Projet GDELT

Sonia Bouden, Habib Aouani, Kaelig Castor, Cyrille Nouboué, Thomas Mensch

Plan

1. Objectif et contraintes
2. Choix techniques / Architecture
3. Les données et les requêtes
4. Etapes de réalisation
5. Budget
6. Axes d’amélioration et conclusions

Objectif et contraintes

Objectif

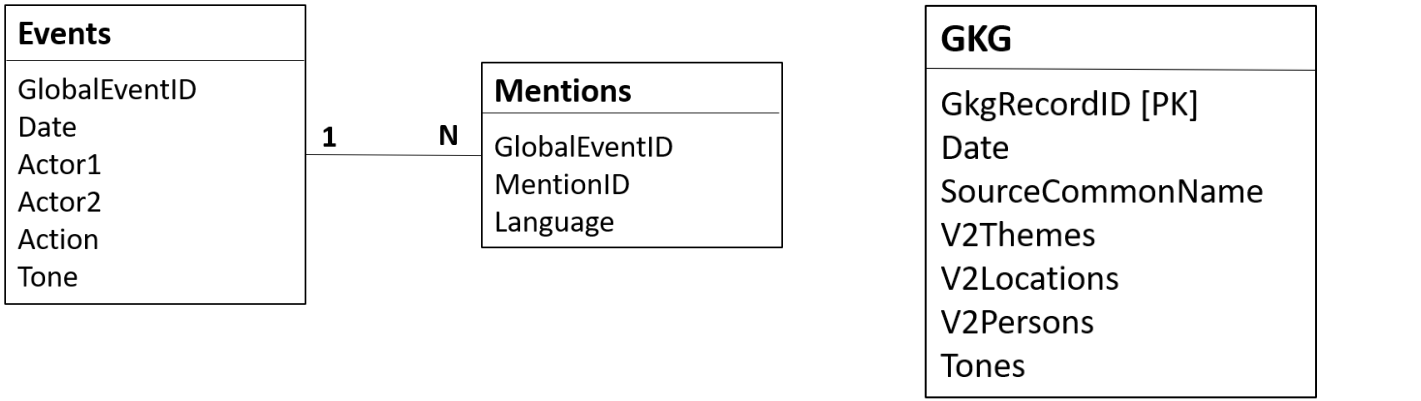
Concevoir un système qui permet d'analyser le jeu de données GDELT et ses sources de données

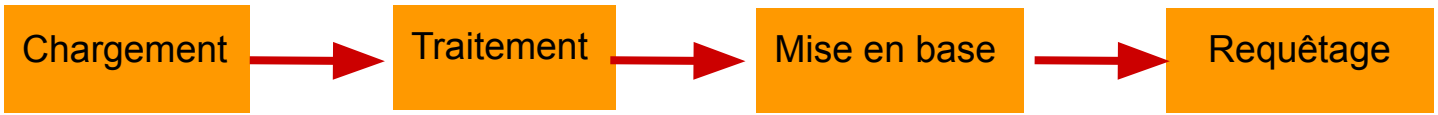
Contraintes

- Utiliser au moins 1 technologie du cours (SQL / Cassandra / Spark)
- Concevoir un système distribué et tolérant aux pannes
- Charger une année de données dans votre cluster
- Déployer le cluster sur AWS (compte Amazon Educate) / Budget < 300 euros

Les données

- Collection de fichiers .csv compressés ~700 Go pour l'année 2019. (2Go)
- 3 tables - Events, Mentions et GKG
  - Article de langue anglaise / Autres (traduction automatique)



