

Documentation du Processus : Mise en Place d'un Environnement Big Data avec Spark, Elasticsearch et Kibana

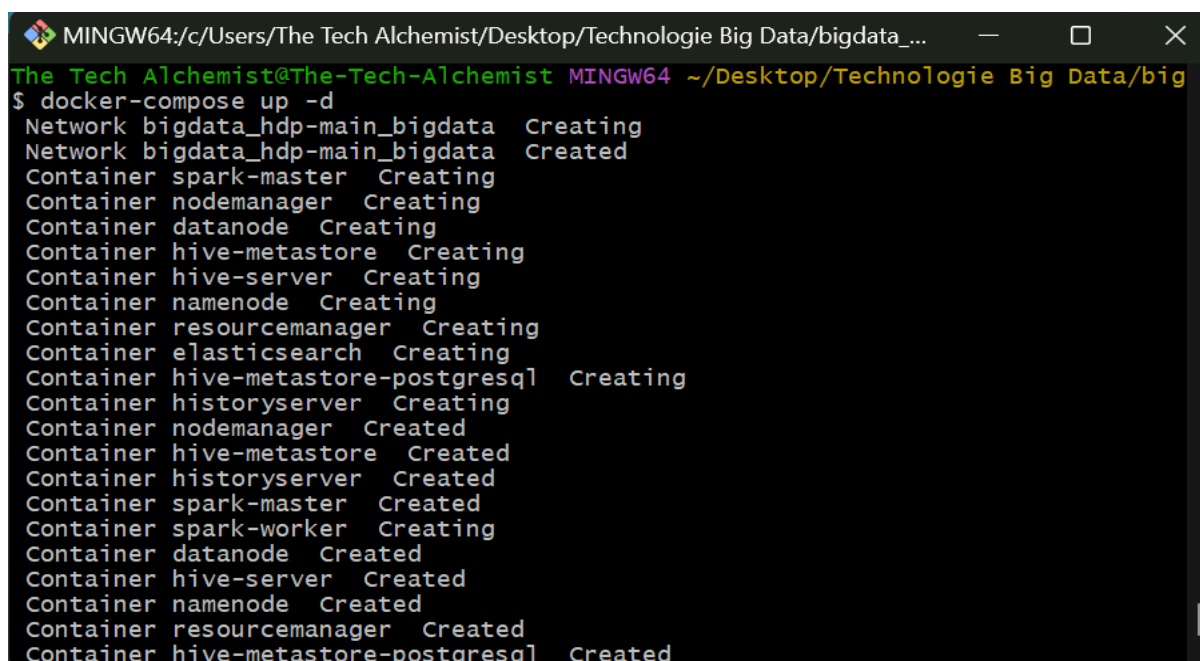
Introduction

Le projet consiste à créer un environnement Big Data pour l'analyse des données d'un fichier `avocado.csv`. Nous utiliserons les technologies suivantes :

- **Spark** : Pour le traitement des données.
- **Elasticsearch** : Pour l'indexation des données.
- **Kibana** : Pour la visualisation des données.

