

Projet Big Data

Rapport

Equipe:

KHADROUF Oumaima
HOUMAIR Abdelaziz
BIROUK Yassine

Encadrants :

Charlotte LACLAU
Remy GIRODON
Julien TISSIER

Table des matières

Introduction	3
Etape 1: Environnement Hadoop	4
1.1 Hadoop en local.....	4
1.2 Hadoop dans le cloud	5
Etape 2: AWS	8
Etape 3 : Analyse des données	11
3.1 Configurer et utiliser Jupyter Notebook sur AWS	11
3.2 Traitement de données avec Python	12
3.2.1 Equilibrer les données : Random under-sampling [5]	12
3.2.2 Traitement des données manquantes :	13
3.2.3 Numérisation des données :	14
3.2.4 Détection anomalies.....	14
3.2.5 Normalisation des données	15
3.2.6 Traitement des données :	16
Etape 4 : Base de données NoSQL : MongoDB	19
4.1 Importer un fichier csv sur MongoDB	19
4.2 Application Web pour exploiter les données sur MongoDB	20
Bibliographie:	22

Introduction

Ce projet a pour but de mettre en oeuvre les connaissances acquises dans le cloud computing, les algorithmes d'analyse de données et le Nosql. Ainsi, on est invité à traiter des données qui concernent des clients qui cherchent un prêt immobilier afin de prédire s'ils vont rembourser leurs prêts ou non.

En effet, les données sont en format d'un fichier excel qui contient des informations sur les clients notamment leurs emplois, leurs statuts, les types des biens qu'ils souhaitent acheter et un champ "target" qui prend des valeurs binaires (0:si la personne a remboursé son prêt et 1:sinon).

Notre première mission consiste à stocker ces données dans une machine virtuelle Hadoop en local pour les récupérer ensuite et les pousser vers une machine virtuelle Linux hébergée dans le cloud d'Amazon web services tout en respectant certaines règles de sécurité à savoir le chiffrement des données. Ensuite, on peut commencer le traitement des données avec des algorithmes en python. On cite que les données initiales seront divisées en deux parties, la première sera consacrée à l'apprentissage du modèle alors que la deuxième ne va pas contenir le champ "target" et va servir au test de performances des algorithmes qui vont être exécutés dans la machine virtuelle citée auparavant. Les prédictions du champ "target" seront concaténées par la suite aux lignes correspondantes pour sauvegarder le tout dans un fichier de format csv stocké dans le cloud.

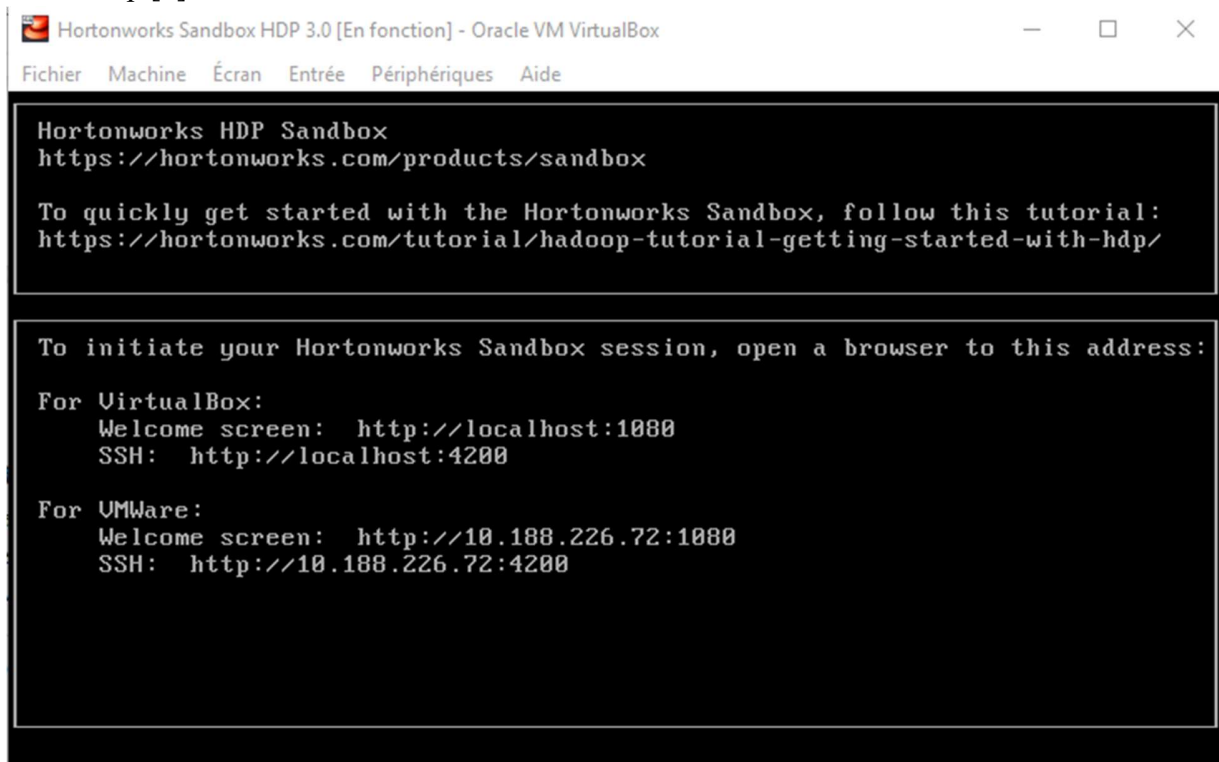
Finalement, on doit transférer les données résultantes vers une base de données Nosql (Mongodb à titre d'exemple) s'exécutant sur une VM en local.

Etape 1: Environnement Hadoop

Cette étape consiste à stocker ces données dans un environnement Hadoop en local pour les récupérer ensuite et les pousser vers une machine virtuel Linux hébergé dans le cloud d'Amazon web services tout en respectant certaines règles de sécurité à savoir le chiffage des données.

1.1 Hadoop en local

D'abord, on installe la machine virtuelle sandbox de Hortonworks qui contient une distribution de Hadoop [1].



