Installation from scratch

0 - Création d'un compte racine AWS avec email + Carte de crédit

1 - Création d'un security group

- Sélectionner la région dans laquelle on souhaite utiliser les serveurs EC2. Dans le cadre de notre projet, les données sont déjà présentes sur un serveur d'Amazon de la région US EAST (N. Virginia). Décider d'utiliser des serveurs de la même région, c'est aussi s'exonérer de frais de transferts.
- Se rendre sur la page [2]
- Sélectionner **Create Security group**:
    - *Security group name*: Nom au choix.
        - Ex: **ProjetNoSQLSecurityGroup**
    - *Description*: Il est conseillé d'indiquer le nom de la région dans laquelle les serveurs sont déployés. Il faut que le security group soit défini dans la même zone afin d'autoriser les machines à opérer.
        - Ex: **Region des instances_US East (N. Virginia)**
    - *VPC*: Laisser la valeur par défaut
- Créer une première règle dans les connexions entrantes des serveurs(**Inbound)**. Cliquer sur **Add rule.** Cette règle autorise n'importe qui à créer un pont SSH sur le port 22 via le protocol TCP.
    - *Type*: **SSH**
    - *Protocol*: **TCP**
    - *Port Range*: **22**
    - *Source*: **Anywhere**
- Valider
- Le nouveau groupe de sécurité est créé. Copier le **Group ID** l'identifiant
- Sélectionner le groupe de sécurité. Cliquer sur Actions -> **Edit Inbound Rules**
- Ajouter les ports comme défini sur l'image en bas de la page [1] (ou ci-dessous). Indiquer le Group ID pour les connexions à n'autoriser qu'aux membres du groupe de sécurité .

2 - Création d'un compte Datastax

- Créer un compte sur la page d'enregistrement de Datastax [3]
    - Afin d'éviter les potentiels problèmes d'authentification ensuite sur les serveurs EC2, éviter de choisir un mot de passe avec des caractères spéciaux. L'expérience du '@'
    - Une fois authentifié, aller noter le nom d'utilisateur de Datastax dans **My account > Edit** (Ex: ursula.leguin_2187)

3 - Configurer 6 instances EC2

- Création d'une paire de **clefs privée/publique** pour s'authentifier sur les serveurs
    - Aller dans la console Amazon > **EC2**
    - Dans **Ressources**, cliquer sur **Key Pairs**
    - **Create Key Pair**. Choisir un nom. Ex: ProjetNoSQLKey
    - **Télécharger la clef privée** ProjetNoSQLKey.pem
    - Donner des droits de lecture au owner sur la clef:
        - chmod 400 ProjetNoSQLKey.pem
- Lancer les instances EC2
    - Aller dans Amazon > EC2 > Create Instance > **Launch Instance**
    - Sélectionner **Community AMIs** (Amazon Machine Image)
    - Taper datastax dans la barre de recherche
    - Step 1. Sélectionner **DataStax Auto-Clustering AMI 2.6.3-1404-hvm** - ami-711ca91a
    - Step 2. Choisir des instances **m3.medium.** Next
    - Step 3. Choisir le nombre d'instances: **6 .**
    - Step 3. Dans **Advanced details** écrire:

```
--clustername wikitrends
--totalnodes 6
--version enterprise
--username <YourDataStaxUsername>
--password <YourDataStaxPassword>
--analyticsnodes 6
--cfsreplicationfactor 2
--opscenter Yes
```

    - Step 6. Sélectionner le **groupe de sécurité** défini auparavant. Review and Launch
    - Choisir la **clef publique** à charger sur les instances
    - **Launch instances**.
    - **View instances.**
- Pour arrêter les instances et ainsi ne plus être facturé, il faut sélectionner toutes les instances, puis cliquer sur **Actions > Instante State > Terminate**

4 - Connexion au master

- Identifier le premier serveur lancé.
    - Cliquer successivement sur les instances et chercher celle qui a l' **AMI launch index** égale à **0**.
    - Noter la **Public DNS** de ce serveur.
- Ouvrir un tunnel SSH avec le master
    - Ouvrir un terminal taper la commande:

      ssh -i ProjetNoSQLKey.pem ubuntu@ec2-54-152-98-239.compute-1.amazonaws.com
        - Attention à bien définir le chemin vers la clef et avoir le port 22 ouvert sur le réseau.
    - L'installation de Cassandra devrait se lancer
    - Pour accéder aux logs:
        - AMI log: ~/datastax_ami/ami.log
        - Cassandra log: /var/log/cassandra/system.log
    - Taper **nodetool status**, les 6 noeuds devraient être listés en UN (UP + Normal)

5 - Séquence du script shell d'installation de l'environnement

- Installation de Cassandra-Spark contenue dans l'AMI Datastax chargée se lance automatiquement à l'établissement de la connexion SSH avec le master(Voir section 4). Exemple:

```
ssh -i ProjetNoSQLKey.pem ubuntu@ec2-54-152-98-239.compute-1.amazonaws.com
```

