

Objectif

Ce projet est destiné à une compagnie d'assurances qui propose des assurances-vie à ses clients.

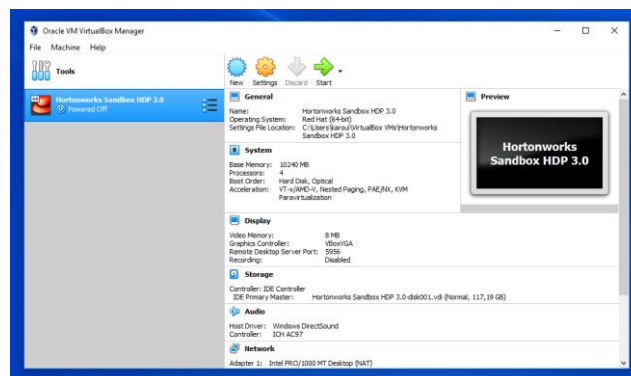
On doit leur fournir un outil permettant de prédire le niveau de risque lié à l'ouverture d'un nouveau contrat client.

On dispose d'un ensemble de données anciennes, où on a pour chaque client le niveau de risque correspondant. En se basant sur ces données ainsi que notre savoir-faire on va pouvoir prédire le niveau de risque associé à chaque nouveau client d'une manière automatique.

Etapes

On commence par télécharger et installer une machine virtuelle Hadoop « HDP Sandbox 3.0.1 » sur « Oracle VM VirtualBox » local.

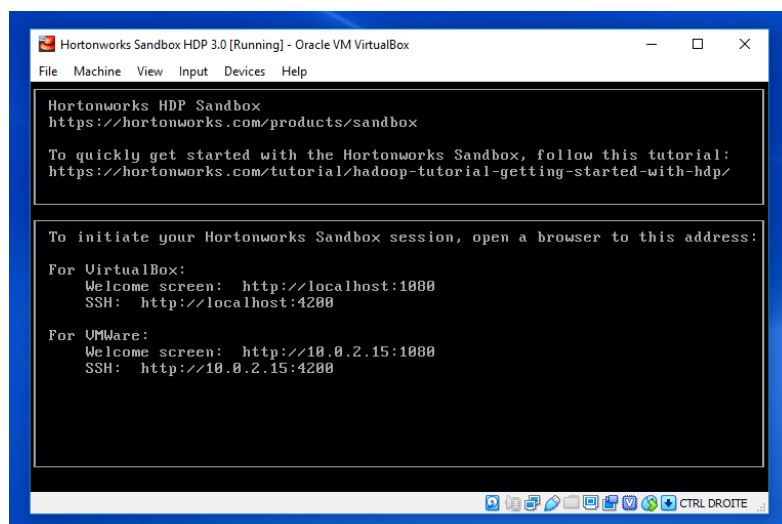
C'est une distribution toute prête d'où son volume de 20Go.



HDP est une plate-forme de données basée sur Hadoop qui contient les systèmes Hadoop Distributed File System (HDFS).

On va se servir de ce système de stockage HDFS pour stocker nos données.

Après exécution de la machine on se trouve face à cette fenêtre.



On accède à « localhost:4200 » pour faire les configurations nécessaires, notamment changer le mot de passe du compte administrateur.

