Université de Versailles
Saint-Quentin-en-Yvelines

Université Paris-Saclay

Master 2 DataScale

Projet : Data Pipeline

Indexation et visualisation de données massives

Binôme :

Mohammed Nassim FELLAH
Mohamed Ali BARDI

Année universitaire 2025–2026

1 Objectif du projet

L'objectif est de construire un pipeline complet permettant : (1) la collecte régulière de données via une API publique, (2) la transmission via Kafka, (3) la transformation et l'indexation dans Elasticsearch via Logstash, (4) l'analyse/visualisation via Kibana, et (5) un traitement distribué via Spark.

2 Choix des données et de l'API

Nous avons choisi des données de type **e-commerce** :
— **Produits** (catalogue)
— **Commandes** (événements)
La collecte est effectuée **régulièrement (cron job)** conformément à l'énoncé. Remarque : le catalogue produit est relativement stable, tandis que les commandes sont générées régulièrement pour simuler un flux temporel exploitable (séries temporelles / agrégations).

3 Architecture et flux de données

3.1 Vue globale du pipeline

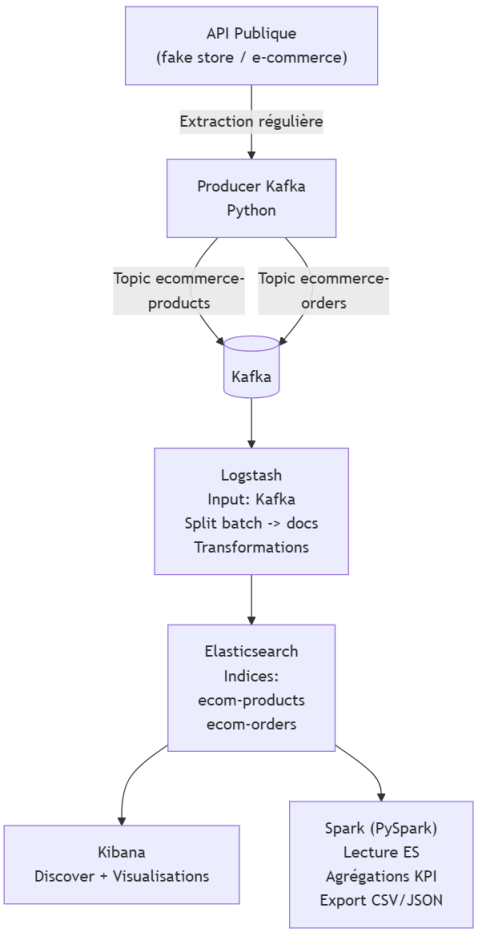

FIGURE 1 – Flowchart du pipeline de bout en bout.

3.2 Extraction régulière (cron job) + Kafka

Un producteur Kafka (Python) récupère ou génère les données régulièrement et les publie dans deux topics :

- `ecommerce-products`
- `ecommerce-orders`

Le format envoyé à Kafka est **batché** (un message = un lot). Logstash découpe ensuite le lot en documents unitaires pour l'indexation.

4 Transformation et indexation (Logstash + Elasticsearch)

4.1 Logstash

Logstash :

- lit depuis Kafka (input),
- **split** le champ `data` (batch → documents),
- applique des transformations (ex. `rating_rate`, `rating_count`, `total_items`),
- indexe dans Elasticsearch dans deux indices :
  - `ecommerce-products`
  - `ecommerce-orders`

4.2 Mapping Elasticsearch (analyzers)

Un mapping spécifique est défini, incluant :

- un analyzer **N-gram** sur le titre produit (recherche partielle),
- des champs `keyword` pour filtres/agrégations,
- des champs `date` pour séries temporelles (`ingest_ts`, `@timestamp`),
- un champ `nested` pour les lignes `products` dans les commandes.

5 Requêtes Elasticsearch obligatoires (les 5)

Le cahier des charges demande : 1 textuelle, 1 agrégation, 1 N-gram, 1 fuzzy, 1 série temporelle. Les requêtes ci-dessous sont prêtes à copier-coller dans *Kibana Dev Tools*.

