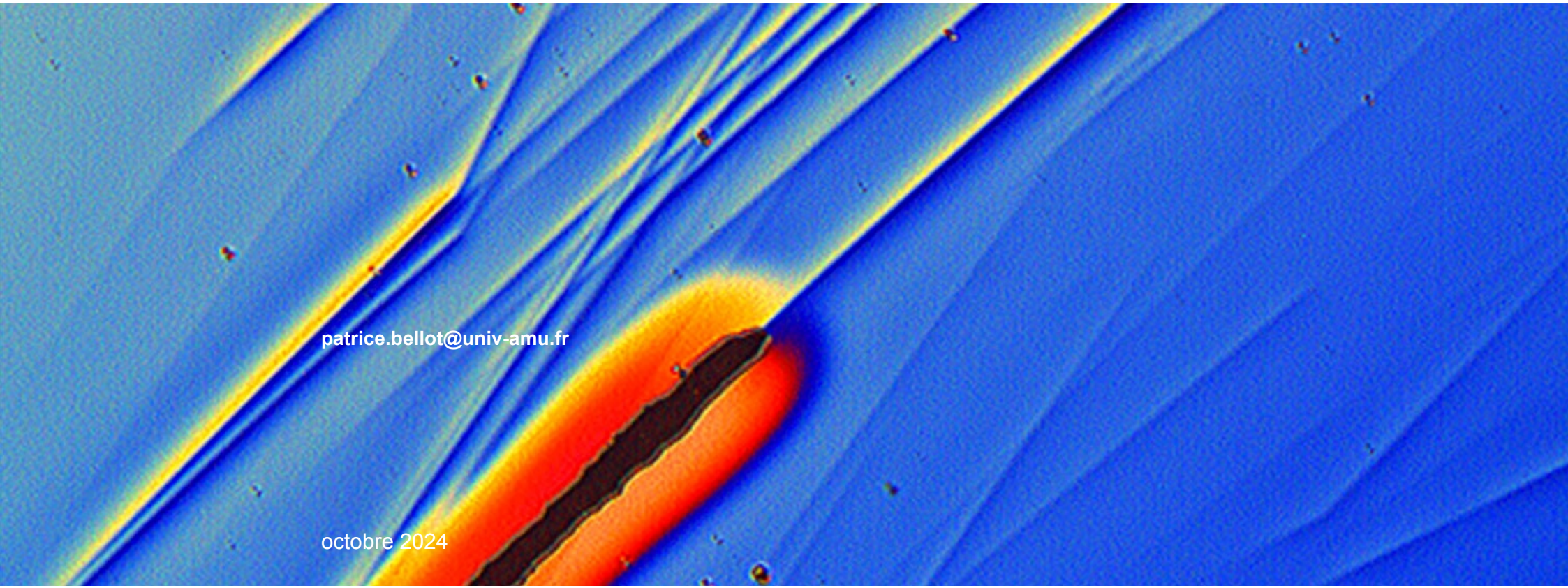
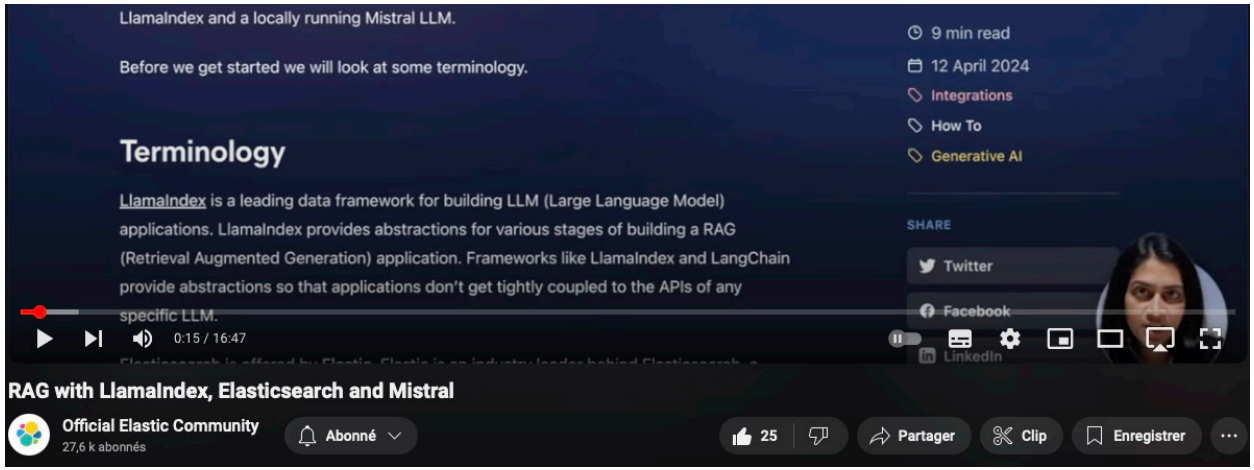
Retrieval Augmented Generation (RAG) avec ElasticSearch



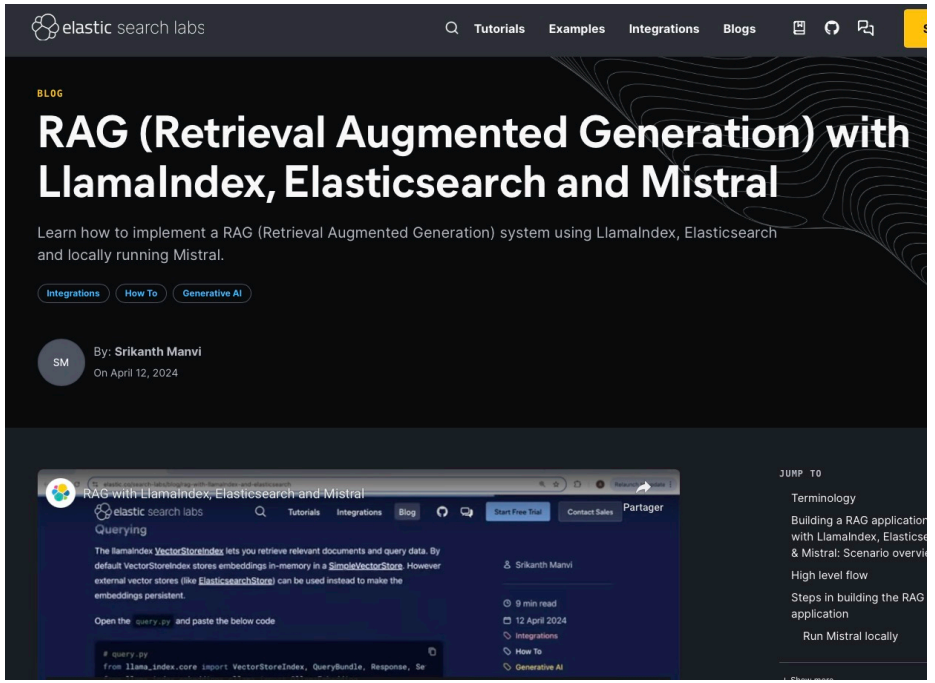
patrice.bellot@univ-amu.fr

octobre 2024

Démonstration basée sur :



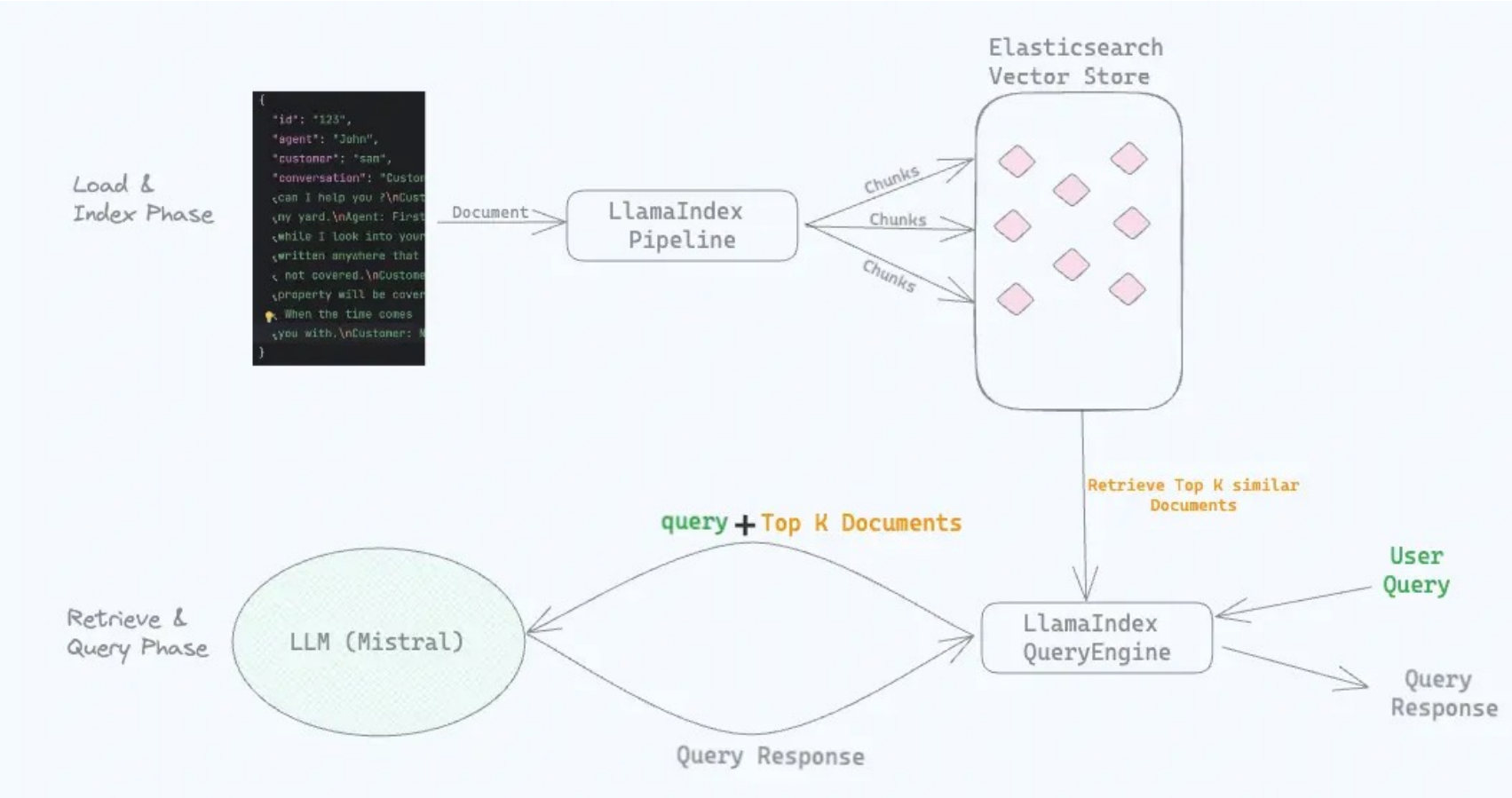
https://www.youtube.com/watch?v=5QofwkEel2I&t=808s



