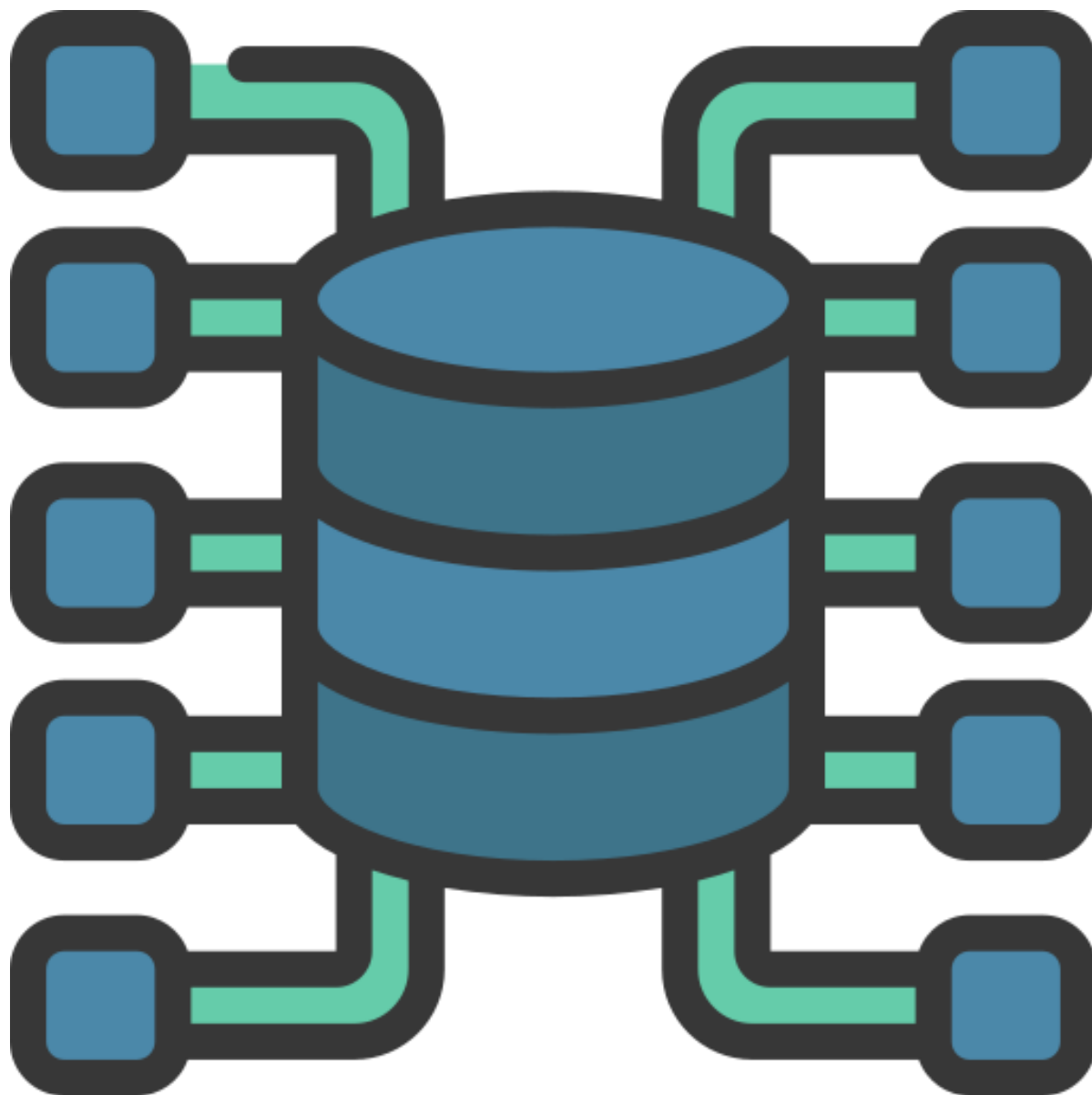




Rapport Big Data



Auteur :

NAJMI Mehdi
COLLATY Srikanth
BUI Sylvain

Table des matières

<i>Introduction</i>	<i>4</i>
<i>1. Problématique.....</i>	<i>4</i>
<i>2. Outils.....</i>	<i>4</i>
<i>3. Architecture.....</i>	<i>6</i>
<i>4. Déploiement.....</i>	<i>16</i>
<i>5. Résultat.....</i>	<i>21</i>
<i>6. Problèmes rencontrés et améliorations</i>	<i>25</i>
<i>Conclusion.....</i>	<i>26</i>

Introduction

Dans le cadre de notre formation, nous devons mettre en place une architecture permettant de traiter les données avec les outils du big data vu en classe.

1. Problématique

Le but de ce projet est de déterminer les films qui ont obtenu une note au box-office, les acteurs ayant le plus joués en se basant uniquement sur les données issues de chez IMDB.

2. Outils



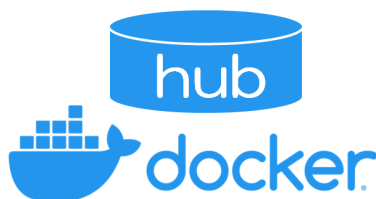
Grafana



THE BOURNE-AGAIN SHELL



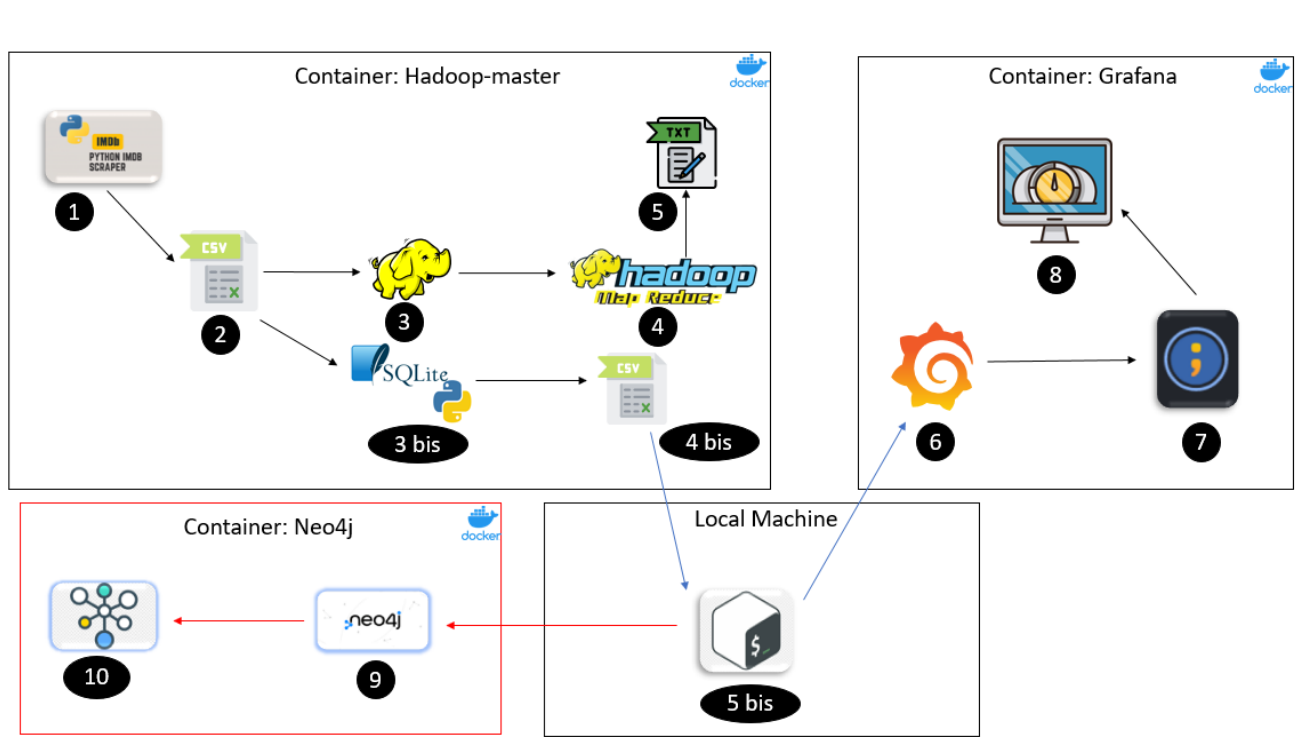
The #1 Database for Connected Data



Les raisons pour laquelle nous avons choisies ses outils :