Cela se fait la première fois uniquement et en exécutant la commande « ambari-admin-password-reset ».

```
root@sandbox-hdp:~ - Shell In / x Hortonworks Sandbox with HDP x +
localhost:4200
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
You are required to change your password immediately (root enforced)
Last login: Thu Nov 29 18:47:14 2018
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox-hdp ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

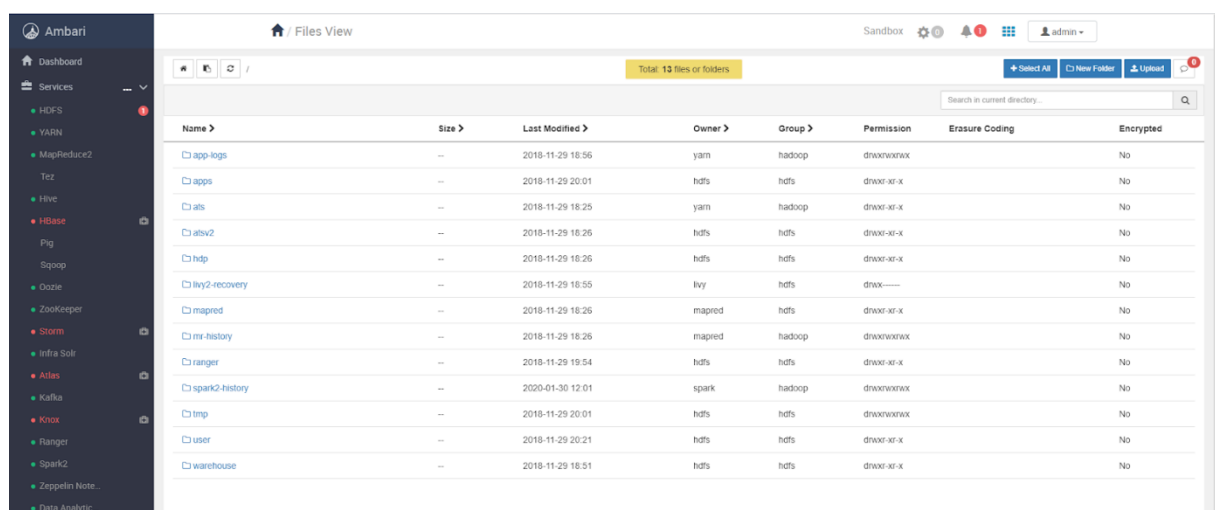
The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....
Server started listening on 8080

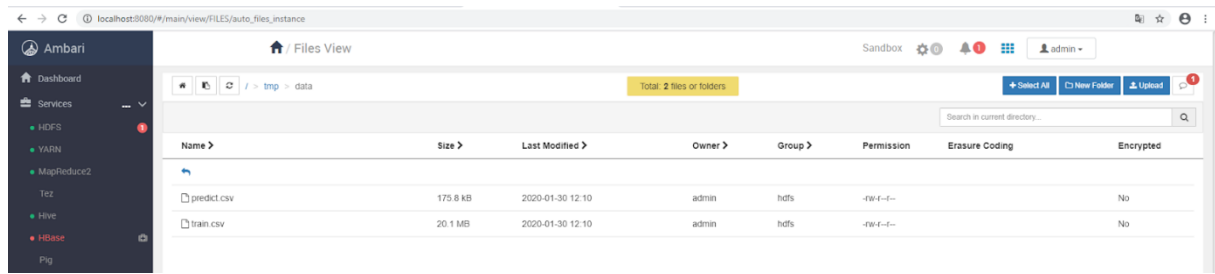
DB configs consistency_check: no errors and warnings were found.
[root@sandbox-hdp ~]#
```

On peut maintenant accéder à « localhost:1080 » et se connecter à « Ambari ».

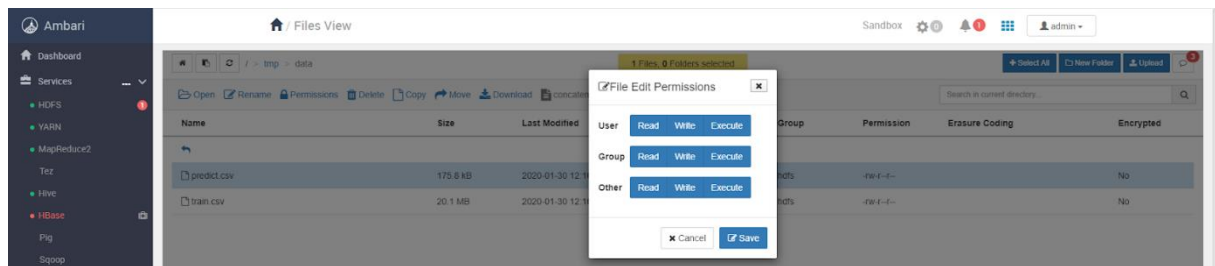
« Ambari » est une interface qui va nous permettre d'interagir avec le système de fichiers.



On va créer un dossier nommé « data » sous le dossier « tmp » où on va importer nos fichiers : « train.csv » et « predict.csv ».



Ces fichiers auront toutes les permissions nécessaires pour assurer leur manipulation par la suite.



