

Elastic & food fact checking

Géraud Dugé de Bernonville

02/12/2022

Outline

- 1 Contexte
- 2 Les outils
- 3 Entraînement
- 4 Produit final
- 5 Conclusion

Qualité des aliments & sécurité sanitaire

- Vache folle
- Grippe aviaire
- Perturbateurs endocriniens (pesticides, plastiques et autres substances chimiques...)
- OGM
- Allergènes (gluten, crustacés, oeufs, arachides, soja, ...)
- Cancérogènes (E171 - oxyde de titane ?)

Questions :

- Où trouve-t-on ces éléments ?
- Quelles catégories de produit sont les plus concernées ?
- Quelles marques ?

Mais surtout... Y a t'il du E171 dans la bière ?

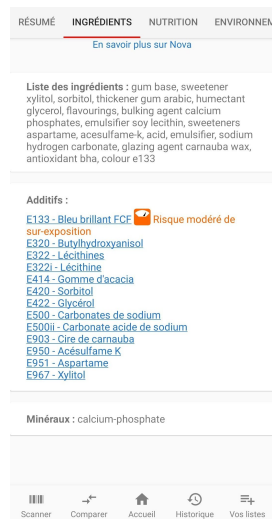
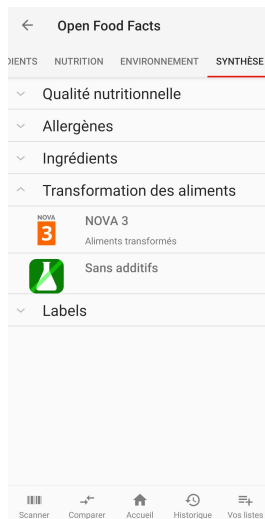
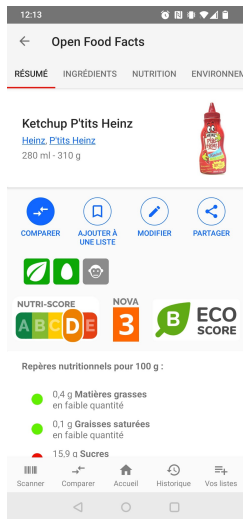


Open Food Facts

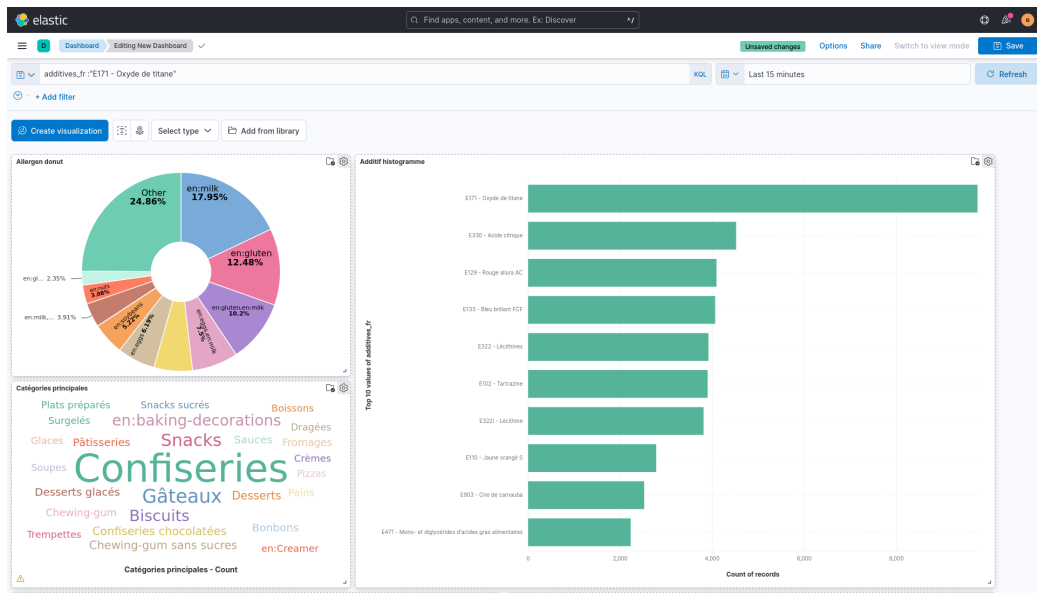


Base de données sur les produits alimentaires faite par tout le monde, pour tout le monde.

