement de l'ensemble des données. Il fait la correspondance entre un fichier et ses blocs associés (les metadata d'un fichier), et il sait également sur quels nœuds chaque bloc se situe. Sur les autres noeuds se trouvent les DataNode. Un DataNode va gérer les blocs de données présent sur son nœud. Le DataNode tiens très souvent le NameNode au courant des blocs qu'il gère, et c'est avec ce principe qu'il est possible au NameNode de détecter des problèmes et de demander la réplication des blocs. Les Datanodes ne gèrent pas de fichiers, mais seulement des blocs. La notion de fichier est géré par le NameNode. Il va pouvoir ouvrir, fermer, supprimer des fichiers, et répercuter ces changements aux Datanodes concernés. Il va donc demander aux DataNodes de créer des blocs, de les supprimer, de les lire ou écrire dedans. Le NameNode peut poser problème en cas de défaillance de son nœud, c'est pour cela qu'il existe un Secondary NameNode, qui va recevoir de temps en temps les données du NameNode, et qui va pouvoir, en cas de défaillance du NameNode prendre sa place. Les données sont donc bien sécurisées dans la plupart des cas, et leur accès est donc garantie[23]. (comme illustrée par la FIGURE 1.9)

Apache Hadoop YARN : est une technologie de gestion de clusters. Elle rend l'environnement Hadoop mieux adapté aux applications opérationnelles qui ne peuvent pas attendre la fin des traitements par lots[25].

Le fonctionnement de son architecture se fait avec un système maîtres/esclaves. Le Res-

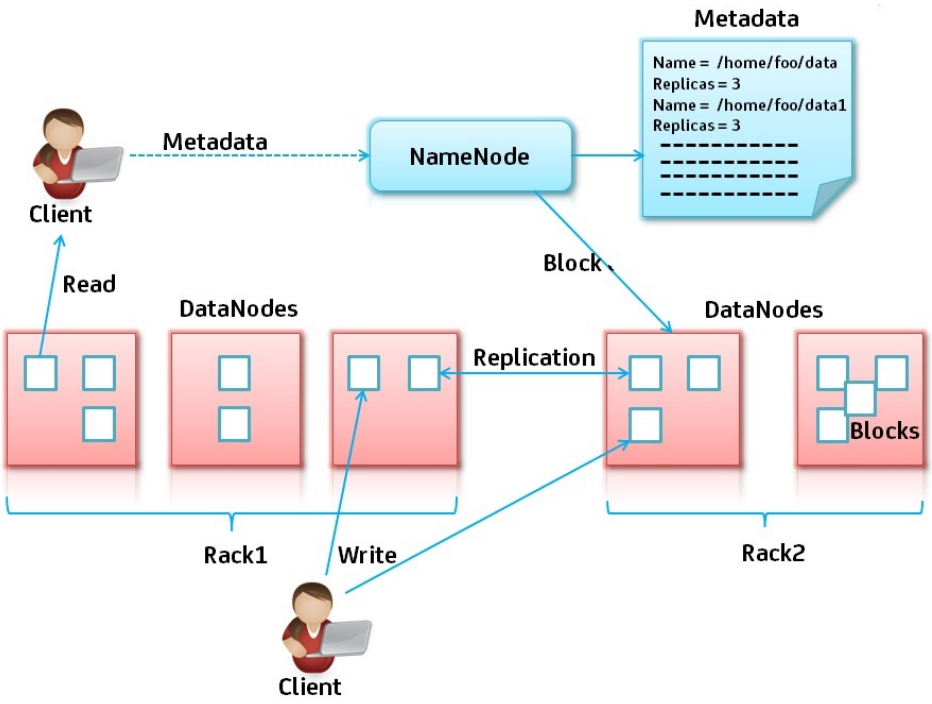


FIGURE 1.7 – Architecture HDFS[5]

sourceManager représente l'autorité suprême du cluster. Il est composé de deux rôles : la gestion des ressources du cluster et la gestion des applications. Il va donc gérer la soumission des applications sur le cluster, et va donc assigner à chaque application des ressources d'un nœud (ce qu'on appelle un conteneur ou Container) qui pourra gérer l'exécution de cette application. L'exécution des applications n'est donc pas centralisé sur un seul nœud. Chaque application aura donc son ApplicationMaster tournant sur un nœud du cluster. La gestion des ressources du cluster se fait avec le Scheduler, qui fait parti du ResourceManager. Il va devoir assigner aux ApplicationMaster des ressources venant de nœuds suivant la demande de ces-derniers, et suivant le type d'ordonnancement. Les applications peuvent être différentes, mais elles ne sont traitées selon leurs types par le Scheduler, elles sont traitées par leurs demande en ressources sur le cluster. La FIGURE 1.8 schématise un exemple de distribution des ressources d'un cluster pour deux applications[23].

Chaque nœud du cluster est composé d'un NodeManager, qui va gérer les demandes de ressources sur ce nœud. Il va tenir le ResourceManager au courant grâce au heartbeat. Le heartbeat est envoyé par tous les nœuds au ResourceManager pour donner ses informations. Les ressources demandées, regroupées en conteneurs, sont des ressources d'une

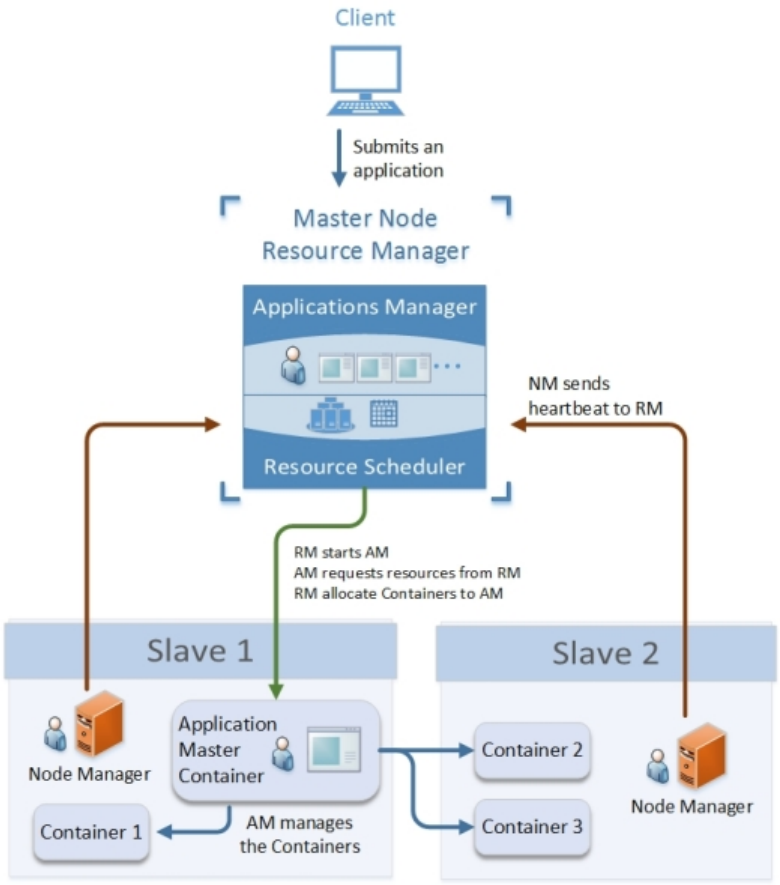


FIGURE 1.8 – Architecture YARN[6]

machine : la mémoire, le disque et le réseau, etc [23].

Apache Hadoop MapReduce : est un modèle de programmation créé par Google pour le traitement et la génération de larges ensembles de données sur des clusters d'ordinateurs dans lequel sont effectués des calculs parallèles, et souvent distribués, de données potentiellement très volumineuses, typiquement supérieures en taille à 1 téraoctet.
Il s'agit d'un composant central du Framework logiciel Apache Hadoop, qui permet le traitement résilient et distribué d'ensembles de données non structurées massifs sur des clusters d'ordinateurs, au sein desquels chaque nœud possède son propre espace de stockage.
MapReduce est composée de deux processus essentiels sont « *JobTracker* » et des nœuds appelés « *TaskTracker* ». Le JobTracker doit savoir l'état du cluster et l'ensemble des nœuds qui sont actifs et c'est lui qui recevra les différentes tâches à effectuer pour les distribuer par la suite aux autres nœuds ou TaskTracker qui vont réaliser ces tâches. Un

TaskTracker est chargé d'exécuter des tâches de Map ou de Reduce [2].

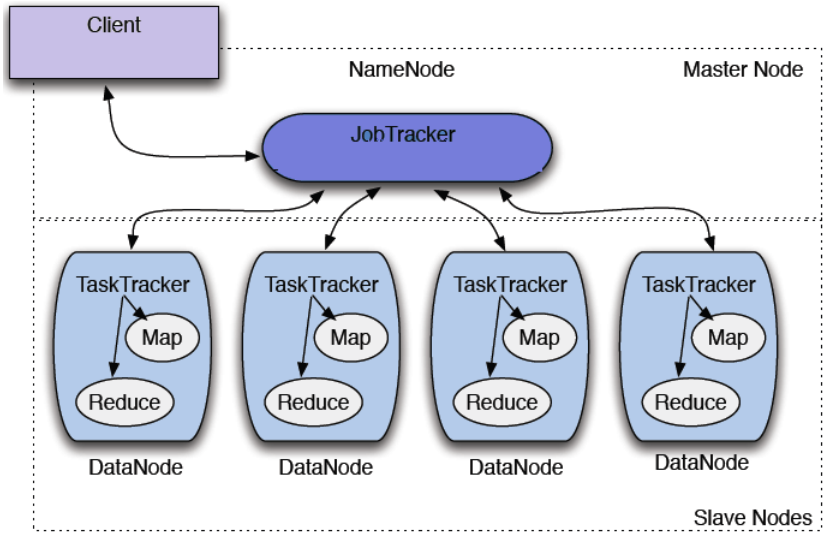


FIGURE 1.9 – Architecture de MapReduce[7]

1.3 La sécurité dans le Big Data

La sécurité apparaît comme un des obstacles majeurs dans le Big Data. Ces obstacles ne sont pas encore vraiment résolus, Avec la variabilité et l'augmentation du volume des données échangées. Plusieurs entreprises spécialisées dans la sécurité des données sont apparues, mais elles se focalisent sur un ou plusieurs composants seulement.
D'autre part, les progrès technologiques vont vite que l'évolution de ces solutions. Mais afin de minimiser les risques, il faut au moins respecter les quatre règles de sécurité suivantes :

1.3.1 Authentification

L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il prétend être. Pour protéger le big data, il est essentiel d'intégrer un outil permettant l'identification de la personne ou un processus, avant de lui permettre de se connecter.

Il existe plusieurs techniques d'authentification qu'on peut classer en deux catégories :

— L'authentification basique dite simple :

Qui suis-je : identifiant *(login)*.

Prouve-le : authentifiant *(mot de passe)*.

— L'authentification forte : authentification simple renforcée

Pour renforcer l'authentification simple, pas assez sécurisée, il faut rajouter des « *verrous* » : empreinte biométrique, carte à puce, etc [26].

1.3.2 Autorisation

Une fois connectée, la personne n'est pas autorisée à tout faire ou à accéder à l'ensemble des ressources. Plusieurs techniques peuvent être utilisées.
La plus ancienne est celle utilisant ABAC *(Attribute Based Access Control)* : contrôle d'accès basé sur les attributs. Une autre technique, la plus répandue, est celle des RBAC *(Role Based Access Control)* qui permet de contrôler les accès en se basant sur les rôles [26].

1.3.3 Audit

Bien que la personne soit identifiée, authentifiée et autorisée à accéder à certaines ressources, un système bien sécurisé doit pister toutes les opérations de cette dernière. Il s'agit de la traçabilité, un processus essentiel [26].

1.3.4 Cryptage des données

Afin de lutter efficacement contre les menaces toujours évolutives, il faut également adopter une approche centrée sur les données afin de protéger les informations sensibles *(telles que les informations des cartes bancaires ou des dossiers confidentiels de patients)*. Pour se faire, vous pouvez également recourir au cryptage de certaines données, stockées en sécurité et rendues inintelligibles via un algorithme de cryptage préservant leur confidentialité. La clé de cryptage doit être strictement protégée afin de rendre efficace ce procédé [26].

Les quatre règles déjà présentées devront être la base sur laquelle faut s'appuyer pour réaliser votre stratégie big data. Si vous ne respectez pas ces règles, vous mettez en péril vos données. Pour cette raison vous devez bien respecter ces règles de manière rigoureuse [26].

1.4 Conclusion

Big Data ne peut être décrit uniquement par sa taille, c'est un écosystème large et complexe. Il nécessite la maitrise des technologies matérielles et logicielles diverses, il demande de la compétence et de l'expertise dans la maitrise et l'analyse des données. L'accumulation de données contribue à améliorer le service à la clientèle de plusieurs façons.

Toutefois, de telles quantités énormes de données peuvent également soulever de nombreux problèmes de confidentialité, ce qui fait de la sécurité des Big Data une préoccupation majeure pour toutes organisations.

Chapitre 2

Authentification

2.1 Introduction

L'authentification est une procédure, par laquelle un système informatique certifie l'identité d'une personne ou d'un ordinateur. Le but de cette procédure étant d'autoriser la personne à accéder à certaines ressources sécurisées. Il va comparer les informations des utilisateurs autorisés stockées dans une base de données *(en local ou sur un serveur d'authentification)* à celles fournies. L'accès sera autorisé seulement si les informations sont identiques. C'est l'administrateur du système d'information qui octroie les droits et paramètre l'accès.

