Projet M2 DSIA : Évaluation des Pipelines de Données AWS et GCP

Le projet vise à concevoir, implémenter et évaluer aux étudiants deux pipelines de traitement de données équivalents, l'un sur **Amazon Web Services (AWS)** et l'autre sur **Google Cloud Platform (GCP)**.

https://immobilier-au-senegal.com/

Objectifs Pédagogiques

1. **Maîtriser** l'intégration de services *Serverless* et *Compute* pour des tâches de *data engineering*.
2. **Comprendre** les similarités et les différences de mise en œuvre entre AWS et GCP.
3. **Évaluer** les aspects de coût, performance, et maintenabilité des deux architectures.
4. **Développer** des compétences en **Scripting Python** (Scraping, ETL)

Architecture du Projet

Partie 1 : Pipeline de Données AWS

Étape	Service	Rôle
1. Gestion du Script	**Image Docker**	Héberger le script Python de scraping.
2. Scraping & Compute	**EC2 (Elastic Compute Cloud)**	Une instance EC2 exécute l'image du script de scraping.
3. Sortie du Scraping	**S3**	L'instance EC2 écrit les données brutes scrapées dans un *bucket* S3.

4. Déclenchement / Transfert	**Lambda**	Le code de la fonction Lambda lit les données de S3 et les envoie directement vers le *bucket* **GCS** de GCP.

Partie 2 : Ingestion et Analyse GCP

Étape	Service GCP	Rôle
6. Stockage Intermédiaire	**GCS (Cloud Storage)**	Réceptionne les données brutes envoyées par la fonction Lambda AWS.
7. Ingestion Serverless	**Cloud Function**	Déclenchée par l'arrivée d'un nouveau fichier dans le *bucket* GCS.
8. Chargement & Transformation	**Cloud Function**	Charge les données dans **BigQuery (BQ)** après une potentielle validation/légère transformation.
9. Analyse	**BigQuery (BQ)**	Stockage final des données structurées et exécution de requêtes d'analyse.

Détail des Tâches pour les Étudiants

Phase 1 : Préparation et Scraping (AWS)

1. **Création du Scraper :** Écrire un script Python (e.g., avec BeautifulSoup ou Scrapy) qui scrape des données d'un site web public (e.g., une liste de produits, un classement sportif, etc.). Le script doit sauvegarder la sortie en format **CSV**.

2. **Mise en place de S3 :** Créer le *bucket* S3 pour le stockage du script et le *bucket* de sortie pour les données brutes.
3. **Configuration EC2 :**
    - Lancer une instance EC2 (e.g., t2.micro) avec un rôle IAM permettant l'écriture dans le *bucket* S3 de sortie.
    - Installer les dépendances nécessaires et configurer la tâche pour qu'elle s'exécute (e.g., via un cron job ou manuellement).

Phase 2 : Transfert Cross-Cloud (AWS -> GCP)

1. **Configuration GCP (Clé de Service) :** Créer un compte de service GCP et sa clé pour permettre à AWS d'écrire dans GCS.
2. **Configuration GCS :** Créer le *bucket* de destination sur GCP (GCS).
3. **Fonction Lambda (Transfert) :**
    - Créer une fonction **Lambda** Python avec un déclencheur **S3 Object Created**.
    - Le code de la Lambda doit :
        - Récupérer le fichier fraîchement écrit dans S3.
        - Utiliser la bibliothèque cliente GCP (google-cloud-storage) pour copier le contenu dans le *bucket* GCS.

Phase 3 : Ingestion et Analyse (GCP)

1. **Cloud Function (Ingestion) :**
    - Créer une **Cloud Function** Python avec un déclencheur **Cloud Storage Object Finalize/Create**.
    - Créer le jeu de données et la table de destination dans **BigQuery**.
    - Le code de la Cloud Function doit :
        - Lire les données du fichier GCS déclencheur.
        - Effectuer une validation/nettoyage de base (e.g., s'assurer que les types de données correspondent au schéma BQ).
        - Utiliser la bibliothèque BigQuery pour charger les données dans la table BQ.
2. **Analyse BigQuery :** Les étudiants doivent écrire et exécuter au moins **3 requêtes SQL d'analyse complexes** dans BigQuery pour démontrer que le pipeline a fonctionné et que les données sont utilisables (e.g., agrégation, jointures, fonctions analytiques).

Rapport d'Évaluation (Livrable)

Le livrable principal sera un rapport qui met l'accent sur l'évaluation comparative des deux environnements.

Documentation Technique

- Diagrammes d'architecture pour les deux parties (**AWS** et **GCP**).
- Détails de l'implémentation (code des scripts, des fonctions *serverless*, configuration IAM/Rôles).