

ELK -Scraping et Analyse de sentiment

Realisé par : Karim Chidekh

I. Installation de la Stack ELK

- **Mise en Place de la VM Ubuntu sur Contabo :**

La première étape a consisté à choisir et configurer un plan de machine virtuelle (VM) sur Contabo. Lors de la configuration de la VM, il a été vérifié que la VM disposait d'une adresse IP publique, une exigence clé pour l'accessibilité externe.

- **Configuration Réseau avec Contabo :**

Le panneau de gestion Contabo offrait des outils pour configurer les paramètres réseau. Les actions suivantes ont été effectuées :

- **Redirection de Port :**

La redirection de port a été configurée pour diriger le trafic entrant depuis l'adresse IP publique vers la VM où les composants ELK devaient être installés.

- **Règles de Groupe de Sécurité :**

Les règles du groupe de sécurité dans le panneau de gestion Contabo ont été modifiées pour autoriser le trafic entrant sur le port 5601, qui est le port par défaut de Kibana.

- **Règles de Pare-feu sur la VM :**

Le pare-feu sur la VM elle-même a été configuré pour autoriser le trafic entrant sur le port 5601. Les commandes spécifiques dépendaient de la distribution Linux utilisée. Par exemple, sur Ubuntu avec ufw, la commande suivante a été utilisée pour autoriser le trafic sur le port 5601 :

```
shell

sudo ufw allow 5601/tcp
```

Figure 1 : Port 5601

II. Installation des Composants ELK :

La Stack ELK a été installée comme suit :

Installation d'Elasticsearch :

Elasticsearch, le moteur central de stockage et de recherche de données, a été installé à l'aide de commandes de gestion de paquets standard. Par exemple, sur Ubuntu :

```
shell

sudo apt update
sudo apt install -y elasticsearch
```

Figure 2: Elasticsearch installation

Installation de Logstash :

Logstash, responsable de la collecte, du traitement et de l'envoi des journaux, a été installé avec la commande suivante :

```
shell

sudo apt install -y logstash
```

Figure 3: Logstash installation

Installation de Kibana :

Kibana, l'outil de visualisation et d'analyse, a été installé avec des commandes similaires :

```
shell

sudo apt install -y kibana
```

Figure 4: Kibana installation

III. Accès à Kibana

Configuration de Kibana :

Le fichier de configuration de Kibana, généralement situé à `/etc/kibana/kibana.yml`, a été modifié pour permettre l'accessibilité externe. Plus précisément, la ligne suivante a été ajoutée ou modifiée pour permettre à Kibana d'écouter sur toutes les interfaces réseau :

```
yaml
server.host: "0.0.0.0"
```

Redémarrage de Kibana :

Après les modifications de configuration, le service Kibana a été redémarré pour appliquer les modifications :

```
shell
sudo service kibana restart
```

Accès à Kibana :

Suite à ces étapes, Kibana a été rendu accessible via un navigateur web en accédant à l'adresse IP publique de la VM à l'URL fournie : <http://173.212.251.245:5601/>

Les utilisateurs ont pu interagir avec l'interface web de Kibana et accéder à la Stack ELK pour l'analyse des journaux.

Configuration de l'Authentification d'ELK :

Pour renforcer la sécurité, l'authentification a été ajoutée pour l'accès à Elasticsearch et Logstash en modifiant le fichier de configuration de ces composants. Les lignes suivantes ont été ajoutées pour spécifier le nom d'utilisateur et le mot de passe pour l'authentification :

```
yaml
elasticsearch.username: "elastic"
elasticsearch.password: "uB9x2YBTaBeltNqc-+HI"
```

Copier password « uB9x2YBTaBeltNqc-+HI »

