

المدرسة العليا للتكنولوجيا - الداخلة
+ΞΙCΠ +οο.ΧΗΗο+ | +ΞΚΙ:Π:Ξ+ - ΛΛοΧΗο
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

Introduction.....	5
Définition.....	6
Mis en place des Environnements	7
i. Prérequis :	7
ii. Installation de JAVA.....	7
iii. Configurer Java (JAVA_HOME)	8
iv. Téléchargement de Hadoop 3.4.2.....	9
Configuration des fichiers nécessaires	9
a. Les Variables d’Environnements	9
b. Configuration de Hadoop	11
Configuration de SSH (sans mot de passe).....	14
Initialiser HDFS	15
Démarrage de Hadoop	16
start-dfs.sh	16
start-yarn.sh.....	16
Visualisation de Hadoop Interface Web.....	17
A. NameNode.....	17
B. YARN	17
Pratique : Création des répertoires de tests	18
Hadoop & Cybersécurité	18
Introduction.....	18
Téléchargement du fichier logs	19
Création & configuration des fichiers .py.....	19
1. Détecter les erreurs 404.....	19
2. Trouver les pages les plus attaquées	21
3. Identifier les IP suspectes.....	23
Téléchargement des Résultats Interface Web	26
Conclusion Générale.....	28

Table des Figures

Figure 1 : Commande de mise à jour de Ubuntu	7
Figure 2 : Vérification de la version de JAVA installée	7
Figure 3 : Vérification du chemin de JAVA_HOME	8
Figure 4 : Edit le fichier <code>/.bashrc</code>	9
Figure 5 : Déplacement du fichier HADOOP vers le répertoire <code>/usr/local/HADOOP</code>	9
Figure 6 : Configuration des variable d'environnement	10
Figure 7 : Vérification du fichier <code>/.bashrc</code>	10
Figure 8 : Vérification de la version de HADOOP installée	10
Figure 9 : Le dossier HADOOP	11
Figure 10 : Configuration du fichier <code>hadoop-env.sh</code>	11
Figure 11 : Configuration du fichier <code>core-site.xml</code>	12
Figure 12 : Configuration du fichier <code>hdfs-site.xml</code>	12
Figure 13 : Configuration du fichier <code>mapred-site.xml</code>	13
Figure 14 : Configuration du fichier <code>yarn-site.xml</code>	14
Figure 15 : Configuration de SSH (sans mot de passe)	14
Figure 16 : La connexion SSH vers la machine local	15
Figure 17 : Initialisation de système de fichiers HDFS en formatant le NameNode	15
Figure 18 : Démarrage du service HDFS	16
Figure 19 : Démarrage du service yarn	16
Figure 20 : Verification des services démarres	16
Figure 21 : NameNode Interface Web	17
Figure 22 : YARN Interface Web	17
Figure 23 : Création d'un répertoire de travail pour les tests	18
Figure 24 : Vérification Par Interface Web	18
Figure 25 : Script bash pour détecter les erreurs 404 logs du fichier <code>mapper_404.py</code>	19
Figure 26 : Script bash pour détecter les erreurs 404 logs du fichier <code>reducer_404.py</code>	20
Figure 27 : Chargement de fichier logs dans le repertoire de travail <code>/data/logs</code>	20
Figure 28 : Verification avec <code>hdfs dfs -ls /data/logs</code>	20
Figure 29 : Exécution des deux fichiers <code>mapper_404.py</code> & <code>reducer_404.py</code>	21
Figure 30 : Le résultats des exécutions du script de détection des erreurs_404	21
Figure 31 : Script bash pour trouver les pages les plus attaquées : <code>mapper_page_attack.py</code>	22
Figure 32 : Script bash pour trouver les pages les plus attaquées : <code>reducer_page_attack.py</code>	22
Figure 33 : Exécution des deux fichiers <code>mapper__page_attack.py</code> & <code>reducer_page_attack.py</code>	23
Figure 34 : Le résultats des exécutions du script de détection des pages attaquées	23
Figure 35 : Script bash pour identifier ip suspects du fichier logs : <code>mapper_ip.py</code>	23
Figure 36 : Script bash pour identifier ip suspects du fichier logs : <code>reducer_ip.py</code>	24
Figure 37 : Les résultats des exécutions du script d'identification des ip suspects	25
Figure 38 : Verification des nombres de Reducers	25
Figure 39 : Afficher les résultats de tous les Reducers	25
Figure 40 : Les résultats avec l'interface Web	26
Figure 41 : Comment télécharger les résultats avec l'interface Web	26
Figure 42 : Visualisation des résultats	27

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

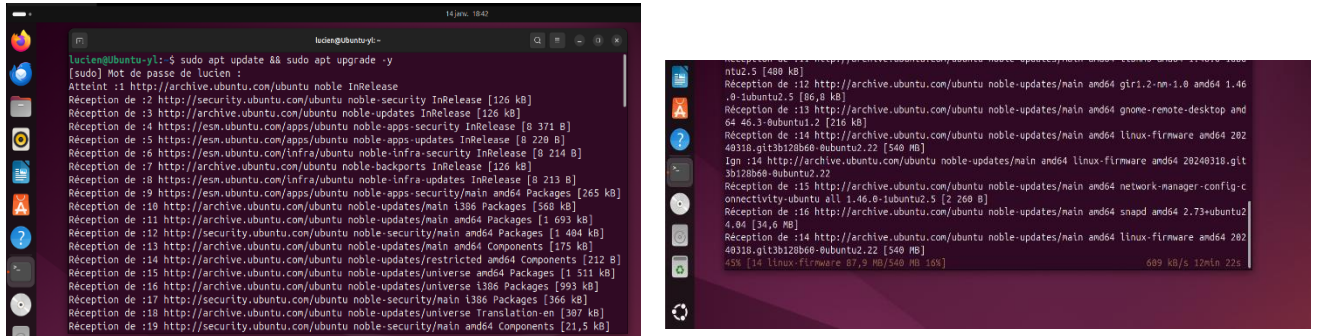
i. Prérequis :

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
```



```
lucien@Ubuntu-yl:~$ sudo apt update && sudo apt upgrade -y
[sudo] Mot de passe de Lucien :
Atteint :1 http://archive.ubuntu.com/ubuntu noble InRelease
Réception de :2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Réception de :3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Réception de :4 https://esm.ubuntu.com/apps/ubuntu noble-apps-security InRelease [8 371 B]
Réception de :5 https://esm.ubuntu.com/apps/ubuntu noble-apps-updates InRelease [8 229 B]
Réception de :6 https://esm.ubuntu.com/infra/ubuntu noble-infra-security InRelease [8 214 B]
Réception de :7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Réception de :8 https://esm.ubuntu.com/infra/ubuntu noble-infra-updates InRelease [8 213 B]
Réception de :9 https://esm.ubuntu.com/apps/ubuntu noble-apps-security/main amd64 Packages [265 kB]
Réception de :10 https://archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [568 kB]
Réception de :11 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1 693 kB]
Réception de :12 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1 484 kB]
Réception de :13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Réception de :14 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Réception de :15 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1 511 kB]
Réception de :16 https://archive.ubuntu.com/ubuntu noble-updates/universe i386 Packages [593 kB]
Réception de :17 https://security.ubuntu.com/ubuntu noble-security/main i386 Packages [366 kB]
Réception de :18 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [387 kB]
Réception de :19 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21,5 kB]
ntu2.5 [488 kB]
Réception de :12 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 gir1.2-nm-1.0 amd64 1.46
.9-ubuntu2.5 [186,8 kB]
Réception de :13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 gnome-remote-desktop and
64 46.3-0ubuntu1.2 [216 kB]
Réception de :14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 linux-firmware amd64 202
40318.git3b120b60-0ubuntu2.22 [540 MB]
Ign :14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 linux-firmware amd64 20240318.git
3b120b60-0ubuntu2.22
Réception de :15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 network-manager-config-c
onnectivity-ubuntu all 1.46.0-1ubuntu2.5 [2 268 B]
Réception de :16 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 snapd amd64 2.73+ubuntu2
4.04 [34,6 MB]
Réception de :14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 linux-firmware amd64 202
40318.git3b120b60-0ubuntu2.22 [540 MB]
[14 Linux-Firmware 87,9 MB/540 MB 16%] 689 kB/s 12min 22s
```

Figure 1 : Commande de mise à jour de Ubuntu

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

