Hadoop

Apache Hadoop est un framework open-source pour stocker et traiter les données volumineuses sur un cluster. Il est utilisé par un grand nombre de contributeurs et utilisateurs. Il a une licence Apache 2.0.

Hadoop et Docker :

Pour déployer le framework Hadoop, nous allons utiliser des contenaires Docker. L'utilisation des contenaires va garantir la consistance entre les environnements de développement et permettra de réduire considérablement la complexité de configuration des machines (dans le cas d'un accès natif) ainsi que la lourdeur d'exécution (si on opte pour l'utilisation d'une machine virtuelle).

Installation :

Nous allons utiliser pour ce projet trois contenaires représentant respectivement un noeud maître (Namenode) et deux noeuds esclaves (Datanodes).

Nous avons pour cela installé docker sur notre machine, et l'avoir correctement configuré.

Téléchargement de l'image docker uploadée sur dockerhub:

```
C:\Users\pc>docker pull liliasfaxi/spark-hadoop:hv-2.7.2
```

notre image est uploadée :

```
C:\Users\pc>docker images
REPOSITORY                TAG         IMAGE ID       CREATED         SIZE
redis                     latest      e10bd12f0b2d   2 months ago    138MB
postgres                  13          d28d5ae8775c   2 months ago    413MB
apache/airflow            2.7.3       56d2fbb018bb   2 months ago    1.54GB
alpine/git                latest      b337a04161f7   22 months ago   38.2MB
liliasfaxi/spark-hadoop   hv-2.7.2    d64a47823a96   4 years ago     1.94GB
```

Création du réseau nommé hadoop qui permettra de relier les trois contenaires:

```
C:\Users\pc>  docker network create --driver=bridge hadoop
```

le réseau hadoop a été bien crée :

```
C:\Users\pc>docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
f8ee19ffb55c   bridge    bridge    local
fc721d77bbdc   hadoop    bridge    local
64edab956fdd   host      host      local
f8eeff81af43   none      null      local
```

Création des conteneurs :

Conteneur **hadoop-master :**

```
C:\Users\pc>docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 7077:7077 -p 16010:16010 ^
           --name hadoop-master --hostname hadoop-master ^
           liliasfaxi/spark-hadoop:hv-2.7.2
```

Conteneur **hadoop-slave1 :**

```
C:\Users\pc>docker run -itd -p 8040:8042 --net=hadoop ^
            --name hadoop-slave1 --hostname hadoop-slave1 ^
            liliasfaxi/spark-hadoop:hv-2.7.2
```

Conteneur **hadoop-slave2 :**

```
C:\Users\pc>docker run -itd -p 8041:8042 --net=hadoop ^
            --name hadoop-slave2 --hostname hadoop-slave2 ^
           liliasfaxi/spark-hadoop:hv-2.7.2
```

les instructions -p permettent de faire un mapping entre les ports de la machine hôte et ceux du contenaire.

Les 3 Conteneurs ont été bien crées :

```
C:\Users\pc>docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS                      PO
RTS                                                                                                    NAMES
a3577cec732e   liliasfaxi/spark-hadoop:hv-2.7.2   "sh -c 'service ssh …"   5 weeks ago     Exited (255) 2 weeks ago    0.
0.0.0:8041->8042/tcp                                                                                   hadoop-slave2
447eeebffe8d   liliasfaxi/spark-hadoop:hv-2.7.2   "sh -c 'service ssh …"   5 weeks ago     Exited (255) 4 weeks ago    0.
0.0.0:8040->8042/tcp                                                                                   hadoop-slave1
e29202d6f11f   liliasfaxi/spark-hadoop:hv-2.7.2   "sh -c 'service ssh …"   5 weeks ago     Exited (255) 2 weeks ago    0.
0.0.0:7077->7077/tcp, 0.0.0.0:8088->8088/tcp, 0.0.0.0:16010->16010/tcp, 0.0.0.0:50070->50070/tcp   hadoop-master
0737b8bbe864   liliasfaxi/spark-hadoop:hv-2.7.2   "sh -c 'service ssh …"   6 weeks ago     Exited (255) 5 weeks ago
                                                                                                       condescending_wilbur
1ad89847a0ce   alpine/git                     "git clone https://g…"   21 months ago   Exited (0) 21 months ago
                                                                                                       repo
```