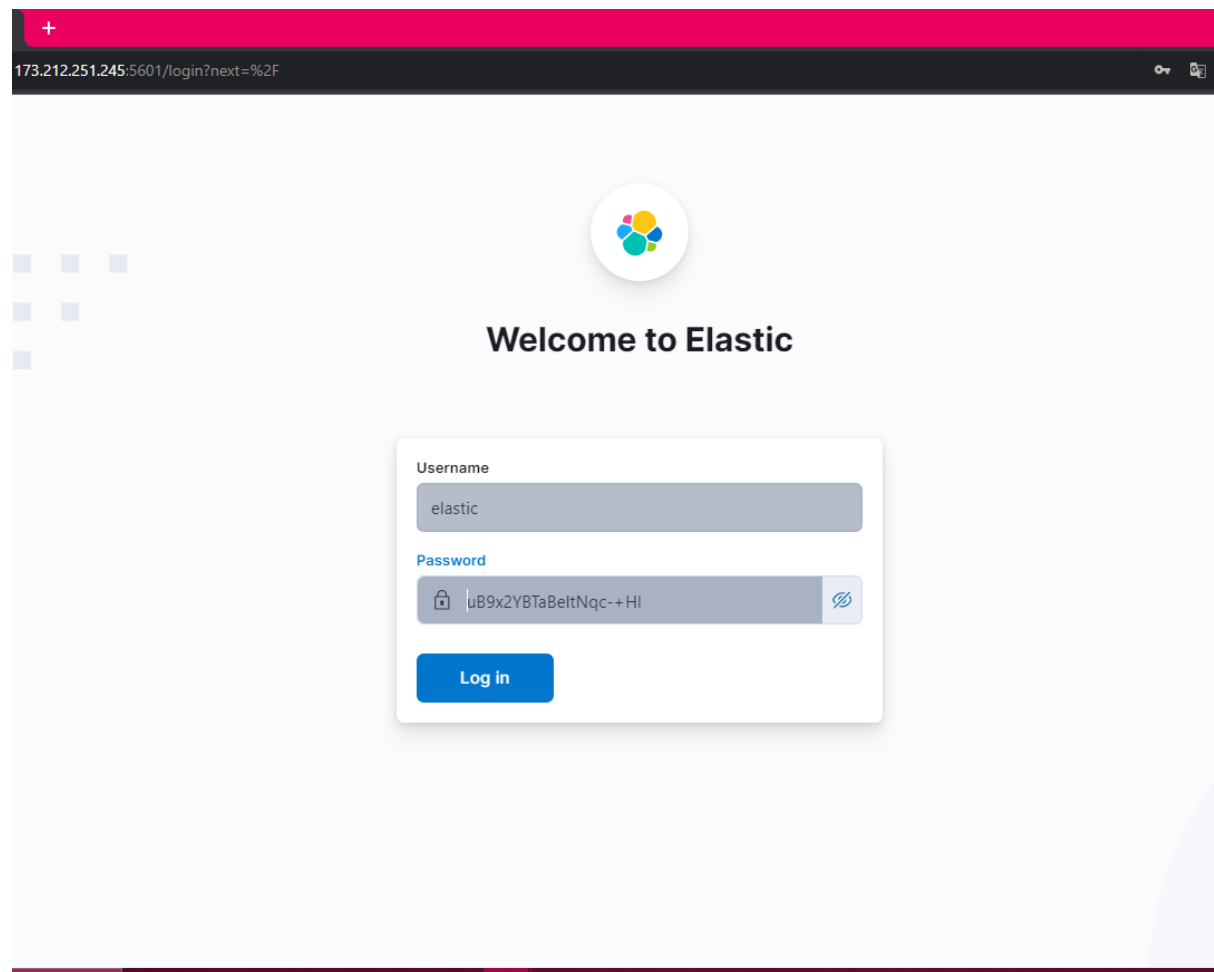
Sauvegarde de la Configuration et Redémarrage d'ELK :

Les modifications ont été sauvegardées dans les fichiers de configuration appropriés, et les services Elasticsearch et Logstash ont été redémarrés pour appliquer les paramètres d'authentification.

```
shell

sudo service elasticsearch restart
```

Authentification avec Username et Password.