https://www.elastic.co/search-labs/blog/rag-with-llamaindex-and-elasticsearch

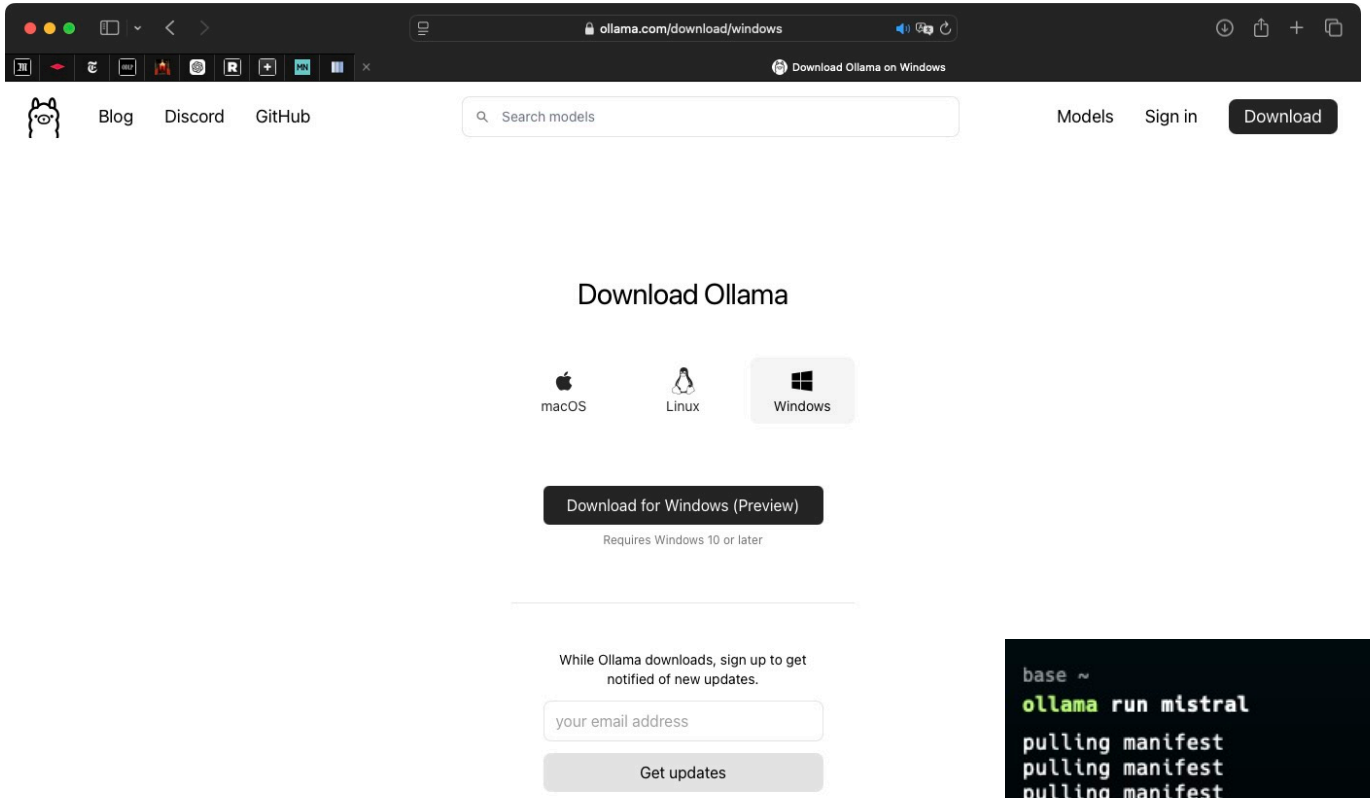
Démo avec LlamaIndex et Mistral via Ollama

https://www.elastic.co/search-labs/blog/rag-with-llamaIndex-and-elasticsearch



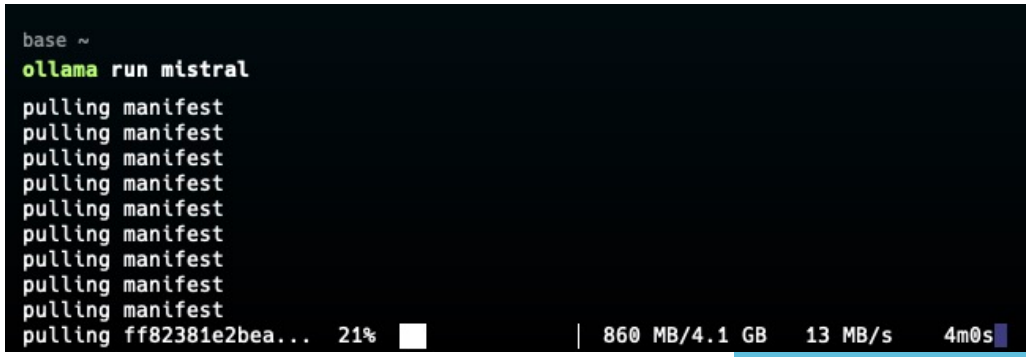
LLM local : Mistral via Ollama

https://ollama.com/download



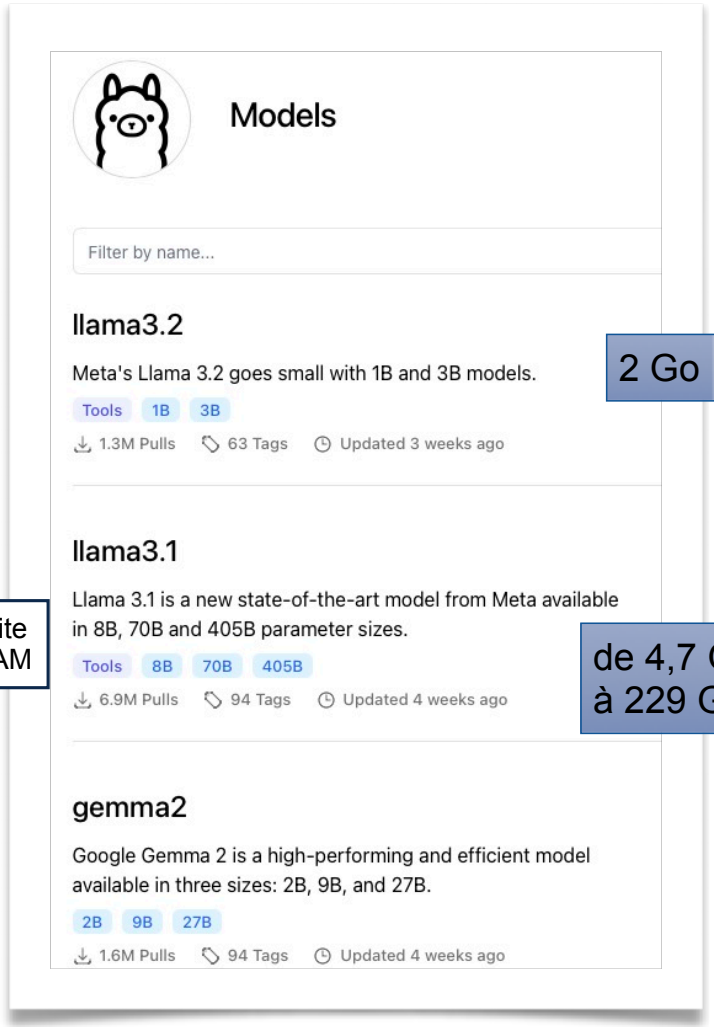
Ollama

MacOS et Linux (stables)
Window 10 et 11 (Preview)
LLM Mistral : 4 Go



LLM local : Mistral via Ollama

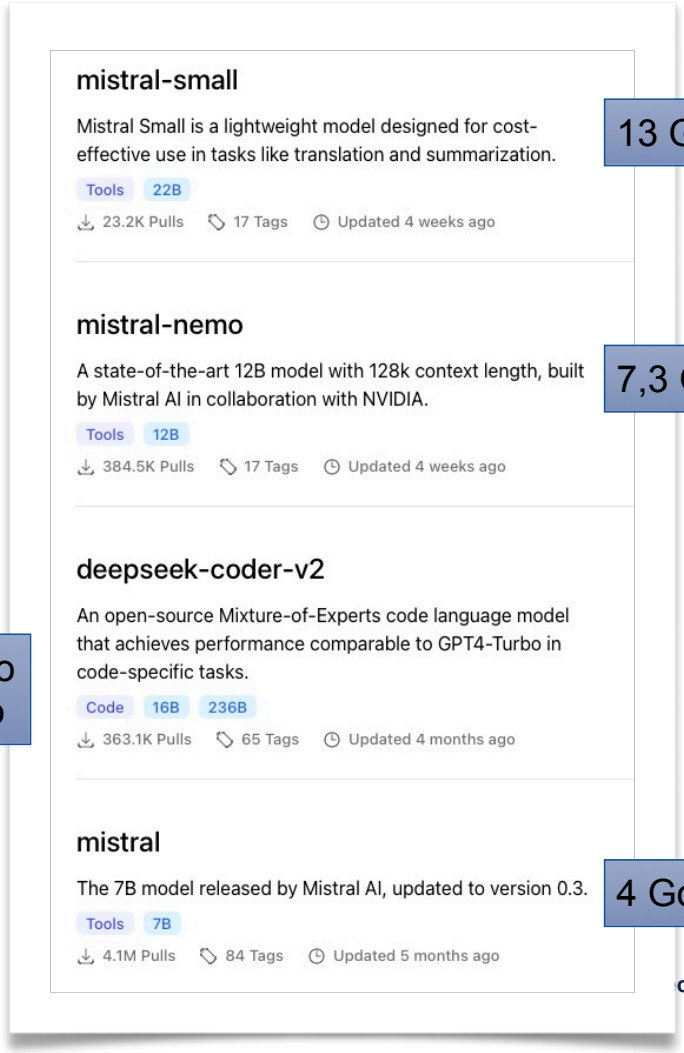
https://ollama.com/library



2 Go

8B nécessite 16 Go VRAM

de 4,7 Go à 229 Go



13 Go

7,3 Go

4 Go

Ollama

Nombreux LLMs disponibles

https://ai.meta.com/blog/meta-llama-3-1/

Performance des LLMs

Category / Benchmark	Llama 3.1 405B	Nemotron 4 340B Instruct	GPT-4 (0125)	GPT-4 Omni	Claude 3.5 Sonnet
General MMLU (0-shot, CoT)	88.6	78.7 (non-CoT)	85.4	88.7	88.3
MMLU PRO (5-shot, CoT)	73.3	62.7	64.8	74.0	77.0
IFEval	88.6	85.1	84.3	85.6	88.0
Code HumanEval (0-shot)	89.0	73.2	86.6	90.2	92.0
MBPP EvalPlus (base) (0-shot)	88.6	72.8	83.6	87.8	90.5
Math GSM8K (8-shot, CoT)	96.8	92.3 (0-shot)	94.2	96.1	96.4 (0-shot)
MATH (0-shot, CoT)	73.8	41.1	64.5	76.6	71.1
Reasoning ARC Challenge (0-shot)	96.9	94.6	96.4	96.7	96.7
GPQA (0-shot, CoT)	51.1	-	41.4	53.6	59.4
Tool use BFCL	88.5	86.5	88.3	80.5	90.2
Nexus	58.7	-	50.3	56.1	45.7
Long context ZeroSCROLLS/QuALITY	95.2	-	95.2	90.5	90.5
InfiniteBench/En.MC	83.4	-	72.1	82.5	-
NIH/Multi-needle	98.1	-	100.0	100.0	90.8
Multilingual Multilingual MGSM (0-shot)	91.6	-	85.9	90.5	91.6

Category / Benchmark	Llama 3.1 8B	Gemma 2 9B IT	Mistral 7B Instruct	Llama 3.1 70B	Mixtral 8x22B Instruct	GPT 3.5 Turbo
General MMLU (0-shot, CoT)	73.0	72.3 (5-shot, non-CoT)	60.5	86.0	79.9	69.8
MMLU PRO (5-shot, CoT)	48.3	-	36.9	66.4	56.3	49.2
IFEval	80.4	73.6	57.6	87.5	72.7	69.9
Code HumanEval (0-shot)	72.6	54.3	40.2	80.5	75.6	68.0
MBPP EvalPlus (base) (0-shot)	72.8	71.7	49.5	86.0	78.6	82.0
Math GSM8K (8-shot, CoT)	84.5	76.7	53.2	95.1	88.2	81.6
MATH (0-shot, CoT)	51.9	44.3	13.0	68.0	54.1	43.1
Reasoning ARC Challenge (0-shot)	83.4	87.6	74.2	94.8	88.7	83.7
GPQA (0-shot, CoT)	32.8	-	28.8	46.7	33.3	30.8
Tool use BFCL	76.1	-	60.4	84.8	-	85.9
Nexus	38.5	30.0	24.7	56.7	48.5	37.2
Long context ZeroSCROLLS/QuALITY	81.0	-	-	90.5	-	-
InfiniteBench/En.MC	65.1	-	-	78.2	-	-
NIH/Multi-needle	98.8	-	-	97.5	-	-
Multilingual Multilingual MGSM (0-shot)	68.9	53.2	29.9	86.9	71.1	51.4

LLM local : Mistral via Ollama

https://ollama.com/download

```
base ~
ollama run mistral

pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling manifest
pulling ff82381e2bea... 100%  4.1 GB
pulling 43070e2d4e53... 100%   11 KB
pulling 491dfa501e59... 100%  801 B
pulling ed11eda7790d... 100%   30 B
pulling 42347cd80dc8... 100%  485 B
verifying sha256 digest
writing manifest
success
>>> /?
Available Commands:
  /set            Set session variables
  /show           Show model information
  /load <model>   Load a session or model
  /save <model>   Save your current session
  /clear          Clear session context
  /bye            Exit
  /?, /help       Help for a command
  /? shortcuts    Help for keyboard shortcuts

Use """ to begin a multi-line message.

>>> Send a message (/? for help)
```

```
>>> /show info
  Model
        parameters              7.2B
        quantization            Q4_0
        arch                    llama
        context length          32768
        embedding length        4096

  Parameters
        stop    "[INST]"
        stop    "[/INST]"

  License
        Apache License
        Version 2.0, January 2004
```