Dans ce chapitre nous allons définir de manière générale les différentes techniques d'authentification les plus utilisées.

2.2 Définition

L'authentification c'est le moyen qui consiste à vérifier l'identité d'un utilisateur avant de lui donner l'accès à une ressource, généralement l'authentification est précédée d'une identification qui permet à cette entité de se faire reconnaître du système par un élément dont on l'a doté. En résumé, s'identifier c'est communiquer son identité, s'authentifier c'est apporter la preuve de son identité. [27].

2.3 Facteurs d'authentification

Un facteur d'authentification est une catégorie des données d'authentification qui sert à vérifier l'identité. On dénombre trois grandes catégories :

1. Facteurs mémoriels : cette catégorie des données d'authentification se compose des informations que l'utilisateur connaît *(exemple un code confidentiel, un nom d'utilisateur, un mot de passe ou la réponse à une question secrète).*
2. Facteurs matériels : cette catégorie s'appuie sur des objets que l'utilisateur possède. Il s'agit généralement d'un dispositif matériel, comme un jeton de sécurité ou un téléphone mobile utilisé conjointement à un jeton logiciel.
3. Facteurs corporels : cette catégorie d'informations d'authentification d'un utilisateur est formée des éléments constitutifs de la personne en question sous forme de données biométriques*(exemple : voix, empreinte digitale).*

Le lieu où se trouvent l'utilisateur et l'heure en cours sont parfois considérés comme les quatrième et cinquième facteurs de l'authentification. [28].

2.4 Types de méthodes d'authentification

On distingue 3 familles de protocoles d'authentification :

1. Authentification simple : se base sur la vérification d'un seul facteur d'authentification*(Exemple : login et mot de passe).*
2. Authentification forte : combinaison de la vérification de deux facteurs ou plus. L'authentification est considérée comme réussie seulement si toutes les vérifications individuelles des éléments sont validées*(Exemple : jeton de sécurité + scan de l'iris).*
3. Authentification unique (Single-Sign-On) : on s'authentifie une fois pour toute et ensuite on peut accéder à plusieurs applications ou services situés sur des serveurs différents *(Exemple : Kerberos)* [29].

2.5 Les protocoles d'authentification

Il existe plusieurs protocoles d'authentification et méthodes disponibles. Chaque protocole d'authentification emploie certaines méthodes pour réaliser l'authentification, bien que la mise en œuvre puisse différer en termes de robustesse et des processus impliqués. Presque tous les protocoles d'authentification ont la particularité d'utiliser des secrets, soit pré-partagés ou dérivés pour mener le processus d'authentification d'identité. Ils exploitent habituellement des nombres aléatoires, des fonctions de hachage, des défis, des estampilles temporelles pour améliorer la robustesse ou ajouter des fonctionnalités au protocole. Comme exemple de protocoles d'authentification, nous avons le protocole Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), SSL/TLS, RADIUS, Kerberos, LDAP, etc.[30].

2.5.1 Kerberos

Kerberos est un protocole d'authentification développé par le MIT *(Massachusetts Institue of Technology)*. Il a été conçu afin de fournir une authentification unifiée pour les applications de type client/serveur sur des réseaux qualifiés de non sûrs à l'aide de chiffrement symétrique. L'authentification repose sur une tierce partie de confiance nommée Key Distribution Center *(KDC)* pour l'attribution de tickets permettant l'accès aux différents services du réseau [31]. Son fonctionnement est présenté sur la FIGURE 2.1

Kerberos utilise la notion de « *ticket* » pour éviter à un utilisateur de devoir s'authentifier constamment aux différents serveurs auxquels il se connecte. L'utilisateur s'authentifie sur le KDC puis utilise un ticket pour s'authentifier sur chaque service demandé [9]. Le protocole Kerberos sépare le rôle du KDC en deux services.

— AS *(Authentification Service)* : il s'agit du service sur lequel l'utilisateur s'authentifie. Il délivre un ticket en cas d'authentification réussie. Ce ticket est en fait une demande d'accès au TGS.

— TGS *(Ticket Granting Service)* : ce service fourni les tickets d'accès aux différents services du réseau. On les appelle TS *(Ticket Service)* [32].

Les étapes de fonctionnement de l'authentification dans Kerberos :

1. Le client envoie au serveur d'authentification le message 1 ≪*KRB AS REQ*≫ : où il précise son nom et demande un ticket qui va le présenter ensuite au TGS afin de contacter le destinataire.
2. Le serveur d'authentification lui répond avec le message 2 ≪*KRB AS REP* ≫ : le serveur d'authentification cherche le client dans sa base de données. S'il le trouve, il engendre une clé de session qui devra être utilisée entre le client et le TGS. Cette clé est chiffrée avec la clé secrète du client : c'est la première partie du message.
   Ensuite, il crée un ticket pour le client afin qu'il puisse s'authentifier auprès du TGS, ce ticket est chiffré avec la clé secrète du TGS. Le client ne pourra pas le déchiffrer mais pourra le présenter tel quel est à chaque requête au TGS. Dans ce cas particulier, le ticket est appelé TGT.
3. Ensuite, avec le message 3 ≪*KRB TGS REQ*≫ le client s'authentifie auprès de TGS et demande le ticket de service qu'il souhaite avoir accès. Pour cela, le client fourni au TGS d'une part le nom du serveur qu'il souhaite contacter, d'autre part le ticket TGT et un identificateur qui possède des informations crypté avec la clé de session cet identificateur est vérifiable à partir du ticket par le TGS.
4. Grâce à sa clé secrète, le TGS déchiffre le ticket, et récupère la clé de session et peut ainsi déchiffrer l'identificateur.
   Il compare le contenu de l'identificateur avec les informations contenues dans le ticket et si tout concorde*(le client est authentifiée)*, il peut engendrer une clé de session *(qui sera utilisée entre le client et le service souhaité à avoir accès)* qu'il chiffre avec la clé de session et un nouveau ticket que le client devra présenter au service. Ces deux derniers seront la réponse de TGS au client contenant dans le message 4 ≪*KRB TGS REP*≫. Après réception de ce message et déchiffrement, le client dispose donc en plus de la clé de session et de TGT *(qu'il conserve jusqu'à expiration du ticket pour dialoguer avec TGS)* déjà obtenue par le AS, d'une nouvelle clé de session et d'un nouveau ticket qu'il pourra utiliser avec service à accéder.
5. Le message 5 ≪*KRB AP REQ*≫ correspond à la demande de service souhaité par le client, il s'authentifier auprès de ce service de la même manière qu'avec le TGS*(message 3(KRB TGS REQ))*.

6. Et le message 6≪*KRB AP REP*≫ correspond à la réponse à la demande.
   De son coté, le service accédé s'authentifie en prouvant qu'il a pu déchiffrer le ticket reçu par le client et donc, il possède la clé de session. Pour cela, il faut qu'il renvoie une information vérifiable par le client et chiffrée avec cette clé [8].

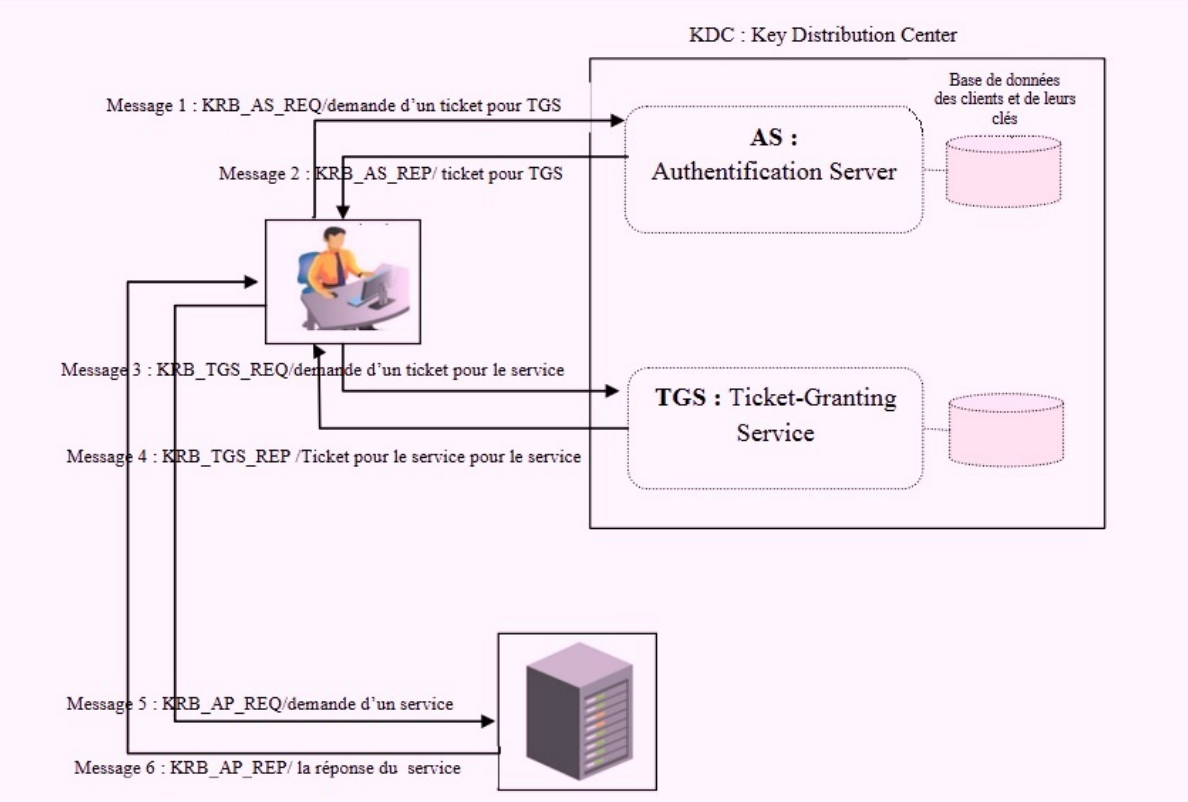


FIGURE 2.1 – L'authentification dans Kerberos[8]

Avantages de Kerberos

Kerberos est un protocole conçu pour être sûr même lorsqu'il est exécuté sur un réseau peu sûr.

— Kerberos garantit l'intégrité des données, leur confidentialité, la non répudiation et l'authentification mutuelle des clients services.

— Les transmissions sont chiffrées avec une clé secrète appropriée, l'attaquant ne peut pas avoir un ticket valide pour gagner l'accès non autorisé à un service sans compromettre une clé de cryptage.

— Diminue le nombre de bugs d'implémentation [8].

Inconvénients de Kerberos

Il n'y a pas de système parfait et il s'agit d'être bien conscient des limitations de ce système. Les grandes lignes des faiblesses du système Kerberos sont :

— Kerberos chiffre uniquement la phase d'authentification, il ne chiffre pas les données qui seront transmises lors de la session.

— Tous les services du réseau doivent être « *Kerberisé* », c'est-à-dire compatible avec le protocole Kerberos. Les services doivent être capables de comprendre le système de ticket, sinon aucune authentification ne sera possible.

— Si l'AS de Kerberos est compromis, un attaquant pourra accéder à tous les services avec un unique login [8].

2.5.2 RADIUS

Le protocole RADIUS *(Remote Authentication Dial-In User Service)*, mis au point initialement par Livingston Entreprise, est un protocole d'authentification distante de type AAA très fréquemment utilisé, défini par un certain nombre de RFC [33].

Le fonctionnement de RADIUS est basé sur un système client/serveur chargé de définir les accès d'utilisateurs distants à un réseau. Il s'agit du protocole de prédilection des fournisseurs d'accès à internet car il est relativement standard et propose des fonctionnalités de comptabilité permettant aux FAI *(Fournisseur d'Accès Internet)* de facturer précisément leurs clients.
Il permet de centraliser les données d'authentification : les politiques d'autorisation, les droits d'accès et la traçabilité. Ce processus doit être relié à une source d'informations.

Le protocole RADIUS repose principalement sur un serveur *(le serveur RADIUS)*, relié à une base d'identification et un client RADIUS, appelé NAS *(Network Access Server)*, faisant office d'intermédiaire entre l'utilisateur final et le serveur. L'ensemble des transactions entre le client RADIUS et le serveur RADIUS est chiffrée et authentifiée grâce à un secret partagé.

Les acteurs du RADIUS sont :

— L'utilisateur : émetteur de la requête d'authentification *(poste de travail, un portable...)*.

— Client RADIUS : le point d'accès au réseau *(NAS, firewall(pare-feu), point d'accès, etc...).*
— Serveur RADIUS : relié à une base d'authentification *(Base de données)* [33].

Son fonctionnement est simple :

1. Un utilisateur envoie une requête au NAS *(Network Acces Server)* afin d'autoriser une connexion à distance.
2. Le client RADIUS demande à l'utilisateur son nom et son mot de passe, et il les communique de manière sécurisée à un serveur RADIUS relié à une base de données.
3. En fonction de la zone d'accès demandée et des droits de l'utilisateur, le serveur RADIUS peut exiger des informations supplémentaires pour l'authentification. le serveur RADIUS retourne ainsi une des quatre réponses suivantes :
   — La réponse CHALLENGE permet d'éviter de transmettre le mot de passe. Le client envoie alors une autre requête répondant au Challenge pour s'authentifier.
   — Si l'identification réussie, le Serveur répond par un ACCEPT *(REJECT dans le cas contraire).*
4. Le serveur RADIUS envoie enfin les autorisations de l'utilisateur [9].