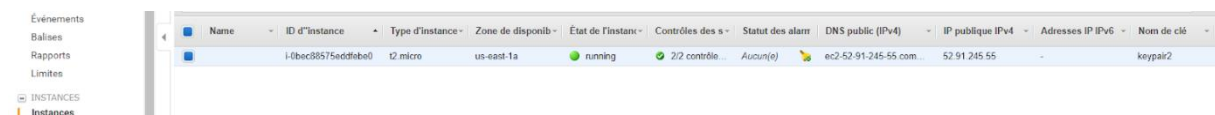
Maintenant que nos fichiers sont stockés sur HDFS ; distribués sur les machines de cluster. On peut les récupérer, à tout moment, vers notre machine locale Windows.

Pour ce faire, il suffit d'utiliser la commande suivante en précisant le chemin complet des fichiers depuis la racine.

```
[root@sandbox-hdp ~]# hadoop fs -get /tmp/data/* /root
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg  predict.csv  train.csv
```

Ces données récupérées vont être transférés à une machine virtuelle Linux dans le cloud AWS.

On va utiliser Amazon EC2 pour créer notre instance.



Pendant la création de cette machine virtuelle Linux, on spécifie les groupes de sécurité ainsi que la paire de clés qui va nous permettre d'y accéder d'une manière sécurisée.

Ce chiffrement est mis en place suite au caractère sensible des données manipulées.

Il faut importer la clé privée sur HDFS puis notre local de la même manière que les fichiers « train.csv » et « predict.csv » mais cette fois en limitant les permissions accordées.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@          WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0644 for 'keypair2.pem' are too open.
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "keypair2.pem": bad permissions
Permission denied (publickey).
lost connection
[root@sandbox-hdp ~]# sudo chmod 400 keypair2.pem
```

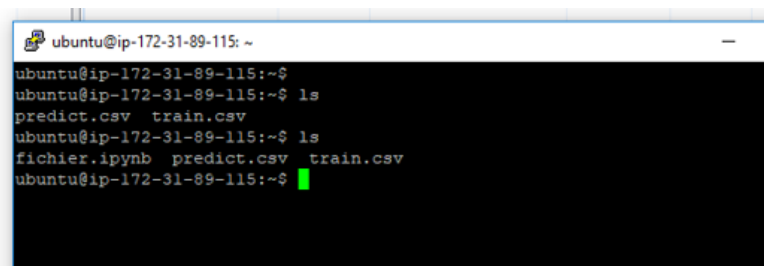
On a bien tous nos fichiers déposés localement.

```
[root@sandbox-hdp ~]# hadoop fs -get /tmp/data/keypair2.pem /root
[root@sandbox-hdp ~]# ls
apps  boot  core.16406  dev  hadoop  kafka-logs  lib64  mnt
bin  cgroups  test  core.20215  etc  home  lib  media  mysql-connector-java-5.1.45  opt  packer-files  proc  run  sandbox-flavour  srv  tmp  var
[root@sandbox-hdp ~]# ls -l /root
total 20756
-rw-r--r-- 1 root root 3302 May 31 2018 anaconda-ks.cfg
-rw-r--r-- 1 root root 1670 Jan 30 12:48 keypair2.pem
-rw-r--r-- 1 root root 179969 Jan 30 12:21 predict.csv
-rw-r--r-- 1 root root 21063261 Jan 30 12:21 train.csv
[root@sandbox-hdp ~]#
```

On passe maintenant au transfert de données vers AWS. Pour ce faire, on utilise la commande suivante :

```
[root@sandbox-hdp ~]# sudo scp -i keypair2.pem train.csv ubuntu@ec2-52-91-245-55.compute-1.amazonaws.com:~
train.csv 100% 20MB 5.4MB/s 00:03
[root@sandbox-hdp ~]# sudo scp -i keypair2.pem predict.csv ubuntu@ec2-52-91-245-55.compute-1.amazonaws.com:~
predict.csv 100% 176KB 353.6KB/s 00:00
[root@sandbox-hdp ~]#
```

SCP va permettre de transférer d'une manière sécurisée nos fichiers via une connexion SSH.



Traitement de données

En se basant sur les données fournies et les algorithmes de Machine Learning, on va tester puis sélectionner un modèle d'apprentissage approprié.

On va développer ce modèle en utilisant Python et les bibliothèques nécessaires.

L'algorithme développé sera exécuté sur l'instance AWS vue précédemment.

- Pour consulter l'analyse détaillée, vous pouvez consulter le repo. GitHub :

https://github.com/WOQUQ/BigDataProjet/tree/master/Data_Process

Nous pouvons voir qu'après l'exécution du fichier python, un nouveau fichier est généré.