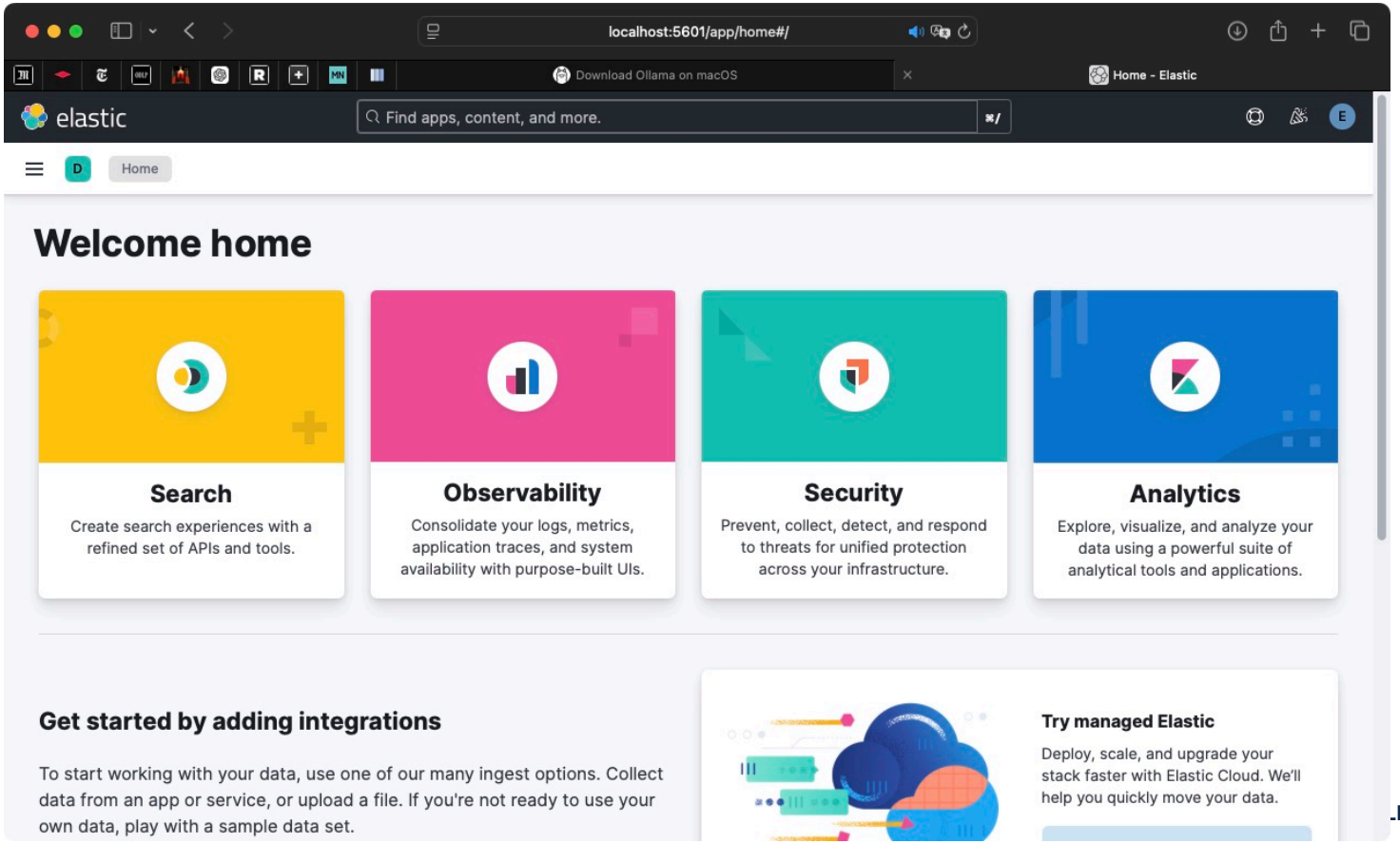
```
>>> Quelle est la capitale de la Belgique ?
 La capitale de la Belgique est Bruxelles.

>>> Send a message (/? for help)
```

Configuration

Démarrage ElasticSearch et Kibana

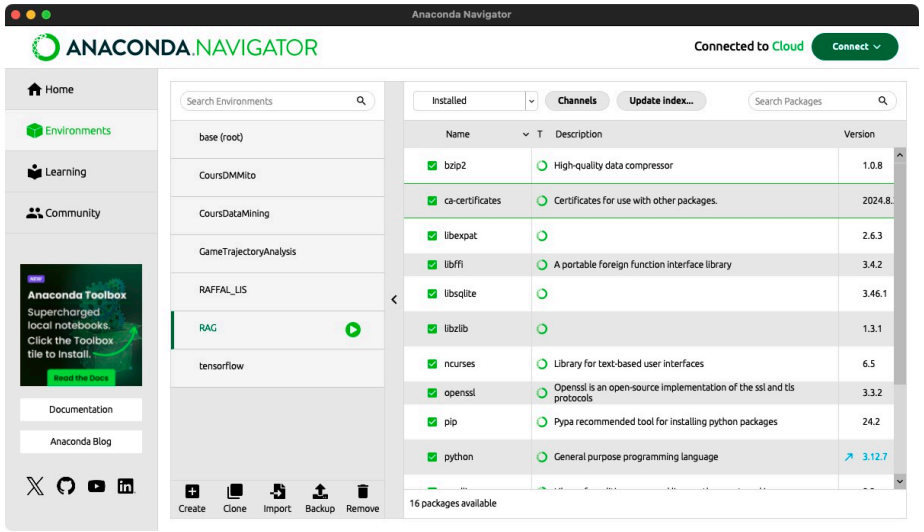
```
base ~/Applications/Elastic/elasticsearch-8.15.2
bin/elasticsearch
```

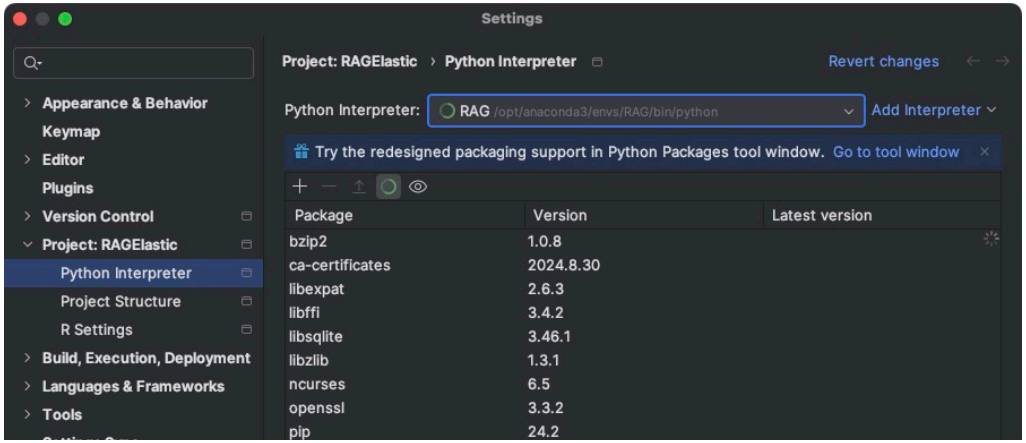
```
base ~/Applications/Elastic/kibana-8.15.2
bin/kibana
```

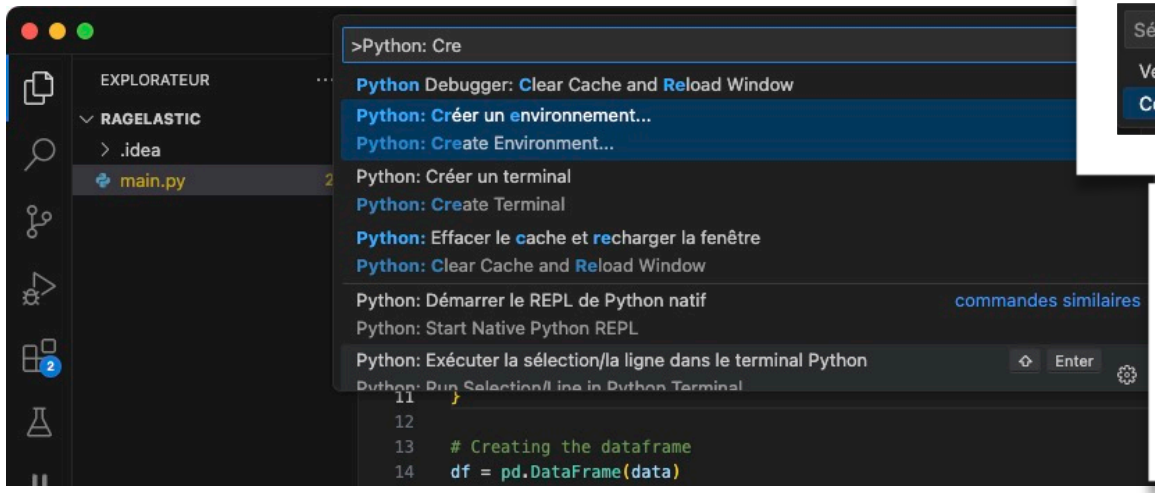


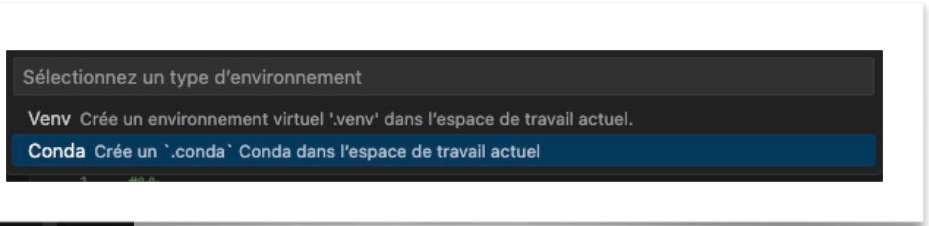
Création d'un environnement Python dédié avec conda

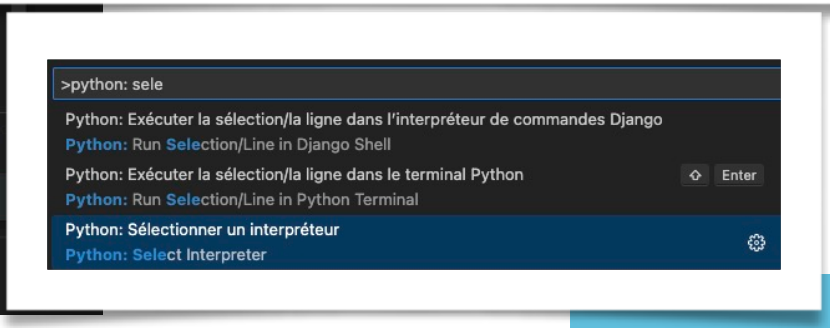
```
conda create -n env-01 python=3.9 scipy=0.15.0 numpy
```



