Rapport des Ateliers d'architecture décisionnelle

Encadré par : Rakib Sheikh

Présentés par :

- Sokhna Awa Falilou SOW
- Ndeye Khady KASSE
- Gustave Rachid MABIKANA
- Khalifa MBOUP
- Kévine Josias DIANTOUADI-FRAY

Année scolaire : 2023/2024

Sommaire

A. TP 1 : Business Intelligence

Dans le TP 1 nous avons d'abord par importer les bibliothèques suivantes :

```
import os
import urllib.request

from minio import Minio, S3Error
```

os : Ce module fournit des fonctions permettant d'interagir avec le système d'exploitation, comme la gestion de fichiers et de répertoires.

urllib.request: Ce module permet d'effectuer des requêtes HTTP (comme télécharger des fichiers) en utilisant les protocoles Internet.

Minio et S3Error depuis le module minio : Le module Minio fournit une interface pour interagir avec un serveur Minio, tandis que S3Error est une exception spécifique au module Minio qui est levée lorsqu'une erreur se produit lors de l'interaction avec le serveur Minio.

On a ajouté dans la fonction **main ()** :

```
def main() -> object:
    grab_data()
    write_data_minio()
    return None
```

def main() -> object: : Cette ligne déclare une fonction appelée main() qui ne prend aucun argument et renvoie un objet de type object. La spécification -> object indique que la fonction retournera un objet de type object, bien que dans ce cas précis, elle retourne None.

grab_data() : Cette ligne appelle la fonction grab_data(). Cette fonction est responsable du téléchargement des données à partir des URL spécifiées et de leur sauvegarde localement.

write_data_minio() : Cette ligne appelle la fonction write_data_minio(). Cette fonction prend les données téléchargées et les envoie vers le serveur Minio pour les stocker.

return None : Cette ligne renvoie explicitement None. Bien que cela ne soit pas nécessaire car en Python, si une fonction ne retourne rien, elle renvoie automatiquement None. C'est juste une indication explicite que la fonction main() ne retourne rien d'autre que None.

Puis nous avons ajouté dans la fonction **grap data ()** :

```python
def grab_data() -> None:
    urls = [
        'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-11.parquet',
        'https://d37ci6vzurychx.cloudfront.net/trip-data/yellow_tripdata_2023-12.parquet',
    ]

    save_dir = "../../data/raw"
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)
        print(f"Dossier {save_dir} créé.")

    for url in urls:
        file_name = url.split('/')[-1]
        save_path = os.path.join(save_dir, file_name)
        print(f"Tentative de téléchargement de {file_name}...")

        try:
            urllib.request.urlretrieve(url, save_path)
            print(f"{file_name} téléchargé avec succès et enregistré dans {save_path}.")
        except Exception as e:
            print(f"Échec du téléchargement de {file_name}. Erreur : {e}")
```

def grab_data() -> None: : Cette ligne déclare une fonction appelée grab_data() qui ne prend aucun argument et ne retourne rien (None).

urls = [...] : Cette ligne définit une liste urls contenant les URL à partir desquelles les données doivent être téléchargées.

save_dir = "../../data/raw" : Cette ligne définit le répertoire dans lequel les fichiers téléchargés seront enregistrés.

if not os.path.exists(save_dir): ... : Cette structure conditionnelle vérifie si le répertoire de sauvegarde spécifié existe déjà. Si ce n'est pas le cas, il crée le répertoire en utilisant os.makedirs(save_dir) et affiche un message indiquant que le répertoire a été créé.

for url in urls: ... : Cette boucle itère sur chaque URL dans la liste urls.

file_name = url.split('/')[-1] : Cette ligne extrait le nom de fichier à partir de l'URL en utilisant split('/')[-1]. Cela permet de récupérer le dernier élément de l'URL, qui est le nom du fichier après le dernier slash ("/").

save_path = os.path.join(save_dir, file_name) : Cette ligne crée le chemin complet de sauvegarde en joignant le répertoire de sauvegarde avec le nom de fichier à l'aide de os.path.join().

try: ... except Exception as e: ... : Cette structure try-except tente de télécharger le fichier à partir de l'URL en utilisant urllib.request.urlretrieve(url, save_path). Si le téléchargement réussit, un message de succès est affiché. Si une exception est levée (par exemple, en cas d'erreur de connexion), un message d'échec est affiché avec des détails sur l'erreur.