- **Le langage de programmation python** : Nous avons choisie Python car c'est un de langages de programmation faciles à apprendre et comprendre et surtout parce que c'est un langage populaire pour le big data.
- **Hadoop** : Dans notre projet nous utilisons Hadoop pour stocker les données.
- **SQLite** : nous utilisons SQLite dans du python car l'implémentation est plus simple à mettre en place que Spark SQL et nous avons choisie SQLite pour ces performances.
- **Docker** : Pour que le groupe puisse collaborer nous avons décidé de développer le projet dans un container, qui est portable et léger.
- **Grafana** : Pour afficher les Dashboard, nous avons choisie d'utiliser Grafana.
- **Bash** : Nous utilisons des scripts Bash pour l'automatisation des tâches.
- **Docker hub** : Nous avons décidé de sauvegarder notre avancement sur docker hub est aussi qui nous servira de rendu final.
- **Crontab** : Nous utilisons pour exécuter automatiquement des scripts.
- **Neo4j** : Nous avons choisie de déployer un conteneur Neo4j pour afficher des graphes à partir d'un csv.

3. Architecture



Explication de l'architecture pour répondre à notre problématique

Etape 1 : A l'aide d'un script python, nous avons scraper le site [IMDB](#) pour récolter les informations sur les fils quels que, le classement, le nom du film, les noms des acteurs, la note obtenue et l'années de sortie des films.

```
root@hadoop-master:~# cat Webscrapping.py
#!/usr/bin/env python3

from bs4 import BeautifulSoup
import requests
import re
import pandas as pd

# Downloading imdb top 250 movie's data
url = 'http://www.imdb.com/chart/top'
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")
movies = soup.select('td.titleColumn')
crew = [a.attrs.get('title') for a in soup.select('td.titleColumn a')]
ratings = [b.attrs.get('data-value')
            for b in soup.select('td.posterColumn span[name=ir]')]

# create a empty list for storing
# movie information
list = []

# Iterating over movies to extract
# each movie's details
```

```
root@hadoop-master: ~
File Edit View Search Terminal Help

# Iterating over movies to extract
# each movie's details
for index in range(0, len(movies)):

    # Separating movie into: 'place',
    # 'title', 'year'
    movie_string = movies[index].get_text()
    movie = (' '.join(movie_string.split()).replace('.', ''))
    movie_title = movie[len(str(index))+1:-7]
    year = re.search('\((.*?)\)', movie_string).group(1)
    place = movie[:len(str(index))-(len(movie))]
    data = {"place": place,
            "movie_title": movie_title,
            "rating": ratings[index],
            "year": year,
            "star_cast": crew[index],
            }
    list.append(data)

# printing movie details with its rating.
# for movie in list:
#     print(movie['place'], '-', movie['movie_title'], '('+movie['year'] +
#           ') - ', 'Starring:', movie['star_cast'], movie['rating'])

df = pd.DataFrame(list)
df.to_csv('/root/bigdata/imdb.csv', index=False)
df.to_csv('/root/imdb.csv', index=False)
```

Les captures au-dessus correspondent au scrapping pour le site IMDB.

Etape 2 : Après avoir effectué le scrapping, les informations sont stockées dans un csv.

```
root@hadoop-master:~# ls -l
total 444108
-rw-rw-r-- 1 1000 1000      1499 Jan 30 09:12 Webscrapping.py
drwxr-xr-x 4 root root      4096 Jan 30 09:08 bigdata
-rw-r--r-- 1 root root       673 Dec  5 16:37 derby.log
drwxr-xr-x 1 root root      4096 Feb 22  2019 hdfs
-rw-r--r-- 1 root root    24642 Jan 30 15:15 imdb.csv
-rw-r--r-- 1 root root       0 Jan 27 12:14 log.txt
```

Le fichier imdb.csv a bien été créé.

Etape 3 : Après avoir stocker les informations dans un csv, nous avons stocker le csv dans le HDFS.

```
root@hadoop-master: ~/bigdata
File Edit View Search Terminal Help
-rw-r--r-- 1 root root      0 Jan 27 10:50 test2.txt
root@hadoop-master:~/bigdata# hdfs dfs -ls input
Found 3 items
-rw-r--r--  2 root supergroup    24643 2023-01-28 15:58 input/imdb.csv
-rw-r--r--  2 root supergroup      0 2023-01-27 10:42 input/test1.txt
-rw-r--r--  2 root supergroup      0 2023-01-27 10:50 input/test2.txt
```

Nous avons bien Importé le csv dans HDFS.

Etape 4 : Par la suite nous avons créé un script MapReduce avec python pour compter le nombre de mots dans le csv.

Le script MapReduce est à retrouver sur GitHub.

```
root@hadoop-master:~/bigdata# cat imdb.csv | python map.py | sort -k1,1 | python reduce.py
```

Voici au-dessus la commande exécuter pour le MapReduce.


```

Activities  MATE Terminal  janv. 28 18:48  🔔
root@hadoop-master: ~/bigdata

File Edit View Search Terminal Help

si      1
silence 1
soldat  1
sous    1
sud,132,8.17693987119289,\"Peter 1
sur     4
temps   1
the     2
tombeau 1
troisi?me      1
trousses,98,8.246569479692095,\"Alfred 1
truand\",1,8.78990067892573,\"Sergio 1
tu      2
tueras  1
un      1
une     2
vendetta,158,8.12321528197811,\"James 1
vengeance,165,8.111614500636831,\"Martin 1
vent,159,8.122423022674454,\"Victor 1
vers    2
verte,27,8.562826554576299,\"Frank 1
vie     3
vie,225,8.027623492906033,\"William 1
ville,52,8.432782277042548,\"Charles 1
voisin  1
von     3
voyage  1
yeux,163,8.117796383229123,\"Juan 1
root@hadoop-master:~/bigdata#

```

Nous obtenons bien le comptage des mots contenue dans le csv.

Etape 5 : Le résultat du comptage des mots sera enregistré dans un fichier txt comme on peut le voir dans l'architecture au-dessus.

Etape 3 bis : Dans cette étape nous avons décidé d'utiliser le csv du scrapping pour faire des requêtes SQL implémenter avec du python. Pour cela nous utilisons SQLite comme base de données.

```

root@hadoop-master: ~/bigdata
File Edit View Search Terminal Help
root@hadoop-master:~# cd bigdata/
root@hadoop-master:~/bigdata# cat database.py
#!/usr/bin/env python3
import sqlite3
import csv
import pandas as pd

con = sqlite3.connect(":memory:") # change to 'sqlite:///your_filename.db'
cur = con.cursor()
cur.execute("CREATE TABLE imdb (movie_title, place, rating, star_cast, year);") # use your column names here

with open('imdb.csv', 'r') as fin: # `with` statement available in 2.5+
    # csv.DictReader uses first line in file for column headings by default
    dr = csv.DictReader(fin) # comma is default delimiter
    to_db = [(i['movie_title'], i['place'], i['rating'], i['star_cast'], i['year']) for i in dr]
cur.executemany("INSERT INTO imdb (movie_title, place, rating, star_cast, year) VALUES (?, ?, ?, ?, ?);", to_db)
# print(to_db)
# display all the database
data = cur.execute('SELECT * FROM imdb').fetchall()
with open('output.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['movie_title', 'place', 'rating', 'star_cast', 'year'])
    writer.writerows(data)
# find specific director
data2 = cur.execute('SELECT movie_title, star_cast FROM imdb WHERE star_cast LIKE "%James Cameron%"').fetchall()

