

SASSI YOUSSEF
RAHMOUNI FARIS

ITS 2



UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

Projet DataMining



1. Remerciements

Avant toute chose nous tenons à remercier l'enseignant SOUIDI Mohamed qui nous a répondu à chacune de nos interrogations et qui nous a permis de mettre en œuvre ce projet très enrichissant qui nous permet une première réelle initiation au monde du Big Data.

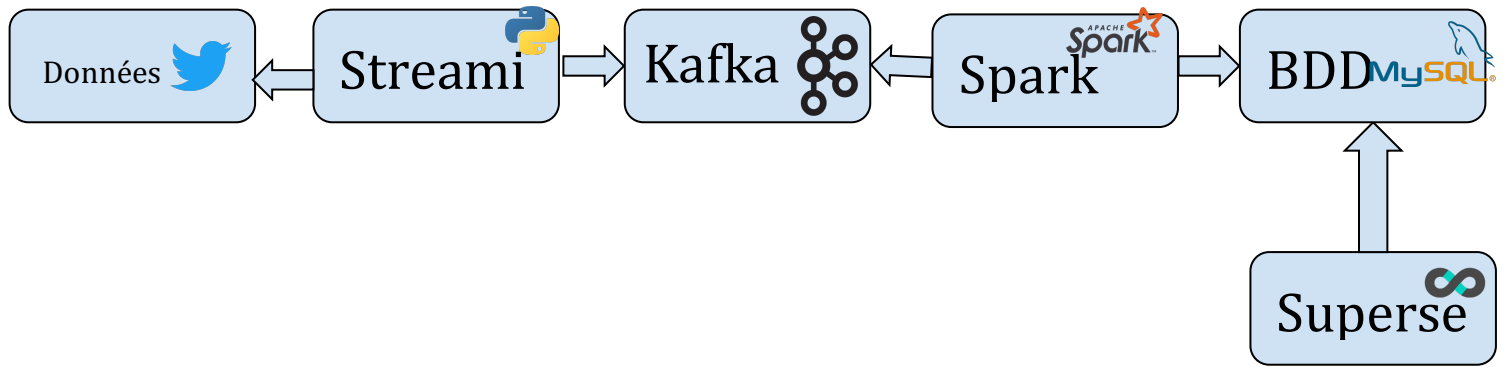
2. Objectifs

Ce projet, très intéressant, nous a été présenté lors de la séance du XX/XX/XXXX. L'idée de ce projet est de réussir à streamer des données (ici de Twitter, Youtube, JSON, fichier .csv...) et d'avoir un compte rendu en temps réel de l'évolution de ces données.

Pour cela nous allons utiliser différents frameworks, logiciels :

- *Python* afin de faire le code qui va nous permettre de récupérer la donnée
- *Apache Kafka* qui est une plateforme de traitement de données en streaming en temps réel.
- *Apache Spark* qui lui est un moteur d'analyses unifiées ultra-rapide pour le big data et le machine learning.
- *MYSQL*, un élément essentiel en effet c'est ici que les données vont pouvoir être enregistrées.
- *Superset*, il s'agit d'un outil de visualisation et d'exploration de données.
- *VirtualBox*, logiciel de virtualisation qui nous permet de travailler avec le système d'exploitation Linux.