ELK Home Plateforme sur une adresse IP public

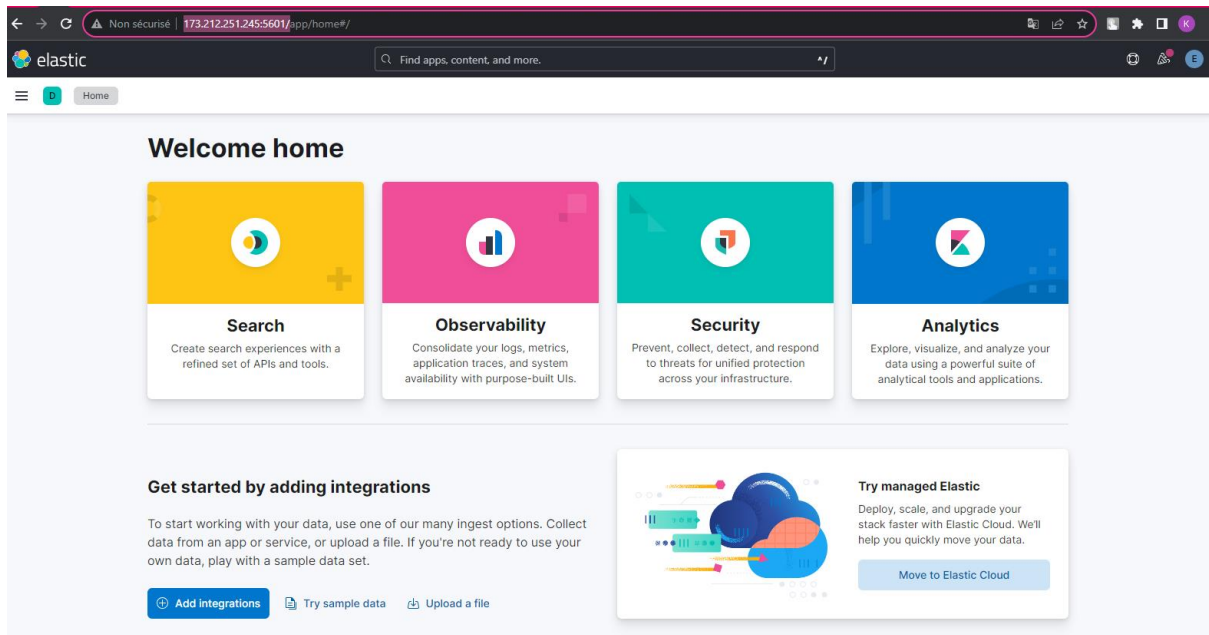


Figure : ELK Home

IIII. Changement de Logo Kibana

Localisation du Logo Kibana :

La première étape consiste à trouver l'emplacement du fichier du logo Kibana existant. Pour cela, j'ai exécuté la commande suivante pour rechercher le fichier logo_kibana.svg :

```
shell
find /usr/share/kibana -type f -name '*kibana.svg'
```

Remplacement du Logo :

Après avoir créé une copie de sauvegarde, j'ai remplacé le logo Kibana existant par le nouveau logo de mon choix. La commande suivante a été utilisée pour effectuer le remplacement, en remplaçant /path/to/your/new_logo.svg par le chemin du nouveau logo :

```
shell
sudo cp /Karim/images/simple-octo.svg /usr/share/kibana/node_modules
```

IIII. Intégration d'une Fonctionnalité de Classification de Sentiment en Anglais dans ELK, FastAPI

🔧 Installation de TextBlob et FastAPI :

La première étape consistait à installer les bibliothèques TextBlob et FastAPI pour pouvoir intégrer une fonctionnalité de classification de sentiment dans ELK. J'ai utilisé des commandes de gestion de paquets Python pour installer ces bibliothèques.

shell

```
pip install textblob  
pip install fastapi
```

🔧 Création d'un Service d'analyse de Sentiment :

Ensuite, j'ai créé un service de classification de sentiment en utilisant FastAPI. J'ai développé une API simple qui accepte du texte en entrée et renvoie la classification de sentiment associée.