```

```

root@hadoop-master: ~/bigdata
File Edit View Search Terminal Help
data4 = cur.execute('SELECT movie_title, rating FROM imdb WHERE rating BETWEEN '5' AND '9').fetchall()
with open('output4.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['movie_title', 'rating'])
    writer.writerows(data4)
# rating > 9
data5 = cur.execute('SELECT movie_title, rating FROM imdb WHERE rating > '9').fetchall()
with open('output5.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['movie_title', 'rating'])
    writer.writerows(data5)
# number of movies per year
data6 = cur.execute('SELECT year, COUNT(*) FROM imdb GROUP BY year HAVING COUNT(*) > 1').fetchall()
with open('output6.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['year', 'count'])
    writer.writerows(data6)
data7 = cur.execute('SELECT star_cast, COUNT(*) FROM imdb GROUP BY star_cast HAVING COUNT(*) > 0').fetchall()
with open('output7.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(['star_cast', 'count'])
    writer.writerows(data7)
con.commit()
con.close()
root@hadoop-master:~/bigdata#

```

Les 2 captures au-dessus correspondent aux requêtes SQL stocker dans 7 csv qui correspondent aux 7 requêtes implémentées.

Etape 4 bis : Après avoir mis en place des requêtes SQL avec un script python, nous avons décidé de stocker les résultats des requêtes (calculs, traitements données) dans des nouveaux csv. Au total dans cette étape nous avons 7 csv ce qui correspond à 7 requêtes.

```

root@hadoop-master: ~
File Edit View Search Terminal Help
2384 SecondaryNameNode
2544 ResourceManager
3252 Jps
root@hadoop-master:~# ls -l
total 444108
-rw-rw-r-- 1 1000 1000      1499 Jan 30 09:12 Webscrapping.py
drwxr-xr-x 4 root root      4096 Jan 30 09:08 bigdata
-rw-r--r-- 1 root root        673 Dec  5 16:37 derby.log
drwxr-xr-x 1 root root      4096 Feb 22  2019 hdfs
-rw-r--r-- 1 root root     24642 Jan 30 15:33 imdb.csv
-rw-r--r-- 1 root root         0 Jan 27 12:14 log.txt
drwxr-xr-x 5 root root      4096 Dec  5 16:37 metastore_db
-rw-r--r-- 1 root root     24893 Jan 30 15:33 output.csv
-rw-r--r-- 1 root root        271 Jan 30 15:33 output2.csv
-rw-r--r-- 1 root root        120 Jan 30 15:33 output3.csv
-rw-r--r-- 1 root root       9132 Jan 30 15:33 output4.csv
-rw-r--r-- 1 root root         80 Jan 30 15:33 output5.csv
-rw-r--r-- 1 root root        532 Jan 30 15:33 output6.csv
-rw-r--r-- 1 root root     14070 Jan 30 15:33 output7.csv

```

Les 7 csv ont bien été généré.

Etape 5 bis : Le but est d'envoyer les 7 csv dans un conteneur Grafana pour cela nous avons décidé de faire un script Bash sur un pc local qui va récupérer les 7 csv sur le conteneur Hadoop-master et les envoyer dans le conteneur Grafana grâce à la commande « docker cp ».

```

root@ubuntudata: /home/sc
File Edit View Search Terminal Help
root@hadoop-master:~# exit
exit
root@ubuntudata:/home/sc# cat file.sh
#!/bin/bash

docker cp e798f045a56e:/root/output.csv /
docker cp e798f045a56e:/root/output2.csv /
docker cp e798f045a56e:/root/output3.csv /
docker cp e798f045a56e:/root/output4.csv /
docker cp e798f045a56e:/root/output5.csv /
docker cp e798f045a56e:/root/output6.csv /
docker cp e798f045a56e:/root/output7.csv /

docker cp /output.csv d6b9e9e1081c:/
docker cp /output2.csv d6b9e9e1081c:/
docker cp /output3.csv d6b9e9e1081c:/
docker cp /output4.csv d6b9e9e1081c:/
docker cp /output5.csv d6b9e9e1081c:/
docker cp /output6.csv d6b9e9e1081c:/
docker cp /output7.csv d6b9e9e1081c:/
root@ubuntudata:/home/sc# █

```

Script Bash pour récupérer et envoyer les fichiers csv.

Etape 6 : Une fois dans le conteneur grafana, nous devons faire quelques configurations, pour le déployer en localhost.

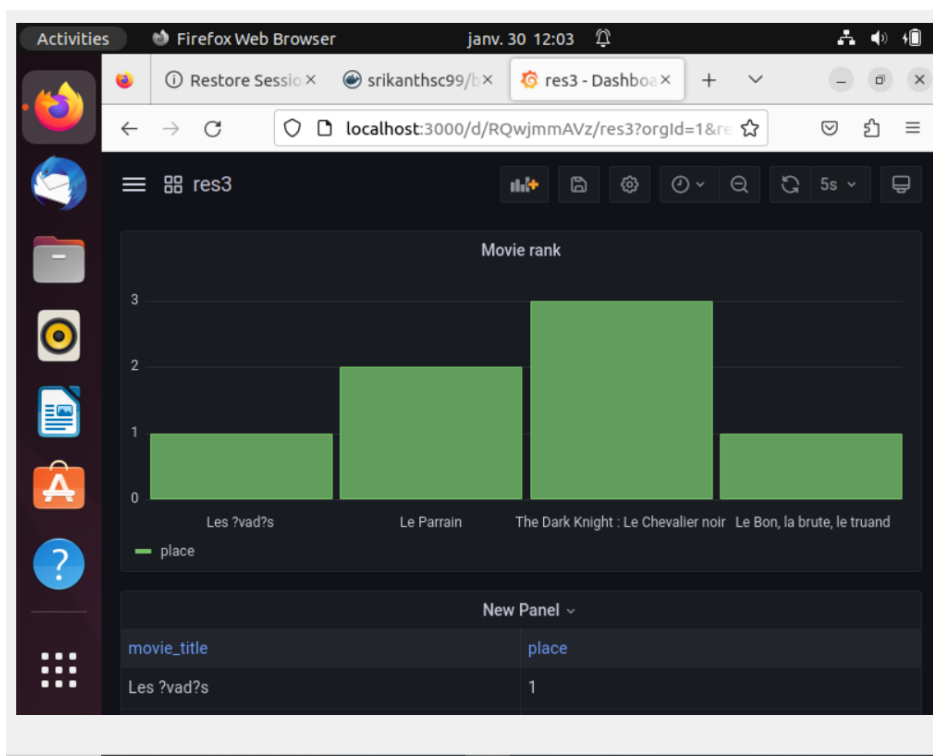
Voici la commande utiliser pour la configuration de port et l'installation de grafana. « `docker run -d -p 3000:3000 grafana/grafana-oss` ».

Etape 7 : Toujours dans le conteneur grafana, nous devons télécharger le plugin CSV data source for Grafana pour exploiter nos 7 csv. Voici leur documentation d'installation et de configuration : <https://grafana.github.io/grafana-csv-datasource/>

Dans notre cas pour pouvoir télécharger le plugin pour le csv, nous avons dû accéder à l'intérieur du conteneur grafana pour installer le plugin mentionner précédemment.

Voici la commande : « grafana-cli plugins install marcusolsson-csv-datasource ».

Etape 8 : Apres avoir effectuer les configurations nécessaires, nous importons les 7 csv pour pouvoir faire des Dashboard.



Exemple de dashboard sur l'un des 7 csv.

Les étapes citées au-dessus correspondent à l'architecture que nous avons réellement mis en place.