On télécharge les fichier dataset.csv sur le local

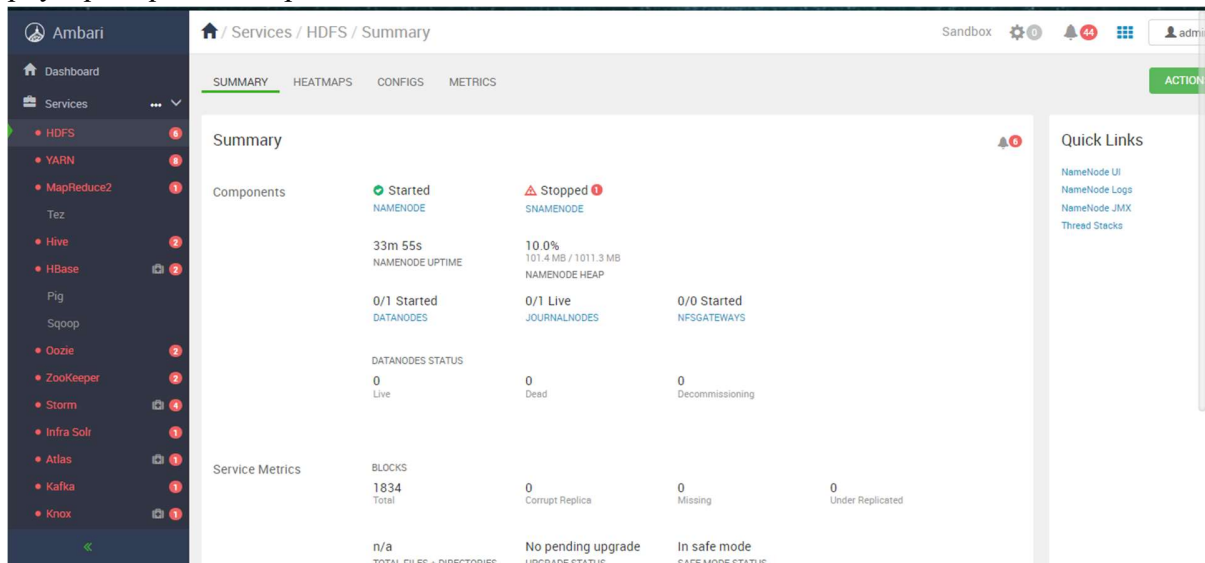
```
[root@sandbox-hdp ~]# wget https://filezemp.univ-st-etienne.fr/7yvltku/download
--2019-01-20 10:34:12-- https://filezemp.univ-st-etienne.fr/7yvltku/download
Resolving filezemp.univ-st-etienne.fr (filezemp.univ-st-etienne.fr)... 161.3.155.60
Connecting to filezemp.univ-st-etienne.fr (filezemp.univ-st-etienne.fr)|161.3.155.6
HTTP request sent, awaiting response... 200 OK
Length: 102592320 (98M) [text/csv]
Saving to: 'download'

100%[=====>] 102,592,320 3.20MB/s in 38s

2019-01-20 10:34:51 (2.57 MB/s) - 'download' saved [102592320/102592320]

[root@sandbox-hdp ~]# ls
anaconda-ks.cfg download
```

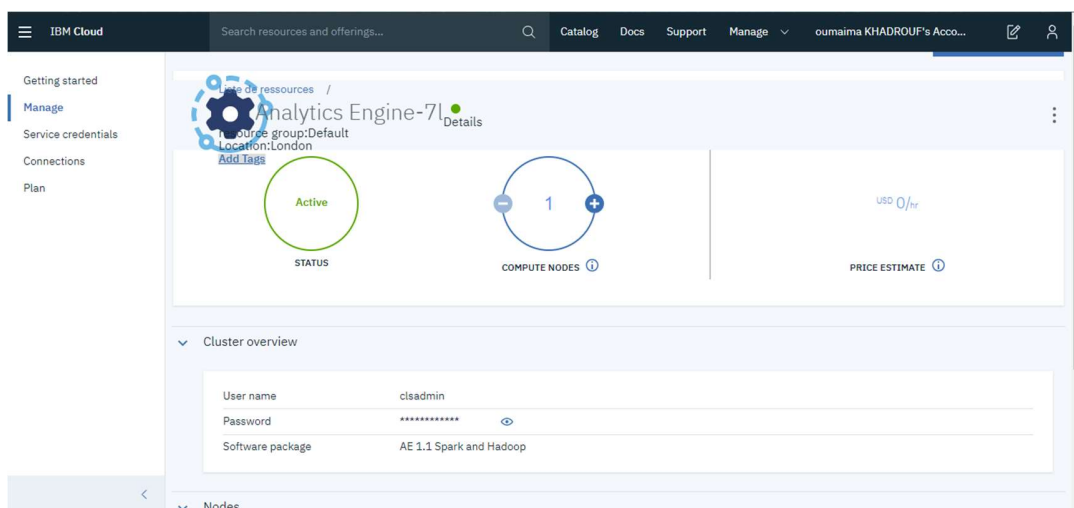
On remarque qu'il y a des problèmes dans le fonctionnement de Hadoop à cause des ressources physiques qui ne sont pas suffisantes.

