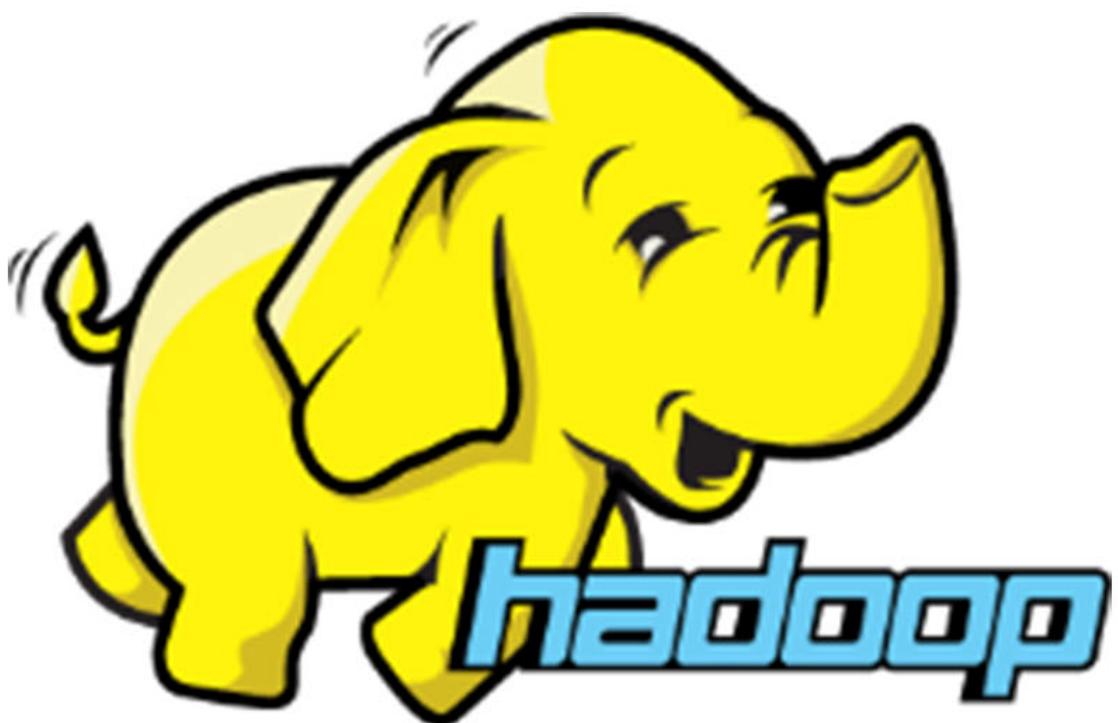


TP3 HDFS & HIVE



RABHI Sofiene
30 novembre 2023



Introduction

Dans ce travail pratique, nous allons explorer plusieurs composants clés de l'écosystème Big Data. Notre voyage commencera avec HDFS (Hadoop Distributed File System), où nous créerons une structure de répertoires et manipulerons des fichiers. Ensuite, nous plongerons dans Hive, une plateforme de gestion de base de données, pour créer et interroger des bases de données et des tables. Après cela, nous utiliserons Apache NiFi pour automatiser le traitement et la distribution des données, y compris l'exposition d'une API, la réception de fichiers de données, et leur dépôt dans HDFS. Chaque étape de ce TP est conçue pour vous donner une compréhension pratique de ces technologies et de leur rôle dans le traitement et l'analyse des données à grande échelle.

Nous avons tenté de télécharger et d'installer HDFS sur notre matériel de base, mais cela n'a pas fonctionné. Nous avons donc utilisé une machine virtuelle avec Ubuntu. Nous avons donc réinstallé NIFI et avons procéder à toutes les installations de Java et autres logiciels nécessaires pour le bon fonctionnement et l'installation puis la configuration de HDFS et Hive. Nous voilà après l'installation pour pouvoir commencer toutes les étapes du TP.