Les scripts Bash et python sont automatisés grâce à crontab pour actualiser les données en continues.




```

root@ubuntudata: /home/sc
File Edit View Search Terminal Help

docker cp /output.csv d6b9e9e1081c:/
docker cp /output2.csv d6b9e9e1081c:/
docker cp /output3.csv d6b9e9e1081c:/
docker cp /output4.csv d6b9e9e1081c:/
docker cp /output5.csv d6b9e9e1081c:/
docker cp /output6.csv d6b9e9e1081c:/
docker cp /output7.csv d6b9e9e1081c:/
root@ubuntudata:/home/sc# docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS      PORTS
dae13283fd09   docker                              "docker-entrypoint.s..." 18 hours ago   Exited (127)
s
d6b9e9e1081c   grafana/grafana-oss                "/run.sh"               20 hours ago   Up         0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
88689604747c   totofunku/bigdata-cours:latest     "sh -c 'service ssh ..." 7 weeks ago    Up         0.0.0.0:8041->8042/tcp, :::8041->8042/tcp
f89f9f269e0d   totofunku/bigdata-cours:latest     "sh -c 'service ssh ..." 7 weeks ago    Up         0.0.0.0:8040->8042/tcp, :::8040->8042/tcp
e798f045a56e   totofunku/bigdata-cours:latest     "sh -c 'service ssh ..." 7 weeks ago    Up         0.0.0.0:7077->7077/tcp, :::7077->7077/tcp, 0.0.0.0:16010->16010/tcp, :::16010->16010/tcp, 0.0.0.0:50070->50070/tcp, :::50070->50070/tcp
root@ubuntudata:/home/sc#

```

On peut voir que les conteneurs hadoop-master et grafana sont en marche

```

root@hadoop-master:~# crontab -l
*/1 * * * * /usr/bin/python3 /root/Webscrapping.py
*/1 * * * * /usr/bin/python3 /root/bigdata/database.py >> /root/log.txt
*/1 * * * * ./root/bigdata/script.sh
*/1 * * * * /usr/local/hadoop/bin/hdfs dfs -put -f /root/bigdata/imdb.csv input

root@hadoop-master:~#

root@ubuntudata:/home/sc# crontab -l
*/1 * * * * /home/sc/file.sh

```

Dans les 2 captures au-dessus, nous avons mis en place crontab pour automatiser les scripts

Le conteneur Neo4j est un plus dans notre architecture, que nous n'avons pas mis en place techniquement.

Etape 9 : Si on l'avait réalisé réellement, le pc local va envoyer les csv sur le conteneur Neo4j grâce à un script Bash.

Etape 10 : Après avoir importer les csv, nous pouvons faire des graphes.

Concernant les sauvegardes : Nous avons décidé de sauvegarder sur docker hub et GitHub.

```
root@ubuntudata:/home/sc# docker image rm bigdata/bigdata2:v2
Untagged: bigdata/bigdata2:v2
Deleted: sha256:a72b1010547c9d03c6a86dc21db0c537c2717c1487350bb4f5d2f97a1a4faa0f
Deleted: sha256:503951f54a84fe0ec2ac5d97b6c1b42fd93d66e2e1160d21afe99da3f752aa46
root@ubuntudata:/home/sc# docker commit e798f045a56e srikanthsc99/bigdata2:v2
sha256:207414e231b9531bcd8b5f7960fa71d1ac41bb74477687dc16426e774f66a96a
root@ubuntudata:/home/sc# docker push srikanthsc99/bigdata2:v2
The push refers to repository [docker.io/srikanthsc99/bigdata2]
0269717f4447: Pushed
0eda749afd77: Pushed
55cff7bc8ce0: Pushed
25fd6e0719bd: Pushed
642eaf745c3e: Pushed
ddcf50d80853: Pushed
6c68cab27947: Pushed
daa17a9131ed: Pushed
b9cfea02b7b1: Pushed
6f9965be7792: Pushed
256dfbd8fbad: Pushed
64e7ac095720: Pushed
e81092a6c7a0: Pushed
6f4ce6b88849: Pushed
92914665e7f6: Pushed
c98ef191df4b: Pushed
9c7183e0ea88: Pushed
ff986b10a018: Pushed
v2: digest: sha256:10945e17f4440adf44c958612351120f028d4dd8f4a2d9a6ff0a9f1fce2143d9 size:
4093
root@ubuntudata:/home/sc#
```

Le push a bien été effectués dans le docker hub.

4. Déploiement

Voici le détail du déploiement :

1. Télécharger les images à l'adresse suivantes :

- Container hadoop-master :

<https://hub.docker.com/repository/docker/srikanthsc99/hadoop-master/general>

Avec la commande suivante:

“docker push srikanthsc99/hadoop-master:v3”

- Container Grafana:

<https://hub.docker.com/repository/docker/srikanthsc99/grafanabigdata/general>

Avec la commande suivante:

```
See 'docker run --help'.
root@ubuntudata:/home/sc# docker run -d -p 3000:3000 srikanthsc99/grafanabigdata
e915dc5f56786cbaee0d7ad3028b5e20f397be0178c594574615bfdd8789b504
root@ubuntudata:/home/sc#
```

Avec cette commande grafana et active à l'adresse suivante :
localhost :3000

- Sur le pc local télécharger le script bash avec la commande git clone le fichier file.sh (« git clone git@github.com:srikanthsc/bigdatascript.git ») qui permet d'envoyer les csv d'un conteneur a un autre, à l'adresse suivante :

<https://github.com/srikanthsc/bigdatascript>

Voici les images téléchargées :

```
root@ubuntudata:/home/sc# docker ps
CONTAINER ID   IMAGE                                COMMAND      CREATED      STATUS      PORTS      NAMES
root@ubuntudata:/home/sc# docker images
REPOSITORY          TAG      IMAGE ID      CREATED      SIZE
srikanthsc99/grafanabigdata   latest   0ebf2da33557   2 minutes ago   334MB
srikanthsc99/hadoop-master    v3       acd32ad809b4   9 minutes ago   2.67GB
srikanthsc99/bigdata2        v2       207414e231b9   42 hours ago   2.45GB
docker                latest   ed2962a40289   3 days ago     155MB
grafana/grafana-oss         latest   3b2b2ecc5e78   4 days ago     317MB
tototunku/bigdata-cours     latest   d64a47823a96   3 years ago     1.94GB
root@ubuntudata:/home/sc# docker pull srikanthsc99/hadoop-master:v3
v3: Pulling from srikanthsc99/hadoop-master
Digest: sha256:ed1d80671aa4afb65374511bdb793b2264c2bb058edd3bdd1e707ee9536f1119
Status: Image is up to date for srikanthsc99/hadoop-master:v3
docker.io/srikanthsc99/hadoop-master:v3
```

- Pour rentrer dans le container hadoop-master, faire la commande suivante :

```
root@ubuntudata:/home/sc# docker run -it acd32ad809b4
* Starting OpenBSD Secure Shell server sshd
root@0c9834d6c3bf:~# ls -l
total 444108
-rw-rw-r-- 1 1000 1000      1499 Jan 30 09:12 Webscrapping.py
drwxr-xr-x 4 root root      4096 Jan 30 09:08 bigdata
-rw-r--r-- 1 root root       673 Dec  5 16:37 derby.log
drwxr-xr-x 1 root root      4096 Feb 22  2019 hdfs
-rw-r--r-- 1 root root     24645 Jan 30 17:22 imdb.csv
-rw-r--r-- 1 root root         0 Jan 27 12:14 log.txt
drwxr-xr-x 5 root root      4096 Dec  5 16:37 metastore_db
-rw-r--r-- 1 root root     24896 Jan 30 17:22 output.csv
-rw-r--r-- 1 root root       271 Jan 30 17:22 output2.csv
-rw-r--r-- 1 root root       120 Jan 30 17:22 output3.csv
-rw-r--r-- 1 root root      9137 Jan 30 17:22 output4.csv
-rw-r--r-- 1 root root        78 Jan 30 17:22 output5.csv
-rw-r--r-- 1 root root       532 Jan 30 17:22 output6.csv
-rw-r--r-- 1 root root     14070 Jan 30 17:22 output7.csv
-rw-r--r-- 1 root root 211312924 Feb  8  2017 purchases.txt
-rw-r--r-- 1 root root 243309628 Apr  9  2018 purchases2.txt
-rwxr-xr-x 1 root root       695 Mar  4  2018 run-wordcount.sh
-rw-r--r-- 1 root root         0 Jan 27 13:10 script.sh
-rwxr-xr-x 1 root root       120 Mar  4  2018 start-hadoop.sh
-rwxr-xr-x 1 root root       218 Mar  4  2018 start-kafka-zookeeper.sh
-rw-r--r-- 1 root root         0 Jan 27 10:02 test.txt
root@0c9834d6c3bf:~# service cron start
* Starting periodic command scheduler cron
root@0c9834d6c3bf:~#
```