```
ec2-user@ip-172-31-80-173:/home/big_data
Last login: Thu Feb 13 17:08:07 2020 from wifiroam058204.univ-st-etienne.fr

 _ _ | _ _ | _ )
 _ | ( _ _ /   Amazon Linux 2 AMI
 _ _ | \ _ _ | _ _ |

https://aws.amazon.com/amazon-linux-2/
19 package(s) needed for security, out of 57 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-80-173 ~]$ cd ..
[ec2-user@ip-172-31-80-173 home]$ ls
big_data  ec2-user
[ec2-user@ip-172-31-80-173 home]$ cd big_data/
[ec2-user@ip-172-31-80-173 big_data]$ ls
algo.py  predict.csv  train.csv
[ec2-user@ip-172-31-80-173 big_data]$ python algo.py
Traceback (most recent call last):
  File "algo.py", line 6, in <module>
    import pandas as pd
ImportError: No module named pandas
[ec2-user@ip-172-31-80-173 big_data]$ python3 algo.py
[ec2-user@ip-172-31-80-173 big_data]$ ls
algo.py  predict.csv  result.csv  train.csv
[ec2-user@ip-172-31-80-173 big_data]$
```

Ce fichier nommé « result.csv » fait une correspondance entre chaque ligne du fichier « predict.csv » et la prédiction du modèle choisie. Il est stocké actuellement dans le système de fichiers de la machine virtuelle AWS.

Nous allons utiliser la même instruction SCP vue précédemment pour récupérer ce fichier depuis le serveur distant.

```
zhi@zhi-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
zhi@zhi-VirtualBox:~/Downloads$ scp -i test.pem ec2-user@ec2-18-233-156-216.compu
te-1.amazonaws.com:/home/big_data/result.csv /tmp
result.csv                               100% 167KB 262.9KB/s 00:00
zhi@zhi-VirtualBox:~/Downloads$
```

Ensuite, nous devons stocker les données dans « MongoDB ».

Nous utilisons :

- Le package *panda* pour lire le fichier csv.
- Le package *json* pour convertir les données au format json.
- *pymongo* pour nous connecter à la base de données et stocker les données.

```

zhi@zhi-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
GNU nano 2.9.3 test.py Modified

import pymongo
import pandas as pd
import json

client = pymongo.MongoClient("mongodb://localhost:27017")
db = client["mydatabase"]
name="CompustatName"
collection=db[name]
CompustatName = pd.read_csv("result.csv")
data = json.loads(CompustatName.to_json(orient="records"))
collection.insert(data)

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line

```

Nous pouvons voir que les données ont été stockées dans la base de données au format json.