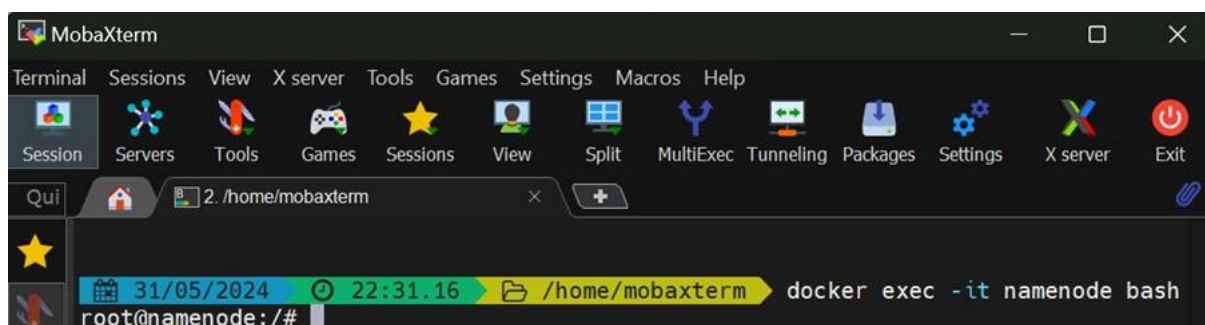
Étape 1 : Demarrage et initialisation

1. Demarrer les machines docker :



```
MINGW64/c/Users/The Tech Alchemist/Desktop/Technologie Big Data/bigdata_...
The Tech Alchemist@The-Tech-Alchemist MINGW64 ~/Desktop/Technologie Big Data/big
$ docker-compose up -d
Network bigdata_hdp-main_bigdata Creating
Network bigdata_hdp-main_bigdata Created
Container spark-master Creating
Container nodemanager Creating
Container datanode Creating
Container hive-metastore Creating
Container hive-server Creating
Container namenode Creating
Container resourcemanager Creating
Container elasticsearch Creating
Container hive-metastore-postgresql Creating
Container historyserver Creating
Container nodemanager Created
Container hive-metastore Created
Container historyserver Created
Container spark-master Created
Container spark-worker Creating
Container datanode Created
Container hive-server Created
Container namenode Created
Container resourcemanager Created
Container hive-metastore-postgresql Created
```

2. Connexion a la machine namenode :

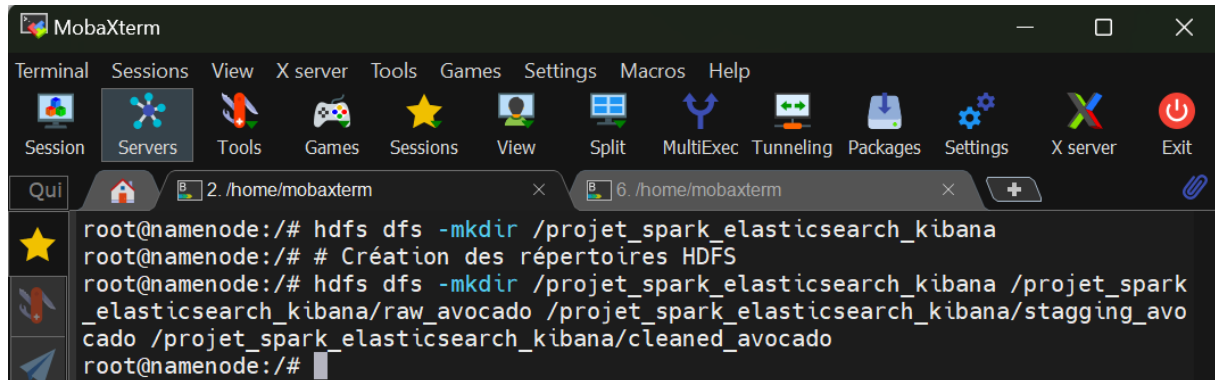


```
MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings X server Exit
Qui 2. /home/mobaxterm
31/05/2024 22:31.16 /home/mobaxterm docker exec -it namenode bash
root@namenode:/#
```

