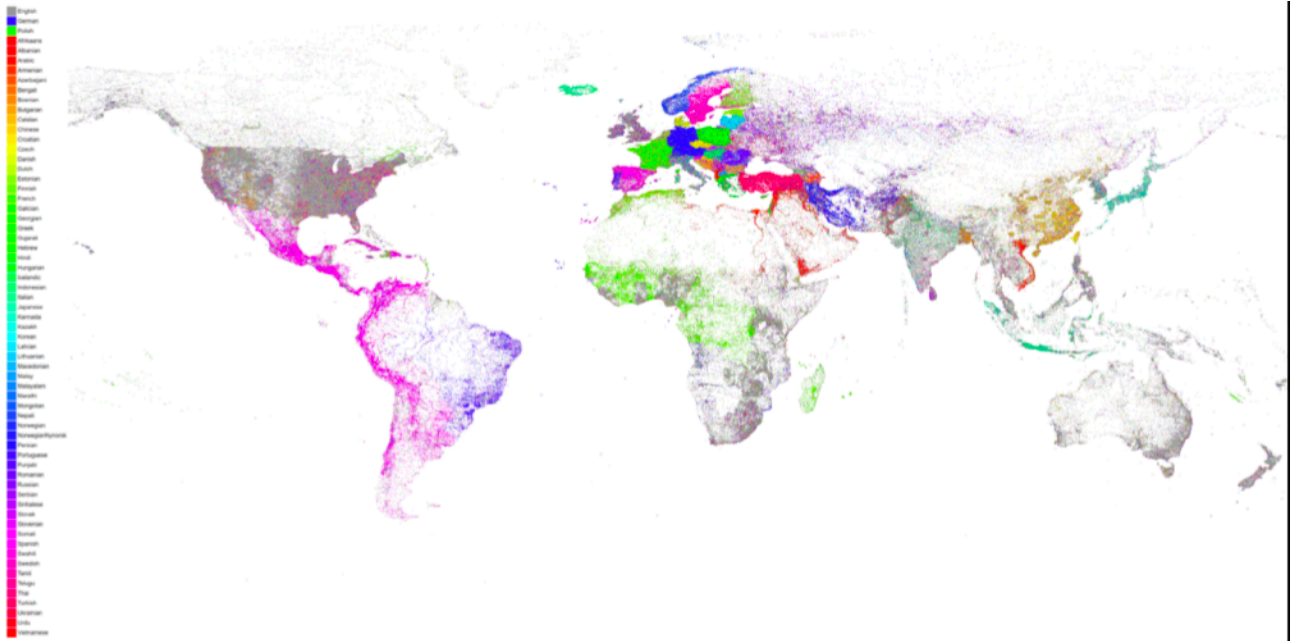


Projet Big Data

1. Présentation du projet

But : Analyser l'année 2018 via la base de données GDELT en utilisant le tone des articles dans les médias des divers pays du monde. "The Global Database of Events, Language, and Tone, est une initiative pour construire un catalogue de comportements et de croyances sociales à travers le monde, reliant chaque personne, organisation, lieu, thème, source d'information, et événement à travers la planète en un seul réseau massif qui capture ce qui se passe dans le monde, le contexte, les implications ainsi que la perception des gens sur chaque jour".



Architecture : Il faut concevoir un système pour analyser l'évolution des relations entre les différents pays en utilisant le ton des mentions (positives/négatives) dans les articles medias de chaque pays.

Nous disposons de trois types de fichiers :

- les events (schema : <https://bigquery.cloud.google.com/table/gdelt-bq:gdeltv2.events?tab=schema&pli=1>, CAMEO Ontology : <http://data.gdeltproject.org/documentation/CAMEO.Manual.1.1b3.pdf>, documentation: http://data.gdeltproject.org/documentation/GDELT-Event_Codebook-V2.0.pdf)
- les mentions (schema: <https://bigquery.cloud.google.com/table/gdelt-bq:gdeltv2.eventmentions>, documentation: http://data.gdeltproject.org/documentation/GDELT-Event_Codebook-V2.0.pdf)
- le graph des relations ⇒ GKG, Global Knowledge Graph (schema: <https://bigquery.cloud.google.com/table/gdelt-bq:gdeltv2.gkg>, documentation: http://data.gdeltproject.org/documentation/GDELT-Global_Knowledge_Graph_Codebook-V2.1.pdf)

L'index des fichiers est expliqué ici :

- masterlelist.txt Master CSV Data File List – English (<http://data.gdeltproject.org/gdeltv2/masterfilelist.txt>)
- masterlelist-translation.txt Master CSV Data File List – GDELT Translingual (<http://data.gdeltproject.org/gdeltv2/masterfilelist-translation.txt>)

L'accès aux données se fait via :

- access libre sur <http://data.gdeltproject.org/gdeltv2/>

- disponible également sur Google BigQuery : <https://www.gdeltproject.org/data.html#googlebigquery>
- pratique pour explorer la structure des données, generation des types de données

L'objectif étant de proposer un système de stockage distribué, résilient et performant sur AWS pour les données de GDELT. Les fonctionnalités de base attendues sont les suivantes ;

- le nombre d'articles/événements pour chaque (jour, pays de l'événement, langue de l'article)
- pour un acteur(pays/organisation ...) -> afficher les événements qui y font reference
- les sujets (acteurs) qui ont eu le plus d'articles positifs/négatifs (mois, pays, langue de l'article).
- acteurs/pays/organisations qui divisent le plus

Les fonctionnalités supplémentaires éventuelles sont :

- comment mieux comprendre l'évolution des relations entre les différents pays

Les contraintes du projet sont les suivantes :

- au moins 1 technologie vue en cours en expliquant les raisons de votre choix
- système distribué et tolérant aux pannes
- une année de données
- utiliser AWS pour déployer le cluster.
 - *Budget: 300E par groupe.

Les livrables attendus sont :

- code source (lien sur github...)
- presentation: architecture, modélisation, les avantages et inconvénients, des choix de modélisation et d'architecture, volumétrie, limites et contraintes

Pour la soutenance :

- Présentation de 10 minutes
- Démonstration de 10 minutes. Les données doivent être préalablement chargées dans le cluster. Les fonctionnalités doivent être démontrées. Il faut également démontrer la résilience du système (chaos monkey).
- Questions et réponses de 10 minutes

2. Données

Téléchargement des données depuis : <http://data.gdeltproject.org/>. Le stockage peut s'effectuer sur S3 ou en local.