« Docker run -it <image-id> », comme on peut le voir on retrouver bien les codes sources et les csv (output.csv, output2.csv, output3.csv, output4.csv, output5.csv, output6.csv, output7.csv et imdb.csv)

- Dans de conteneur hadoop-master exécuter crontab pour exécuter les scripts automatiquement :

```
root@0c9834d6c3bf:~# service cron start
* Starting periodic command scheduler cron
root@0c9834d6c3bf:~#
```

- Dans le pc local activer crontab avec la commande suivante :
« service cron start » pour automatiser le fichier file.sh

Vérifier dans le fichier file.sh que l'id du conteneur son correctes, si ce n'est pas le cas, faire la commande « docker ps -a » pour avoir les nouveaux id des conteneurs. (Entouré en rouge)

```
root@ubuntu1404:/home/sc# docker ps -a
```

CONTAINER ID	IMAGE	PORTS	COMMAND	CREATED	NAMES
e915dc5f5678	srikanthsc99/grafanabigdata	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	"/run.sh"	About an hour ago	kind_
5fbc67051f27	0ebf2da33557		"/run.sh"	About an hour ago	noyce
cc02c173dafd	acd32ad809b4		"sh -c 'service ssh ..."	About an hour ago	us nash

Dans le cas les id conteneurs ont changer, éditer le fichier file.sh

```
#!/bin/bash

docker cp cc02c173dafd:/root/output.csv /
docker cp cc02c173dafd:/root/output2.csv /
docker cp cc02c173dafd:/root/output3.csv /
docker cp cc02c173dafd:/root/output4.csv /
docker cp cc02c173dafd:/root/output5.csv /
docker cp cc02c173dafd:/root/output6.csv /
docker cp cc02c173dafd:/root/output7.csv /

docker cp /output.csv e915dc5f5678:/
docker cp /output2.csv e915dc5f5678:/
docker cp /output3.csv e915dc5f5678:/
docker cp /output4.csv e915dc5f5678:/
docker cp /output5.csv e915dc5f5678:/
docker cp /output6.csv e915dc5f5678:/
docker cp /output7.csv e915dc5f5678:/
root@ubuntu14:~# docker ps -a
```

Id conteneur Hadoop-master

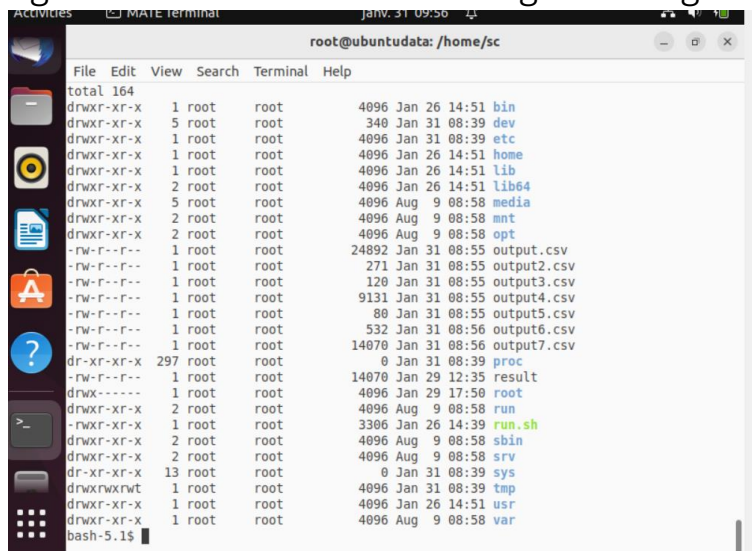
Id conteneur grafanabigdata

- Après avoir effectué les configurations nécessaires pour le déploiement, nous pouvons voir que dans les nouveaux conteneurs les fichier csv se mettent à jour automatiquement, et les script s'exécutent aussi automatiquement.
Voici quelque capture :

Dans le conteneur hadoop-master les csv sont à jours, et on retrouve également les codes sources créés.

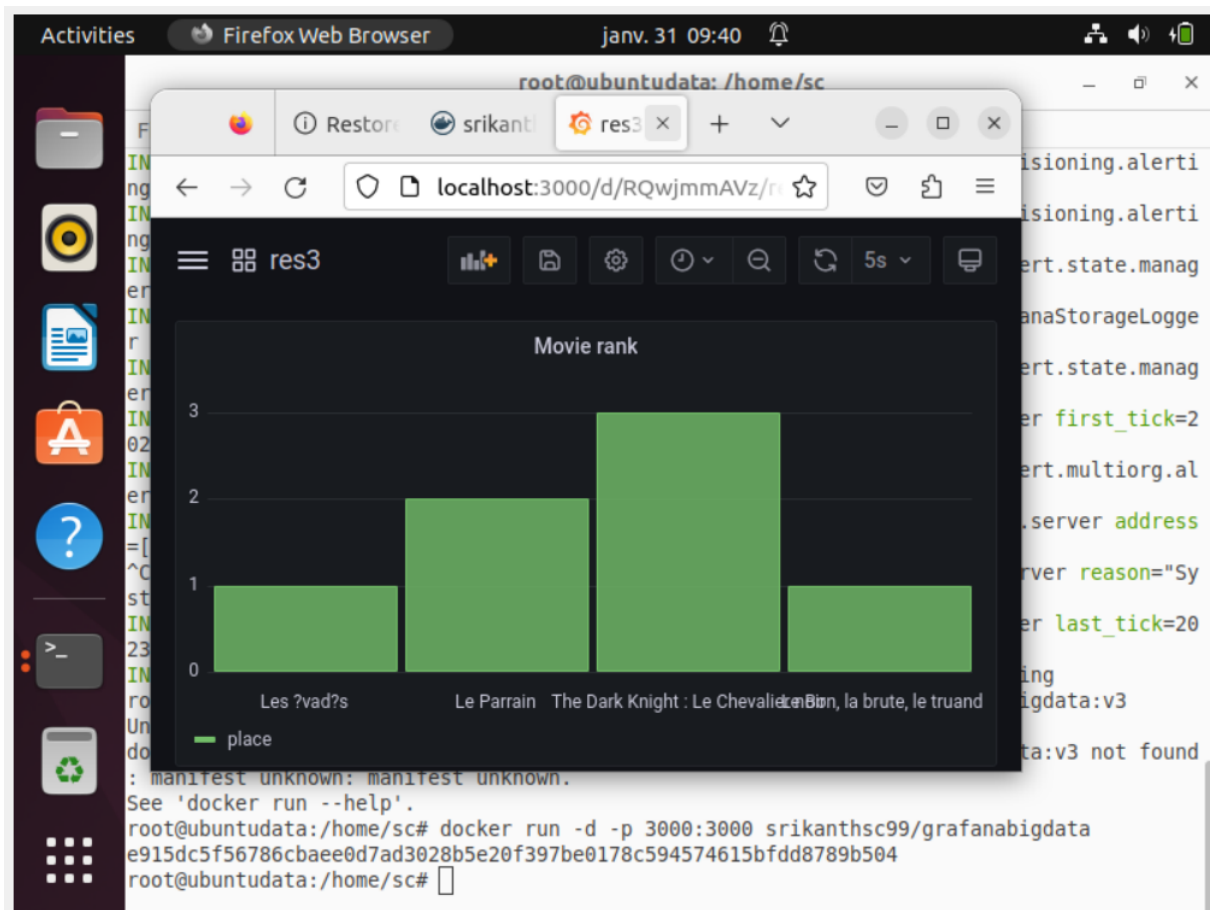

```
root@0c9834d6c3bf:~# ls -l
total 444112
-rw-rw-r-- 1 1000 1000      1499 Jan 30 09:12 Webscrapping.py
drwxr-xr-x 1 root root      4096 Jan 30 09:08 bigdata
-rw-r--r-- 1 root root        673 Dec  5 16:37 derby.log
drwxr-xr-x 1 root root      4096 Feb 22  2019 hdfs
-rw-r--r-- 1 root root     24641 Jan 31 08:30 imdb.csv
-rw-r--r-- 1 root root         0 Jan 27 12:14 log.txt
drwxr-xr-x 5 root root      4096 Dec  5 16:37 metastore_db
-rw-r--r-- 1 root root     24896 Jan 31 08:30 output.csv
-rw-r--r-- 1 root root        271 Jan 31 08:30 output2.csv
-rw-r--r-- 1 root root        120 Jan 31 08:30 output3.csv
-rw-r--r-- 1 root root       9137 Jan 31 08:30 output4.csv
-rw-r--r-- 1 root root         78 Jan 31 08:30 output5.csv
-rw-r--r-- 1 root root        532 Jan 31 08:30 output6.csv
-rw-r--r-- 1 root root     14070 Jan 31 08:30 output7.csv
-rw-r--r-- 1 root root 211312924 Feb  8  2017 purchases.txt
-rw-r--r-- 1 root root 243309628 Apr  9  2018 purchases2.txt
-rwxr-xr-x 1 root root        695 Mar  4  2018 run-wordcount.sh
-rw-r--r-- 1 root root         0 Jan 27 13:10 script.sh
-rwxr-xr-x 1 root root        120 Mar  4  2018 start-hadoop.sh
-rwxr-xr-x 1 root root        218 Mar  4  2018 start-kafka-zookeeper.sh
-rw-r--r-- 1 root root         0 Jan 27 10:02 test.txt
root@0c9834d6c3bf:~#
```