```
# sudo apt update
```

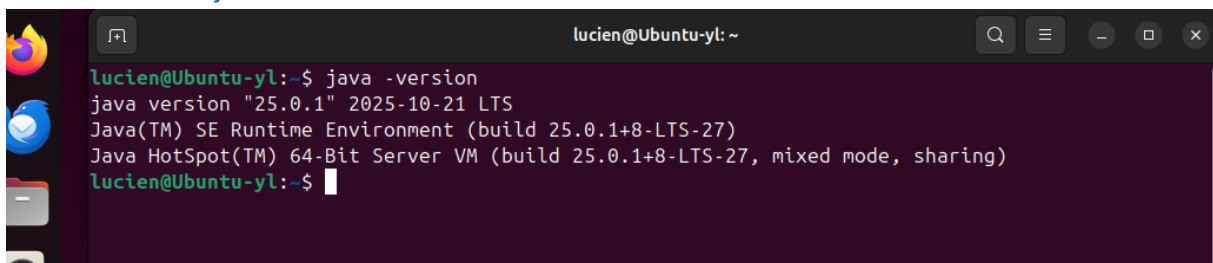
ii. Installation de JAVA

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
```



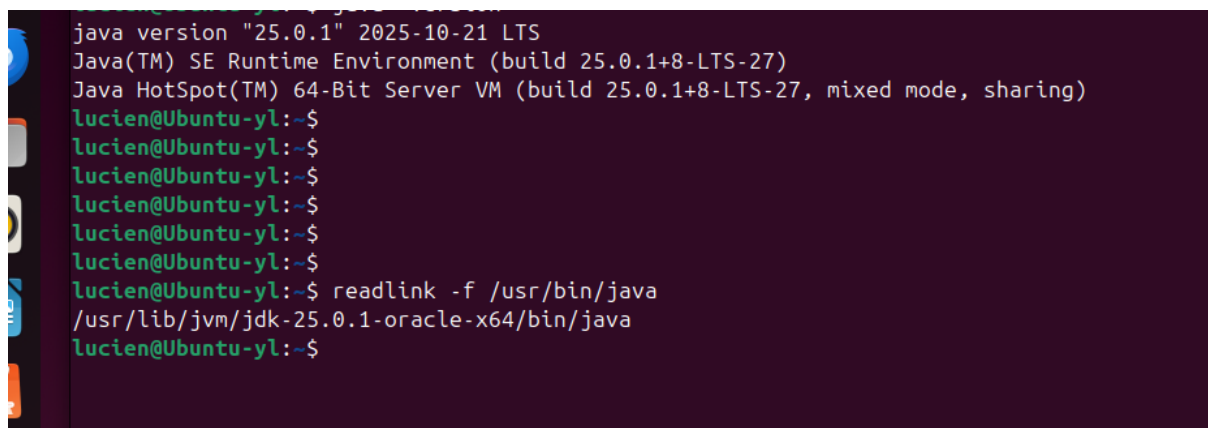
```
lucien@Ubuntu-yl:~$ java -version
java version "25.0.1" 2025-10-21 LTS
Java(TM) SE Runtime Environment (build 25.0.1+8-LTS-27)
Java HotSpot(TM) 64-Bit Server VM (build 25.0.1+8-LTS-27, mixed mode, sharing)
lucien@Ubuntu-yl:~$
```

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
```



```
java version "25.0.1" 2025-10-21 LTS
Java(TM) SE Runtime Environment (build 25.0.1+8-LTS-27)
Java HotSpot(TM) 64-Bit Server VM (build 25.0.1+8-LTS-27, mixed mode, sharing)
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$ readlink -f /usr/bin/java
/usr/lib/jvm/jdk-25.0.1-oracle-x64/bin/java
lucien@Ubuntu-yl:~$
```

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
```

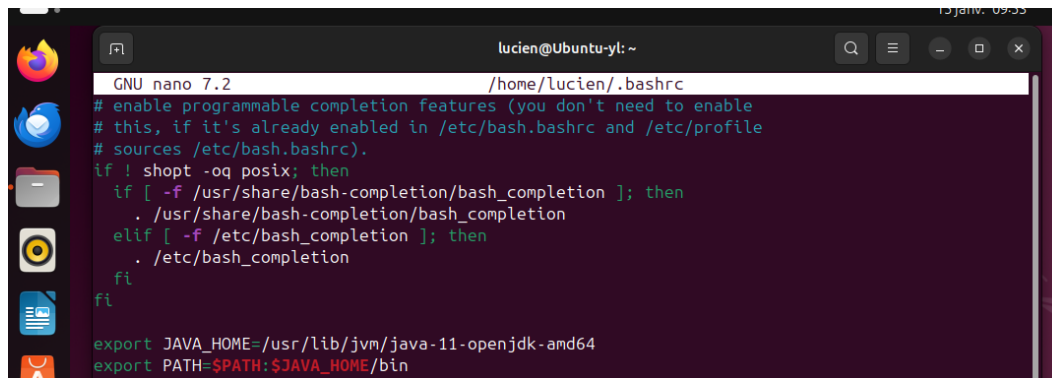
Ce chemin doit-être ajouter dans le fichier `~/.bashrc`.

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`



```
GNU nano 7.2 /home/lucien/.bashrc
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

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.

- o La commande de téléchargement sous Firefox de Ubuntu.

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

- o 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
```

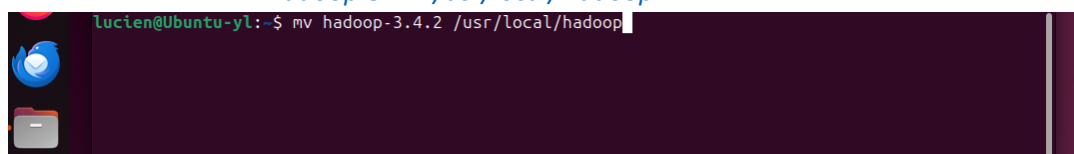
Quand le téléchargement fini, il faut décompresser le dossier que vous venez de télécharger.

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
```



```
lucien@Ubuntu-yl:~$ 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'environnements

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 scripts pour que notre système puisse reconnaître Hadoop.

