

# Chapitre 1

## MongoDB

### Code source

Voir <https://github.com/InternetHealthReport/tartiflette>.

### Création de l'environnement virtualdev

Voici la liste des dépendances :

```
(tartiflette -env) 162558@inet-bigd:~$ pip freeze
appdirs==1.4.3
awscli==1.15.50
backports.functools-lru-cache==1.5
backports.os==0.1.1
boto3==1.7.44
botocore==1.10.49
certifi==2018.4.16
chardet==3.0.4
colorama==0.3.9
configparser==3.5.0
cyclr==0.10.0
decorator==4.3.0
docutils==0.14
enum34==1.1.6
fs==2.0.23
fs-s3fs==0.1.8
future==0.16.0
futures==3.2.0
get==1.0.3
idna==2.7
jmespath==0.9.3
kiwisolver==1.0.1
matplotlib==2.1.1
networkx==1.11
numpy==1.14.5
pandas==0.23.1
pathlib==1.0.1
patsy==0.5.0
post==1.0.2
psycpg2==2.7.5
psycpg2-binary==2.7.5
public==1.0.3
py-radix==0.10.0
```

```

pyasn1==0.4.3
pygeoip==0.3.2
pymongo==3.6.1
pyparsing==2.2.0
python-dateutil==2.7.3
pytz==2018.4
PyYAML==3.12
query-string==1.0.2
request==1.0.2
requests==2.19.1
ripe.atlas.cousteau==1.4.2
rsa==3.4.2
s3transfer==0.1.13
scipy==1.1.0
six==1.11.0
socketIO-client==0.7.2
statsmodels==0.9.0
subprocess32==3.5.2
typing==3.6.4
urllib3==1.23
websocket-client==0.48.0

```

**Importation des traceroutes dans MongoDB** Voir README.md.

**Configurer l'analyse** Il faut mettre à jour le fichier de configuration de l'analyse disponible sur : <https://github.com/InternetHealthReport/tartiflette/blob/master/analysis/plot.py>  
Exemple :

```

{
    "comment": "60 min May and June 2015",
    "af": "",
    "minASNEntropy": 0.5,
    "alpha": 0.01,
    "end": {
        "$date": 1518393600000
    },
    "binMult": 3,
    "timeWindow": 4600,
    "minSeen": 3,
    "start": {
        "$date": 1517961600000
    },
    "minASN": 3,
    "nbProcesses": 24,
    "experimentDate": {
        "$date": 1546093776000
    },
    "confInterval": 0.05,
    "prefixes": ""
}

```

**Préparer les données avec la fonction getRTTData** Voir *getRttData* dans <https://github.com/InternetHealthReport/tartiflette/blob/master/analysis/plot.py>

**Traiter les liens : dessiner l'évolution de chaque lien** Voir *rttEvolution* dans <https://github.com/InternetHealthReport/tartiflette/blob/master/analysis/plot.py>

**Fichier Python : rttEvolution.py** Le fichier *rttEvolution.py* doit être créé au même endroit que le script *plot.py*.

```
#imports
from plot import *
import time
import datetime

start = time.time()
## Prepare data
a,b,c=getRttData()

## Process each link
finalResult=[]
for k, v in a.iteritems():
    result= rttEvolution([a[k],c[k]],k,"")
    finalResult.append(result)

## Save link's details
now = datetime.datetime.now()
filePath= str(now.strftime("%Y-%m-%d_%H-%M"))
fp = open('%s.json'%filePath, 'w')
for result in finalResult:
    json.dump(result, fp, default=str)

end = time.time()
print("\n Total Time is "+ str(end - start))
```

**Lancer une analyse** Etapes :

- configurer l'analyse ;
- importer les données vers MongoDB
- activer VirtualEnv ;
- lancer l'analyse via `python rttEvolution.py`.

## Chapitre 2

# Amazon S3 et Amazon Athena

**Créer les partitions dans Amazon S3** Voir l'arborescence dans le rapport de mémoire.

**Importer des traceroutes dans Amazon S3** Via l'interface graphique, l'API ou bien à travers AWS Command Line Interface (CLI).

