TP : Apache Spark Streaming

1 Script Python pour générer des données

Téléchargez le fichier data-generator.py disponible sur Arche et examinez son contenu. Ce script Python génère en continu des phrases aléatoires et les envoie à un client via un socket. Il constitue la source de données utilisée par Spark Streaming pour recevoir et analyser un flux en temps réel.

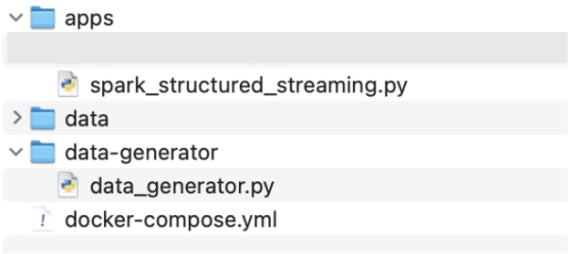
Le script agit comme un serveur et réalise les opérations suivantes :

1. ouvre un socket en attente de connexions entrantes
2. accepte la connexion lorsqu'un client (comme un programme Spark Streaming) se connecte
3. envoie périodiquement une phrase aléatoire au client
4. poursuit l'envoi tant que le programme reste actif
5. gère les erreurs éventuelles et ferme correctement la connexion si nécessaire.

Après avoir téléchargé le fichier, placez-le dans un répertoire nommé data-generator.

2 Script PySpark pour Spark Structured Streaming

Téléchargez le fichier spark_structured_streaming.py et placez-le dans le dossier app. Votre répertoire aura alors la structure suivante :



Ce script implémente un job Spark Structured Streaming permettant d'analyser des flux de texte en temps réel de manière optimisée. Le programme utilise Apache Spark Structured Streaming pour :

- Se connecter à une source de données en continu (le data-generator).
- Lire les phrases envoyées via un socket en tant que DataFrame de streaming.
- Nettoyer et traiter les données en éliminant la ponctuation et les espaces superflus.
- Compter les occurrences des mots reçus en temps réel.
- Afficher les résultats dans la console à intervalles réguliers.

Une fois le fichier téléchargé, placez-le dans un dossier nommé apps.

3 Configuration Docker

Téléchargez le fichier docker-compose.yml disponible sur Arche et examinez sa structure.

Ce fichier décrit l'ensemble des services Docker nécessaires au déroulement du TP, notamment:

- spark-master : le conteneur correspondant au nœud maître Spark, chargé de la coordination du cluster ;
- spark-worker-a et spark-worker-b : les conteneurs des nœuds workers qui exécutent les tâches distribuées ;
- data-generator : un conteneur exécutant un script Python chargé de produire des phrases aléatoires et de les transmettre en continu via un socket.
- Le dossier ./data-generator:/app est monté dans le conteneur afin d'y exécuter data_generator.py.
- Un réseau Docker nommé spark-network assure la communication entre les différents conteneurs.
- Les ports exposés donnent accès à l'interface web de Spark et permettent de soumettre des tâches au cluster.

4 Exécuter le Job Spark Structured Streaming

- Placez-vous dans votre répertoire du travail et lancer votre cmd (Terminal).
- Lancez vos conteneurs en utilisant la commande suivante :

```
docker compose up -d
```

- Accédez au conteneur Master de Spark

```
docker exec -it spark-master bash
```

- Pour soumettre le job Spark Structured Streaming, exécutez la commande suivante :

```
spark-submit --master spark://spark-master:7077 --name
StructuredStreamingExample /opt/spark-apps/spark_structured_streaming.py
```

- Une fois le job Spark Streaming lancé, vous devriez voir les résultats dans la console du conteneur spark-master. Par exemple :

```
-------------------------------------------
Batch: 1
-------------------------------------------
+-----------+-----+
|       word|count|
+-----------+-----+
|       beau|    1|
|     canapé|    1|
|         Le|    2|
|       dort|    1|
|         La|    3|
|    machine|    1|
|        sur|    1|
|   learning|    1|
|       chat|    1|
|         Il|    1|
|   amusante|    3|
|    langage|    2|
|  Streaming|    1|
|     génial|    2|
|       fait|    1|
|      monde|    1|
|     Python|    2|
|         un|    2|
|     change|    1|
|aujourd'hui|    1|
+-----------+-----+
only showing top 20 rows
```