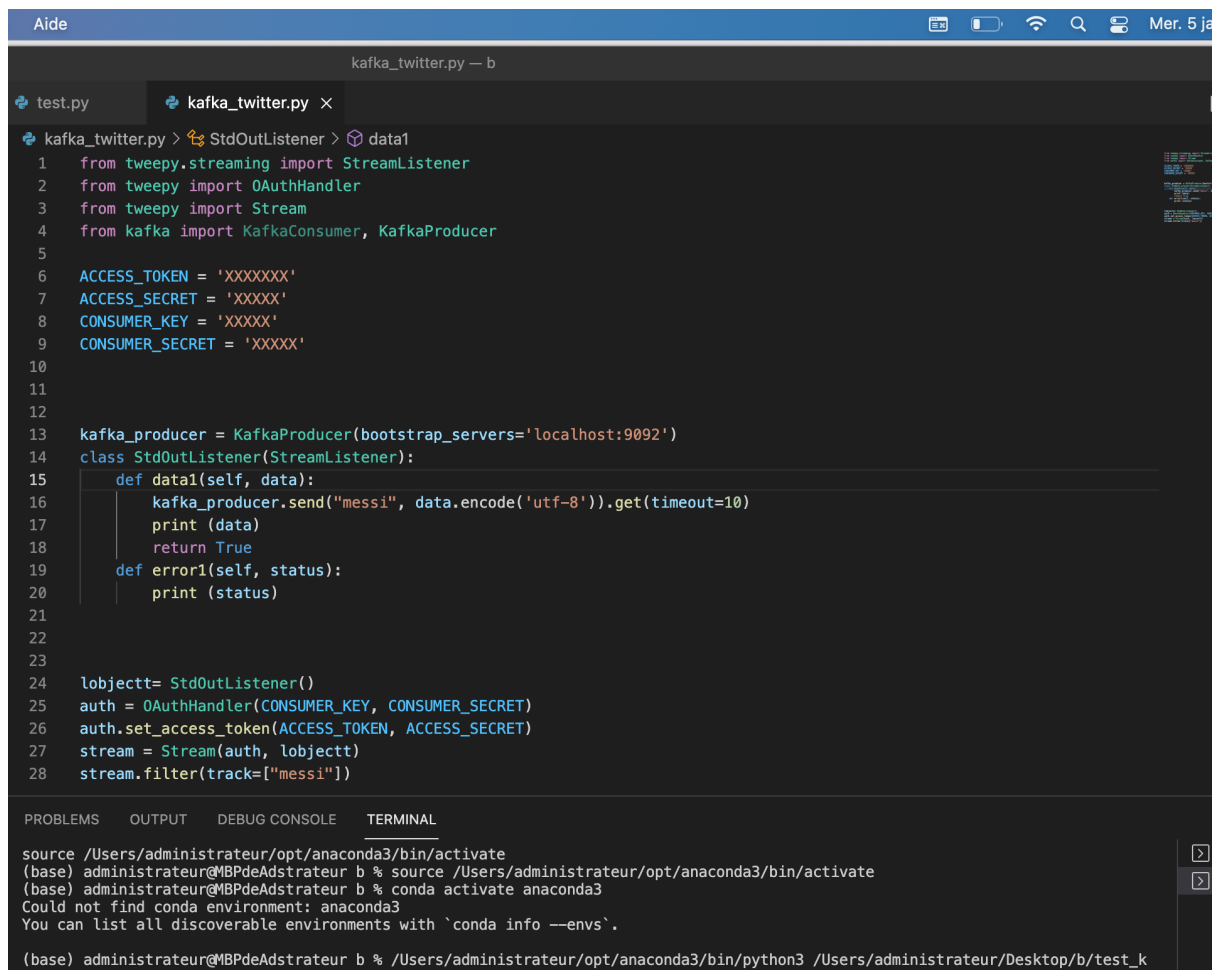
L'architecture du projet est parfaitement décrite dans le schéma suivant :



3. Streaming

Cette étape est la première que nous avons eu à réaliser, c'est à partir de cette étape que tout le reste du projet va s'articuler. Nous avons fait cette étape en deux partie :

a) *Streaming Twitter*



```
Aide
kafka_twitter.py — b

test.py  kafka_twitter.py x
kafka_twitter.py > StdOutListener > data1
1 from tweepy.streaming import StreamListener
2 from tweepy import OAuthHandler
3 from tweepy import Stream
4 from kafka import KafkaConsumer, KafkaProducer
5
6 ACCESS_TOKEN = 'XXXXXXX'
7 ACCESS_SECRET = 'XXXXXX'
8 CONSUMER_KEY = 'XXXXXX'
9 CONSUMER_SECRET = 'XXXXXX'
10
11
12
13 kafka_producer = KafkaProducer(bootstrap_servers='localhost:9092')
14 class StdOutListener(StreamListener):
15     def data1(self, data):
16         kafka_producer.send("messi", data.encode('utf-8')).get(timeout=10)
17         print(data)
18         return True
19     def error1(self, status):
20         print(status)
21
22
23
24 lobjectt= StdOutListener()
25 auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
26 auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)
27 stream = Stream(auth, lobjectt)
28 stream.filter(track=["messi"])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
source /Users/administrateur/opt/anaconda3/bin/activate
(base) administrateur@MBPdeAdstrateur b % source /Users/administrateur/opt/anaconda3/bin/activate
(base) administrateur@MBPdeAdstrateur b % conda activate anaconda3
Could not find conda environment: anaconda3
You can list all discoverable environments with `conda info --envs`.

(base) administrateur@MBPdeAdstrateur b % /Users/administrateur/opt/anaconda3/bin/python3 /Users/administrateur/Desktop/b/test_k
```

