Plan de Projet

DAT

Résumé

1. Explication des outils : utilité / conso
2. Outils retenus, pourquoi ?
3. Structure
4. Explication de la méthode de projet

Mise en œuvre

I- Explication des outils

a) Utilité

1) Spark /(kafka)

Spark : Facilité d'utilisation, scalable, tolérance aux pannes (On peut recréer un nœud à partir d'une nœud parents).

2) Elastic, la suite

ElasticSearch: indexation et recherche de données en temps réel, moteur multi-entité à travers une interface REST

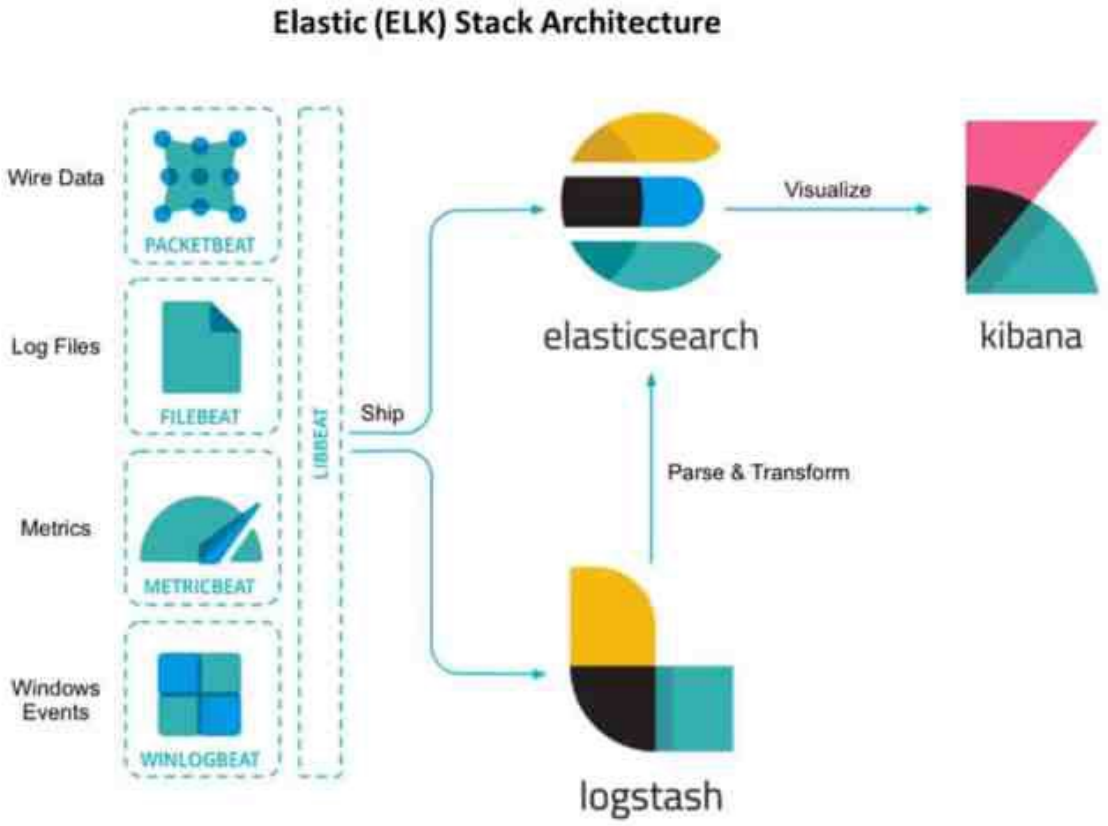
Logstash: ETL

Kibana: visualisateur de données

b) Conso

Logstash est un outil consommant énormément de ressources il est préférable d'opter pour un autre moyen de transfert de l'API à Elasticsearch pour de simples traitements.

Exemple de Structures possibles



II- Outils retenus

1) Pourquoi ?

Python

Nous avons retenus l'outil python pour plusieurs raisons:

- Facilité d'utilisation du langage qui est permissif.
- "Débugage" possible.
- Opérations de traitement sur le flux de données.

ElasticSearch

ETL performant de référence dans le milieu de la data

Kibana

Visualisation des données simples ainsi qu'un bon fonctionnement en tandem avec Elastic.
Pour manipuler la donnée.