Finalisation du lancement de Hive et Hadoop

```
hadoop@JUSTE:~$ cd hive/bin
hadoop@JUSTE:~/hive/bin$ ./hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/
  slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4
  j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = f7955c6f-bd2f-48ee-b1ff-4a8ce4274f72

Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-3.1
  .3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
  using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = 6cbb1b2e-83ec-47e0-a4b1-445f260b7462
hive> hadoop@JUSTE:~/hive/bin$ cd
hadoop@JUSTE:~$ ls
1183775907689799760  hadoop-3.3.6.tar.gz  hive      snap
apache-hive-3.1.3-bin  hdfs                 hive.tar.gz wget-log
hadoop@JUSTE:~$ rm -rf apache-hive-3.1.3-bin/
hadoop@JUSTE:~$ ls
1183775907689799760  hadoop-3.3.6.tar.gz  hdfs  hive  hive.tar.gz  snap  wget-log
hadoop@JUSTE:~$ █
```

On voit bien que tout est fonctionnelles (je n'ai pas fait de capture d'encrant pour pas que le TP soit trop long mais je voulais souligner le fait que j'ai passé des nuits pour comprendre et réparer toutes les erreurs et les soucis rencontré dans cette installation pour que mon travail soit plus représentative. Nous allons donc commencer le TP avec les premières instructions.

→ Créer en lignes de commande HDFS l'arborescence suivante

Nous avons déjà démarré notre session Hive et nous sommes prêt à travailler avec HDFS. Pour créer l'arborescence demandée dans HDFS, Nous devons utiliser la commande **hdfs dfs**.

La première étape consiste à créer le répertoire **/data/common/raw/DATABASE_M1/ETUDIANT_M1** :

```
hadoop@JUSTE:~$ hdfs dfs -mkdir -p /data/common/raw/DATABASE_M1/ETUDIANT_M1
2023-12-11 10:52:07,009 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
```

On peut également vérifier si le répertoire c'est bien créé :

```
hadoop@JUSTE:~$ hdfs dfs -ls /data/common/raw/DATABASE_M1
2023-12-11 10:54:54,423 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x  - hadoop supergroup          0 2023-12-11 10:52 /data/common/raw/DATABASE_M1
/ETUDIANT_M1
```

On voit bien que cette commande a bien créé le chemin complet des répertoires.

→Créer un fichier studentM1_1.csv dans ce répertoire

Cette tâche consiste à créer un fichier nommé **studentM1_1.csv** dans ce répertoire, et on le remplit avec les trois colonnes : **firstName**, **lastName**, et **email**. Nous allons d'abord créer ce fichier localement sur Nôtre système, puis le copier dans HDFS. Voici les commandes utilisées :

```
hadoop@JUSTE:~$ vim studentM1_1.csv
hadoop@JUSTE:~$ hdfs dfs -put studentM1_1.csv /data/common/raw/DATABASE_M1/ETUDIANT_M1/
2023-12-11 11:10:41,814 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
hadoop@JUSTE:~$
```

→ Afficher le contenu du répertoire en lignes de commande HDFS

Pour vérifier que le fichier **studentM1_1.csv** a bien été copié dans le répertoire HDFS, on exécute la commande suivante, elle liste les fichiers présents dans le répertoire:

```
hadoop@JUSTE:~$ hdfs dfs -ls /data/common/raw/DATABASE_M1/ETUDIANT_M1
2023-12-11 11:21:36,773 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--  1 hadoop supergroup      98 2023-12-11 11:10 /data/common/raw/DATABASE_M1
/ETUDIANT_M1/studentM1_1.csv
hadoop@JUSTE:~$
```

→ Afficher le contenu du fichier en lignes de commande HDFS

Pour afficher le contenu du fichier **studentM1_1.csv** que Nous avons placé dans HDFS, on utilise la commande suivante, elle va lire et afficher le contenu du fichier **studentM1_1.csv** depuis HDFS.

```
hadoop@JUSTE:~$ hdfs dfs -cat /data/common/raw/DATABASE_M1/ETUDIANT_M1/studentM1_1.csv
2023-12-11 11:28:12,813 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
firstName,lastName,email
Alice,Dupont,alice.dupont@example.com
Bob,Durand,bob.durand@example.com

hadoop@JUSTE:~$
```

Le contenu de notre fichier **studentM1_1.csv** a été affiché avec succès depuis HDFS, ce qui signifie que tout fonctionne correctement jusqu'à présent. Maintenant, passons à la prochaine étape, qui concerne Hive.

