Integrating Elasticsearch with OpenAI for Document Indexing and Retrieval

---

Setting Up the Environment: Before we begin, ensure you have the necessary libraries installed. If not, you can install them using pip:

```
pip install elasticsearch
```

```
Requirement already satisfied: elasticsearch in /usr/local/lib/python3.10/dist-packages (8.14.0)
Requirement already satisfied: elastic-transport<9,>=8.13 in /usr/local/lib/python3.10/dist-packages (from elasticsearch) (8.13.1)
Requirement already satisfied: urllib3<3,>=1.26.2 in /usr/local/lib/python3.10/dist-packages (from elastic-transport<9,>=8.13->elasticse
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from elastic-transport<9,>=8.13->elasticsearch) (2024
```

```
pip install openai
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.35.10)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from openai) (1.7.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from openai) (0.27.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai) (2.8.0)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.4)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from openai) (4.12.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (3.7)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (1.2.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (2024.6.2)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->openai)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (0.7.
Requirement already satisfied: pydantic-core==2.20.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai) (2.20.
```

Importing Libraries: Let's start by importing the required libraries: Elasticsearch for managing our document store and OpenAI for utilizing its language API.

```
#Import libraries
from openai import OpenAI
from elasticsearch import Elasticsearch
```

Initializing Elasticsearch Client: To connect to Elasticsearch hosted on Elastic Cloud, initialize the Elasticsearch client with your cloud ID and credentials.

```
# Initialize Elasticsearch client for Elastic Cloud
es = Elasticsearch(
    cloud_id="Insert your Elasticsearch Cloud ID"
    basic_auth=("Insert the Elastic username", "Insert the password")
)
```

Initializing OpenAI API Client: Next, initialize the OpenAI API client using your API key.

```
# Initialize OpenAI API client with your API key
openai_api_key = 'Insert your OpenAI Api key'
openai = OpenAI(api_key=openai_api_key)
```

Assign Index name: Assign the desired name of the index you want on Elasticsearch to the variable index_name

```
# Index creation
index_name = 'documents'
```

Dene Elasticsearch Mappings:

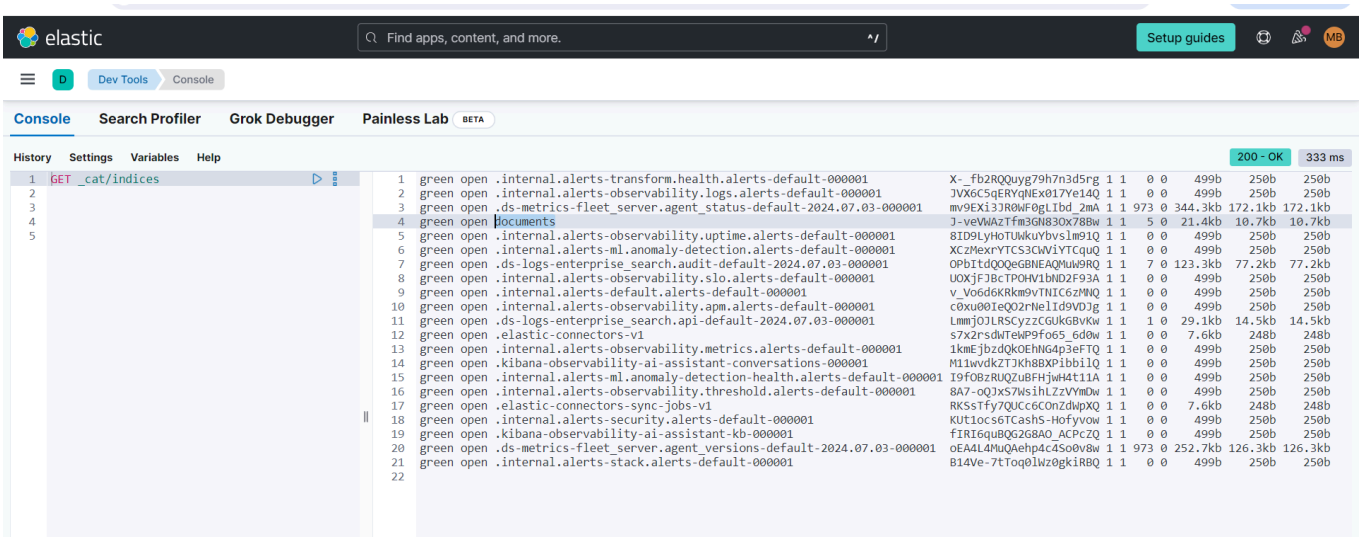
Mappings in Elasticsearch define the data types and settings for each field in an index. Let's define mappings for the documents index:

```
index_mappings = {
    "mappings": {
        "properties": {
            "content": {"type": "text"},
            "timestamp": {"type": "date"}
        }
    }
}
```

Create the Index if it does not exsist in Elasticsearch

```
# Create index (if not already exists)
if not es.indices.exists(index=index_name):
    es.indices.create(index=index_name)
```

Insert Structured Data into Elasticsearch: Now, let's insert structured data into the documents index with defined mappings. : We insert structured documents into the documents index. Each document includes a content field and a timestamp field formatted as a date string. This allows for organized and searchable document storage within Elasticsearch



As you can see it has created the index "documents" in Elasticsearch.

Inserting the data that we want to ingest in the index.

