



INF728 Projet Bigdata 2020-2021

Etudier l'évolution de la pandémie COVID19 via son impact media

-

Choix d'une infrastructure de stockage et analyse

Encadré par :

- M. ARION Andrei

Présenté par :

- BENALI Amal
- BINUANI Nicolas
- JAIT Fatima-Ezzahra
- JIA Delin
- TANKIPINOU Celia

PLAN :

- 1.Présentation et choix de l'architecture
- 2.Modélisation des données et requêtage
- 3.Budget
- 4.Performances, limites et amélioration
- 5.Demo

I. ARCHITECTURE

-IDÉE PRINCIPALE :

Vision métier de la DATA et leurs applications

-Développement :

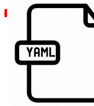
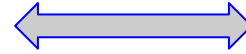
Ce qui a été réalisé VS points bloquants

-Améliorations :

Ce qui pourrait être implémenté ...

CLI = Command Line Interface
-> automatisation du déploiement infra

Local

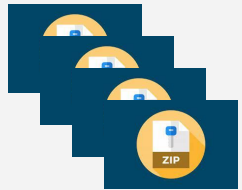


ANSIBLE

Yml files = fichiers de configuration des clusters spark / db (via docker)

DATA
engineer

Data Lake - S3 Bucket



Zip files :
-simple
-translations

Cluster SPARK - TRAITEMENT + IA



Master

Slave



Slave

Prétraitement

- ZipToCsv
 - Event
 - Gkg
 - mentions

Traitement

- Dataframe (agreg / filter ...)

Spark  **Scala** 

Cluster DB

Stockage des tables BU

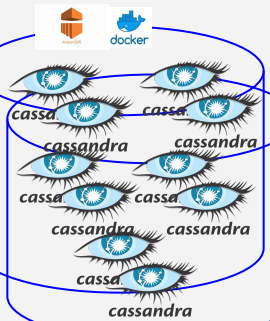


table associée à sa BU

API Gateway



- Dashboard
- Reporting
- APIs

BI
DataViz

AWS - VPC



Data
Scientist

• Choix d'architecture :



Les avantages :

- Facilité d'emploi tout en réduisant le Coût
- Stockage des données hautement disponible et durable
- Exécution de requêtes sur place

Les avantages :

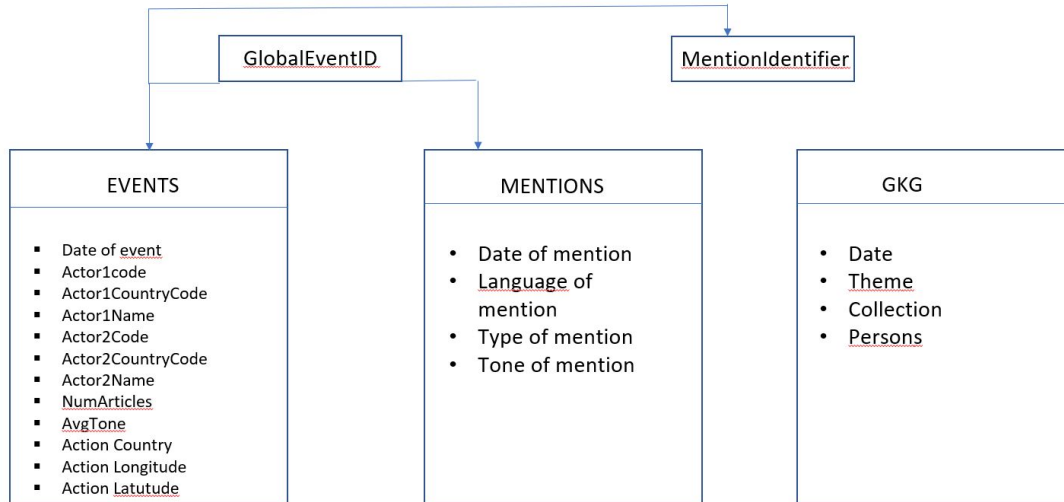
- Coûts avantageux
- Fiabilité : EMR est optimisé pour le cloud et surveille en permanence le cluster.
- Flexibilité : contrôle total sur votre cluster avec un accès racine à chaque instance.
- Simplicité d'utilisation: il s'occupe du provisionnement, de la configuration et de l'optimisation des clusters
- Facilité d'exploitation de l'infrastructure SPARK sur amazon EMR

Les avantages :

