المدرسة العليا للتكنولوجيا - الداخلة
ⵜⵉⵎⵙⵙⵓⵔⵉⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵙⵙⵏⵜⵉⵙⵜ - ⴰⴷⴰⵅⵍⴰ
ÉCOLE SUPÉRIEURE DE TECHNOLOGIE - DAKHLA

École Supérieure de Technologie - DAKHLA

Module : **Big Data**

Professeur : **Pr. DIB**

PROJET HADOOP

Big Data

Rapport théorique et pratique complet
MapReduce, Yarn, HDFS

Réalisé par :

Lucien YAOGO

Filière : Cybersécurité & IA

Année universitaire : 2025-2026

Table des matières

Table des Figures

Introduction

Avec la croissance exponentielle des données, les entreprises ont besoin d'outils capables de stocker, traiter et analyser efficacement de grands volumes. Hadoop, Framework Big Data, offre une solution distribuée avec HDFS pour le stockage et MapReduce pour le traitement.

Ce TP vise l'installation et la configuration de Hadoop 3.4.2 sous Ubuntu, incluant HDFS, YARN, Java, les variables d'environnement, SSH sans mot de passe et l'initialisation de HDFS, afin de préparer un environnement opérationnel pour le traitement distribué.

Cette infrastructure permettra d'analyser des logs pour identifier des IP suspectes, repérer les pages ciblées et détecter des erreurs HTTP (403, 404), contribuant à la cybersécurité et fournissant des données exploitables pour des modèles d'intelligence artificielle. Ce TP illustre ainsi l'intérêt de Hadoop pour le traitement de grandes données et la sécurisation des systèmes.

Définition

Hadoop est un Framework open source pour le stockage et le traitement distribué de grandes données. Il utilise **HDFS** (Hadoop Distributed File System) pour le stockage réparti et **MapReduce** pour le traitement parallèle.

Dans ce projet, Hadoop permet d'analyser des logs pour détecter des IP suspectes, repérer les pages ciblées et identifier des erreurs HTTP (403, 404), contribuant à la **cybersécurité** et à l'exploitation des données pour l'**intelligence artificielle**.

Mis en place des Environnements

i. Prérequis :

Mis à jour de Ubuntu

Avant le téléchargement de Hadoop, assurez-vous que votre Système d'Exploitation UBUNTU est à jour.

Si votre Système n'est pas à jour il faut le mettre à jour avant de commencer l'installation en utilisant les commandes suivantes :

 # sudo apt update && sudo apt upgrade -y

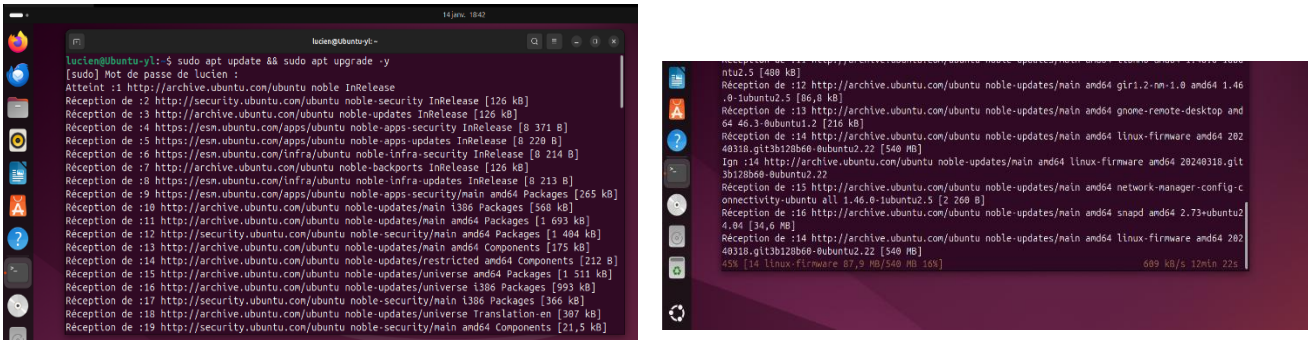


Figure 1 : Commande de mise à jour de Ubuntu

La commande pour vérifier que l'installation s'est bien passée :

 # sudo apt update

ii. Installation de JAVA

Apache Hadoop est développé en Java et s'exécute sur la Java Virtual Machine (JVM), ce qui lui confère portabilité, stabilité et compatibilité sur différents systèmes d'exploitation. Cette dépendance à Java est essentielle, car Hadoop ne peut fonctionner sans un environnement Java correctement installé. Ainsi, avant de procéder à l'installation et à la configuration de Hadoop, il est indispensable de mettre en place Java sur le système. Hadoop 3.x fonctionne très bien avec OpenJDK 11.

 # sudo apt install openjdk-11-jdk -y

Il est important de vérification si JAVA a été bien installé avec la commande :

 # java -version

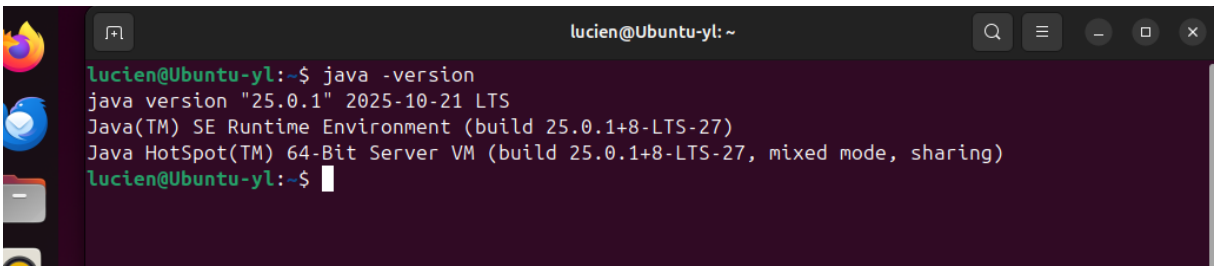


Figure 2 :Vérification de la version de JAVA installée

iii. Configurer Java (JAVA_HOME)

Il faut trouver le chemin de Java avec la commande suivante :

 # readlink -f /usr/bin/java

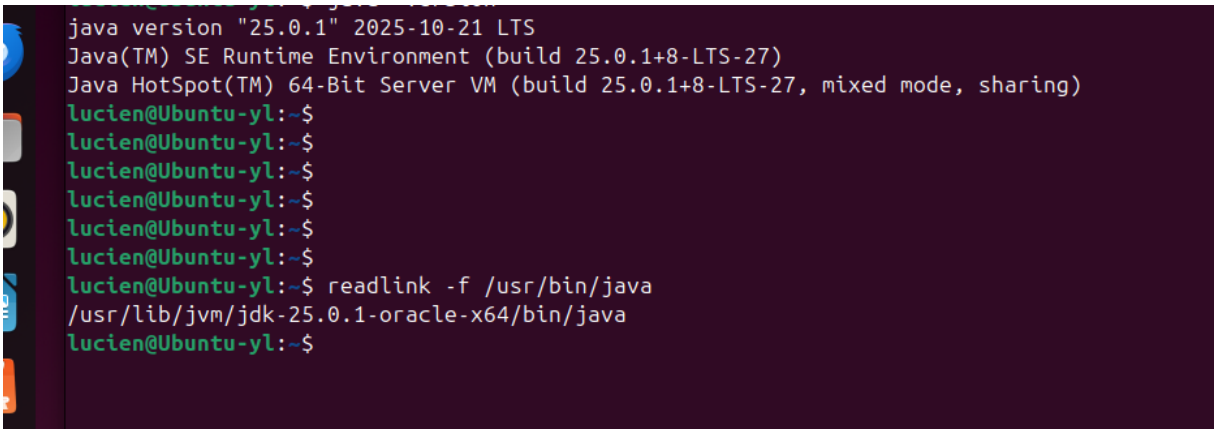


Figure 3 : Vérification du chemin de JAVA_HOME

Sur la photo on peut facilement retrouver le chemin (JAVA_HOME) de JAVA.

JAVA_HOME = */usr/lib/jvm/java-11-openjdk-amd64*

Ce chemin doit-être ajouter dans le fichier *~/.bashrc.*

Le fichier *~/.bashrc* joue un rôle essentiel dans la configuration de l'environnement de travail sous Linux. Il permet de définir et de charger automatiquement les variables d'environnement nécessaires au bon fonctionnement des applications à chaque ouverture d'un terminal. Dans le cadre de l'installation de Hadoop, ce fichier est particulièrement important, car il permet de spécifier des chemins critiques tels que **JAVA_HOME** et **HADOOP_HOME**. Une configuration correcte du fichier *.bashrc* garantit que Java et Hadoop sont correctement reconnus par le système, assurant ainsi le démarrage et l'exécution sans erreur des services Hadoop.