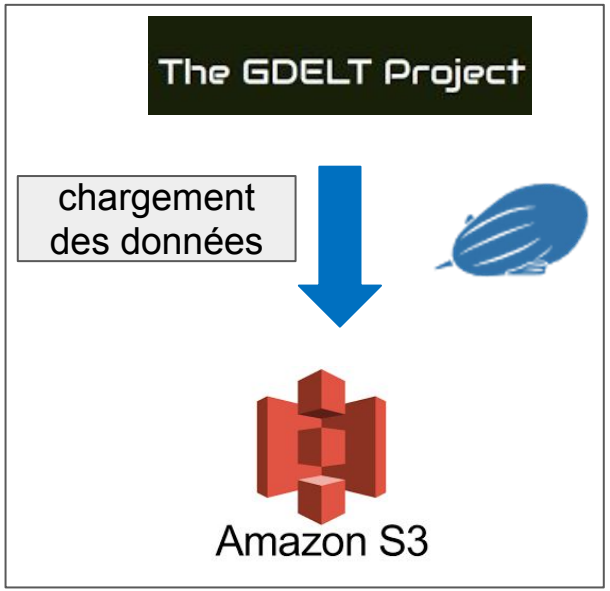


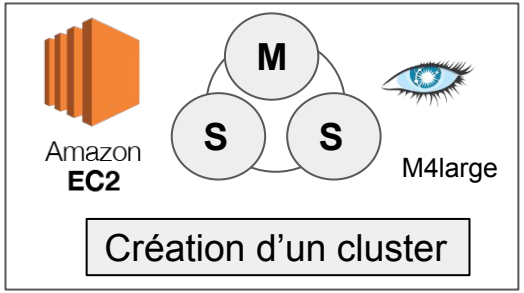
Choix techniques

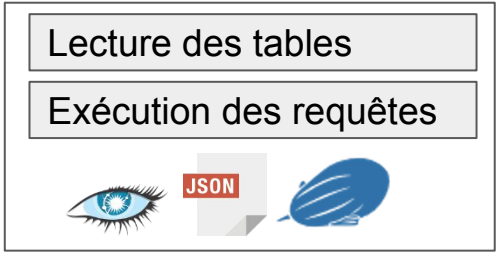
Approche et technologies utilisées

- **Approche**
  - Machine locale -- Comprendre les données / requêtes
  - AWS (*Amazon Educate)* -- Configuration de l'infrastructure / Mise en production
- **Cassandra**
  - Stockage de gros volumes de données / passage à l'échelle
  - Robustesse / résilience, RF à 3
- **Spark**
  - Moteur de traitement de données très performant (nécessite bcp de mémoire)
  - Structure DataFrame -- flexibilité dans la manipulation des données

Architecture



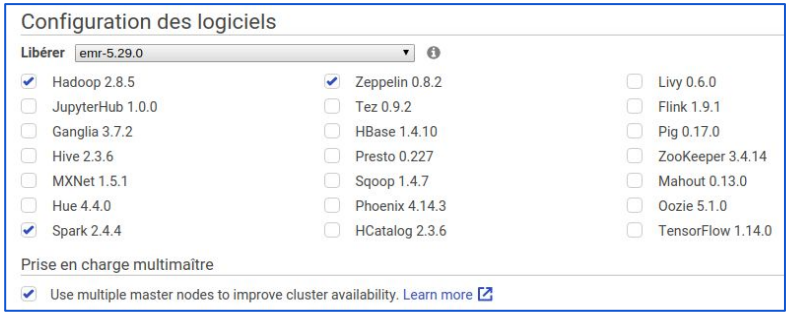




Étapes de réalisation

1. Créer des instances EC2 en utilisant EMR
2. Paramétrer Cassandra et Zeppelin
3. Créer les tables dans Cassandra
4. Charger les données sur Zeppelin
5. Vérifier le résultat des requêtes

Créer des instances sur EC2



Node type	Type d'instance	Nombre d'instances	Option d'achat	Auto Scaling
Maître Groupe d'instances maître - 1	**m4.large** 4 Cœurs virtuels, 8 GiO de mémoire, stockage EBS uniquement Stockage sur EBS : 32 Gio Ajouter des paramètres de configuration	3 Instances Prise en charge multimaître activée	A la demande Spot Utiliser le prix à la demande comme prix ma›	Not available for Master
Principal Groupe d'instances principal - 2	**m4.large** 4 Cœurs virtuels, 8 GiO de mémoire, stockage EBS uniquement Stockage sur EBS : 32 Gio Ajouter des paramètres de configuration	2 Instances	A la demande Spot Utiliser le prix à la demande comme prix ma›	Not enabled