→ Création d'une base de données dans Hive

Nous voulons exécuter une commande dans nôtres environnement Hive. Pour ce faire, Nous allons entrez dans Hive en tapant **hive** sur notre terminal, puis exécutez la commande ci-dessus.

Lancement de hive

```
hadoop@JUSTE:~$ cd hive/bin
hadoop@JUSTE:~/hive/bin$ ./hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hive/lib/log4j-slf4j-impl-2.17.1.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-reload4
j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 63ea8507-9a07-4870-97df-07b53d812032

Logging initialized using configuration in jar:file:/home/hadoop/hive/lib/hive-common-3.1
.3.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.Hive
Session ID = 6d1a902d-9dde-492b-b7b8-15754a4ad7d1

hive>
```

Commande pour crée une base de données

```
> CREATE DATABASE DATABASE_M1;
OK
Time taken: 0.052 seconds
hive>
```

→ Crée une base de données DATABASE_M2 avec HQL

Maintenant, nous allons créer une autre base de données, cette fois nommée **DATABASE_M2**. La procédure est similaire à celle que nous venons de suivre. On Utilise la commande suivante dans l'environnement Hive :

```
hive> CREATE DATABASE DATABASE_M2;
OK
Time taken: 0.036 seconds
hive> [ ]
```

→ Crée une table Hive ETUDIANT_M1 dans la base de données

Nous allons maintenant créer une table Hive nommée **ETUDIANT_M1** dans la base de données **DATABASE_M1**. Cette table va pointer sur le répertoire **/data/common/raw/DATABASE_M1/ETUDIANT_M1** que nous avons créé dans HDFS.

```
hive> USE DATABASE_M1;
OK
Time taken: 0.025 seconds
hive> CREATE EXTERNAL TABLE ETUDIANT_M1 (firstName STRING, lastName STRING, email STRING)
  > STORED AS TEXTFILE
  > LOCATION '/data/common/raw/DATABASE_M1/ETUDIANT_M1';
OK
Time taken: 0.079 seconds
hive> [ ]
```

Cette commande définit une table externe avec les colonnes **firstName**, **lastName** et **email**, et indique que la table utilise le répertoire HDFS que vous avez créé pour stocker ses données.

→ Afficher le contenu de la table ETUDIANT_M1 avec HQL

Pour afficher le contenu de la table **ETUDIANT_M1**, on utilise la commande suivante dans Hive. Cette commande va nous permettre de voir les données que nous avons chargées dans le fichier **studentM1_1.csv**:

```
Time taken: 0.079 seconds
hive> SELECT * FROM ETUDIANT_M1;
OK
Time taken: 0.733 seconds
hive> SELECT * FROM ETUDIANT_M1;
OK
Time taken: 0.09 seconds
hive> [ ]
```

Cependant, cela n'affiche pas les informations que l'on attendait, ce qui est normal nous n'avons encore ajouter aucune information. Nous verrons par la suite d'où provient sûrement le problème pour le diagnostiquer. En attendant, nous passons à l'étape d'après.

→ Créer une table ETUDIANT_M1_PART dans la base de données DATABASE_M1 partitionnée sur le champ DateRecep et pointant vers le répertoire /common/raw/DATABASE_M1/ETUDIANT_M1_PART

```
> CREATE TABLE ETUDIANT_M1_PART (firstName STRING, lastName STRING, email STRING)
> PARTITIONED BY (DateRecep STRING)
> STORED AS TEXTFILE
> LOCATION '/common/raw/DATABASE_M1/ETUDIANT_M1_PART';
OK
Time taken: 0.058 seconds
hive>
```

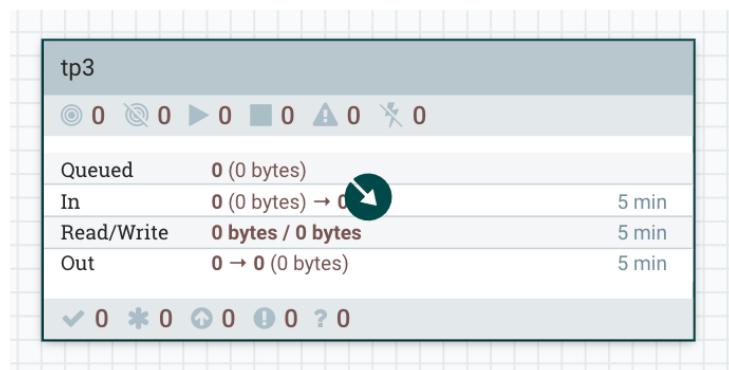
On donc bien le message qui nous indique que tout a bien été créé