Donc éditons ce fichier et ajoutons ces deux lignes de code dans les variables d'environnement de notre système :

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export PATH=$PATH:$JAVA_HOME/bin

La commande pour éditer le fichier : *# nano ~/.bashrc*

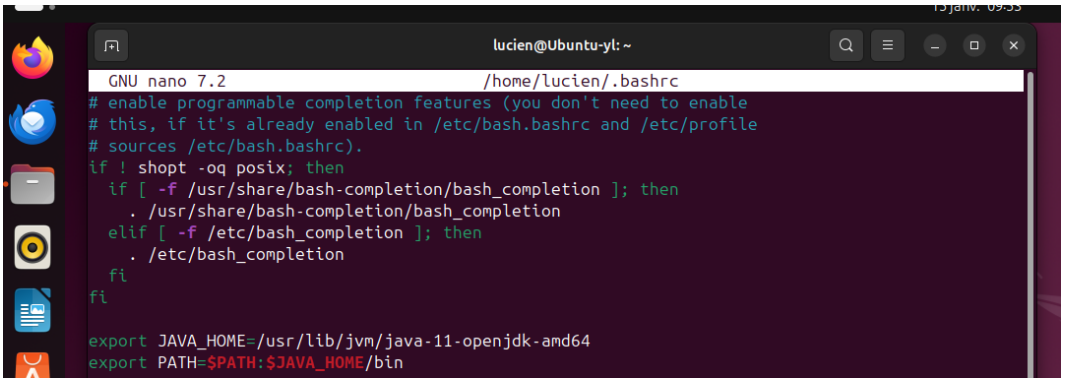


Figure 4 : Edit le fichier /.bashrc

Appliquer cette commande pour vérifier qu'il n'y pas eu de problème avec le fichier :

 # source ~/.bashrc

iv. Téléchargement de Hadoop 3.4.2

Pour le téléchargement de Hadoop, rendez-vous sur la page officielle de Hadoop et télécharger la version récente 3.4.2 c'est un dossier compressé.

- La commande de téléchargement sous Firefox de Ubuntu.

  https://dlcdn.apache.org/hadoop/common/hadoop-3.4.2/hadoop-3.4.2.tar.gz
- La commande de téléchargement avec la ligne de commande (Terminale) sous Ubuntu.

  # wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.2/hadoop-3.4.2.tar.gz

Quand le téléchargement fini, il faut décompresser le dossier que vous venez de télécharger.
La commande pour la décompression du dossier :

 # tar -xzf hadoop-3.4.2.tar.gz

Il faut déplacer le dossier décompressé vers le répertoire ***/usr/local/hadoop***

 # mv hadoop-3.4.2 /usr/local/Hadoop

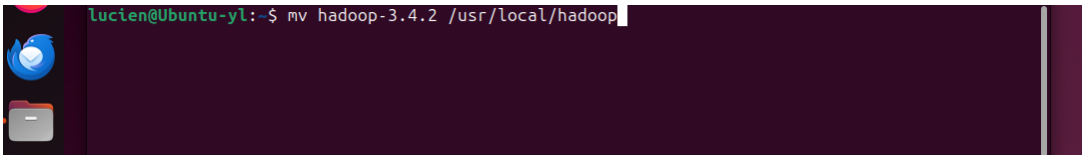


Figure 5 : Déplacement du fichier HADOOP vers le répertoire /usr/local/HADOOP

Configuration des fichiers nécessaires

a. Les Variables d'environments

Comme nous l'avions énuméré ci-dessus, le fichier *~/.bashrc* est le fichier qui définit et charge les variables d'environnement de notre système, donc il nécessaire de finir les chemins et les scriptes pour que notre système puisse reconnaitre Hadoop.
Éditons à nouveau notre fichier *~/.bashrc* et ajoutons ces scriptes à la fin du fichier (à la suite des scriptes utilisés pour JAVA) :

export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

La commande pour éditer le fichier :

 # nano ~/.bashrc

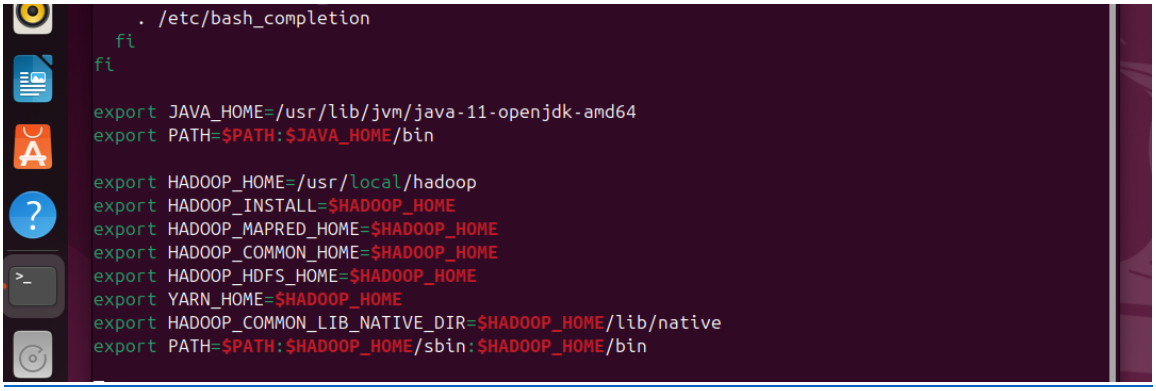


Figure 6 : Configuration des variable d'environnement

On applique cette commande à nouveau :

 # source ~/.bashrc

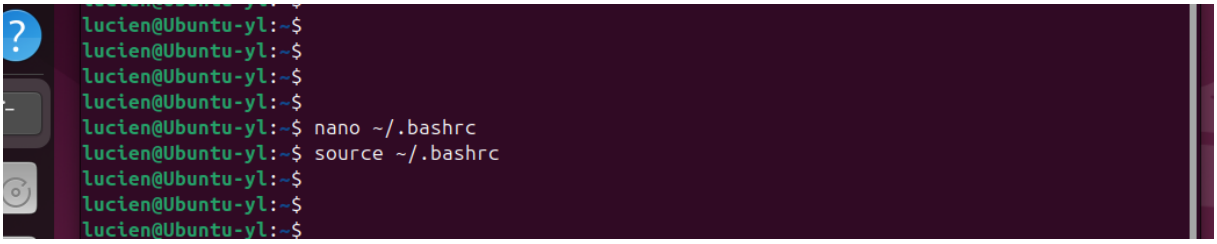


Figure 7 : Vérification du fichier /.bashrc

Nos variables d'environnements ont été bien chargées dans notre fichier.
Vérifions si Hadoop a été bien installer avec la commande suivante :

 # hadoop version

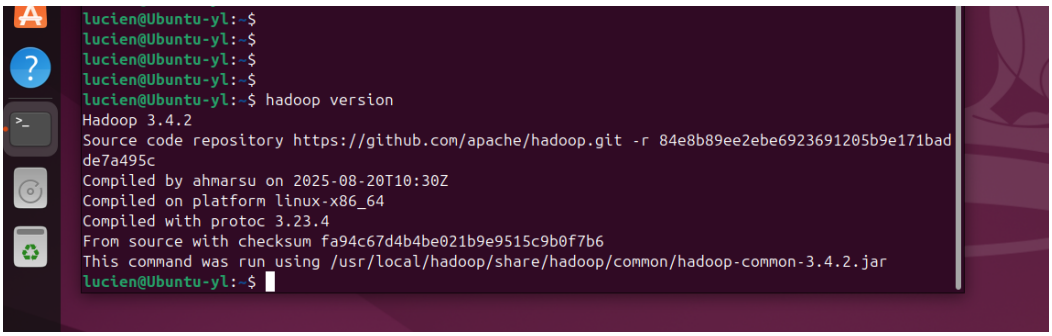


Figure 8 : Vérification de la version de HADOOP installée

Nous voyons bien maintenant la version de Hadoop qui est : Hadoop 3.4.2. Cela montre que Hadoop a été bien installé, mais pas configuré pour le moment.

b. Configuration de Hadoop

Pour la configuration de Hadoop est très important de modifier certains fichiers qui se trouve le Hadoop. Chaque fichier joue rôle cruciale pour bon fonctionnement de Hadoop.

Nous allons dans les dossiers en config utilisant le chemin de Hadoop (HADOOP_HOME) :

 # cd $HADOOP_HOME/etc/hadoop

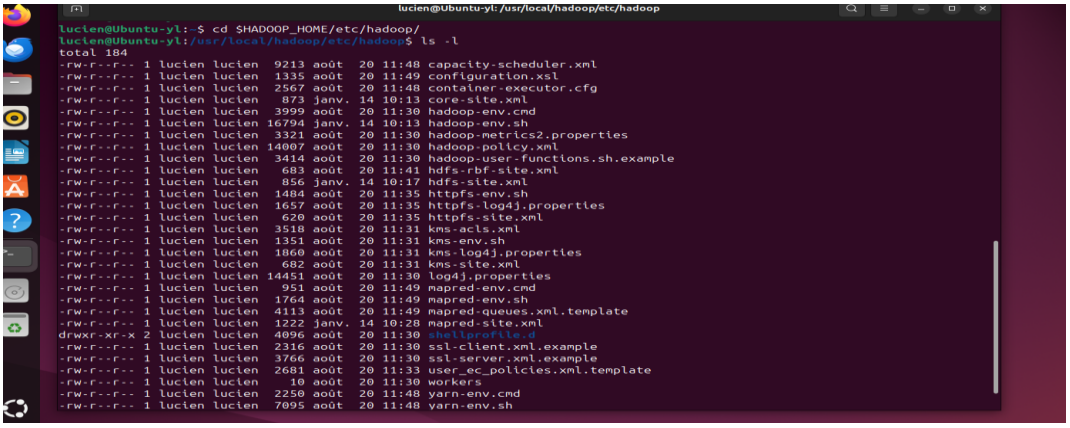


Figure 9 : Le dossier HADOOP

La suite de notre configuration de Hadoop sera consacrée à la configuration des fichier Hadoop nécessaire pour le bon fonctionnement de Hadoop.

