# Compte rendu

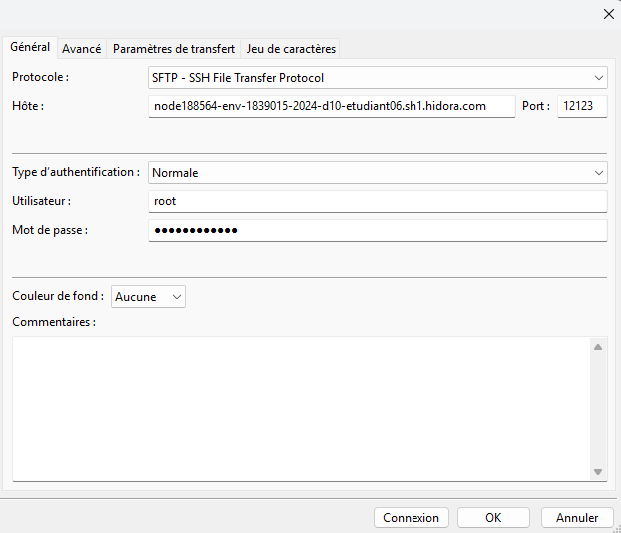
## Information complémentaire

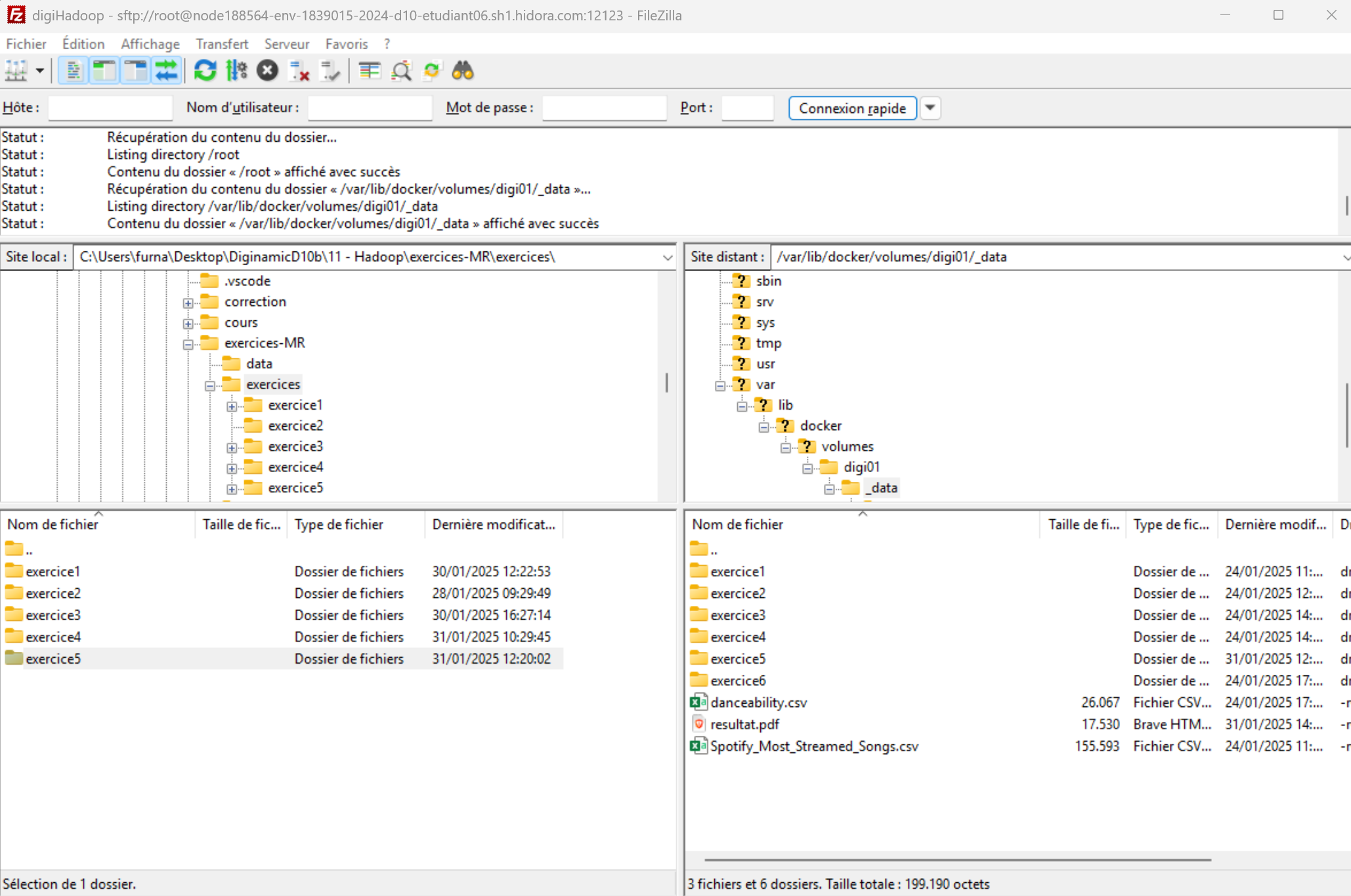
  
  
ci-dessus les informations pour pouvoir se connecter à distance

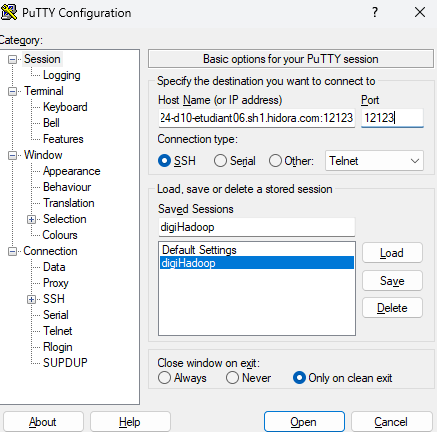
## Étapes du TP

### Connexion à la machine virtuelle

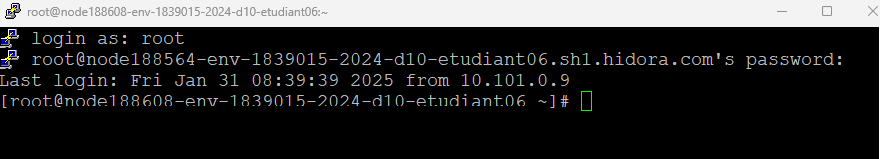
Pour pouvoir transférer les fichiers si cela est nécessaire nous avons utilisé Fillezilla.  
pour cela j’ai enregistré une connexion comme la photo ci-dessous.

   
  
  
- Hôte : correspond à l’url d’accès de la machine   
- Port : le port public de la machine qui correspond au port 22 (SSH)  
- Utilisateur : a l’utilisateur de ma machine ici c’est root  
- MDP : mot de passe de l’utilisateur

Quand tout cela est fini nous pourrons voir la connexion qui est établie  
le chemin pour transférer des fichiers à la VM correspond a /var/lib/docker/volumes/digi01/\_data/  
si des fichiers sont ici nous y aurons accès à la VM avec le chemin /datavolume1/  
  


Pour pouvoir se connecter en SSH nous avons utilisé Putty  
les identifiants de connexions sont pareil que Fillezilla  
  


- Hôte : correspond à l’url d’accès de la machine   
- Port : le port public de la machine qui correspond au port 22 (SSH)

