Projet de spark

**Questions préliminaires**

1) La première contrainte que l’architecture de la solution devra prendre en compte est la mémoire générée par les drones. En effet, chaque drone devrait produire 100 Gb de données par jour.

On peut également identifier une deuxième contrainte, le fait que PrestaCop veuille garder chaque message généré par les drones, il va donc falloir des serveurs pour les stocker.

Pour gérer les contraintes ci-dessus, il va falloir mettre en place du streaming. En effet, si les données récupérées ne servent qu’à faire des statistiques, il est inutile de tout stocker, on peut faire directement des analyses en temps réel avec le streaming.

Pour avoir une architecture qui réponde aux différents critères, il faut récupérer les logs générés par les drones (ils deviennent Data Producers), mettre en place un serveur de stockage (HDFS/S3) qui va récupérer les données puis diviser les données en batches et les traiter avec spark streaming (cluster)

2) L’architecture de la solution devra prendre en compte le fait qu’il soit possible qu’un évènement particulier et rare se déclenche (1% des violations observées), cet évènement nécessite une prise de contrôle par un humain pour entrer le bon code de violation.

Pour gérer ces contraintes, il faudra réserver un emplacement de stockage spécifique où seront stockées toutes les données pour lesquelles le drone ne sait pas qualifier la violation, elles seront ensuite traitées par un policier puis renvoyées dans le stream.

3) Prestacop a commis quelques erreurs qui expliquent son premier échec. En effet, ils ont voulu envoyer les données de la police de New York vers les ordinateurs de PrestaCop sans respecter pleinement les contraintes techniques, notamment en termes de mémoire (il est fort probable qu’ils n’aient pas utilisé le streaming).

4) Pour rendre le produit de Prestacop plus rentable, il serait intéressant de rajouter un champ status pour savoir si un policier a modifié le code de violation. En effet, cette information pourrait nous donner des statistiques intéressantes, on pourrait alors savoir à quel point l’intervention humaine est nécessaire pour que la solution soit fonctionnelle, et essayer de réduire cette part au maximum à l’avenir.

**Eléments pour l’architecture**

1) Petit programme python qui va simuler le drone et envoyer les données à notre solution via un stream de données

2) Gérer le message d’alerte du stream

3) Stockage du message dans un serveur de stockage distribué (ex: HDFS/S3)

4) Analyse des données avec du traitement distribué (spark) et répondre à 4 questions pertinentes

5) Chargement du CSV (anciennes infractions) dans notre serveur de stockage distribué. (lignes par lignes)

**Schéma représentant l’architecture de notre solution**

STREAM

Script drone

Ping



/





Alerte



Stock image + code violation

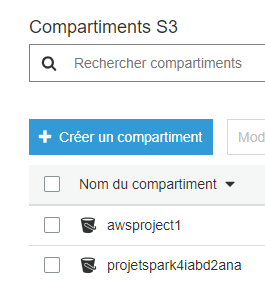
Sortie CSV pour ML / Reporting



Chargement ligne par ligne



**Etape 1** : Création d’un S3

****

**Etape 2** : Création du script qui va générer les messages du drone, puis les envoyer dans des dossiers spécifiques du S3 (logs\_drone, violation\_drone)

Infos utiles

|  |  |
| --- | --- |
| Information | Valeur |
| Fréquence des logs | 5 secondes |
|  |  |

Quand le drone n’arrive pas à lire, le mettre en pause et attendre une saisie manuelle (on enverra un log un peu différent lors des saisies manuelles)

**Etape 3** : Mise en place du chargement ligne par ligne du CSV de la police de New York (NYPD)

**Etape 4** : Ecrire le code Spark Streaming pour récupérer les données S3 et trouver des statistiques intéressantes

**Etape 5** : Sortir les données intéressantes dans un CSV