1. hadoop-env.sh

Notre premier fichier est : ***hadop-env.sh.***

Le fichier `hadoop-env.sh` est utilisé pour définir les variables d'environnement spécifiques à Hadoop, notamment le chemin vers Java (`JAVA_HOME`). Son édition est indispensable, car Hadoop s'appuie sur ces paramètres pour démarrer correctement ses services (NameNode, DataNode, YARN, etc.). Une configuration correcte de ce fichier garantit que Hadoop utilise la bonne version de Java et fonctionne de manière stable sur le système.

 # nano hadoop-env.sh

On décommente la ligne JAVA_HOME et on écrit :

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

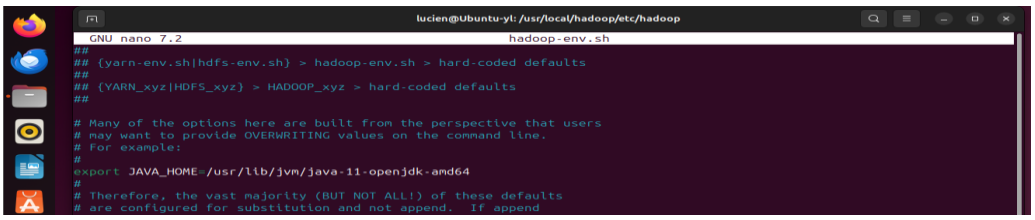


Figure 10 : Configuration du fichier hadoop-env.sh

2. core-site.xml

Notre deuxième fichier est : core-site.xml

Le fichier `core-site.xml` est un fichier de configuration central de Hadoop. Il permet de définir les paramètres fondamentaux du système, notamment l'adresse du NameNode et le système de fichiers par défaut (HDFS). Sa configuration est essentielle, car elle indique à Hadoop où et comment accéder aux données, assurant ainsi la communication correcte entre les différents composants du cluster.

 # nano core-site.xml

On ajoute ces scriptes dans notre fichier :

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://localhost:8020</value>
         </property>
</configuration>
```

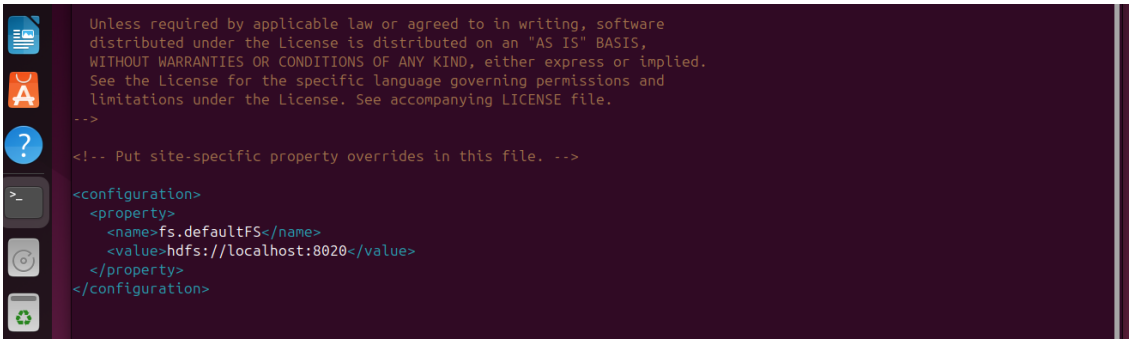


Figure 11 : Configuration du fichier core-site.xml

3. hdfs-site.xml

Notre troisième fichier est : ***hdfs-site.xml***

Le fichier `hdfs-site.xml` permet de configurer le système de fichiers distribué HDFS. Il définit des paramètres essentiels tels que le facteur de réplication, les répertoires de stockage des données et des métadonnées, ainsi que le mode de fonctionnement du NameNode et des DataNodes. Une configuration correcte de ce fichier garantit la fiabilité, la disponibilité et les performances du stockage des données dans Hadoop.

 # nano hdfs-site.xml

On ajoute ces scriptes dans notre fichier :

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
</configuration>
```

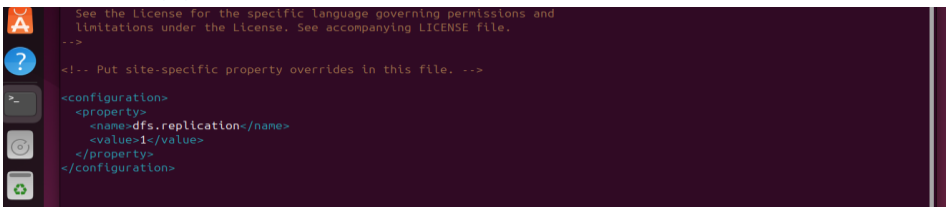


Figure 12 : Configuration du fichier hdfs-site.xml

4. mapred-site.xml

Notre quatrième fichier est : ***mapred-site.xml***

Dans certain cas ce dossier ***mapred-site.xml*** n'existe pas dans le dossier Hadoop. Donc créer ce dossier il faudra copier le contenue du dossier ***mapred-site.xml.template*** vers ***mapred-site.xml***

 # cp mapred-site.xml.template mapred-site.xml

Le fichier `mapred-site.xml` est utilisé pour configurer le framework MapReduce de Hadoop. Il permet de définir le mode d'exécution des jobs, notamment l'utilisation de YARN comme gestionnaire de ressources, ainsi que d'autres paramètres liés au traitement des données. Sa configuration est indispensable pour assurer l'exécution correcte et efficace des tâches MapReduce dans l'environnement Hadoop.

 # nano mapred-site.xml

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
                <name>yarn.app.mapreduce.am.env</name>
                <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
        </property>
        <property>
                <name>mapreduce.map.env</name>
                <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
        </property>
        <property>
                <name>mapreduce.reduce.env</name>
                <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
        </property>
</configuration>
```

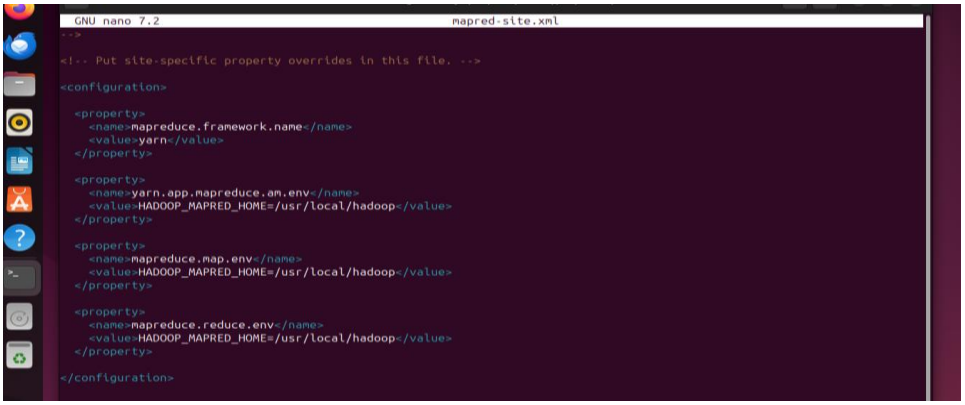


Figure 13 : Configuration du fichier mapred-site.xml

5. yarn-site.xml

Notre cinquième fichier est : ***yarn-site.xml***

Le fichier `yarn-site.xml` configure le framework YARN (Yet Another Resource Negotiator), qui gère les ressources et l'exécution des applications dans Hadoop. Il permet de définir des paramètres essentiels comme l'adresse du ResourceManager, la gestion des conteneurs et la répartition des ressources entre les nœuds du cluster. Une configuration correcte de ce fichier est cruciale pour assurer une allocation efficace des ressources et le bon fonctionnement des jobs Hadoop.

```
# nano yarn-site.xml
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
</configuration>
```

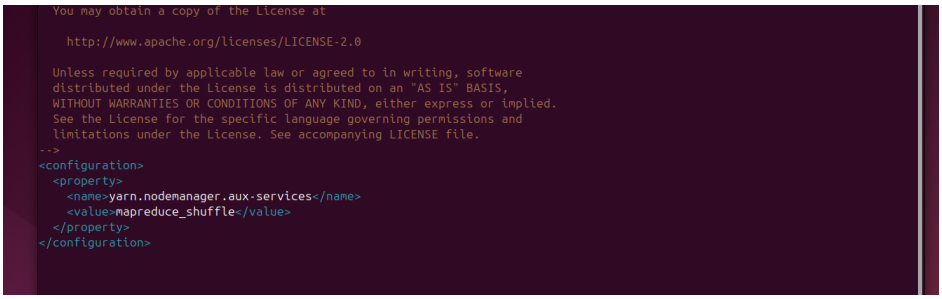


Figure 14 : Configuration du fichier yarn-site.xml

Configuration de SSH (sans mot de passe)

L'authentification SSH sans mot de passe permet à un système de se connecter automatiquement à un autre sans intervention manuelle. Dans le cadre de Hadoop, cette configuration est indispensable pour permettre au NameNode de démarrer, arrêter et gérer les services sur les différents nœuds du cluster de manière transparente. Elle repose sur l'utilisation de clés SSH, garantissant à la fois la sécurité et l'automatisation des opérations nécessaires au bon fonctionnement du cluster Hadoop.