Nous avons eu comme idée de streamer des tweets. Ces tweets-là seraient des tweets politiques, le nom des candidats qui ressortent le plus dans les débats sur twitter pour l'élection présidentielle 2022.

Ci dessus, le code python que nous avons utilisé afin de faire ce stream, cependant, on a rencontré un problème sur la partie authentification. Ce stream est possible grâce à la bibliothèque Tweepy.

Pour travailler avec l'API Twitter, il nous est nécessaire de posséder un compte Twitter, une fois connecté à ce compte il faut se rendre sur le site web Twitter Developer, et y créer une "Application". Une fois cette application créée, nous obtenons les informations nécessaires à la récupération de tweet dans l'onglet "Keys and Access tokens", il s'agit de nos identifiants :

- une clé d'API (aussi appelée "Consumer Key") et sa clé secrète ;
- un "Access Token" et son pendant secret.

Une fois que ces identifiants ont été créés, on les utilise sur python de la façon suivante (toute la procédure est détaillé sur la documentation officiel de Tweepy)

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(key, secret)
acces_api = tweepy.API(auth)
```

Cependant une fois que le programme est exécuté, nous rencontrons une erreur, il s'agit d'une sortie infinie affichant le chiffre "403" comme affiché sur la dernière capture. Mais grâce à la séance d'aujourd'hui (05/01/2022), on a trouvé la réponse à notre problème : il faut faire une demande d'autorisation à Twitter pour nous laisser manipuler les tweets. Nous allons donc faire cette demande afin de continuer notre projet avec twitter