Par la suite je « Save » pour garder les différentes informations   
quand on clic sur Open nous pouvons voir la fenêtre qui s’ouvre il ne manquera plus qu’à mettre les informations login et mdp de la machine  


### Lancer les conteneurs et services Hadoop

Quand nous somme sur la machine virtuelle   
il faudra lancer plusieurs .sh qui permet de lancer les services nécessaires.  
Voici les différents .sh et les commandes utilisées  
  
./start\_docker\_digi.sh  
./lance\_srv\_slaves.sh  
./bash\_hadoop\_master.sh

./start-hadoop.sh  
./services\_habse\_thrift.sh

Le start\_docker\_digi.sh permet de lancer le docker qui démarre les diffèrent machines maitre et esclave que l’on aura besoin.  
le lance\_srv\_slaves.sh démarre les services hbase et thrift dans les machines esclaves.  
bash\_hadoop\_master.sh permet d’utiliser la machine master  
  
dans le master le start-hadoop.sh permet de lancer le service DFS nécessaire pour communiquer avec hadoop grâce à la commande hdfs dfs et le service yarn qui est le gestionnaire de ressources utiliser par Hadoop  
  
et pour finir le services\_habse\_thrift.sh qui démarre les services hbase et thrift sur la machine actuelle qui est master  
  
pour démarrer une commande il faut juste l’appeler, par exemple en faisant ./start\_docker\_digi.sh