Également dans le conteneur grafanabigdata les csv sont aussi à jour.



```
root@ubuntudata: /home/sc
File Edit View Search Terminal Help
total 164
drwxr-xr-x 1 root root      4096 Jan 26 14:51 bin
drwxr-xr-x 5 root root      340 Jan 31 08:39 dev
drwxr-xr-x 1 root root      4096 Jan 31 08:39 etc
drwxr-xr-x 1 root root      4096 Jan 26 14:51 home
drwxr-xr-x 1 root root      4096 Jan 26 14:51 lib
drwxr-xr-x 2 root root      4096 Jan 26 14:51 lib64
drwxr-xr-x 5 root root      4096 Aug  9 08:58 media
drwxr-xr-x 2 root root      4096 Aug  9 08:58 mnt
drwxr-xr-x 2 root root      4096 Aug  9 08:58 opt
-rw-r--r-- 1 root root     24892 Jan 31 08:55 output.csv
-rw-r--r-- 1 root root        271 Jan 31 08:55 output2.csv
-rw-r--r-- 1 root root        120 Jan 31 08:55 output3.csv
-rw-r--r-- 1 root root       9131 Jan 31 08:55 output4.csv
-rw-r--r-- 1 root root         80 Jan 31 08:55 output5.csv
-rw-r--r-- 1 root root        532 Jan 31 08:56 output6.csv
-rw-r--r-- 1 root root     14070 Jan 31 08:56 output7.csv
dr-xr-xr-x 297 root root         0 Jan 31 08:39 proc
-rw-r--r-- 1 root root     14070 Jan 29 12:35 result
-rw-r--r-- 1 root root      4096 Jan 29 17:50 root
-rw-r--r-- 2 root root      4096 Aug  9 08:58 run
-rwxr-xr-x 1 root root      3306 Jan 26 14:39 run.sh
-rw-r--r-- 2 root root      4096 Aug  9 08:58 sbin
-rw-r--r-- 2 root root      4096 Aug  9 08:58 srv
-rw-r--r-- 13 root root         0 Jan 31 08:39 sys
drwxrwxrwt 1 root root      4096 Jan 31 08:39 tmp
drwxr-xr-x 1 root root      4096 Jan 26 14:51 usr
drwxr-xr-x 1 root root      4096 Aug  9 08:58 var
bash-5.1$
```

Grafana est bien été déployé à partir d'une nouvelle image.

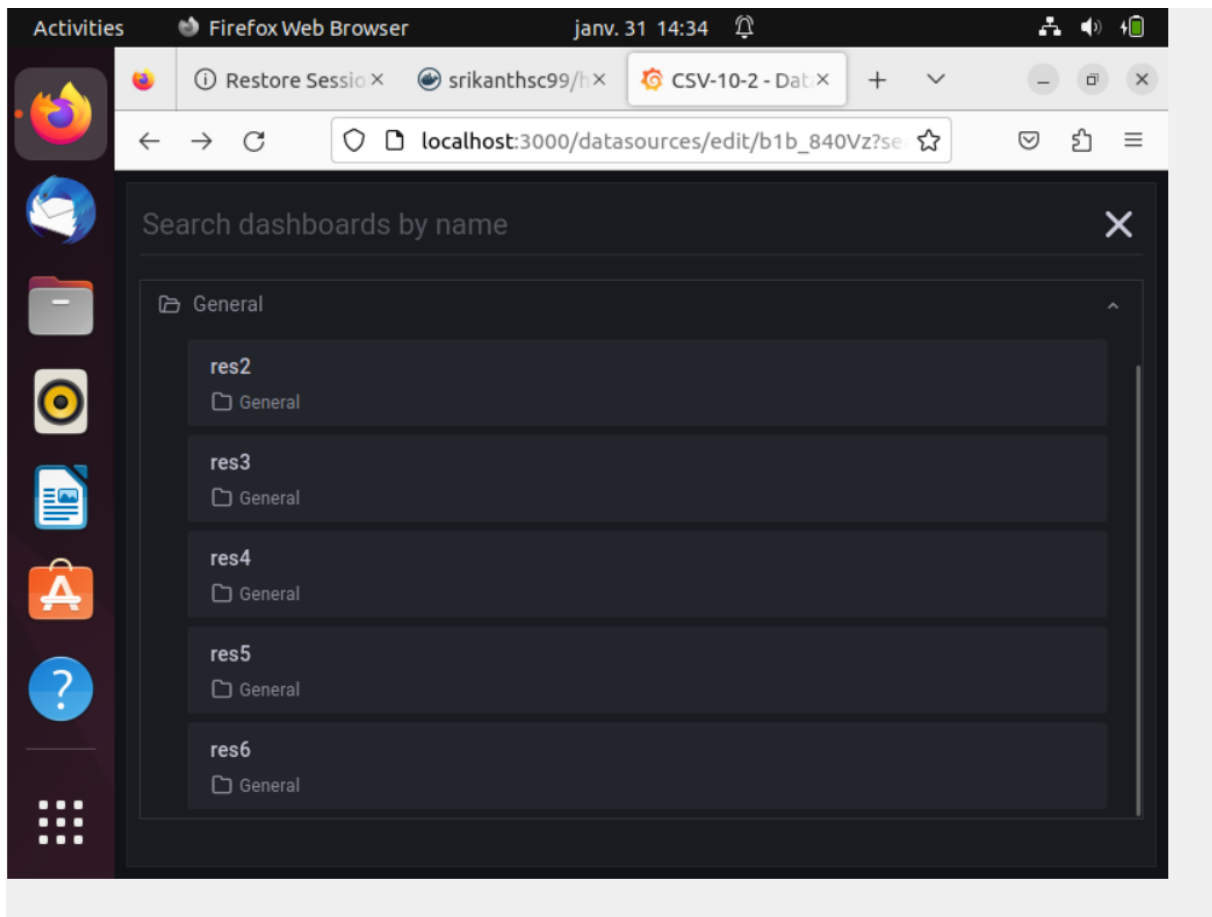


- Pour finir l'ensemble de l'architecture est fonctionnel est automatisé.