Pour stocker les données sur S3 :

```
import com.amazonaws.services.s3.AmazonS3Client
import com.amazonaws.auth.BasicAWSCredentials
val AWS_ID = "AKIAJKA524RDCQGWZ5VQ"
val AWS_KEY = "kEE67F3n7s3R+A6nIdQbJeXJE9gmmCTCFKqQQ8Y7"
val awsClient = new AmazonS3Client(new BasicAWSCredentials(AWS_ID,
AWS_KEY))
sc.hadoopConfiguration.set("fs.s3a.access.key", AWS_ID) //(1) mettre
votre ID du fichier credentials.csv
sc.hadoopConfiguration.set("fs.s3a.secret.key", AWS_KEY) //(2) mettre
votre secret du fichier credentials.
sc.hadoopConfiguration.set("fs.s3a.connection.maximum","1000") //(3) 15
par default !!!
awsClient.putObject("john-doe-telecom-gdelt2018", "masterfilelist-
translation.txt",
new File( "/mnt/tmp/masterfilelist-translation.txt" ) )
```

Pour stocker les données de manière parallèle :

```
sqlContext.read.  
  option("delimiter", " ").  
  option("infer_schema", "true").  
  csv("/home/aar/bigdata/proj2018/data/masterfilelist-  
translation.txt").  
  withColumnRenamed("_c2", "url").  
  filter(col("url").contains("/201812")).  
  repartition(200).foreach( r=> {  
    val URL = r.getAs[String](0)  
    val fileName = r.getAs[String](0).split("/").last  
    val dir = "/home/aar/bigdata/proj2018/data/"  
    val localFileName = dir + fileName  
    fileDownloader(URL, localFileName)  
    val localFile = new File(localFileName)  
    /* AwsClient.s3.putObject("john-doe-telecom-gdelt2018", fileName,  
localFile )  
    localFile.delete() */  
  })
```

Pour charger les events :

```
val eventsRDD = sc.binaryFiles("../data/20181[0-9]*.export.CSV.zip",  
100).  
  flatMap { // decompresser les fichiers  
    case (name: String, content: PortableDataStream) =>  
      val zis = new ZipInputStream(content.open)  
      Stream.continually(zis.getNextEntry).  
        takeWhile(_ != null).  
        flatMap { _ =>  
          val br = new BufferedReader(new  
InputStreamReader(zis))  
            Stream.continually(br.readLine()).takeWhile(_ !=  
null)  
        }  
  }  
val cachedEvents = eventsRDD.cache // RDD
```

Utiliser les types stockés en BigQuery :

```
~/b/p/schema >>> google-cloud-sdk/bin/bq show --format=prettyjson gdelt-  
bq:gdeltv2.events {  
  "creationTime": "1463679409113",  
  "etag": "O5/wP5Yg/W1CN+rDMM85/w==",  
  "id": "gdelt-bq:gdeltv2.events",  
  "kind": "bigquery#table",  
  "lastModifiedTime": "1545240340088",  
  "location": "US",  
  "numBytes": "171542041673",  
  "numLongTermBytes": "0",  
  "numRows": "399903677",  
  "schema": {  
"fields": [ {  
  "description": "Globally unique identifier assigned to each event  
record that uniquely identifies  
  "mode": "NULLABLE",  
  "name": "GLOBALEVENTID",
```

3. Lancement d'un cluster

Création d'un bucket AWS S3

Nous allons sauvegarder une copie sur AWS du jeu de données utilisées. En utilisant un bucket AWS S3 dans ce but nous nous protégeons d'une éventuelle panne du site web de GDELT et nous allons avoir une source de données distribuée et répliquée avec une très grande tolérance aux pannes.

1) Allez sur la console AWS S3 (<https://s3.console.aws.amazon.com/s3/home?region=us-east-1#>)

2) Créez un bucket nom-prenom-telecom-gdelt2018 (! le nom du bucket doit être unique sur l'ensemble des utilisateurs S3). Assurez-vous que le bucket créé est dans la région US-East(N. Virginia). Gardez tous les autres paramètres de configuration du bucket à leur valeur par défaut.

Créer un compartiment

1 Nom et région 2 Configurer des options 3 Définir des autorisations 4 Vérification

Nom et région

Nom du compartiment ⓘ

fabien-mael-telecom-gdelt2018

Région

USA Est (Virginie du Nord) ▼

Copier les paramètres d'un compartiment existant

Sélectionner un compartiment (facultatif) 1 compartiments ▼

Créer Annuler Suivant

Compartiments S3

Découvrez la nouvelle console Astuces rapides

Rechercher compartiments Tous les types d'accès ▼

+ Créer un compartiment Modifier les paramètres d'accès public Vider Supprimer 2 Compartiments 2 Régions ↻

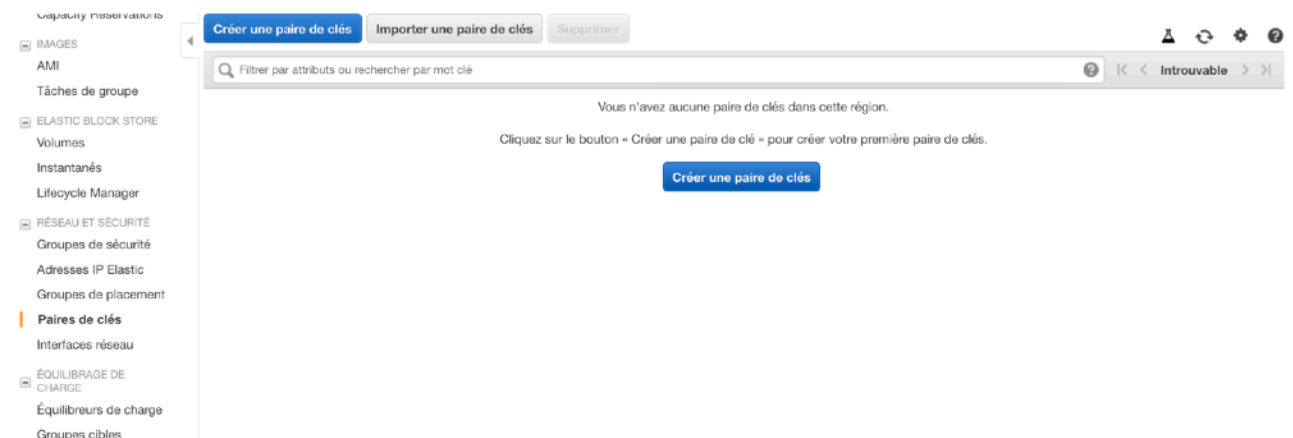
<input type="checkbox"/> Nom du compartiment ↑	Accès ⓘ ↑	Région ↑	Date de création ↑
<input type="checkbox"/> elasticbeanstalk-eu-west-3-239314173390	Les objets peuvent être publics	UE (Paris)	nov. 24, 2018 5:24:56 PM GMT+0100
<input type="checkbox"/> fabien-mael-telecom-gdelt2018	Compartiments et objets non publics	USA Est (Virginie du Nord)	déc. 13, 2018 9:20:54 AM GMT+0100

Démarrage d'un cluster sur AWS EMR

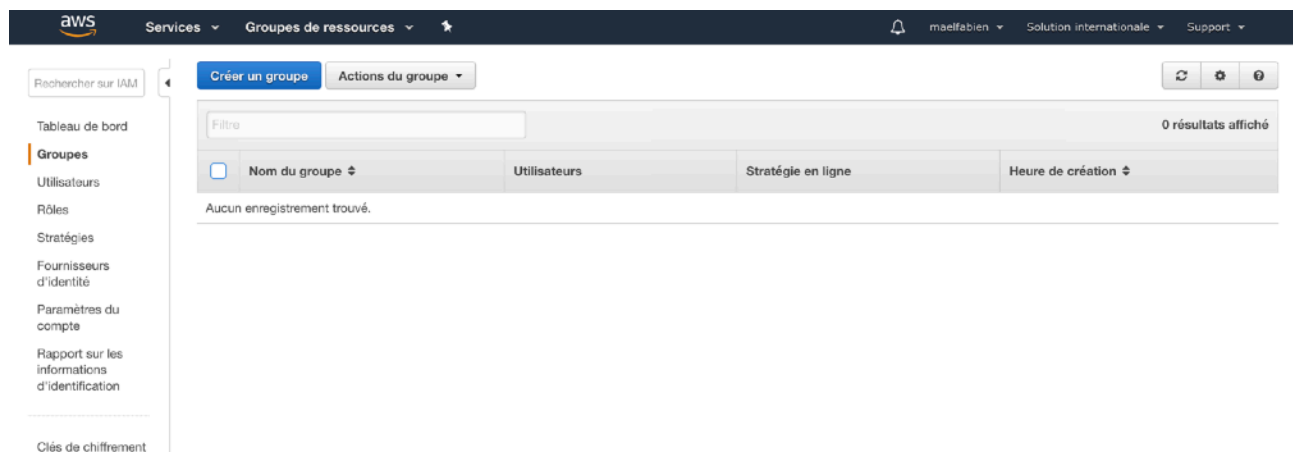
1) Rendez-vous sur la console AWS et créez une paire de clés gdeltKeyPair:

<https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#KeyPairs:sort=keyName>

La clé privée sera automatiquement sauvegardée après la création dans un fichier `gdelKeyPair.pem`. Notez l'emplacement de ce fichier, vous en aurez besoin pour plus tard















2) Allez sur <https://console.aws.amazon.com/iam/home?region=us-east-1#/groups> et créez un group admin avec une AdministratorAccess Policy



Attacher la stratégie

Sélectionnez une ou plusieurs stratégies à attacher. Jusqu'à 10 stratégies peuvent être attachées à chaque groupe.

Filtre : Type de stratégie <input type="text" value="Filtre"/>					404 résultats affichés
	Nom de la stratégie ↕	Entités attachées ↕	Heure de création ↕	Heure de modification ↕	
<input type="checkbox"/>	 AWSElasticBeanstalkEnhancedHe...	1	2016-02-09 00:17 UTC+0100	2018-04-10 00:12 UTC+0100	
<input type="checkbox"/>	 AWSElasticBeanstalkMulticontaine...	1	2016-02-09 00:15 UTC+0100	2016-06-07 01:45 UTC+0100	
<input type="checkbox"/>	 AWSElasticBeanstalkService	1	2016-04-11 22:27 UTC+0100	2018-04-19 18:34 UTC+0100	
<input type="checkbox"/>	 AWSElasticBeanstalkWebTier	1	2016-02-09 00:08 UTC+0100	2017-12-19 22:52 UTC+0100	
<input type="checkbox"/>	 AWSElasticBeanstalkWorkerTier	1	2016-02-09 00:12 UTC+0100	2016-12-21 03:01 UTC+0100	
<input type="checkbox"/>	 AWSLambdaBasicExecutionRole-...	1	2018-11-24 16:57 UTC+0100	2018-11-24 16:57 UTC+0100	
<input type="checkbox"/>	 AdministratorAccess	0	2015-02-06 19:39 UTC+0100	2015-02-06 19:39 UTC+0100	
<input type="checkbox"/>	 AlexaForBusinessDeviceSetup	0	2017-11-30 17:47 UTC+0100	2017-11-30 17:47 UTC+0100	
<input type="checkbox"/>	 AlexaForBusinessFullAccess	0	2017-11-30 17:47 UTC+0100	2018-06-26 01:53 UTC+0100	
<input type="checkbox"/>	 AlexaForBusinessGatewayExecution	0	2017-11-30 17:47 UTC+0100	2017-11-30 17:47 UTC+0100	
<input type="checkbox"/>	 AlexaForBusinessReadOnlyAccess	0	2017-11-30 17:47 UTC+0100	2018-06-26 01:52 UTC+0100	
<input type="checkbox"/>	 AmazonAPIGatewayAdministrator	0	2015-07-09 19:34 UTC+0100	2015-07-09 19:34 UTC+0100	

Annuler

Précédent

Étape suivante

Créer un groupe

Actions du groupe

Filtre

1 résultats affiché

	Nom du groupe ↕	Utilisateurs	Stratégie en ligne	Heure de création ↕
<input type="checkbox"/>	admin	0		2018-12-13 09:29 UTC+0100

3) Allez sur la console IAM et cliquez sur Add User. Mettez comme nom d'utilisateur: gdeltUser et cochez la case Programmatic access pour créer un identifiant d'accès et une clé de sécurité (access key ID and secret access key)

Définissez les informations utilisateur

Vous pouvez ajouter plusieurs utilisateurs en même temps avec les mêmes types et autorisations d'accès. [En savoir plus](#)

Nom d'utilisateur*

 [Ajoutez un autre utilisateur](#)


Sélectionnez un type d'accès AWS


Sélectionnez comment ces utilisateurs accèderont à AWS. Les clés d'accès et les mots de passe générés automatiquement sont fournis à la dernière étape. [En savoir plus](#)


- Type d'accès* ☒ **Accès par programmation**
Active une **ID de clé d'accès** et **clé d'accès secrète** pour AWS API, CLI, SDK et d'autres outils de développement.
- ☐ **Accès à AWS Management Console**
Active un **mot de passe** qui permet aux utilisateurs de vous connecter à l'AWS Management Console.

A l'étape suivante mettez votre utilisateur dans le groupe administrateur et validez la création de l'utilisateur.

▼ Définir des autorisations

 Ajouter un utilisateur au groupe

 Copier les autorisations à partir de l'utilisateur existant

 Attacher directement les stratégies existantes

Ajoutez un utilisateur à un groupe existant ou créez-en un nouveau. L'utilisation des groupes est une façon respectant les bonnes pratiques de gérer les autorisations de l'utilisateur en fonction de ses activités professionnelles. [En savoir plus](#)


Ajouter un utilisateur au groupe

Créer un groupe Actualiser

Affichage de : 1 résultat


Groupe ▼	Stratégies attachées
<input checked="" type="checkbox"/> admin	AdministratorAccess

Sur la page de confirmation de la création de votre utilisateur, cliquez sur *Download csv* pour sauvegarder dans un fichier *credentials.csv* l'ID et la clé de sécurité.

 **Opération réussie**
Vous avez créé les utilisateurs suivants avec succès. Vous pouvez consulter et télécharger les informations d'identification de sécurité utilisateur : Vous pouvez aussi envoyer par e-mail aux utilisateurs les instructions de connexion à AWS Management Console. C'est la dernière fois que ces informations d'identification sont disponibles pour le téléchargement. Cependant, vous pouvez créer des informations d'identification à tout moment.

Les utilisateurs ayant accès à AWS Management Console peuvent se connecter à :
<https://239314173390.signin.aws.amazon.com/console>

Téléchargez .csv

Utilisateur	ID de clé d'accès	Clé d'accès secrète
 gdeltUser	AKIAJIKEYNJBQ44IA	***** Afficher

Pour simplifier les procédures nous allons utiliser un utilisateur avec des droits d'admin. En général ce n'est pas recommandé, pour des raisons de sécurité d'utiliser des comptes avec trop de droits (si quelqu'un arrive à mettre la main sur votre identifiant d'accès et votre clé de sécurité il pourra démarrer des machines en votre nom). Nous vous conseillons de désactiver cet utilisateur à la fin du TP et créer un avec des droits plus spécifiques.