### 3. Importer les données dans HDFS

Pour importer les données nous allons utiliser la commande hdfs dfs qui permet de manipuler, crée, supprimer, modifier les données  
par exemple :  
- hdfs dfs -mkdir input : permet de crée in fichier input  
- hdfs dfs -put /chemin/ : permet d’ajouter un fichier test dans le chemin  
- hdfs dfs -get /chemin/: permet de récupérer un fichier test dans chemin  
- hdfs dfs -ls : permet de voir les données  
- hdfs dfs -rm /chemin/: permet de supprimer un fichier test dans chemin

Plusieurs autres commandes existent  
  
nous allons automatiser tout ça grâce à un scipt shell qui s’appelle job.sh

### Créer et exécuter un job MapReduce

Pour pouvoir exécuter notre mapper et réducer nous allons utilisez un job.sh  
qui indique les différent commandes hdfs pour mener à bien les différentes opération  
  
comme exécuter le mapper (sert à décomposer et à cartographier les données.),  
exécuter le réducer (mélange et réduit les données.)  
  
le job.sh ressemblera à :

# Définition des variables d'entrée et de sortie

INPUT\_FILE="Spotify\_Most\_Streamed\_Songs.csv"

OUTPUT\_FILE="spotify.txt"

# Création du répertoire d'entrée et de sortie s'ils n'existent pas

hdfs dfs -mkdir -p input

hdfs dfs -mkdir -p output

# Copie du fichier d'entrée dans le HDFS

hdfs dfs -put $INPUT\_FILE input

# Suppression du répertoire de sortie s'il existe déjà

hdfs dfs -rm -r output/$OUTPUT\_FILE

# Lancement du job Hadoop avec les scripts mapper et reducer

hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.7.2.jar -file mapper.py -mapper "python3 mapper.py" -file reducer.py -reducer "python3 reducer.py" -input input/$INPUT\_FILE -output output/$OUTPUT\_FILE