Étape 2 : Configuration de l'Environnement HDFS

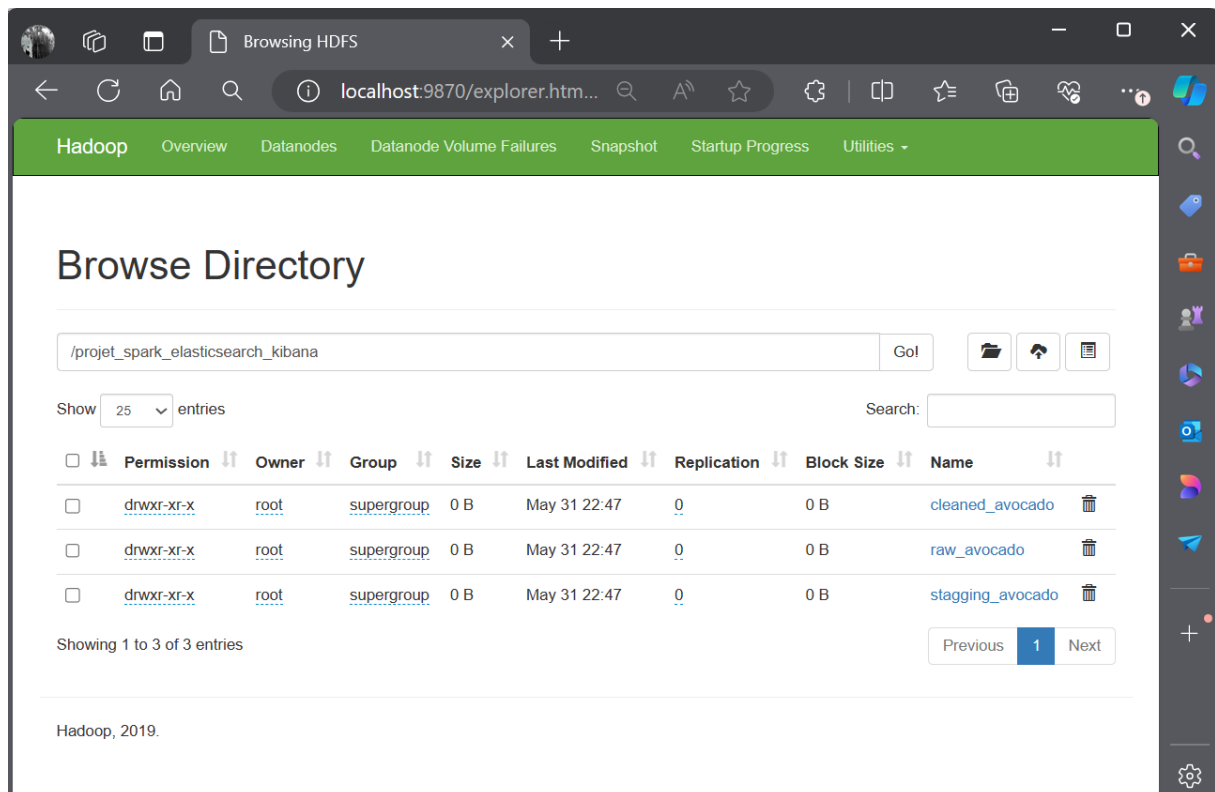
3. Création des répertoires HDFS :

- o /raw_avocado
- o /staging_avocado
- o /cleaned_avocado



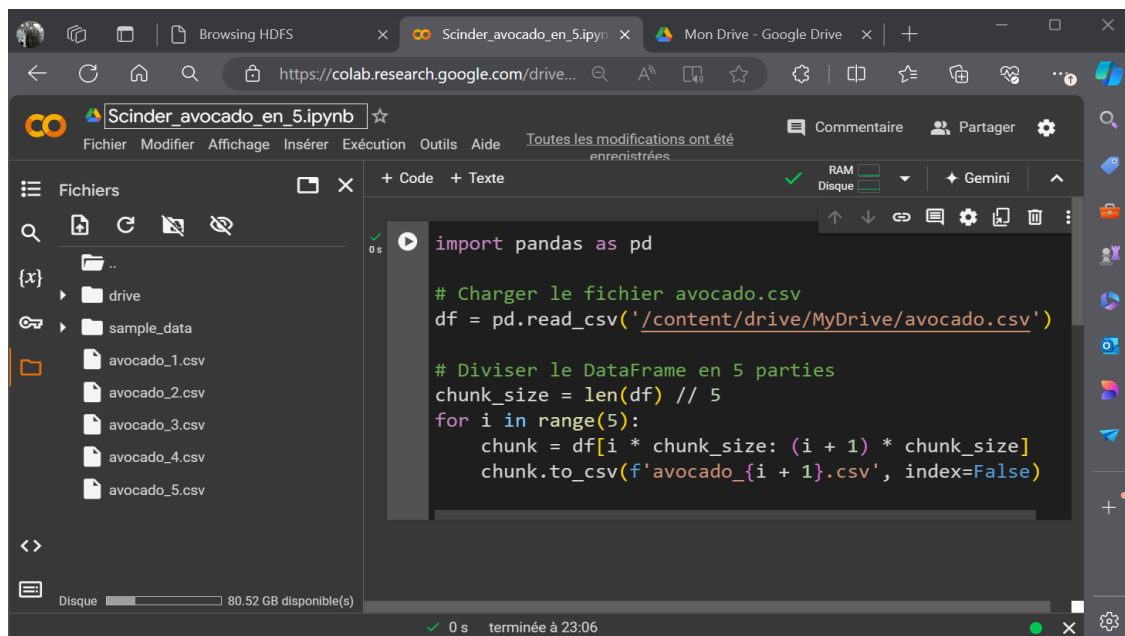
```
root@namenode:/# hdfs dfs -mkdir /projet_spark_elasticsearch_kibana
root@namenode:/# # Création des répertoires HDFS
root@namenode:/# hdfs dfs -mkdir /projet_spark_elasticsearch_kibana /projet_spark_elasticsearch_kibana/raw_avocado /projet_spark_elasticsearch_kibana/staging_avocado /projet_spark_elasticsearch_kibana/cleaned_avocado
root@namenode:/#
```