Open Food Facts - Mobile



Ce qu'on aimerait avoir



Elastic Stack



- Moteur de recherche
- Analyse et stockage de données



- Ingestion des données



- Visualisation

Topo Elasticsearch

Document JSON

```
{  
  "name": "Chips au vinaigre",  
  "category" : "apero",  
  "lipides" : 20,  
  "glucides" : 10,  
  "proteines" : 5  
}
```

API REST

<GET|POST|PUT|DELETE>

http[s]://<hostname>:<port>/[<index>]/[<type>]/[_<keyword>]]

- index
- type: _doc
- _keyword : _search, _mapping,...

Installation

Pré-requis

- Docker et plugin docker compose

Version 8.x

- 1 `cd elk-compose`
- 2 Lancer la stack: `docker compose up -d`
- 3 Ouvrir `http://localhost:5601` (elastic / elastic_password)
- 4 Aller dans Dev Tools (`http://localhost:5601/app/kibana#/dev_tools/console`)

Jouons avec Elasticsearch

Indexer un document

```
POST /store/_doc
{
  "name": "Chips au vinaigre",
  "category" : "apero",
  "lipides" : 20,
  "glucides" : 10,
  "proteines" : 5
}
```

```
POST /store/_doc
{
  "name": "Langues piquantes",
  "category" : "confiserie",
  "lipides" : 0,
  "glucides" : 90,
  "proteines" : 5
}
```

Requêter

```
GET /store/_search
```

```
GET /store/_search?q=langues
```

```
GET /store/_search
{
  "query": {
    "match": {
      "name": "langues"
    }
  }
}
```

Topo Logstash

Lancement

```
docker compose up logstash
```

Fichier conf

```
input { ... }  
filter { ... }  
output { ... }
```

Jouons avec Logstash - Données de test

- 1 Récupérer le fichier CSV `sample-fr.openfoodfacts.org.products.csv`
- 2 Vérifier le fichier `file-input.conf` dans le répertoire `pipelines/student`

```
input {  
  file {  
    path => "/data/*.csv"  
    mode => "read"  
    file_completed_action => "log"  
    file_completed_log_path => "/data/input.log"  
    exit_after_read => true  
  }  
}
```

- 3 Vérifier le fichier `debug-output.conf`

```
output {  
  stdout { codec => "rubydebug" }  
}
```

- 4 Lancer logstash

```
docker compose up --force-recreate logstash
```

- 5 Patienter...

Ajout du filtre CSV

- 1 Vérifier le fichier `filter.conf` dans le répertoire `pipelines/student`

```
filter {  
  mutate { gsub => [ "message", "\"", "'" ] }  
  csv {  
    separator => "\t"  
    autogenerate_column_names => false  
    autodetect_column_names => true  
    skip_header => true  
    skip_empty_columns => true  
  }  
}
```

- 2 Mettre à jour la propriété `path.config` du fichier `pipelines/student/pipelines.yml`
`path.config: "/pipelines/student/{file-input,debug-output,filter}.conf"`
- 3 Supprimer le fichier `since_db`
- 4 Relancer `logstash`
`docker compose up --force-recreate logstash`

Ajout de la sortie Elasticsearch

1 Vérifier le fichier elastic-output.conf

```
output {  
  elasticsearch {  
    hosts => [ "https://es01:9200" ]  
    ssl => true  
    cacert => "config/certs/ca/ca.crt"  
    api_key => "${LOGSTASH_FOODFACTS_API_KEY}"  
    index => "openfoodfacts"  
    template => "/pipelines/student/openfoodfacts.template.json"  
    template_name => "openfoodfacts"  
    template_overwrite => true  
    data_stream => false  
  }  
}
```

2 Mettre à jour la propriété path.config du fichier pipelines/student/pipelines.yml

path.config: "/pipelines/student/{file-input,debug-output,filter,elastic-output}.conf"

3 Relancer logstash

```
docker compose up --force-recreate logstash
```

Dans Kibana > Dev Tools

```
GET /openfoodfacts/_search
```

```
GET /openfoodfacts/_search?q=Snacks
```

Query time !