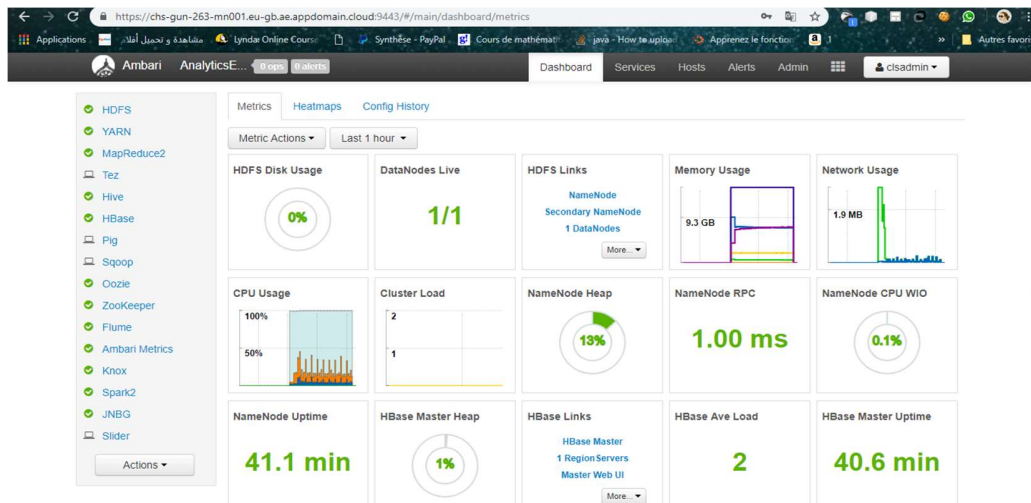


Une raison pour laquelle on a décidé d'utiliser un service Hadoop dans le cloud IBM.

1.2 Hadoop dans le cloud

Après le démarrage de l'instance on obtient un login et mot de passe pour accéder à l'interface Ambari.





On obtient aussi une adresse IP et un mode passe pour accéder au service Hadoop via Git Bash.

```

clsadmin@chs-gun-263-mn003:~
$ ssh clsadmin@158.175.143.94
The authenticity of host '158.175.143.94 (158.175.143.94)' can't be established.
ECDSA key fingerprint is SHA256:l4akGun8biQ7cZ6nyq5lwtNbMgPjMqjLIYN/YLhFzDY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '158.175.143.94' (ECDSA) to the list of known hosts.
clsadmin@158.175.143.94's password:
Last login: Thu Jan 24 11:24:59 2019

```

Après on a téléchargé le fichier excel dans le local.

```

clsadmin@chs-gun-263-mn003:~
drwxr-xr-x - mapred bihdfs 0 2019-01-24 11:24 /mapred
drwxrwxrwx - mapred hadoop 0 2019-01-24 11:25 /mr-history
drwx----- - clsadmin biusers 0 2019-01-24 11:35 /securedir
drwxrwxrwx - spark hadoop 0 2019-01-24 12:26 /spark2-history
drwxrwxrwx - hdfs bihdfs 0 2019-01-24 11:26 /tmp
drwxr-xr-x - hdfs bihdfs 0 2019-01-24 11:29 /user
[clsadmin@chs-gun-263-mn003 ~]$ wget https://filezemp.univ-st-etienne.fr/7yvltku/download
--2019-01-24 12:28:38-- https://filezemp.univ-st-etienne.fr/7yvltku/download
Resolving filezemp.univ-st-etienne.fr (filezemp.univ-st-etienne.fr)... 161.3.155
.60
Connecting to filezemp.univ-st-etienne.fr (filezemp.univ-st-etienne.fr)|161.3.155
5.60|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 102592320 (98M) [text/csv]
Saving to: 'download'

100%[=====>] 102,592,320 289KB/s in 5m 29s

2019-01-24 12:34:07 (305 KB/s) - 'download' saved [102592320/102592320]

[clsadmin@chs-gun-263-mn003 ~]$ ls
download hadoop-mapreduce.jobsummary.log pipAnaconda2Packages pipAnaconda3Packages R s

```

Ensuite, on a utilisé la commande put pour pousser le fichier vers Hadoop.

```
[clsadmin@chs-gun-263-mn003 ~]$ hadoop fs -put download hadoopfiles
[clsadmin@chs-gun-263-mn003 ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x  - clsadmin biusers          0 2019-01-24 12:38 hadoopfiles
[clsadmin@chs-gun-263-mn003 ~]$ hadoop fs -ls hadoopfiles
Found 1 items
-rw-r--r--  3 clsadmin biusers 102592320 2019-01-24 12:38 hadoopfiles/download
```

On accède au view files dans la plateforme Ambari pour voir le fichier téléchargé.

Name >	Size >	Last Modified >	Owner >	Group >	Permission
hadoopfiles	--	2019-01-24 13:38	clsadmin	biusers	drwxr-xr-x

Name >	Size >	Last Modified >	Owner >	Group >	Permission
download	97.8 MB	2019-01-24 13:38	clsadmin	biusers	-rw-r--r--

Dernièrement on utilise la commande get pour télécharger le fichier depuis Hadoop vers le système de fichiers local.

```
[clsadmin@chs-gun-263-mn003 ~]$ mkdir receivedfiles
[clsadmin@chs-gun-263-mn003 ~]$ hadoop fs -get hadoopfiles/download receivedfiles
[clsadmin@chs-gun-263-mn003 ~]$ ls receivedfiles
download
[clsadmin@chs-gun-263-mn003 ~]$ |
```

Etape 2: AWS

L'étape 2 consiste à pousser l'ensemble des données vers une machine virtuelle dans le cloud AWS. Sachant que ces données sont d'une nature confidentielle, on doit assurer une certaine protection à ces derniers.

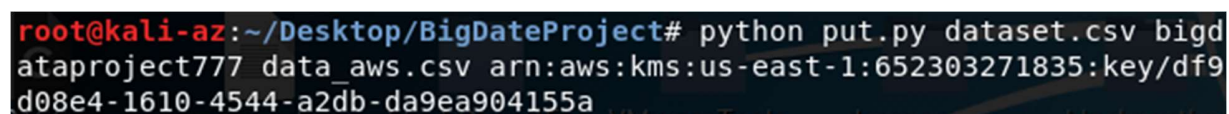
Pour arriver à notre but, on va tout d'abord crypter ces données côté client, c'est à dire que nous gérons le processus de chiffrement, les clés de chiffrement et les outils associés.

AWS propose un chiffrement côté serveur, c'est à dire qu'on peut demander à Amazon de chiffrer nos objets avant de les enregistrer sur les disques des centres de données, et de les déchiffrer lorsque nous téléchargeons les objets, mais on ne peut pas malheureusement assurer la confidentialité des données dans le réseau. C'est pour ça qu'on a opté pour du chiffrement côté client.