python

```
from fastapi import FastAPI  
from textblob import TextBlob  
  
app = FastAPI()  
  
@app.post("/classify-sentiment")  
def classify_sentiment(text: str):  
    analysis = TextBlob(text)  
    sentiment = analysis.sentiment.polarity  
    if sentiment > 0:  
        sentiment_label = "Positif"  
    elif sentiment < 0:  
        sentiment_label = "Négatif"  
    else:  
        sentiment_label = "Neutre"  
    return {"sentiment": sentiment_label}
```

🚦 Intégration dans ELK :

Pour intégrer ce service de classification de sentiment dans ELK, j'ai utilisé Elasticsearch et Kibana. J'ai créé un index Elasticsearch pour stocker les données textuelles sur lesquelles la classification de sentiment serait appliquée. Les données, une fois stockées, étaient ensuite analysées en utilisant le service FastAPI que j'ai développé.

🚦 Affichage des Résultats dans Kibana :

J'ai utilisé Kibana pour créer des tableaux de bord et des visualisations qui affichent les résultats de la classification de sentiment. Les visualisations étaient configurées pour montrer la distribution des sentiments parmi les données textuelles stockées.

IIII. Collecte des Données, Stockage dans MongoDB et Pipeline d'Indexation vers Elasticsearch

🚦 Modélisation de la Solution proposé

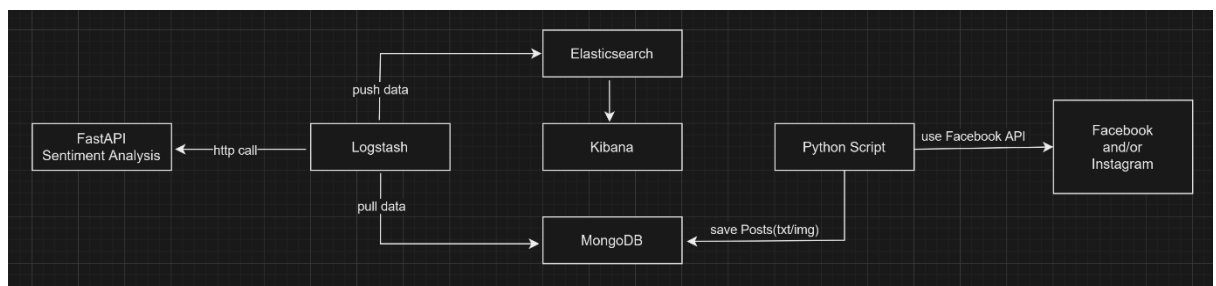


Figure Diagramme

🚦 Collecte des Données depuis Facebook, IMdb movies (Facebook Api, BeautifulSoup)

Pour collecter les publications, les images et les commentaires liés à un sujet donné depuis Facebook et Instagram, un script Python a été créé. Ce script a utilisé l'API Facebook pour Facebook et des techniques de scraping pour Instagram. Les données collectées ont été extraites et stockées dans une structure de données appropriée.

```

class FacebookClient:
    """A class for interacting with the Instagram API."""

    def __init__(self, access_token, date_range):
        """Initialize the Instagram post collector.

        Args:
            access_token: A Instagram access token.
            start_date: The start date to collect posts from.
            end_date: The end date to collect posts to.
        """

        self.graph = fbmeta.GraphAPI(access_token)
        self.start_date = date_range[0]
        self.end_date = date_range[1]

    def get_posts(self, subject):
        """Get a list of Instagram posts for a given subject.

        Args:
            subject: The subject to get posts for.

        Returns:
            A list of Instagram posts.
        """

        posts = []

```

Figure Facebook_Scraper

```

from bs4 import BeautifulSoup
import pandas as pd
import yaml
from pymongo import MongoClient

# The IMDb URL for the movie's reviews page
url = 'https://www.imdb.com/title/tt0111161/reviews'

# Function to scrape reviews
def scrape_imdb_reviews(url, num_reviews=1999):
    reviews = []
    page_num = 1

    while len(reviews) < num_reviews:
        response = requests.get(f"{url}?start={page_num * 10}")
        soup = BeautifulSoup(response.text, 'html.parser')
        review_containers = soup.find_all("div", class_="text show-more__control")

        if not review_containers:
            break

        for review in review_containers:
            text = review.get_text(strip=True)
            reviews.append(text)

        page_num += 1

    return reviews[:num_reviews]