Démarrage d'un cluster de 3 noeuds via AWS EMR

Nous allons utiliser la console AWS EMR pour démarrer notre cluster:

- 1) Allez dans la console AWS EMR: <https://console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#> et cliquer sur Create cluster

Modifiez les paramètres suivants puis validez:

- GdeltCluster pour le nom du cluster
- Sélectionnez dans Applications -> Spark
- instance_type -> m3.large
- key pair -> gdeltKeyPair

Amazon EMR

Clusters

Configurations de la sécurité

Sous-réseaux VPC

Événements

Notebooks

Aide

What's new

Cloner

Résilier

Exporter AWS CLI

Cluster : GdeltCluster Démarrage en cours

Récapitulatif

Historique de l'application

Surveillance

Matériel

Événements

Étapes

Configurations

Actions d'amorçage

Connexions : --

DNS public principal : --

Balises : -- [Afficher tout/Modifier](#)

Récapitulatif

Détails de configuration

ID : j-2HG0FK10UB2T9

Date de création : 13-12-2018 10:06 (UTC+1)

Temps écoulé : 0 secondes

Résiliation Non automatique : --

Protection de la résiliation : Désactivé [Modification](#)

Étiquette de version : emr-5.19.0

Distribution Amazon Hadoop :

Applications : Ganglia 3.7.2, Spark 2.3.2, Zeppelin 0.8.0

URI de connexion : s3://aws-logs-239314173390-us-east-1/elasticmapreduce/

Vue Désactivé cohérente EMRFS :

ID d'AMI -- personnalisée :

Réseau et matériel

Sécurité et accès

Zone de disponibilité : --

ID de sous-réseau [subnet-b922b897](#)

(subnet) :

Maître : Mise en service 1 m3.xlarge

Nom de clé : gdeltKeyPair

Profil EMR_EC2_DefaultRole

d'instance EC2 :

Rôle EMR : EMR_DefaultRole

Visible pour tous les Tous [Modification](#)

2) Rajoutez une règle de firewall dans le security group du master pour permettre l'accès SSH:

Cluster : GdeltCluster Démarrage en cours Configuring cluster software

Récapitulatif

Historique de l'application

Surveillance

Matériel

Événements

Étapes

Configurations

Actions d'amorçage

Connexions : [Activer la connexion Web](#) – Zeppelin, Serveur d'historique Spark, Ganglia, Gestionnaire de ressources ... (Tout afficher)

DNS public principal : ec2-35-175-245-214.compute-1.amazonaws.com [SSH](#)

Balises : -- [Afficher tout/Modifier](#)

Récapitulatif

Détails de configuration

ID : j-2HG0FK10UB2T9

Date de création : 13-12-2018 10:06 (UTC+1)

Temps écoulé : 5 minutes

Résiliation Non automatique : --

Protection de la résiliation : Désactivé [Modification](#)

Étiquette de version : emr-5.19.0

Distribution Amazon Hadoop :

Applications : Ganglia 3.7.2, Spark 2.3.2, Zeppelin 0.8.0

URI de connexion : s3://aws-logs-239314173390-us-east-1/elasticmapreduce/

Vue Désactivé cohérente EMRFS :

ID d'AMI -- personnalisée :

Réseau et matériel

Sécurité et accès

Zone de disponibilité : us-east-1a

ID de sous-réseau [subnet-b922b897](#)

(subnet) :

Maître : En cours d'exécution 1 m3.xlarge

Principal : En cours d'exécution 2 m3.xlarge

Tâche : --

Nom de clé : gdeltKeyPair

Profil EMR_EC2_DefaultRole

d'instance EC2 :

Rôle EMR : EMR_DefaultRole

Visible pour tous les utilisateurs : Tous [Modification](#)

Groupes de sécurité [sg-09879a44210845501](#)

pour le principal : (ElasticMapReduce-master)

Créer un groupe de sécurité

Actions

search : sg-09879a44210845501

Ajouter filtre

Créer un groupe de sécurité Actions

search : sg-09879a4421084

<input type="checkbox"/>	Name	ID du groupe	Nom du groupe	ID de VPC	Description
<input type="checkbox"/>		sg-0853fb4eaa61e2535	ElasticMapReduce-slave	vpc-1d71da67	Slave group for Elastic MapReduce created on 2018-12-13T08:37:30...
<input checked="" type="checkbox"/>		sg-09879a44210845501	ElasticMapReduce-master	vpc-1d71da67	Master group for Elastic MapReduce created on 2018-12-13T08:37:30...

Supprimer le groupe de sécurité
 Ajouter/Modifier des balises
 Copier vers le nouveau
 Modifier les règles entrantes
 Modifier les règles sortantes

Modifier les règles entrantes

Règle TCP pe	TCP	8443	Personnali	72.21.198.64/29	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	54.240.217.16/29	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	54.239.98.0/24	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	207.171.167.101/32	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	207.171.167.26/32	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	72.21.217.0/24	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	54.240.217.80/29	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	54.240.217.64/28	Par exemple, SSH for Admin De	✕
Règle TCP pe	TCP	8443	Personnali	207.171.172.6/32	Par exemple, SSH for Admin De	✕
Tous les UDP	UDP	0 - 65535	Personnali	sg-0853fb4eaa61e2535	Par exemple, SSH for Admin De	✕
Tous les UDP	UDP	0 - 65535	Personnali	sg-09879a44210845501	Par exemple, SSH for Admin De	✕
Tous les ICMP	ICMP	0 - 65535	Personnali	sg-0853fb4eaa61e2535	Par exemple, SSH for Admin De	✕
Tous les ICMP	ICMP	0 - 65535	Personnali	sg-09879a44210845501	Par exemple, SSH for Admin De	✕

Ajouter une règle

REMARQUE : Les modifications apportées à des règles existantes se traduiront par la suppression de la règle modifiée et par la création d'une nouvelle règle avec les nouveaux détails. Le trafic lié à cette règle sera alors abandonné pendant un temps très limité jusqu'à ce que la nouvelle règle puisse être créée.

SSH TCP 22 N'importe 0.0.0.0/0, ::0 SSH from anywhere ✕

3) Pour accéder aux services qui tournent sur le master (Zeppelin -notebook spark, Ganglia - monitoring de ressources, Spark History Server / SparkUI etc) on doit passer par un tunnel SSH. Vous pouvez utiliser la direction des ports via un tunnel ssh comme montre en TP pour accéder aux services via <http://localhost:PORT>.

Cependant, on peut rediriger tous les ports à la fois via la meme commande SSH et on peut utiliser un proxy web (ex: FoxProxy) pour faire directement la translation master-public-dns-name:PORT ⇒ localhost:PORT. Pour cela cliquez sur Enable Web Connection et suivez la documentation fournie.