Nos 3 conteneurs sont actives et prêtes à être utilisés :

On entre dans le conteneur master pour commencer à l'utiliser :

```
C:\Users\pc>    docker exec -it hadoop-master bash

root@hadoop-master:~#
root@hadoop-master:~#
```

Nous nous retrouvons dans le shell du namenode, et nous pourrons maintenent manipuler le cluster à notre guise. La première chose à faire, une fois dans le conteneur, est de lancer hadoop et yarn :

```
root@hadoop-master:~#   ./start-hadoop.sh


Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.18.0.4' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.2' (ECDSA) to the list of known hosts.
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out


starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.2' (ECDSA) to the list of known hosts.
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out
```

Création du répertoire data dans HDFS :

```
root@hadoop-master:~# hadoop fs -mkdir -p data
root@hadoop-master:~# hadoop fs -ls
Found 5 items
drwxr-xr-x   - root supergroup          0 2024-01-13 19:37 data
-rw-r--r--   2 root supergroup          0 2023-12-03 22:34 file1.txt
drwxr-xr-x   - root supergroup          0 2023-12-03 15:18 input
drwxr-xr-x   - root supergroup          0 2023-12-03 16:09 inputs
drwxr-xr-x   - root supergroup          0 2023-12-03 16:06 root
root@hadoop-master:~#
```

Copier les fichiers csv scrapés de ma machine locale vers le conteneur **hadoop-master :**

```
C:\Users\pc>docker cp Downloads/commitments_cleaned.csv hadoop-master:/

C:\Users\pc>docker cp Downloads/question_ecrites.csv hadoop-master:/
CreateFile C:\Users\pc\Downloads\question_ecrites.csv: The system cannot find the file specified.

C:\Users\pc>docker cp Downloads/question_e´crites.csv hadoop-master:/

C:\Users\pc>docker cp Downloads/question_orales.csv hadoop-master:/

C:\Users\pc>
```

On revient maintenant vers hadoop-master pour vérifier si les fichiers existent bien dans le namenode :

```
root@hadoop-master:~# ls ..
bin   commitments_cleaned.csv  etc   lib    media  opt   question_ecrites.csv  root  sbin  sys  usr
boot  dev                      home  lib64  mnt    proc  question_orales.csv   run   srv   tmp  var
root@hadoop-master:~#
```

Copier les fichiers csv du **hadoop-master** vers **HDFS** dans le répertoire **data**:

```
root@hadoop-master:~# hadoop fs -copyFromLocal /commitments_cleaned.csv /data/
root@hadoop-master:~# hadoop fs -copyFromLocal /question_orales.csv /data/
root@hadoop-master:~# hadoop fs -copyFromLocal /question_ecrites.csv /data/
root@hadoop-master:~#
```

Nos fichiers existent bien dans le répertoire **data** dans HDFS**:**

```
root@hadoop-master:~# hadoop fs -ls /data
Found 3 items
-rw-r--r--   2 root supergroup     644196 2024-01-13 20:26 /data/commitments_cleaned.csv
-rw-r--r--   2 root supergroup   23625795 2024-01-13 20:43 /data/question_ecrites.csv
-rw-r--r--   2 root supergroup    9143517 2024-01-13 20:36 /data/question_orales.csv
root@hadoop-master:~#
```

