

Compte Rendu TP2

Programmation avec l'API HDFS

Nom et Prénom : AMRANI Hamza

Filière : IDF

I. Démarrer le Cluster Hadoop

Pour le démarrage du cluster Hadoop, veuillez vous référer au TP1 (Section I - Installation Cluster Hadoop).

Les étapes principales sont :

- Démarrer les conteneurs Docker
- Accéder au conteneur master
- Lancer Hadoop et YARN avec `./start-hadoop.sh`
- Vérifier les interfaces web (localhost :9870 et localhost :8088)
- Vérifier les processus avec la commande `jps`

N.B :

Voir le compte rendu du TP1 pour les détails complets de cette partie.

II. Programmation avec l'API HDFS

1. Installation de l'environnement de développement

Sur ma machine locale j'ai :

- Installer le JDK Java
- Vérifier l'installation avec `java -version`
- Installer un IDE (j'ai utilisé IntelliJ IDEA pour le développement du projet Maven.)
- Créer une librairie Hadoop_lib avec les JAR :
 - `hadoop-hdfs-3.2.0.jar`
 - `hadoop-common-3.2.0.jar`
 - `hadoop-mapreduce-client-core-3.2.0.jar`
- Créer un projet BigData avec JRE 1.8
- Créer le package : `ma.ensias`

```
root@hadoop-master: ~  
root@hadoop-master:~# java -version  
openjdk version "1.8.0_362"  
OpenJDK Runtime Environment (build 1.8.0_362-8u372-ga~us1-0ubuntu1~18.04-b09)  
OpenJDK 64-Bit Server VM (build 25.362-b09, mixed mode)
```

N.B :

Pour chaque exemple Java dans ce TP, nous suivrons les mêmes étapes de compilation et déploiement :

1. Compiler le projet avec Maven :

```
mvn clean package
```

2. Copier le JAR généré vers le dossier partagé :

```
cp target/nom-du-jar.jar ~/documents/Big\ Data/  
hadoop_project/
```

3. Exécuter le JAR dans le conteneur Hadoop

Ces étapes seront répétées pour tous les exemples suivants.

2. Premier exemple : HadoopFileStatus

Projet 1 : hadoop_java_E1 (sans paramètres)

Compilation et copie du JAR :

```
mvn clean package  
cp target/hadoop-hdfs-HadoopFileStatus.jar ~/documents/Big\ Data/  
/hadoop_project/
```

```
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_java_E1  
$ mvn clean package  
[INFO] Scanning for projects...  
[INFO]  
[INFO] -----< ma.ensias:hadoop_java_E1 >-----  
[INFO] Building hadoop_java_E1 1.0-SNAPSHOT  
[INFO] from pom.xml  
[INFO] -----[ jar ]-----  
[INFO]  
[INFO] BUILD SUCCESS  
[INFO]  
[INFO] Total time: 4.741 s  
[INFO] Finished at: 2025-10-11T06:50:43+02:00  
[INFO]  
hamza@Hamza-amr MINGW64 ~/Documents/Big Data/hadoop_project/hadoop_java_E1  
$ cp target/hadoop-hdfs-HadoopFileStatus.jar ~/documents/Big\ Data/hadoop_project/
```

Exécuter le JAR sans paramètres :

```
hadoop jar /shared_volume/hadoop-hdfs-HadoopFileStatus.jar
```

```
root@hadoop-master: ~  
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-HadoopFileStatus.jar  
2549 octets  
File Name: purchases.txt  
File Size: 2549  
File Replication: 2  
File Block Size: 134217728
```

Projet 2 : hadoop_java_E2 (avec paramètres)

Modifier la classe pour accepter des paramètres en ligne de commande, puis compiler et exécuter :

```
hadoop jar /shared_volume/hadoop-hdfs-HadoopFileStatusPar.jar /
  user/root/input/purchases.txt achats.txt
hdfs dfs -ls /user/root/input/
```

```
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-HadoopFileStatusPar.jar /user/root/input/purchases.txt achats.txt
2549 octets
File Name: purchases.txt
File Size: 2549
File Replication: 2
File Block Size: 134217728
? Fichier renommé de purchases.txt vers achats.txt
root@hadoop-master:~# hdfs dfs -ls /user/root/input/
Found 1 items
-rw-r--r--  2 root supergroup      2549 2025-10-11 05:04 /user/root/input/achats.txt
```

3. Info d'un fichier sur HDFS : HDFSInfo**Projet 3 : hadoop_java_E3**

Cette classe permet de vérifier l'existence d'un fichier et afficher ses informations (taille, taille des blocs, réplication).