La liste complete des services/ports est la suivante:

Name of interface	URI
YARN ResourceManager	http:// <i>master-public-dns-name</i> :8088/
YARN NodeManager	http:// <i>coretask-public-dns-name</i> :8042/
Hadoop HDFS NameNode	http:// <i>master-public-dns-name</i> :50070/
Hadoop HDFS DataNode	http:// <i>coretask-public-dns-name</i> :50075/
Spark HistoryServer	http:// <i>master-public-dns-name</i> :18080/
Zeppelin	http:// <i>master-public-dns-name</i> :8890/
Hue	http:// <i>master-public-dns-name</i> :8888/
Ganglia	http:// <i>master-public-dns-name</i> /ganglia/
HBase UI	http:// <i>master-public-dns-name</i> :16010/

Récapitulatif

Historique de l'application

Surveillance

Matériel

Événements

Étapes

Configurations

Actions d'amorçage

Connexions : [Activer la connexion Web](#) – Zeppelin, Serveur d'historique Spark, Ganglia, Gestionnaire de ressources ... (Tout afficher)

DNS public principal : ec2-35-175-245-214.compute-1.amazonaws.com [SSH](#)

Balises : -- [Afficher tout/Modifier](#)

Utiliser le port 8891 coté local car 8890 est déjà utilisé par Jupyter :

```
MacBook-Pro-6324:AWS maelfabien$ ssh -L 8891:127.0.0.1:8890 -i
gdeltKeyPair.pem hadoop@ec2-35-175-245-214.compute-1.amazonaws.com
bind [127.0.0.1]:8890: Address already in use
channel_setup_fwd_listener_tcpip: cannot listen to port: 8890
Could not request local forwarding.
Last login: Thu Dec 13 09:48:13 2018
```

```

_ _ | ( _ _ /
_ | \ | |

```

Amazon Linux AMI

```
https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
20 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.
```

EEEEEEEEEEEEEEEEEEEE	MMMMMMM	MMMMMMM	RRRRRRRRRRRRRRR			
E::::::::::::::::::::E	M::::::M	M::::::M	R::::::::::::::::R			
EE:::::EEEEEEEEEE::E	M::::::M	M::::::M	R:::::RRRRRR:::::R			
E::::E	EEEE	M::::::M	M::::::M	RR::::R	R::::R	
E::::E		M::::::M	M::M	M::::M	R::R	R::::R
E:::::EEEEEEEEEE	M::::M	M::M	M::M	M::::M	R::RRRRRR:::::R	
E::::::::::::::::::::E	M::::M	M::M	M::M	M::::M	R::::::::::::RR	
E:::::EEEEEEEEEE	M::::M	M::::M	M::::M	R::RRRRRR:::::R		
E::::E	M::::M	M::M	M::::M	R::R	R::::R	
E::::E	EEEE	M::::M	MMM	M::::M	R::R	R::::R
EE:::::EEEEEEEEEE::E	M::::M		M::::M	R::R	R::::R	
E::::::::::::::::::::E	M::::M		M::::M	RR::::R	R::::R	
EEEEEEEEEEEEEEEEEEEE	MMMMMMM	MMMMMMM	RRRRRR	RRRRR		

Installation de AWS CLI

- 1) Pour certaines operation ca peut être pratique d'installer sur votre machine le client AWS([awscli](#)).

```
MacBook-Pro-6324:AWS maelfabien$ pip install awscli --upgrade
```

- 2) Une fois le client installe, utilisez la commande `aws configure` pour configurer votre installation (insérez votre *Access Key ID* et votre *Secret Access Key* (que vous avez sauvegardé dans `credentials.csv`), spécifiez la région par default à `us-east-1`, et le type de log par défaut à `text`:

Une fois la connexion SSH établie :

```
MacBook-Pro-6324:~ maelfabien$ aws configure
AWS Access Key ID [None]: AKIAJIKEYNJBQ44IA
AWS Secret Access Key [None]: IJ5ogQ1HRASverlSAKgs0gggwgev4I0t43gkjrrT
Default region name [None]: us-east-1
Default output format [None]: text
```

- 3) Pour verifier que la configuration est correcte on peut essayer d'afficher le contenu du bucket S3 précédemment crée :

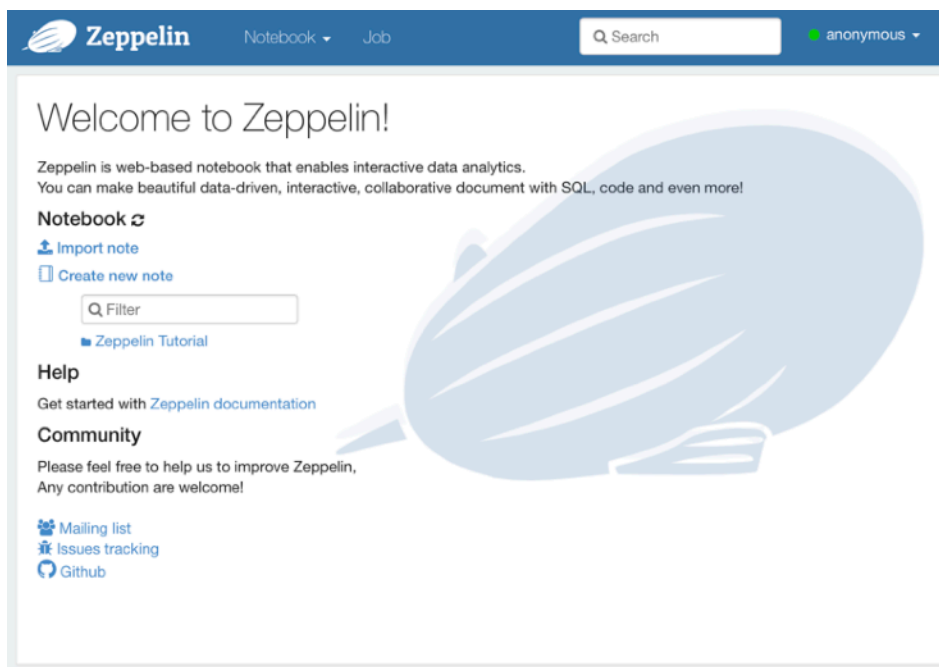
```
[hadoop@ip-172-31-86-166 ~]$ aws s3 ls --summarize --human-readable --recursive s3://fabien-mael-telecom-gdelt2018/
```

```
Total Objects: 0
Total Size: 0 Bytes
```

Connexion à l'interface du Zeppelin

Ouvrez un navigateur vers <http://master-public-dns-name:8890> (avec la configuration FoxProxy active!) et vous aurez accès à l'interface du Zeppelin :

<http://localhost:8891/#/>



Importer le notebooks suivants dans Zeppelin et suivez les instructions

- gdeltETL.json

- gdeltExploration.json

Notebook

 Import note

 Create new note

 Zeppelin Tutorial

 gdeltETL

 gdeltExploration

GDELT ETL

Dans ce notebook nous allons télécharger les fichiers GDELT pour la journée de 1er décembre 2018.