Verification

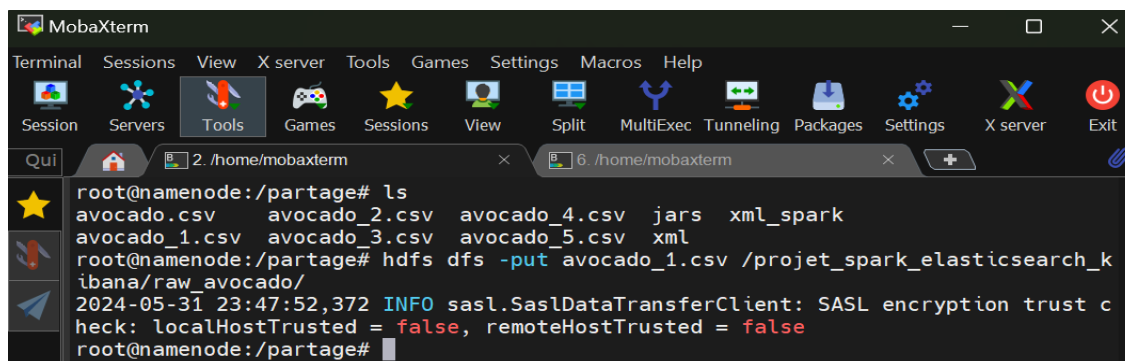


4. Scinder le fichier avocado.csv en 5 fichiers :

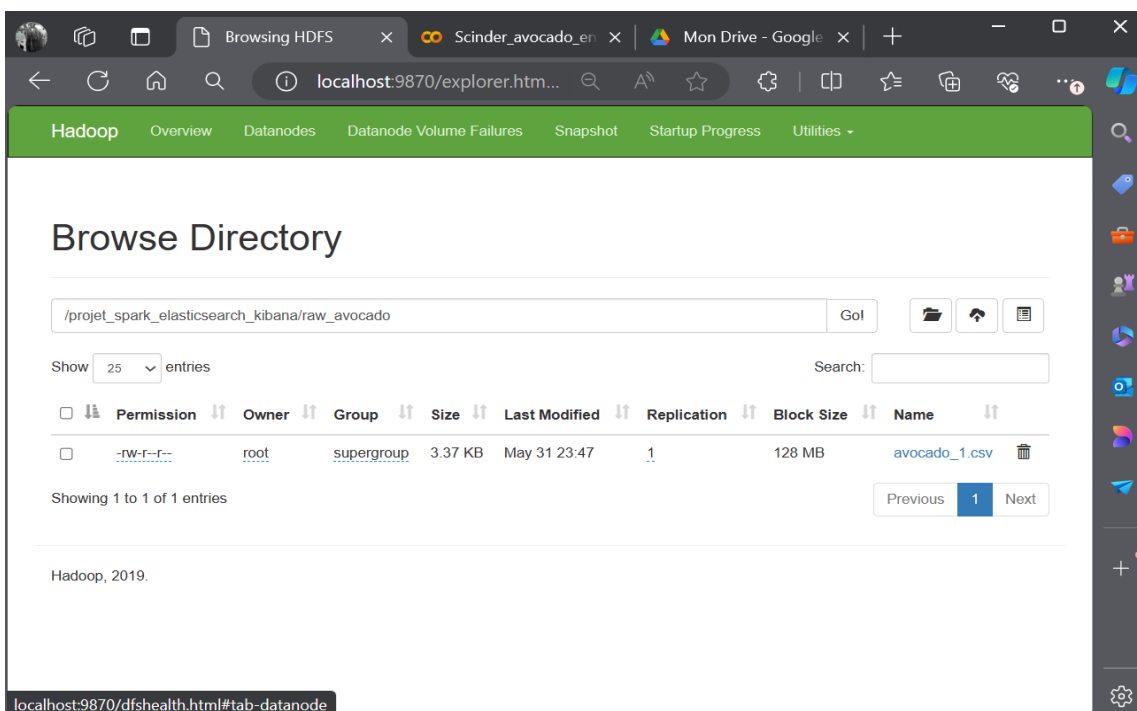
- o avocado_1.csv
- o avocado_2.csv
- o avocado_3.csv
- o avocado_4.csv
- o avocado_5.csv