ID d'instance	Type d'instance	Zone de disponib	État de l'instance	Contrôles des s	Statut des alarmes	DNS public (IPv4)	IP publique IPv4
i-076d5ab6780f6372b	m4.large	us-east-1e	running	2/2 contrôle…	Aucun(e)	ec2-34-207-155-109.co…	34.207.155.109
i-0987c8a02561e21b7	m4.large	us-east-1e	running	2/2 contrôle…	Aucun(e)	ec2-54-237-67-10.com…	54.237.67.10
i-09f1a783eb7b4cae7	m4.large	us-east-1e	running	2/2 contrôle…	Aucun(e)	ec2-18-234-107-154.co…	18.234.107.154
i-09f74335eecbb0455	m4.large	us-east-1e	running	2/2 contrôle…	Aucun(e)	ec2-18-207-1-99.comp…	18.207.1.99
i-0f469fd37ac746272	m4.large	us-east-1e	running	2/2 contrôle…	Aucun(e)	ec2-54-157-134-204.co…	54.157.134.204

Installation de cassandra sur les noeuds

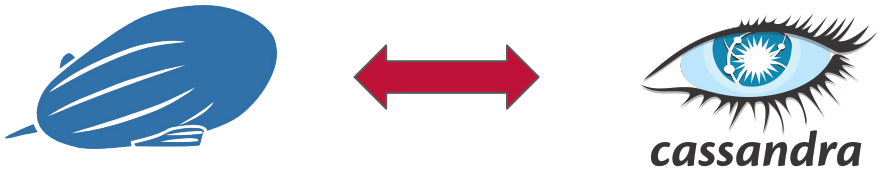
- **Configuration des paramètres sur cassandra.yaml**
  - seeds
  - listen_address
  - rpc_address
  - endpoint_snitch: GossipingPropertyFileSnitch
  - auto_bootstrap: false

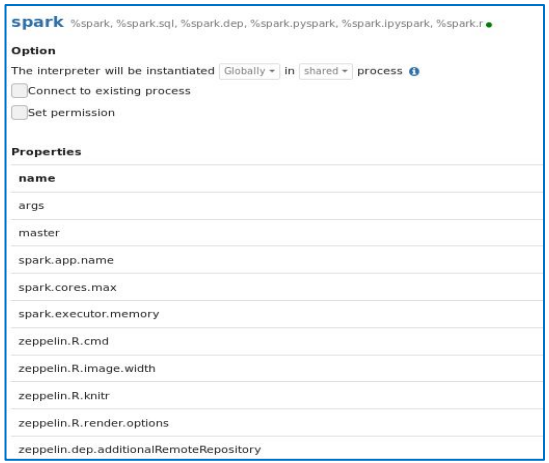
- **Configuration des propriétés sur cassandra-rackdc.properties**
  - 1 rack
  - 1 data center

```
[ec2-user@ip-172-31-87-57 ~]$ sudo nodetool status
Datacenter: dc1
===============
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address        Load       Tokens       Owns (effective)  Host ID                               Rack
UN  172.31.83.94   104.8 KiB  256          39,9%             0264b58b-e26c-4745-ad6d-70e3d5d00878  rack1
UN  172.31.87.57   103.67 KiB  256          39,3%             42e5888a-6020-48ae-92e7-2b65b4c9d85e  rack1
?N  172.31.81.86   69.96 KiB  256          40,3%             10ebaa82-c9d4-493b-9cb8-8e0d0419207f  rack1
UN  172.31.87.67   172.11 KiB  256          41,1%             23a8afe4-1e91-4c95-8558-268b3482e45b  rack1
?N  172.31.91.18   69.96 KiB  256          39,3%             179f2e5d-08f5-4e4b-aeea-590d7df131a6  rack1
```