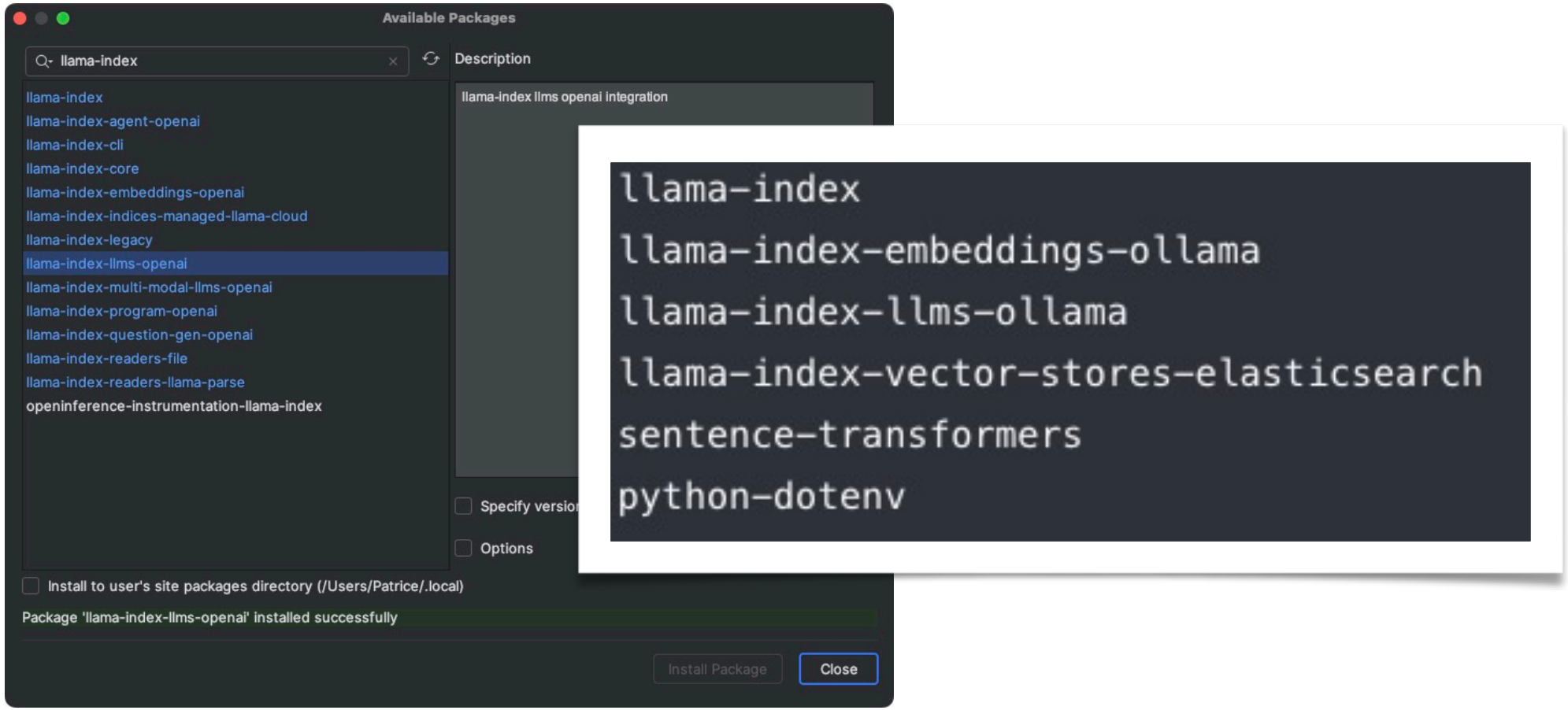








Ajout des paquets nécessaires

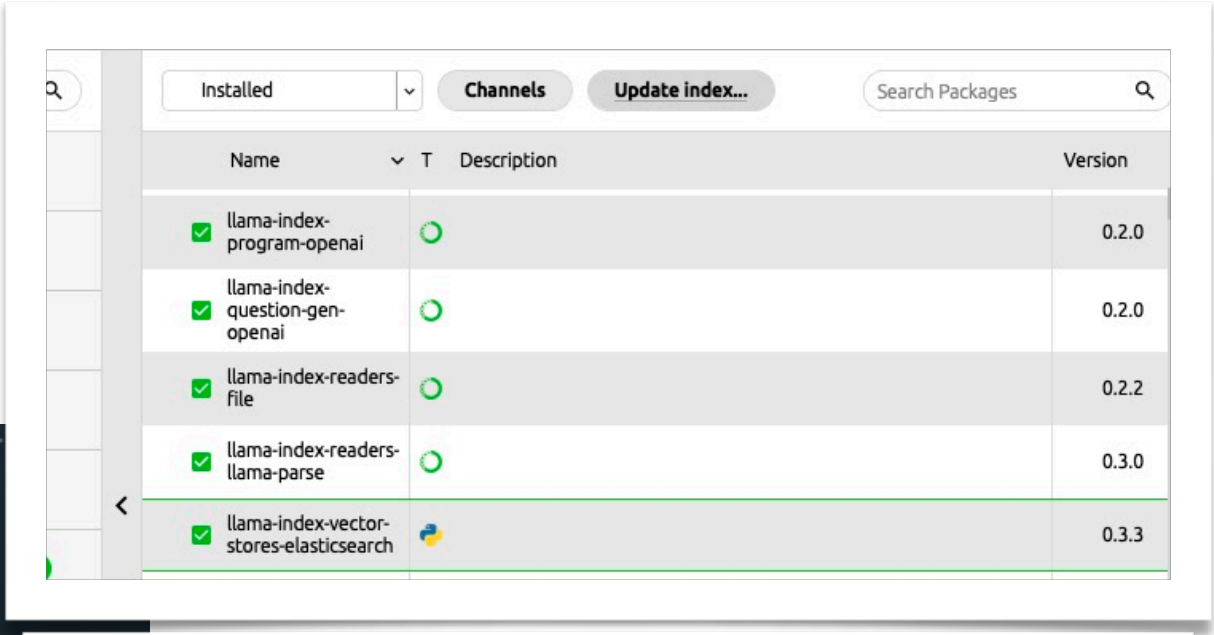


Ajout d'un paquet absent de conda avec pip

```
base ~/Applications/Elastic/kibana-8.15.2 (0.28s)
conda activate RAG

RAG ~/Applications/Elastic/kibana-8.15.2
```

```
RAG ~/Applications/Elastic/kibana-8.15.2 (11.151s)
conda install llama-index-vector-stores-elasticsearch
Channels:
 - conda-forge
Platform: osx-arm64
Collecting package metadata (repodata.json): done
Solving environment: failed

PackagesNotFoundError: The following packages are not available from
hannels:

  - llama-index-vector-stores-elasticsearch
```

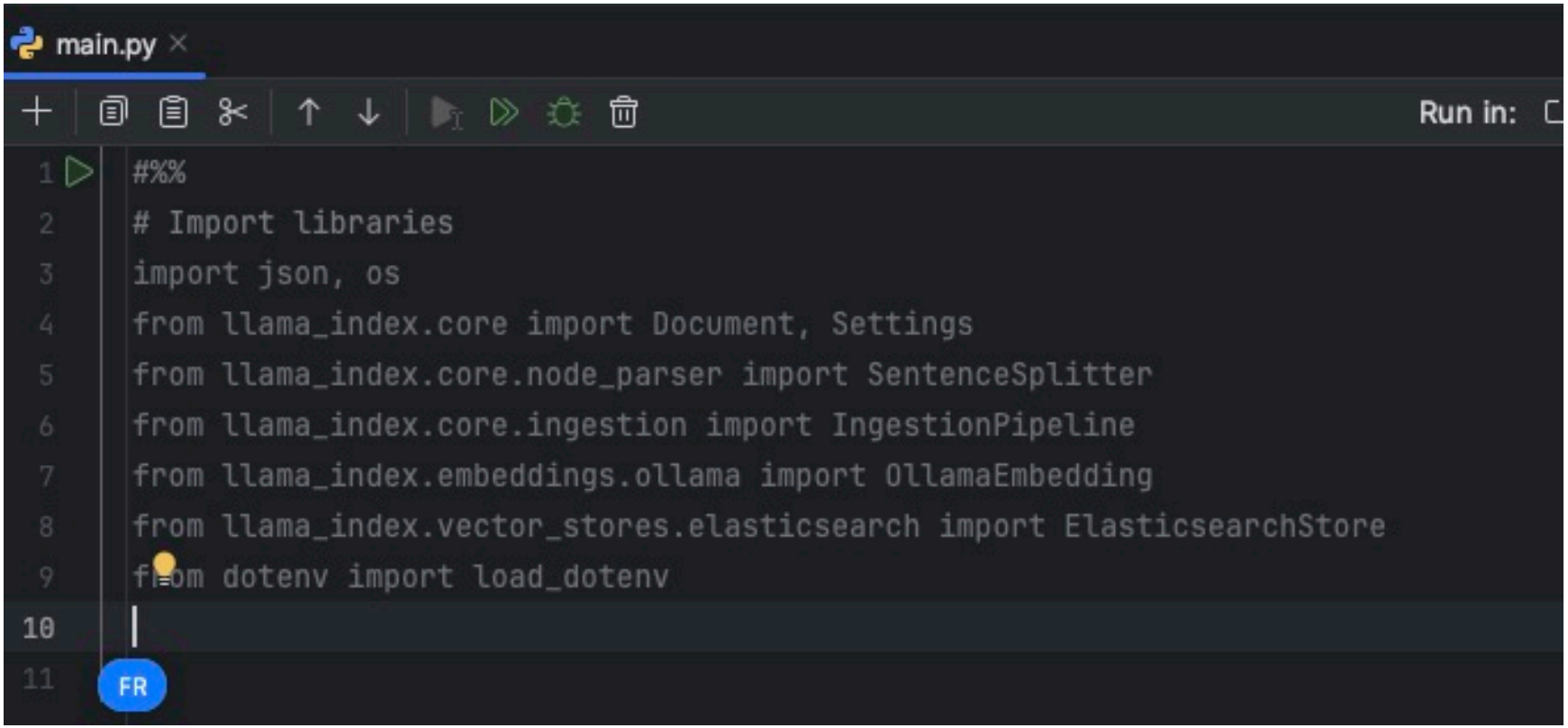
```
RAG ~/Applications/Elastic/kibana-8.15.2 (2.797s)
pip install llama-index-vector-stores-elasticsearch
Collecting llama-index-vector-stores-elasticsearch
  Downloading llama_index_vector_stores_elasticsearch-
```




```
RAG ~/Applications/Elastic/kibana-8.15.2 (0.684s)
conda list
llama-index-multi-modal-llms-openai 0.2.2                pyhd8ed1ab_0     conda-f
orge
llama-index-program-openai 0.2.0             pyhd8ed1ab_1     conda-forge
llama-index-question-gen-openai 0.2.0             pyhd8ed1ab_1     conda-forge
llama-index-readers-file  0.2.2             pyhd8ed1ab_0     conda-forge
llama-index-readers-llama-parse 0.3.0             pyhd8ed1ab_0     conda-forge
llama-index-vector-stores-elasticsearch 0.3.3                    pypi_0    pyp
i
llama-parse               0.5.10            pyhd8ed1ab_0     conda-forge
llamaindex-py-client      0.1.19            pyhd8ed1ab_0     conda-forge
```

Indexation dans ElasticSearch des documents avec les embeddings (plongements) du LLM Mistral

Début du code en Python



conda et pip ne vont pas toujours bien ensemble…

Erreurs…

```
  File "/opt/anaconda3/envs/RAG/lib/python3.12/site-packages/
    from openai.openai_object import OpenAIObject
  File "/Applications/DataSpell.app/Contents/plugins/python-c
    module = self._system_import(name, *args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ModuleNotFoundError: No module named 'openai.openai_object'
>>> # Import libraries
... import json, os
... from llama_index.core import Document, Settings
... from llama_index.core.node_parser import SentenceSplitter
```

```
RAG ~/Applications/Elastic/kibana-8.15.2
conda uninstall ...
```

Solution : ajout des llama-index avec pip

```
pip install llama-index
pip install llama-index-embeddings-ollama
pip install llama-index-llms-ollama
pip install llama-index-vector-stores-elasticsearch
pip install sentence-transformers
pip install python-dotenv
```

```
>>> # Import libraries
... import json, os
... from llama_index.core import Document, Settings
... from llama_index.core.node_parser import SentenceSplitter
... from llama_index.core.ingestion import IngestionPipeline
... from llama_index.embeddings.ollama import OllamaEmbedding
... from llama_index.vector_stores.elasticsearch import Elasticsearch
... from dotenv import load_dotenv
...

>>>
```

Téléchargement d’un fichier de documents (conversations.json)

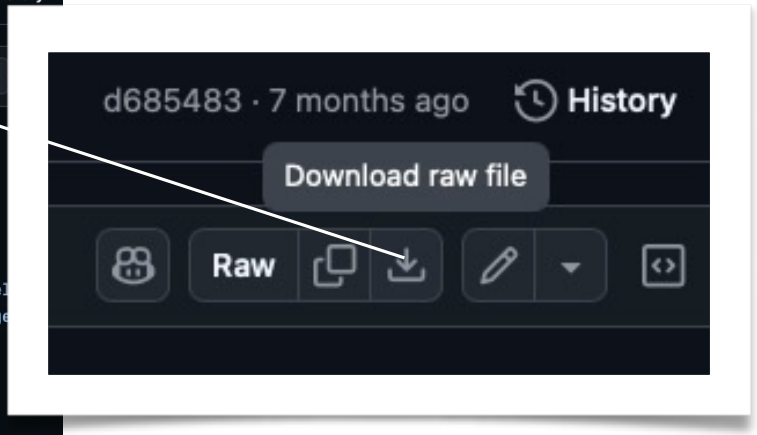
https://github.com/srikanthmanvi/RAG-InsuranceCompany/blob/main/conversations.json