```
# ssh-keygen -t rsa -P
# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

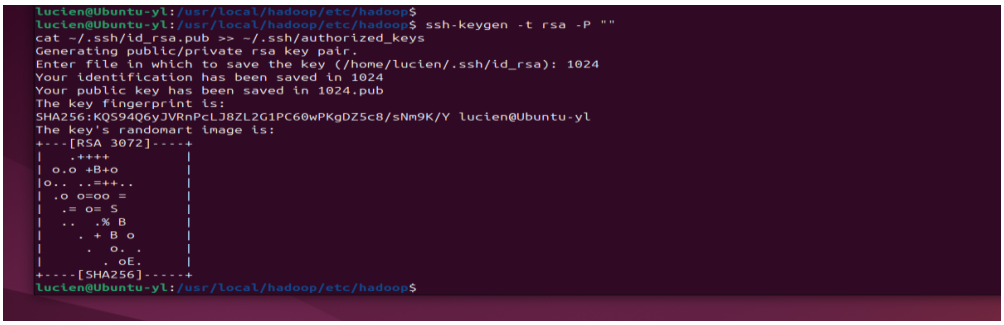


Figure 15 : Configuration de SSH (sans mot de passe)

La commande pour tester **la connexion SSH vers la machine locale**

 # ssh localhost

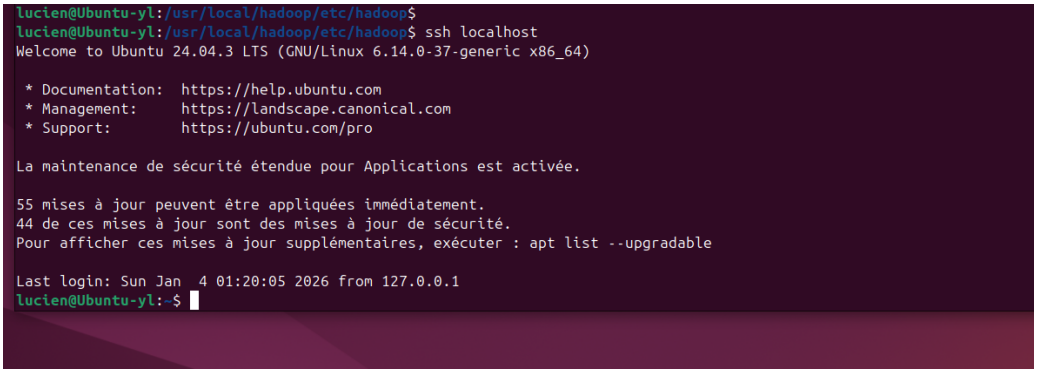


Figure 16 : La connexion SSH vers la machine local

Initialiser HDFS

La commande `hdfs namenode -format` initialise le système de fichiers HDFS en formatant le NameNode. Elle crée la structure de métadonnées nécessaire pour suivre les fichiers et blocs dans le cluster. Cette étape est indispensable lors de la première configuration d'Hadoop, car elle prépare HDFS à stocker et gérer les données.

 # hdfs namenode -format

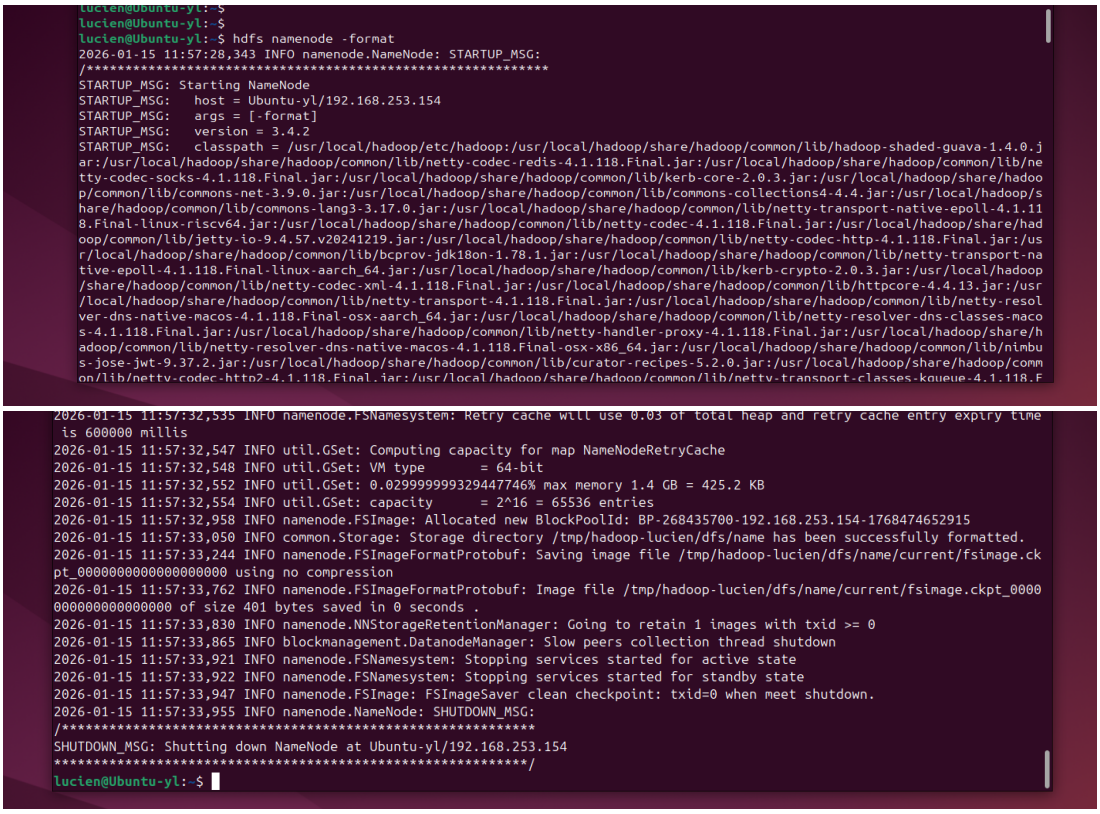


Figure 17 : Initialisation de système de fichiers HDFS en formatant le NameNode

Hadoop est bien configuré et prête à être démarrer.

Démarrage de Hadoop

Pour le démarrage de Hadoop deux commandes est nécessaires pour le démarrage complet de Hadoop.

 # start-dfs.sh

 # start-yarn.sh

`start-dfs.sh`

Cette commande démarre les services du système de fichiers HDFS, à savoir le NameNode, le SecondaryNameNode et les DataNodes. Elle permet au cluster Hadoop de stocker et gérer les données de manière distribuée.

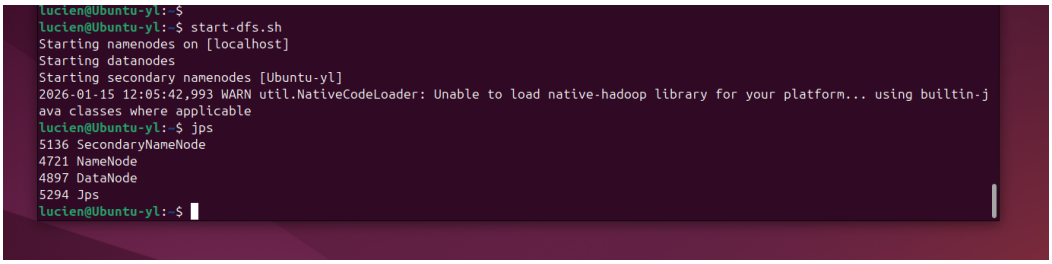


Figure 18 : Démarrage du service HDFS

`start-yarn.sh`

Cette commande lance les services YARN, incluant le ResourceManager et les NodeManagers. Elle prépare le cluster à exécuter des jobs et applications distribués sur les nœuds Hadoop.

Figure 19 : Démarrage du service yarn

Comme on peut constater l'utilisation de la commande ***jps,*** elle a pour rôle de vérifier que tous les services Hadoop sont bien démarrés.