```
You can list all discoverable environments with 'conda info --envs'.
```

```
(base) administrateur@MBPdeAdstrateur b % /Users/administrateur/opt/anaconda3/bin/python3 /Users/administrateur/Desktop/b/test_kafka_twitter.py
403
403
```

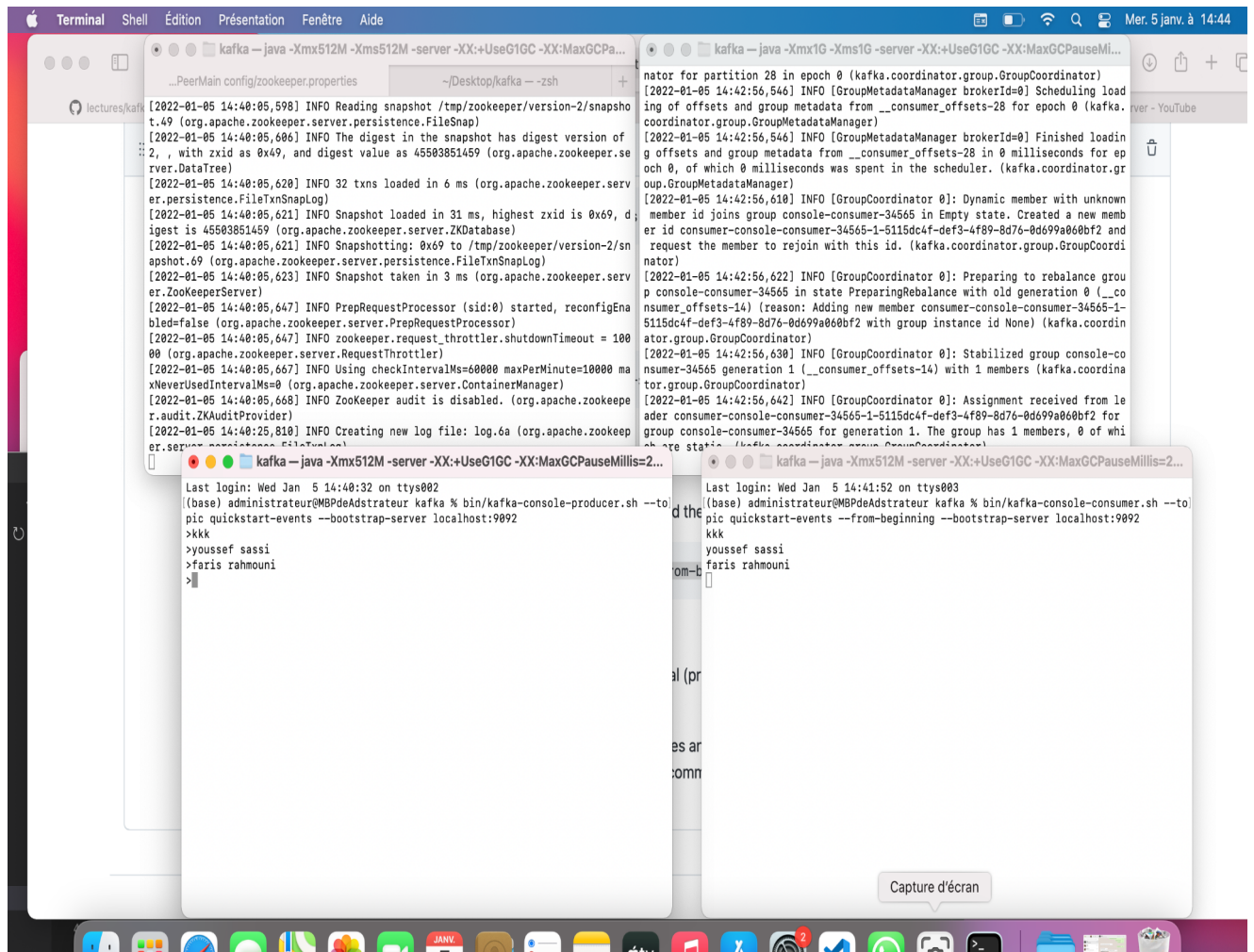
Capture d'écran

b) Streaming fichier .csv

Afin de ne pas perdre trop de temps sur le problème rencontré avec tweepy, nous avons décidé d'utiliser une source différente pour le streaming de données et nous allons utiliser un fichier au format csv qui contient des données.

4. Pipeline Kafka - Spark

Voici l'étape qui nous à pour l'instant demandé le plus de travail et de surmonter les difficultés lors de ce projet, relier kafka et spark, cependant après avoir sollicité le professeur et passer des heures, nous avons réussi à connecter le serveur consumer (Kafka) avec le producer (Spark) avec la configuration d'adressage ip : 192.168.33.13 . Nous pouvons envoyer des messages de la fenêtre consumer vers le produit , et à partir de cette étape on peut indexer les données de twitter (scrapping tweets) avec un code python vers le kafka.



```

Last login: Wed Jan  5 14:42:33 on ttys004
[(base) administrateur@MBPdeAdstrateur ~ % spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/01/05 15:22:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://mbpdeadstrateur:4040
Spark context available as 'sc' (master = local[*], app id = local-164139258119).
Spark session available as 'spark'.
Welcome to

      /---\
     /   \_--\
    /___\_/ .__/\_\_/ / / \_\_\_ version 3.2.0
   /___\_/ .__/\_\_/ / / \_\_\_
  /___\_/ .__/\_\_/ / / \_\_\_

Using Scala version 2.12.15 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_311)
Type in expressions to have them evaluated.
Type :help for more information.
```

On va faire le lien entre kafka et spark, nous allons configurer Spark en lui adressant une adresse dans le même sous réseau que Kafka à savoir **192.168.33.12** et finalement nous allons utiliser vagrant, d'où la nécessité d'avoir une vm

Nous allons lancer le framework spark à l'aide de la commande suivante :

```
/usr/local/spark/bin/spark-submit spark-streaming-kafka.py
```

et un host avec une adresse : **192.168.33.1**

5. Ce qu'il nous reste à faire :

Nous avons jusqu'à présent réussi à faire les parties que nous estimons les plus difficiles de ce projet et cela nous a permis d'accroître notre motivation ainsi que le temps passé sur ce projet, il nous reste à connecter la base de données afin de sauvegarder toutes les données que nous récoltons et puis de relier Superset à cette base de données afin d'avoir un graphique explicite de ce qu'il se passe au niveau des données en temps réel.