Éditons à nouveau notre fichier `~/.bashrc` et ajoutons ces scripts à la fin du fichier (à la suite des scripts 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
```

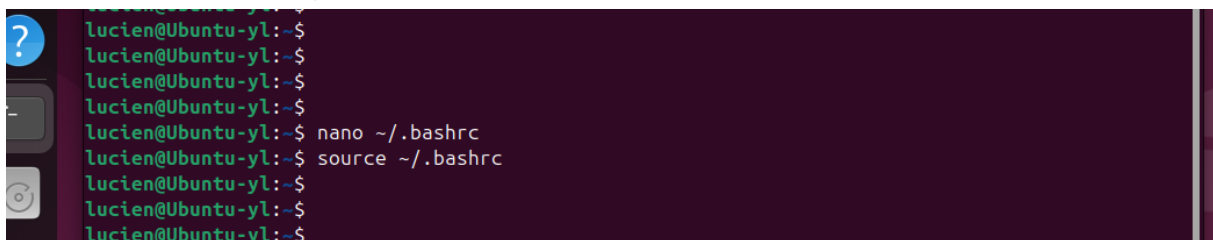


```
. /etc/bash_completion
fi
fi
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

Figure 6 : Configuration des variable d'environnement

On applique cette commande à nouveau :

```
# source ~/.bashrc
```



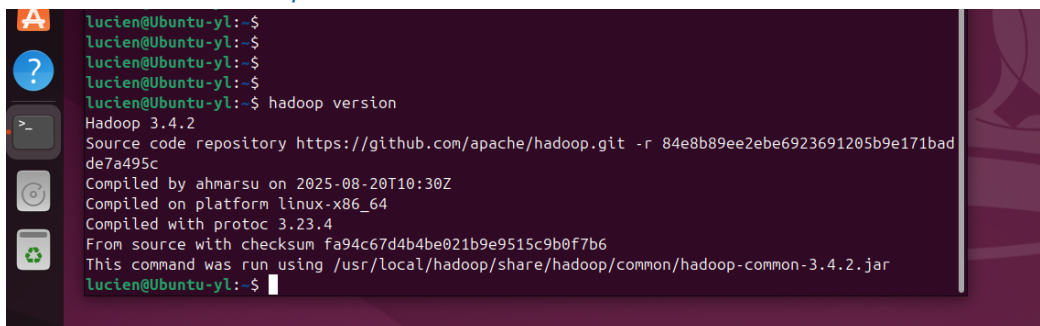
```
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$ nano ~/.bashrc
lucien@Ubuntu-yl:~$ source ~/.bashrc
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
```

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
```



```
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$
lucien@Ubuntu-yl:~$ hadoop version
Hadoop 3.4.2
Source code repository https://github.com/apache/hadoop.git -r 84e8b89ee2ebe6923691205b9e171bad
de7a495c
Compiled by ahmarsu on 2025-08-20T10:30Z
Compiled on platform linux-x86_64
Compiled with protoc 3.23.4
From source with checksum fa94c67d4b4be021b9e9515c9b0f7b6
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.4.2.jar
lucien@Ubuntu-yl:~$
```

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 trouvent dans le dossier Hadoop. Chaque fichier joue un rôle crucial pour le bon fonctionnement de Hadoop. Nous allons dans les dossiers de configuration en utilisant le chemin de Hadoop (HADOOP_HOME) :

```
# cd $HADOOP_HOME/etc/hadoop
```

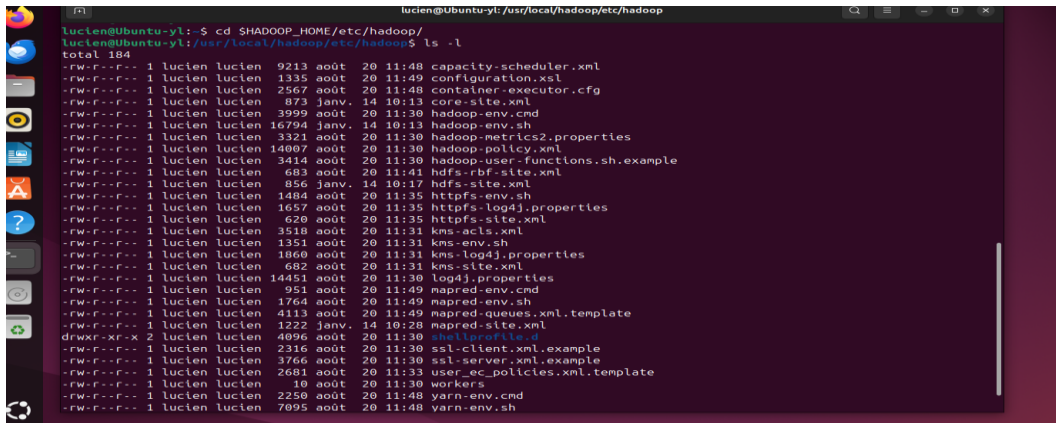


Figure 9 : Le dossier HADOOP

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

1. `hadoop-env.sh`

Notre premier fichier est : **`hadoop-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
```

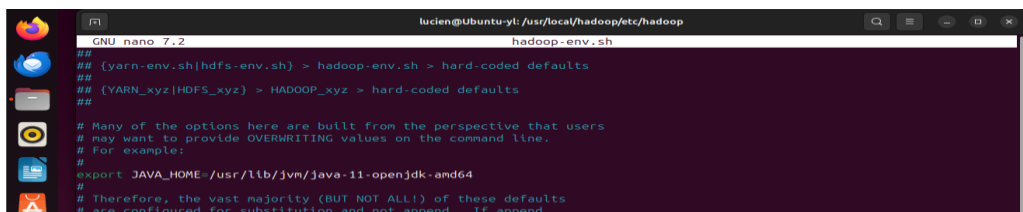


Figure 10 : Configuration du fichier `hadoop-env.sh`

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>
```

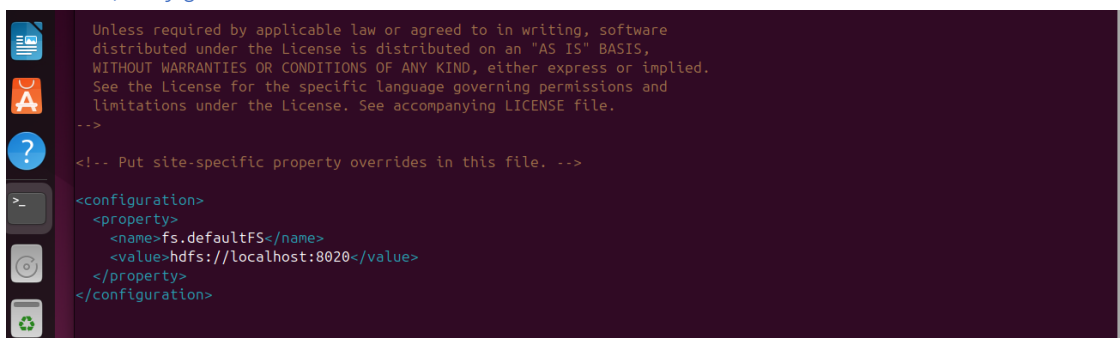


Figure 11 : Configuration du fichier core-site.xml

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>
```