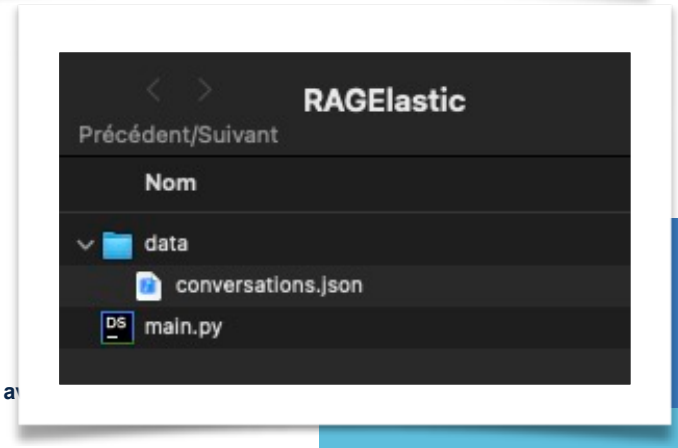
RAG-InsuranceCompany / conversations.json

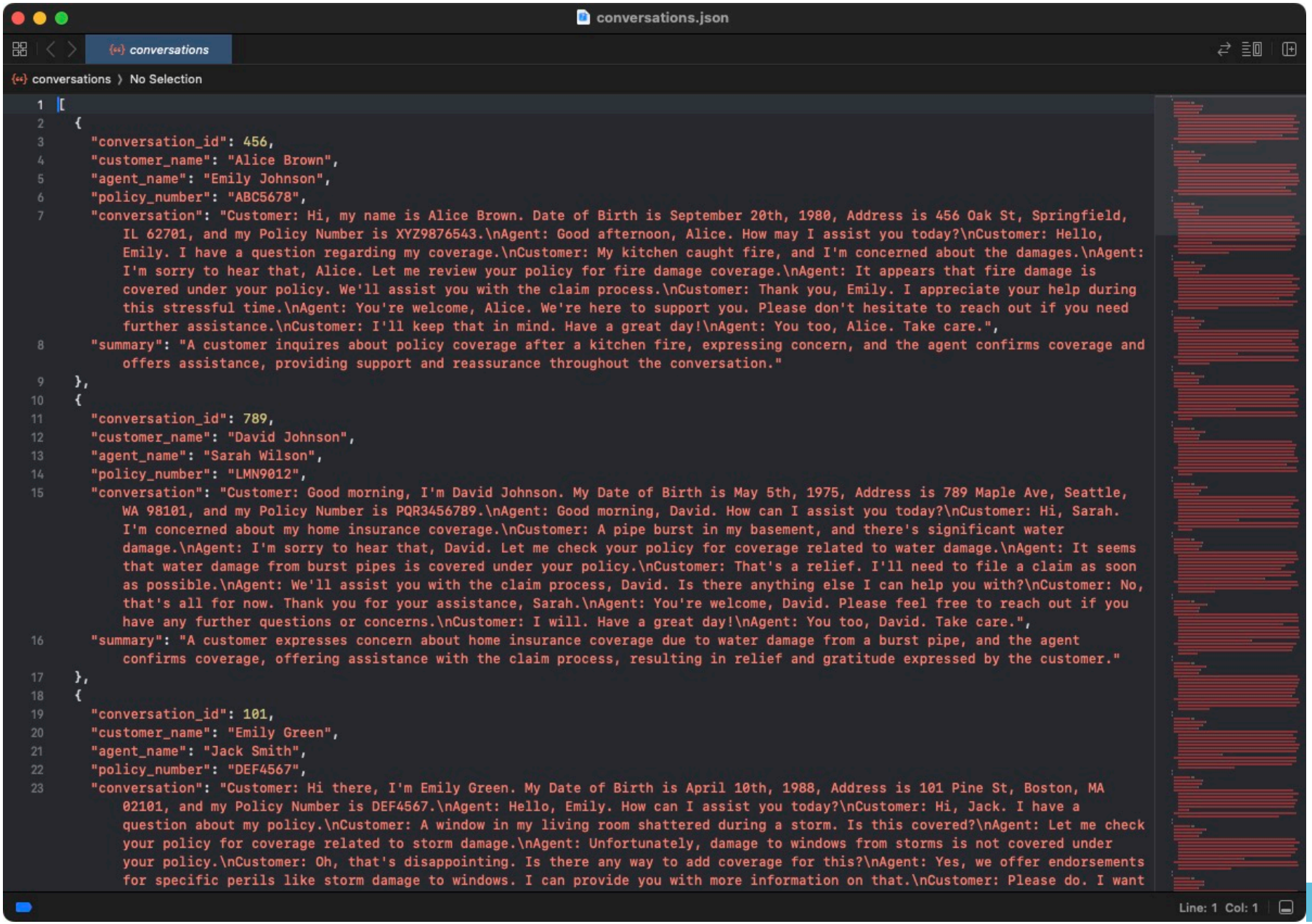
Srikanth Manvi Initial commit — d685483 · 7 months ago — History

Code | Blame | 282 lines (282 loc) · 50.9 KB | Raw


```
[
  {
    "conversation_id": 456,
    "customer_name": "Alice Brown",
    "agent_name": "Emily Johnson",
    "policy_number": "ABC5678",
    "conversation": "Customer: Hi, my name is Alice Brown. Date of Birth is September 20th, 1980, Address is 456 Oak St, Springfiel
    "summary": "A customer inquires about policy coverage after a kitchen fire, expressing concern, and the agent confirms coverage
  },
  {
    "conversation_id": 789,
    "customer_name": "David Johnson",
    "agent_name": "Sarah Wilson",
    "policy_number": "LMN9012",
    "conversation": "Customer: Good morning, I'm David Johnson. My Date of Birth is May 5th, 1975, Address is 789 Maple Ave, Seattle, WA
    "summary": "A customer expresses concern about home insurance coverage due to water damage from a burst pipe, and the agent confirms
  },
  {
    "conversation_id": 101,
    "customer_name": "Emily Green",
    "agent_name": "Jack Smith",
    "policy_number": "DEF4567",
    "conversation": "Customer: Hi there, I'm Emily Green. My Date of Birth is April 10th, 1988, Address is 101 Pine St, Boston, MA 02101
    "summary": "A customer inquires about coverage for a shattered window after a storm, but it's not covered under the policy. The agen
  },
  {
    "conversation_id": 102,
    "customer_name": "Michael White",
```




conversations.json
conversations
conversations 〉 No Selection
[
{
"conversation_id": 456,
"customer_name": "Alice Brown",
"agent_name": "Emily Johnson",
"policy_number": "ABC5678",
"conversation": "Customer: Hi, my name is Alice Brown. Date of Birth is September 20th, 1980, Address is 456 Oak St, Springfield, IL 62701, and my Policy Number is XYZ9876543.\nAgent: Good afternoon, Alice. How may I assist you today?\nCustomer: Hello, Emily. I have a question regarding my coverage.\nCustomer: My kitchen caught fire, and I'm concerned about the damages.\nAgent: I'm sorry to hear that, Alice. Let me review your policy for fire damage coverage.\nAgent: It appears that fire damage is covered under your policy. We'll assist you with the claim process.\nCustomer: Thank you, Emily. I appreciate your help during this stressful time.\nAgent: You're welcome, Alice. We're here to support you. Please don't hesitate to reach out if you need further assistance.\nCustomer: I'll keep that in mind. Have a great day!\nAgent: You too, Alice. Take care.",
"summary": "A customer inquires about policy coverage after a kitchen fire, expressing concern, and the agent confirms coverage and offers assistance, providing support and reassurance throughout the conversation."
},
{
"conversation_id": 789,
"customer_name": "David Johnson",
"agent_name": "Sarah Wilson",
"policy_number": "LMN9012",
"conversation": "Customer: Good morning, I'm David Johnson. My Date of Birth is May 5th, 1975, Address is 789 Maple Ave, Seattle, WA 98101, and my Policy Number is PQR3456789.\nAgent: Good morning, David. How can I assist you today?\nCustomer: Hi, Sarah. I'm concerned about my home insurance coverage.\nCustomer: A pipe burst in my basement, and there's significant water damage.\nAgent: I'm sorry to hear that, David. Let me check your policy for coverage related to water damage.\nAgent: It seems that water damage from burst pipes is covered under your policy.\nCustomer: That's a relief. I'll need to file a claim as soon as possible.\nAgent: We'll assist you with the claim process, David. Is there anything else I can help you with?\nCustomer: No, that's all for now. Thank you for your assistance, Sarah.\nAgent: You're welcome, David. Please feel free to reach out if you have any further questions or concerns.\nCustomer: I will. Have a great day!\nAgent: You too, David. Take care.",
"summary": "A customer expresses concern about home insurance coverage due to water damage from a burst pipe, and the agent confirms coverage, offering assistance with the claim process, resulting in relief and gratitude expressed by the customer."
},
{
"conversation_id": 101,
"customer_name": "Emily Green",
"agent_name": "Jack Smith",
"policy_number": "DEF4567",
"conversation": "Customer: Hi there, I'm Emily Green. My Date of Birth is April 10th, 1988, Address is 101 Pine St, Boston, MA 02101, and my Policy Number is DEF4567.\nAgent: Hello, Emily. How can I assist you today?\nCustomer: Hi, Jack. I have a question about my policy.\nCustomer: A window in my living room shattered during a storm. Is this covered?\nAgent: Let me check your policy for coverage related to storm damage.\nAgent: Unfortunately, damage to windows from storms is not covered under your policy.\nCustomer: Oh, that's disappointing. Is there any way to add coverage for this?\nAgent: Yes, we offer endorsements for specific perils like storm damage to windows. I can provide you with more information on that.\nCustomer: Please do. I want
Line: 1 Col: 1

Les objets Document de LlamaIndex

https://docs.llamaindex.ai/en/stable/module_guides/loading/documents_and_nodes/

Documents / Nodes

Concept

Document and Node objects are core abstractions within LlamaIndex.

A **Document** is a generic container around any data source - for instance, a PDF, an API output, or retrieved data from a database. They can be constructed manually, or created automatically via our data loaders. By default, a Document stores text along with some other attributes. Some of these are listed below.

- `metadata` - a dictionary of annotations that can be appended to the text.
- `relationships` - a dictionary containing relationships to other Documents/Nodes.

Defining Documents

Documents can either be created automatically via data loaders, or constructed manually.