On commence par définir une fonction *fileDownloader* qui telecharge un fichier a partir d'un URL.

```
import sys.process._
import java.net.URL
import java.io.File
import java.io.File
import java.nio.file.{Files, StandardCopyOption}
import java.net.HttpURLConnection
import org.apache.spark.sql.functions._

def fileDownloader(urlOfFileToDownload: String, fileName: String) = {
  val url = new URL(urlOfFileToDownload)
  val connection = url.openConnection().asInstanceOf[HttpURLConnection]
  connection.setConnectTimeout(5000)
  connection.setReadTimeout(5000)
  connection.connect()

  if (connection.getResponseCode >= 400)
    println("error")
  else
    url #> new File(fileName) !!
}
```

On peut tester cette fonction pour télécharger en local le masterfilelist GDELT.

```
fileDownloader("http://data.gdeltproject.org/gdeltv2/masterfilelist.txt",
"/mnt/tmp/masterfilelist.txt") // save the list file to the Spark Master
```

Nous allons uploader le *masterfilelist.txt* dans notre bucket S3 via l'api scala AWS

```
import com.amazonaws.services.s3.AmazonS3Client
import com.amazonaws.auth.BasicAWSCredentials

val AWS_ID = "AKIAJIKEYNJBVTBQ44IA"
val AWS_KEY = "IJ5ogQlHRASverlSAKgs0gggwgev4I0t43gkjrrT"
val awsClient = new AmazonS3Client(new BasicAWSCredentials(AWS_ID,
AWS_KEY))

sc.hadoopConfiguration.set("fs.s3a.access.key", AWS_ID) // mettre votre
ID du fichier credentials.csv
```

```
sc.hadoopConfiguration.set("fs.s3a.secret.key", AWS_KEY) // mettre votre
secret du fichier credentials.csv
```

```
awsClient.putObject("fabien-mael-telecom-gdelt2018",
"masterfilelist.txt", new File( "/mnt/tmp/masterfilelist.txt") )
```

Verifions que le fichier a bien ete uploadé dans le bucket S3 via un dataframe Spark

```
import org.apache.spark.sql.SQLContext
```

```
val sqlContext = new SQLContext(sc)
val filesDF = sqlContext.read.
    option("delimiter"," ").
    option("infer_schema","true").
    csv("s3a://fabien-mael-telecom-gdelt2018/
masterfilelist.txt").
    withColumnRenamed("_c0","size").
    withColumnRenamed("_c1","hash").
    withColumnRenamed("_c2","url").
    cache
```

```
filesDF.show(false)
```

```
-----+-----+-----+
|size|hash|url|
-----+-----+-----+
|150383|297a16b493de7cf6ca809a7cc31d0b93|http://data.gdeltproject.org/gdeltv2/20150218230000.export.CSV.zip|
|318084|bb27f78ba45f69a17ea6ed7755e9f8ff|http://data.gdeltproject.org/gdeltv2/20150218230000.mentions.CSV.zip|
|10768507|ea8dde0beb0ba98810a92db068c0ce99|http://data.gdeltproject.org/gdeltv2/20150218230000.gkg.csv.zip|
|149211|2a91041d7e72b0fc6a629e2ff867b240|http://data.gdeltproject.org/gdeltv2/20150218231500.export.CSV.zip|
|339037|dec3f427076b716a8112b9086c342523|http://data.gdeltproject.org/gdeltv2/20150218231500.mentions.CSV.zip|
|10269336|2f1a504a3c4558694ade0442e9a5ae6f|http://data.gdeltproject.org/gdeltv2/20150218231500.gkg.csv.zip|
|149723|12268e821823aae2da90882621fedal8|http://data.gdeltproject.org/gdeltv2/20150218233000.export.CSV.zip|
|357229|744acad14559f2781a8db67715d63872|http://data.gdeltproject.org/gdeltv2/20150218233000.mentions.CSV.zip|
|11279827|66b03e2efd7d51dabf916b1666910053|http://data.gdeltproject.org/gdeltv2/20150218233000.gkg.csv.zip|
|158842|a5298ce3c6df1a8a759c61b5c0b6f8bb|http://data.gdeltproject.org/gdeltv2/20150218234500.export.CSV.zip|
|374528|dd322c888f28311aca2c735468405551|http://data.gdeltproject.org/gdeltv2/20150218234500.mentions.CSV.zip|
|11212939|cd20f295649b214ddl6666ca451b9994|http://data.gdeltproject.org/gdeltv2/20150218234500.gkg.csv.zip|
|362610|c4268d558bb22c02b3c132c17818c68b|http://data.gdeltproject.org/gdeltv2/20150219000000.export.CSV.zip|
|287807|e7f464a7a451ad2af6e9c8fa24f0ccea|http://data.gdeltproject.org/gdeltv2/20150219000000.mentions.CSV.zip|
|9728953|8f4b26e134bd6605cce2d32e92e5d3d7|http://data.gdeltproject.org/gdeltv2/20150219000000.gkg.csv.zip|
|251605|7685a6c71f010918f3be0d4ed2be977e|http://data.gdeltproject.org/gdeltv2/20150219001500.export.CSV.zip|
|263793|23ee65a60a1577dc74b979a54da406e|http://data.gdeltproject.org/gdeltv2/20150219001500.mentions.CSV.zip|
|9459370|6031464dfdc331551d491916d400c18|http://data.gdeltproject.org/gdeltv2/20150219001500.gkg.csv.zip|
|255259|f41066efb05d4024fca9dc1c2c6b9112|http://data.gdeltproject.org/gdeltv2/20150219003000.export.CSV.zip|
|308019|061133d1ef429c66c7ecba0d52063927|http://data.gdeltproject.org/gdeltv2/20150219003000.mentions.CSV.zip|
-----+-----+-----+
only showing top 20 rows
```

Par la suite on va charger uniquement les fichiers qui correspond a la journee du 1er decembre 2018

```
val sampleDF = filesDF.filter(col("url").contains("/20181201")).cache
```

```
sampleDF.show(false)
```

```
-----+-----+-----+
|size|hash|url|
-----+-----+-----+
|286749|7745b74ca805d90a86a19909a67e410a|http://data.gdeltproject.org/gdeltv2/20181201000000.export.CSV.zip|
|249784|95ea33b0393a85214eebfa58a02ceb38|http://data.gdeltproject.org/gdeltv2/20181201000000.mentions.CSV.zip|
|9694304|c0a3d85a1f3a5263b4c9f7eela632f7|http://data.gdeltproject.org/gdeltv2/20181201000000.gkg.csv.zip|
|250319|47317d0bd9cd56e7b96732677a918435|http://data.gdeltproject.org/gdeltv2/20181201001500.export.CSV.zip|
|304863|37513cccabac99a9e50a36dc8a590cb6|http://data.gdeltproject.org/gdeltv2/20181201001500.mentions.CSV.zip|
|10119350|50c3fa1e60385a2ca1314af48dfbad4f|http://data.gdeltproject.org/gdeltv2/20181201001500.gkg.csv.zip|
|224825|8ca8b3a6b4ff6a960b8db5ac8eb9174a|http://data.gdeltproject.org/gdeltv2/20181201003000.export.CSV.zip|
|298101|68369aec62706a2b5a3752d2d725caa9|http://data.gdeltproject.org/gdeltv2/20181201003000.mentions.CSV.zip|
|9705075|8cd7aa2af725aa02b42412e0944cfe26|http://data.gdeltproject.org/gdeltv2/20181201003000.gkg.csv.zip|
|191077|906e6599092855ba2965fe59fce70062|http://data.gdeltproject.org/gdeltv2/20181201004500.export.CSV.zip|
|252312|5e9048e047049b3b6ac86c0029cae9c|http://data.gdeltproject.org/gdeltv2/20181201004500.mentions.CSV.zip|
|9370422|da674db2a8d110d27753b97617431688|http://data.gdeltproject.org/gdeltv2/20181201004500.gkg.csv.zip|
|187888|cee25cd1ab712b7ac4e2538c48353f21|http://data.gdeltproject.org/gdeltv2/20181201010000.export.CSV.zip|
|274338|1e2ffac3b7d73d9c9a93032a50f41|http://data.gdeltproject.org/gdeltv2/20181201010000.mentions.CSV.zip|
|10008236|48764cfe5bc415441ed34b4b5a1b0372|http://data.gdeltproject.org/gdeltv2/20181201010000.gkg.csv.zip|
|145719|df1010eb1a3466fef5c9a5b89368328c|http://data.gdeltproject.org/gdeltv2/20181201011500.export.CSV.zip|
|255772|781ffbf723e8b5595f0c8517ccc6c91b|http://data.gdeltproject.org/gdeltv2/20181201011500.mentions.CSV.zip|
|8933180|e1fal1f13da14b85e2ba31c35cbbac8b1|http://data.gdeltproject.org/gdeltv2/20181201011500.gkg.csv.zip|
|133461|1bcaee75fe0a316dac205752030a05e5|http://data.gdeltproject.org/gdeltv2/20181201013000.export.CSV.zip|
|246396|2c9bac79d58fc5a141651ddc072d0cd6|http://data.gdeltproject.org/gdeltv2/20181201013000.mentions.CSV.zip|
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
```