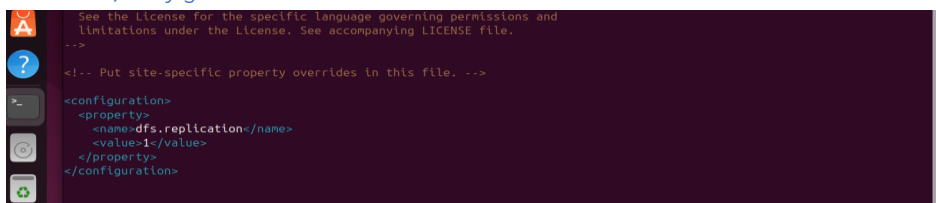


Figure 12 : Configuration du fichier hdfs-site.xml

4. mapred-site.xml

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

Dans certains cas ce dossier **mapred-site.xml** n'existe pas dans le dossier Hadoop. Donc créer ce dossier il faudra copier le contenu 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>
```

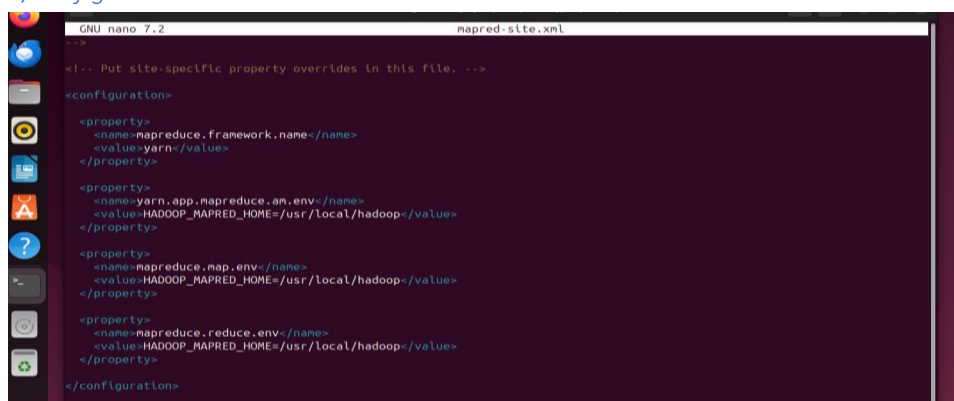


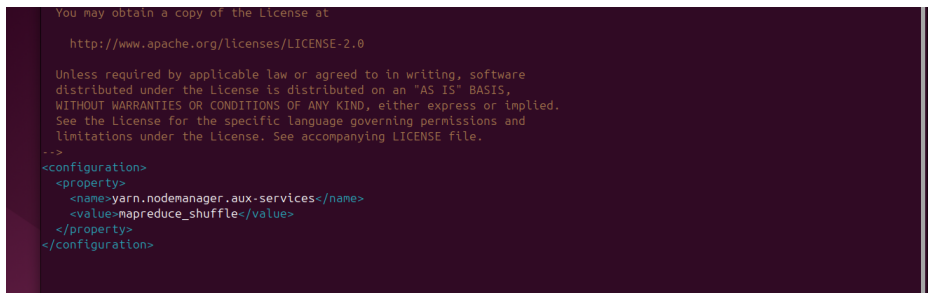
Figure 13 : Configuration du fichier `mapred-site.xml`

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>
```



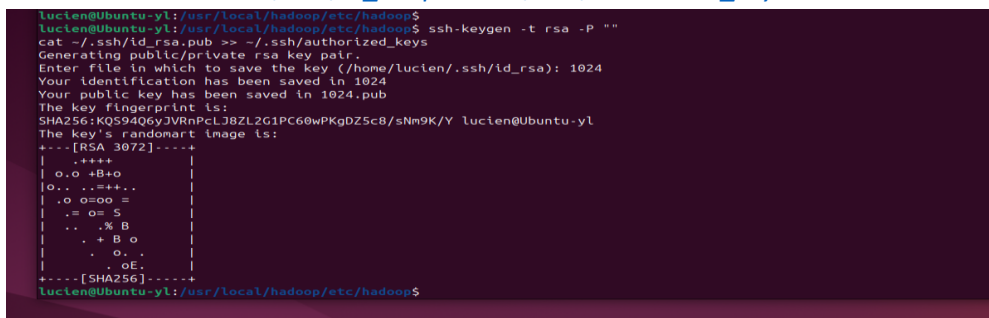
```
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->
<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
```



```
lucien@Ubuntu-yl:~/usr/local/hadoop/etc/hadoop$
lucien@Ubuntu-yl:~/usr/local/hadoop/etc/hadoop$ ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
Generating public/private rsa key pair.
Enter file in which to save the key (/home/lucien/.ssh/id_rsa): 1024
Your identification has been saved in 1024
Your public key has been saved in 1024.pub
The key fingerprint is:
SHA256:KQ594Q6yJVRnPcLJ8ZL2G1PC60wPKgDZ5c8/sNm9K/Y lucien@Ubuntu-yl
The key's randomart image is:
+--[RSA 3072]-----+
| .+++          |
| 0.o +B+0     |
| 0.. .++++.   |
| .o o=oo =    |
| .o = 5       |
| .. .% B      |
| . + B o      |
| . 0. .       |
| . oE.        |
+---[SHA256]-----+
lucien@Ubuntu-yl:~/usr/local/hadoop/etc/hadoop$
```

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

La commande pour tester la connexion SSH vers la machine locale

```
# ssh localhost
```

```
Lucien@Ubuntu-yl:~/usr/local/hadoop/etc/hadoop$
Lucien@Ubuntu-yl:~/usr/local/hadoop/etc/hadoop$ ssh localhost
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

La maintenance de sécurité étendue pour Applications est activée.

55 mises à jour peuvent être appliquées immédiatement.
44 de ces mises à jour sont des mises à jour de sécurité.
Pour afficher ces mises à jour supplémentaires, exécuter : apt list --upgradable

Last login: Sun Jan  4 01:20:05 2026 from 127.0.0.1
Lucien@Ubuntu-yl:~$
```

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
```

```
Lucien@Ubuntu-yl:~$
Lucien@Ubuntu-yl:~$
Lucien@Ubuntu-yl:~$ hdfs namenode -format
2026-01-15 11:57:28,343 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = Ubuntu-yl/192.168.253.154
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 3.4.2
STARTUP_MSG: classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/hadoop-shaded-guava-1.4.0.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-redis-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-socks-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/kerb-core-2.0.3.jar:/usr/local/hadoop/share/hadoop/p/common/lib/commons-net3-3.9.0.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-collections4-4.4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-lang3-3.17.0.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-native-epoll-4.1.118.Final-linux-riscv64.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-io-9.4.57.v20241219.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-http-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/bcprov-jdk18on-1.78.1.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-native-epoll-4.1.118.Final-linux-aarch_64.jar:/usr/local/hadoop/share/hadoop/common/lib/kerb-crypto-2.0.3.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-xml-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/httpcore-4.4.13.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-resolver-dns-native-macos-4.1.118.Final-osx-aarch_64.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-resolver-dns-classes-macos-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-handler-proxy-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-resolver-dns-native-macos-4.1.118.Final-osx-x86_64.jar:/usr/local/hadoop/share/hadoop/common/lib/nimbus-jose-jwt-9.37.2.jar:/usr/local/hadoop/share/hadoop/common/lib/curator-recipes-5.2.0.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-http-4.1.118.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-classes-kqueue-4.1.118.F
2026-01-15 11:57:32,535 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry time is 600000 millis
2026-01-15 11:57:32,547 INFO util.GSet: Computing capacity for map NameNodeRetryCache
2026-01-15 11:57:32,548 INFO util.GSet: VM type = 64-bit
2026-01-15 11:57:32,552 INFO util.GSet: 0.0299999999329447746% max memory 1.4 GB = 425.2 KB
2026-01-15 11:57:32,554 INFO util.GSet: capacity = 2^16 = 65536 entries
2026-01-15 11:57:32,958 INFO namenode.FSImage: Allocated new BlockPoolId: BP-268435700-192.168.253.154-1768474652915
2026-01-15 11:57:33,050 INFO common.Storage: Storage directory /tmp/hadoop-lucien/dfs/name has been successfully formatted.
2026-01-15 11:57:33,244 INFO namenode.FSImageFormatProtobuf: Saving image file /tmp/hadoop-lucien/dfs/name/current/fsimage.ckpt_000000000000000000 using no compression
2026-01-15 11:57:33,762 INFO namenode.FSImageFormatProtobuf: Image file /tmp/hadoop-lucien/dfs/name/current/fsimage.ckpt_000000000000000000 of size 401 bytes saved in 0 seconds .
2026-01-15 11:57:33,830 INFO namenode.NNSStorageRetentionManager: Going to retain 1 images with txid >= 0
2026-01-15 11:57:33,865 INFO blockmanagement.DatanodeManager: Slow peers collection thread shutdown
2026-01-15 11:57:33,921 INFO namenode.FSNamesystem: Stopping services started for active state
2026-01-15 11:57:33,922 INFO namenode.FSNamesystem: Stopping services started for standby state
2026-01-15 11:57:33,947 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2026-01-15 11:57:33,955 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at Ubuntu-yl/192.168.253.154
*****/
Lucien@Ubuntu-yl:~$
```

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

Hadoop est bien configuré et prête à être démarré.

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.