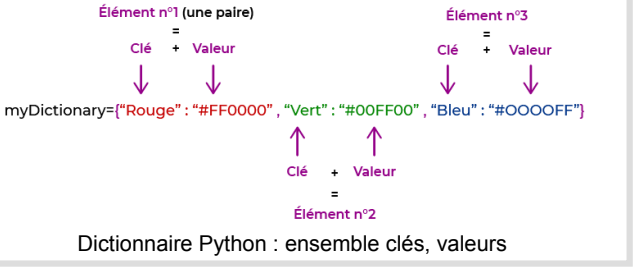
By default, all of our data loaders (including those offered on LlamaHub) return `Document` objects through the `load_data` function.

```
from llama_index.core import SimpleDirectoryReader

documents = SimpleDirectoryReader("./data").load_data()
```

You can also choose to construct documents manually. LlamaIndex exposes the `Document` struct.

```
from llama_index.core import Document

text_list = [text1, text2, ...]
documents = [Document(text=t) for t in text_list]
```

Lecture du fichier JSON



Dictionnaire Python : ensemble clés, valeurs


```
[
  {
    "conversation_id": 456,
    "customer_name": "Alice Brown",
    "agent_name": "Emily Johnson",
    "policy_number": "ABC5678",
    "conversation": "Customer: Hi, my name is Alice Brown. Date of Birth is
        September 20th, 1980, Address is 456 Oak St, Springfield, IL 62701, and my
        Policy Number is XYZ9876543.\nAgent: Good afternoon, Alice. How may I
        assist you today?\nCustomer: Hello, Emily. I have a question regarding my
        coverage.\nCustomer: My kitchen caught fire, and I'm concerned about the
        damages.\nAgent: I'm sorry to hear that, Alice. Let me review your policy
        for fire damage coverage.\nAgent: It appears that fire damage is covered
```


```python
def get_documents_from_file(file):
    """Reads a json file and returns list of Documents"""
    with open(file, 'rt') as f:
        conversations_dict = json.loads(f.read())

    documents = [Document(text=item['conversation'],
                          metadata={"conversation_id": item['conversation_id']})
                 for item in conversations_dict]

    return documents
```

The json. load() is used to read the JSON document from file and The json. loads() is used to convert the JSON String document into the Python dictionary. fp file pointer use

Vérification fichier conversations bien lu

```
documents = get_documents_from_file("data/conversations.json")
print("id du premier document :", documents[0].metadata["conversation_id"])
print("texte: ", documents[0].text)
```

```
id du premier document : 456
texte:  Customer: Hi, my name is Alice Brown. Date of Birth is September 20th, 1980, Address is 456 Oak St, Springfield, IL 62701, and my Policy Number is XYZ9876543.
Agent: Good afternoon, Alice. How may I assist you today?
Customer: Hello, Emily. I have a question regarding my coverage.
Customer: My kitchen caught fire, and I'm concerned about the damages.
Agent: I'm sorry to hear that, Alice. Let me review your policy for fire damage coverage.
Agent: It appears that fire damage is covered under your policy. We'll assist you with the claim process.
Customer: Thank you, Emily. I appreciate your help during this stressful time.
Agent: You're welcome, Alice. We're here to support you. Please don't hesitate to reach out if you need further assistance.
Customer: I'll keep that in mind. Have a great day!
Agent: You too, Alice. Take care.
```

Connexion à ElasticSearch

Version locale :

```
# ElasticsearchStore is a VectorStore that
# takes care of ES Index and Data management.
es_vector_store = ElasticsearchStore(index_name="calls",
                                     vector_field='conversation_vector',
                                     text_field='conversation',
                                     es_user="elastic",
                                     es_password="UcnisewsseNTUmlUfdy+",
                                     es_url=« http://localhost:9200")
```

Attention : il faut avoir auparavant désactivé le chiffrage de la connexion

- éditer le fichier config/elasticsearch.yml et mettre xpack.security.transport.ssl : enabled à false puis redémarrer le serveur Plastic

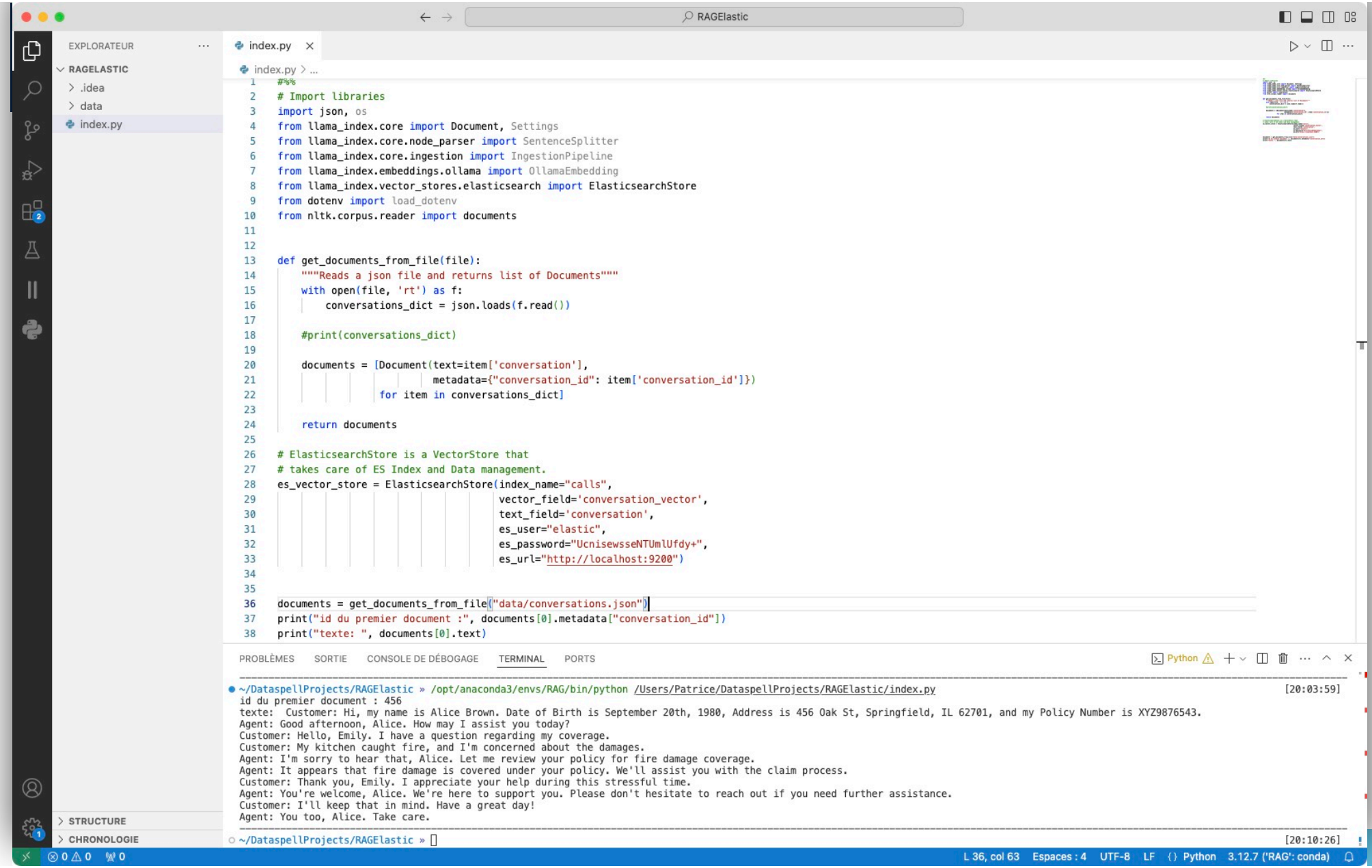
```
# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
xpack.security.http.ssl:
  enabled: false
  keystore.path: certs/http.p12
```

Version Cloud : création d'un fichier .env

Firstly, add the Elasticsearch CloudID and API keys that you obtained in the `Install Elasticsearch` section to the `.env` file. Your `.env` file should look like the below (with real values).

```
ELASTIC_CLOUD_ID=<REPLACE WITH YOUR CLOUD ID>
ELASTIC_API_KEY=<REPLACE WITH YOUR API_KEY>
```

```
# Load .env file contents into env
# ELASTIC_CLOUD_ID and ELASTIC_API_KEY are expected to be in the .env file.
load_dotenv('.env')

# ElasticsearchStore is a VectorStore that
# takes care of ES Index and Data management.
es_vector_store = ElasticsearchStore(index_name="calls",
                                     vector_field='conversation_vector',
                                     text_field='conversation',
                                     es_cloud_id=os.getenv("ELASTIC_CLOUD_ID"),
                                     es_api_key=os.getenv("ELASTIC_API_KEY"))
```


RAGElastic
EXPLORATEUR
RAGELASTIC
.idea
data
index.py
index.py
#%%
# Import libraries
import json, os
from llama_index.core import Document, Settings
from llama_index.core.node_parser import SentenceSplitter
from llama_index.core.ingestion import IngestionPipeline
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.vector_stores.elasticsearch import ElasticsearchStore
from dotenv import load_dotenv
from nltk.corpus.reader import documents
def get_documents_from_file(file):
"""Reads a json file and returns list of Documents"""
with open(file, 'rt') as f:
conversations_dict = json.loads(f.read())
#print(conversations_dict)
documents = [Document(text=item['conversation'],
metadata={"conversation_id": item['conversation_id']})
for item in conversations_dict]
return documents
# ElasticsearchStore is a VectorStore that
# takes care of ES Index and Data management.
es_vector_store = ElasticsearchStore(index_name="calls",
vector_field='conversation_vector',
text_field='conversation',
es_user="elastic",
es_password="UcnisewsseNTUmlUfdy+",
es_url="http://localhost:9200")
documents = get_documents_from_file("data/conversations.json")
print("id du premier document :", documents[0].metadata["conversation_id"])
print("texte: ", documents[0].text)
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
Python
~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/RAGElastic/index.py
[20:03:59]
id du premier document : 456
texte: Customer: Hi, my name is Alice Brown. Date of Birth is September 20th, 1980, Address is 456 Oak St, Springfield, IL 62701, and my Policy Number is XYZ9876543.
Agent: Good afternoon, Alice. How may I assist you today?
Customer: Hello, Emily. I have a question regarding my coverage.
Customer: My kitchen caught fire, and I'm concerned about the damages.
Agent: I'm sorry to hear that, Alice. Let me review your policy for fire damage coverage.
Agent: It appears that fire damage is covered under your policy. We'll assist you with the claim process.
Customer: Thank you, Emily. I appreciate your help during this stressful time.
Agent: You're welcome, Alice. We're here to support you. Please don't hesitate to reach out if you need further assistance.
Customer: I'll keep that in mind. Have a great day!
Agent: You too, Alice. Take care.
~/DataspellProjects/RAGElastic »
[20:10:26]
STRUCTURE
CHRONOLOGIE
L 36, col 63
Espaces : 4
UTF-8
LF
Python
3.12.7 ('RAG': conda)

Indexation des conversations en utilisant les représentations vectorielles du LLM Mistral

```
def main():
    # Embedding Model to do local embedding using Ollama.
    ollama_embedding = OllamaEmbedding("mistral")

    # LlamaIndex Pipeline configured to take care of chunking, embedding
    # and storing the embeddings in the vector store.
    pipeline = IngestionPipeline(
        transformations=[
            SentenceSplitter(chunk_size=350, chunk_overlap=50),
            ollama_embedding,
        ],
        vector_store=es_vector_store
    )

    # Load data from a json file into a list of LlamaIndex Documents
    documents = get_documents_from_file("data/conversations.json")

    pipeline.run(documents=documents)
    print(".... exécution du pipeline .... \n")


if __name__ == "__main__":
    main()
```

Le code Python suivant :

```python
if __name__ == "__main__":
    main()
```

Copier

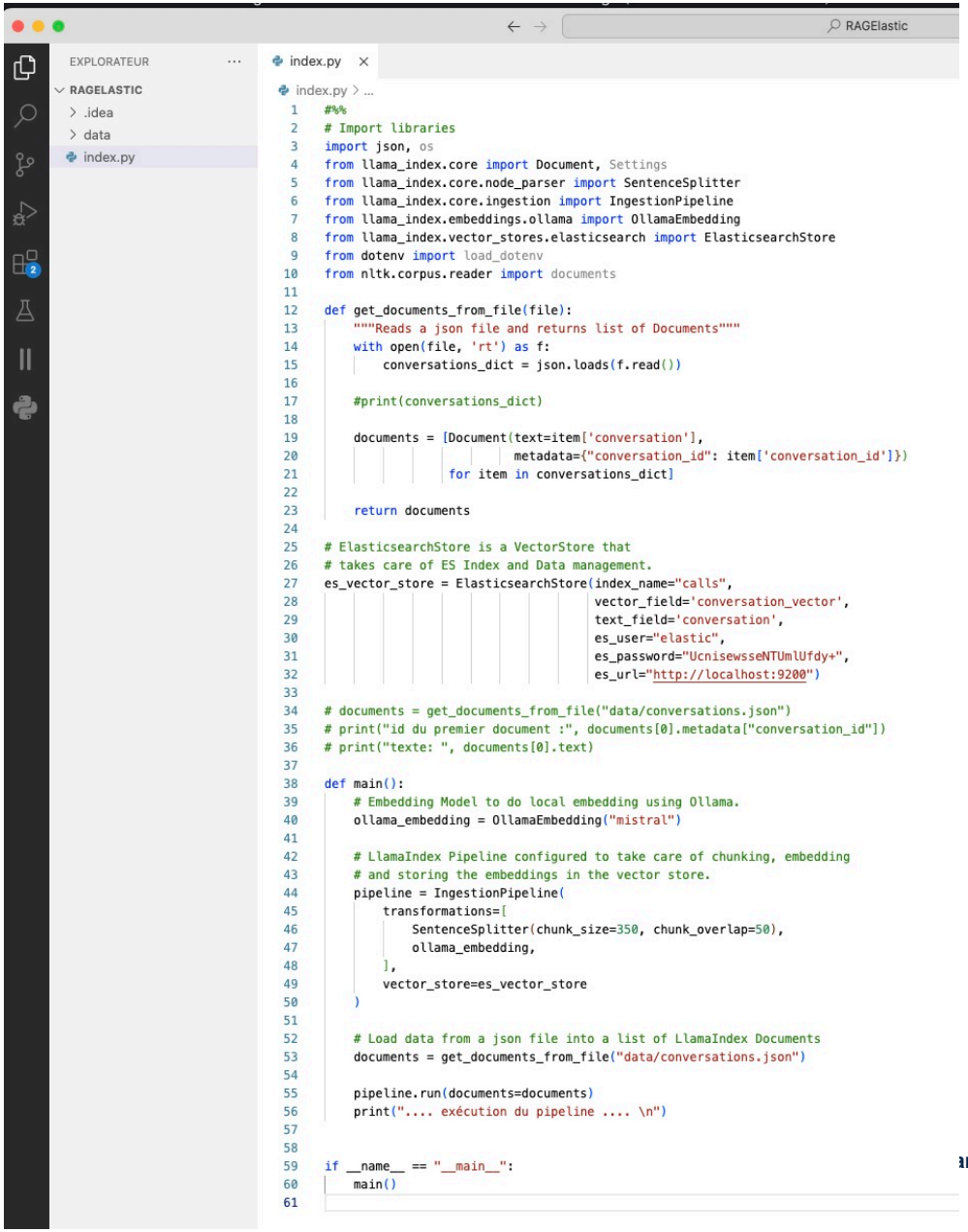
est un **idiome standard** utilisé pour vérifier si le fichier Python est exécuté directement (en tant que script principal) ou s'il est importé en tant que module dans un autre fichier.