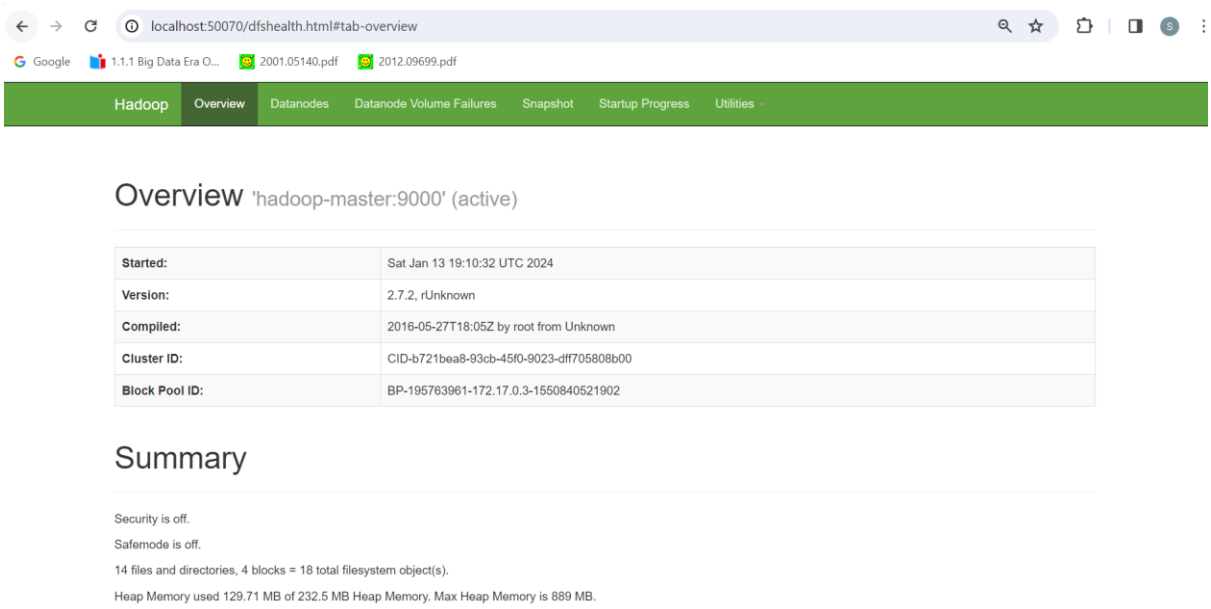
Interfaces web pour Hadoop :

Hadoop offre plusieurs interfaces web pour pouvoir observer le comportement de ses différentes composantes. Vous pouvez afficher ces pages en local sur votre machine grâce à l'option -p de la commande `docker run`. En effet, cette option permet de publier un port du contenaire sur la machine hôte. Pour pouvoir publier tous les ports exposés, vous pouvez lancer votre contenaire en utilisant l'option `-P`.

En regardant le contenu du fichier `start-container.sh` fourni dans le projet, vous verrez que deux ports de la machine maître ont été exposés:

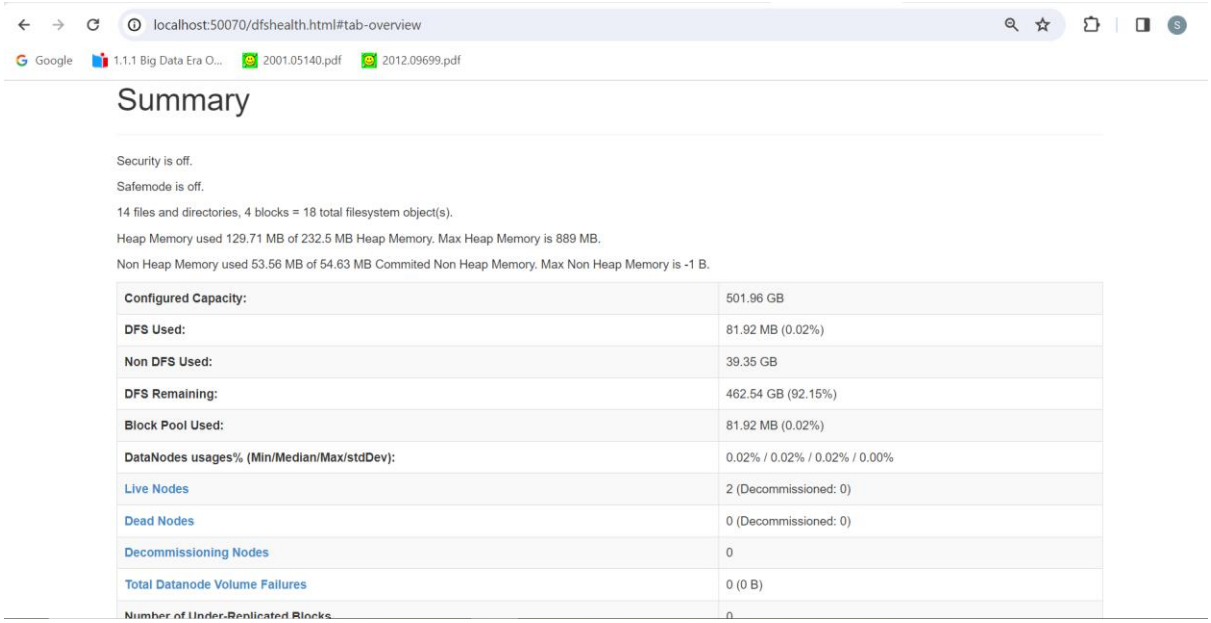
- Le port 50070: qui permet d'afficher les informations du namenode.
- Le port 8088: qui permet d'afficher les informations du resource manager de Yarn et visualiser le comportement des différents jobs.

Notre cluster est lancé et prêt à l'emploi sur : http://localhost:50070.

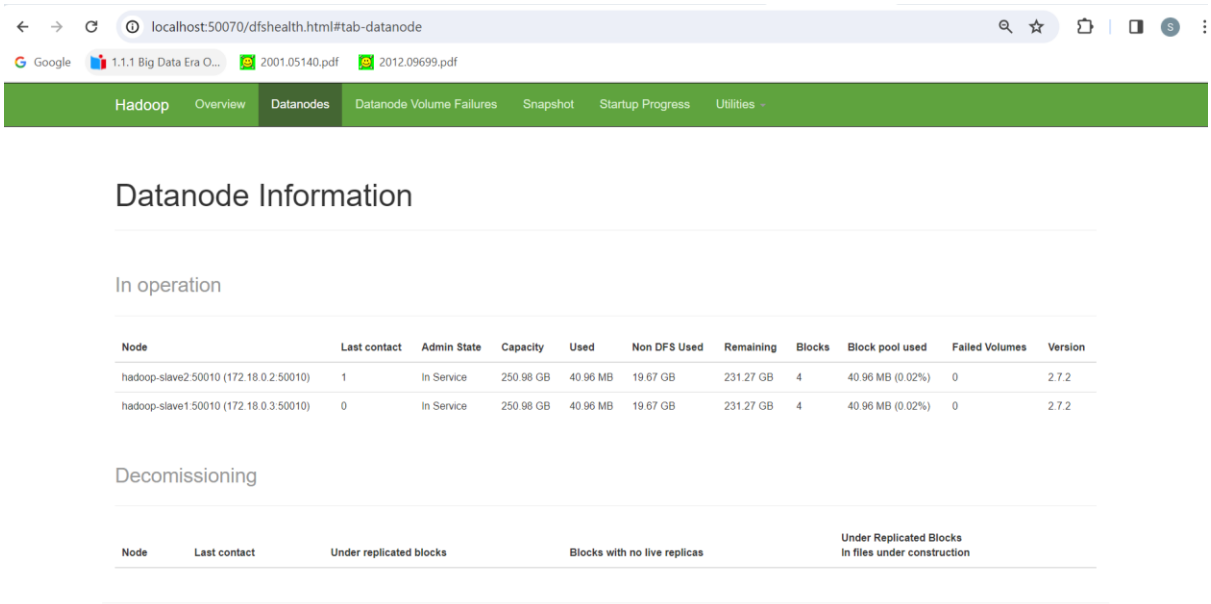


Dans la section summary on obtient des informations supplémentaires telles que :