5. Charger le premier fichier `avocado_1.csv` dans le répertoire HDFS `/raw_avocado`.



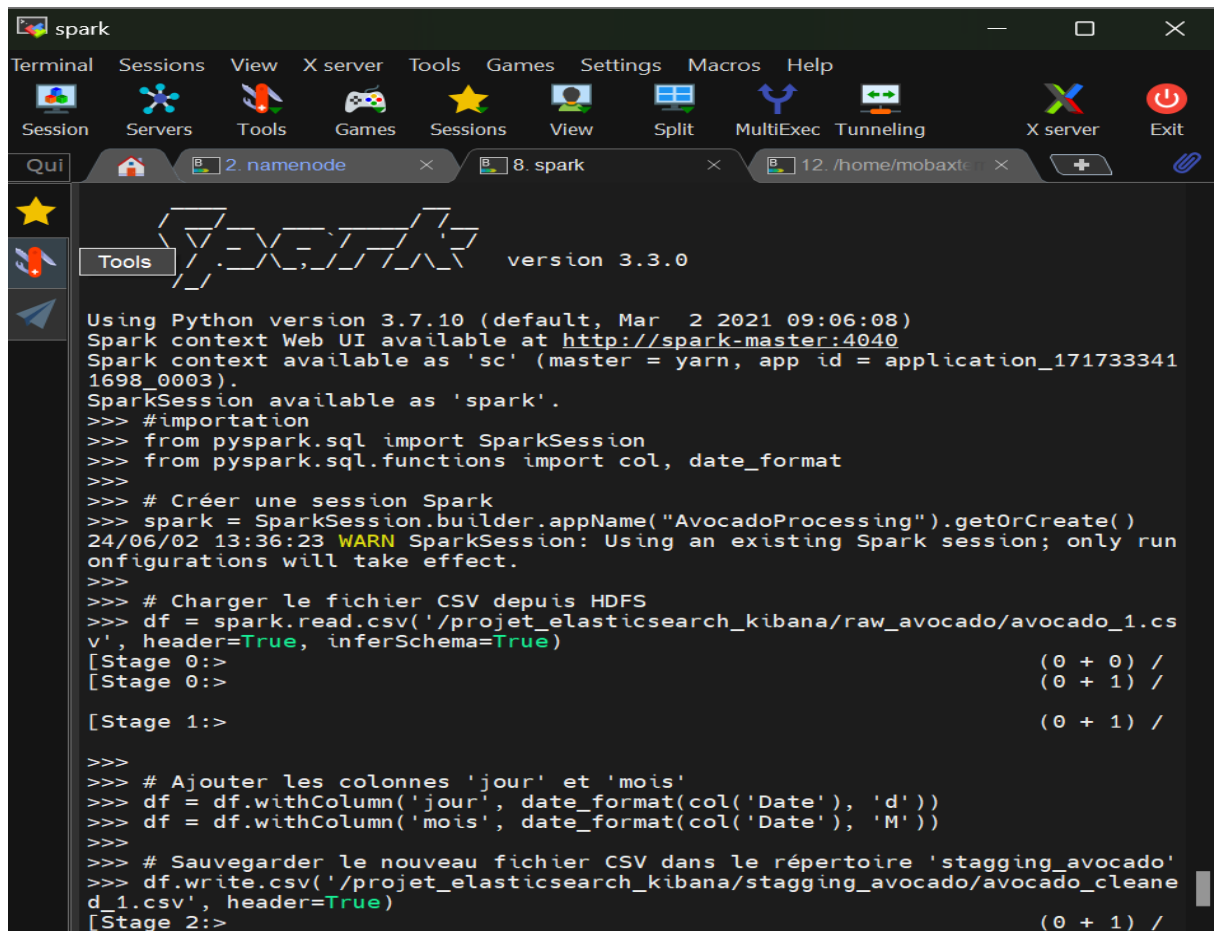
Verification



Étape 3 : Traitement des Données avec Spark

1. Script Spark (PYSPARK) pour :

- Récupérer le fichier CSV depuis /raw_avocado.
- Ajouter les colonnes 'jour' et 'mois' en extrayant les informations du champ 'date'.
- Sauvegarder le résultat dans un fichier CSV (par exemple, avocado_cleaned_1.csv) dans /staging_avocado.



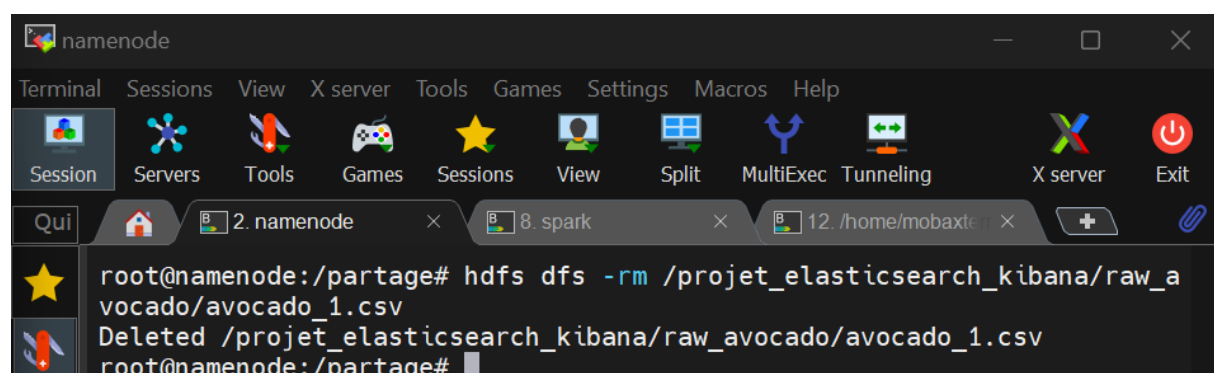
```
spark
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling X server Exit

Qui 2. namenode 8. spark 12. /home/mobaxte...

Tools version 3.3.0

Using Python version 3.7.10 (default, Mar 2 2021 09:06:08)
Spark context Web UI available at http://spark-master:4040
Spark context available as 'sc' (master = yarn, app id = application_1717333411698_0003).
SparkSession available as 'spark'.
>>> #importation
>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import col, date_format
>>>
>>> # Créer une session Spark
>>> spark = SparkSession.builder.appName("AvocadoProcessing").getOrCreate()
24/06/02 13:36:23 WARN SparkSession: Using an existing Spark session; only run onfigurations will take effect.