Enfin nous avons ajouté dans la fonction **write_data_minio()** :

```python
def write_data_minio():
    client = Minio(
        "localhost:9000",
        access_key="minio",
        secret_key="minio123",
        secure=False
    )

    bucket_name = "atl-datamart-project"
    try:
        found = client.bucket_exists(bucket_name)
        if not found:
            client.make_bucket(bucket_name)
            print(f"Bucket {bucket_name} créé.")
        else:
            print(f"Bucket {bucket_name} existe déjà.")

        data_dir = "../../data/raw"
        for file_name in os.listdir(data_dir):
            if file_name.endswith(".parquet"):
                file_path = os.path.join(data_dir, file_name)
                print(f"Tentative de téléversement de {file_name} dans {bucket_name}...")

                with open(file_path, "rb") as file_data:
                    file_stat = os.stat(file_path)
                    client.put_object(
                        bucket_name,
                        file_name,
                        file_data,
                        file_stat.st_size
                    )
                print(f"{file_name} téléversé avec succès dans {bucket_name}.")

    except S3Error as e:
        print(f"Erreur MinIO lors de l'accès au bucket {bucket_name} : {e}")
    except Exception as e:
        print(f"Erreur inattendue : {e}")


if __name__ == '__main__':
    main()
```

client = Minio(...) : Cette ligne crée une instance du client Minio en spécifiant l'adresse du serveur, les clés d'accès et d'autres paramètres nécessaires.

bucket_name = "atl-datamart-project" : Cette ligne définit le nom du bucket dans lequel les données seront stockées sur le serveur Minio.

try: ... except S3Error as e: ... except Exception as e: ... : Cette structure try-except gère les erreurs potentielles lors de l'accès au serveur Minio ou lors de l'écriture des données. Si une exception de type S3Error est levée, elle est interceptée et un message d'erreur spécifique à Minio est affiché. Les autres exceptions sont également interceptées, et un message générique d'erreur est affiché.

found = client.bucket_exists(bucket_name) : Cette ligne vérifie si le bucket spécifié existe déjà sur le serveur Minio.

if not found: ... else: ... : Si le bucket n'existe pas, il est créé en utilisant client.make_bucket(bucket_name). Sinon, un message indiquant que le bucket existe déjà est affiché.

data_dir = "../../data/raw" : Cette ligne définit le répertoire local à partir duquel les fichiers doivent être téléversés vers le serveur Minio.

for file_name in os.listdir(data_dir): ... : Cette boucle itère sur chaque fichier dans le répertoire local spécifié.

if file_name.endswith(".parquet"): ... : Cette condition vérifie si le fichier a une extension ".parquet", ce qui indique qu'il s'agit d'un fichier de type Parquet à téléverser.

file_path = os.path.join(data_dir, file_name) : Cette ligne construit le chemin complet du fichier à partir du répertoire local et du nom de fichier.

with open(file_path, "rb") as file_data: ... : Cette structure with ouvre le fichier en mode lecture binaire ("rb") et le traite en tant que file_data.

file_stat = os.stat(file_path) : Cette ligne récupère les informations sur le fichier, notamment sa taille, en utilisant os.stat().

client.put_object(...) : Cette ligne envoie le contenu du fichier au bucket spécifié sur le serveur Minio en utilisant la méthode put_object(). Elle inclut le nom du bucket, le nom du fichier, le contenu du fichier, et sa taille.

B. TP 2 : Le système d'alimentation ETL

Dans le TP 2 nous avons presque rien changé à part dans la fonction **write_data_progres()**

```
except Exception as e:
    print(f"Error connecting to the database: {e}")
    return False

return True
```

Dans ce bloc de code, lorsqu'une exception est levée, elle est capturée, et un message d'erreur est affiché avec print(). Ensuite, la fonction retourne False pour indiquer qu'une erreur s'est produite lors de la connexion à la base de données. Si aucune exception n'est levée, la fonction retourne True, ce qui signifie que la connexion à la base de données a réussi.