5.1 A) Requête textuelle (produits)

```
GET ecommerce-products/_search
{
  "query": {
    "match": {
      "title": "jacket"
    }
  }
}
```

5.2 B) Requête avec agrégation (produits par catégorie)

```
GET ecommerce-products/_search
{
  "size": 0,
  "aggs": {
    "par_categorie": {
      "terms": { "field": "category" }
    }
  }
}
```

5.3 C) Requête N-gram (recherche partielle)

Nécessite le champ `title.ngram`.

```
GET ecommerce-products/_search
{
  "query": {
    "match": {
      "title.ngram": "jack"
    }
  }
}
```

5.4 D) Requête fuzzy (tolérance fautes)

```
GET ecommerce-products/_search
{
  "query": {
    "match": {
      "title": {
        "query": "jaket",
        "fuzziness": "AUTO"
      }
    }
  }
}
```

5.5 E) Série temporelle (commandes par minute)

```
GET ecommerce-orders/_search
{
  "size": 0,
  "aggs": {
    "orders_over_time": {
      "date_histogram": {
        "field": "@timestamp",
        "calendar_interval": "minute"}}}}
```

6 Visualisation Kibana

Nous avons créé des visualisations pertinentes :

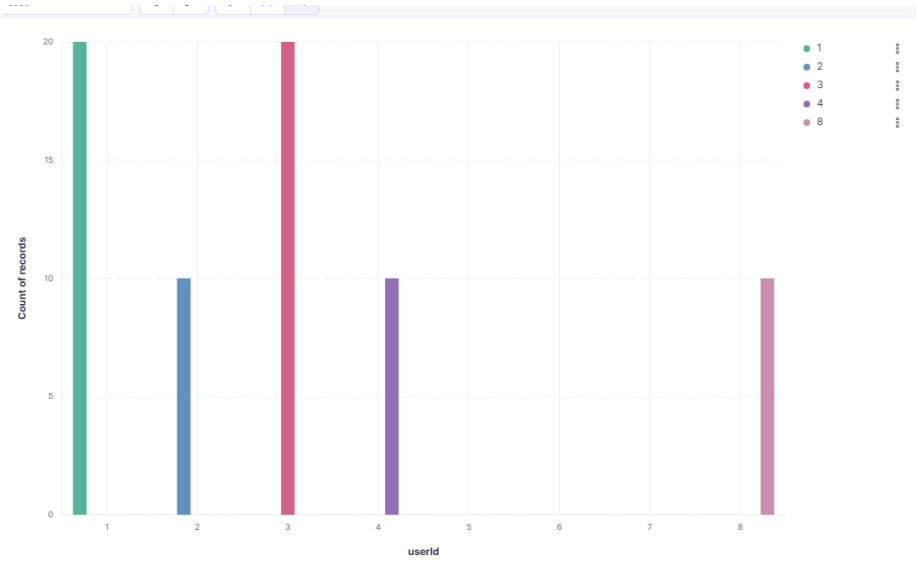

FIGURE 2 – Orders by user (Top users)