Une fois notre fichier crypter, on va l'envoyer sur Amazon S3, qui est un site d'hébergement de fichiers proposé par Amazon Web Services. Ensuite, on va récupérer ses données cryptées depuis une instance exécuté sur EC2 (machine virtuelle), pour pouvoir les traiter après.

Pour assurer le cryptage et l'envoi de données, on a utilisé un code sous licence MIT [2], il suffit donc de lancer la commande suivante :

```
python put.py dataset.csv bigdataproject777 data_aws.csv arn:aws:kms:us-east-1:652303271835:key/df9d08e4-1610-4544-a2db-da9ea904155a
```

A screenshot of a terminal window with a dark background. The prompt is 'root@kali-az:~/Desktop/BigDateProject#'. The command entered is 'python put.py dataset.csv bigdataproject777 data_aws.csv arn:aws:kms:us-east-1:652303271835:key/df9d08e4-1610-4544-a2db-da9ea904155a'. The command is partially highlighted in blue and red. The output is not visible.

```
root@kali-az:~/Desktop/BigDateProject# python put.py dataset.csv bigdataproject777 data_aws.csv arn:aws:kms:us-east-1:652303271835:key/df9d08e4-1610-4544-a2db-da9ea904155a
```

Où :

- **Put.py** : le script python qui permettra de crypter nos données, et ensuite de les envoyer dans Amazon S3.
- **Dataset.csv** : le nom du fichier qui contient les données.
- **Bigdataproject777** : le nom du bucket sur lequel on va stocker nos données.
- **Data_aws.csv** : le nom du fichier avec lequel on veut enregistrer nos données sur Amazon S3.
- **Arn:aws:us-east....** : L'id de la clé du chiffrement utilisé pour crypter.

Une fois la commande exécutée, on remarque alors que nos données ont bien été enregistrées sur le bucket, et qu'elles sont cryptées.

Affichage 1 à 1			
<input type="checkbox"/> Nom ▼	Dernière modification ▼	Taille ▼	Classe de stockage ▼
<input type="checkbox"/> data_aws.csv	janv. 18, 2019 6:58:00 PM GMT+0100	97.8 Mo	Standard

Affichage 1 à 1			
18 ç-²0Ö!üÊh±Ñu T...			
19 uöFä Oß=ç à»sö ñ			
20 „NUc_ 6'ûÆ*G yTä1			
21 ^i Ö·!Üà G1"« #Ups±			
22 ?Kh éýxos NÁéù")			

Après, on se connecte avec ssh sur notre instance EC2, avec la commande suivante :

```
ssh -i "groupe1.pem" ec2-user@ec2-3-84-50-179.compute-1.amazonaws.com
```

Où :

- Groupe1.pem : Clé pour se connecter sur la machine virtuelle.
- Ec2-user : le nom de l'utilisateur sur lequel on se connecte.
- Ec2-3-84-50-179.compute-1.amazonaws.com : le nom de domaine de notre machine.

Et on exécute le script qui va permettre de télécharger les données et de les décrypter, avec la commande suivante : `python3 get.py bigdataproject777 data_aws.csv new_dataset.csv`

```
[ec2-user@ip-172-31-93-99 s3-client-side-encryption]$ python3 get.py bigdataproject777 data_aws.csv new_dataset.csv
```

Où :

- **get.py** : le script python qui permettra de télécharger nos données, et ensuite de les décrypter.
- **Bigdataproject777** : le nom du bucket sur lequel on va télécharger nos données.
- **Data_aws.csv** : le nom du fichier avec lequel on a enregistré nos données sur Amazon S3.
- **New_dataset.csv** : le nom avec lequel on souhaite enregistrer nos données téléchargées

On remarque alors que nos données ont bien été téléchargées, et qu'elles sont décryptées.

```

[ec2-user@ip-172-31-93-99 s3-client-side-encryption]$ ls
LICENSE  decrypted-new_dataset.csv  get.py  new_dataset.csv  put.py
[ec2-user@ip-172-31-93-99 s3-client-side-encryption]$ cat decrypted-new_dataset.csv
TARGET,NAME CONTRACT_TYPE,CODE GENDER,FLAG_OWN_CAR,FLAG_OWN_REALTY,CNT_CHILDREN,AMT_INCOME_TOTAL,AMT_CREDIT,A
MT_ANNUITY,AMT_GOODS_PRICE,NAME_TYPE_SUITE,NAME_INCOME_TYPE,NAME_EDUCATION_TYPE,NAME_FAMILY_STATUS,NAME_HOUSI
NG_TYPE,REGION_POPULATION_RELATIVE,DAYS_BIRTH,DAYS_EMPLOYED,DAYS_REGISTRATION,DAYS_ID_PUBLISH,OWN_CAR_AGE,FLA
G_MOBIL,FLAG_EMP_PHONE,FLAG_WORK_PHONE,FLAG_CONT_MOBILE,FLAG_PHONE,FLAG_EMAIL,OCCUPATION_TYPE,CNT_FAM_MEMBERS
,REGION_RATING_CLIENT,REGION_RATING_CLIENT_W_CITY,WEEKDAY_APPR_PROCESS_START,HOURLY_APPR_PROCESS_START,REG_REGION
ON_NOT_LIVE_REGION,REG_REGION_NOT_WORK_REGION,LIVE_REGION_NOT_WORK_REGION,REG_CITY_NOT_LIVE_CITY,REG_CITY_NOT
_WORK_CITY,LIVE_CITY_NOT_WORK_CITY,ORGANIZATION_TYPE,EXT_SOURCE_1,EXT_SOURCE_2,EXT_SOURCE_3,APARTMENTS_AVG,BA
SEMENTAREA_AVG,YEARS_BEGINEXPLUATATION_AVG,YEARS_BUILD_AVG,COMMONAREA_AVG,ELEVATORS_AVG,ENTRANCES_AVG,FLOORSM
AX_AVG,FLOORSMIN_AVG,LANDAREA_AVG,LIVINGAPARTMENTS_AVG,LIVINGAREA_AVG,NONLIVINGAPARTMENTS_AVG,NONLIVINGAREA_A
VG,APARTMENTS_MODE,BASEMENTAREA_MODE,YEARS_BEGINEXPLUATATION_MODE
0,Cash loans,F,N,Y,0,256500.0,1008117.0,29475.0,841500.0,Family,Commercial associate,Secondary / secondary sp
ecial,Married,House / apartment,0.006852,-19202,-6839,-6496.0,-2723,,1,1,0,1,0,0,Sales staff,2.0,3,3,MONDAY,1
0,0,0,0,0,0,Other,0.5725263168170026,0.1004515314269062,0.324891229465852,,,,,,,,,,,,,
0,Cash loans,M,Y,N,0,202500.0,1575000.0,43312.5,1575000.0,Unaccompanied,Working,Secondary / secondary special
Married,House / apartment,0.011656000000000000,14257,1645,2714.0,4115,12,0,1,1,1,1,0,0,Managers,2,0,1,1