La figure suivante explique brièvement le fonctionnement du protocole RADIUS.

Avantages de RADIUS

— L'ensemble des transactions entre le client RADIUS et le serveur RADIUS est chiffré .
— Fonctionnement basé sur un système client/serveur chargé de définir les accès d'utilisateurs distants à un réseau [33].

Inconvénients de RADIUS

— Il base son identification sur le seul principe du couple *(nom, mot de passe).*
— Il n'assure pas des mécanismes d'identification du serveur [33].

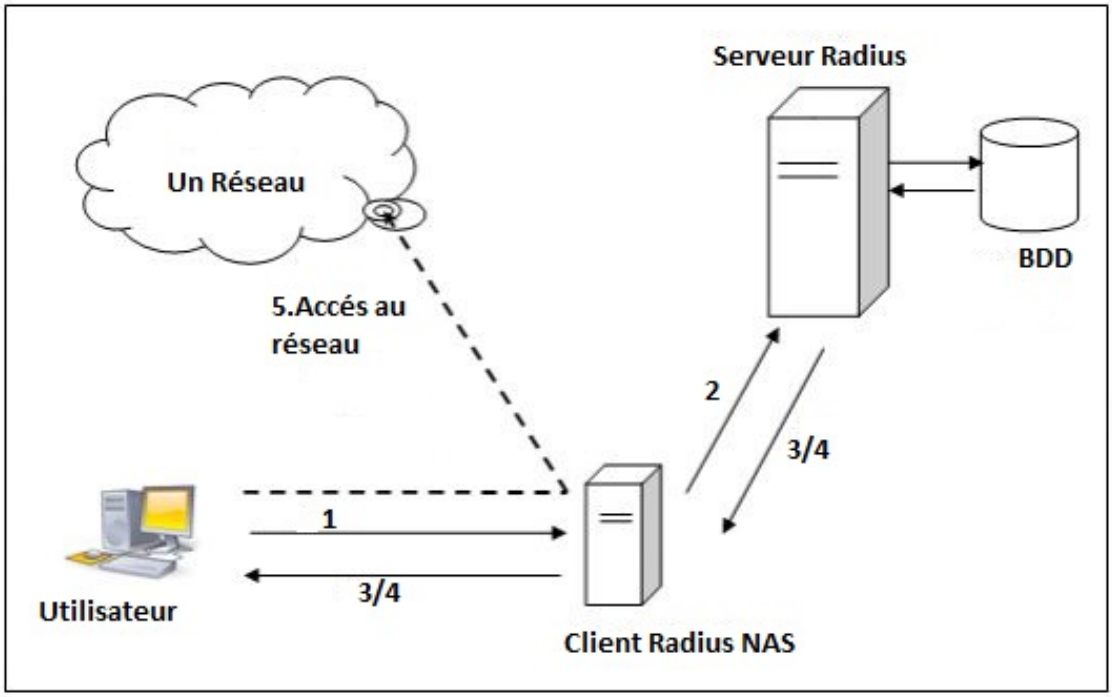


FIGURE 2.2 – Fonctionnement du Radius[9]

2.5.3 SSL

SSL *(Secure Socket Layer)* est un protocole utilisée pour sécuriser la transmission de données sur Internet : elle chiffre et protège les données transmises à l'aide du protocole HTTPS. Le SSL garantit aux visiteurs de votre site web que leurs données ne seront pas interceptées de manière frauduleuse [34].

Il a pour but de sécuriser les transactions Internet, par authentification du client (un navigateur la plupart du temps) et du serveur, et par chiffrement de la session. La sécurisation des connexions à l'aide du protocole SSL doit assurer que :

— La connexion assure la confidentialité des données transmises.

— La connexion assure que les données transmises sont intègres.

— L'identité des correspondants peut être authentifiée.

— La connexion est fiable [35].

Principe de fonctionnement de SSL :

SSL utilise les principes des chiffrements et d'authentification qui reposent sur les algorithmes à clé publique et les algorithmes à clé secrète. Il permet ainsi de créer un tunnel chiffré entre deux entités. Les étapes d'initialisation du tunnel, appelé SSL Handshake(protocole de négociation), sont les suivantes :

— Le client fait une requête SSL auprès du serveur.

— Le serveur présente alors son certificat afin de créer un contexte de confiance.

— Le client peut éventuellement lancer une requête OCSP *(Online Certificate Status Protocol)* pour vérifier la validité du certificat.

— Le serveur peut demander au client de lui présenter son certificat *(optionnel).*

— Les deux entités négocient, en fonction de leurs capacités cryptographique, les types et longueurs de clé qu'ils vont utiliser pour le chiffrement des échanges.

— Le client génère une clé secrète de session de façon aléatoire *(cette clé sera inférieure à 56 bits en SSL V2, inférieure à 128 bits en SSL V3).*

— Il chiffre cette clé secrète avec la clé publique du serveur et l'envoie à ce dernier.

— Le serveur déchiffre la clé secrète *(clé de session)* avec sa clé privée.

— En fin, les deux entités dialoguent en utilisant un algorithme à clé secrète pendant toute la durée de la session SSL [9].

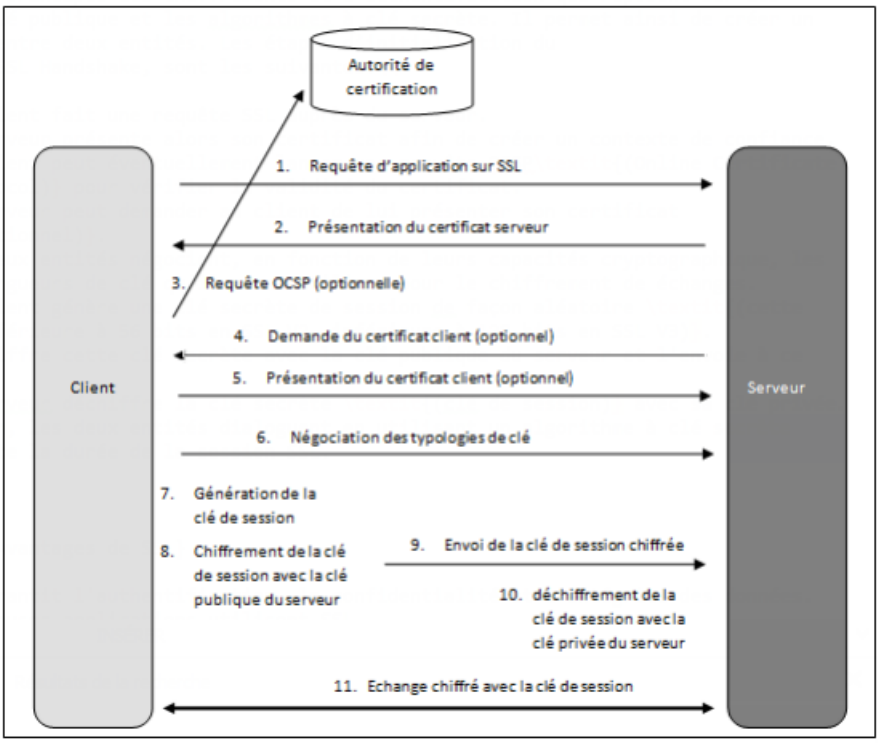


FIGURE 2.3 – Fonctionnement du SSL[9]

Ainsi, SSL est basé sur un échange de clé en utilisant la cryptographie à clé publique, puis les échanges de données sur une session à base de cryptographie à clé secrète[9].

Avantages de SSL

— Il garantit l'authentification, la confidentialité et l'intégrité des données[36].

— Nombreuses applications utilisent SSL [37].

— Authentification forte du client [37].

Inconvénients de SSL

— SSL est faible dans le sens où il n'impose pas l'authentification client *(ce serait d'ailleurs difficilement gérable en pratique)* [36].

— SSL est faible enfin car il présente des souplesses dans son implémentation, notamment en ce qui concerne la vérification des certificats des serveurs [36].

2.6 Conclusion

L'authentification fait partie de la vie quotidienne à l'ère numérique. Bien que vos informations personnelles restent confidentielles, elles ne sont pas infaillibles. *(Par exemple, si quelqu'un connaît votre adresse e-mail, il pourra accéder à votre compte en devinant simplement votre mot de passe).* A cette fin, ils ont utilisé les protocoles d'authentification qu'on a vu précédemment au cour de ce chapitre .

Dans le chapitre suivant, nous allons mettre en place un système d'authentification pour hadoop.

Chapitre 3

Mise en place d'un système d'authentification sur un cluster Hadoop

3.1 Introduction

Depuis quelques années, l'usage de l'informatique a permis une production en très grande quantité de données à travers plusieurs secteurs différents. Il existe quelques technologies permettant de traiter et manipuler ces énormes volumes de données. Hadoop est l'une des technologies qui stocke les données qui peuvent inclure explicitement des informations sensibles (financières, personnelles et commerciales) dans un cluster.

L'authentification et l'établissement de l'identité d'un utilisateur est une base afin de sécuriser l'accès à Hadoop. En particulier, pour une forte authentification et une propagation d'identités pour les utilisateurs et les services, nous allons utiliser le protocole Kerberos pour authentifier les différents utilisateurs qui sont autorisés à accéder à un cluster Hadoop.

Dans ce chapitre, nous allons présenter l'architecture de notre système, son principe de fonctionnement et les différentes étapes nécessaires pour son installation et sa configuration.

3.2 Architecture du système

L'architecture de notre système est présentée dans la FIGURE 3.1.

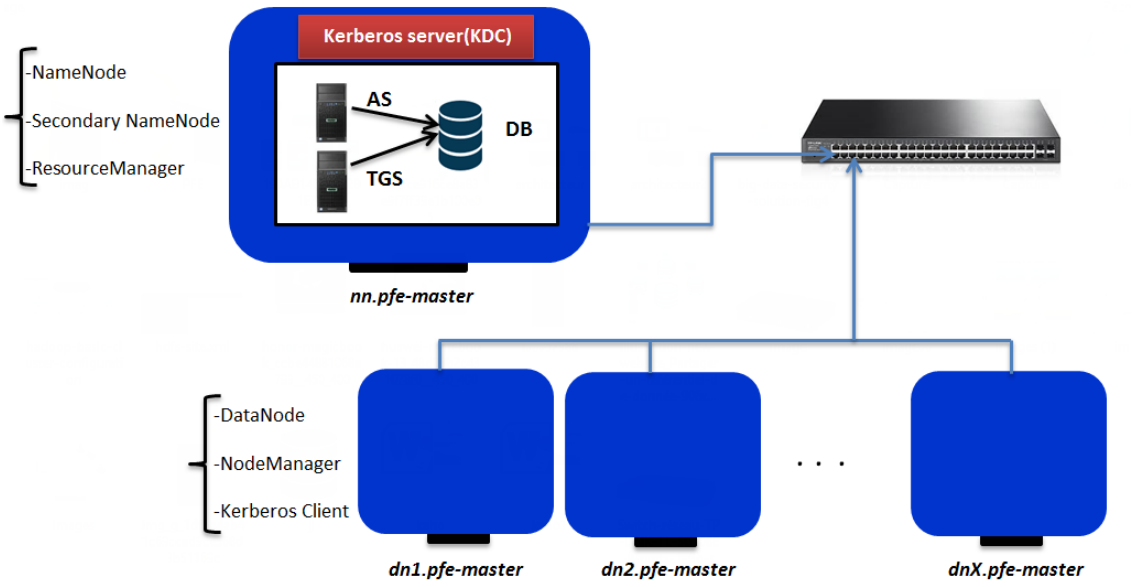


FIGURE 3.1 – Architecture du Kerberos dans Hadoop

Sur la machine nommée **nn.pfe-master** nous allons installer Hadoop en mode Master (Maître). Cette machine va contenir les composants suivants : NameNode, Secondary NameNode, ResourceManager.

— NameNode : c'est un serveur maître qui contient des métadonnées *(informations sur les données stockées dans les différents nœuds de données)* [38].

— Secondary NameNode : ce NameNode secondaire est important dans le processus de récupération du système en cas de plantage du NameNode principal.[39]

— ResourceManager : c'est Le nœud maître central qui assure la gestion et la planification des ressources *(clusters)* Hadoop et décide de ce qui doit se passer dans chaque nœud de données, il gère toutes les demandes de traitement [38].

Sur la même machine nous allons aussi installer le serveur Kerberos KDC *(Key Distribution Center)*, qui est constitué de deux services, AS *(Authentication Service)* et TGS *(Ticket Granting Service)* qui seront reliés à une base de données*(DB)*.

Cette architecture dispose de X DataNodes, chacun prend un nom spécifique *(dni.pfe-master)*, où $i = 1, \ldots, X$.

Nous allons aussi installer sur chaque DataNode Hadoop en mode Slave (esclave). Cette machine va contenir les composants suivants :

— DataNode : ce nœud va simplement gérer le stockage des données de leur système.

Il permet de répondre aux demandes de lecture et d'écriture des clients du système de fichiers et permet aussi d'effectuer des opérations telles que supprimer, créer des blocs par exemple[40].

— NodeManager : c'est l'agent d'infrastructure par ordinateur responsable des conteneurs, de la surveillance de l'utilisation des ressources *(unité centrale, mémoire, disque, réseau)* et de la génération de rapports à l'aide de ResourceManager[41].

Sur chaque DataNodes nous allons installer Kerberos Client qui est une entité pouvant obtenir un ticket auprès du KDC pour utiliser les différents services et ressources d'un réseau.

Remarque : les machines de notre cluster se trouvent sur le même réseau.