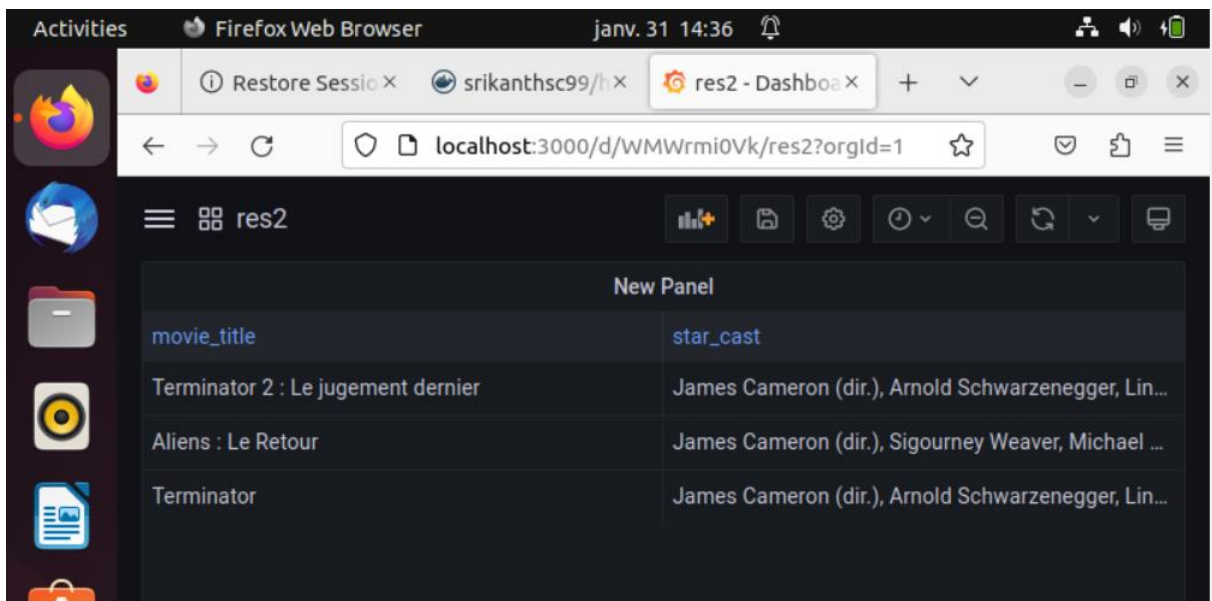
5. Résultat

Voici quelques Dashboard produit :

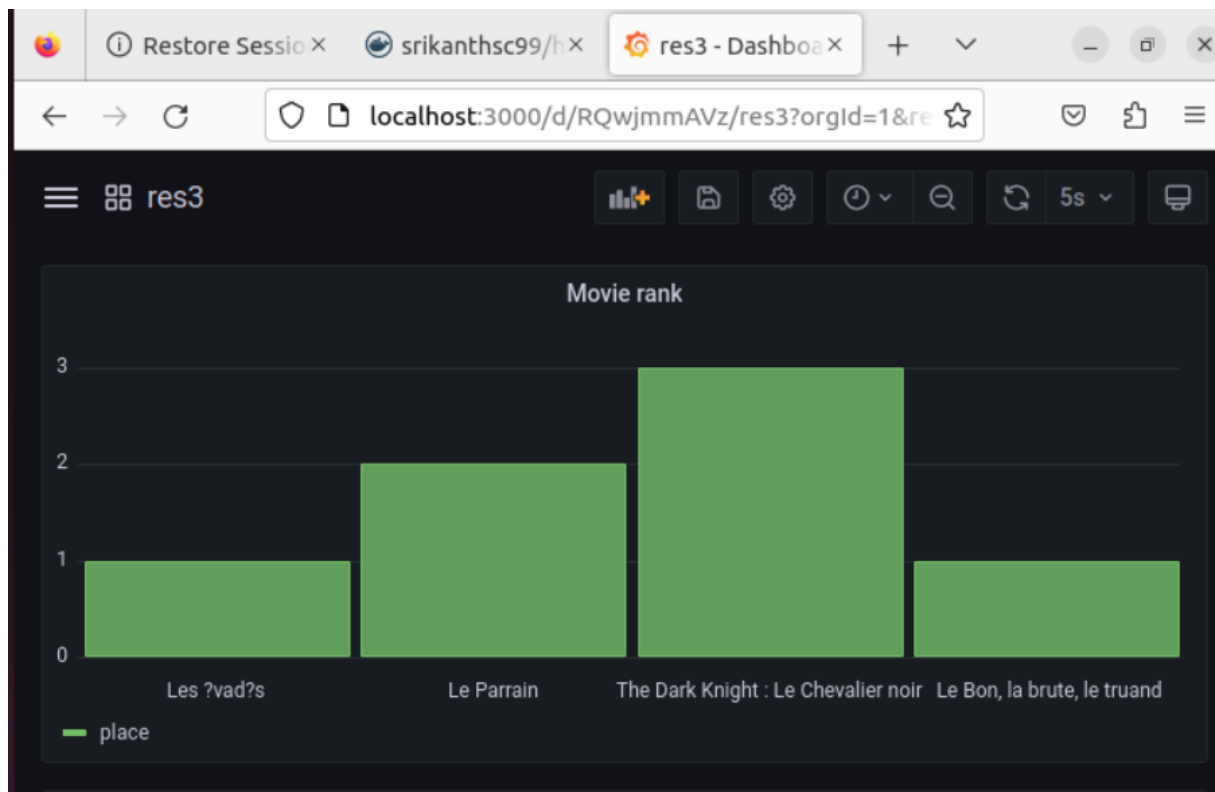
- Tous nos csv sont importés dans grafana et mis à jour automatiquement. (Conteneur grafana)



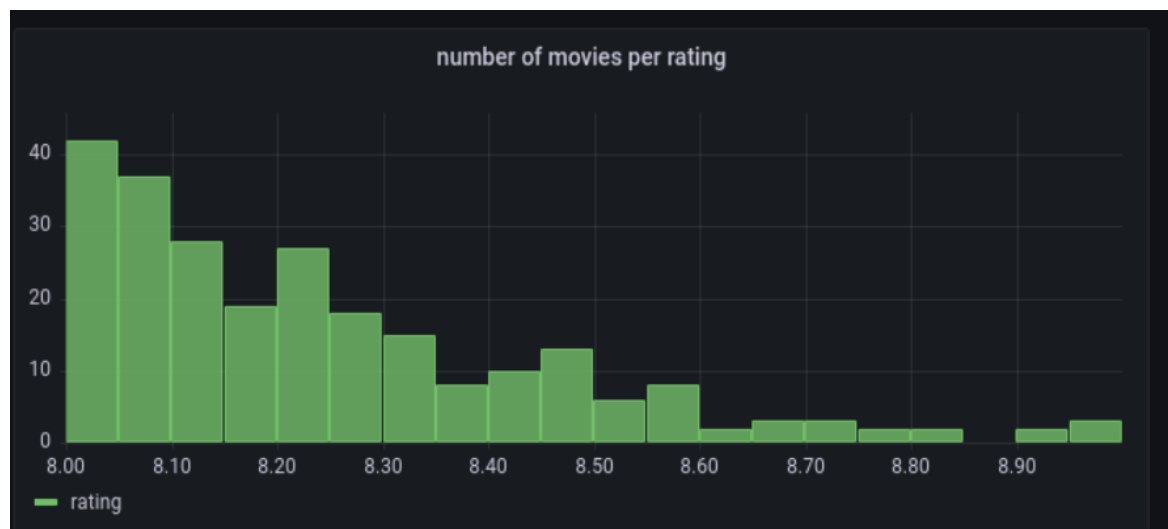
- Liste le film ou « James Cameron est impliqués »