Figure 20 : Verification des services démarres

Visualisation de Hadoop Interface Web

A. NameNode

Le lien pour se connecter directement au service NameNode sur Firefox : *http://localhost:9870*

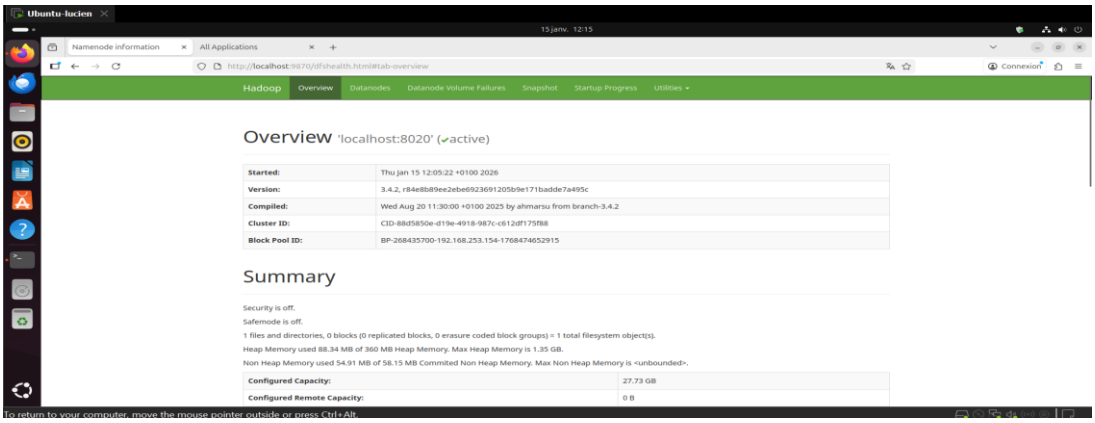


Figure 21 : NameNode Interface Web

B. YARN

Le lien pour se connecter directement au service Yarn sur Firefox : *http://localhost:8088*

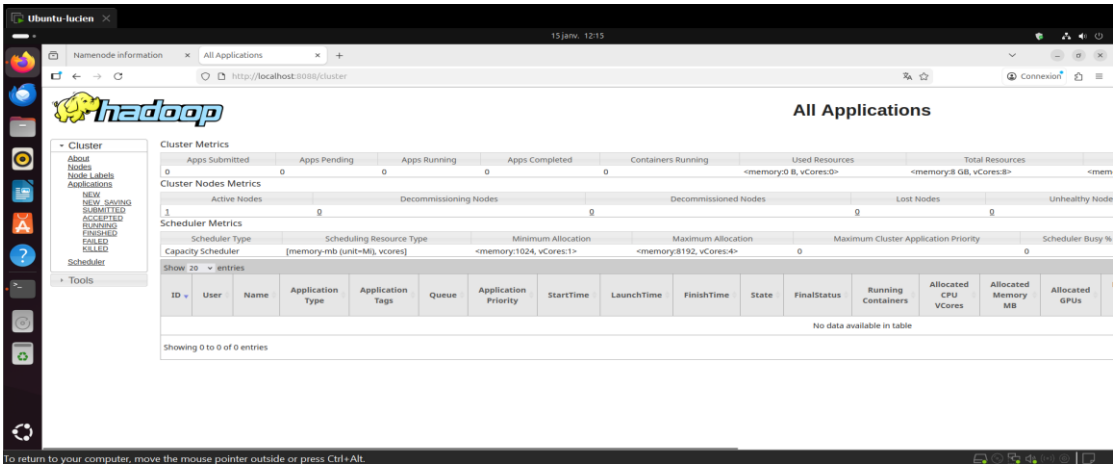


Figure 22 : YARN Interface Web

Pratique : Création des répertoires de tests

Pour commencer à faire de tests, il est crucial de créer un (ou des) répertoire(s) de test. Dans la suite de travail, nous allons créer un répertoire nommé ***/test*** en utilisant la commande suivante :

 # hdfs dfs -mkdir /test

La commande suivante permet de vérifier si le répertoire a bien été créer.

 # hdfs dfs -ls /

Terminal :

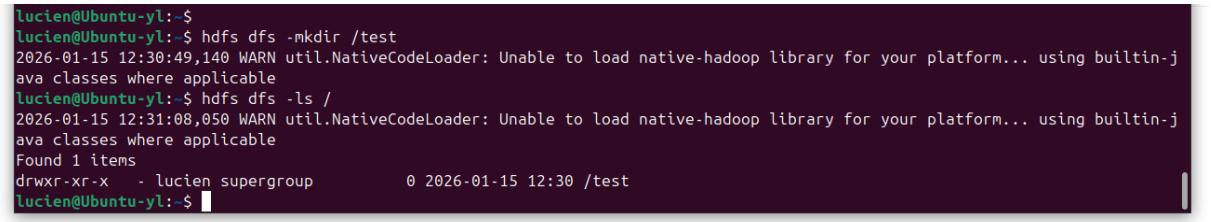


Figure 23 : Création d'un répertoire de travail pour les tests

Interface Web :

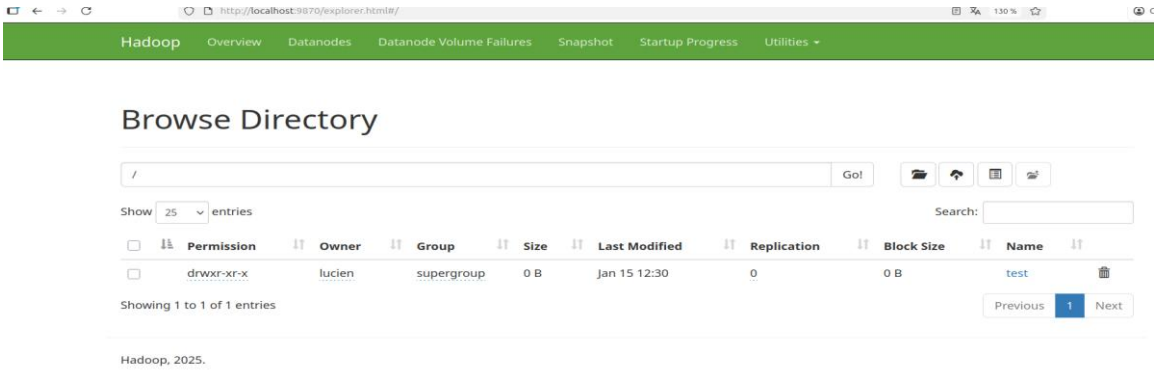


Figure 24 : Vérification Par Interface Web

Dès à présent nous pouvons confirmer que

Hadoop 3.4.2 installé

HDFS + YARN fonctionnels correctement.

Hadoop & Cybersécurité

Introduction

Hadoop n'est pas seulement un outil de stockage et de traitement massif de données : il peut aussi jouer un rôle clé en cybersécurité. Grâce à sa capacité à analyser rapidement de grandes quantités de données, Hadoop permet de traiter des logs de serveurs pour détecter des adresses IP suspectes, identifier des tentatives d'accès non autorisées ou repérer des erreurs HTTP comme les codes 403 et 404. Cela offre aux administrateurs un moyen efficace de surveiller et de sécuriser les systèmes d'information face aux menaces potentielles.
Dans cette nous allons télécharger un fichier logs pour analyser les données comme :

- ***Détecter les erreurs 404***

- ***Trouver les pages les plus attaquées***
- ***Identifier les IP suspectes.***

Il est important de noter que dans chaque point que nous allons aborder, nous allons créer deux fichiers **.py** qui seront implémenter et analyser avec Hadoop.

Téléchargement du fichier logs

Téléchargement du fichier **apache_logs** depuis le Github (logs Apache) sous Ubuntu avec la commande (Terminal) suivante :

wget https://raw.githubusercontent.com/elastic/examples/master/Common%20Data%20Formats/apache_logs/apache_logs

Création & configuration des fichiers .py

1. Détecter les erreurs 404

Pour les erreurs **404** nous allons programmer un script qui permet d'extraire et compter les erreurs **404** à partir des logs web, afin d'identifier les pages les plus ciblées et détecter d'éventuelles activités malveillantes dans une approche cybersécurité.

Ce programme sera enregistré dans un fichier python que nous allons dans notre système :

mapper_404.py

```
# nano mapper_404.py
#!/usr/bin/env python3  # ← Cette ligne est importante pour l'exécution du scripte du fichier.
import sys
for line in sys.stdin:
        parts = line.strip().split()
        if len(parts) > 8:
                status = parts[8]  # ← code HTTP
                url = parts[6]     # ← page demandée
                if status == '404':
                        print(f"{url}\t1")
```

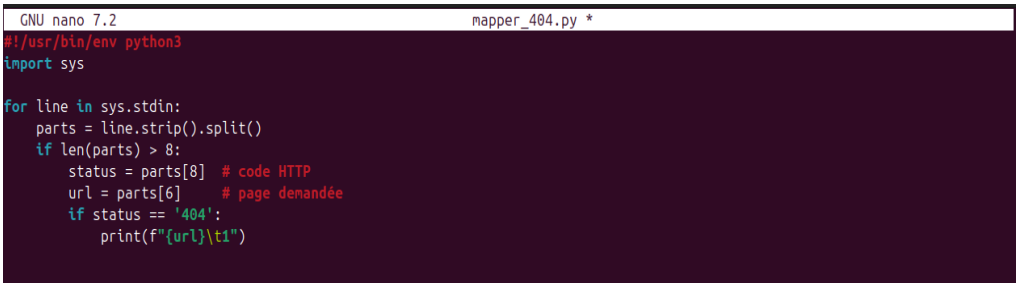


Figure 25 : Script bash pour détecter les erreurs 404 logs du fichier mapper_404.py

reducer_404.py

Ce script regroupe et additionne les erreurs **404** par URL afin d'identifier les pages les plus fréquemment ciblées. Il contribue à l'analyse des logs web dans un contexte de surveillance et de cybersécurité.

```
# nano reducer_404.py
#!/usr/bin/env python3
import sys
from collections import defaultdict
counts = defaultdict(int)
for line in sys.stdin:
    url, count = line.strip().split('\t')
    counts[url] += int(count)
for url, count in sorted(counts.items(), key=lambda x: x[1], reverse=True):
    print(f"{url}\t{count}")
```