2) Coûts Utilisation Mémoire

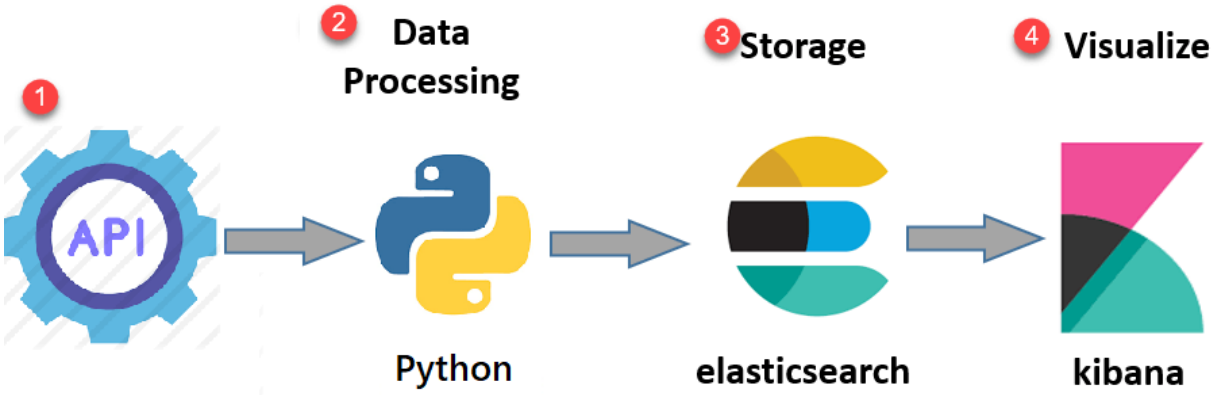
Avec ces 3 outils, le coût d'utilisation de la mémoire est d'environ 2 gigas.

Schémas / Synthèse

III- Structure

API URL pour récupération des logs :
https://data.rennesmetropole.fr/api/records/1.0/search/?dataset=etat-du-trafic-en-temps-reel&q=&facet=denomination

Notre Architecture (Stack)



Tâches pour déployer la Stack

Création de la stack:

- Création du docker compose
- Run du docker compose
- test de la stack et de l'état des conteneurs
- Mise en place d'une procédure de déploiement

Déploiement:

```yaml
version: '2.2'
services:

  kibana:
    image: docker.elastic.co/kibana/kibana:7.6.0
    container_name: kibana
    environment:
        SERVER_NAME: kibana
        ELASTICSEARCH_HOSTS: '["http://elasticsearch:9200"]'
    ports:
      - 5601:5601
    networks:
      - esnet
    depends_on:
      - "elasticsearch"

  elasticsearch:
    container_name: elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:7.6.0
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      - discovery.type=single-node
      - cluster.name=docker-
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    networks:
      - esnet
networks:
  esnet:
```

1) API URL pour récupérer des logs :
https://data.rennesmetropole.fr/api/records/1.0/search/?dataset=etat-du-trafic-en-temps-reel&q=traveltimereliability%20%3E=%2050&rows=1000

2) Script python : (modification des données et envoie à ElasticSearch)

Etape 1: Premier Index Elastic=> récupération des données

a) Modification de la requête HTTP GET sur Api pour traitement spécifique sur traveltimereliability / rows à 1000
b) Connexion à elasticsearch grâce au module Elasticsearch
c) Création de l'index et implémentation des données dans l'index en format json
d) Lancement du programme avec le module schedule. Réglé pour toutes les 5min pendant 1h. Lancement avec une boucle while et un sleep de 1 s.

Etape 2: Deuxième index Elastic => récupération geo_points uniquement

e) Requête HTTP pour récupérer les données triées (pareil que étape "a")
f) Utilisation du module Elasticsearch pour la connexion à l'ETL
g) Transformation au format geo_point de la donnée : "geo_point_2d", cela lui permet d'être interprété dans kibana et d'afficher la map avec les points géo
h) Parcours des données pour récupérer uniquement les "geo_point_2d"
i) Ajout dans un dictionnaire
j) Création de l'index et implémentation des données au format dictionary.
k) Lancement du programme avec le module schedule. Réglé pour toutes les 5min pendant 1h. Lancement avec une boucle while et un sleep de 1 s.