Projet Spark

Traitement distribué

Sommaire :

I. État d'avancement

1. Travail réalisé :

- Conception et implémentation d'un générateur de logs HTTP paramétrable (genlogs_v2.py).
- Développement d'un job Spark Structured Streaming (log_analyzer.py) capable de :
    - Consommer un flux TCP via Netcat.
    - Parser et filtrer les logs d'erreur.
    - Agréger les métriques par fenêtre de temps.
    - Déclencher des alertes en cas de volumétrie élevée.
    - Sauvegarder les logs d'erreur dans un système de fichiers local.
- Automatisation du lancement et du nettoyage via un script bash (run_spark_streaming.sh).
- Validation des formats, du déclenchement d'écriture (gestion des checkpoints, parsing du timestamp "Z", réencodage JSON).

Points en cours / à venir :

- **Intégration Kafka** : remplacer Netcat par un topic Kafka local pour simuler un véritable pipeline de données.

2. Statut global :

La preuve de concept est opérationnelle : génération de logs, traitement streaming, alerting et persistance fonctionnent en local. Les prochaines itérations porteront sur la production-readiness, l'industrialisation et l'enrichissement fonctionnel.

II. Script de génération de logs

Le script **genlogs_v2.py** est un générateur de logs HTTP minimaliste, capable de produire un flux continu de requêtes factices pour alimenter un pipeline de traitement en streaming.

1. Fonctionnalités principales :

- Génération d'un horodatage précis au format ISO 8601 UTC (ex. 2025-06-09T09:42:40.001034Z).
- Attribution aléatoire d'une adresse IPv4.
- Sélection aléatoire d'une méthode HTTP parmi GET, POST, PUT, DELETE.
- Choix d'une URL de la forme /resource/{i} pour i de 1 à N (paramétrable).
- Attribution d'un code de statut HTTP selon une répartition paramétrable (200, 404, 500 par défaut 90/5/5 %).
- Sortie en JSON (par défaut) ou CSV (; délimiteur).
- Contrôle du débit de génération (logs par seconde).

2. Usage :

Méthode pour rendre le script exécutable ou l'invoquer via python3 :

```
python .\utils\genlogs_v2.py --urls 1000 --rate 1000 --format json #1000 logs par seconde
python .\utils\genlogs_v2.py # 1 log par seconde
python3 ./utils/genlogs_v2.py --urls 1000 --rate 1000 --format json | nc -lk 9999
python3 ./utils/genlogs_v2.py --urls 1000 --rate 1000 --format json --status-dist 0 0 100 | nc -lk 9999
```

Options disponibles :

- --urls N : nombre d'URLs différentes à simuler (par défaut 10).
- --rate R : taux de génération (logs par seconde, par défaut 1).
- --format json|csv : format de sortie (JSON par défaut).
- --methods M1 M2 … : liste des méthodes HTTP à utiliser.
- --status-dist P_OK P_404 P_500 : répartition en pourcentages pour 200, 404, 500 (par défaut 90 5 5).

III. Analyseur Spark Streaming

Le composant **log_analyzer.py** utilise Spark Streaming pour consommer, filtrer et analyser un flux de logs en temps réel.

Le script **run_spark_streaming.sh** automatise le nettoyage des anciens résultats et le lancement du job Spark.

1. Fonctionnalités de log_analyzer.py :

- **Connexion TCP** au port 9999 (flux Netcat).
- **Parsing JSON** via un schéma explicite (timestamp, ip, method, url, status).
- **Conversion du timestamp** ISO 8601 UTC avec gestion du suffixe Z (to_timestamp(..., "yyyy-MM-dd'T'HH:mm:ss.SSSSSSX")).
- **Filtrage** : extraction uniquement des logs d'erreur (status >= 400).
- **Fenêtrage** de 30 secondes (watermark 10 s) pour calculer :
    - nombre d'erreurs (error_count),
    - nombre approximatif d'IP uniques (unique_ips),
    - liste des URLs concernées (error_urls).
- **Alertes** : affichage dans la console si error_count > 100.
- **Sauvegarde** : chaque log d'erreur est réencodé en JSON (to_json(struct(*))) et écrit en mode texte dans output/errors (par défaut trigger toutes les 10 s).
- **Affichage live** des métriques agrégées toutes les 10 s.

2. run_spark_streaming.sh

Ce script shell effectue :

- Suppression des dossiers output/errors et checkpoint/errors pour repartir d'un état vierge.
- Lancement de spark-submit sur spark-streaming/log_analyzer.py.
- Retour du code de sortie pour indiquer succès ou échec.

3. Exécution :

Pour préparer le flux de logs dans un terminal :

1. Dans un terminal, lancer la commande :

```
python3 ./utils/genlogs_v2.py --urls 1000 --rate 1000 --format json | nc -lk 9999
```

2. Dans un autre terminal, démarrer l'analyse :

Option recommandée :

```
chmod +x ./run_spark_streaming.sh
```

Option manuelle :

```
spark-submit ./spark-streaming/log_analyzer.py
```

Remarque : Vous trouverez toute la procédure d'exécution dans le répo Git dans /doc/procedure.md

Sorties attendues :

- **Console** : alertes et tableaux de métriques fenêtre par fenêtre.
- **Fichiers** : logs d'erreur JSON dans output/errors, horodatés par batch.
- **Checkpoint** : état Spark dans checkpoint/errors.

Historiques des modifications

Date	Nom	Date de la modification
19/06/2025	RENAUDIN Bertrand	19/06/2025
21/06/2025	RENAUDIN Bertrand	21/06/2025