Explication détaillée

1. `__name__` :
En Python, chaque module (fichier Python) a une variable spéciale appelée `__name__`. Cette variable contient le nom du module.
   - Si le fichier est exécuté directement, c'est-à-dire lancé comme un script, Python assigne à `__name__` la valeur `"__main__"`.
   - Si le fichier est importé dans un autre fichier comme module, `__name__` prend le nom du fichier (ou module) sans exécuter le code sous ce bloc conditionnel.

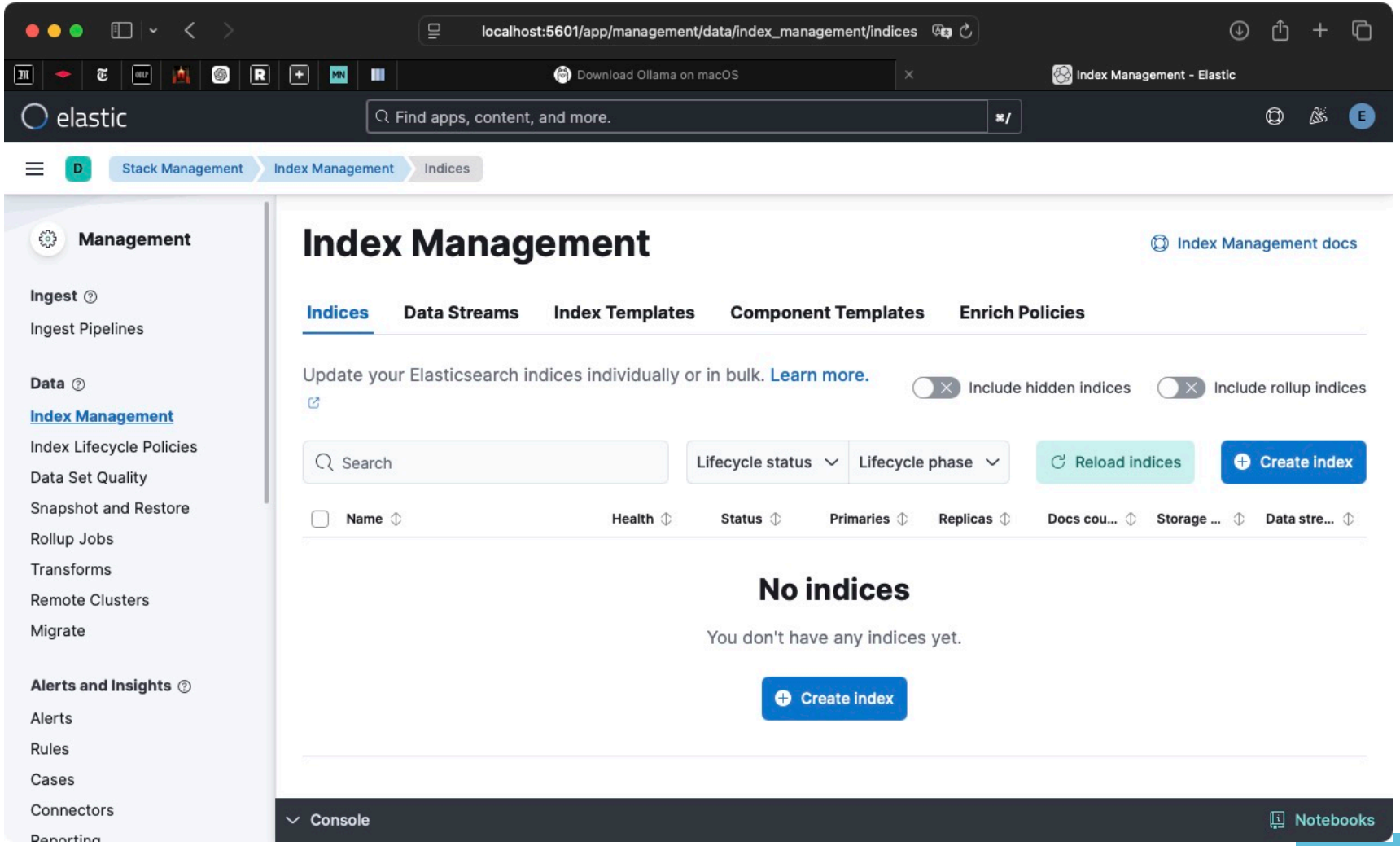
2. `if __name__ == "__main__":`
Cette condition vérifie si le fichier est exécuté directement ou importé comme module.
   - Si le fichier est **exécuté directement**, la condition est vraie (`__name__` vaut `"__main__"`), et le code sous cette condition (dans ce cas, l'appel à `main()`) est exécuté.
   - Si le fichier est **importé comme module** dans un autre script, la condition est fausse, et le code sous ce bloc ne sera **pas exécuté**.

Code complet



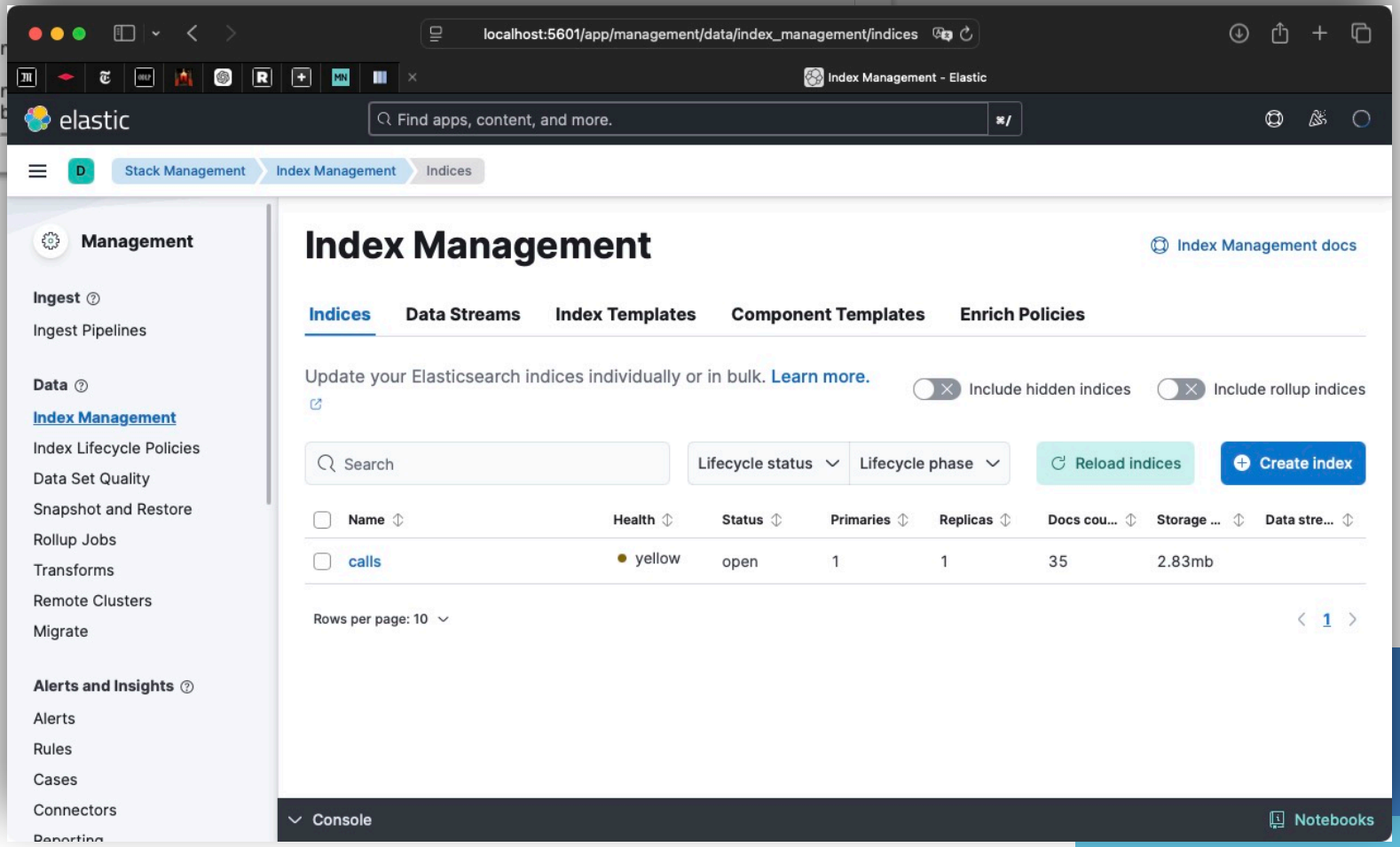

```python
#%%
# Import libraries
import json, os
from llama_index.core import Document, Settings
from llama_index.core.node_parser import SentenceSplitter
from llama_index.core.ingestion import IngestionPipeline
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.vector_stores.elasticsearch import ElasticsearchStore
from dotenv import load_dotenv
from nltk.corpus.reader import documents

def get_documents_from_file(file):
    """Reads a json file and returns list of Documents"""
    with open(file, 'rt') as f:
        conversations_dict = json.loads(f.read())

    #print(conversations_dict)

    documents = [Document(text=item['conversation'],
                          metadata={"conversation_id": item['conversation_id']})
                 for item in conversations_dict]

    return documents

# ElasticsearchStore is a VectorStore that
# takes care of ES Index and Data management.
es_vector_store = ElasticsearchStore(index_name="calls",
                                     vector_field='conversation_vector',
                                     text_field='conversation',
                                     es_user="elastic",
                                     es_password="UcnisewsseNTUmlUfdy+",
                                     es_url="http://localhost:9200")

# documents = get_documents_from_file("data/conversations.json")
# print("id du premier document :", documents[0].metadata["conversation_id"])
# print("texte: ", documents[0].text)

def main():
    # Embedding Model to do local embedding using Ollama.
    ollama_embedding = OllamaEmbedding("mistral")

    # LlamaIndex Pipeline configured to take care of chunking, embedding
    # and storing the embeddings in the vector store.
    pipeline = IngestionPipeline(
        transformations=[
            SentenceSplitter(chunk_size=350, chunk_overlap=50),
            ollama_embedding,
        ],
        vector_store=es_vector_store
    )

    # Load data from a json file into a list of LlamaIndex Documents
    documents = get_documents_from_file("data/conversations.json")

    pipeline.run(documents=documents)
    print(".... exécution du pipeline .... \n")


if __name__ == "__main__":
    main()
```

Avant l'exécution : aucun index (indices) présent dans ElasticSearch (visualisation des index dans Kibana)