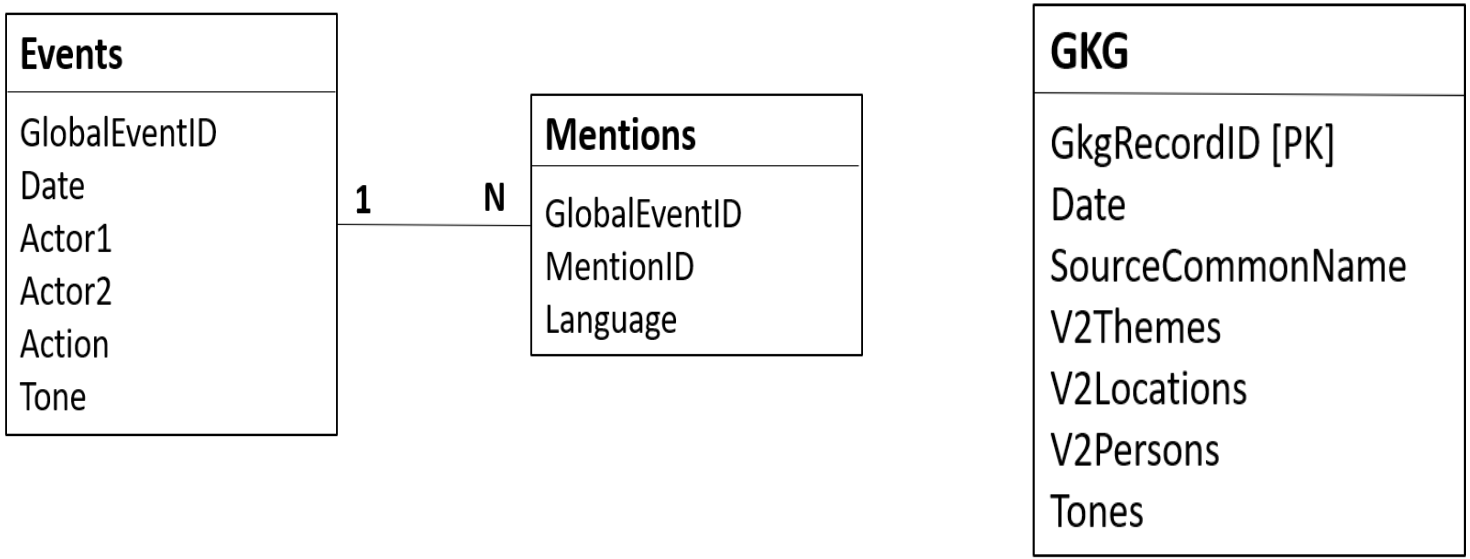
Configuration des interpréteurs sur Zeppelin

- **Connexion à Cassandra**
  - spark.jars.packages => datastax:spark-cassandra-connector:2.4.0-s_2.11
  - spark.cassandra.connection.host
  - Ajouter un nouveau repository dans la configuration de zeppelin pour prise en compte de l'upgrade Zeppelin





Les données et les requêtes



Requête 1 – *Afficher le **nombre d'articles / évènements** qu'il y a eu pour chaque triplet (**jour**, **pays de l'évènement**, **langue de l'article**).*

```
// Join dataframe
val dfRequest1 = dfMentionsSelect
    .dropDuplicates("GlobalEventID", "MentionIdentifier")
    .join(dfEventsSelect, "GlobalEventID")
    .groupBy("GlobalEventID", "Year", "Month", "Day", "Language", "ActionGeoCountryCode")
    .agg(count("MentionIdentifier").as("numArticles"))


dfRequest1.show()
```

SPARK JOB FINISHED

```
+-------------+----+-----+---+--------+-------------------+-----------+
|GlobalEventID|Year|Month|Day|Language|ActionGeoCountryCode|numArticles|
+-------------+----+-----+---+--------+-------------------+-----------+
|    890018652|2019|   12|  1|     eng|                 LE|          1|
|    890019133|2019|   12|  1|     eng|                 UK|         10|
|    890019133|2019|   12|  1|     spa|                 UK|          3|
|    890019133|2019|   12|  1|     tur|                 UK|          8|
|    890019293|2019|   12|  1|     eng|                 US|          3|
```