```

Etape 3 : Analyse des données

3.1 Configurer et utiliser Jupyter Notebook sur AWS

Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont: Python, Julia, Ruby, R, ou encore Scala2. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte en markdown et du code en Julia, Python, R... Ces notebooks sont utilisés en science des données pour explorer et analyser des données.

On commence tout d'abord par se connecter en SSH a notre instance EC2.

```
ssh -i thisIsmyKey.pem ubuntu@ec2-54-91-13-53.compute-1.amazonaws.com
```

Ensuite, on installe l'ensemble des packages nécessaire pour faire notre analyse de données. [4]

Installation

Update the `pip3` tool :

```
sudo pip3 install --upgrade pip --proxy=https://cache.univ-st-etienne.fr:3128
```

Install the required Python3 packages :

```
sudo pip3 install --upgrade ipython numpy pandas matplotlib scipy scikit-learn jupyter --proxy=https://cache.univ-st-e
```

Install TK for Python 3 (used by Matplotlib) :

```
sudo apt install python3-tk
```

Juste après on entre la commande suivante qui permet d'exécuter jupyter notebook sur notre instance sur le port spécifié : `jupyter notebook --no-browser --port=8888`

La raison pour laquelle nous ajoutons «`--no-browser`» est parce que si nous ne le faisons pas, nous allons obtenir une erreur comme «Jupyter Notebook requires JavaScript».

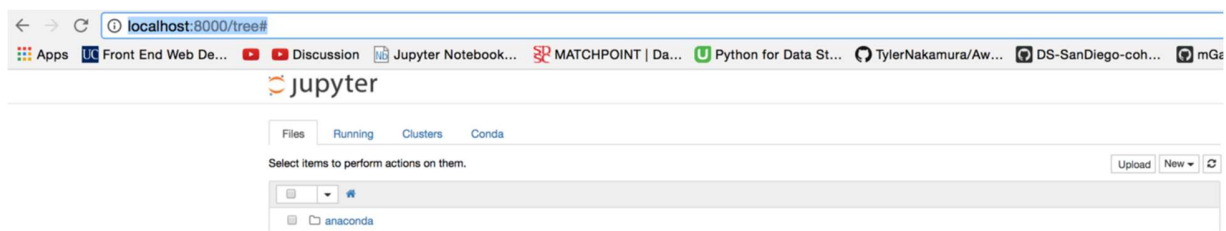
La prochaine étape consiste à ouvrir une connection SSH en locale avec l'application jupyter notebook exécuté sur notre instance.

```
ssh -i thisIsmyKey.pem -L 8000:localhost:8888 ubuntu@ec2-34-227-222-100.compute-1.amazonaws.com
```

-i Spécifie le fichier d'identification à utiliser pour l'authentification par clé publique.

-L spécifie que le port donné sur l'hôte local (client) et le port côté distant (AWS). Cela signifie que tout ce qui est exécuté sur le deuxième numéro de port (8888) sur AWS apparaîtra sur le premier numéro de port (8000) sur votre ordinateur local.

Et finalement, il suffit d'ouvrir un navigateur et taper **localhost:8000** pour accéder à notre application.



3.2 Traitement de données avec Python

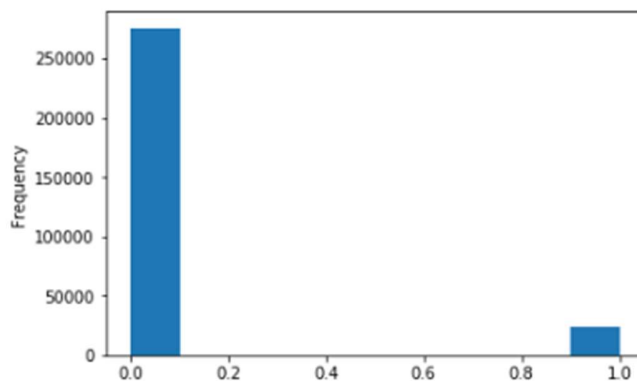
3.2.1 Equilibrer les données : Random under-sampling [5]

Cette technique se base sur la suppression de certaines lignes qui appartiennent à la classe majoritaire d'une manière aléatoire afin d'ajuster la distribution de cette dernière dans la base de données et d'avoir des portions équilibrées de chaque classe.

En effet, l'utilisation de cette méthode engendre une perte de données mais il nous reste un nombre non négligeable de lignes à traiter (24 191 dans chaque classe).

```
Entrée [10]: dataset['TARGET'].astype(int).plot.hist()
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2784b9c278>
```



```

# Class count
count_class_0, count_class_1 = y.value_counts()

# Divide by class
df_class_0 = data[data['TARGET'] == 0]
df_class_1 = data[data['TARGET'] == 1]
#we can see that class 0 is *11 bigger than class 1
print("Class 0:",count_class_0)
print("Class 1:",count_class_1)

Class 0: 275809
Class 1: 24191

# #Random under-sampling
# #we will take (count_class_1=24191) random samples from class 0
df_class_0_under = df_class_0.sample(count_class_1)
# #we will merge the random samples of class 0 with the samples from class 1 to form a new balanced dataset
df_test_under = pd.concat([df_class_0_under, df_class_1], axis=0)

y = df_test_under.TARGET
X = df_test_under.iloc[:,1:].copy()

print("Class 0:",len(df_test_under[df_test_under['TARGET'] == 0]))
print("Class 1:",len(df_test_under[df_test_under['TARGET'] == 1]))

Class 0: 24191
Class 1: 24191

```

3.2.2 Traitement des données manquantes :

On a remarqué qu'environ 19 colonnes de la base de données sont à plus que 50% vides donc ils n'aident pas à repartir ou à classifier les données, une raison pour laquelle on a décidé de supprimer ces dimensions car si on essaye de remplacer ces données manquantes par les valeurs les plus fréquentes on va biaiser la base de données.