>>>
>>> # Charger le fichier CSV depuis HDFS
>>> df = spark.read.csv('/projet_elasticsearch_kibana/raw_avocado/avocado_1.csv', header=True, inferSchema=True)
[Stage 0:> (0 + 0) /
[Stage 0:> (0 + 1) /
[Stage 1:> (0 + 1) /
>>>
>>> # Ajouter les colonnes 'jour' et 'mois'
>>> df = df.withColumn('jour', date_format(col('Date'), 'd'))
>>> df = df.withColumn('mois', date_format(col('Date'), 'M'))
>>>
>>> # Sauvegarder le nouveau fichier CSV dans le répertoire 'staging_avocado'
>>> df.write.csv('/projet_elasticsearch_kibana/staging_avocado/avocado_cleaned_1.csv', header=True)
[Stage 2:> (0 + 1) /
```

- Suppression du fichier avocado_1.csv du répertoire /raw_avocado.



```
namenode
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling X server Exit

Qui 2. namenode 8. spark 12. /home/mobaxte...

root@namenode:/partage# hdfs dfs -rm /projet_elasticsearch_kibana/raw_avocado/avocado_1.csv
Deleted /projet_elasticsearch_kibana/raw_avocado/avocado_1.csv
root@namenode:/partage#
```

2. Script Spark pour :

- Récupérer le fichier avocado_cleaned_1.csv depuis /staging_avocado.
- Indexer les données dans Elasticsearch sous l'indexe 'avocado'.
- Déplacer le fichier CSV dans le répertoire /cleaned_avocado.