```

> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
nydatabase 0.022GB
> use nydatabase
switched to db nydatabase
> show collections
compustatName
> db.nycol.find()
{}
> db.collection.find()
{}
> db.compustatName.find()
[{"_id": ObjectId("5e3beabeb00f447e9087d01"), "Medical_Keyword_23": 0, "Ht": 0.563636364, "InsuredInfo_9": "Muslim", "Medical_Keyword_27": 0, "Family_Hist_3": null, "Family_Hist_2": 0.68115942, "Family_Hist_1": 3, "Family_Hist_5": null, "Family_Hist_4": 0.492957746, "Wt": 0.225941423, "Medical_Keyword_9": 0, "Medical_Keyword_8": 0, "Medical_Keyword_15": 0, "Medical_Keyword_14": 0, "Medical_Keyword_17": 0, "Medical_Keyword_16": 0, "Medical_Keyword_11": 0, "Medical_Keyword_10": 0, "Medical_Keyword_13": 0, "Medical_Keyword_12": 0, "Medical_Keyword_19": 0, "Medical_Keyword_18": 0, "Medical_Keyword_7": 7, "Medical_History_10": 1, "Medical_History_11": 3, "Medical_History_14": 3, "Medical_History_15": null, "Medical_History_12": 2, "InsuredInfo_4": 3, "Medical_History_16": null, "Medical_History_11": 3, "Medical_Keyword_20": 0, "Medical_Keyword_21": 0, "Medical_Keyword_22": 0, "BMI": 0.510719005, "Medical_Keyword_24": 0, "Medical_Keyword_25": 0, "Medical_History_18": 1, "Medical_History_19": 1, "InsuredInfo_8": null, "InsuredInfo_2": 2, "InsuredInfo_3": 3, "InsuredInfo_1": 2, "InsuredInfo_6": 1, "InsuredInfo_7": "Female", "Insurance_History_9": 3, "Insurance_History_8": 1, "Insurance_History_7": 3, "Insurance_History_5": null, "Insurance_History_4": 3, "Insurance_History_3": 1, "Insurance_History_2": 1, "Insurance_History_1": 2, "Medical_Keyword_26": 0, "Medical_Keyword_48": 0, "Medical_Keyword_22": 0, "Medical_Keyword_43": 0, "Medical_Keyword_49": 0, "Medical_Keyword_41": 0, "Medical_Keyword_46": 0, "Medical_Keyword_47": 0, "Medical_Keyword_44": 0, "Medical_Keyword_45": 0, "Medical_Keyword_39": 0, "Medical_Keyword_38": 0, "Medical_Keyword_37": 0, "Medical_Keyword_36": 0, "Medical_Keyword_35": 0, "Medical_Keyword_34": 0, "Medical_Keyword_33": 0, "Medical_Keyword_32": 0, "Medical_Keyword_31": 0, "Medical_Keyword_30": 0, "Medical_History_38": 1, "Medical_History_39": 3, "Medical_History_30": 2, "Medical_History_31": 3, "Medical_History_32": null, "Medical_History_33": 3, "Medical_History_34": 3, "Medical_History_35": 1, "Medical_History_36": 3, "Medical_History_37": 2, "Medical_History_8": 2, "Medical_History_9": 2, "Medical_History_4": 2}]]

```

Ensuite, nous pouvons interroger les données dans la base de données. Par exemple, imprimez l'id et la réponse correspondante :

```
zhi@zhi-VirtualBox: ~  
File Edit View Search Terminal Help  
    "Medical_Keyword_28" : 0,  
    "Medical_Keyword_29" : 0,  
    "Medical_History_13" : 3,  
    "Ins_Age" : 0.432835821  
}  
Type "it" for more  
> db.CompustatName.find({}, { _id: 1, Response: 1 }).pretty()  
{ "_id" : ObjectId("5e3beabeb008f447e9087d01"), "Response" : 7 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d02"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d03"), "Response" : 1 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d04"), "Response" : 4 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d05"), "Response" : 2 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d06"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d07"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d08"), "Response" : 2 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d09"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0a"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0b"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0c"), "Response" : 2 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0d"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0e"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0f"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d10"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d11"), "Response" : 7 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d12"), "Response" : 2 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d13"), "Response" : 4 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d14"), "Response" : 2 }  
Type "it" for more  
>
```

Il est tout de même possible de définir des conditions de requête. Par exemple, pour afficher les données avec un résultat supérieur ou égal à 6 :

```
zhi@zhi-VirtualBox: ~  
File Edit View Search Terminal Help  
> db.CompustatName.find({"Response": {$gte:6}}, { _id: 1, Response: 1 }).pretty()  
{ "_id" : ObjectId("5e3beabeb008f447e9087d01"), "Response" : 7 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d02"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d06"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d07"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d09"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0a"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0b"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0d"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0e"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d0f"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d10"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d11"), "Response" : 7 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d15"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d16"), "Response" : 7 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d18"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d19"), "Response" : 6 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d1d"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d1e"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d1f"), "Response" : 8 }  
{ "_id" : ObjectId("5e3beabeb008f447e9087d20"), "Response" : 8 }  
Type "it" for more  
>
```

Donc on a nos données stockées localement dans une base de données et sont consultables avec un format approprié.