Out[4]:

	missing values	missing percent
COMMONAREA_AVG	34806	71.9
NONLIVINGAPARTMENTS_AVG	34561	71.4
LIVINGAPARTMENTS_AVG	34094	70.5
FLOORSMIN_AVG	33909	70.1
YEARS_BUILD_AVG	33328	68.9
OWN_CAR_AGE	32624	67.4
LANDAREA_AVG	29971	61.9
BASEMENTAREA_MODE	29707	61.4
BASEMENTAREA_AVG	29707	61.4
NONLIVINGAREA_AVG	28167	58.2
EXT_SOURCE_1	28165	58.2
ELEVATORS_AVG	27304	56.4
APARTMENTS_MODE	26121	54.0
APARTMENTS_AVG	26121	54.0
ENTRANCES_AVG	25938	53.6
YEARS_BEGINEXPLUATATION_MODE	25896	53.5
LIVINGAREA_AVG	25798	53.3
FLOORSMAX_AVG	25681	53.1
YEARS_BEGINEXPLUATATION_AVG	25152	52.0
OCCUPATION_TYPE	13885	28.7
EXT_SOURCE_3	10240	21.2
NAME_TYPE_SUITE	172	0.4
EXT_SOURCE_2	107	0.2

En plus, on va remplacer les données manquantes dans les lignes (Après la numérisation) par la valeur la plus fréquente en utilisant le **SimpleImputer**.

```
imputer2=SimpleImputer(missing_values=np.nan,strategy='most_frequent')
trans_data_scaled = imputer2.fit_transform(data_scaled)

data_trans2=pd.DataFrame(trans_data_scaled)
data_trans2.isna().sum(axis=1)
```

3.2.3 Numérisation des données :

On a utilisé la bibliothèque **sklearn.preprocessing.LabelEncoder** afin de transformer les valeurs non numériques qui contiennent au maximum deux valeurs différentes à des valeurs numériques.

Pour le cas des valeurs numériques qui contiennent plus que deux valeurs différentes on a utilisé **pandas.getdummies()** pour éviter que notre algorithme donne des valeurs numériques très élevées qui vont biaiser notre base de données par la suite.

```
le = LabelEncoder()
le_count = 0

# Iterate through the columns
for col in data:
    if data[col].dtype == 'object':
        # If 2 or fewer unique categories
        if len(list(data[col].unique())) <= 2:
            # Train on the training data
            le.fit(data[col])
            # Transform both training and testing data
            data[col] = le.transform(data[col])
            #app_test[col] = le.transform(app_test[col])

            # Keep track of how many columns were label encoded
            le_count += 1

print('%d columns were label encoded.' % le_count)

4 columns were label encoded.
```

```
Entrée [7]: print('Data Features shape before encoding: ', data.shape)

# one-hot encoding of categorical variables
data = pd.get_dummies(data)

print('Data Features shape after encoding: ', data.shape)
```

```
Data Features shape before encoding: (48382, 41)
Data Features shape after encoding: (48382, 146)
```

3.2.4 Détection anomalies

Avant de continuer le traitement de données, on veut s'assurer que les valeurs existantes dans la base de données ne sont pas biaisées et ne vont pas affecter ou falsifier la suite du traitement du jeu de données.

```
Entrée [12]: # Searching for anomalies
(data['DAYS_BIRTH'] / -365).describe()
```

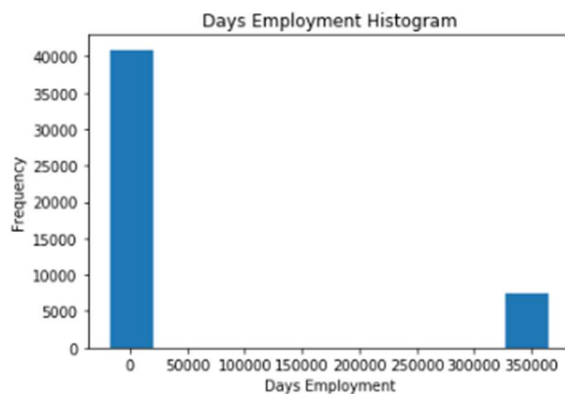
```
Out[12]: count    48382.000000
mean       42.509697
std        11.854790
min        21.021918
25%        32.542466
50%        41.275342
75%        52.150000
max        69.032877
Name: DAYS_BIRTH, dtype: float64
```

```
Entrée [13]: data['DAYS_EMPLOYED'].describe()
#The maximum value is about 1000 years.
```

```
Out[13]: count    48382.000000
mean    54505.667356
std    132647.379908
min   -17583.000000
25%   -2460.000000
50%   -1116.000000
75%   -332.000000
max    365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

```
Entrée [14]: import matplotlib.pyplot as plt
data['DAYS_EMPLOYED'].plot.hist(title = 'Days Employment Histogram');
plt.xlabel('Days Employment');
```

```
: import matplotlib.pyplot as plt
data['DAYS_EMPLOYED'].plot.hist(title = 'Days Employment Histogram');
plt.xlabel('Days Employment');
```



3.2.5 Normalisation des données

Les unités de mesures des dimensions de cette base de données sont incompatibles une raison pour laquelle une phase de normalisation est primordiale.


```

Entrée [13]: # Normalisation des données
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(data_trans)

X_scaled = scaler.transform(X)
data_scaled=pd.DataFrame(X_scaled)

```

3.2.6 Traitement des données :

Random forest

L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents. Ce modèle est très performant surtout lorsqu'on utilise un grand nombre d'arbres. Dans notre cas on va utiliser 100 arbres.