Nombre de catégories:

```
GET /openfoodfacts/_search
```

```
{  
  "aggs": {  
    "categories_count": {  
      "value_count": {  
        "field": "main_category"  
      }  
    }  
  }  
}
```


Query time !

Répartition des additifs par catégories:

GET /openfoodfacts/_search

```
{
  "aggs": {
    "par_categorie": {
      "terms": {
        "field": "main_category_fr",
        "size": 10
      },
      "aggs": {
        "par_additif": {
          "terms": {
            "field": "additives_fr"
          }
        }
      }
    }
  }
}
```

Problème de taille

Index management

Update your Elasticsearch indices individually or in bulk.

☐ × Include system indices

[Reload indices](#)

| <input type="checkbox"/> Na... | Health | Status | Primaries | Replicas | Docs count | Storage si... |
|--|----------|--------|-----------|----------|------------|---------------|
| <input type="checkbox"/> store | ● yellow | open | 5 | 1 | 2 | 10.5kb |
| <input type="checkbox"/> openfoodfacts | ● yellow | open | 5 | 1 | 100 | 1.4mb |

Rows per page: 10 ▾

Configuration du mapping

```
DELETE openfoodfacts
```

```
PUT /openfoodfacts
```

```
{
  "settings" : {
    "number_of_shards": 3,
    "number_of_replicas": 0
  },
  "mappings": {
    "dynamic_templates": [
      {
        "strings": {
          "match_mapping_type": "string",
          "mapping": {
            "type": "keyword"
          }
        }
      ]
    }
  }
}
```

Jouons avec Kibana

Navigation dans les données

- ❶ Configurer la **data view**, dans **Stack Management**, puis **Data Views**
 - ❶ Renseigner openfoodfacts pour le nom
 - ❷ Sélectionner — I don't want to use the time filter —
- ❷ Accéder à l'onglet **Discover**
- ❸ Sélectionner les champs `additives_fr`, `main_category_fr`,...

Kibana - Première visualisation

Nuage des principales catégories

- ➊ Accéder à l'onglet **Visualize**
- ➋ Sélectionner **Aggregation based** puis **Tag Cloud**
- ➌ Configurer un bucket **Tags**
 - Aggregation = Terms
 - Field = `main_category_fr`
 - Size = 50
 - Custom Label = Catégories principales
- ➍ Sauvegarder le widget

Kibana - Suite

Tableau des marques

- ❶ Sélectionner **Lens**
- ❷ Glisser le champ `brands` vers la visu
- ❸ Sélectionner le type **Table**
- ❹ Cliquer sur `=Top 5 values of brands*`
 - Number of values = 20
- ❺ Sauvegarder

Kibana - Mmmmm Donut

Donut des allergènes

- 1 Sélectionner **Lens**
- 2 Glisser le champ `allergens` vers la visu
- 3 Sélectionner le type **Donut**
- 4 Cliquer sur `=Top 5 values of brands`
 - Number of values = 10
- 5 Sauvegarder

Kibana - Fin (?)

Histogramme des additifs

- 1 Sélectionner Bar horizontal
- 2 À vous de jouer...

Tag cloud des produits

On veut ça:



Dashboard

- 1 Ajouter tous les widgets dans un nouveau dashboard
- 2 Sauvegarder

Chargeons toute la base !

- L'objectif est de voir le résultat avec l'ensemble des données
- Pour éviter les doublons, on supprime l'index openfoodfacts
- Mettre à jour la propriété `path.config` du fichier `pipelines/student/pipelines.yml`
`path.config: "/pipelines/student/{file-input-full,debug-output,filter,elastic-output}.conf"`
- Relancer logstash
`docker compose up --force-recreate logstash`

Beer



Mission accomplie !

- Requêtes avec Elasticsearch
- Ingestion de données avec Logstash
- Visualisation avec Kibana

Pour aller plus loin

- Fixer problèmes d'import
 - Champs trop longs
 - Encodage
 - Guillemets mal positionnés
- Découper les champs, par exemple :
 - E330 - Acide citrique,E150c - Caramel ammoniacal,E300 - Acide ascorbique
 - Frais,Produits laitiers,Desserts,Fromages,Fromages blancs,Fromages-blancs-aromatisés
- Configurer l'analyseur pour utiliser la langue française
- Utiliser les informations de géolocalisation

Merci

?

Questions