**Conception d’un Data Lake pour Wish**

1. **Conception du Data Lake**
   1. **Identification des Sources de Données**
2. **Transactions Clients (Base de Données Relationnelle)** :
   * **Type de Données** : Structurées (informations clients, commandes, paiements).
   * **Volume** : Plusieurs millions d'enregistrements mensuels.
   * **Vélocité** : Fréquence de mise à jour élevée (par batch).
   * **Utilisation** : Analyse des comportements d'achat, suivi des tendances de vente, recommandations de produits.
3. **Logs des Serveurs Web** :
   * **Type de Données** : Non structurées (logs d'accès, logs d'erreur, parcours utilisateur sur le site).
   * **Volume** : Enorme, surtout lors des périodes de forte activité (ex. : Black Friday).
   * **Vélocité** : Collecte en batch, regroupée en fichiers à intervalles réguliers.
   * **Utilisation** : Analyse de la performance, optimisation de l'expérience utilisateur, détection des problèmes en post-analyse.
4. **Données des Médias Sociaux** :
   * **Type de Données** : Semi-structurées (posts, likes, commentaires, sentiments des utilisateurs, format JSON).
   * **Volume** : Variable, dépend des campagnes et de l'engagement des clients.
   * **Vélocité** : Traitement en temps réel pour optimiser les campagnes.
   * **Utilisation** : Analyse des sentiments, optimisation des campagnes marketing, suivi de la perception de la marque.
5. **Flux de Données en Temps Réel (Campagnes Publicitaires)** :
   * **Type de Données** : En streaming (clics, impressions, conversions).
   * **Volume** : Très élevé pendant les campagnes marketing.
   * **Vélocité** : Traitement en temps réel nécessaire pour ajuster les dépenses publicitaires.
   * **Utilisation** : Optimisation des dépenses, analyse de l'efficacité des annonces, détection des clics frauduleux.
   1. **Architecture complète du Data Lake**
6. **Ingestion des Données**

L’ingestion des données peut se faire en temps réel ou par batch, en fonction de la nature et de la source des données.

* 1. **Sources de Données** :
  + **Transactions Clients** : Ingestion en batch à intervalle régulier
  + **Logs des Serveurs** : Ingestion en batch à intervalle régulier
  + **Données des Médias Sociaux** : Ingestion en batch à intervalle régulier
  + **Flux de Données Publicitaires** : Ingestion en streaming pour suivre les clics, les impressions, et les conversions des campagnes.
  1. **Services d'Ingestion sur GCP :**
  + **Cloud Storage** : Pour le stockage initial des fichiers bruts (comme les fichiers JSON, CSV, ou logs).
  + **Big Query**: Pour analyser les données nettoyées

1. **Zones de Stockage dans le Data Lake**

Le Data Lake est organisé en plusieurs zones pour gérer efficacement les données à différents stades de traitement.

* 1. **Zone Brute (Raw Zone)** :
  + **Stockage**: **Google Cloud Storage (GCS)**
  + **Contenu** : Données brutes ingérées, non transformées, telles qu'elles sont collectées des sources. Par exemple, les fichiers logs, les fichiers JSON de médias sociaux, et les exports de bases de données relationnelles.
  + **Utilisation** : Archivage des données pour assurer la traçabilité et servir de point de départ pour toutes les transformations.
  1. **Zone Nettoyée (Cleansed Zone)** :
  + **Stockage**: **Google Cloud Storage (GCS)**
  + **Contenu** : Données nettoyées, sans doublons, formatées de manière uniforme. Toutes les anomalies sont corrigées, et les formats de données sont standardisés.
  1. **Zone Enrichie (Enriched Zone)** :
  + **Stockage** : **BigQuery**
  + **Contenu** : Données enrichies, prêtes pour l'analyse. Par exemple, les transactions **clients** enrichies avec des scores de recommandation, ou des logs de serveurs agrégés pour l'analyse de la performance.
  + **Traitement** : **Apache Spark** pour les transformations avancées et l'enrichissement.