```
Lucien@Ubuntu-yl: ~$
Lucien@Ubuntu-yl: ~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Ubuntu-yl]
2026-01-15 12:05:42,993 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j
ava classes where applicable
Lucien@Ubuntu-yl: ~$ jps
5136 SecondaryNameNode
4721 NameNode
4897 DataNode
5294 Jps
Lucien@Ubuntu-yl: ~$
```

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.

```
Lucien@Ubuntu-yl: ~$
Lucien@Ubuntu-yl: ~$
Lucien@Ubuntu-yl: ~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
Lucien@Ubuntu-yl: ~$ jps
5136 SecondaryNameNode
4721 NameNode
4897 DataNode
5397 ResourceManager
5915 Jps
5550 NodeManager
Lucien@Ubuntu-yl: ~$
```

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.

```
Lucien@Ubuntu-yl: ~$
Lucien@Ubuntu-yl: ~$
Lucien@Ubuntu-yl: ~$ jps
5136 SecondaryNameNode
4721 NameNode
4897 DataNode
5397 ResourceManager
5550 NodeManager
5951 Jps
Lucien@Ubuntu-yl: ~$
```

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>

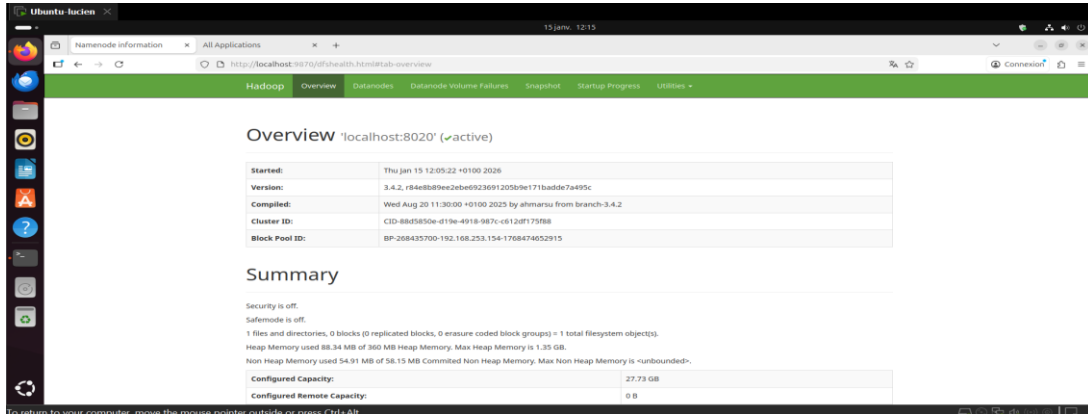


Figure 21 : NameNode Interface Web

B. YARN

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

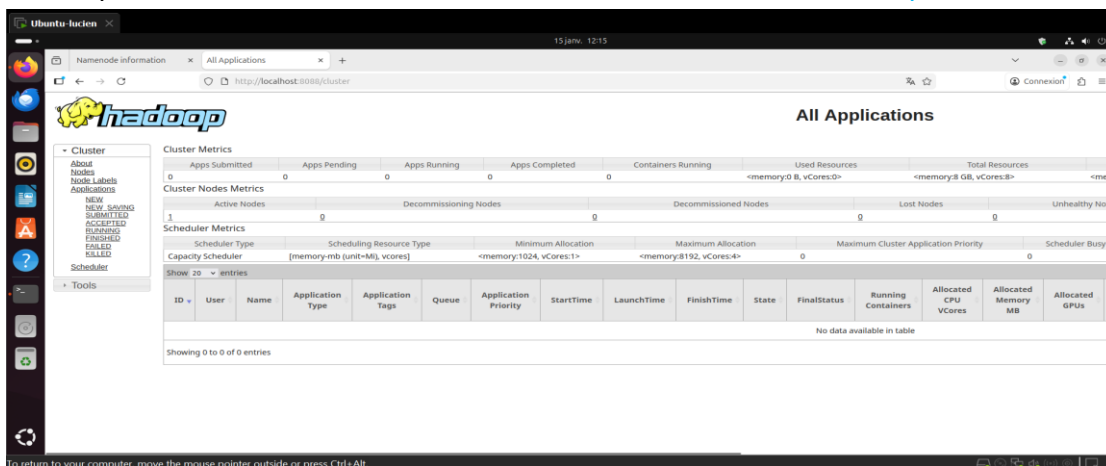


Figure 22 : YARN Interface Web

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 créé.

```
# hdfs dfs -ls /
```

Terminal :