3.3 Fonctionnement du système

Le fonctionnement de notre architecture précédemment expliquée est montré sur la FIGURE 3.2.

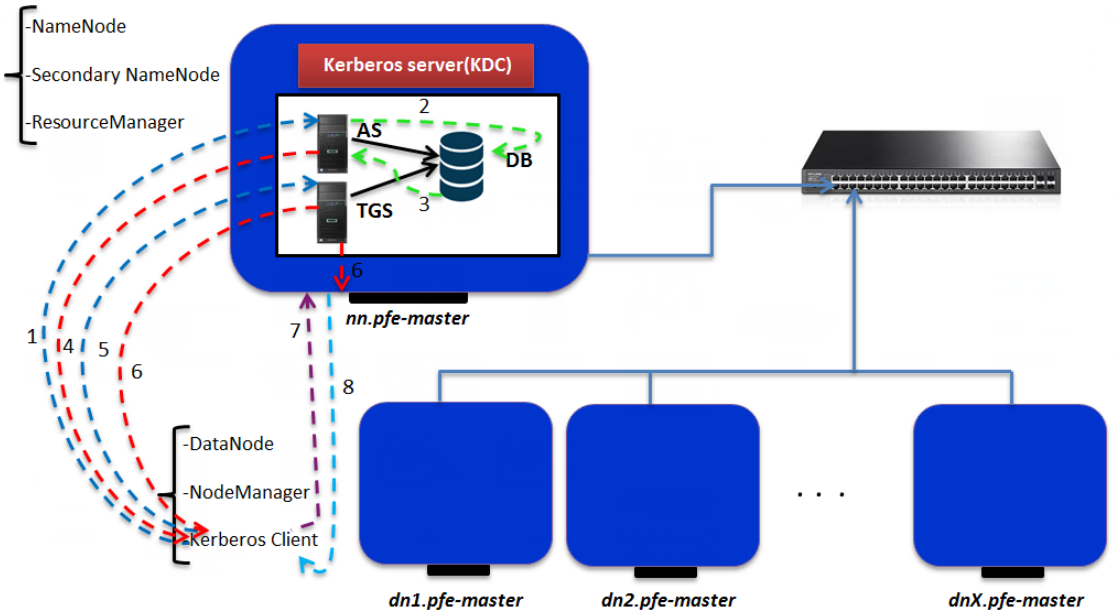


FIGURE 3.2 – Fonctionnement de Kerberos dans Hadoop

Les étapes de fonctionnement :

1. Pour que Kerberos Client puisse accéder au service AS, il utilise la commande **Kinit**.

2. AS cherche si le client *(l'utilisateur)* existe dans la DB.
3. DB peut répondre avec deux réponses *(l'éxistance ou non du l'utilisateur).*
4. En cas d'une authentification réussie, le service AS répond avec un TGT *(qui contient les information d'identification de l'utilisateur).*
5. Kerberos Client utilise son TGT afin de contacter TGS, pour demander un ticket de service *(TS)* pour accéder à Hadoop master *(NameNode).*
6. TGS accorde le ticket de service à Kerberos Client avec son TGT et un TS seul à Hadoop master*(NameNode).*
7. Kerberos Client envoie sa demande *(TGT)* à Hadoop master(NameNode).
8. Hadoop master envoie sa réponse à Kerberos Client.

Les étapes (de 1 à 4) désigne l'authentification. Et les étapes (de 5 à 6) désigne l'autorisation.

3.4 Installation et Configuration

Il existe trois modes d'installation de Hadoop, dans notre cas, nous allons installer Hadoop en mode totalement distribué sur plusieurs machines. Sur une machine on installe Hadoop en mode master (maître) et sur les autres machines on installe Hadoop en mode slave (esclave).

3.4.1 L'environnement de travail

Afin de réaliser notre travail nous allons utiliser trois machines qui seront reliées sur le même réseau :

Machines	Nom de la machine	Type de nœud	Système d'exploitation
Machine 01	nn.pfe-master	Hadoop master	Ubuntu 19.04
Machine 02	dn1.pfe-master	Hadoop slave	Ubuntu 18.04
Machine 03	dn2.pfe-master	Hadoop slave	Ubuntu 18.04

TABLE 3.1 – Les machines utilisées.

Version des outils logiciels utilisés

— Hadoop version 3.2.1

— Java 8

— Kerberos 5

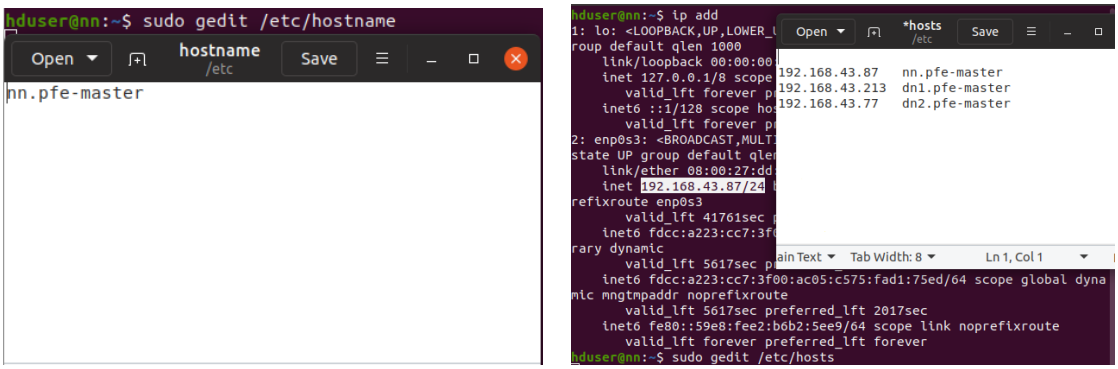
3.4.2 Configuration du réseau

Avant d'entamer l'installation et la configuration de notre système nous devons tout d'abord configurer notre réseau. Pour cela, nous allons modifier les nom des différents noeuds (machines) de notre cluster pour leur donner des noms significatifs. Nous allons modifier le fichier **/etc/hostname** sur chaque machine et mettre le nom de la machine correspondant suivant le Tableau 3.1.

Ensuite, pour faciliter notre travail et réduire le risque d'erreur par la suite dans la configuration de Hadoop et de Kerberos, nous allons utiliser les noms des différents noeuds dans les différents fichiers de configuration qui sont plus facile à retenir et à utiliser que les adresses IP, et pour cela nous devons faire la correspondance de chaque nom de machine avec son adresse IP, et pour cela nous allons modifier le fichier **/etc/hosts** sur chaque noeud du cluster en suivant le Tableau 3.2 (Voir Figure 3.3b).

Nom de la machine	**Adresse IP**
nn.pfe-master	192.168.43.87
dn1.pfe-master	192.168.43.213
dn2.pfe-master	192.168.43.77

Table 3.2 – Les adresses IP de chaque machine.



(a) Le nom d'hôte .

(b) Le fichier hosts .

3.4.3 Installation et configuration de Hadoop

Pré-installation de Hadoop

a) Nous devons tout d'abord installer Java 8 pour pouvoir exécuter Hadoop (Voir Figure ci-dessous pour la commande d'installation).

```
soumia@nn:~$ sudo apt install openjdk-8-jdk-headless
```

```
soumia@nn:~$ java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-1ubuntu1~18.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

b) Ensuite, nous devons télécharger Hadoop [1] et décompresser Hadoop et déplacer le contenu de Hadoop dans le répertoire **/usr/local/hadoop**.

```
soumia@nn:~/Desktop$ tar -xzvf hadoop-3.2.1.tar.gz
```

Création du répertoire Hadoop dans **/usr/local**.

```
soumia@nn:~$ sudo mkdir /usr/local/hadoop
```

Puis nous déplaçons le contenu de ce dossier Hadoop que nous avons téléchargé vers le répertoire **/usr/local/hadoop**.

```
soumia@nn:~/Desktop$ sudo mv hadoop-3.2.1/* /usr/local/hadoop
```

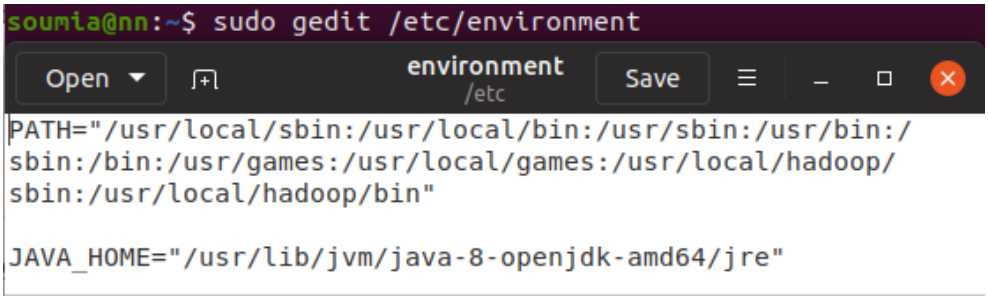
c) Nous installons **ssh** pour la communication entre les différents noeuds du cluster.

```
soumia@nn:~$ sudo apt install ssh
```

1. https ://hadoop.apache.org/releases.html/

ssh : c'est un protocole de réseau cryptographique permettant d'exploiter des services réseau en toute sécurité sur un réseau non sécurisé.

d) Ensuite, nous modifions le fichier **/etc/environment** et nous mettons à jour la ligne PATH pour inclure les répertoires binaires Hadoop et ajouter également la variable JAVA_HOME sur tous les nœuds du cluster.



g) Maintenant, nous devons créer un nouvel utilisateur qu'on nommera **hduser** et nous lui donnons les permissions nécessaires sur le répertoire **/usr/local/hadoop**.

1. Ajout de l'utilisateur **hduser** :

```
soumia@nn:~$ sudo adduser hduser
```

adduser ça veut dire ajoute un utilisateur, ou un groupe, au système.

2. Affectation des droits d'accès et des permission pour l'utilisateur **hduser** :

```
soumia@nn:~$ sudo usermod -aG hduser hduser
```

— **usermod** : modifie les paramètres d'un compte utilisateur.

— **-aG** : ajoute l'utilisateur **hduser** au groupe **hduser** sans supprimer **hduser** de ses groupes d'origine.

```
soumia@nn:~$ sudo chown hduser:root -R /usr/local/hadoop
```

— **chown** : changer le propriétaire et le groupe propriétaire d'un fichier ou d'un répertoire.

— **-R** : modifie récursivement un répertoire et tout ce qu'il contient

```
soumia@nn:~$ sudo chmod g+rwx -R /usr/local/hadoop
```

— **chmod** : modifie les permissions d'accès à un fichier ou à un répertoire.

— **g** : groupe propriétaire du fichier

— + : ajouter une permission.

— **r** : lecture.

— **w** : écriture.

— **x** : exécution.

```
soumia@nn:~$ sudo adduser hduser sudo
Adding user `hduser' to group `sudo' ...
Adding user hduser to group sudo
Done.
```

— **sudo** : permet d'exécuter des commandes en tant qu'un autre utilisateur, avec d'autres privilèges que les siens.

Génération des clés SSH

Nous devons générer la clé SSH pour l'utilisateur hduser sur chaque nœud du cluster et copier chaque clé sur les autres nœuds.

a) Génération de la clé ssh sur la machine **nn.pfe-master**.

```
hduser@nn:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9ApJvEPUgaRMoOsOY4rlanNce7JYJZLRG4LTW+JqBoQ hduser@nn.pfe-master
The key's randomart image is:
+---[RSA 2048]----+
|  ....oo..       |
|..oo.+. .        |
|E+ =o++ .        |
|..o Boo+ .       |
|o  = o=.S .      |
|o . ..oo .       |
|o*o ... .        |
|O* ooo .         |
|=o+. .+          |
+----[SHA256]-----+
```

```
hduser@nn:~$ ssh-copy-id hduser@nn.pfe-master
hduser@nn:~$ ssh-copy-id hduser@dn1.pfe-master
hduser@nn:~$ ssh-copy-id hduser@dn2.pfe-master
```

On refait la même opération sur les autres nœuds (**dn1.pfe-master**, **dn2.pfe-master**).

c) Nous définissons les variables d'environnement Hadoop pour l'utilisateur **hduser** en ajoutant les commandes suivantes à la fin du fichier ∼/**.bashrc**.

```
hduser@nn:~$ sudo gedit ~/.bashrc
```

Open .bas... ~/ Save

```
export HADOOP_HOME="/usr/local/hadoop"
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
```

Configuration de Hadoop

a) Ce fichier **core-site.xml** nous permet d'indiquer le noeud et le port du système de fichier HDFS [42]. Dans notre cas le noeud est la machine **nn.pfe-master** et le port est 9000.

```
hduser@nn:/usr/local/hadoop/etc/hadoop$ sudo gedit core-site.xml
```

Open *core-site.xml /usr/local/hadoop/etc/hadoop Save

```
<configuration>

<property>
<name>fs.default.name</name>
<value>hdfs://nn.pfe-master:9000</value>
</property>

</configuration>
```

b) Le fichier **hdfs-site.xml** permet de configurer où le NameNode va stocker l'historique des transactions et où les DataNode vont stocker leurs blocks. C'est également ici où le coefficient de réplication est configuré [42]. Dans notre cas le coefficient de replication est égale à 2.