Spark

```
Connected to Test Cluster at 172.31.14.27:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> select * from gdelt_project.request1;

 year | month | day | actioncountry | language | eventid   | numarticles
------+-------+-----+---------------+----------+-----------+-------------
 2019 |    12 |   1 |            JE |      tel | 890029206 |           2
 2019 |    12 |   1 |            SZ |      tur | 890031925 |           2
 2019 |    12 |   1 |            SZ |      tur | 890096104 |           2
 2019 |    12 |   1 |            SZ |      tur | 890104198 |          10
 2019 |    12 |   1 |            SZ |      tur | 890104199 |          10
 2019 |    11 |  24 |            AE |      rus | 890075075 |           2
```

cassandra

Jointure entre Events et Mentions
+ agrégation

⇒ 1 table agrégée dans Cassandra

Requête 2 – *Pour un **pays donné** en paramètre, affichez **les évènements** qui y ont eu place triées par le **nombre de mentions** (tri décroissant); permettez une agrégation par **jour / mois / année**.*

```
val dfRequest2 = dfMentionsSelect
    .join(dfEventsSelect.filter(!(s"ActionGeoCountryCode" === "")), "GlobalEventID")
    .groupBy("ActionGeoCountryCode", "Year", "Month", "Day", "GlobalEventID")
    .agg(count("GlobalEventID").as("NumMentions"))
    .orderBy(s"ActionGeoCountryCode", s"Year", s"Month", s"Day", desc("NumMentions"), s"GlobalEventID")
```

FINISHED

Spark

```
Connected to Test Cluster at 172.31.11.150:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> select * from gdelt_project.request2;

 actioncountry | year | month | day | nummentions | eventid
---------------+------+-------+-----+-------------+-----------
            BA | 2019 |    11 |  24 |           4 | 890127904
            BA | 2019 |    11 |  24 |           1 | 890018549
            IS | 2019 |    12 |   1 |         106 | 890047648
            IS | 2019 |    12 |   1 |         101 | 890018665
            IS | 2019 |    12 |   1 |          92 | 89
            IS | 2019 |    12 |   1 |          90 | 89
            IS | 2019 |    12 |   1 |          88 | 89
```

cassandra

Jointure entre Events et Mentions
\+ agrégation

⇒ 1 table agrégée dans Cassandra

Requête 3 – *Pour une **source de données** passée en paramètre (`gkg.SourceCommonName`) affichez les **thèmes**, **personnes**, **lieux** dont les articles de cette source parlent, ainsi que le nombre d'articles et le ton moyen des articles (pour chaque thème/personne/lieu); permettez une agrégation par **jour** / **mois** / **année**.*

Spark

```
// Themes
// Explode + cleaning
val df3GkgThemes = dfGkg_3
```

```
// Persons
val df3GkgPersons = dfGkg_3
```

```
// Locations
val df3GkgLocations = dfGkg_3
    .withColumn("Location", explode_outer($"V2Locations"))
    .withColumn("Location", split($"Location", "#")(2))
    .select("GkgRecordID",
            "SourceCommonName",
            "Year", "Month", "Day",
            "Tone", "Location")
    .filter(!($"Location" === ""))
    .dropDuplicates("GkgRecordID", "Location")

// Aggregation
val df3GkgLocationsAgg = df3GkgLocations
    .groupBy("SourceCommonName", "Year", "Month", "Day", "Location")
    .agg(count($"GkgRecordID").as("NumArticles"),
         avg($"Tone").as("AvgTone"))
```