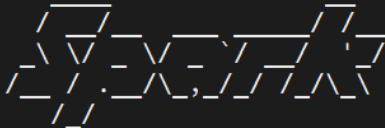
MobaXterm

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling X server Exit

Qui 2. namenod: 8. /home/mol: 13. /home/mo: 12. elas

To adjust logging level use `sc.setLogLevel(newLevel)`. For SparkR, use `s`
24/06/02 15:42:27 **WARN** Utils: Service 'SparkUI' could not bind on port
24/06/02 15:42:34 **WARN** Client: Neither spark.yarn.jars nor spark.yarn.a
Welcome to

 version 3.3.0

Using Python version 3.7.10 (default, Mar 2 2021 09:06:08)
Spark context Web UI available at <http://spark-master:4041>
Spark context available as 'sc' (master = yarn, app id = application_17
SparkSession available as 'spark'.
>>> `df = spark.read.option("header", True).option("inferSchema", "true`
>>> `df.show()`

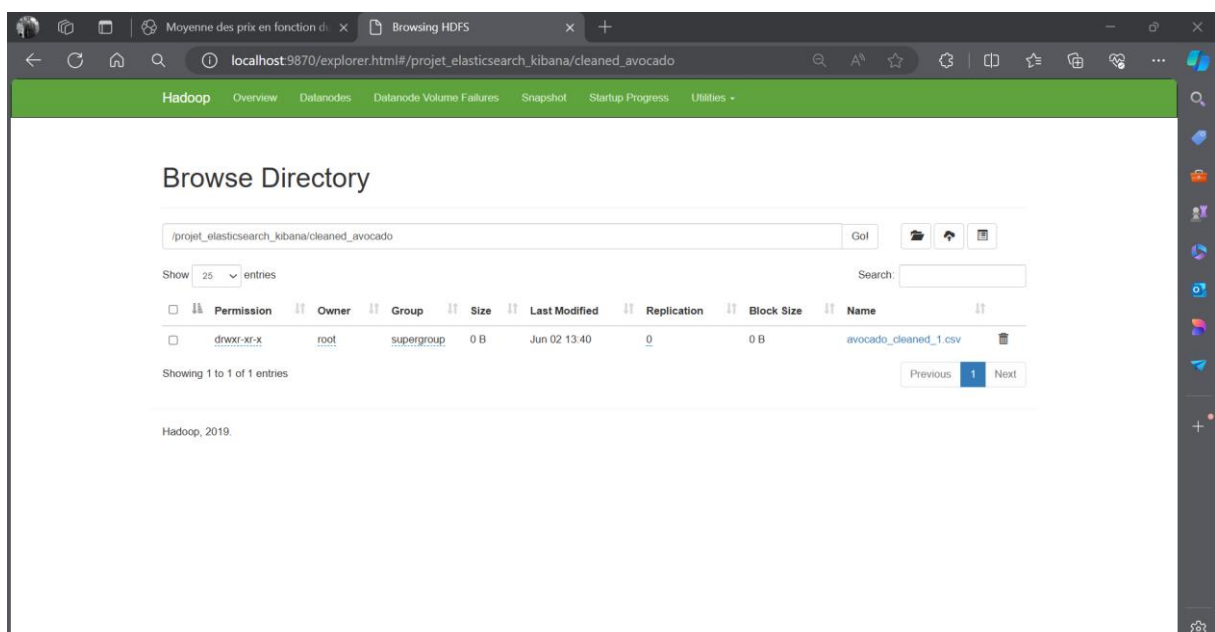
Date	AveragePrice	Volume	type	year	region
2015-01-04 00:00:00	1.0	435021.49	conventional	2015	Atlanta
2015-01-04 00:00:00	1.22	40873.28	conventional	2015	Albany
2015-01-11 00:00:00	1.24	41195.08	conventional	2015	Albany