Nous allons charger tous ces fichiers dans le bucket S3 via un ETL Spark:

```
object AwsClient{
    val s3 = new AmazonS3Client(new BasicAWSCredentials(AWS_ID, AWS_KEY))
}

sampleDF.select("url").repartition(100).foreach( r=> {
    val URL = r.getAs[String](0)
    val fileName = r.getAs[String](0).split("/").last
    val dir = "/mnt/tmp/"
    val localFileName = dir + fileName
    fileDownloader(URL, localFileName)
    val localFile = new File(localFileName)
    AwsClient.s3.putObject("fabien-mael-telecom-gdelt2018",
fileName, localFile )
    localFile.delete()

})
```

Vérifiez via awscli sur votre machine que vous avez bien chargé les données sur 1er décembre 2018 :

```
MacBook-Pro-6324:~ maelfabien$ aws s3 ls --summarize --human-readable --
recursive s3://fabien-mael-telecom-gdelt2018/
2018-12-13 12:02:20   280.0 KiB 20181201000000.export.CSV.zip
2018-12-13 12:02:10    9.2 MiB 20181201000000.gkg.csv.zip
2018-12-13 12:03:03   243.9 KiB 20181201000000.mentions.CSV.zip
2018-12-13 12:02:46   244.5 KiB 20181201001500.export.CSV.zip
2018-12-13 12:02:37    9.7 MiB 20181201001500.gkg.csv.zip
2018-12-13 12:02:12   297.7 KiB 20181201001500.mentions.CSV.zip
2018-12-13 12:02:06   219.6 KiB 20181201003000.export.CSV.zip
2018-12-13 12:02:24    9.3 MiB 20181201003000.gkg.csv.zip
2018-12-13 12:02:13   291.1 KiB 20181201003000.mentions.CSV.zip
2018-12-13 12:02:39   186.6 KiB 20181201004500.export.CSV.zip
2018-12-13 12:02:48    8.9 MiB 20181201004500.gkg.csv.zip
2018-12-13 12:01:59   246.4 KiB 20181201004500.mentions.CSV.zip
2018-12-13 12:02:05   183.5 KiB 20181201010000.export.CSV.zip
2018-12-13 12:02:02    9.5 MiB 20181201010000.gkg.csv.zip
2018-12-13 12:02:13   267.9 KiB 20181201010000.mentions.CSV.zip
2018-12-13 12:02:29   142.3 KiB 20181201011500.export.CSV.zip
2018-12-13 12:02:19    8.5 MiB 20181201011500.gkg.csv.zip
2018-12-13 12:02:40   249.8 KiB 20181201011500.mentions.CSV.zip
2018-12-13 12:02:37   130.3 KiB 20181201013000.export.CSV.zip
2018-12-13 12:02:47    8.2 MiB 20181201013000.gkg.csv.zip
2018-12-13 12:02:30   240.6 KiB 20181201013000.mentions.CSV.zip
```

GDELT ETL Exploration

Dans ce notebook nous allons commencer à explorer les données GDELT qu'on a stocké sur S3.

```
sc.hadoopConfiguration.set("fs.s3a.access.key",
"AKIAJIKEYNJBQ44IA") // mettre votre ID du fichier credentials.csv
sc.hadoopConfiguration.set("fs.s3a.secret.key",
"IJ5ogQ1HRASverlSAKgs0gggwgev4I0t43gkjrrT") // mettre votre secret du
fichier credentials.csv
```

Les fichiers sont stockés et compressés, on a besoin d'un bout de code pour les décompresser en parallèle sur les workers au fur et à mesure qu'on les lit depuis S3:

```

import org.apache.spark.input.PortableDataStream
import java.util.zip.ZipInputStream
import java.io.BufferedReader
import java.io.InputStreamReader
// 20181201000000.export.CSV.zip

val textRDD = sc.binaryFiles("s3a://fabien-mael-telecom-
gdelt2018/20181201[0-9]*.export.CSV.zip"). // charger quelques fichiers
via une regex
  flatMap { // decompresser les fichiers
    case (name: String, content: PortableDataStream) =>
      val zis = new ZipInputStream(content.open)
      Stream.continually(zis.getNextEntry).
        takeWhile(_ != null).
        flatMap { _ =>
          val br = new BufferedReader(new
InputStreamReader(zis))
            Stream.continually(br.readLine()).takeWhile(_ !=
null)
        }
  }
textRDD.take(1)

res21: Array[String] = Array(806754250      20171201      201712  2017    2017.9068
                                USA      UNITED STATES  USA
                                1      100      100      10      3      -5.0      4      1      4
-0.6666666666666667      0
Connecticut, United States  US      USCT      41.3082 -72.9282      209231  3      New Haven,
Connecticut, United States  US      USCT      41.3082 -72.9282      209231  20181201000000
https://bismarcktribune.com/news/national/the-latest-immigrant-s-supporters-end-courthouse-protest/
article_76d71ef6-d0e0-5e02-ab72-4c8a0d31778e.html)

```


4. Exploration des données

Charger les données vers SparkSQL

Désormais, il faut ajouter un label aux données :