```
Lucien@Ubuntu-yl:~$  
Lucien@Ubuntu-yl:~$ hdfs dfs -mkdir /test  
2026-01-15 12:30:49,140 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j  
ava classes where applicable  
Lucien@Ubuntu-yl:~$ hdfs dfs -ls /  
2026-01-15 12:31:08,050 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-j  
ava classes where applicable  
Found 1 items  
drwxr-xr-x - lucien supergroup          0 2026-01-15 12:30 /test  
Lucien@Ubuntu-yl:~$
```

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

Interface Web :

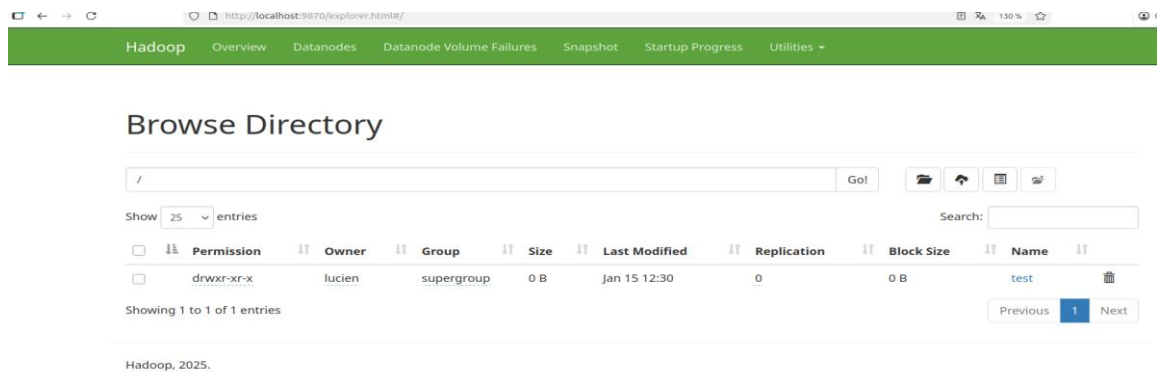


Figure 24 : Vérification Par Interface Web

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

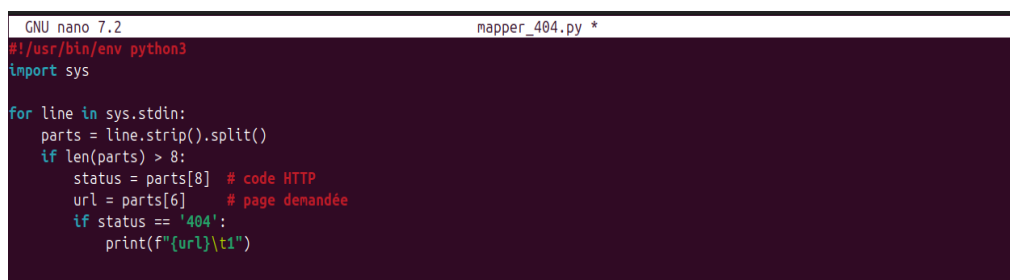
1. Détecter les erreurs 404

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")
```



```
GNU nano 7.2 mapper_404.py *
#!/usr/bin/env python3
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

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}")
```

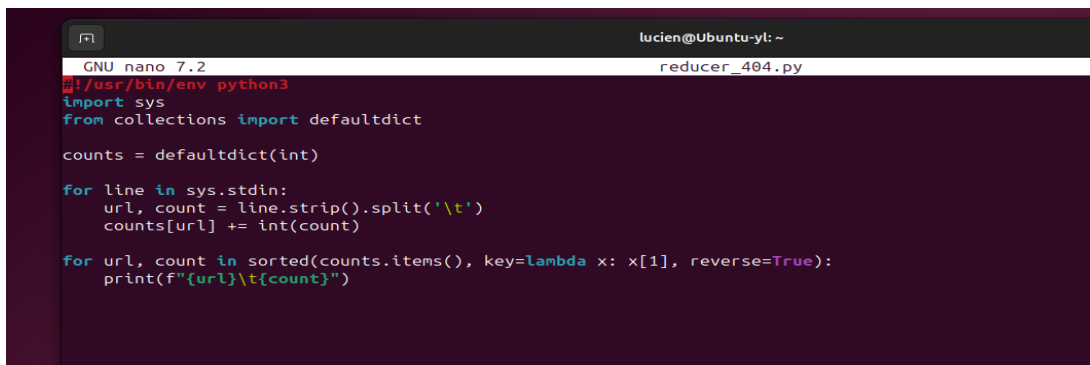


Figure 26 : Script bash pour détecter les erreurs 404 logs du fichier reducer_404.py

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

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

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

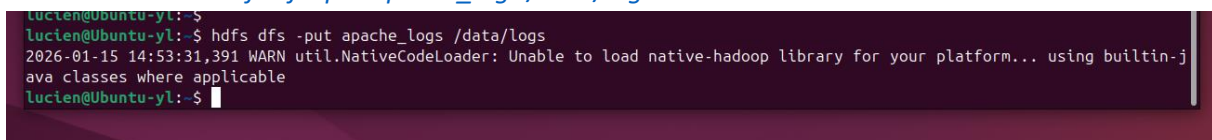


Figure 27 : Chargement de fichier logs dans le repertoire de travail /data/logs

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

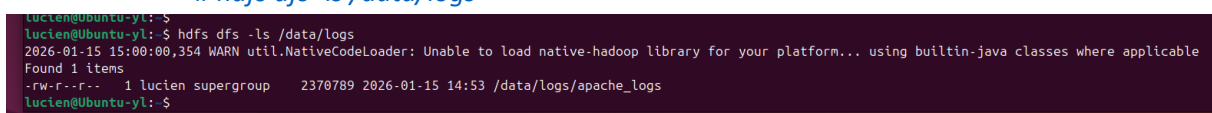


Figure 28 : Verification avec hdfs dfs -ls /data/logs



```
GNU nano 7.2 lucien@Ubuntu-yl: ~
#!/usr/bin/env python3
import sys

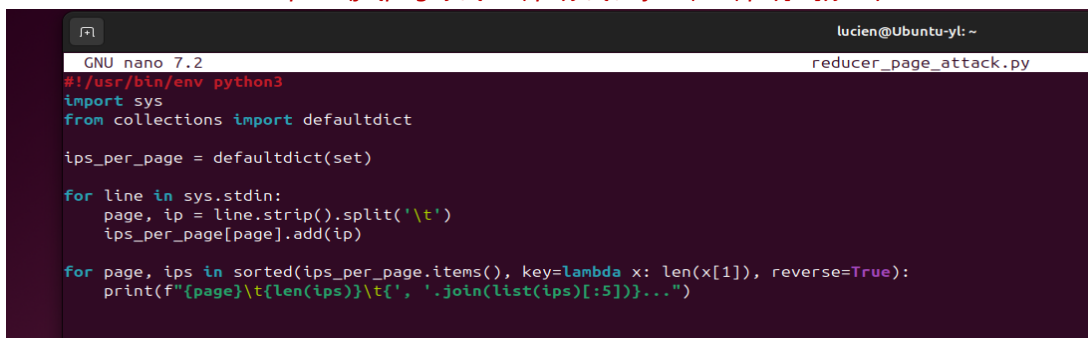
for line in sys.stdin:
    parts = line.strip().split()
    if len(parts) > 8:
        ip = parts[0]
        page = parts[6]
        print(f"{page}\t{ip}")
```

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])}...")
```



```
GNU nano 7.2 lucien@Ubuntu-yl: ~
#!/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
```