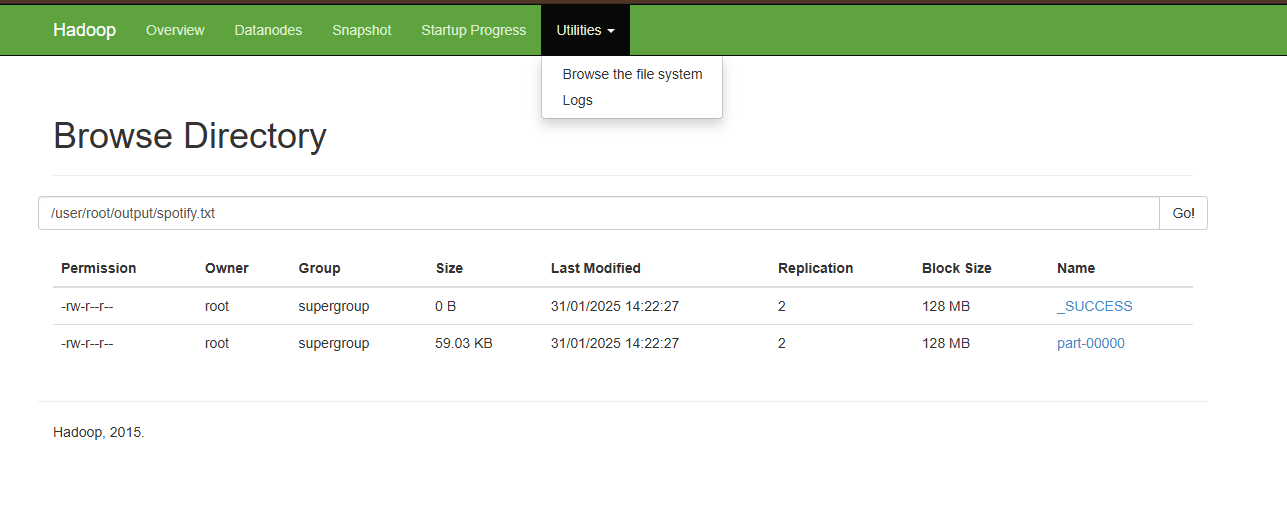
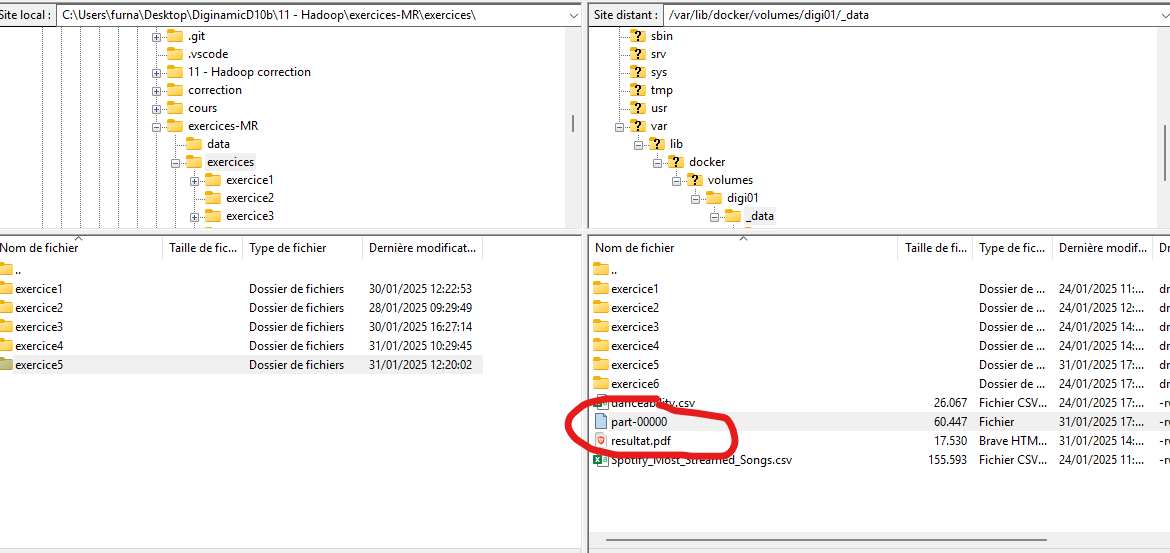
Avec ce job.sh tous les action nécessaires seront effectué,  
bien entendu il faut que les différent fichiers soit présent comme le mapper.py, le reducer.py le fichier input et les dossiers input et output  
pour cela nous utiliserons Fillezilla comme vu plus haut et pourrons faire une commande comme  
mv /datavolume1/exercice5/reducer.py .  
qui permettra de bouger les différent fichier nécessaire (/datavolume1/exercice5/\* . Permet de tout déplacer