```

Figure IMDb Crawler

🚦 Stockage dans MongoDB :

Pour gérer les données collectées de manière efficace, j'ai mis en place une base de données MongoDB. J'ai créé une classe MongoDBClient qui a permis de se connecter à la base de données MongoDB, de créer des collections si elles n'existaient pas déjà, de créer des index sur les champs pertinents, et d'insérer les données collectées dans ces collections. Les publications de Facebook et d'Instagram ont été stockées dans des collections distinctes.

```
class MongoDBClient:
    """A class for connecting to and interacting with MongoDB."""

    def __init__(self, host="localhost", port=27017, database="my_database", username='mongodb_connector',
                 password='mongodb_connector'):
        """Initialize the MongoDB client.

        Args:
            host: The host address of the MongoDB server.
            port: The port number of the MongoDB server.
            database: The name of the MongoDB database to connect to.
        """
        print(host)
        print(port)
        print(database)
        self.client = pymongo.MongoClient(host, port, username=username, password=password)
        self.db = self.client[database]

    def connect(self):
        """Connect to the MongoDB server."""
        self.client.connect()

    def disconnect(self):
        """Disconnect from the MongoDB server."""
        self.client.close()
```

Configuration yml :

```
mongodb:
  host: 173.212.251.245
  port: 27017
  database: db_post
  collection: coll_post

facebook:
  token: EAAZAGgdusflcB05CDRjXfs0ttKf0jnz0LNxr0NZATBDgIND8bg84BKfLj89MwuYZA3rR91mUWYm47qnT0A79oM84B40qApURR73s6EoLCNY3uohJQLwNaSZB8puXMZCqYXQZBdSNNK3H7
  base_url: https://graph.facebook.com/v18.0/

instagram:
  token: AAZAGgdusflcB01jLTDroqW2IX47Tg8xVhXnp70H0xALd7Tj0fc1ZBH81WhodJ9iNUV8PQkcFQRI0sd1ZBqMR9orfLYX0ZC3FjvZAY30aaPrTDfhakKcgWFEsKuTR0xLw4t4cB1AGxUfL9
  endpoint: https://graph.facebook.com/v13.0/search

subject:
  - décès Jacques Chirac'
  - cat
  - space

date-range:
  - 2023-01-01~2023-01-30
  - 2023-02-01~2023-02-30
  - 2023-03-01~2023-03-30
```