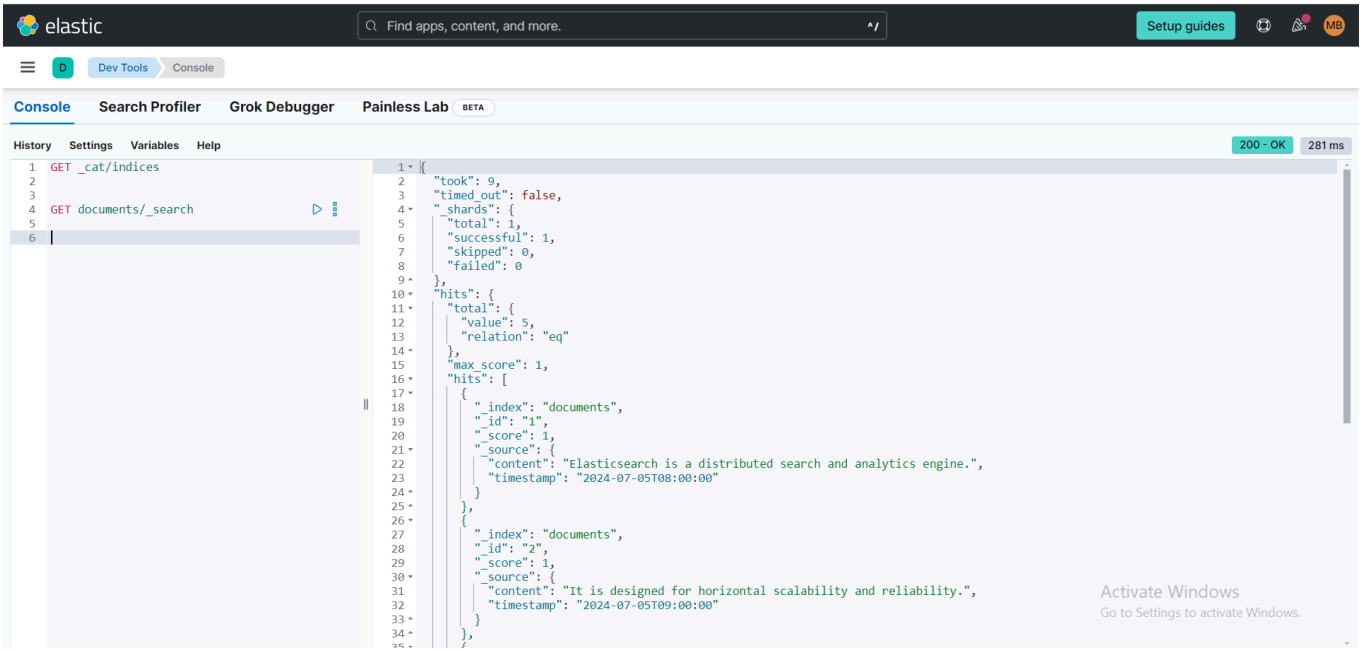
```
# structured_documents for indexing
structured_documents = [
    {"content": "Elasticsearch is a distributed search and analytics engine.", "timestamp": "2024-07-05T08:00:00"},
    {"content": "It is designed for horizontal scalability and reliability.", "timestamp": "2024-07-05T09:00:00"},
    {"content": "Elasticsearch is often used for log and event data analysis.", "timestamp": "2024-07-05T10:00:00"},
    {"content": "It supports full-text search, structured search, and analytics.", "timestamp": "2024-07-05T11:00:00"},
    {"content": "Elasticsearch integrates with Kibana for data visualization.", "timestamp": "2024-07-05T12:00:00"},
]
```

Indexing the structured documents into the index

```
# Index structured documents
for i, doc in enumerate(structured_documents):
    es.index(index=index_name, id=i+1, body=doc)

print(f"Index '{index_name}' created with mappings and structured documents indexed successfully.")
```

```
Index 'documents' created with mappings and structured documents indexed successfully.
```



Our documents have been indexed successfully into the index.

Document Retrieval: We perform a search for documents containing the term "Elasticsearch" in the content field. Elasticsearch's search capabilities allow for efficient retrieval of relevant documents based on specified criteria.

```
# Example search query
search_query = 'Elasticsearch'


# Define search parameters
search_params = {
    'query': {
        'match': {
            'content': search_query
        }
    }
}


# Perform search
search_results = es.search(index=index_name, body=search_params)


# Process and print search results
print(f"Documents matching query '{search_query}':")
for hit in search_results['hits']['hits']:
    print(f"Document ID: {hit['_id']}, Content: {hit['_source']['content']}")
```

```
Documents matching query 'Elasticsearch':
Document ID: 5, Content: Elasticsearch integrates with Kibana for data visualization.
Document ID: 1, Content: Elasticsearch is a distributed search and analytics engine.
Document ID: 3, Content: Elasticsearch is often used for log and event data analysis.
```

We've explored how to integrate Elasticsearch with OpenAI for document indexing, retrieval, and enhanced search capabilities. By defining Elasticsearch mappings, inserting structured data, and performing document retrieval operations, we've demonstrated a practical approach to building intelligent document management systems.

This setup lays a foundation for leveraging Elasticsearch's robust indexing and search features alongside OpenAI's language processing capabilities, enabling the development of sophisticated search and analysis applications.