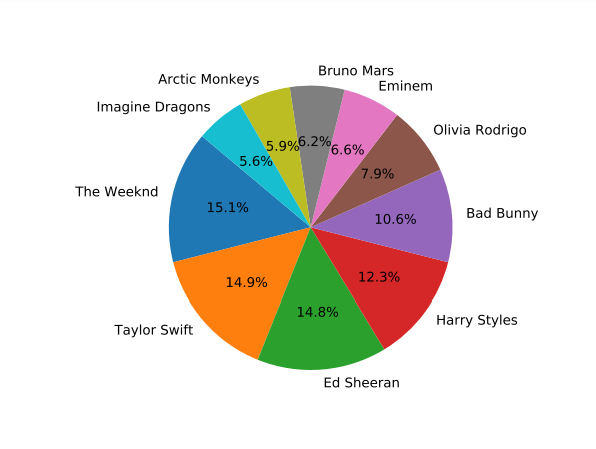
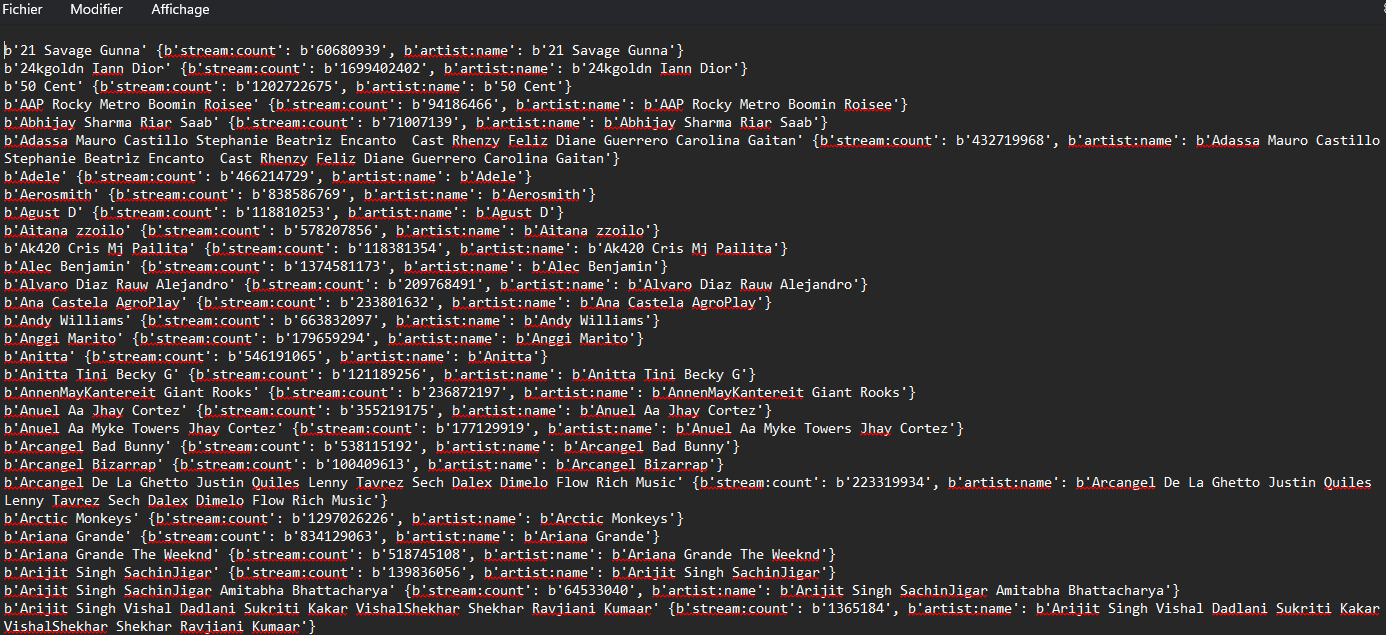
### Visualiser les résultats

Après avoir lancé le job il y a plusieurs façons de voir les résultats.  
Première façon si dans le script nous mettons des print() nous pouvons voir des informations directement en lançant le script (cela n’est pas une bonne solution sachant qu’il faut les enlever quand nous envoyons le code en production)  
l’autre façon et qu'après le travail des job nous indiquons d’avoir un fichier en sortie  
OUTPUT\_FILE="spotify.txt"  
grâce à ce fichier nous pouvons avoir des informations et des résultats.  
nous pouvons le voir grâce à la commande  
hdfs dfs -ls output  
a savoir que si l’on doit récupérer un fichier il faudra récupérer le par-00000 qui se trouve dans la commande  
hdfs dfs -ls output/spotify.txt  
  
à savoir que je génère un pdf directement dans /datavolume1/resultat.pdf  
quand le job est fini dans le code cela ressemble a cela  
with PdfPages('/datavolume1/resultat.pdf') as pdf:

pdf.savefig() # Sauvegarder le graphe dans le fichier PDF

### Récupérer les résultats

Pour récupérer tous ces résultats :  
on peut passer par l’interface graphique  
http://node188608-env-1839015-2024-d10-etudiant06.sh1.hidora.com:12119/  
12119 correspond au port 9070  
ensuite, on peut aller dans utilities -> browse the file system  
  
l’autre maniéré et de mettre les fichiers qui nous intéressent directement dans /datavolumes1/ et les récupérer via fillezilla  
hdfs dfs -get output/spotify.txt/part-00000 /datavolume1/  
qui permet de le récuperer dans hadoop et de le mettre dans le datavolume1  
pour le pdf je génère directement le résultat dans datavolume1 donc il y est déjà  
  
sur fillezilla je vais dans le chemin/var/lib/docker/volumes/digi01/\_data  
et je peux donc récupérer mes fichier part-00000 et resultat.pdf  


On aura donc le pdf  
  
et le fichier txt  
  
  
  
on aura récupéré nos données.