cassandra

```
Connected to Test Cluster at 172.31.14.237:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> select * from gdelt_project.request31;

 sourcecommonname | year | month | day | theme
```

```
cqlsh> select * from gdelt_project.request32;

 sourcecommonname | year | month | day | person | avgtone | numarticles
```

```
cqlsh> select * from gdelt_project.request33;

 sourcecommonname | year | month | day | location | avgtone | numarticles
------------------+------+-------+-----+----------+---------+------------
 bearingarms.com | 2019 | 12 | 1 | FR | -4 | 1
 bearingarms.com | 2019 | 12 | 1 | IN | -1 | 1
 bearingarms.com | 2019 | 12 | 1 | PK | -1 | 1
 bearingarms.com | 2019 | 12 | 1 | UK | -1 | 1
 bearingarms.com | 2019 | 12 | 1 | US | -2 | 4
 harwichandmanningtreestandard.co.uk | 2019 | 12 | 1 | AS | 0 | 1
 harwichandmanningtreestandard.co.uk | 2019 | 12 | 1 | BE | 1 | 1
 harwichandmanningtreestandard.co.uk | 2019 | 12 | 1 | CA | 0 | 1
 harwichandmanningtreestandard.co.uk | 2019 | 12 | 1 | CH | -6 | 2
 harwichandmanningtreestandard.co.uk | 2019 | 12 | 1 | CI | -2 | 1
```

agrégation sur la table GKG

⇒ 3 tables agrégées dans Cassandra

Requête 4 – *Dresser la cartographie des relations entre les **pays** d'après **le ton** des articles: pour chaque paire (**pays1**, **pays2**), calculer le **nombre d'article**, le **ton moyen** (agrégations sur **jour** / **mois** / **année**, filtrage par **pays** ou carré de coordonnées)*

Spark

```
val request4 = dfMentionsAgg
    .join(dfEventsSorted, "eventid")
    .select("actor1countrycode", "actor2countrycode",
            "day", "month", "year", "avgtone", "numarticles")

request4: org.apache.spark.sql.DataFrame = [actor1countrycode: string, actor2countrycode: string ... 5 more fields]
```

FINISHED

Jointure entre Events et Mentions
\+ agrégation

⇒ 1 table agrégée dans Cassandra

```
cqlsh:gdelt_project> select * from request4;

 actor1countrycode | actor2countrycode | year | month | day | avgtone   | numarticles
-------------------+-------------------+------+-------+-----+-----------+-------------
               VNM |               JPN | 2019 |    12 |   1 |   1.25294 |           1
               USA |               SOM | 2019 |    12 |   1 |  -8.05687 |           1
               MCO |               ARE | 2019 |    12 |   1 |  -1.42857 |           1
               KHM |               KHM | 2019 |    12 |   1 |  -6.66667 |           1
               SYR |               IRL | 2019 |    12 |   1 |  -4.09091 |           1
               EUR |               GMB | 2019 |    12 |   1 | -0.242718 |           1
               EGY |               NOR | 2019 |    12 |   1 | -0.704225 |           1
               EST |               LTU | 2019 |    12 |   1 |  0.884956 |           1
               BHS |               LBY | 2019 |    12 |   1 |  -2.71605 |           1
               CHN |               YEM | 2019 |    12 |   1 |  -3.31675 |           1
               VNM |               EUR | 2019 |    12 |   1 | -10.47619 |           1
               CHN |               BRN | 2019 |    12 |   1 |  0.630517 |           1
               ESP |               CUB | 2019 |    12 |   1 |  -2.20501 |           1
               SAU |               MLI | 2019 |    12 |   1 |  -8.65385 |           1
               USA |               IND | 2019 |    11 |  30 | -0.995025 |           2
               USA |               IND | 2019 |    12 |   1 |  -2.11566 |           1
               BLR |               POL | 2019 |    12 |   1 |  -2.03291 |           1
```