La prochaine étape a été de configurer un pipeline Logstash pour extraire les données depuis MongoDB et les pousser dans Elasticsearch, qui était hébergé sur une adresse IP publique. Le pipeline a été conçu pour gérer les données efficacement, en utilisant les plugins MongoDB pour extraire les données de MongoDB et le plugin Elasticsearch pour les indexer dans Elasticsearch.

```
root@vmi1007507: /etc/logstash/conf.d
input {
  mongodb {
    uri => 'mongodb://173.212.251.245/db_post?ssl=false'
    placeholder_db_dir => '/opt/logstash-mongodb/'
    placeholder_db_name => 'logstash_sqlite.db'
    collection => 'coll_post'
    batch_size => 5000
  }
}

filter {
  mutate {
    rename => {"_id" => "id_mongo"}
    remove_field => ["host", "@version", "log_entry"]
  }
}

http {
  verb => "GET"
  url => "http://localhost:8000/sentiment_analysis"
  body => {
    "text" => "%{[text]}"
  }
  body_format => "json"
  target_body => api_response
}

json {
  source => "api_response"
}

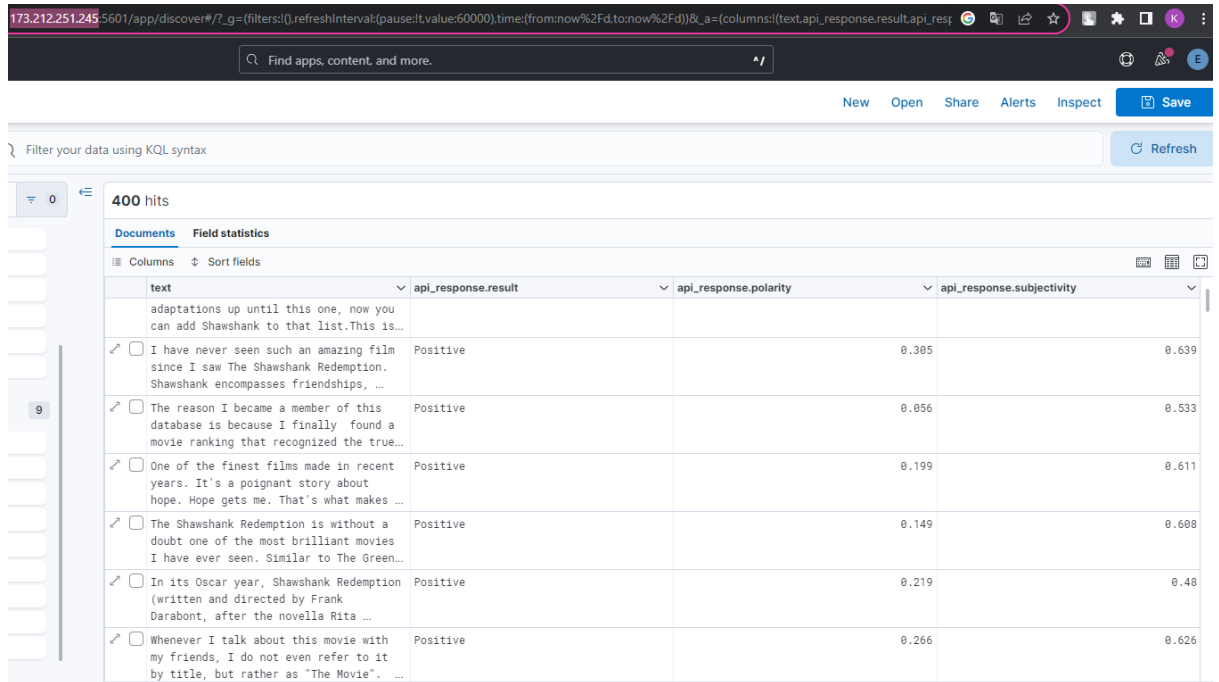
output {
  stdout {
    codec => rubydebug
  }
  elasticsearch {
    index => "idx_post-%{+YYYY.MM}"
    user => logstash
    password => logstash
  }
}
```

"mongodb-pipeline.conf" 38L, 816B

Figure Logstash Pipeline



Validation Réussie du Pipeline Logstash pour l'Indexation dans Elasticsearch depuis MongoDB et l'Appel à l'API de Classification de Sentiment.



Documents	Field statistics
Columns	Sort fields
text	api_response.result
api_response.polarity	api_response.subjectivity
adaptations up until this one, now you can add Shawshank to that list.This is...	
<input checked="" type="checkbox"/> I have never seen such an amazing film since I saw The Shawshank Redemption. Shawshank encompasses friendships, ...	Positive 0.305 0.639
<input checked="" type="checkbox"/> The reason I became a member of this database is because I finally found a movie ranking that recognized the true...	Positive 0.056 0.533
<input checked="" type="checkbox"/> One of the finest films made in recent years. It's a poignant story about hope. Hope gets me. That's what makes ...	Positive 0.199 0.611
<input checked="" type="checkbox"/> The Shawshank Redemption is without a doubt one of the most brilliant movies I have ever seen. Similar to The Green...	Positive 0.149 0.608
<input checked="" type="checkbox"/> In its Oscar year, Shawshank Redemption (written and directed by Frank Darabont, after the novella Rita ...	Positive 0.219 0.48
<input checked="" type="checkbox"/> Whenever I talk about this movie with my friends, I do not even refer to it by title, but rather as "The Movie". ...	Positive 0.266 0.626

Vous pouvez vérifier sur <http://173.212.251.245:5601/>

Username = elastic

Password = uB9x2YBTaBeltNqc-+HI