```
hduser@nn:/usr/local/hadoop/etc/hadoop$ sudo gedit hdfs-site.xml
```

*hdfs-site.xml — /usr/local/hadoop/etc/hadoop

```
<configuration>

<property>
<name>dfs.namenode.name.dir</name>
<value>/usr/local/hadoop/data/nameNode</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>/usr/local/hadoop/data/dataNode</value>
</property>

<property>
<name>dfs.replication</name>
<value>2</value>
</property>

</configuration>
```

c) Le fichier **yarn-site.xml** : ce fichier contient des informations de configuration qui remplacent les valeurs par défaut des paramètres YARN. Les remplacements des valeurs par défaut pour les propriétés de configuration principales sont stockés dans le fichier Paramètres YARN par défaut [43].

```
hduser@nn:/usr/local/hadoop/etc/hadoop$ sudo gedit yarn-site.xml
```

*yarn-site.xml — /usr/local/hadoop/etc/hadoop

```
<configuration>

<property>
<name>yarn.resourcemanager.hostname</name>
<value>nn.pfe-master</value>
</property>

</configuration>
```

d) Pour spécifier les noeuds esclaves de notre cluster nous devons modifier le fichier **/usr/local/hadoop/etc/hadoop/workers** sur le nœud Hadoop Master (**nn.pfe-master**) et ajoutez le nom des noeuds esclaves (**dn1.pfe-master** et **dn2.pfe-master** pour notre cas).

```
hduser@nn:/usr/local/hadoop/etc/hadoop$ sudo gedit workers
```

workers — /usr/local/hadoop/etc/ha...

```
dn1.pfe-master
dn2.pfe-master
```

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

Remarque : on peut configurer Hadoop sur toutes les machines ou bien copier la configuration depuis le nœud maître vers les noeuds esclaves en utilisant la communication SSH.

```
hduser@nn:~$ scp /usr/local/hadoop/etc/hadoop/* dn1.pfe-master:/usr/local/hadoop/etc/hadoop/
capacity-scheduler.xml                                        100% 8260   927.3KB/s   00:00
configuration.xsl                                             100% 1335   219.0KB/s   00:00
container-executor.cfg                                        100% 1940   292.8KB/s   00:00
core-site.xml                                                 100%  868   131.5KB/s   00:00
hadoop-env.cmd                                                100% 3999   247.7KB/s   00:00
hadoop-env.sh                                                 100%   16KB   1.1MB/s   00:00
hadoop-metrics2.properties                                    100% 3323   541.1KB/s   00:00
hadoop-policy.xml                                             100%   11KB   1.0MB/s   00:00
hadoop-user-functions.sh.example                              100% 3414   573.8KB/s   00:00
hdfs-site.xml                                                 100% 1057   176.3KB/s   00:00
httpfs-env.sh                                                 100% 1484   132.2KB/s   00:00
httpfs-log4j.properties                                       100% 1657   249.1KB/s   00:00
httpfs-signature.secret                                       100%   21     3.4KB/s   00:00
httpfs-site.xml                                               100%  620    87.9KB/s   00:00
kms-acls.xml                                                  100% 3518   524.1KB/s   00:00
kms-env.sh                                                    100% 1351   215.7KB/s   00:00
kms-log4j.properties                                          100% 1747   296.5KB/s   00:00
kms-site.xml                                                  100%  682    99.7KB/s   00:00
log4j.properties                                              100%   13KB   1.1MB/s   00:00
mapred-env.cmd                                                100%  951    98.4KB/s   00:00
mapred-env.sh                                                 100% 1764   283.8KB/s   00:00
mapred-queues.xml.template                                    100% 4113   621.2KB/s   00:00
mapred-site.xml                                               100%  758   123.5KB/s   00:00
/usr/local/hadoop/etc/hadoop/shellprofile.d: not a regular file
slaves                                                        100%   28     4.3KB/s   00:00
ssl-client.xml.example                                        100% 2316   375.0KB/s   00:00
ssl-server.xml.example                                        100% 2697   388.2KB/s   00:00
user_ec_policies.xml.template                                 100% 2642   299.0KB/s   00:00
workers                                                       100%   28     4.9KB/s   00:00
yarn-env.cmd                                                  100% 2250   352.4KB/s   00:00
yarn-env.sh                                                   100% 6056   655.4KB/s   00:00
yarnservice-log4j.properties                                  100% 2591   401.1KB/s   00:00
yarn-site.xml                                                 100%  786   229.8KB/s   00:00
```

e) On termine l'installation de la configuration de Hadoop par formater le système de fichiers HDFS (formater le NameNode).

```
hduser@nn:~$ hdfs namenode -format
2019-11-23 15:58:30,533 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = nn.pfe-master/192.168.43.87
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.1
```

(a) la commande de formatage.

```
/usr/local/hadoop/data/nameNode has been successfully formatted.
ving image file /usr/local/hadoop/data/nameNode/current/fsimage.

age file /usr/local/hadoop/data/nameNode/current/fsimage.ckpt_00

: Going to retain 1 images with txid >= 0
```

(b) le formatage est reussi.

Lancement de Hadoop

b) Nous pouvons démarrer Hadoop sur la machine(nn.pfe-master) par :

— **start-dfs.sh** : permet de lancer le NameNode, les DataNodes et le Secondary NameNode.

```
hduser@nn:~$ start-dfs.sh
Starting namenodes on [nn.pfe-master]
Starting datanodes
Starting secondary namenodes [nn.pfe-master]
```

— **start-yarn.sh** : permet de lancer le ResourceManager et le NodeManager.

```
hduser@nn:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

— Pour vérifier si les processus Hadoop tels que NameNode, DataNode, ResourceManager, NodeManager et secondary NameNode sont actifs et en cours d'exécution on utilise la commande **jps** (*Java Virtual Machine Process Status Tool*) :

```
hduser@nn:~$ jps
2661 ResourceManager
2421 SecondaryNameNode
2233 NameNode
2971 Jps
hduser@nn:~$
```

c) Et sur les machines esclaves on tape seulement la commande *"jps"*. Les services des noeuds esclaves sont lancés automatiquement à distance par le noeud maître.

```
hduser@dn1:~$ jps
4692 DataNode
4868 NodeManager
5003 Jps
hduser@dn1:~$
```

(a) dn1.pfe-master .

```
hduser@dn2:~$ jps
4305 Jps
4180 NodeManager
3999 DataNode
hduser@dn2:~$
```

(b) dn2.pfe-master .

Les interfaces utilisateurs d'administrations d'Hadoop

La distribution Hadoop par Apache fournit des interfaces utilisateurs pour l'administration. Ceux-ci sont accessibles via des applications Web. La première concerne l'état du cluster et est accessible via l'adresse http ://nn.pfe-master :8088. Il sera possible d'avoir une vue globale sur les nœuds du cluster et sur les jobs en cours d'exécution, la figure donnée ci-dessous :

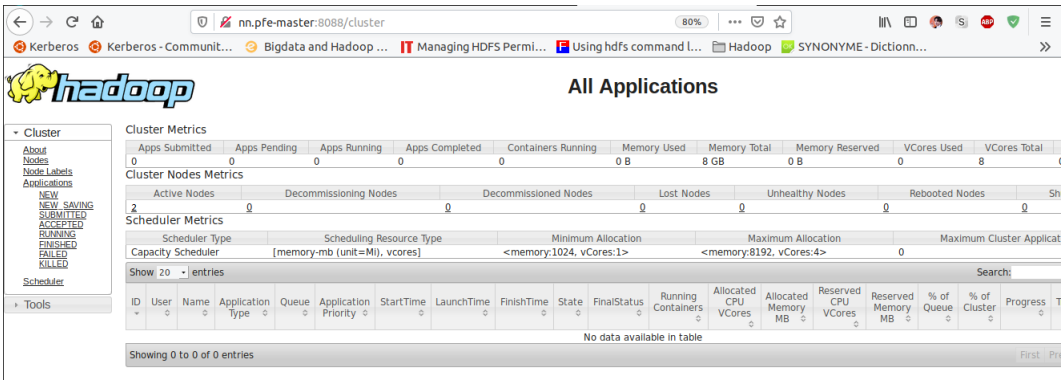


FIGURE 3.3 – Interface utilisateurs, d'accueil.

La deuxième interface utilisateur concerne l'accès aux données contenues dans le nœud NameNode et est accessible via l'adresse http ://nn.pfe-master :9870. Elle permet d'obtenir des informations sur la capacité totale et connaître l'état de disponibilité des nœuds. Elle permet également d'avoir des informations sur les fichiers et de naviguer dans le HDFS du cluster.

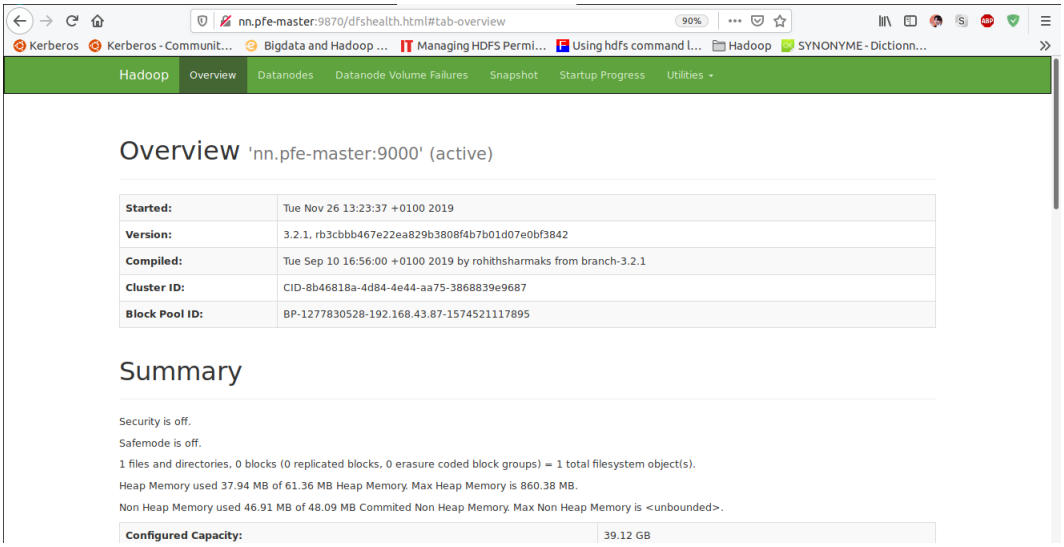


FIGURE 3.4 – Interface utilisateurs, du NameNode.

3.4.4 Installation kerberos

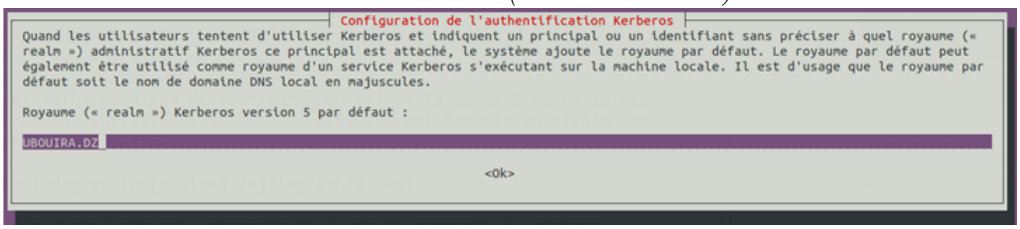
Kerberos client

Il est moins complexe de configurer un client Kerberos 5 qu'un serveur. Nous devons installer les packages clients sur tous les nœuds de notre cluster.

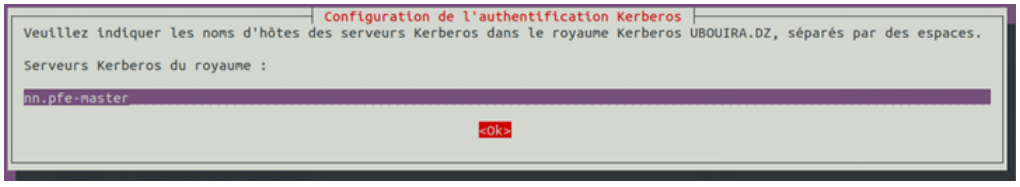
```
hduser@dn2:~$ sudo apt-get install krb5-user libpam-krb5 libpam-ccreds
auth-client-config
```

Les capteurs ci-dessus présentent des questions lors de l'installation, elles sont utilisées pour configurer le fichier " */etc/krb5.conf*".

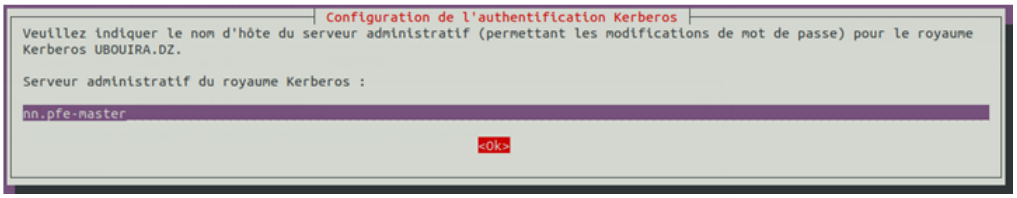
— Le nom du royaume *(realm)* Kerberos par défaut, le royaume doit être en majuscule, dans notre cas on a choisi comme realm *(UBOUIRA.DZ)*.



— Le nom du serveur Kerberos (*la machine sur laquelle est installé Kerberos server (AS, TGS)*). Dans notre cas, on va installer le serveur Kerberos sur la machine **nn.pfe-master**, la même machine sur laquelle on a installé Hadoop en mode master.



— Le nom du serveur d'administration.