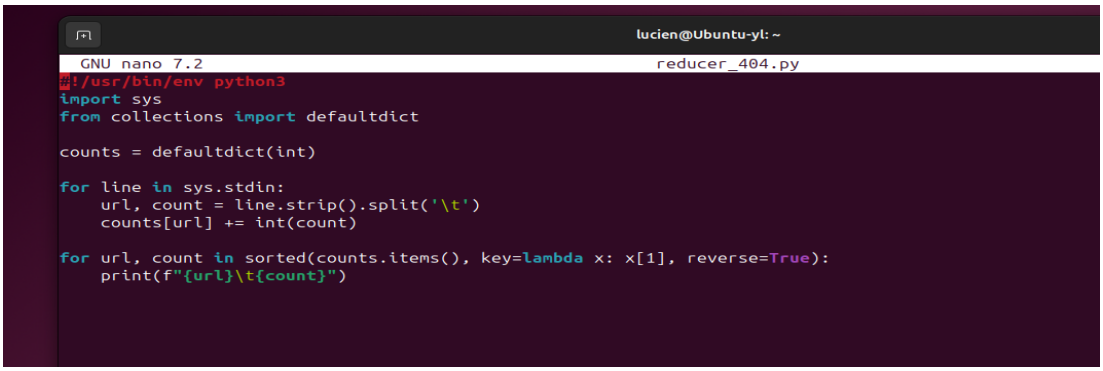


Figure 26 : Script bash pour détecter les erreurs 404 logs du fichier reducer_404.py

Nous allons exécuter les scriptes au fur à mesure pour voir les résultats de nos tests. Pour cela est important de charger notre fichier Apach_logs dans un répertoire de travail /data/logs. Avant de commencer les tests rassurer vous que Hadoop est bien démarrer avec la commande jps.

La commande de chargement de fichier Apach_logs que nous avons téléchargé dans Hadoop :

```
# hdfs dfs -put apache_logs /data/logs
```

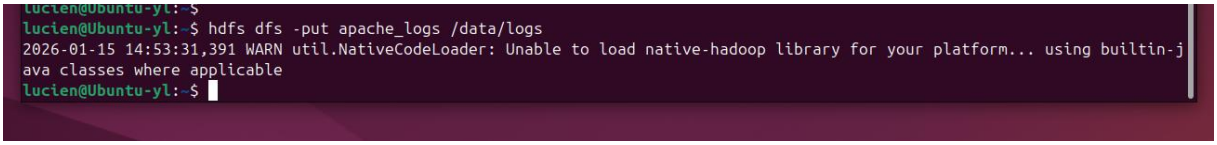


Figure 27 : Chargement de fichier logs dans le repertoire de travail /data/logs

```
# hdfs dfs -ls /data/logs
```

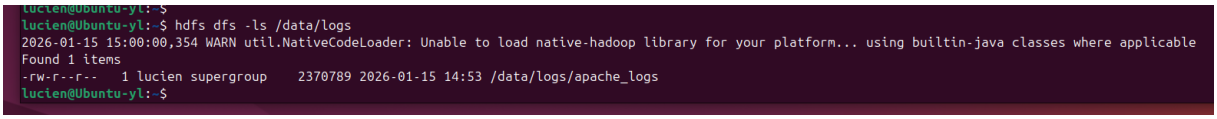


Figure 28 : Verification avec hdfs dfs -ls /data/logs

Pour Tester les deux fichiers *mapper_404.py* et *reducer_404.py :*

 # hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-.jar -input /data/logs/apache_logs -output /output/erreur_404 -mapper mapper_404.py -reducer reducer_404.py -file mapper_404.py -file reducer_404.py*

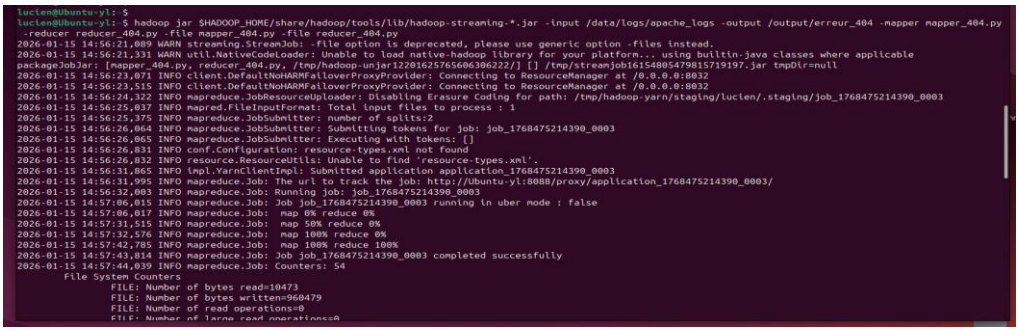


Figure 29 : Exécution des deux fichiers mapper_404.py & reducer_404.py

Pour voir le résultat on utilise la commande suivante :

*# hdfs dfs -cat /output/errors_404/part-**

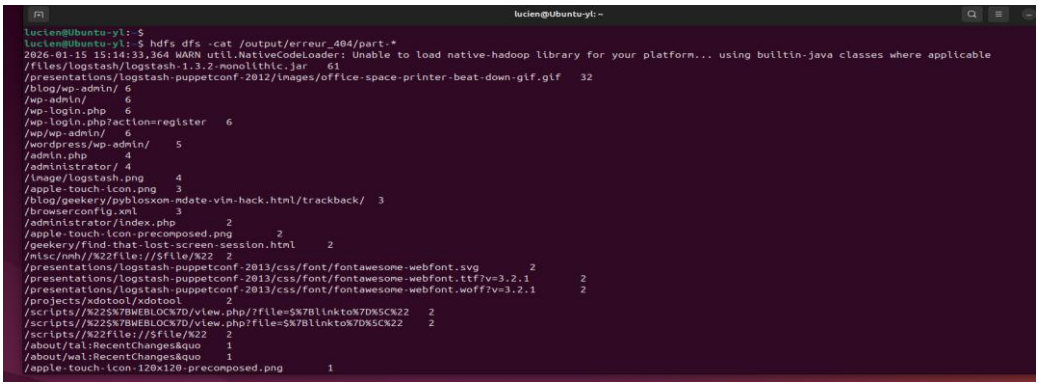


Figure 30 : Le résultats des exécutions du script de détection des erreurs_404

2. Trouver les pages les plus attaquées

mapper_page_attack.py

Ce fichier Mapper (mapper_page_attack.py) a pour but d'extraire les pages web et les adresses IP à partir des logs serveur afin d'analyser les comportements d'accès et identifier des activités suspectes dans un contexte de cybersécurité.

 # nano mapper_page_attack.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    parts = line.strip().split()
    if len(parts) > 8:
        ip = parts[0]
        page = parts[6]
        print(f"{page}\t{ip}")
```

Figure 31 : Script bash pour trouver les pages les plus attaquées : mapper_page_attack.py

reducer_page_attack.py

Ce fichier Reducer (reducer_page_attack.py) a pour but d'analyser les accès aux pages web en regroupant les adresses IP distinctes, afin d'identifier les pages les plus sollicitées et potentiellement ciblées par des attaques. Il contribue à la détection de comportements suspects dans un contexte de cybersécurité.