```

lucien@ubuntu-yl: ~$
lucien@ubuntu-yl: ~$ 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
2026-01-15 15:46:20,717 WARN streaming.StreamJob: file option is deprecated, please use generic option -files instead.
2026-01-15 15:46:21,523 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
packageJobJar: [mapper_page_attack.py, reducer_page_attack.py, /tmp/hadoop-unjar17028994841573155304/] [] /tmp/streamjob1360755971834996393.jar tmpDir=null
2026-01-15 15:46:29,279 INFO client.DefaultHadoopFallbackProxyProvider: Connecting to ResourceManager at 0.0.0.0:8032
2026-01-15 15:46:35,965 INFO client.DefaultHadoopFallbackProxyProvider: Connecting to ResourceManager at 0.0.0.0:8032
2026-01-15 15:46:39,638 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/lucien/.staging/job_1768475214390_0004
2026-01-15 15:46:44,262 INFO mapred.fileoutputformat: Total input files to process : 1
2026-01-15 15:46:45,837 INFO mapreduce.JobSubmitter: number of splits:2
2026-01-15 15:46:46,691 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1768475214390_0004
2026-01-15 15:46:46,692 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-01-15 15:46:47,944 INFO conf.Configuration: resource-types.xml not found
2026-01-15 15:46:47,944 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2026-01-15 15:46:50,534 INFO Impl.YarnClientImpl: Submitted application application_1768475214390_0004
2026-01-15 15:46:50,758 INFO mapreduce.Job: The url to track the job: http://Ubuntu-yl:8088/proxy/application_1768475214390_0004/
2026-01-15 15:46:50,770 INFO mapreduce.Job: Running Job: Job_1768475214390_0004
2026-01-15 15:47:31,167 INFO mapreduce.Job: Job job_1768475214390_0004 running in uber mode : false
2026-01-15 15:47:31,174 INFO mapreduce.Job: map 0% reduce 0%
2026-01-15 15:48:38,806 INFO mapreduce.Job: map 100% reduce 100%
2026-01-15 15:48:38,806 INFO mapreduce.Job: Job job_1768475214390_0004 completed successfully
2026-01-15 15:48:45,410 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=493213
FILE: Number of bytes written=1926159

```

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-*
```

```

lucien@ubuntu-yl: ~$
lucien@ubuntu-yl: ~$ hdfs dfs -cat /output/page_attack/part-*
2026-01-15 15:54:15,019 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
/favicon.ico 683 185.2.138.125, 213.193.72.155, 84.52.150.17, 112.110.247.250, 109.195.177.171...
/style2.css 516 185.2.138.125, 213.193.72.155, 84.52.150.17, 112.110.247.250, 109.195.177.171...
/reset.css 509 185.2.138.125, 213.193.72.155, 84.52.150.17, 109.195.177.171, 81.61.214.106...
/images/Jordan-80.png 588 185.2.138.125, 213.193.72.155, 84.52.150.17, 109.195.177.171, 81.61.214.106...
/images/web2009/banner.png 494 185.2.138.125, 213.193.72.155, 84.52.150.17, 109.195.177.171, 81.61.214.106...
/projects/xdotool/ 187 185.2.138.125, 213.193.72.155, 14.141.56.98, 122.146.177.66, 84.52.150.17...
/ 153 118.170.125.5, 184.154.15.210, 187.60.96.7, 88.0.30.135, 66.187.233.202...
/projects/xdotool/xdotool.shtml 136 71.59.29.242, 79.164.36.106, 41.251.155.27, 213.124.13.242, 84.52.150.17...
/robots.txt 43 180.76.5.114, 207.241.237.224, 216.107.155.114, 199.30.20.11, 180.76.6.65...
/articles/dynamic-dns-with-dhcp/ 118 184.60.23.120, 114.69.226.80, 83.175.127.2, 81.61.214.106, 72.215.249.101...
/presentations/logstash-scale11x/images/ahhh__rage_face_by_samusmxx-d5g5zap.png 114 14.1.42.114, 99.179.126.76, 200.50.183.132, 71.171.122.192, 193.65.112.2...
/images/googleusercontent.com 100 202.134.13.137, 37.241.118.235, 106.66.30.77, 112.110.247.244, 200.28.244.131...
/articles/ssh-security/ 50 110.170.125.5, 65.55.213.73, 193.40.6.84, 222.77.201.107, 180.153.236.122...
/blog/geekery/ssl-latency.html 50 112.202.18.45, 143.233.204.28, 117.195.177.223, 193.252.149.222, 75.67.42.229...
/presentations/logstash-puppetconf-2012/ 48 182.253.73.51, 204.93.54.177, 85.115.58.180, 83.10.253.20, 101.119.18.35...
/7flavrs20 43 180.76.5.114, 74.125.19.82, 54.219.117.220, 74.125.176.146, 74.125.40.19...
/presentations/puppet-at-loggly/puppet-at-loggly.pdf.html 35 182.253.73.51, 204.93.54.177, 87.169.99.232, 204.93.54.173, 108.184.124.170...
/blog/geekery/xvfb-firefox.html 34 176.14.63.246, 195.37.190.67, 208.43.251.180, 91.177.205.119, 61.246.186.198...
/blog/geekery/installing-windows-8-consumer-preview.html 30 65.55.213.73, 5.255.72.160, 184.154.15.210, 187.60.96.7, 74.221.220.196...
/articles/ppp-over-ssh/ 28 23.105.131.2, 65.55.213.73, 80.118.116.26, 193.33.2.113, 54.211.215.107...
/presentations/logstash-puppetconf-2012/images/kibana-logstash-downloads.png 28 85.115.58.180, 101.119.18.35, 61.140.183.41, 85.43.182.12, 82.98.148.169...
presentations/logstash-puppetconf-2012/images/office-space-printer-beat-down-gif.gif 27 74.208.180.23, 85.115.58.180, 101.119.18.35, 61.140.183.41, 85.43.182.12
...

```

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")

```

```

lucien@ubuntu-yl: ~$ nano mapper_ip.py
GNU nano 7.2 mapper_ip.py
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    ip = line.split()[0]
    print(f"{ip}\t1")

```

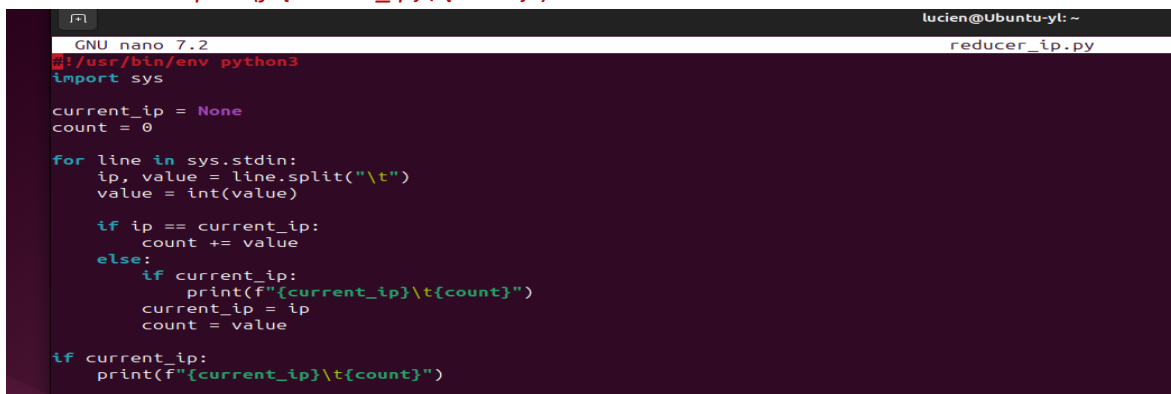
Figure 35 : Script bash pour identifier ip suspects du fichier logs : mapper_ip.py

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 reducer_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}")
```



```
GNU nano 7.2                                lucien@Ubuntu-yl: ~
#!/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.