Kerberos serveur

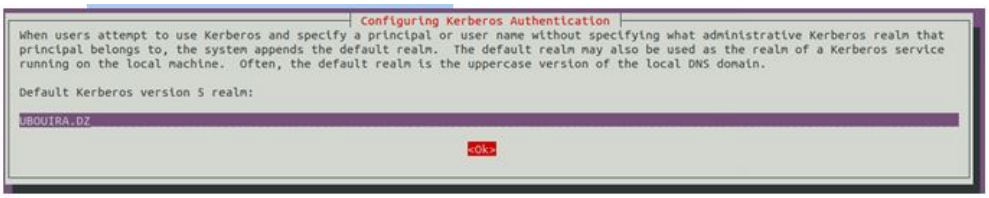
Avant de commencer l'installation de MIT Kerberos , nous devant synchroniser l'heure, car si les horloges du serveur et du client diffèrent de plus de cinq minutes, les clients ne pourront pas s'authentifier sur le serveur.

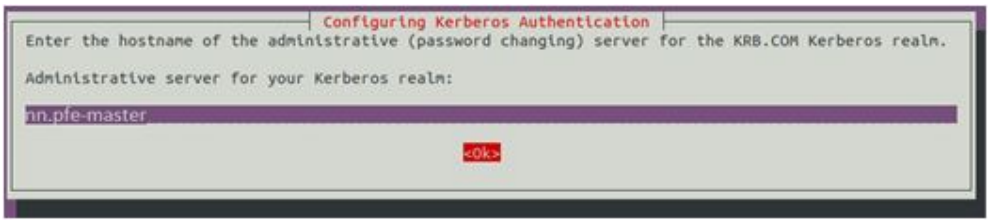
1. Pour installer MIT Kerberos nous allons exécuter la commande suivante :

```
hduser@nn:~$ sudo apt-get krb5-kdc krb5-admin-server
```

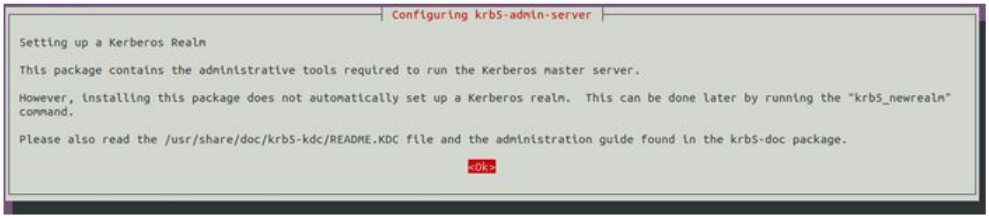
2. L'installation du serveur Kerberos se fait suivant les étapes *(similaires à celles*

d'installation de Kerberos client) ci après :





Mise en place d'un royaume de kerberos : ce paquet contient les outils d'administration nécessaires à l'exécution du serveur maître kerberos.



3. Ensuite, nous allons initialiser le royaume kerberos *(realm)* à l'aide de l'utilitaire kdb5_newrealm :

```
hduser@nn:~$ sudo krb5_newrealm
```

royaume : désigne un domaine administratif d'authentification kerberos [44].

Ce script va être exécuté sur le serveur KDC/admin-sever pour initialiser un royaume kerberos. Il nous demandera de saisir un mot de passe de clé principale, et initialise également la base de données pour le royaume « *UBOUIRA.DZ* ».

Configuration de kerberos serveur :

1. On ouvre le fichier**/etc/krb5kdc/kdc.conf** pour ajouter quelque type de cryptage.

Un type de chiffrement kerberos est une combinaison spécifique d'un algorithme de chiffrement et d'un algorithme d'intégrité pour assurer la confidentialité et l'intégrité des données et améliore la sécurité globale du service kerberos.

Dans la section suivante [realms] du fichier kdc.conf on trouve :

— **acl_file** : emplacement du fichier de liste de contrôle d'accès.

— **admin_keytab** : emplacement du fichier kadm5.keytab.

— **database_name** : cette relation spécifie l'emplacement de la base de données Kerberos pour ce domaine.

— **key_stash_file** : spécifie l'emplacement où la clé principale a été stockée *(via la stash kdb5_util).*

— **kdc_ports** : avant la version 1.15, cette relation répertorie les ports du démon krb5kdc à l'écoute des demandes UDP.

— **master_key_type** : spécifie le type de clé de la clé principale.

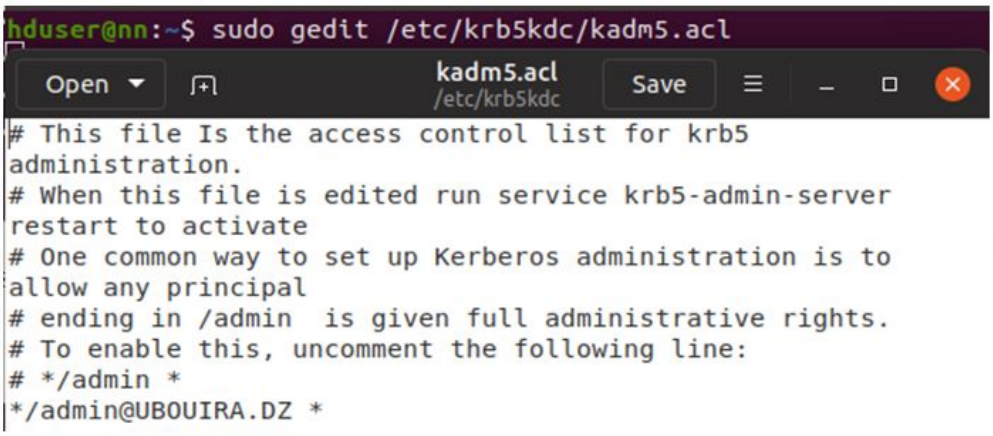
— **max_life** : spécifie la période maximale pendant laquelle un ticket peut être valide dans ce domaine.

— **max_renewable_life** : spécifie la période maximale pendant laquelle un ticket valide peut être renouvelé dans ce domaine.

— **supported_enctypes** : spécifie la clé par défaut des principaux pour ce realm.

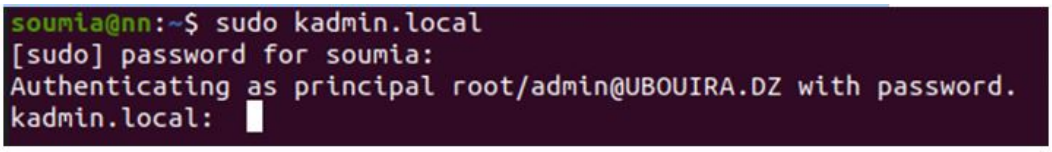
2. Kerberos utilise un fichier de liste de contrôle d'accès *(ACL)* pour gérer les droits d'accès à la base de données Kerberos.

L'emplacement par défaut du fichier Kerberos *(ACL)* c'est : " /etc/krb5kdc/kadm5.acl" [44].



Kerberos Principals :

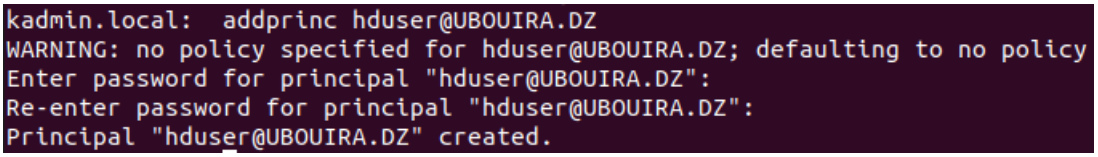
Pour accéder directement à la base de données KDC on tape la commande suivante :



Kadmin. Local est une interface de ligne de commande du système d'administration Kerberos V5.

1. Création des principals :

   Un principal *(ou nom principal)* est le nom unique d'un utilisateur ou d'un service autorisé à s'authentifier à l'aide de kerberos [44].

   addprinc : permet d'ajouter un nouveau principal, en demandant deux fois un mot de passe [44].

   Pour chaque instance du service Hadoop on doit créer un principal Kerberos correspondant à ce service.

2. Une fois que les nouveaux principaux sont ajoutés, on tape list_principals dans kadmin pour les afficher.

```
kadmin.local:  list_principals
K/M@UBOUIRA.DZ
hduser@UBOUIRA.DZ
kadmin/admin@UBOUIRA.DZ
kadmin/changepw@UBOUIRA.DZ
kadmin/nn.pfe-master@UBOUIRA.DZ
kiprop/nn.pfe-master@UBOUIRA.DZ
krbtgt/UBOUIRA.DZ@UBOUIRA.DZ
mapred@UBOUIRA.DZ
yarn@UBOUIRA.DZ
kadmin.local:
```

3. Maintenant que tout est configuré nous pouvons redémarre les services krb5-kdc et krb5-admin-server pour que les modifications seront prises en compte.

```
hduser@nn:~$ service krb5-kdc restart
hduser@nn:~$ service krb5-admin-server restart
hduser@nn:~$
```

4. Obtenir des Tickets :
   — Si on exécute la commande **klist** pour afficher les tickets on reçoit le message suivant :

```
hduser@nn:~$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1001)
hduser@nn:~$
```

   Cela veut dire que nous avons aucun utilisateur *(principal)* authentifié, il n'y a pas de ticket.
   — Pour obtenir des tickets on tape simplement kinit suivi du nom d'un principal, puis on tape le mot de passe. kinit obtient et échange un ticket initial d'attribution de ticket TGT pour le principal.

```
hduser@nn:~$ kinit hduser@UBOUIRA.DZ
Password for hduser@UBOUIRA.DZ:
hduser@nn:~$
```

5. Affichage des tickets : on utilise la commande **klist** pour afficher les information sur le ticket, lorsque on obtiens des tickets pour la première fois, nous n'aurons que le TGT.

```
hduser@nn:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1001
Default principal: hduser@UBOUIRA.DZ

Valid starting       Expires              Service principal
2019-11-20T16:54:33  2019-11-21T02:54:33  krbtgt/UBOUIRA.DZ@UBOUIRA.DZ
        renew until 2019-11-21T16:54:21
hduser@nn:~$
```

«**Ticket cache**» est l'emplacement du fichier de ticket [44]. Dans l'exemple ci-dessus, ce fichier s'appelle / tmp / krb5cc_1001. Le principal par défaut est notre principal Kerberos *'hduser@UBOUIRA.DZ'*.

Les champs «*Valid starting*» et «*Expires*» décrivent la période de validité du ticket. Le «*Service principal*» décrit chaque ticket. Le tgt a un premier composant krbtgt et un second composant qui est le nom de domaine [44].

6. Détruire un ticket (**kdestroy**) : cet utilitaire sert a détruire les tickets d'autorisation Kerberos actifs de l'utilisateur en supprimant le cache des informations d'identification qui les contient [44].

```
soumia@nn:~$ kinit hduser@UBOUIRA.DZ
Password for hduser@UBOUIRA.DZ:
soumia@nn:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: hduser@UBOUIRA.DZ

Valid starting       Expires              Service principal
2019-11-27T12:22:17  2019-11-27T22:22:17  krbtgt/UBOUIRA.DZ@UBOUIRA.DZ
        renew until 2019-11-28T12:21:40
soumia@nn:~$ kdestroy
soumia@nn:~$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1000)
soumia@nn:~$
```

Keytab :

Un Keytab *(abréviation de «key table»)* stocke les clés à long terme pour un ou plusieurs principaux. Les keytabs sont normalement représentés par des fichiers dans un format standard. Les keytabs sont le plus souvent utilisés pour permettre aux applications serveur d'accepter les authentifications des clients, mais ils peuvent également être utilisés pour obtenir les informations d'identification initiales des applications client [44].

1. Génération des Keytabs pour les principals : chacun des service hadoop doit avoir un fichier keytab, a titre d'exemple on va créer un fichier keytab pour le service yarn :

```
kadmin.local:  xst -norandkey -k yarn.keytab yarn@UBOUIRA.DZ
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type aes256-cts-h
mac-sha1-96 added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type aes128-cts-h
mac-sha1-96 added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type arcfour-hmac
 added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des3-cbc-sha
1 added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des-cbc-crc
added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des-cbc-md5
added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des-cbc-md5
added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des-cbc-md5
added to keytab WRFILE:yarn.keytab.
Entry for principal yarn@UBOUIRA.DZ with kvno 1, encryption type des-cbc-md5
added to keytab WRFILE:yarn.keytab.
kadmin.local:
```

-norandkey : Cette option est uniquement disponible dans kadmin.local. Les clés et leurs numéros de version restent inchangés et ne pas randomiser les clés.

-k : Utilisez un Keytab pour déchiffrer la réponse KDC au lieu de demander un mot de passe[44].

2. Affiche un Keytab : Un keytab peut être affiché à l'aide de la commande klist avec l'option -e -k -t . Nous pouvons créer ou ajouter des keytabs en extrayant les clés de la base de données KDC à l'aide de la commande kadmin.local. Les keytabs peuvent être manipulés à l'aide des commandes ktutil.

```
hduser@nn:~$ klist -e -k -t /etc/security/keytabs/hduser.keytab
Keytab name: FILE:/etc/security/keytabs/hduser.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------------
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (aes256-cts-hmac-sha1-96)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (aes128-cts-hmac-sha1-96)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (arcfour-hmac)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des3-cbc-sha1)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des-cbc-crc)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des-cbc-md5)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des-cbc-md5)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des-cbc-md5)
   1 2019-11-19T01:22:25 hduser@UBOUIRA.DZ (des-cbc-md5)
hduser@nn:~$
```

-e : Affiche les types de cryptage de la clé de session et du ticket pour chaque clé dans le fichier keytab.