```
# nano reducer_page_attack.py
#!/usr/bin/env python3
import sys
from collections import defaultdict
ips_per_page = defaultdict(set)
for line in sys.stdin:
    page, ip = line.strip().split('\t')
    ips_per_page[page].add(ip)
    for page, ips in sorted(ips_per_page.items(), key=lambda x: len(x[1]), reverse=True):
        print(f"{page}\t{len(ips)}\t{', '.join(list(ips)[:5])}...")
```

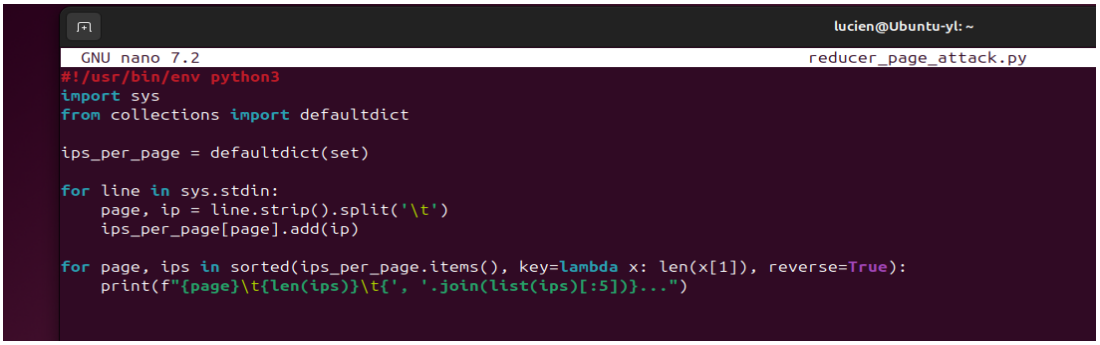


Figure 32 : Script bash pour trouver les pages les plus attaquées : reducer_page_attack.py

Pour Tester les deux fichiers *mapper_page_attack.py* et *reducer_page_attack.py :*

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-.jar -input /data/logs/apache_logs -output /output/page_attack -mapper mapper_page_attack.py -reducer reducer_page_attack.py -file mapper_page_attack.py -file reducer_page_attack.py*

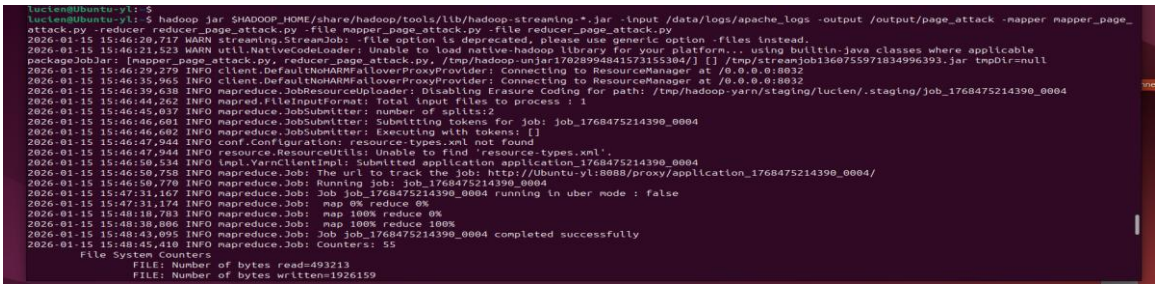


Figure 33 : Exécution des deux fichiers mapper_page_attack.py & reducer_page_attack.py

Pour voir le résultat on utilise la commande suivante :

*# hdfs dfs -cat /output/page_attack/part-**

Figure 34 : Le résultats des exécutions du script de détection des pages attaquées

3. Identifier les IP suspectes.

mapper_ip.py

Ce fichier Mapper (*mapper_ip.py*) extrait les adresses IP et les codes de réponse HTTP à partir des logs serveur afin d'analyser le comportement des clients et identifier des activités suspectes dans un contexte de cybersécurité.

 # nano mapper_ip.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    ip = line.split()[0]
    print(f"{ip}\t1")
```

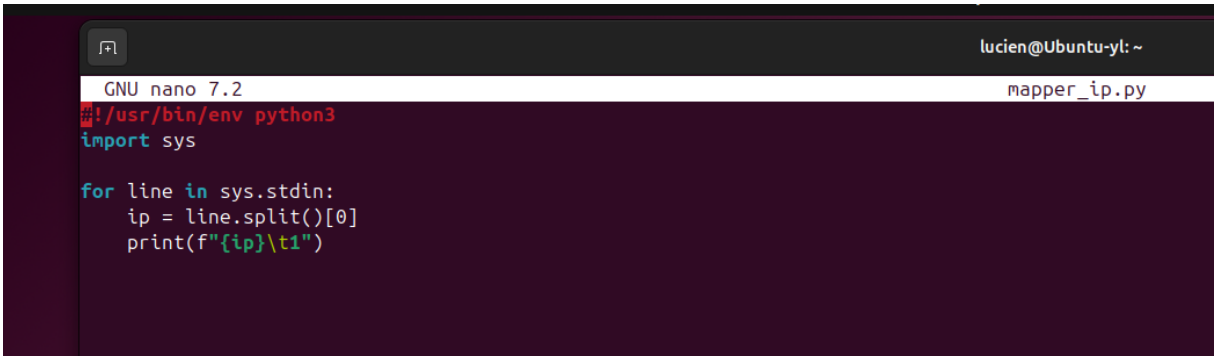


Figure 35 : Script bash pour identifier ip suspects du fichier logs : mapper_ip.py

reducer_ip.py

Ce Reducer regroupe et additionne les occurrences par adresse IP afin d'identifier les clients les plus actifs ou potentiellement suspects à partir des logs serveur.

```
                    # nano reduce_ip.py
#!/usr/bin/env python3
import sys
current_ip = None
count = 0
for line in sys.stdin:
        ip, value = line.split("\t")
        value = int(value)
        if ip == current_ip:
                count += value
        else:
                if current_ip:
                        print(f"{current_ip}\t{count}")
                current_ip = ip
                count = value
if current_ip:
        print(f"{current_ip}\t{count}")
```

Figure 36 : Script bash pour identifier ip suspects du fichier logs : reducer_ip.py

Pour Tester les deux fichiers *mapper_ip.py* et *reducer_ip.py :*

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-.jar -D mapreduce.job.reduces=3 -input /data/logs/apache_logs -output /output/ip_activity -mapper mapper_ip.py -reducer reducer_ip.py -file mapper_ip.py -file reducer_ip.py*

L'option « `-D mapreduce.job.reduces=3` » permet de spécifier le nombre de Reducers afin de paralléliser le traitement des données et d'améliorer les performances du job MapReduce.

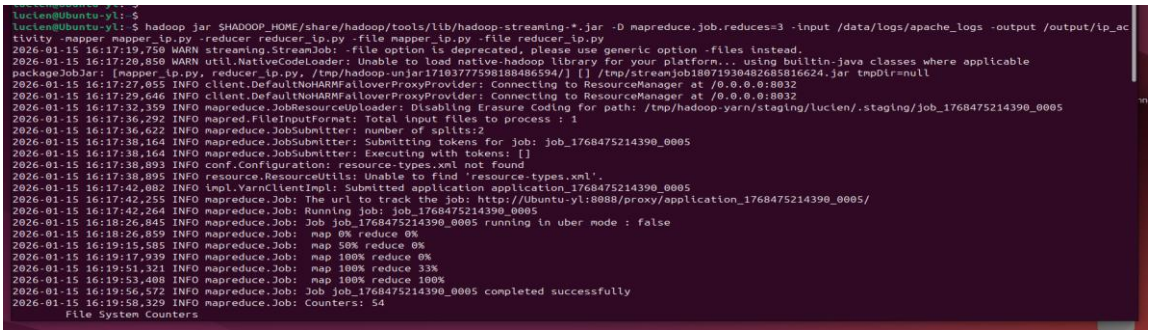


Figure 37 : Les résultats des exécutions du script d'identification des ip suspectes

Pour Vérifier les nombres de Reducers utilisés :

 # hdfs dfs -ls /output/ip_activity

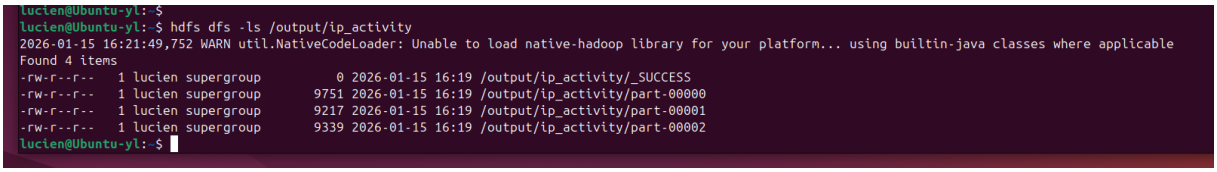


Figure 38 : Verification des nombres de Reducers

Le nombre de Reducers est vérifié par la présence de plusieurs fichiers part-r-* dans le répertoire de sortie HDFS ou via l'interface Web YARN, confirmant la bonne exécution parallèle du job MapReduce.

*# hdfs dfs -cat /output/ip_activity/part-**

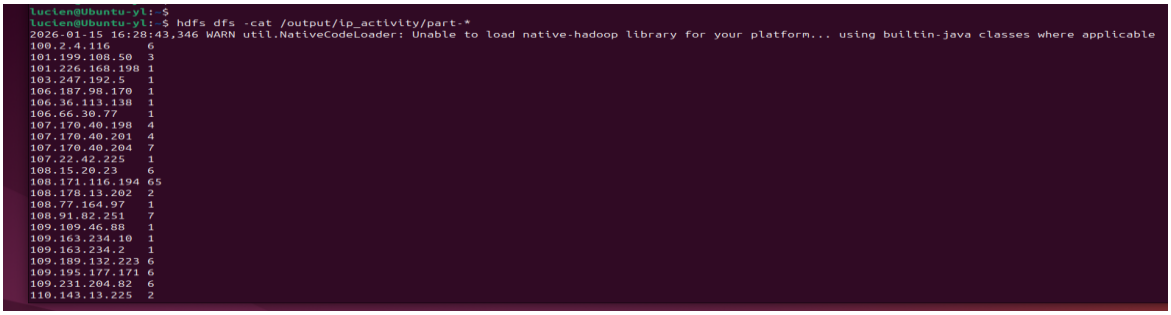
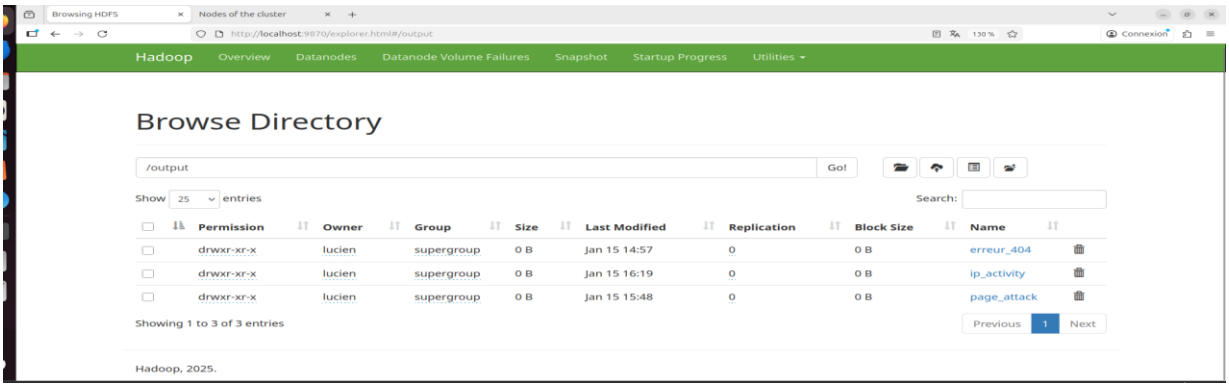