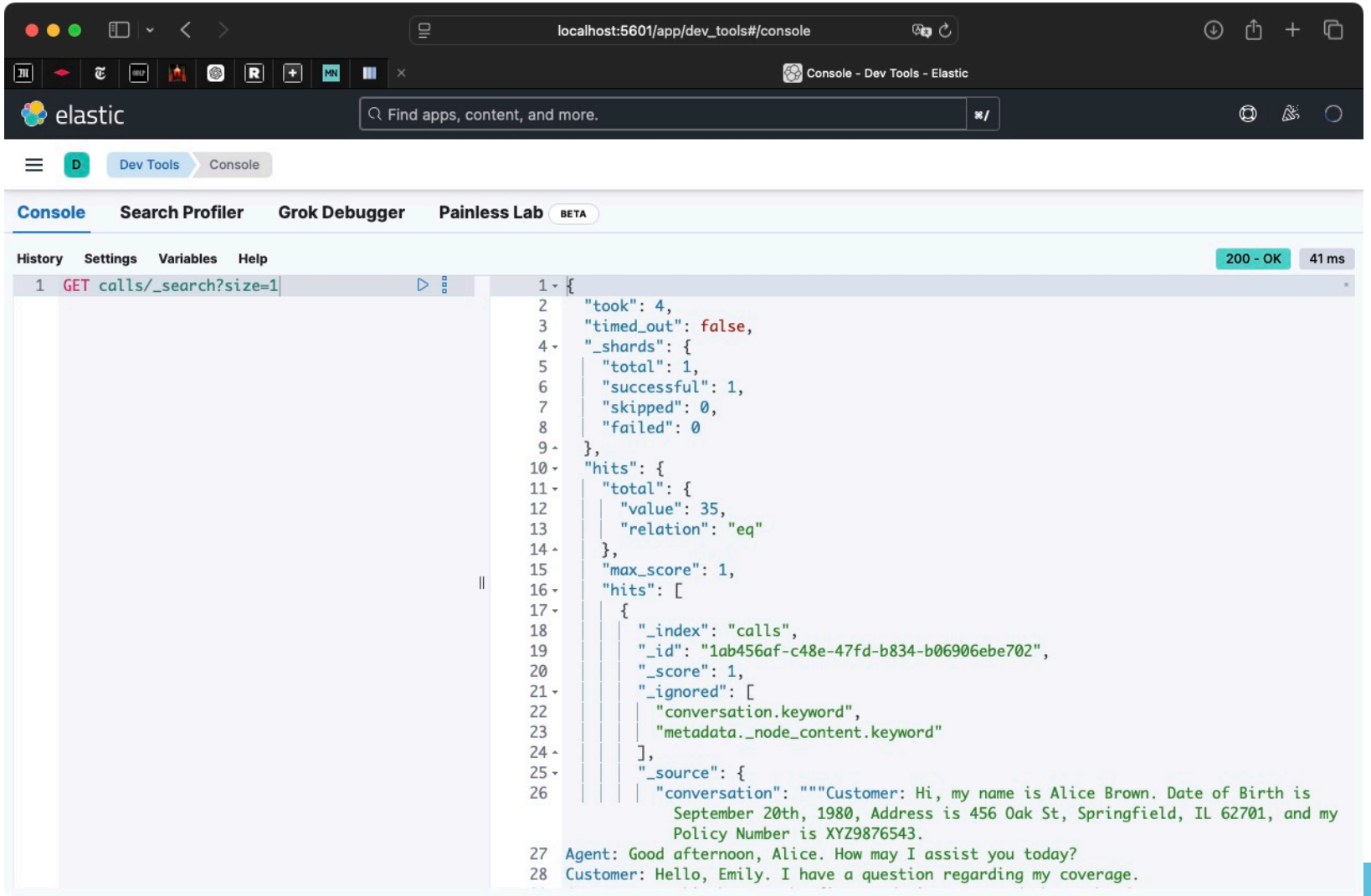


Exécution du pipeline

```
~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/RAGElastic/index.py
.... exécution du pipeline ....

Unclosed client session
client_session: <aiohttp.client.ClientSessior
Unclosed connector
connections: ['[(<aiohttp.client_proto.Respor
connector: <aiohttp.connector.TCPConnector ob
```



Visualisation du contenu de l'index



```
  "hits": {
    "total": {
      "value": 35,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "_index": "calls",
        "_id": "1ab456af-c48e-47fd-b834-b06906ebe702",
        "_score": 1,
        "_ignored": [
          "conversation.keyword",
          "metadata._node_content.keyword"
        ],
        "_source": {
          "conversation": """Customer: Hi, my name is Alice Brown. Date of Birth is
            September 20th, 1980, Address is 456 Oak St, Springfield, IL 62701, and my
            Policy Number is XYZ9876543.
Agent: Good afternoon, Alice. How may I assist you today?
Customer: Hello, Emily. I have a question regarding my coverage.
Customer: My kitchen caught fire, and I'm concerned about the damages.
Agent: I'm sorry to hear that, Alice. Let me review your policy for fire damage
  coverage.
Agent: It appears that fire damage is covered under your policy. We'll assist you with
  the claim process.
Customer: Thank you, Emily. I appreciate your help during this stressful time.
Agent: You're welcome, Alice. We're here to support you. Please don't hesitate to reach
  out if you need further assistance.
```

```
  out if you need further assistance.
Customer: I'll keep that in mind. Have a great day!
Agent: You too, Alice. Take care.""",
  "metadata": {
    "conversation_id": 456,
    "_node_content": """{"id_": "1ab456af-c48e-47fd-b834-b06906ebe702",
      "embedding": null, "metadata": {"conversation_id": 456},
      "excluded_embed_metadata_keys": [], "excluded_llm_metadata_keys": [],
      "relationships": {"1": {"node_id": "ef766590-99ea-41cb-8c39-6c02337aa240"
      , "node_type": "4", "metadata": {"conversation_id": 456}, "hash":
      "bfc62834363b1db3fb6a9652469a93b68116faaebb0783bf07aa89e271372a19",
      "class_name": "RelatedNodeInfo"}}, "text": "", "mimetype": "text/plain",
      "start_char_idx": 0, "end_char_idx": 837, "text_template":
      "{metadata_str}\n\n{content}", "metadata_template": "{key}: {value}",
      "metadata_seperator": "\n", "class_name": "TextNode"}""",
    "_node_type": "TextNode",
    "document_id": "ef766590-99ea-41cb-8c39-6c02337aa240",
    "doc_id": "ef766590-99ea-41cb-8c39-6c02337aa240",
    "ref_doc_id": "ef766590-99ea-41cb-8c39-6c02337aa240"
  },
  "conversation_vector": [
    2.7598750591278076,
    -0.6350980401039124,
    -1.8095593452453613,
    -3.714324474334717,
    -1.0254573822021484,
    5.799200534820557,
    5.297024726867676,
    -5.098982334136963,
```

Requêtes (questions)

Création d'une requête (query) et interrogation d'Elastic + LLM

```
from llama_index.core import VectorStoreIndex, QueryBundle, Response, Settings
from llama_index.embeddings.ollama import OllamaEmbedding
from llama_index.llms.ollama import Ollama
from index import es_vector_store

# Local LLM to send user query to
local_llm = Ollama(model="mistral")
Settings.embed_model = OllamaEmbedding("mistral")

index = VectorStoreIndex.from_vector_store(es_vector_store)
query_engine = index.as_query_engine(local_llm, similarity_top_k=10)

query="Quelle est la capitale de la Belgique"
bundle = QueryBundle(query, embedding=Settings.embed_model.get_query_embedding(query))
result = query_engine.query(bundle)
print(result)
```

Réponse : presque identique à celle avec l'application Ollama (LLM seul)

```
~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/RAGElastic/query.py
 The capital of Belgium is Brussels.
Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x311983b60>
Unclosed connector
connections: ['[(<aiohttp.client_proto.ResponseHandler object at 0x3119aca70>, 35723.31844475)]']
connector: <aiohttp.connector.TCPConnector object at 0x311983ad0>
```

La réponse n'est pas dans les conversations : le LLM se débrouille seul

Une deuxième requête

```
query="Give me summary of water related issues"
bundle = QueryBundle(query, embedding=Settings.embed_model.get_query_embedding(query))
result = query_engine.query(bundle)
print(result)
```

```
~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/RAGElastic/query.py
                                                    [21:05:22]
1) In conversation 139, Rachel Clark was denied a claim for water damage due to it being deemed as a result of gradual wear and tear, which
 is not covered under her policy. She expressed frustration over this decision.

2) In conversation 789, David Johnson had a pipe burst in his basement causing significant water damage. His agent confirmed that water dam
age from burst pipes is covered under his policy.

3) In conversation 104, Ethan Wilson's garage door was damaged during a storm. The agent confirmed that damage to the garage door from stor
ms is covered under his policy.

4) In conversation 138, Jason Miller was furious with the company's lack of responsiveness regarding a claim for water damage. Despite not
specifying the nature of the water-related issue in this conversation, it can be inferred that there might have been a water damage claim i
nvolved due to his reference to trying to contact the claims department for days related to an issue.

5) In conversations 141, 142, and 143, Matthew Lopez, Amanda Thompson, and Jennifer Lee all had their claims for water-related damages (wat
er damage, fire damage resulting from a water source, and fire damage due to arson respectively) denied. They were all disappointed and exp
ressed frustration with the denials.

In summary, while some customers encountered water-related issues, the outcomes varied between having their claims accepted or denied. Thos
e with accepted claims received confirmation that they were covered, while those whose claims were denied expressed disappointment and frus
tration due to their expectations of coverage.
```

alors qu'avec le LLM seul...

```
>>> Give me summary of water related issues
 Water-related issues encompass a wide range of challenges that affect the environment, economy, and human
wellbeing worldwide. Some key areas include:

1. **Water scarcity**: Droughts, climate change, and population growth are leading to reduced access to freshwater
in many regions, threatening agriculture, industry, and ecosystem health.
2. **Water pollution**: Industrial waste, agricultural runoff, and untreated sewage are contaminating rivers,
lakes, and groundwater resources, making them unsafe for consumption, irrigation, or recreation.
3. **Infrastructure development**: Rapid urbanization and industrialization often result in the construction of
dams, reservoirs, and water supply systems that can disrupt ecosystems, displace communities, and exacerbate
conflicts over water resources.
4. **Water-related disasters**: Floods, landslides, and other natural disasters pose significant threats to human
life, property, and infrastructure, particularly in areas prone to extreme weather events.
5. **Equity and justice**: Access to safe and affordable water is a fundamental right, but millions of people
around the world still lack access to clean drinking water or basic sanitation facilities. Social, economic, and
political factors can exacerbate these inequalities, particularly for marginalized communities.
6. **Climate change adaptation and mitigation**: Water resources will be impacted by climate change, with rising
temperatures, changing precipitation patterns, and increased frequency of extreme events expected to further
strain water resources and challenge efforts to ensure water security.
```

D’autres requetes…

"Que dit Emily Rodriguez?"

```
● ~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/
  RAGElastic/query.py        [21:16:52]
   Emily Rodriguez assisted customers with their home insurance policies and provided information about
   coverage and the claim process.
```

"Que dit Emily Rodriguez? (merci de répondre en français)"

```
  ------------------------------------------------------------------------------------------------------
● ~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/
  RAGElastic/query.py
   Emily Rodriguez a indiqué qu'elle est déçue et frustrée par la décision de rejet de sa demande de pr
  ise en charge, qui a été rejetée sans explication. Elle considère que cela est inacceptable, car elle
   a payé des cotisations pendant plusieurs années, attendait une couverture lorsqu'elle en avait besoi
  n le plus, et pensait qu'elle était assurée contre ce genre de dommages.
```

Et encore...

Quelles sont les personnes qui interviennent dans les conversations ?

```
~/DataspellProjects/RAGElastic » /opt/anaconda3/envs/RAG/bin/python /Users/Patrice/DataspellProjects/
RAGElastic/query.py
 Dans chaque conversation figurent un client et un agent de l'assurance. Les clients sont William And
erson, Sophia Jones, Emily Green, Ethan Wilson, Michael White, David Johnson, Olivia Taylor, Michael
Johnson, Alice Brown et Andrew Brown. L'agent commun à toutes les conversations est Jack, mais il y a
 également Emily, Sarah et Emily dans certaines conversations.
```