```

# Make the random forest classifier
random_forest = RandomForestClassifier(n_estimators = 100, random_state = 50, verbose = 1, n_jobs = -1)
# Train on the training data
random_forest.fit(X_train, y_train)

# Extract feature importances
feature_importance_values = random_forest.feature_importances_
feature_importances = pd.DataFrame({'feature': data_trans.columns, 'importance': feature_importance_values})

# Make predictions on the test data
predictions = random_forest.predict(X_test)

[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 10.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 0.4s finished

Entrée [17]: print("Accuracy : ",random_forest.score(X_test,y_test))
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,predictions))

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Accuracy : 0.66005291005291

```

	precision	recall	f1-score	support
0	0.66	0.67	0.66	6082
1	0.66	0.65	0.66	6014
micro avg	0.66	0.66	0.66	12096
macro avg	0.66	0.66	0.66	12096
weighted avg	0.66	0.66	0.66	12096

PCA et Random forest :

On remarque que la contribution de plusieurs variables n'est pas assez intéressante, pour cela on a diminué le nombre des variables par la méthode PCA (principale component anlysis) à 30 variable et après on a appliqué l'algorithme Random forest:

```
[20]: print("Accuracy : ",random_forest.score(X_test,y_test))
      print("Accuracy : ",random_forest.score(X_train,y_train))
      from sklearn.metrics import classification_report, confusion_matrix
      print(classification_report(y_test,predictions))
```

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor

Accuracy : 0.6324404761904762
Accuracy : 1.0

	precision	recall	f1-score	support
0	0.63	0.64	0.64	6082
1	0.63	0.63	0.63	6014
micro avg	0.63	0.63	0.63	12096
macro avg	0.63	0.63	0.63	12096
weighted avg	0.63	0.63	0.63	12096

[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 0.9s finished

KNN :

```
Entrée [15]: # tester un algorithme d'apprentissage
      from sklearn.neighbors import KNeighborsClassifier

      model = KNeighborsClassifier(n_neighbors=20)
      # training
      model.fit(X_train, y_train)

      # evaluation
      accuracy = model.score(X_test, y_test)
      print("Accuracy =", accuracy)
      print("# Misclassified points =", int(round((1-accuracy) * len(X_test))))

      accuracy = model.score(X_train, y_train)
      print("Accuracy =", accuracy)
      print("# Misclassified points =", int(round((1-accuracy) * len(X_train))))
```

Accuracy = 0.6003637566137566
Misclassified points = 4834
Accuracy = 0.6549909055834207
Misclassified points = 12519

```
Entrée [16]: from sklearn.metrics import classification_report, confusion_matrix
      print(classification_report(y_test,model.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.59	0.65	0.62	6082
1	0.61	0.55	0.58	6014
micro avg	0.60	0.60	0.60	12096
macro avg	0.60	0.60	0.60	12096
weighted avg	0.60	0.60	0.60	12096

LGBM:

```
import lightgbm as lgb                                # Create the model
model = lgb.LGBMClassifier(n_estimators=10000, objective = 'binary',
                           class_weight = 'balanced', learning_rate = 0.05,
                           reg_alpha = 0.1, reg_lambda = 0.1,
                           subsample = 0.8, n_jobs = -1, random_state = 50)

# Train the model
model.fit(train_features, train_labels, eval_metric = 'auc',
          eval_set = [(valid_features, valid_labels), (X_train, X_test)],
          eval_names = ['valid', 'train'], categorical_feature = cat_indices,
          early_stopping_rounds = 100, verbose = 200)
```

```
[Parallel(n_jobs=-1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed: 22.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 0.3s finished
```

```
Entrée [20]: predictions = model.predict(X_test)
print("Accuracy : ",model.score(X_test,y_test))
print("Accuracy : ",model.score(X_train,y_train))
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,predictions))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
Accuracy : 0.6324404761904762
```

```
Accuracy : 1.0
```

	precision	recall	f1-score	support
0	0.63	0.64	0.64	6082
1	0.63	0.63	0.63	6014

Remarque:

On peut remarquer que la précision du modèle Random forest est supérieur à celle de KNN et à celle de LGBM. KNN est moins adapté aux jeu de données avec un grand nombre de dimensions.

Etape 4 : Base de données NoSQL : MongoDB

4.1 Importer un fichier csv sur MongoDB

Pourquoi mongodb ?

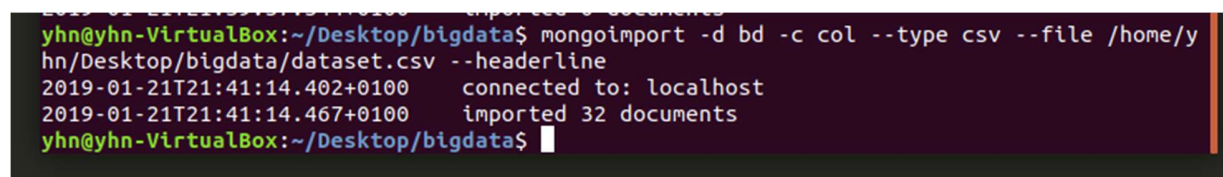
MONGODB est la 1ère base de données NoSQL. OpenSource, elle adopte un modèle de données de type document qui lui confère une grande souplesse d'utilisation et une vraie évolutivité. MongoDB permet de stocker et exploiter les données sous forme de documents JSON; un format compréhensible par tous (humain et machine), Ne dépend d'aucun langage (format d'échange de données ouvert). Comme ce format est très ouvert, il est pris en charge par de nombreux langages : JavaScript, PHP, Perl, Python, Ruby, Java,...

Comment importer les données?