→ Créer une table externe ETUDIANT_M2 dans la base de données DATABASE_M2

```
hive> USE DATABASE_M2;
OK
Time taken: 0.027 seconds
hive> CREATE EXTERNAL TABLE ETUDIANT_M2 (firstName STRING, lastName STRING, email STRING)
> STORED AS TEXTFILE;
OK
Time taken: 0.207 seconds
hive>
```

On donc bien le message qui nous indique que tout a bien été créée

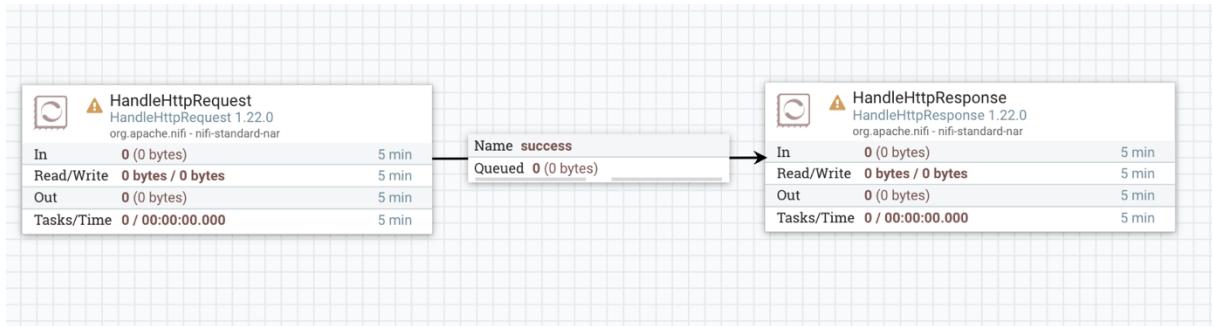
→ Exposer une API NiFi pour recevoir des données fichiers externes



Avant tout chose on crée un Process Groupe tp3 que vous pouvez retrouver sur le fichier Zip en xml, cela nous permet de bien s'organiser, ou y retrouvera à l'intérieure tout nôtres travaille pour cette fin de TP.

Nous allons maintenant aborder la partie Apache NiFi de nôtres TP. Cette partie se concentre sur la configuration de NiFi pour recevoir des fichiers CSV, les convertir en format Avro, et les stocker dans HDFS.

Pour cette première étape, nous devons configurer Apache NiFi pour recevoir des fichiers via une API HTTP. Les composants principaux à utiliser dans NiFi sont **HandleHttpRequest** et **HandleHttpResponse**.



Voici les grandes lignes de cette configuration :

→ **Envoyer, 10 fois, le fichier de données studentM1_2 à l'api nifi :**

Pour cette partie nous avons téléchargé le fichier student présent sur l'énoncé du TP sur Ecampus que nous avons directement déplacer dans le bon répertoire puis nous avons exécuter la commande suivante :

Déplacer le fichier au bon répertoire

```

hadoop@JUSTE:~/nifi-1.24.0-bin/nifi-1.24.0$ sudo mv /home/sofiene/Téléchargement
/student.csv .
[sudo] Mot de passe de hadoop :
mv: impossible d'évaluer '/home/sofiene/Téléchargement/student.csv': Aucun fichier
ou dossier de ce type
hadoop@JUSTE:~/nifi-1.24.0-bin/nifi-1.24.0$ sudo mv /home/sofiene/Téléchargement
s/student.csv .
hadoop@JUSTE:~/nifi-1.24.0-bin/nifi-1.24.0$ 
  