```

lucien@ubuntu-yl: ~$
lucien@ubuntu-yl: ~$ 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
2026-01-15 16:17:19,750 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
2026-01-15 16:17:20,858 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2026-01-15 16:17:27,955 INFO client.DefaultNoHARMFalloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2026-01-15 16:17:29,646 INFO client.DefaultNoHARMFalloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2026-01-15 16:17:32,359 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/lucien/.staging/job_1768475214390_0005
2026-01-15 16:17:36,292 INFO mapreduce.FileInputFormat: Total input files to process : 1
2026-01-15 16:17:36,622 INFO mapreduce.JobSubmitter: number of splits:2
2026-01-15 16:17:38,164 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1768475214390_0005
2026-01-15 16:17:38,164 INFO mapreduce.JobSubmitter: Executing with tokens: []
2026-01-15 16:17:38,893 INFO conf.Configuration: resource-types.xml not found
2026-01-15 16:17:38,895 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2026-01-15 16:17:42,082 INFO impl.YarnClientImpl: Submitted application application_1768475214390_0005
2026-01-15 16:17:42,255 INFO mapreduce.Job: The url to track the job: http://lucien-yl:8088/proxy/application_1768475214390_0005/
2026-01-15 16:17:42,264 INFO mapreduce.Job: Running job: job_1768475214390_0005
2026-01-15 16:18:26,845 INFO mapreduce.Job: Job job_1768475214390_0005 running in uber mode : false
2026-01-15 16:18:26,859 INFO mapreduce.Job: map 0% reduce 0%
2026-01-15 16:19:15,585 INFO mapreduce.Job: map 50% reduce 0%
2026-01-15 16:19:17,939 INFO mapreduce.Job: map 100% reduce 0%
2026-01-15 16:19:51,321 INFO mapreduce.Job: map 100% reduce 33%
2026-01-15 16:19:53,408 INFO mapreduce.Job: map 100% reduce 100%
2026-01-15 16:19:56,572 INFO mapreduce.Job: Job job_1768475214390_0005 completed successfully
2026-01-15 16:19:58,329 INFO mapreduce.Job: Counters: 54
File System Counters

```

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
```

```

lucien@ubuntu-yl: ~$
lucien@ubuntu-yl: ~$ hdfs dfs -ls /output/ip_activity
2026-01-15 16:21:49,752 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
-rw-r--r-- 1 lucien supergroup 0 2026-01-15 16:19 /output/ip_activity/_SUCCESS
-rw-r--r-- 1 lucien supergroup 9751 2026-01-15 16:19 /output/ip_activity/part-00000
-rw-r--r-- 1 lucien supergroup 9217 2026-01-15 16:19 /output/ip_activity/part-00001
-rw-r--r-- 1 lucien supergroup 9339 2026-01-15 16:19 /output/ip_activity/part-00002
lucien@ubuntu-yl: ~$

```

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-*
```

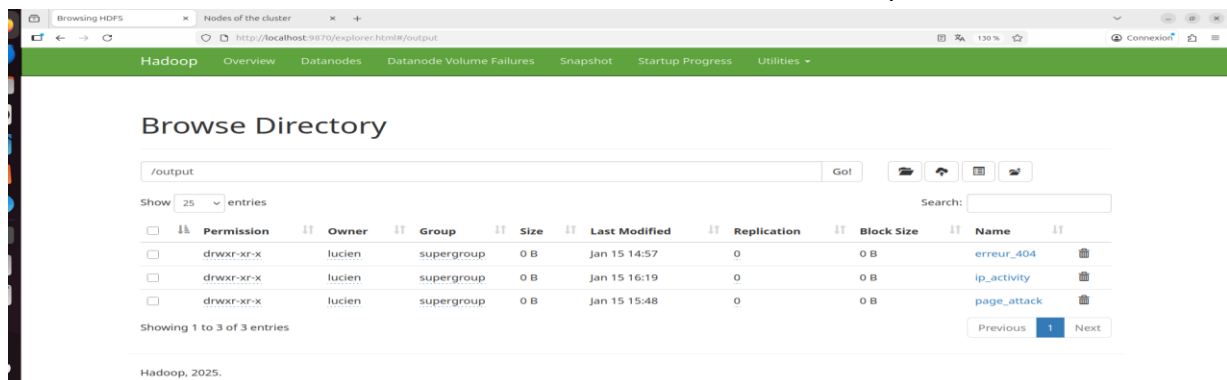
```

lucien@ubuntu-yl: ~$
lucien@ubuntu-yl: ~$ hdfs dfs -cat /output/ip_activity/part-*
2026-01-15 16:28:43,346 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
100.2.4.116 6
101.199.108.50 3
101.226.168.198 1
103.247.192.5 1
106.187.98.179 1
106.36.113.138 1
106.66.30.77 1
107.170.40.198 4
107.170.40.201 4
107.170.40.204 7
107.22.42.225 1
108.15.20.23 6
108.171.116.194 65
108.178.13.202 2
108.77.164.97 1
108.91.82.251 7
109.109.46.88 1
109.163.234.10 1
109.163.234.2 1
109.189.132.223 6
109.195.177.171 6
109.231.204.82 6
110.143.13.225 2

```

Figure 39 : Afficher les résultats de tous les Reducers

Vérification des résultats à travers l'interface Web de Hadoop



ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1768475214390_0005	lucien	streamjob18071930482685816624.jar	MAPREDUCE		root.default	0	Thu Jan 15 16:17:40 +0100 2026	Thu Jan 15 16:17:44 +0100 2026	Thu Jan 15 16:19:55 +0100 2026
application_1768475214390_0004	lucien	streamjob1360755971834996393.jar	MAPREDUCE		root.default	0	Thu Jan 15 15:46:49 +0100 2026	Thu Jan 15 15:46:53 +0100 2026	Thu Jan 15 15:48:41 +0100 2026
application_1768475214390_0003	lucien	streamjob16154805479815719197.jar	MAPREDUCE		root.default	0	Thu Jan 15 14:56:30 +0100 2026	Thu Jan 15 14:56:42 +0100 2026	Thu Jan 15 14:57:42 +0100 2026

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.

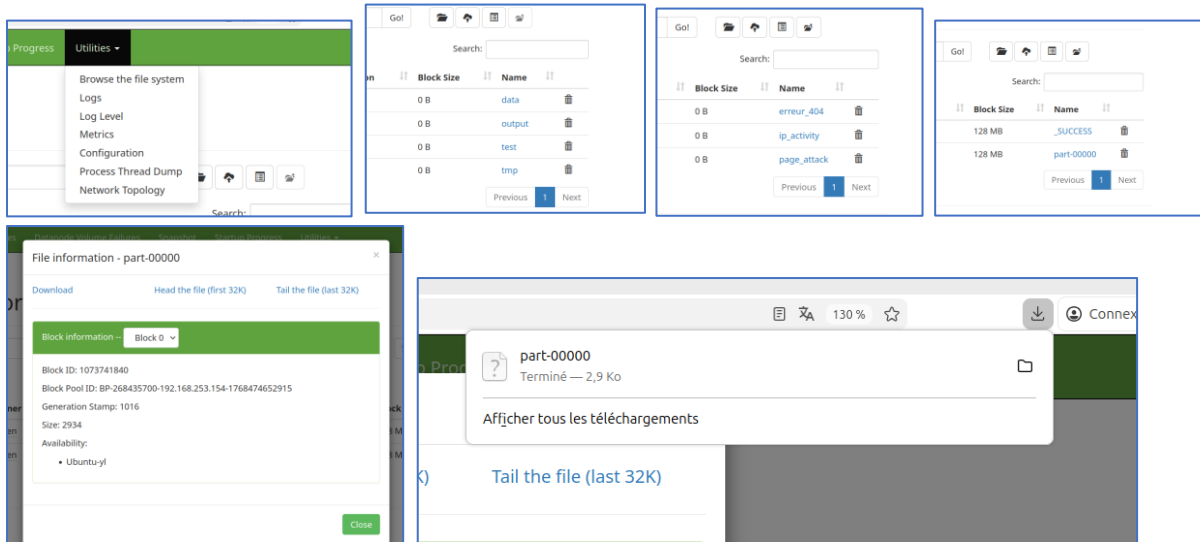


Figure 41 : Comment télécharger les résultats avec l'interface Web

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.