Pour importer les données créées après la prédiction sur une base de données MongoDB on utilise la commande suivante :

```
$ mongoimport -d bd -c col --type csv --file /home/.../dataset.csv --headerline
```

Cette commande nous permet de pousser les données stockés dans le fichier (format csv) dataset.csv sur la collection "col" dans la base mongodb "bd".



```
yhn@yhn-VirtualBox:~/Desktop/bigdata$ mongoimport -d bd -c col --type csv --file /home/yhn/Desktop/bigdata/dataset.csv --headerline
2019-01-21T21:41:14.402+0100    connected to: localhost
2019-01-21T21:41:14.467+0100    imported 32 documents
yhn@yhn-VirtualBox:~/Desktop/bigdata$
```

Après avoir exécuté cette commande on remarque que la collection 'col' dans la base de données 'bd' contient des documents chacun représente une ligne du fichier csv c'est à dire un client.

```
> use bd
switched to db bd
> db.col.find().pretty()
{
  "_id" : ObjectId("5c462e6a3c66aa9dce25418b"),
  "TARGET" : 0,
  "NAME_CONTRACT_TYPE" : "Cash loans",
  "CODE_GENDER" : "F",
  "FLAG_OWN_CAR" : "N",
  "FLAG_OWN_REALTY" : "Y",
  "CNT_CHILDREN" : 0,
  "AMT_INCOME_TOTAL" : 202500,
  "AMT_CREDIT" : 610335,
  "AMT_ANNUITY" : 20299.5,
  "AMT_GOODS_PRICE" : 463500,
  "NAME_TYPE_SUITE" : "Family",
  "NAME_INCOME_TYPE" : "Working",
  "NAME_EDUCATION_TYPE" : "Higher education",
  "NAME_FAMILY_STATUS" : "Married",
  "NAME_HOUSING_TYPE" : "House / apartment",
  "REGION_POPULATION_RELATIVE" : 0.010147,
  "DAYS_BIRTH" : -17692,
  "DAYS_EMPLOYED" : -1294,
  "DAYS_REGISTRATION" : -5583,
  "DAYS_ID_PUBLISH" : -1232,
  "OWN_CAR_AGE" : "",
  "FLAG_MOBIL" : 1,
  "FLAG_EMP_PHONE" : 1,
  "FLAG_WORK_PHONE" : 0,
  "FLAG_CONT_MOBILE" : 1,
  "FLAG_PHONE" : 0,
  "FLAG_EMAIL" : 0,
  "OCCUPATION_TYPE" : "Core staff",
  "CNT_FAM_MEMBERS" : 2,
```

4.2 Application Web pour exploiter les données sur MongoDB

Dans cette partie on a créé une application web pour explorer les données après la phase de la prédiction et les envoyer vers la base de données NoSQL (mongodb). Pour cela on a utilisé le langage php et le driver mongodb pour assurer la connexion entre la base de données et la partie web.

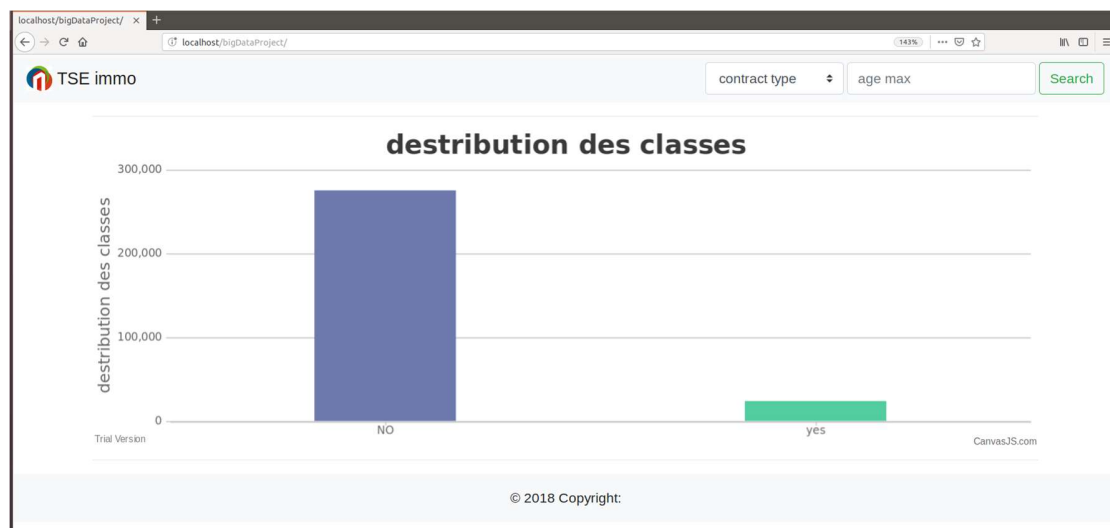
Sur la figure suivante, il y a la page de recherche par le type de contrat. On ajoutera aussi les recherches par d'autres critères et une page pour faire des statistiques (graphiques, histogrammes,...) pour pouvoir exploiter les résultats de la phase de prédiction.

Activities Firefox Web Browser jeu. 15:55 Mozilla Firefox

localhost/bigdata_proj/se... X + localhost/bigdata_proj/search_players.php 143%

TSE immo contract type age max SEARCH

target	NAME_CONTRACT_TYPE	aniv	taille(cm)	poids(kg)
0	Cash loans	a afficher +td	a afficher +td	a afficher +td
0	Cash loans	a afficher +td	a afficher +td	a afficher +td
0	Cash loans	a afficher +td	a afficher +td	a afficher +td
0	Cash loans	a afficher +td	a afficher +td	a afficher +td
0	Cash loans	a afficher +td	a afficher +td	a afficher +td
0	Cash loans	a afficher +td	a afficher +td	a afficher +td



Bibliographie:

- [1]: <https://fr.hortonworks.com/tutorial/learning-the-ropes-of-the-hortonworks-sandbox/>
- [2]: Tedder, s3-client-side-enryption, <https://github.com/tedder/s3-client-side-encryption> 16
Avril 2016
- [3]: <https://towardsdatascience.com/setting-up-and-using-jupyter-notebooks-on-aws-61a9648db6c5>
- [4] : <https://github.com/tca19/data-analysis>
- [5]: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.under_sampling.RandomUnderSampler.html
- [6] : <http://php.net/manual/fr/class.mongodb.php>
- [7] : <https://docs.mongodb.com/>