```

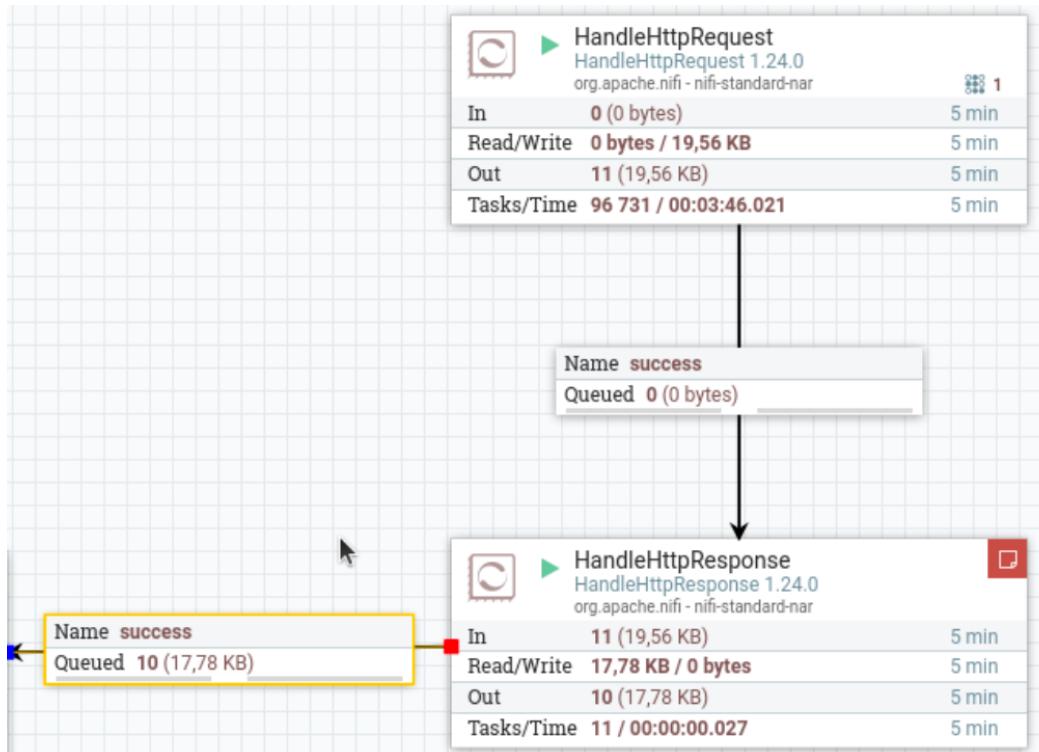
Envoyer les 10 fichier

```

hadoop@JUSTE:~/nifi-1.24.0-bin/nifi-1.24.0$ for i in {1..10}; do curl -X POST -F "f
ile=@student.csv" http://localhost:8080/api; done
NUMERO;PRENOM;NOM;GENDRE;FORMATION;MAIL
1;Hadil;ABDELMOUMEN;F;M1 MIAGE ID;Hadilabdelmoumen12@gmail.com
2;Ga tan;ALEXANDRE;M;M1 MIAGE ID;gaethan.alexandre@outlook.fr
3;Abdoul Karim;BARRY;M;M1 MIAGE ID;karimbarry628@gmail.com
4;Mamadou Billo;BARRY;M;M1 MIAGE ID;mbbarrymissira@gmail.com
5;Mamadou Lamarana;BARRY;M;M1 MIAGE ID;mamalamaranabarry@gmail.com
6;Ranim;BENRHAYEM;F;M1 MIAGE ID;benrhayemranim6@gmail.com
7;Dounia;BOLOUDENE;F;M1 MIAGE ID;bliddounia@gmail.com
8;Dmitrii;CHERNOV;M;M1 MIAGE ID;chernov.dms@gmail.com
9;Oumar;CISSE;M;M1 MIAGE ID;cisseoumar2025@gmail.com
10;Nasma;DAYEDAYE;F;M1 MIAGE ID;dayedayenasma@gmail.com
11;Emeric;DEDRY;M;M1 MIAGE ID;demeric95@gmail.com
12;Thierno;DIALLO;M;M1 MIAGE ID;thibrahimadiallo03@gmail.com
13;Cynthia;FOLLY;F;M1 MIAGE ID;abracynthiafolly@gmail.com
14;Yacouba;KABORE;M;M1 MIAGE ID;kyab99@gmail.com
15;Wissal;KHALYL;F;M1 MIAGE ID;wissalkhalil75@gmail.com
  