>>> `df = spark.read.option("header", True).option("inferSchema", "true`
`projel_elasticsearch_kibana/stagging_avocado/avocado_cleaned_1.csv")`
>>> `df.show()`

Date	AveragePrice	Volume	type	year	region
2015-01-04 00:00:00	1.0	435021.49	conventional	2015	Atlanta
2015-01-04 00:00:00	1.22	40873.28	conventional	2015	Albany
2015-01-11 00:00:00	1.24	41195.08	conventional	2015	Albany
2015-01-11 00:00:00	1.11	397542.72	conventional	2015	Atlanta

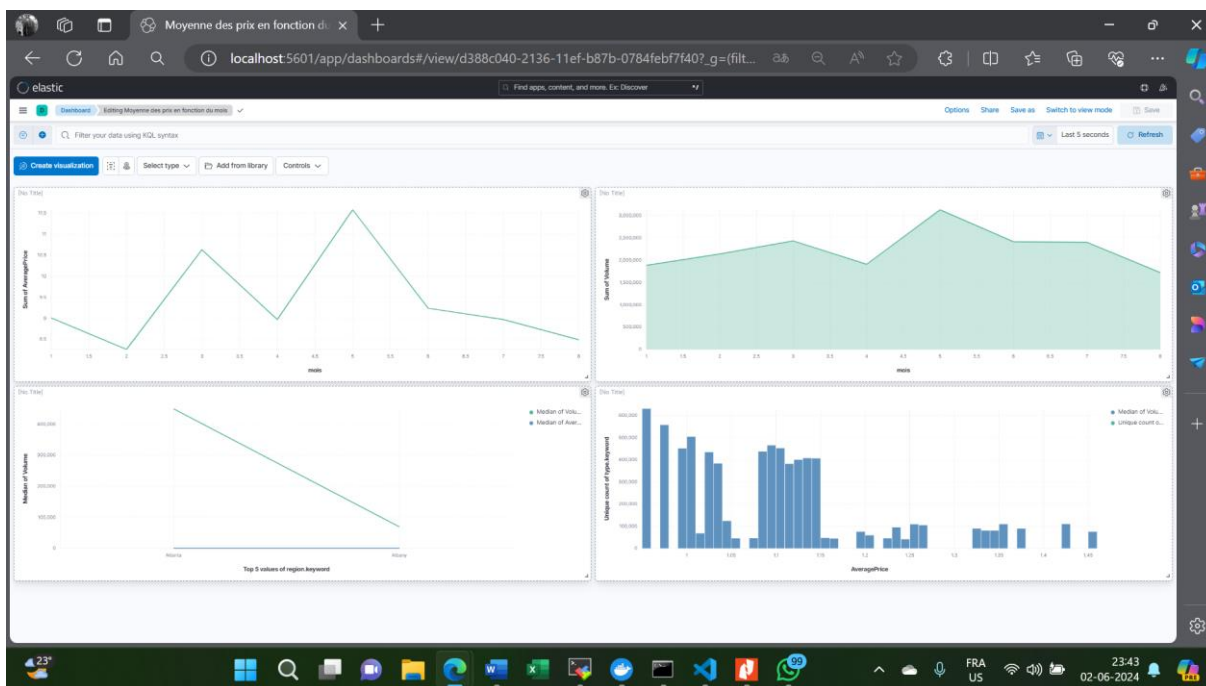
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional

Verification



Étape 4 : Création du Dashboard Kibana

1. Créer un Dashboard Kibana avec quatre graphes pertinents pour décrire les données :



Étape 4 : Vérification et Évolution

Voir PowerPoint en attachement

Conclusion

Ce projet nous a permis de consolider nos connaissances en Spark, Elasticsearch, Kibana et HDFS. Cependant, nous avons rencontré quelques difficultés majeures qui ont ralenti notre progression. L'accès initial à l'interface de Kibana a été entravé par des problèmes d'exécution d'Elasticsearch. Heureusement, la découverte fortuite des logs a révélé que la mémoire disque de notre PC était pleine, ce qui a affecté le bon fonctionnement d'Elasticsearch. Malgré ces obstacles, nous avons réussi à mettre en place un environnement fonctionnel pour l'analyse et la visualisation des données relatives aux avocats.