Figure 39 : Afficher les résultats de tous les Reducers

Vérification des résultats à travers l'interface Web de Hadoop



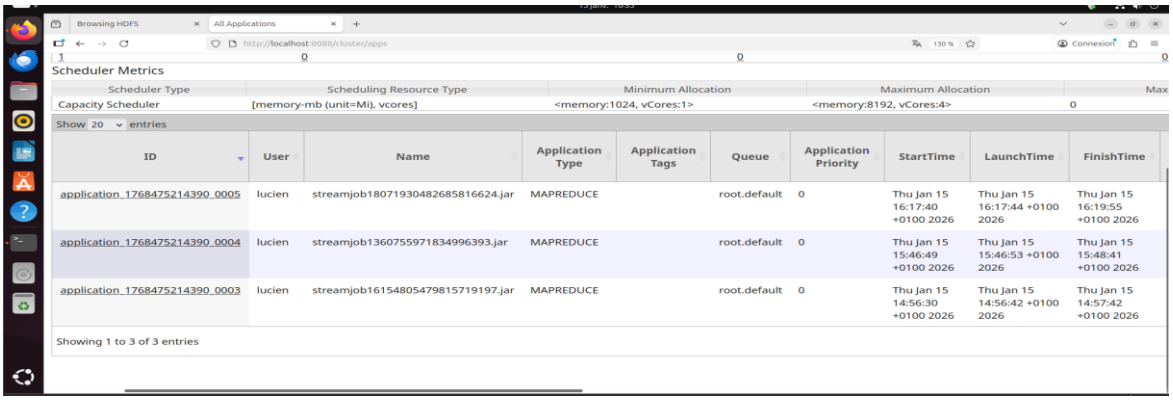


Figure 40 : Les résultats avec l'interface Web

Téléchargement des Résultats Interface Web

Pour télécharger les résultats de nos tests à travers l'interface Web, il faut suivre certaines étapes :

1. Accéder au lien pour se connecter directement au service NameNode sur Firefox : *http://localhost:9870*
2. Aller le bouton Utilities > Browse the file system
3. Cliquez sur le répertoire output
4. Appuyez sur le fichier que vous voulez télécharger (Ici : *error_404*)
5. Appuyez sur error_404 > part-00000
6. Appuyez sur Download pour télécharger.

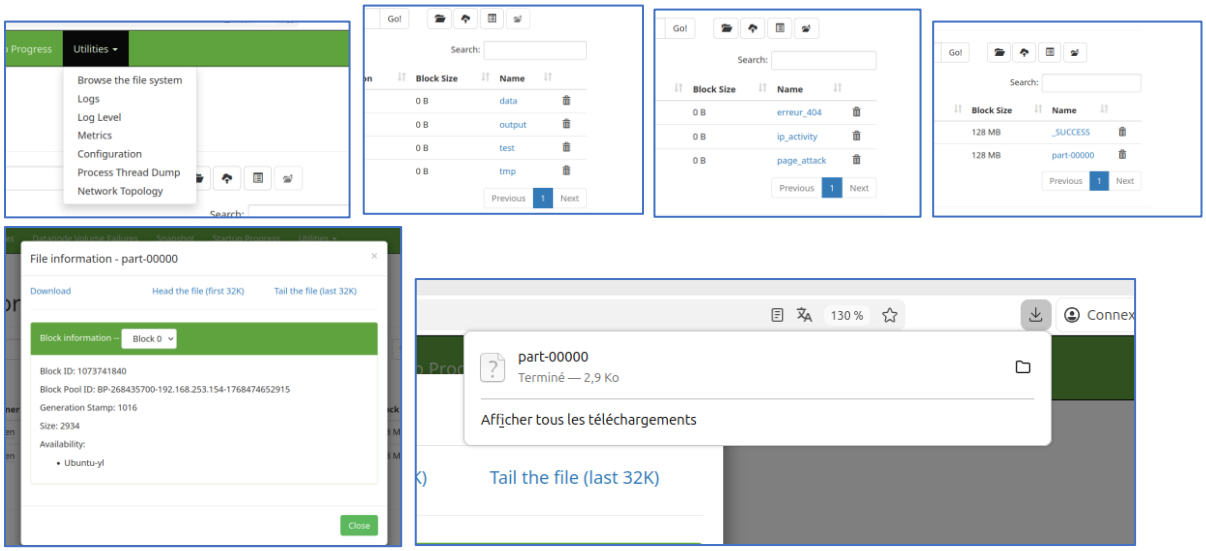


Figure 41 : Comment télécharger les résultats avec l'interface Web

On va ouvrire le fichier pour voir les résultats :

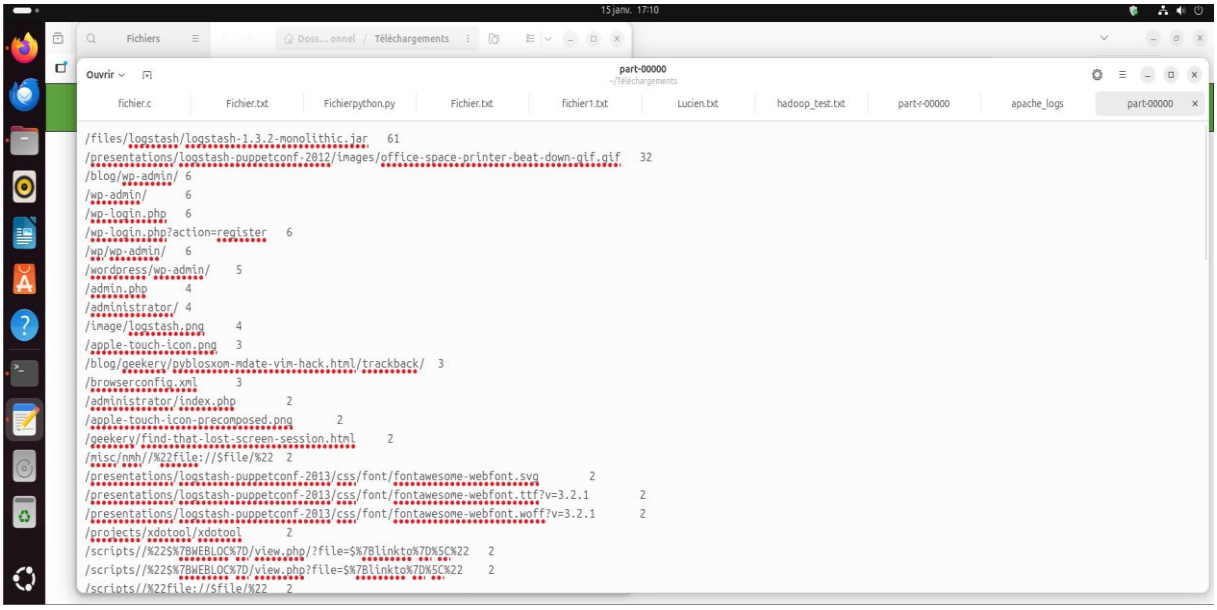


Figure 42 : Visualisation des résultats

Conclusion Générale

Ce TP a permis de découvrir et de mettre en pratique les concepts fondamentaux d'Hadoop et du traitement distribué de données. Nous avons commencé par installer et configurer l'environnement Hadoop, en veillant à la mise en place de Java, à la configuration des fichiers essentiels (`hadoop-env.sh`, `core-site.xml`, `hdfs-site.xml`, `mapred-site.xml`, `yarn-site.xml`) et à l'activation de SSH sans mot de passe pour permettre le bon fonctionnement du cluster.

Par la suite, nous avons exploité ***Hadoop Streaming*** pour analyser des logs Apache, en utilisant des scripts Python Mapper et Reducer. Nous avons détecté des **erreurs 404**, identifié les pages les plus ciblées, et analysé les adresses IP suspectes générant des erreurs ***403 ou 404***. L'utilisation de plusieurs Reducers a permis de paralléliser le traitement et de gérer efficacement de grandes quantités de données.

Enfin, ce TP a mis en évidence l'intérêt de Hadoop dans un contexte de **cybersécurité**, permettant de surveiller et d'analyser des comportements anormaux sur les serveurs web, tels que des scans, des accès non autorisés ou des tentatives d'attaque. Ainsi, Hadoop se révèle être un outil puissant pour la collecte, le traitement et l'analyse de données massives tout en contribuant à la sécurisation des systèmes d'information.