- Installation d'Anaconda afin d'avoir IPython et les librairies qui vont bien

```
cd ~
wget
https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda2-2.4.1-Linux-x86_64.sh
bash Anaconda2-2.4.1-Linux-x86_64.sh -b
cd ~
echo "export PATH=/home/ubuntu/anaconda2/bin:$PATH" >> .bashrc
export PATH=/home/ubuntu/anaconda2/bin:$PATH
```

- Installation de Jupyter notebook sur le cluster
    - Tutorial: http://blog.impiyush.me/2015/02/running-ipython-notebook-server-on-aws.html
    - Attention à spécifier le port 8889 par défaut dans la configuration de jupyter. Le 8888 est déjà pris par le OpsCenter.
    - Lancer le IPython dans un contexte Spark:

```
export PYSPARK_DRIVER_PYTHON=ipython PYSPARK_DRIVER_PYTHON_OPTS="notebook --no-browser"
dse pyspark
```

    - Accès: https://ec2-54-175-192-111.compute-1.amazonaws.com:8889
    - Ne pas oublier d'ouvrir le port 8889 sur les instances EC2.
- Accès interfaces
    - Spark : port 7080 / 7081 (pour les deux premières instances)
    - DevCenter: port 9042
    - OpsCenter: http://Public_DNS_du_master:8888

6 - Chargement des données

- Création des tables via DevCenter
  - Création d'une connexion
    - Contact hosts:
    - Native Protocol Port: 9042 (le port doit être ouvert)
  - Création d'un Keyspace **wikistats**

This simple form allows you to try out different values for your Apache Cassandra cluster and see what the impact is for your application **http://www.ecyrd.com/cassandracalculator/**

(pour le moment, nous sommes conscients que la replication va nous augmenter de ressource, et le budget …)

Typically, a cluster has one keyspace per application. Keyspaces are designed to control data replication for a set of tables.

CREATE KEYSPACE **wikistats**

... WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }

- Requêtes de création des tables / essai1:

CREATE TABLE wikistats.hrpageviews2 (year int, month int, day int, hour int, title text, pageviews int, PRIMARY KEY ((year, month, day, hour),title));

CREATE TABLE wikistats.dapageviews2 (year int, month int, day int, title text, agg_pageviews int, PRIMARY KEY ((year, month, day), title)); WITH CLUSTERING ORDER BY (title DESC));

La clef de partition choisie correspond à la date. La clef de clustering au titre.
On choisit de créer une table par type de requête. Cela permet d'accélerer le temps de réponse de la base de données. Il y a donc une table (hrpageviews) qui est destinée à déterminer les tendances depuis 24h (l'heure est l'unité de temps minimum requise et fait donc partie de la clef de partition). Une autre table (dapageviews2) va permettre de déterminer les tendances sur les 30 derniers jours (le jour est l'unité de temps minimum requise). Dans ce cas, on ne stocke pas l'heure > Gain de stockage + temps de réponses des requêtes.

- Requêtes de création des tables / essai2:

CREATE TABLE wikistats.hrpageviews3 (event_time timestamp, title text, pageviews int, PRIMARY KEY ((event_time), title);

CREATE TABLE wikistats.daypageviews3 (event_time timestamp, title text, pageviews int, PRIMARY KEY ((event_time), title);

- Requêtes de création des tables / quel choix?

Dans l'optique on l'on veut utiliser spark in fine (et donc que l'architecture des données cassandra sera modifiée pour former des RDD), il y a certainement une architecture optimale pour la création des RDD.

En effet, Primary_Key contient 2 parties, typiquement Primary_key((Xi), Yi), où les Xi sont les clés de partition qui doivent être appelées dans select avec un '=' strict. Les clé de clustering Yi peuvent elles

être appelées dans select avec un plus grand nombre d'opérateur, typiquement < et > permettant de définir un intervalle de temps.

Une manière de faire serait de creer une table avec la clé PRIMARY KEY ((event_time), title) et de définir une liste contenant une gamme de timestamp et d'effectuer des requêtes sur chaque timestamp de la liste

Une autre manière de faire serait de creer une table avec la clé PRIMARY KEY ((title), event_time). Ce choix à été tout d'abord écarté car certaines pages avait un même titre (avec langue différente), ce qui risquerait d'écraser des données. Toutefois seules les pages 'en' sont maintenant chargée dans cassandra ce qui rend la configuration des données présentée ci-dessous pertinente

CREATE TABLE wikistats.daypageviews3 (event_time timestamp, title text, pageviews int, PRIMARY KEY (title, event_time);

On choisit également de ne plus trier les données selon la date ou le titre (abandon de la clause WITH CLUSTERING → voir essai1 table wikistats.dapageviews2), car ce n'est pas nécéssaire pour répondre au cahier des charges (ranking par rapport aux pageviews)