1. **Traitement et Transformation des Données**
   1. **Traitement Batch** :
   * **HDFS**
   * **Spark** : Pour simplifier l'extraction, la transformation et le chargement des données.
   1. **Traitement en Temps Réel** :
   * **Kafka** : Pour les pipelines de traitement en streaming. Cela permet de nettoyer et d'analyser les flux de données en temps
   * **BigQuery** : Pour exécuter des requêtes analytiques en temps réel sur les données enrichies.
2. **Analyse et Exploitation des Données**
   1. **BigQuery** :
   * **Utilisation** : Analyse à grande échelle des données enrichies. Vous pouvez exécuter des requêtes SQL pour obtenir des insights sur les ventes, les performances des campagnes marketing, ou le comportement des clients.
   * **Intégration BI** : Connexion facile avec des outils comme **Google Data Studio**, **Tableau**, ou **Power BI** pour créer des tableaux de bord et des rapports.
   1. **Outils de Machine Learning** :
   * **AI Platform** : Pour créer et déployer des modèles de machine learning utilisant les données enrichies. Par exemple, un modèle de recommandation de produits ou une détection de fraude.
   * **BigQuery ML** : Pour entraîner des modèles directement dans BigQuery, sans avoir à déplacer les données.
3. **Sécurité, Gouvernance et Qualité des Données**
   1. **Gouvernance des Données** :
   * **Data Catalog** : Pour cataloguer et suivre les métadonnées des jeux de données. Cela permet de faciliter la découverte des données et de gérer les autorisations d'accès.
   * **Audit et Traçabilité** : Journalisation des accès aux données et des modifications via **Cloud Audit Logs**.
   1. **Sécurité** :
   * **Cloud IAM (Identity and Access Management)** : Pour gérer les permissions d'accès aux ressources GCP.
   * **Chiffrement** : Toutes les données sont chiffrées en transit et au repos. Utilisation de clés de chiffrement gérées par Google ou de clés client pour plus de sécurité.
   * **Data Loss Prevention (DLP)** : Pour détecter et protéger les données sensibles, comme les informations personnelles des clients.
   1. **Diagramme des flux de données**

Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement

* 1. **Scripts pour le déploiement de l’infrastructure**

1. **Création des buckets : zone brute et zone nettoyée**

Code :

# Créez un bucket GCS pour la zone brute (logs serveurs)

gsutil mb -l us-central1 gs://data-lake-zone-brute

# Créez un bucket GCS pour la zone nettoyée

gsutil mb -l us-central1 gs://data-lake-zone-nettoyee

But : Préparer l’espace de stockage des données

**Une image contenant texte, capture d’écran, Police

Description générée automatiquement**

1. **Configuration de la Sécurité et de la Gouvernance (IAM, DLP)**

Code :

# Exemple pour donner l'accès de lecture à un utilisateur spécifique sur le bucket de la zone brute

gsutil iam ch user:victoire.borletsis@gmail.com:objectViewer gs://votre-bucket-zone-brute

Une image contenant texte, capture d’écran, Police

Description générée automatiquementBut : Assurer la sécurité et le contrôle d’accès aux ressources du Data Lake

1. **Ingestion et Transformations des Données**
2. **Ingestion de données brutes**

Il faut d’abord déposer les données brutes dans le bucket : data-lake-zone-brute

Il y a 3 types de données brutes :

* Transactions clients : .csv
* Logs servers : .log
* Médias sociaux : .json

Une fois que les données sont dans la zone brute, on utilise HDFS et Spark pour venir chercher les données.

Il faut donc établir une connexion avec le cloud Google Cloud Platform.

Nous avons installé dans un premier temps google-cloud-sdk :

* sudo snap install google-cloud-sdk –classic

Puis nous nous somme identifier :

* gcloud auth login

Nous avons ensuite récupérer les données brutes sur le cloup :

* gsutil cp -r gs://data-lake-zone-brute /home/stephanelam/DataLakes

Sur HDFS nous avons ensuite créer le répertoire /data/raw pour enrichir la base de données pour chaque type de données .logs / .csv / .json / streaming :