```

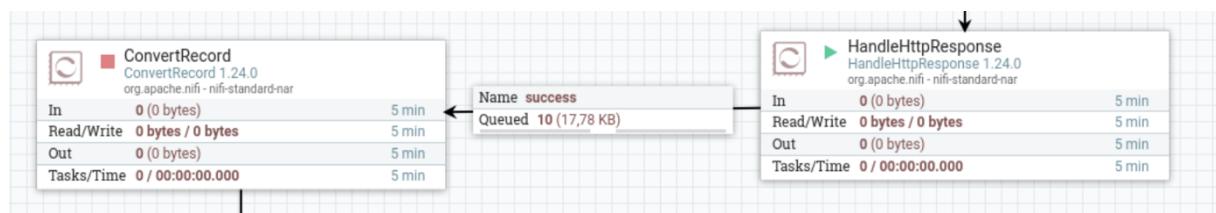
Nous avons donc utilisé la commande ci-dessus afin d'envoyer les difficultés. On voit bien qu'ils sont numérotés d'un jusqu'à la fin. On peut également voir sur la photo ci-dessous que les fichiers ont bien été téléchargé.

Affichage Nifi après exécution de la commande



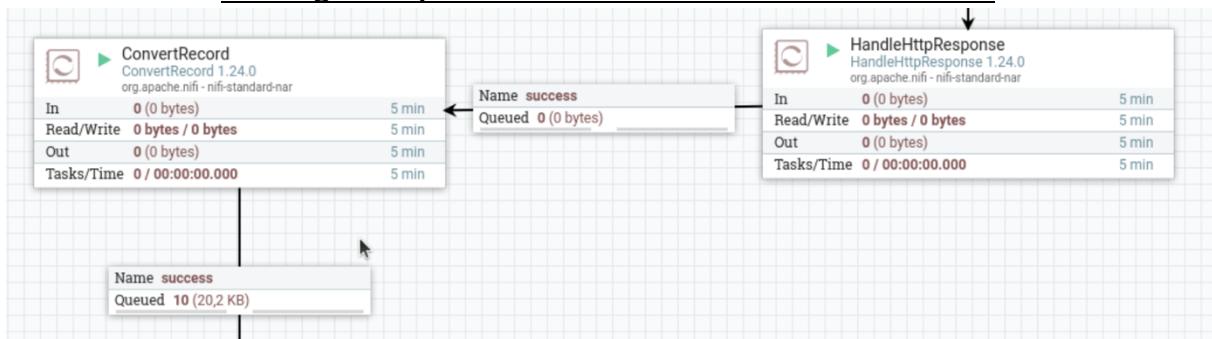
→ Convertir les données reçues avec le format CSV vers le format avro

Pour cela, sur nifi on utilise les processus ConvertRecord, comme on peut le voir si dessous
On voit bien que dans la queue, il y a les 10 fichiers :