Case class event : (JSON Type)

```
~/b/p/schema >>> google-cloud-sdk/bin/bq show --format=prettyjson gdelt-  
bq:gdeltv2.events |  
jq '["schema"]' events.json |jq '[]|.[]|.name +": " + .type + ","|.|.  
[]' |sed "s/INTEGER/Int/" | sed "s/FLOAT/Double/" | sed "s/STRING/String/"  
|sed "s/\"/\\/g"
```

```
case class Event(  
  GLOBALEVENTID: Int,  
  SQLDATE: Int,  
  MonthYear: Int,  
  Year: Int,  
  FractionDate: Double,  
  Actor1Code: String,  
  Actor1Name: String,  
  Actor1CountryCode: String,  
  Actor1KnownGroupCode: String,  
  Actor1EthnicCode: String,  
  Actor1Religion1Code: String,  
  Actor1Religion2Code: String,  
  Actor1Type1Code: String,  
  Actor1Type2Code: String,  
  Actor1Type3Code: String,  
  Actor2Code: String,  
  Actor2Name: String,  
  Actor2CountryCode: String,  
  Actor2KnownGroupCode: String,  
  Actor2EthnicCode: String,  
  Actor2Religion1Code: String,  
  Actor2Religion2Code: String,  
  Actor2Type1Code: String,  
  Actor2Type2Code: String,  
  Actor2Type3Code: String,  
  IsRootEvent: Int,  
  EventCode: String,  
  EventBaseCode: String,  
  EventRootCode: String,  
  QuadClass: Int,  
  GoldsteinScale: Double,  
  NumMentions: Int,  
  NumSources: Int,  
  NumArticles: Int,  
  AvgTone: Double,  
  Actor1Geo_Type: Int,  
  Actor1Geo_FullName: String,  
  Actor1Geo_CountryCode: String,  
  Actor1Geo_ADM1Code: String,  
  Actor1Geo_ADM2Code: String,  
  Actor1Geo_Lat: Double,  
  Actor1Geo_Long: Double,  
  Actor1Geo_FeatureID: String,  
  Actor2Geo_Type: Int,  
  Actor2Geo_FullName: String,
```

```

Actor2Geo_CountryCode: String,
Actor2Geo_ADM1Code: String,
Actor2Geo_ADM2Code: String,
Actor2Geo_Lat: Double,
Actor2Geo_Long: Double,
Actor2Geo_FeatureID: String,
ActionGeo_Type: Int,
ActionGeo_FullName: String,
ActionGeo_CountryCode: String,
ActionGeo_ADM1Code: String,
ActionGeo_ADM2Code: String,
ActionGeo_Lat: Double,
ActionGeo_Long: Double,
ActionGeo_FeatureID: String,
DATEADDED: BigInt,
SOURCEURL: String
)

```

Puis il faut convertir le RDD en DataSet pour pouvoir l'exploiter dans SparkSQL :

RDD -> DataSet -> SparkSQL

```

def toDouble(s : String): Double = if (s.isEmpty) 0 else s.toDouble
def toInt(s : String): Int = if (s.isEmpty) 0 else s.toInt
def toBigInt(s : String): BigInt = if (s.isEmpty) BigInt(0) else
  BigInt(s)

cachedEvents.map(_._split("\t")).filter(_._length==61).map(
  e=> Event(

    toInt(e(0)),toInt(e(1)),toInt(e(2)),toInt(e(3)),toDouble(e(4)),e(5),e(6),
    e(7),e(8),e(9),e(10),e(11),e(12),e(13),e(14),e(15),e(16),e(17),e(18),e(19)
    ,e(20),

    e(21),e(22),e(23),e(24),toInt(e(25)),e(26),e(27),e(28),toInt(e(29)),toDou
    ble(e(30)),toInt(e(31)),toInt(e(32)),toInt(e(33)),toDouble(e(34)),toInt(e
    (35)),e(36),e(37),e(38),e(39),toDouble(e(40)),

    toDouble(e(41)),e(42),toInt(e(43)),e(44),e(45),e(46),e(47),toDouble(e(48)
    ),toDouble(e(49)),e(50),toInt(e(51)),e(52),e(53),e(54),e(55),toDouble(e(5
    6)),toDouble(e(57)),e(58),toBigInt(e(59)),e(60))

  ).toDS.createOrReplaceTempView("export")
  spark.catalog.cacheTable("export")

```

On peut désormais tester que notre code fonctionne :

```

spark.sql(""" SELECT * FROM export LIMIT 10""").show

z.show(spark.sql(""" SELECT * FROM export """))

```

Explorer les données

Attention à corriger les types de données, à corriger les valeurs (lignes incomplètes...).

```

z.show(spark.sql(""" SELECT SQLDATE, count(*) as nbEvents FROM
export_translation GROUP BY SQLDATE """))

```

```

fileDownloader("http://data.gdeltproject.org/gdeltv2/masterfilelist-translation.txt", "/mnt/tmp/masterfilelist_translation.txt")

awsClient.putObject("fabien-mael-telecom-gdelt2018",
"masterfilelist_translation.txt", new File( "/mnt/tmp/
masterfilelist_translation.txt" ) )

val filesDF_translation = sqlContext.read.
    option("delimiter"," ").
    option("infer_schema","true").
    csv("s3a://fabien-mael-telecom-gdelt2018/
masterfilelist_translation.txt").
    withColumnRenamed("_c0","size").
    withColumnRenamed("_c1","hash").
    withColumnRenamed("_c2","url").
    cache

val sampleDF_translation =
filesDF_translation.filter(col("url").contains("/20181201")).cache

object AwsClient{
    val s3 = new AmazonS3Client(new BasicAWSCredentials(AWS_ID, AWS_KEY))
}

sampleDF_translation.select("url").repartition(100).foreach( r=> {
    val URL = r.getAs[String](0)
    val fileName = r.getAs[String](0).split("/").last
    val dir = "/mnt/tmp/"
    val localFileName = dir + fileName
    fileDownloader(URL, localFileName)
    val localFile = new File(localFileName)
    AwsClient.s3.putObject("fabien-mael-telecom-gdelt2018",
fileName, localFile )
    localFile.delete()

})

val textRDD_translation = sc.binaryFiles("s3a://fabien-mael-telecom-
gdelt2018/20181201[0-9]*.export_translation.CSV.zip"). // charger
quelques fichiers via une regex
    flatMap { // decompresser les fichiers
        case (name: String, content: PortableDataStream) =>
            val zis = new ZipInputStream(content.open)
            Stream.continually(zis.getNextEntry).
                takeWhile(_ != null).
                flatMap { _ =>
                    val br = new BufferedReader(new
InputStreamReader(zis))
                    Stream.continually(br.readLine()).takeWhile(_ !=
null)
                }
    }

cachedEvents.map(_ .split("\t")).filter(_ .length==61).map(
    e=> Event(

toInt(e(0)),toInt(e(1)),toInt(e(2)),toInt(e(3)),toDouble(e(4)),e(5),e(6),

```

```
e(7),e(8),e(9),e(10),e(11),e(12),e(13),e(14),e(15),e(16),e(17),e(18),e(19),e(20),
```

```
e(21),e(22),e(23),e(24),toInt(e(25)),e(26),e(27),e(28),toInt(e(29)),toDouble(e(30)),toInt(e(31)),toInt(e(32)),toInt(e(33)),toDouble(e(34)),toInt(e(35)),e(36),e(37),e(38),e(39),toDouble(e(40)),
```

```
toDouble(e(41)),e(42),toInt(e(43)),e(44),e(45),e(46),e(47),toDouble(e(48)),toDouble(e(49)),e(50),toInt(e(51)),e(52),e(53),e(54),e(55),toDouble(e(56)),toDouble(e(57)),e(58),toBigInt(e(59)),e(60))
```

```
).toDS.createOrReplaceTempView("export_translation")  
spark.catalog.cacheTable(« export_translation")
```

```
z.show(spark.sql(""" SELECT SQLDATE, count(*) as nbEvents FROM  
export_translation GROUP BY SQLDATE """))
```

```
z.show(spark.sql(""" SELECT SQLDATE, Actor1Geo_CountryCode, count(*) n  
FROM export GROUP BY Actor1Geo_CountryCode, SQLDATE HAVING n>1000 ORDERED  
BY SQLDATE DESC """))
```

```
z.show(spark.sql(""" SELECT SQLDATE, Actor1Geo_CountryCode, count(*) n  
FROM export_translation GROUP BY Actor1Geo_CountryCode, SQLDATE HAVING  
n>1000 ORDERED BY SQLDATE DESC """))
```