* hdfs dfs -mkdir -p /data/raw/logs
* hdfs dfs -mkdir -p /data/raw/json
* hdfs dfs -mkdir -p /data/raw/csv
* hdfs dfs -mkdir -p /data/raw/streaming
* hdfs dfs -put /home/stephanelam/DataLakes/data-lake-zone-brute/logs-servers/\* /data/raw/logs
* hdfs dfs -put /home/stephanelam/DataLakes/data-lake-zone-brute/medias-sociaux/\* /data/raw/json
* hdfs dfs -put /home/stephanelam/DataLakes/data-lake-zone-brute/transactions-clients/\* /data/raw/csv

Une fois tout mis en place, il ne nous reste plus qu’à analyser et traiter ces données via les 4 scripts que nous avons mis en place, un pour chaque format :

* spark\_logs
* spark\_json
* spark\_csv
* spark\_streaming

Il faudra les lancer avec la commande :

* spark-submit --packages com.crealytics:spark-excel\_2.12:0.13.5 <NOM\_SCRIPT>.py

Si besoin il faudra rajouter les packages nécessaires pour le traitement et la lecture de chaque format :  
- spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.5.2,com.crealytics:spark-excel\_2.12:0.13.5 <NOM\_SCRIPTS>.py

Ces données sont traitées, nettoyées sur HDFS puis mis dans le dossier /data/clean.

Une fois cela fait on récupére les nouvelles données via la commande suivante :

* hdfs dfs -get /data/clean /home/stephanelam/DataLakes/clean-data

Puis déposées dans le bucket data-lake-zone-nettoyee sur GCP :

* gsutil cp -r /home/stephanelam/DataLakes/clean-data gs://data-lake-zone-nettoyee

Concernant le streaming, il faudra créer un topic :

* bin/kafka-topics.sh --create --topic my\_topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

Enrichir la base de données du fichier .csv locale pour le streaming :

* hdfs dfs -put Social\_Network\_Ads.csv /data/raw/streaming

Et executer le code :

* spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.5.2,com.crealytics:spark-excel\_2.12:0.13.5 spark\_streaming.py

1. **Pipeline de Transformation des Données**

Pour automatiser le processus nous avons utiliser Apache Airflow pour pouvoir orchestrer un workflow et définir des pipelines ETL.  
  
Dans un premier temps nous avons installé Apache Airflow :

* pip install apache-airflow

Puis créer un répertoire pour initialiser notre base de données :

* mkdir airflow\_home
* export AIRFLOW\_HOME=$(pwd)/airflow\_home
* airflow db init

Nous avons aussi installé les modules nécessaires Google pour configurer notre connexion par la suite :

* pip install apache-airflow-providers-google

Une fois toutes les installations faites, vous pouvez lancer l’interface Airflow :

* airflow webserver --port 8080

Ainsi que le scheduler pour exécuter les workflows (dans un autre terminal) :

* airflow scheduler

Vous pouvez désormais accéder à l’interface web :

* <http://localhost:8080>

Pour accéder vous devez le compte par défaut :

* **Username** : airflow
* **Password** : airflow

Vous êtes à présent connecté, maintenant nous allons ajouter notre connexion GCP à notre Airflow.

Dans l’interface Airflow, cliquez sur l’onglet **Admin** > **Connections**.

Puis cliquez sur le bouton **+** pour ajouter une nouvelle connexion.  
Ensuite vous allez devoir remplir les informations de connexion :

* **Conn Id** : google\_cloud\_default
* **Conn Type** : Google Cloud
* **Keyfile Path** : Chemin local vers le fichier JSON des informations d'identification de votre clé de service Google Cloud.

Une fois toutes les informations renseigner, vous pouvez faire **« Save »** pour enregistrer votre connexion.

Ensuite nous allons créer notre Pipeline ELT (DIAG) nommé etl\_gcp\_pipeline.py dans le dossier dags d’Airflow (par défaut, $AIRFLOW\_HOME/dags).

Placez le fichier Python dans le répertoire dags d’Airflow.

Redémarrez le serveur web et le scheduler Airflow :

* airflow webserver --port 8080
* airflow scheduler

Accédez à l’interface Airflow <http://localhost:8080>

Activez le DAG (etl\_gcp\_pipeline) et surveillez son exécution.