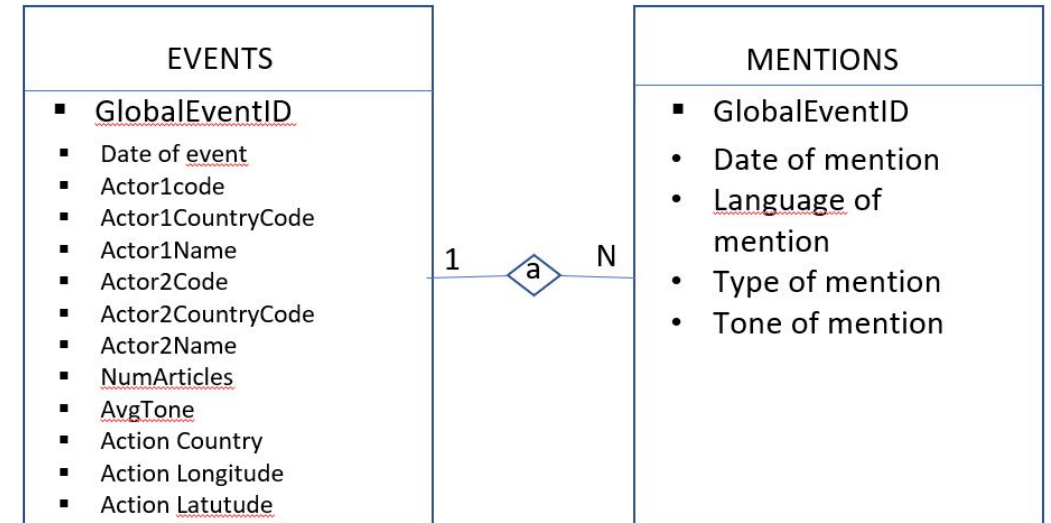
- Haute disponibilité
- Open Source
- Tolérance aux pannes (grâce aux mécanismes de réplication de données)
- Meilleure scalabilité sur données formatées
- Scalabilité linéaire (fonction nombre node)

II. Modélisation des données

Type de données: EVENTS, MENTIONS, GKG



Relation entre les données



Data-Preprocessing et requête



<p>Requête 1: Afficher le nombre d'articles/événements qui parlent de COVID qu'il y a eu pour chaque triplet (jour, pays de l'évènement, langue de l'article)</p> <p>READY</p>	<p>Requête2: Pour un pays donné en paramètre, affichez les événements qui y ont eu place triées par le nombre de mentions (tri décroissant); permettez une agrégation par jour/mois/année</p>	<p>Requête3: Pour une source de données passée en paramètre (<u>gkg.SourceCommonName</u>) affichez les thèmes, personnes, lieux dont les articles de cette sources parlent ainsi que le <u>le</u> nombre d'articles et le ton moyen des articles (pour chaque thème/personne/lieu); permettez une agrégation par jour/mois/année.</p>
<p>Table Event Table Mention Table GKG</p> <p>↓</p> <pre>R1_final = R1_themes2 .filter(col("themes3").contains("COVID")) .filter(\$"ActionGeo_CountryCode" != "") .groupBy("ActionGeo_CountryCode", "DATE", "Langue", "GLOBALEVENTID") .count().sort(desc("count"))</pre>	<p>Table Event Table Mention</p> <p>↓</p> <pre>val question2 = event_mention_DF.select("ActionGeo_CountryCode", "Day", "MonthYear", "Year", "GLOBALEVENTID").filter(\$"ActionGeo_CountryCode" != "").groupBy("ActionGeo_CountryCode", "Day", "MonthYear", "Year", "GLOBALEVENTID").count().sort(\$"count".desc)</pre>	<p>Table GKG</p> <p>↓</p> <pre>val gkgDF_R3_persons1 = gkgDF_R3.na.drop().withColumn("Persons",split(regexp_replace(\$"persons", "(^\\[\\[\\]) (\\[\\]\\])", ""), ";")).withColumn("v2tone2",split_tone_udf(\$"v2tone").drop("v2locations", "v2tone", "themes")) val gkgDF_R3_persons2 = gkgDF_R3_persons1.withColumn("date2", \$"date".cast(StringType)).withColumn("monthyear", compute_month_udf(\$"date")).withColumn("year", compute_year_udf(\$"date")).withColumn("v2tone3", \$"v2tone2".cast(FloatType)).drop("date", "v2tone2") val gkgDF_R3_persons3 = gkgDF_R3_persons2.withColumn("persons3", explode(\$"persons")).drop("persons")</pre>

III. Budget



S3:

- S3 Standard: First 50 TB / Month: \$0.023 per GB



amazon
EMR

EMR:

- m5.xlarge: vCPU 4, Mémoire 16 Gio, 0,192 USD par heure

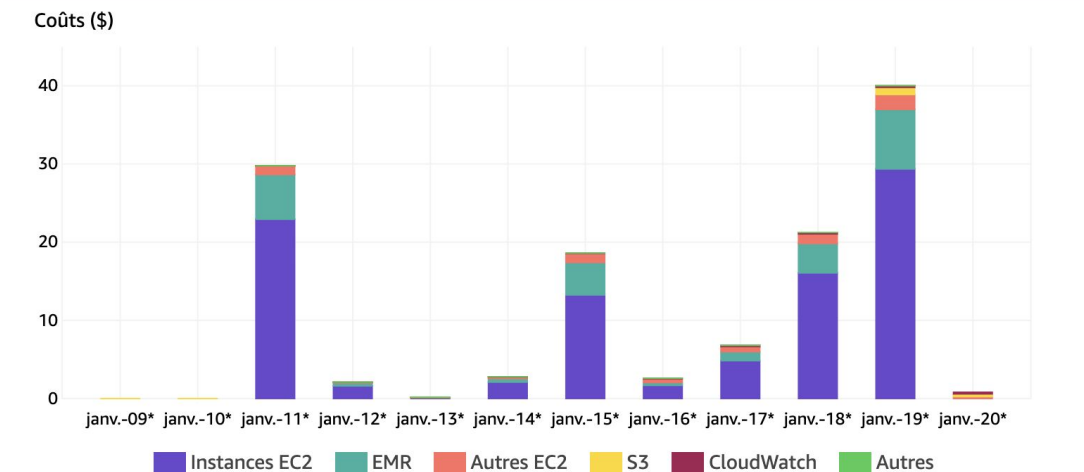
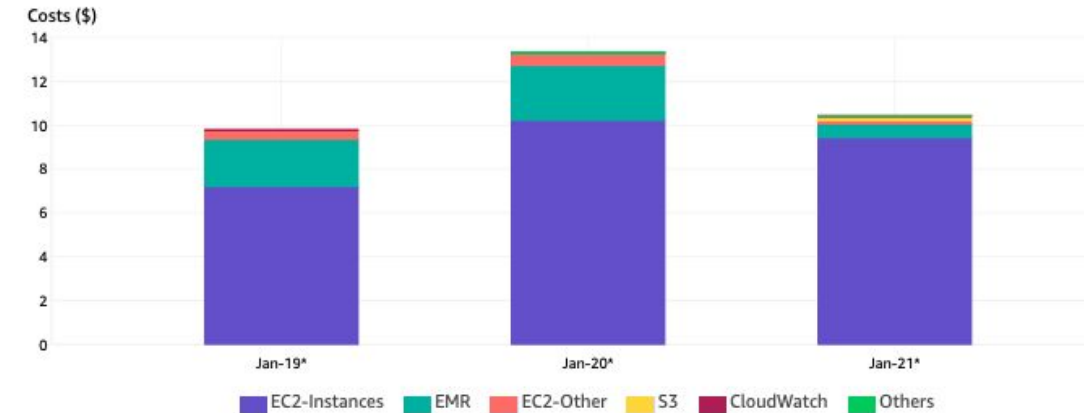


Amazon
EC2

EC2:

- t2.micro: vCPU 1, Mémoire 1 Gio, 0,0116 USD par heure
- m5.xlarge: vCPU 4, Mémoire 16 Gio, 0,192 USD par heure

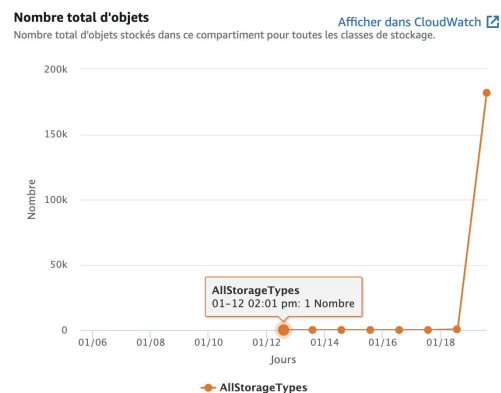
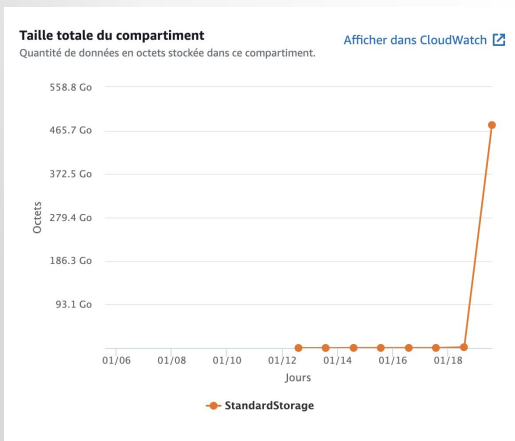
Problème rencontré: nombreux problèmes de connection failed et timeOut à cause de faible capacité de t2.micro



IV. Performance, limites et améliorations :

Performance :

- ❖ Volumétrie pour un an : 500Go
- ❖ Résilience en panne :
 - mise en panne d'un nœud n'a aucun effet sur le fonctionnement du cluster
- ❖ stockage durable



Limites/Améliorations :

- ❖ La perte de temps vu la configuration manuelle d'EC2
- ❖ problème de l'authentification au niveau d'EC2
- ❖ Taille de Données
- ❖ Connecter Zeppelin avec EC2 afin de visualiser les résultats
- ❖ Sécurité: VPC cloud amazon

V. Démonstration