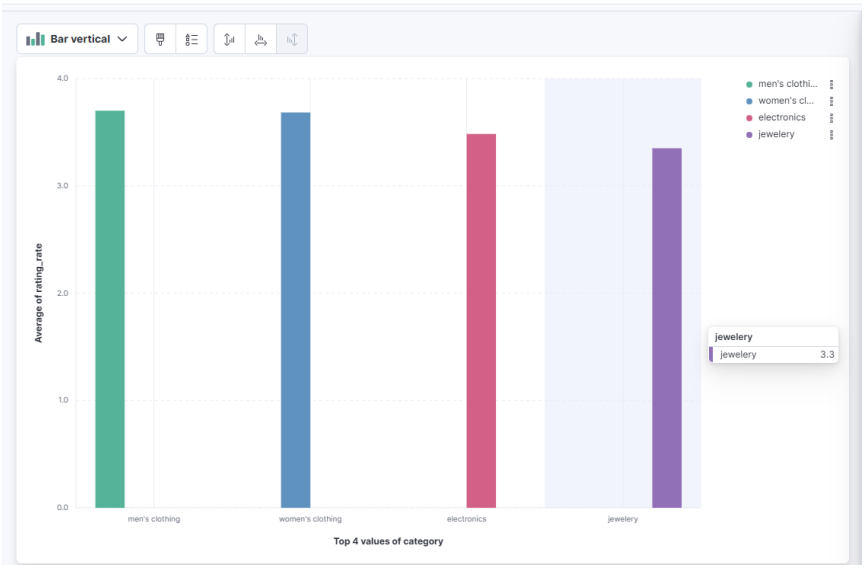

FIGURE 3 – Avg Rating per Category

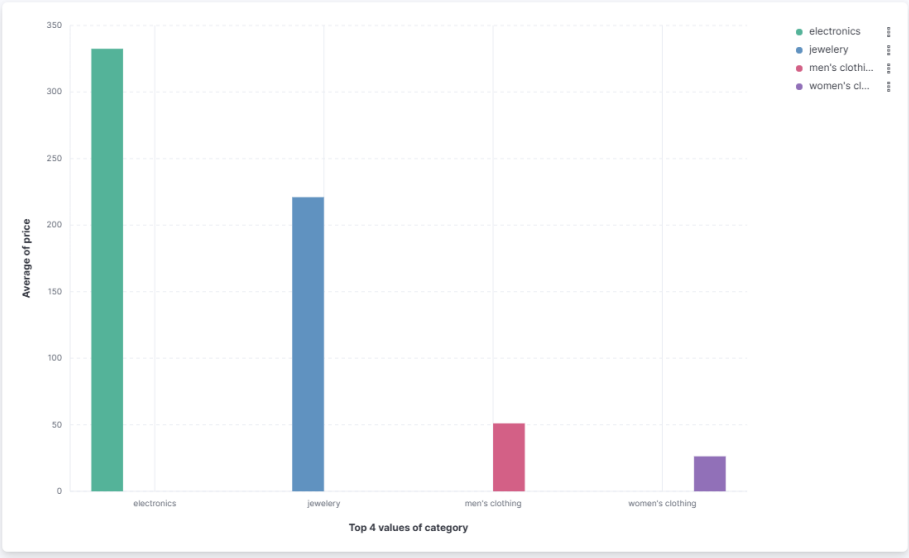

FIGURE 4 – Avg Price per Category

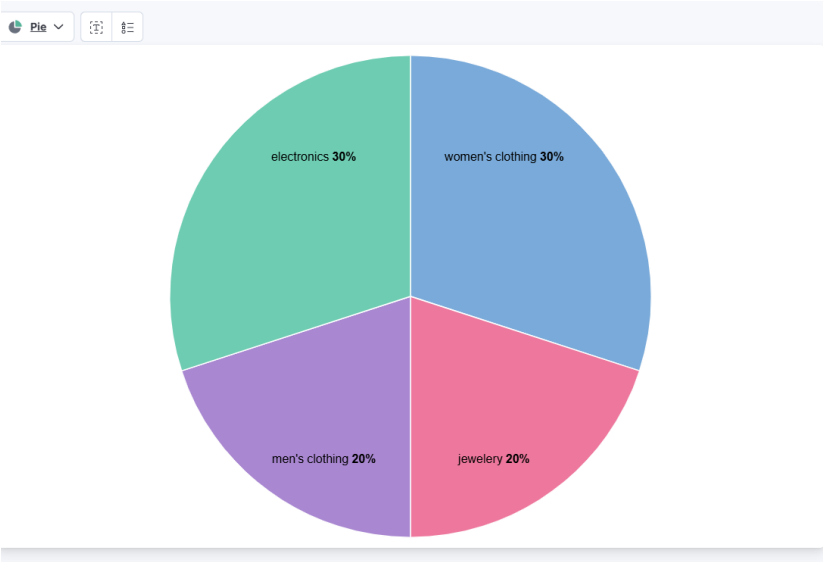

FIGURE 5 – Products by Category

FIGURE 6 – Avg items per order

7 Traitement distribué Spark

Conformément à l'énoncé, nous utilisons **Spark** pour charger les données depuis Elasticsearch et calculer des agrégats distribués (moyenne, somme, etc.). Exemples de résultats exportés :

- `avg_items_per_order.csv`
- `revenue_by_category.csv`
- `top_products.csv`
- `totals.csv`
- `summary.txt`

	category	estimated_revenue	total_qty	lines
1	men's clothing	264643.9945220947	3100	700
2	jewelery	141097.99995422363	400	300
3	electronics	62400.0	600	300
4	women's clothing	985.0000381469727	100	100

FIGURE 7 – Revenue by category

8 Organisation, exécution et livrables fournis

Nous fournissons :

- fichiers `docker-compose` + configs Kafka/Logstash/Elasticsearch/Kibana/Spark,
- scripts Python (producer, traitement Spark),
- mappings Elasticsearch,
- requêtes Elasticsearch,
- exports Spark (CSV/JSON),
- captures Kibana (Discover + Visualisations),
- dépôt GitHub : *https://github.com/ZuxxLo/projet-ivdm-ecommerce-data-pipeline*