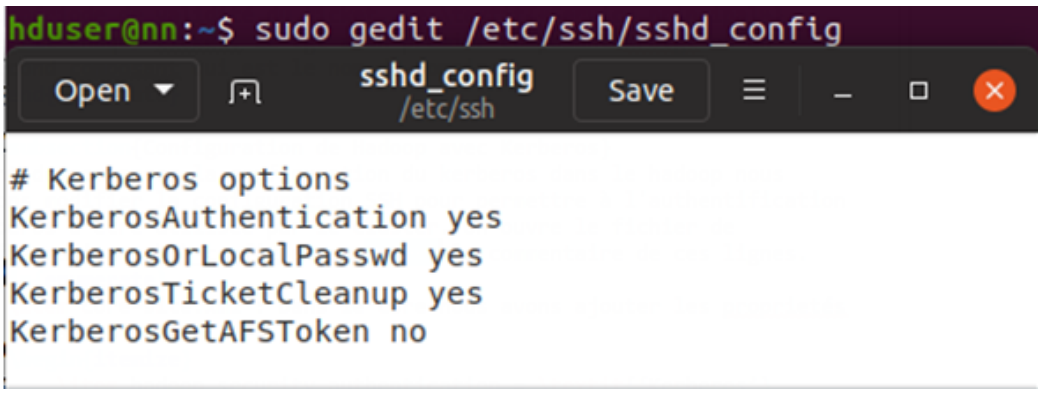
-k : Liste des clés contenues dans un fichier keytab.

-t : Affiche la saisie de l'heure pour chaque entrée de clé dans le fichier keytab [44].

3.4.5 Configuration de Hadoop avec Kerberos

Avant de commancer la configuration de Hadoop pour utiliser l'authentification Kerberos nous devons modifier la configuration SSH pour permettre à l'authentification Kerbe-

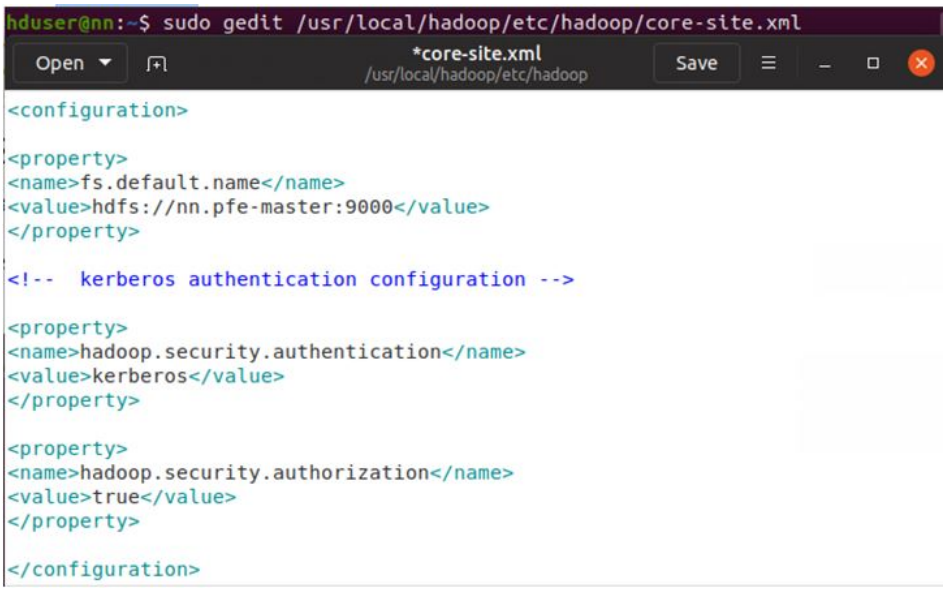
ros d'être utilisée par le cluster. On ouvre le fichier de configuration **/etc/ssh/sshd_config** et on enleve le commentaire sur les lignes suivantes :



Dans ce qui suit, nous expliquerons comment configurer l'authentification pour Hadoop en mode sécurisé, lorsque ceci est activé chaque service de Hadoop et chaque utilisateur doivent être authentifiés par Kerberos.
Maintenant nous allons éditer les fichiers de configuration Hadoop pour permettre au cluster Hadoop d'utiliser l'authentification Kerberos .

1. **core-site.xml** :

   Pour activer l'authentification RPC dans Hadoop, on défini la valeur de la propriété **hadoop.security.authentication** sur *"kerberos"*, puis on défini la valeur de **hadoop.security.authorization** sur *"true"*, cette dernière indique que nous avons activé l'autorisation au niveau de service RPC [45].



   RPC service-level authorization est le mécanisme d'autorisation initial permettant de garantir que les clients qui se connectent à un service Hadoop donné

disposent des autorisations nécessaires préconfigurées et sont autorisés à accéder au service donné [45].

2. Création du dossier keytabs : on va créer un dossier appelé **keytabs** dans le répertoire **/etc/security** pour contenir tous les fichiers Keytab que nous allons utiliser.

```
hduser@nn:~$ sudo mkdir /etc/security/keytabs
```

Tous les service Hadoop doivent être Kerberisé, c'est-à-dire chaque service dois s'authentifier avant de se lancer. Pour cela nous devons créer un principal pour chaque service, le nom de ce principal est sous la forme $< nom\_service > / < host > @REALM$. Ensuite nous générons le fichier keytab pour chaque services.

Le TABLEAU 3.3 ci-dessus montre les différentes services, leur principal et leur fichiers keytab.

Hadoop doit avoir un principal, on export la clé vers un fichier keytab comme illustré dans le tableau suivant :

Service	**Principal**	**Fichier Keytab**
NameNode	nn/nn.pfe-master@UBOUIRA.DZ	nn.service.keytab
DataNode	dn/nn.pfe-master@UBOUIRA.DZ	dn.service.keytab
Secondary NameNode	sn/nn.pfe-master@UBOUIRA.DZ	sn.service.keytab
ResourceManager	rm/nn.pfe-master@UBOUIRA.DZ	rm.service.keytab
NodeManager	nm/nn.pfe-master@UBOUIRA.DZ	nm.service.keytab
webhdfs	http/nn.pfe-master	http.service.keytab

TABLE 3.3 – Les principals des services hadoop.

```
kadmin.local:  addprinc dn/nn.pfe-master@UBOUIRA.DZ
WARNING: no policy specified for dn/nn.pfe-master@UBOUIRA.DZ; defaulting to no p
olicy
Enter password for principal "dn/nn.pfe-master@UBOUIRA.DZ":
Re-enter password for principal "dn/nn.pfe-master@UBOUIRA.DZ":
Principal "dn/nn.pfe-master@UBOUIRA.DZ" created.
```

```
kadmin.local:  xst -norandkey -k dn.service.keytab  dn/nn.pfe-master@UBOUIRA.DZ
Entry for principal dn/nn.pfe-master@UBOUIRA.DZ with kvno 2, encryption type ae
s256-cts-hmac-sha1-96 added to keytab WRFILE:dn.service.keytab.
Entry for principal dn/nn.pfe-master@UBOUIRA.DZ with kvno 2, encryption type ae
s128-cts-hmac-sha1-96 added to keytab WRFILE:dn.service.keytab.
Entry for principal dn/nn.pfe-master@UBOUIRA.DZ with kvno 2, encryption type ar
cfour-hmac added to keytab WRFILE:dn.service.keytab.
Entry for principal dn/nn.pfe-master@UBOUIRA.DZ with kvno 2, encryption type de
s3-cbc-sha1 added to keytab WRFILE:dn.service.keytab.
Entry for principal dn/nn.pfe-master@UBOUIRA.DZ with kvno 2, encryption type de
s-cbc-crc added to keytab WRFILE:dn.service.keytab.
kadmin.local:
```

3. **hdfs-site.xml** et **yarn-site.xml** :

   Les propriétés suivantes doivent figurer dans le fichier hdfs-site.xml de tous les nœuds du cluster.

   dfs.web.authentication.kerberos.principal pour cette propriété on indique le nom du Kerberos principal pour le WebHDFS.

   dfs.web.authentication.kerberos.keytab, là, on designe l'emplacement du fichier keytab pour WebHDFS.

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

Open — hdfs-site.xml — /usr/local/hadoop/etc/hadoop — Save

```
<property>
<name>dfs.web.authentication.kerberos.keytab</name>
<value>/etc/security/keytabs/http.service.keytab</value>
</property>

<property>
<name>dfs.web.authentication.kerberos.principal</name>
<value>http/nn.pfe-master@UBOUIRA.DZ</value>
</property>
```

   dfs.block.access.token.enable on met sa valeur a *true*, cela veut dire qu'on a activé les jetons d'accès aux blocks HDFS pour des opérations sécurisées.

   Chaque instance du service Hadoop doit être configurée avec son principal kerberos et l'emplacement de son fichier keytab.

   dfs.namenode.kerberos.principal a comme valeur le Nom du principal Kerberos pour le Namenode.

   dfs.namenode.keytab.file signifie qu'on doit révéler le chemin pour le fichier keytab du Namenode [45].

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

Open | hdfs-site.xml /usr/local/hadoop/etc/hadoop | Save

```
<!-- Kerberos configuration -->

<property>
<name>dfs.block.access.token.enable</name>
<value>true</value>
</property>

<property>
<name>dfs.namenode.keytab.file</name>
<value>/etc/security/keytabs/nn.service.keytab</value>
</property>

<property>
<name>dfs.namenode.kerberos.principal</name>
<value>nn/nn.pfe-master@UBOUIRA.DZ</value>
</property>

<property>
<name>dfs.secondary.namenode.keytab.file</name>
<value>/etc/security/keytabs/sn.service.keytab</value>
</property>

<property>
<name>dfs.secondary.namenode.kerberos.principal</name>
<value>sn/nn.pfe-master@UBOUIRA.DZ</value>
</property>
```

La même chose sera faite pour le Secondary Namenode et le DataNode dans le fichier **hdfs-site.xml** et de même dans le fichier **yarn-site.xml** avec le Resource manager et le Node manager.

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

Open | *yarn-site.xml /usr/local/hadoop/etc/hadoop | Save

```
<configuration>

<property>
<name>yarn.resourcemanager.hostname</name>
<value>nn.pfe-master</value>
</property>

<!-- Kerberos configuration -->
<property>
<name>yarn.nodemanager.keytab</name>
<value>/etc/security/keytabs/nm.service.keytab</value>
</property>

<property>
<name>yarn.nodemanager.principal</name>
<value>nm/nn.pfe-master@UBOUIRA.DZ</value>
</property>

<property>
<name>yarn.resourcemanager.keytab</name>
<value>/etc/security/keytabs/rm.service.keytab</value>
</property>

<property>
<name>yarn.resourcemanager.principal</name>
<value>rm/nn.pfe-master@UBOUIRA.DZ</value>
</property>
</configuration>
```

Secure DataNode :

Étant donné que le protocole de transfert de données DataNode n'est pas sécurisée

par la structure RPC Hadoop, DataNodes doit s'authentifier à l'aide du protocole SASL et du protocole SSL.

— **SASL** *(Simple Authentication and Security Layer)* est une bibliothèque Java. Hadoop prend en charge l'authentification kerberos via SASL.

SASL peut être utilisé pour authentifier le protocole de transfert de données. Pour activer SASL sur le protocole de transfert de données, tout d'abord on doit définir la propriété **dfs.data.transfer.protection** dans le fichier **hdfs-site.xml**.

Voici les étape a suivre pour que SASL active le mode sécurisé dans les DataNode :

— On défini un port non privilégié pour **dfs.datanode.address**.

— Puis on défini dfs.http.policy sur HTTPS_ONLY.

— **SSL***(Secure Sockets Layer)* est un protocole d'échanges sur Internet, le transfert de données entre la console Web et les clients est protégé à l'aide de SSL *(HTTPS)*. La configuration de SSL est recommandée mais pas obligatoire pour configurer la sécurité Hadoop avec Kerberos.

Afin d'activer SSL pour la console Web de HDFS on spécifie **dfs.http.policy** a HTTPS_ONLY ou HTTP_AND_HTTPS dans **yarn-site.xml** et **hdfs-site.xml** et aussi on indique l'adresse de l'interface Web HTTPS pour le Data Node et le Name Node [45].

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

Open | *hdfs-site.xml /usr/local/hadoop/etc/hadoop | Save

```
<!-- SASL et ssl coniguration -->

<property>
<name>dfs.datanode.address</name>
<value>0.0.0.0:1055</value>
</property>

<property>
<name>dfs.data.transfer.protection</name>
<value>privacy</value>
</property>

<property>
<name>dfs.http.policy</name>
<value>HTTPS_ONLY</value>
</property>

<property>
<name>dfs.namenode.https-address</name>
<value>0.0.0.0:9871</value>
</property>

<property>
<name>dfs.datanode.https.address</name>
<value>0.0.0.0:9865</value>
</property>

</configuration>
```

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

Open | *yarn-site.xml /usr/local/hadoop/etc/hadoop | Save

```
<property>
<name>yarn.http.policy</name>
<value>HTTPS_ONLY</value>
</property>

</configuration>
```

Puis, nous devant configurer le fichier **ssl-server.xml** pour l'utilisation d'un certificat SSL, ce se trouve dans le dossier de configuration de Hadoop.