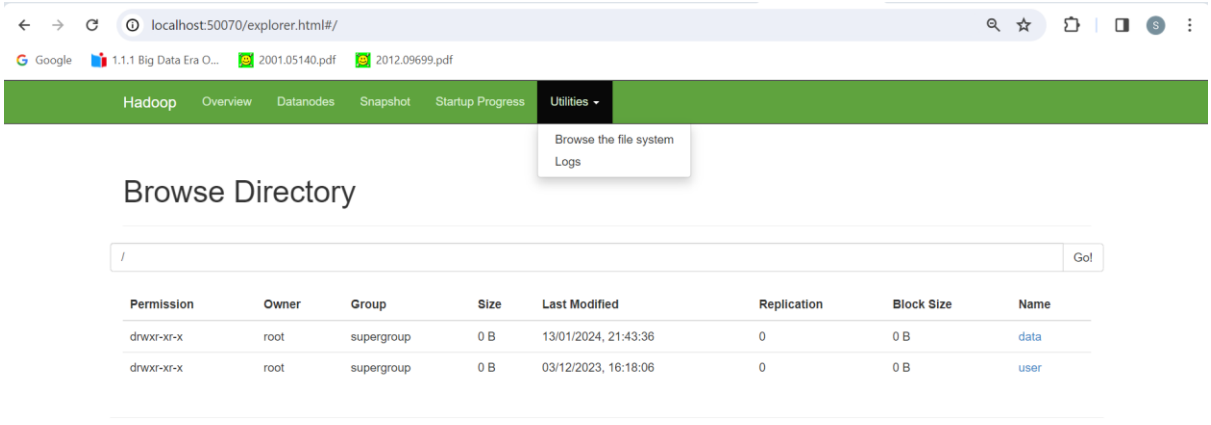
- Distributed File System(DFS) used.
- DFS remaining
- live nodes
- nombre de blocs



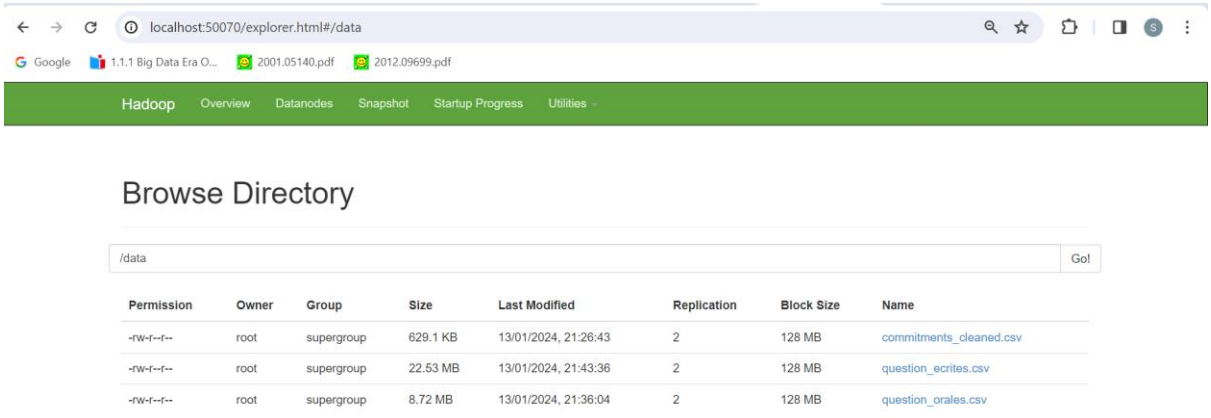
Datanodes :



Sous **Utilities** on peut voir les informations de notre **DFS** :



Sous le repertoire **data** on trouve nos fichiers csv :



On peut également visualiser l'avancement et les résultats de nos Jobs sur : http://localhost:8088