Exercice Grafana Pipeline

Système de gestion de log et d’alerte multi Système

Chaîne d’ingestion logs multi système → Kafka → Solution (Traitement) → Grafana

Une image contenant texte, diagramme, ligne, Plan

Le contenu généré par l’IA peut être incorrect.

0) Contexte & objectif  
On souhaite monitorer un microservice d’ingestion de données. Aujourd’hui il s’exécute une fois toutes les heures et stock dans le fichier logs.json les log. Demain il devra être event-driven (fréquence plus élevée) et obtenir en temps réel et dans un tableau de bords Grafana avec des alertes.

À partir d’un fichier de logs JSON (fourni: logs.json), vous devez construire un flux complet : 1. Producer en Python : lit le fichier de logs et envoie chaque événement dans Kafka.

2. Traitement : consomme ces événements depuis un topic Kafka, parse le JSON, choisit les labels, et pousse vers une solution pour gérer les données des events.

3. Grafana : expose un tableau de bord + alertes en temps réel centrés sur la qualité de l’ingestion (erreurs, latence, CPU/Mémoire, fraîcheur) voir détail dans le document de KPI.  
  
1) Donnée fournie  
Le fichier logs.json contient des événements JSON avec les champs suivants : Etat, Source, Objet, Entreprise, Zone, Debut, Fin, Duree secondes, cpu\_time (sec), memory\_used (bytes), Description.

2) Spécifications technique   
- Utiliser docker et docker compose pour déployer la stack

- Tout les codes (producer et consumer Kafka) le cas échéant doivent être en Python

- La solution doit fonctionné dans docker sur un host windows ou linux

2) Tâches à réaliser  
  
0 Proposer une structure améliorer des event (logs) en JSON

1 Producer Python → Kafka : script producer.py qui lit logs.json et publie sur le topic  
ingestion-logs.  
2 Kafka : déployer un broker (Docker Compose accepté), créer le topic ingestion-logs.  
3 Traitement : consommer le topic ingestion-logs, parser le JSON, définir labels et payloads.  
4 Grafana : créer un tableau de bord selon les spécifications de l’autre document.

5 Alerting : définir au moins 4 alertes (échec, taux d’échec, source muette, latence dégradée).  
  
4) Livrables  
Repository avec une solution ready for production:

- README.md clair avec architecture, instructions d’exécution et preuves de fonctionnement.

- Code & configuration (producer.py, docker-compose.yml, promtail-config.yml, export JSON du dashboard Grafana).

- Captures d’écran du dashboard et des alertes.

- Tout autre éléments jugé nécessaire