Dans le fichier de configuration de SSL nous trouvons les propriété suivantes :

Propriété	**Explication**
ssl.server.keystore.type	Type de fichier keystore (jks)
ssl.server.keystore.location	l'emplacement du fichier keystore
ssl.server.keystore.password	Le mot de pass pour le fichier keystore
ssl.server.truststore.type	Type de fichier truststore (jks)
ssl.server.truststore.location	l'emplacement du fichier truststore
ssl.server.truststore.password	Le mot de pass pour le fichier truststore
ssl.server.truststore.reload.interval	Intervalle de rechargement de truststore , en millisecondes (10000)

TABLE 3.4 – Les propriétés de SSL-server [10].

```
hduser@nn:~$ sudo gedit /usr/local/hadoop/etc/hadoop/ssl-server.xml
```

Open | *ssl-server.xml /usr/local/hadoop/etc/hadoop | Save

```
<configuration>

<property>
  <name>ssl.server.truststore.location</name>
  <value>/usr/local/hadoop/etc/hadoop/cxfca.jks</value>
</property>

<property>
  <name>ssl.server.truststore.password</name>
  <value>password</value>
</property>

<property>
  <name>ssl.server.truststore.type</name>
  <value>jks</value>
</property>

<property>
  <name>ssl.server.truststore.reload.interval</name>
  <value>10000</value>
</property>
```

Keystore et **Truststore** sont à la fois importants et essentiels pour la communication avec un certificat SSL. Les deux sont très similaires en termes de construction et de structure, car ils sont tous les deux gérés par la commande key tool.

Truststore est utilisé pour le stockage de certificats de l'autorité *(CA)*, qui sert à

la vérification du certificat fourni par le serveur dans une connexion SSL. D'autre part, Keystore est utilisé pour stocker la clé privée et son propre certificat d'identité à identifier pour vérification [46].

3.4.6 Test de l'authentification Kerberos

Maintenant que nous avons un cluster Hadoop Kerberisé, voici un test de création d'un répertoire sur le HDFS, avec et sans authentification.

Sans authentification :

Si kerberos est activé correctement ce qui est notre cas, nous ne pouvons pas nous connecter aux système de fichiers Hadoop tant que nous ne sommes pas authentifié avec succès.

```
hduser@nn:~$ hdfs dfs -mkdir /test-kerberos
2019-11-27 21:56:01,809 WARN ipc.Client: Exception encountered while connecting
 to the server : org.apache.hadoop.security.AccessControlException: Client cann
ot authenticate via:[TOKEN, KERBEROS]
mkdir: DestHost:destPort nn.pfe-master:9000 , LocalHost:localPort nn.pfe-master
/192.168.43.87:0. Failed on local exception: java.io.IOException: org.apache.ha
doop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KE
RBEROS]
hduser@nn:~$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1001)
```

Avec authentification :

Avant d'essayé de créer notre répertoire nous devons s'authentifier auprès du serveur Kerberos en utilisant la commande **kinit** pour obtenir un ticket. Après avoir reçu un TGT on essai d'exécuter la commande précédente *hdfs dfs -mkdir /test-kerberos*.

```
hduser@nn:~$ kinit hduser@UBOUIRA.DZ
Password for hduser@UBOUIRA.DZ:
hduser@nn:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1001
Default principal: hduser@UBOUIRA.DZ

Valid starting       Expires              Service principal
2019-11-27T21:57:22  2019-11-28T07:57:22  krbtgt/UBOUIRA.DZ@UBOUIRA.DZ
        renew until 2019-11-28T21:56:41
hduser@nn:~$ hdfs dfs -mkdir /test-kerberos
hduser@nn:~$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - hduser supergroup          0 2019-11-27 21:58 /test-hadoop
drwxr-xr-x   - hduser supergroup          0 2019-11-27 21:58 /test-hdfs
drwxr-xr-x   - hduser supergroup          0 2019-11-27 22:00 /test-kerberos
hduser@nn:~$
```

On remarque que la commande de création du répertoire n'a pas générer d'erreur, cela veut dire que notre authentification Kerberos a bien réussi.

3.5 Conclusion

Dans ce chapitre, nous avons défini l'architecture de notre système d'authentification qui se base sur le protocole Kerberos, qui est un protocole d'authentification très utilisé surtout sur les réseaux peu sûr. Nous avons aussi expliqué le principe de fonctionnement de notre système et comment l'authentification Kerberos est utilisée au niveau de Hadoop. Puis nous avons configuré notre réseau pour relier nos différentes machines ensemble, ensuite on a présenter les différentes étapes nécessaires pour installer Hadoop en mode totalement distribué, et aussi l'installation de Kerberos et son utilisation. Ensuite nous avons pu détailler les différentes étapes et la démarche suivie pour configurer Hadoop à utiliser l'authentificationn Kerberos. A la fin on a terminé par donner un exemple de fonctionnement de notre système.

A travers ce chapitre, on a pu appréhender la notion du protocole Kerberos, son langage utilisé et sa procédure d'authentification ainsi que son interaction avec Hadoop.

Conclusion générale et perspectives

L'authentification dans l'environnement Hadoop a connu une évolution rapide. Les premières versions de Hadoop ne contenait aucune disposition relative à l'authentification des utilisateurs. Les versions ultérieures ont ajouté des fonctionnalités de gestion des autorisations limitées pour les fichiers du système HDFS, mais Hadoop n'offrait toujours pas une sécurité d'authentification optimale. Aujourd'hui, Hadoop peut être utilisé avec le protocole Kerberos comme protocole d'authentification primaire.

Kerberos est une méthode d'authentification développée au MIT et qui a grandi pour devenir l'approche d'authentification la plus largement appliqué. L'identification Kerberos est utilisée par plusieurs sociétés comme Apache, Mac OS X, Microsoft Windows, Oracle et d'autres applications comme Samba, NFT, etc.

L'objectif principal de notre travail consistait à mettre en place une solution d'authentification pour Hadoop, assurant le contrôle des accès des utilisateurs, pour cela, nous avons mis en place kerberos et on a configuré Hadoop pour qu'il utilise l'authentification Kerberos, et on est arrivé a sécuriser les services Hadoop afin de ne pas autoriser l'accès à ses services qu'aux utilisateurs authentifiés auprès du KDC.

A travers ce modeste travail, nous avons tenté de présenter un état de l'art sur le Big Data, et nous avons cité parmi ses technologies Hadoop qui est le plus utilisé actuellement pour manipuler et produire du Big Data. Ainsi, nous avons données une vision globale sur l'authentification et discuter quelques uns de ses protocoles les plus connus, et nous nous sommes focalisés sur le mécanisme de fonctionnement de notre solution Kerberos. Puis nous avons détaillé les différentes étapes d'installation de Hadoop et du serveur Kerberos,

ensuite nous avons expliqué la configuration de Hadoop pour qu'il utilise l'authentification Kerberos.

Comme perspectives, nous souhaitons que ce protocole d'authentification est également mis en œuvre sur d'autres services de l'écosystème Hadoop *(pig, hive, Impala, etc)* et pouvoir développer une application (MapReduce) qui utilise l'authentification Kerberos.

Bibliographie

[1] [consulté le 13-12-2019]. https ://www.filfil.eu/2019/05/13/le-big-data/.

[2] Medfouni Hayet. Validation de clustering des données dans un contexte big data, mémoire de master. Université Larbi Ben M'hidi Oum El Bouaghi, 2017/2018.

[3] https : //data-flair.training/blogs/big-data-applications-various-domains/, consulté le 26-11-2019.

[4] https ://data-flair.training/blogs/apache-flink-big-data-unified-platform/, consulté le 26-11-2019.

[5] https ://medium.com/@y2kshehan/hadoop-the-elephant-in-the-big-data-room-e983892c1936, consulté le 26-11-2019.

[6] https ://subscription.packtpub.com/book/big_data_and_business_intelligence/9781784393960/1/ch yarn-architecture, consulté le 26-11-2019.

[7] https ://www.franciscojavierpulido.com/2013/07/bigdata-hadoop-i-definiciones-basicas-y.html, consulté le 26-11-2019.

[8] BRAHAMI Nabila BOUFOUDI Siham. *La sécurité des réseaux informatique à base de kerberos, Université de Bejaïa, Mémoire de Master*. 2014-2015.

[9] KEHOUL Tarek ADJAOUD Yougourthen. *Authentification unique avec CAS et LDAP, Université A/Mira de Béjaïa, Mémoire de Master*. 2011 - 2012.

[10] https ://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/encryptedshuffle.html, consulté le 28-11-2019.

[11] [consulté le 13-12-2019]. https ://www.institut.capgemini.fr/formation-big-data-l-etat-de-l-art-capgemini-institut_p175.

[12] [consulté le 13-12-2019]. https ://www.lebigdata.fr/definition-big-data.

[13] [consulté le 13-12-2019]. https ://www.piloter.org/business-intelligence/big-data-definition.htm.

[14] Maxime VIGIER. *Les big data : une mine d'informations pour les entreprises, University of Economics and Finance Faculty of Business, Mémoire de Master*. 2014.

[15] Mlle MOUZAIA Chahinas épouse REDJDAL Mlle ABBAS Célia. *Le Contrôle d'Accès au Big Data.Cas d'Etude : Internet des Objets, Université A/Mira de Béjaia, Mémoire de Master*. 2016/2017.

[16] [consulté le 23-09-2019]. https ://www.mba-esg.com/actus/enjeux-big-data.

[17] Seref SDJLURJOX Duygu Sinanc Terzi, Ramazan Terzi. *A Survey on Security and Privacy Issues in Big Data, Gazi University, Computer Engineering Ankara, memoire master*. 2015.

[18] [consulté le 13-12-2019]. https ://www.redsen-consulting.com/fr/inspired/data-analyse/cas-dutilisation-big-data.

[19] Benyamina khadidja Taouche sabrina. *Une approche de Dé-identification des données pour la préservation de la vie privée des Big Data, UNIVERSITE Dr. TAHAR MOULAY SAIDA, Mémoire de Master*. 2017-2018.

[20] Mr MATALLAH Houcine. *Vers un nouveau modèle de stockage et d'accès aux données dans les Big Data et les Cloud Computing, UNIVERSITE ABOU-BEKR BELKAID - TLEMCEN*. 2018.

[21] https ://docplayer.fr/60857217-caracteristiques-du-big-data-les-caracteristiques-majeures-du-big-data-sont-resumees-en-trois-lettres-v-volume-velocite-variete.html, consulté le 22-10-2019.

[22] https ://www.saagie.com/fr/blog/les-technologies-et-applications-du-big-data/, consulté le 22/11/2019.

[23] CLOAREC Erwann. *Hadoop : Optimisation et Ordonnancement, Université François-Rabelais, Tours*. 2013 - 2014.

[24] Brighen Assia. *Conception de bases de données volumineuses sur le Cloud, Universite Béjaia, Mémoire de Master*. 2012.

[25] Margaret Rouse. *YARN (Yet Another Resource Negotiator), TechTarget*. 2015.

[26] Abdoul Seck. *Sécurité et Big Data : 4 briques à mettre en place pour sécuriser votre projet, CCM Benchmark*. 20/11/15.

[27] Hamoudi Asma Tighilt Dihia. *Mise en place d'une solution d'authentification RADIUS, Université de Bejaïa, Mémoire de Master*. 2012-2013.

[28] [consulté le 13-12-2019]. https ://www.lemagit.fr/definition/Authentification.

[29] Sébastien Gambs. *Authentification de messages et mots de passe*. 9 novembre 2015.

[30] DJELLALBIA Amina. *Authentification Anonyme dans un environnement Cloud, Université de Bejaïa, Mémoire de Master*. 2015/2016.

[31] https ://connect.ed-diamond.com/misc/misc-054/attaque-sur-le-protocole-kerberos, consulté le 19/09/2019.

[32] LÉO GONZALES. *Kerberos : Principe de fonctionnement.* 18 JUILLET 2018.

[33] DJAFRI Farouk DJAFRI Driss. *Implémentation d'un protocole d'authentification et de partage de clés dans un système distribué, Université de Bejaïa, Mémoire de Master*. 2015/2016.

[34] https ://www.ovh.com/fr/ssl/fonctionnement-ssl.xml, consulté le 24-11-2019.

[35] aymeric bernard stephane brinster, guillaume lecomte. *Nouvelles technologies reseaux, Université de marne la vallee, Mémoire de Master*. 2002-2003.

[36] https ://www.securiteinfo.com/cryptographie/ssl.shtml, consulté le 25-11-2019.

[37] ZHANG Tuo. *Sécurité avancée des réseaux*. 2013-2014.

[38] https ://fr.talend.com/resources/what-is-hadoop/, consulté le 28-11-2019.

[39] https ://www.bart-konieczny.com/fr/blog/hadoop/hdfs-interaction-entre-namenode-et-datanodes, consulté le 26-11-2019.

[40] https ://www.supinfo.com/articles/single/8093-hdfs, consulté le 26-11-2019.

[41] https ://hadoop.apache.org/docs/r3.2.1/hadoop-yarn/hadoop-yarn-site/yarn.html, consulté le 28-11-2019.

[42] https ://stph.scenari-community.org/contribs/nos/hadoop2/co/comprendre_hdfs.html, consulté le 26-11-2019.

[43] https ://www.lebigdata.fr/yarn-apache-hadoop, consulté le 20-10-2019.

[44] https ://web.mit.edu/kerberos/krb5-1.12/doc/, consulté le 28-11-2019.

[45] https ://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/securemode.html, consulté le 28-11-2019.

[46] https ://dzone.com/articles/differences-between-keystore-amp-truststore, consulté le 28-11-2019.

[47] [consulté le 13-12-2019]. https ://fntic.univ-ouargla.dz/images/biblio/InfoPDF/4.pdf.