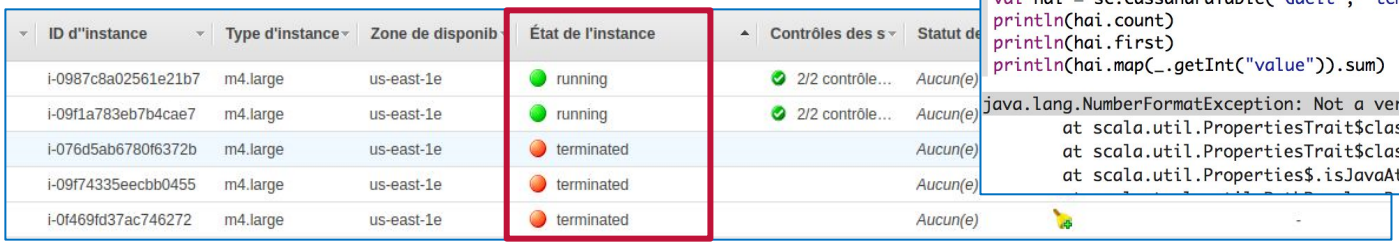
cassandra

Problèmes rencontrés

```
%spark
import com.datastax.spark.connector._
import org.apache.spark.sql.cassandra._
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf

val conf = new SparkConf(true)
        .set("spark.cassandra.connection.host", "172.31.22.102")
        .set("spark.cassandra.auth.username", "cassandra")
        .set("spark.cassandra.auth.password", "cassandra")
        .set("spark.driver.allowMultipleContexts", "true")


//val sc = new SparkContext(conf)

<console>:23: error: object datastax is not a member of package com
       import com.datastax.spark.connector._
                  ^
```

- Les masters tombent fréquemment sur certains **comptes (Educate)** et impossibilité de redémarrer ou changer les droits et niveaux de sécurité
- Les erreurs rencontrées avec les interpréteurs **Zeppelin**:
  - Zeppelin de AWS n'est pas exhaustif
  - Difficulté pour installer l'interpréteur Cassandra
  - Zeppelin ne supporte pas le https dans le management de dépendances

- Problèmes liés à la mémoire: Broken pipe => Connection refusée => relancement du tunnel ssh Zeppelin
- Incompatibilité des versions de Zeppelin d'un compte à un autre
- **Incompréhension par rapport au temps d'exécution / de la simple exécution ou non, d'une même requête d'un jour à l'autre et d'un compte à l'autre**




```
%sparkcassandra
import org.apache.spark.sql.cassandra._

val hai = sc.cassandraTable("Gdelt", "temp1")
println(hai.count)
println(hai.first)
println(hai.map(_.getInt("value")).sum)

java.lang.NumberFormatException: Not a version: 9
        at scala.util.PropertiesTrait$class.parts$1(Properties.scala:184)
        at scala.util.PropertiesTrait$class.isJavaAtLeast(Properties.scala:187)
        at scala.util.Properties$.isJavaAtLeast(Properties.scala:17)
```

Budget

SERVICES :						COUTS PREVUS :	Compte 1	Compte 2
Nombre de requêtes						4		4
Nombre d'heures par requête						60		
Nombre total d'heures						240		
Cout par heure ($)	m4.large	2	6,5	8 Gio	EBS uniquement	0,1		
Coût total prévu ($)						24	83	28

Cout par heure avec (InstaClustr)	m4.large	2	6,5	8 Gio	EBS uniquement	0,32	
Coût total prévu ($)						76,8	

Conclusion

Axes d'amélioration

- Dimensionnement de l'infrastructure
  - Mieux cadrer les besoins et évaluer la taille du cluster
  - Séparer Spark (mémoire) et Cassandra (stockage)
- Automatisation des tâches
  - Configuration des clusters (e.g. ANSIBLE)
  - Utilisation d'une image contenant déjà Cassandra (docker)
- Optimisation des requêtes / Visualisation
  - Optimiser les requêtes avant la jointure et l'agrégation
  - Créer une interface pour l'accès aux résultats des requêtes
- ...

ANSIBLE

Conclusions

- Ecriture des requêtes en local sur quelques jours de données
- Chargement de 1 an de données sur AWS S3 (année 2019)
- Configuration d'un cluster "simple" sur AWS pour la phase de test
  - Beaucoup de problèmes techniques
- Echec du passage à l'échelle
- Intérêt pédagogique de l'exercice
  - Difficulté de concevoir une infrastructure robuste
  - Enjeux du passage à l'échelle

Démo