Exécuter le JAR en passant le chemin du fichier en paramètre :

```
hadoop jar /shared_volume/hadoop-hdfs-HDFSInfo.jar /user/root/
  input/purchases.txt
```

```
root@hadoop-master: ~
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-HDFSInfo.jar /user/root/input/purchases.txt
2549 octets
Informations sur les blocs:
Block size: 134217728
Replication: 2
```

Le programme affiche :

- La taille du fichier en octets
- Les informations sur les blocs (Block Size, Replication)

4. Lire un fichier sur HDFS : ReadHDFS**Projet 4 : hadoop_java_E4**

Cette classe permet de lire et afficher le contenu complet d'un fichier stocké dans HDFS ligne par ligne.

Exécuter le JAR :

```
hadoop jar /shared_volume/hadoop-hdfs-ReadHDFS.jar /user/root/
  input/purchases.txt
```

```
root@hadoop-master: ~
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-ReadHDFS.jar /user/root/input/purchases.txt
2012-01-01 09:00 San Jose Men's Clothing 214.05 Amex
2012-01-01 09:00 Fort Worth Women's Clothing 153.57 Visa
2012-01-01 09:00 San Diego Music 66.08 Cash
2012-01-01 09:00 Pittsburgh Pet Supplies 493.51 Discover
2012-01-01 09:00 Omaha Children's clothing 235.63 MasterCard
2012-01-01 09:00 Stockton Men's Clothing 247.18 MasterCard
2012-01-01 09:00 Austin Cameras 379.6 Visa
```

Le programme lit le fichier et affiche toutes les lignes.

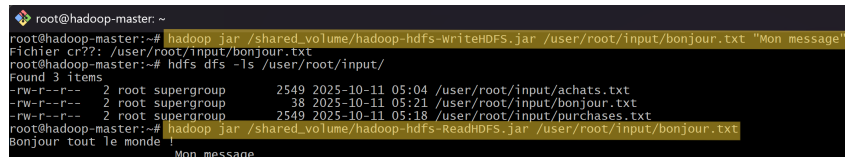
5. Écrire un fichier sur HDFS : WriteHDFS

Projet 5 : `hadoop_java_E5`

Cette classe permet de créer un nouveau fichier sur HDFS et d'y écrire du contenu.

Exécuter le JAR avec le chemin du nouveau fichier et un message :

```
hadoop jar /shared_volume/hadoop-hdfs-WriteHDFS.jar /user/root/
input/bonjour.txt "Mon message"
```



```
root@hadoop-master: ~
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-WriteHDFS.jar /user/root/input/bonjour.txt "Mon message"
Fichier cr???: /user/root/input/bonjour.txt
root@hadoop-master:~# hdfs dfs -ls /user/root/input/
Found 3 items
-rw-r--r--  2 root supergroup      2549 2025-10-11 05:04 /user/root/input/achats.txt
-rw-r--r--  2 root supergroup       38 2025-10-11 05:21 /user/root/input/bonjour.txt
-rw-r--r--  2 root supergroup      2549 2025-10-11 05:18 /user/root/input/purchases.txt
root@hadoop-master:~# hadoop jar /shared_volume/hadoop-hdfs-ReadHDFS.jar /user/root/input/bonjour.txt
Bonjour tout le monde !
Mon message
```

Vérifier la création du fichier et lire son contenu :

```
hdfs dfs -ls /user/root/input/
hadoop jar /shared_volume/hadoop-hdfs-ReadHDFS.jar /user/root/
input/bonjour.txt
```

Le fichier a été créé avec succès et contient :

- "Bonjour tout le monde!"
- Le message passé en paramètre : "Mon message"

III. Conclusion

Dans ce TP, nous avons appris à programmer avec l'API HDFS en Java. Nous avons créé plusieurs programmes permettant de :

- Obtenir des informations sur les fichiers HDFS
- Lire le contenu des fichiers stockés dans HDFS
- Créer et écrire des fichiers dans HDFS
- Manipuler les fichiers (renommer, vérifier l'existence)