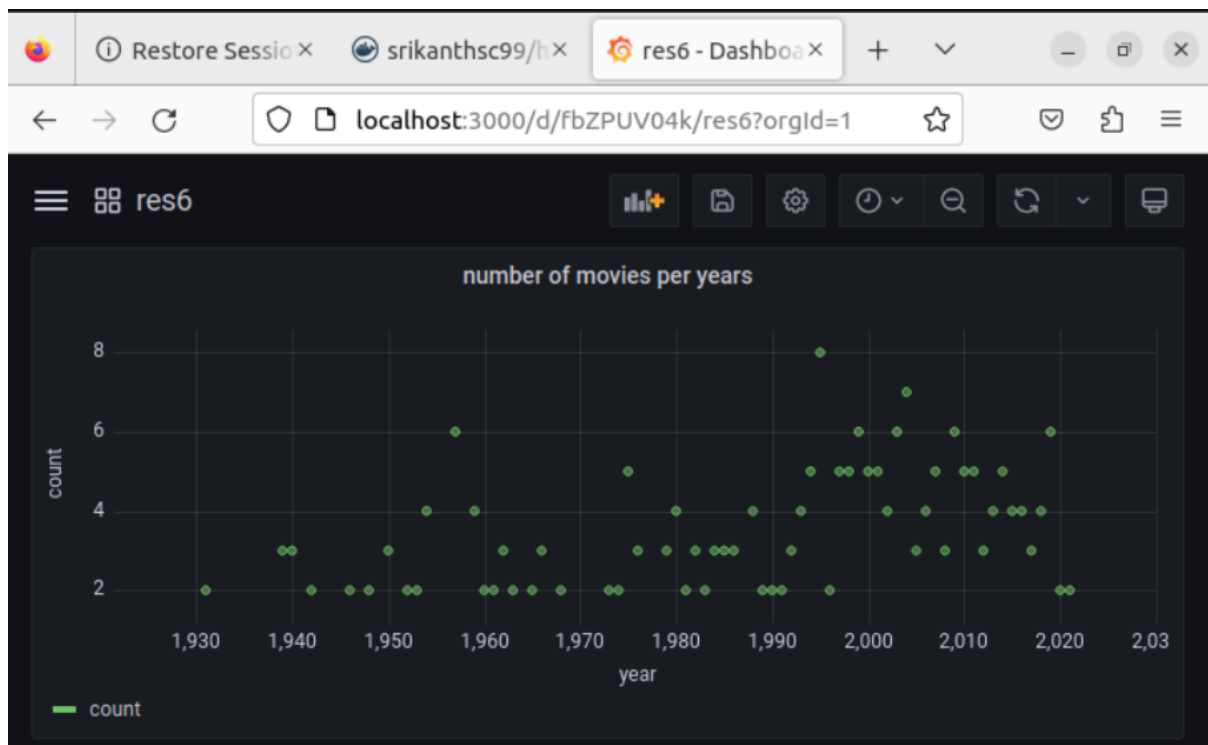
- Le rang des meilleurs films



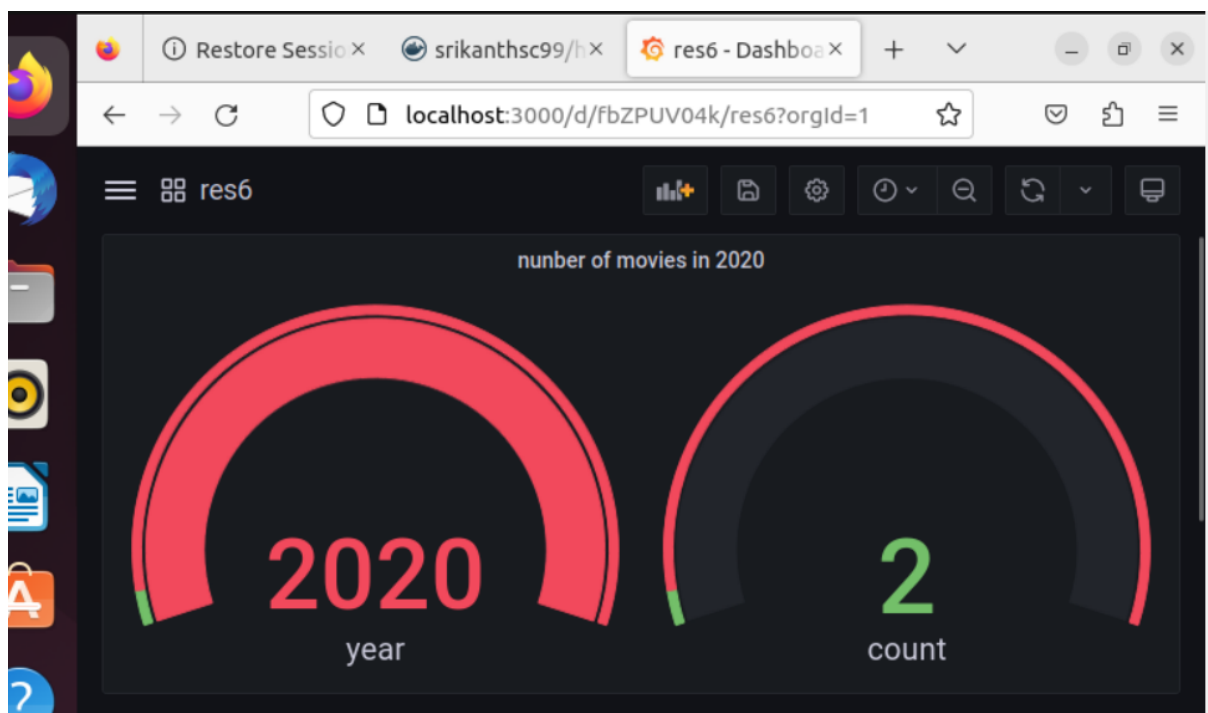
- Nombre de film par notes



- Nombre de film par années contenue dans le csv



- Nombre de films pour l'années 2020





Les 4 Dashboard présentés sont à retrouver sur une nouvelle image à l'adresse suivantes :

https://hub.docker.com/repository/docker/srikanthsc99/grafana-more_dashboard/general

La configuration et le déploiement de ce conteneur se fait de la même manière présenter précédemment dans le rapport

6. Problèmes rencontrés et améliorations

Problèmes rencontrés

Datanode : Avec la commande « jps », les datanodes n'apparaissent plus sur le terminal de temps en temps.

Amélioration

Un troisième conteneur Ubuntu : Pour faciliter la communication entre les conteneurs, on aurait pu utiliser une image Ubuntu, à la place d'utiliser le pc local.

Monter un volume : Pour envoyer les csv d'un conteneur à un autre on aurait pu mettre en place les volumes pour que les conteneurs puissent communiquer en eux, au lieu d'utiliser la commande « docker cp ».

Conclusion

Le module Big data architecture and data processing nous a permis d'apprendre les bases du big data. L'architecture qu'on vient de mettre en place est totalement automatisé même si quelques améliorations auraient pu être faites.

Voici les liens des rendues :

Conteneur hadoop-master:

<https://hub.docker.com/repository/docker/srikanthsc99/hadoop-master/general>

Conteneur Grafana:

<https://hub.docker.com/repository/docker/srikanthsc99/grafana-bigdata/general>

Script Bash à exécuter sur le pc local :

<https://github.com/srikanthsc/bigdatascript>

Conteneur grafana avec plus de Dashboard produit :

<https://hub.docker.com/repository/docker/srikanthsc99/grafana-more-dashboard/general>