- Création d'un bucket avec les données wikistats. D'autres utilisateurs autorisés peuvent y accéder à distance. Avant de reproduire la séquence suivante, on utilise les comptes pour se connecter au bucket d'Aurélien. //TODO: A REPRODUIRE -->
    - Création d'un volume aws avec le snapshot aws wikistats
    - tu attaches ce serveur à une instance aws ec2
    - tu montes le volume sur cette instance
    - sudo mount /dev/xvdf1 /wiki/ par exemple
    - tu synchronises le répertoire /wiki avec ton bucket
    - aws s3 sync /wiki/ s3://..../
- Traitement des fichiers et insertion dans Cassandra

  Lancement de la console iPython :

  ouvrir le port pour le park context (voir figure du step 1). ce port est lisible en lançant la commande dse spark (qui lance spark shell scala). on obtient un message 'Initializing SparkContext with MASTER: spark://172.31.29.1:7077' et donc il faut ouvrir dans ce cas le port 7077 pour pouvoir importer le spark context. il y a certainement un autre moyen de connaitre ce port que nous n'avons pas identifié

  PYSPARK_DRIVER_PYTHON=ipython dse pyspark

  Dans la console iPython pour copier du code sur plusieurs lignes

  %cpaste

  pour cloturer la saisie: --

    - Voir fichier python

  on peut visualiser le chargement des données en entrant dans un navigateur le public DNS (voir step 4) suivi du numéro de port adéquat (8080):

7 - Exemple de requêtage avec un SqlContext. Puis c'est parti avec Python (%cpaste)

- Pour accéder au cluster, les utilisateurs doivent sauvegarder la clé privée ProjetNoSQLKey.pem en local (ne pas oublier de taper la commande 'chmod 400 ProjetNoSQLKey.pem')
  Se connecter en ssh (il faut pour cela faire une manip pour autoriser la connection)
  ssh -i jade_projet_cassandra.pem aurelien@ec2-54-175-192-111.compute-1.amazonaws.com
- Mettre en place l'environnement bash avec la commande 'bash'. Cela permet notamment au système de connaitre les variables d'environnement (anaconda entres autres)
- Intégrer un ipython avec la commande 'PYSPARK_DRIVER_PYTHON=ipython dse pyspark'

- On peut aussi intégre un shell scala avec la commande ‘dse spark’
- Dans ipython entrer mot magique ‘%paste’ pour pouvoir concatener plusieurs lignes à la suite
- Exemple de requêtes d’intérêt pour le projet:

  sqlContext.sql('select * from wikistats.hrpageviews where pageviews >1000').take(10)

  df.sort(df.pageviews.desc()).take(10)

  Avec les tests + explication (de Cassandra schema) , la requête suivante risque ne pas pouvoir marcher car ‘event-time’ fait partie de partition key (cannot use other operations other than = or in)

  df = sqlContext.sql('SELECT * FROM wikistats.hrpageviews2 WHERE event_time > 2013-01-01 00:05+0000 AND event_time < 2013-02-02 10:00+0000')

  On propose donc (si possible) de rétingrer le champs ‘payscode’ :

  df=sqlContext.sql('SELECT * FROM wikistats.hrpageviews2 WHERE paycode=’en’ and event_time > 2013-01-01 00:05+0000 AND event_time < ‘2013-02-02 10:00+0000') #certains groupe ont filtrer des payscodes <> ‘en’

8 - Creation de comptes pour d’autres utilisateurs

- Création de comptes pour accéder en lecture à la configuration du compte de l’initiateur des instances. Ne permet pas d’accéder au cluster. Voir [4]
- Création de comptes (avec clefs privées) pour permettre à des utilisateurs de se connecter au cluster (spark-cassandra-jupyter) [5]
  - Comme dans la section 3. créer une paire de clefs privée/publique (**Key Pairs)** pour l’utilisateur.
  - Se connecter en ssh sur l’instance master, puis éxécuter les commandes suivantes:

```
sudo adduser newuser
sudo su - newuser
mkdir .ssh
chmod 700 .ssh
touch .ssh/authorized_keys
chmod 600 .ssh/authorized_keys
nano .ssh/authorized_keys
```

  - Ecrire dans authorized_keys la **clef publique** générée. Pour l’obtenir, ouvrir un terminal et taper:

```
echo newuser.pem | sudo ssh-keygen -y
```

  - Copier le résultat et sauvegarder. Transmettre la clef privée newuser.pem de manière sécurisée à l’utilisateur.

Ressources:

[1]http://docs.datastax.com/en/datastax_enterprise/4.8/datastax_enterprise/install/installAMIsecurity.html
(Attention, pour les versions plus récentes de datastax, il faur modifier l'url 4.8 → 4.9 par exemple)

[2]https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroups:sort=groupId

[3] https://academy.datastax.com/user/register

[4] http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_how-users-sign-in.html

[5] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-users.html