Compte rendu TP Groupe

MongoDB / Elasticsearch

Selebran Axel, VROMMAN thomas

Préambule :

Ce TP a pour but de récupérer des données d'une API afin de les mettre dans une base MongoDB et les utiliser avec Elasticsearch.

Choix des technos :

Nous avons choisi de prendre des données issues de l'API Open Data Paris - Vélibréférences, cette API référence tous les vélibs de la région parisienne avec leurs coordonnées géographiques en temps réel.

Pour la base de données, nous avons utilisés MongoDB car elle nous était imposée, ainsi qu'Elasticsearch pour le traitement des données.

Pour la visualisation finale de nos données nous avons utilisé l'interface graphique de Kibana, cette dernière nous permet de mettre en valeurs nos données en créant des graphiques sous différentes formes, pour présenter nos données sous plusieurs angles.

Réalisation :

La première étape du projet était de créer une base de données MongoDB pour qu'elle puisse stocker les données provenant de l'API.

```python
def get_data(offset: int) -> Optional[List[Dict[str, Any]]]:
    try:
        r = requests.get(f"https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/velib-disponibilite-en-temps-reel/records?limit=100&offset={offset}")
        r.raise_for_status()
        jsonResponse = r.json()
        return jsonResponse.get("results")
    except HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
    except Exception as err:
        print(f'Other error occurred: {err}')
    return None

def conn() -> tuple[Collection, str]:
    CONNECTION_STRING = "mongodb://localhost:27017"
    client = MongoClient(CONNECTION_STRING)
    current_GMT = time.gmtime()
    ts = calendar.timegm(current_GMT)
    db = client["Velib"]
    collection_name = f"velo_libre_{ts}"
    collection = db[collection_name]
    return collection, collection_name

def set_data(client: Collection, data: List[Dict[str, Any]]) -> None:
    try:
        client.insert_many(data)
    except Exception as err:
        print(f"Error: {err}")

def get_total_count() -> Optional[int]:
    try:
        r = requests.get(f"https://opendata.paris.fr/api/explore/v2.1/catalog/datasets/velib-disponibilite-en-temps-reel/records?limit=1&offset=0")
        r.raise_for_status()
        jsonResponse = r.json()
        return jsonResponse.get("total_count")
    except HTTPError as http_err:
        print(f'HTTP error occurred: {http_err}')
    except Exception as err:
        print(f'Other error occurred: {err}')
    return None
```

Ensuite créer une connexion entre notre base de données et une instance client Elasticsearch, ainsi que l'indexation des données pour les utiliser avec Elasticsearch.

```python
# Password for the 'elastic' user generated by Elasticsearch
ELASTIC_PASSWORD = "6mZByQl=UvxRGUty1I4i"

# Create the client instance
client = Elasticsearch(
    "http://localhost:9200",

)

# Successful response!
client.info()

docu = collName.find()

for doc in docu:
    doc_json = parse_json(doc)
    doc_json_id = doc_json.pop("_id")
    client.index(index="velib", id=doc_json_id, body=doc_json)

print("Indexation terminée.")
```

Une fois indexé, on retrouve toutes nos données prêtes à être utilisées dans l'interface Kibana.

velib-disponibilite-en-temps-reel

View and edit fields in **velib-disponibilite-en-temps-reel**. Field attributes, such as type and searchability, are based on field mappings in Elasticsearch.

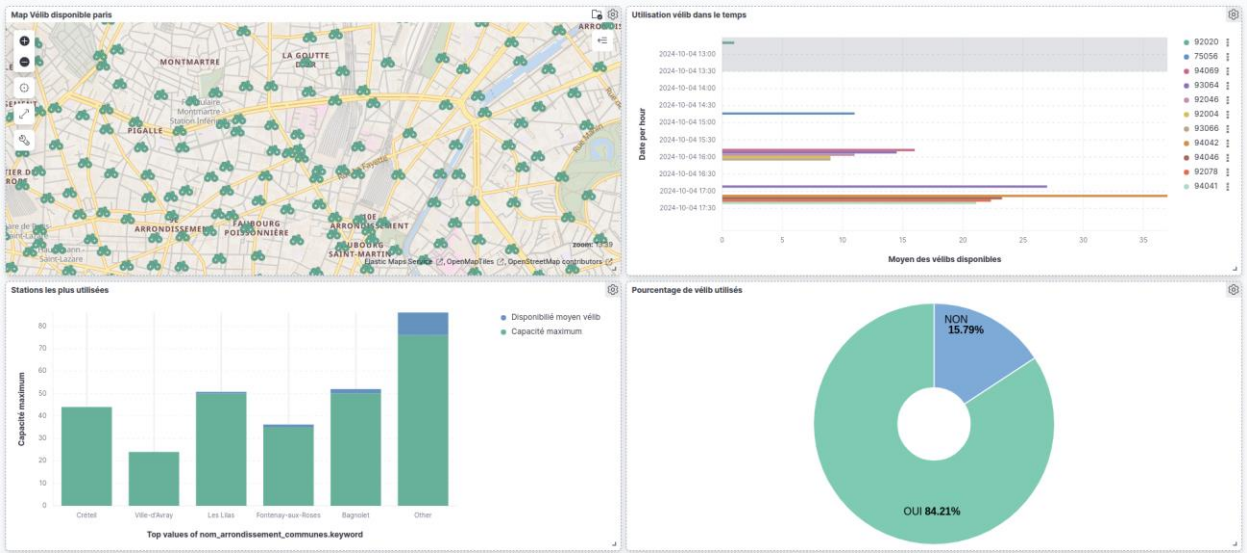
Fields (26) | Scripted fields (0) | Field filters (0)

Search

Name	Type	Format	Searchable	Aggregatable
_id	_id		●	●
_index	_index		●	●
_score				
_source	_source			
_type	_type		●	●
capacity	long		●	●
code_insee_commune	text		●	
code_insee_commune.keyword	keyword		●	●
coordinates	geo_point		●	●
duedate	date		●	●

Rows per page: 10

On peut par exemple faire différents graphiques pour mettre en valeurs nos données et les présenter sous différents angles.



Pour plus de technique je vous invite donc à regarder notre Code.