**Créer la table dans Amazon Athena** Voir la structure dans le rapport de mémoire.

**Configurer l'analyse** Il faut mettre à jour le fichier de configuration appelé *getRttDataAthena.conf*.  
Un exemple de fichier de configuration :

```
{
    "comment" : "60 min May and June 2015",
    "af" : "",
    "minASNEntropy" : 0.5,
    "alpha" : 0.01,
    "end" : {
        "$date" : 1518307200000
    },
    "binMult" : 3,
    "timeWindow" : 3600,
    "minSeen" : 3,
    "start" : {
        "$date" : 1517961600000
    },
    "minASN" : 3,
    "nbProcesses" : 24,
    "experimentDate" : {
        "$date" : 1456747743895
    },
    "confInterval" : 0.05,
    "prefixes" : "",
    "msmIds" : [5004],
    "msmType" : "builtin"
}
```

**Préparer les données avec la fonction getRTTData**

**Traiter les liens : dessiner l'évolution de chaque lien**

**Fichier Python : rttEvolutionAthena.py** Voir <https://github.com/hayatbellafkih/RipeAtlasTraceroutesAnalysis/blob/master/2019/AWSAthenaTools/rttEvolutionAthena.py>

Ou :

```
## imports
from athenaTools import *

import time
start = time.time()
import datetime

## Prepare Data
a,b,c=getRttDataAthena()

## Process each link
finalResult =[]
for k, v in a.iteritems():
    result=rttEvolution([a[k],c[k]],k,"")
    finalResult.append(result)

## Save link's details
now = datetime.datetime.now()
filePath= str(now.strftime("%Y-%m-%d_%H-%M"))
fp = open('%s.json'%filePath, 'w')
for result in finalResult:
    json.dump(result, fp, default=str)

end = time.time()
print("Total Time is "+ str(end - start))
```

**Requête SQL pour la récupération des traceroutes** Nous présentons la requête SQL générique utilisée (Listing 2.1). Elle est formatée pour chaque période d'analyse afin de :

ligne 2 : ajouter les limites de la période en cours : le timestamp minimum et celui maximum ;

ligne 3 : choisir les partitions. Les identifiants de mesures sont ceux dans le fichier de configuration. Les partitions relatives à la date : *year*, *month* et *day* sont déduits de la période en cours.

Listing 2.1 – Requête SQL dans Athena

```
1 with dataset as( SELECT prb_id , "from" ,msm_id,  if (result[1].result[1].err is null and result
  [1].result is not null ,transform(result , x -> transform (x.result , entry -> Map(array[ if (
    entry."from" is not null , concat('' ,cast(entry."from" as varchar),'') , concat('' , 'none' , ''
    ')) ], array[if (entry."from" is not null ,entry."rtt" ,0)]  )),array[]) as datay from
  traceroutes_api
2 where  "timestamp" >= {} and "timestamp"< {} and
3 {}
4 ) select "from" ,prb_id ,msm_id ,datay as result from dataset ;
```

**Traiter les traceroutes obtenus depuis Amazon Athena** Nous avons regroupé les fonctions adaptées dans un seul fichier appelé *athenaTools.py*. Il est disponible sur : <https://github.com/hayatbellafkih/RipeAtlasTraceroutesAnalysis/blob/master/2019/AWSAthenaTools/athenaTools.py> Nous décrivons brièvement les fonctions dans ce fichier.

**load\_sql\_request\_by\_file** : la requête SQL présentée précédemment est créée dans un fichier<sup>1</sup> et cette fonction permet de récupérer cette requête comme étant une chaîne de caractère ;

---

1. <https://github.com/hayatbellafkih/RipeAtlasTraceroutesAnalysis/blob/master/2019/AWSAthenaTools/sqlRequestForAthena.sql>

**generateSQLAthenaRequest** : cette fonction consiste à formater la requête SQL avec les valeurs adéquates (Voir le paragraphe 2);

**computeRtt\_athena** : équivalente à la fonction *computeRtt* dans l'implémentation MongoDB;

**readOneTracerouteAllSigna** : fonction équivalente à la fonction *readOneTraceroute* dans l'implémentation MongoDB;

**mergeRttResults\_athena** : fonction équivalente à la fonction *mergeRttResults* dans l'implémentation MongoDB;

**getRttDataAthena** : fonction équivalente à la fonction *getRttData* dans l'implémentation MongoDB;

## Chapitre 3

# Apache Spark

La fonction *main* se trouve dans l'objet .

**L'application Spark** Il est possible de générer le fichier .jar avec Maven.

**Soumission de l'application** param1 : date début (format timestamp). param2 : date de fin (format timestamp) param3 : fenêtre (en secondes) param4 : endroit des fichiers de données param5 : chemin destination des résultats de l'analyse

**Exemple**