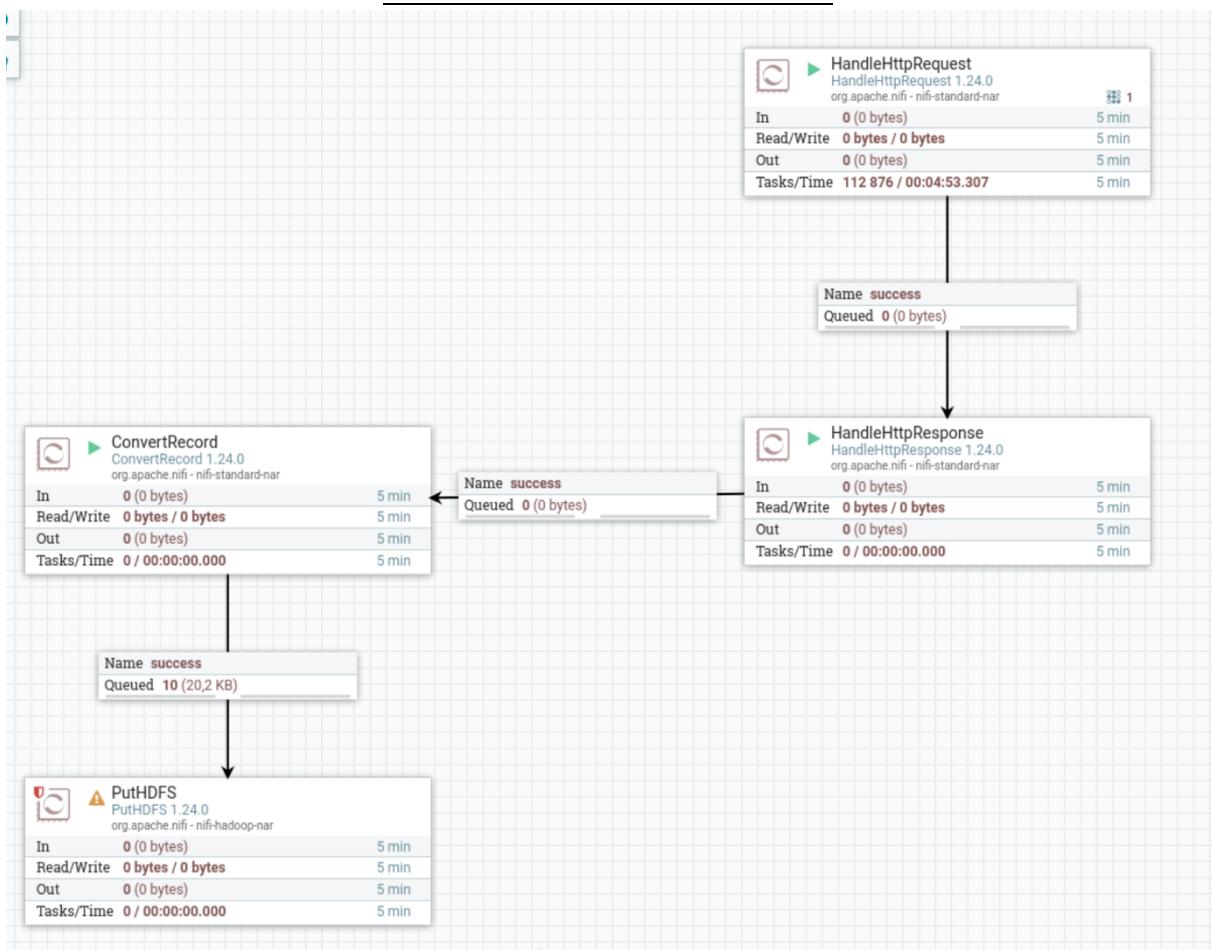


Nous configurons le processus ConvertRecord afin de convertir les fichier CSV en **vers le format avro**, comme on peut le voir sur la photo suivante :

Affichage nifi après exécution du Processus ConvertRecord



Voici le rendue final de nôtre Nifi



CONCLUSION

Nous avons abordé et surmonté des étapes cruciales dans l'installation et la configuration de notre environnement Hadoop et Hive, un processus qui a nécessité patience et persévérance de notre part. En naviguant à travers divers défis, incluant la résolution de problèmes de configuration et l'interprétation des avertissements et des messages de log, nous avons prouvé notre capacité à gérer un environnement de données complexe. Cela a impliqué la mise en place d'un cluster Hadoop, la configuration de Hive pour interagir avec ce dernier, et la création de bases de données et de tables pour la gestion efficace de nos données.

En outre, nous avons accompli plusieurs tâches clés tout au long de ce TP. Cela inclut la construction d'une arborescence requise dans HDFS, où nous avons placé des fichiers CSV, tout en apprenant à gérer le système de fichiers distribué. Nous avons également géré Hive en créant des tables, en gérant des partitions et en effectuant des requêtes HQL, ce qui a enrichi notre compréhension de cet outil. Par ailleurs, l'installation réussie de NiFi sur une machine virtuelle Ubuntu nous a permis d'automatiser et d'orchestrer le flux de données de notre travail pratique ce qui nous a pris de nombreuse heur dus au problèmes de la VM . En configurant des processeurs pour gérer les requêtes HTTP, convertir des fichiers CSV en Avro, et écrire les données transformées dans HDFS, nous avons affiné notre maîtrise de NiFi.

Ce parcours démontre non seulement notre compétence technique, mais aussi notre détermination à maîtriser des outils de pointe pour le traitement et la gestion des données, des compétences qui nous seront précieuses dans nos futurs projets, que ce soit en analyse de données, en science des données, ou dans tout domaine où la manipulation de grandes quantités de données est essentielle.