

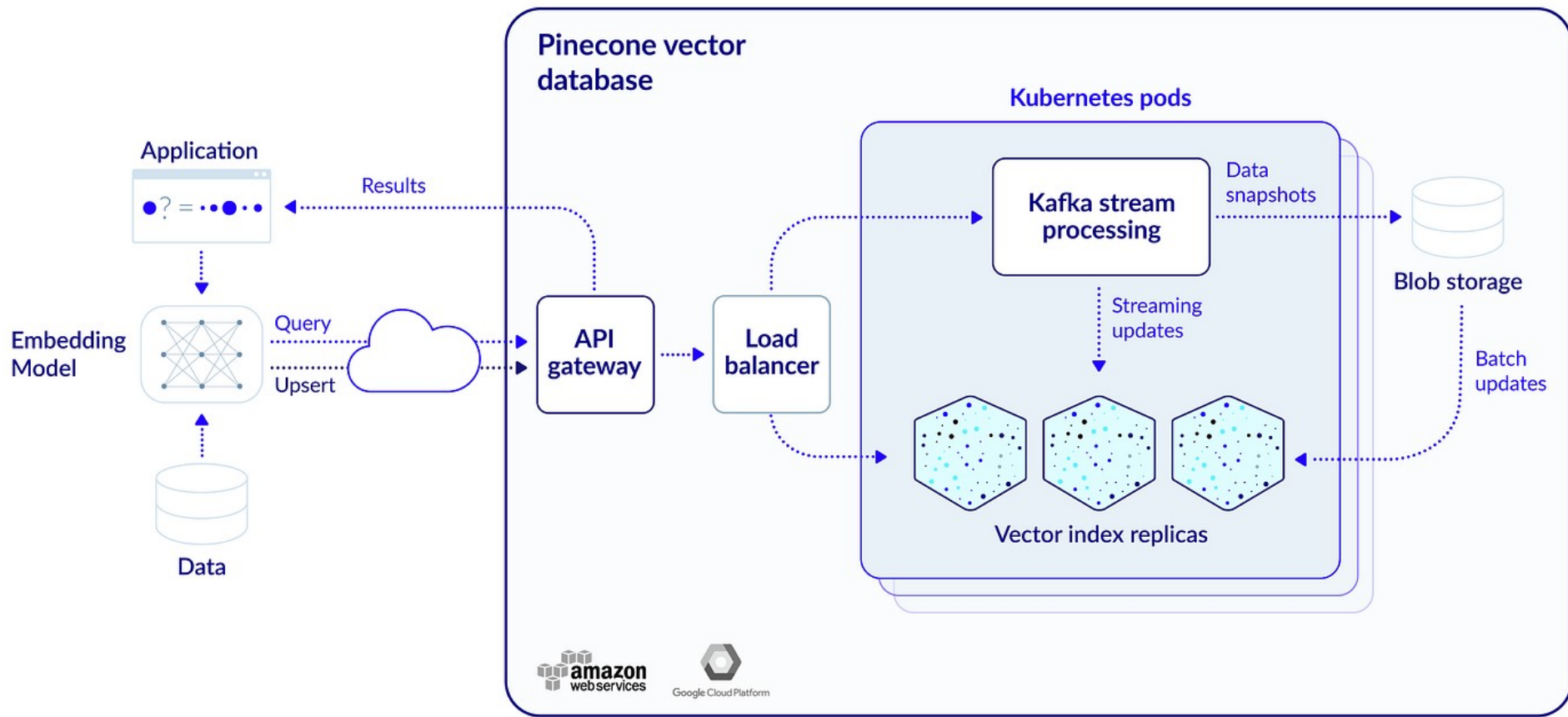
Pinecone

Introduction

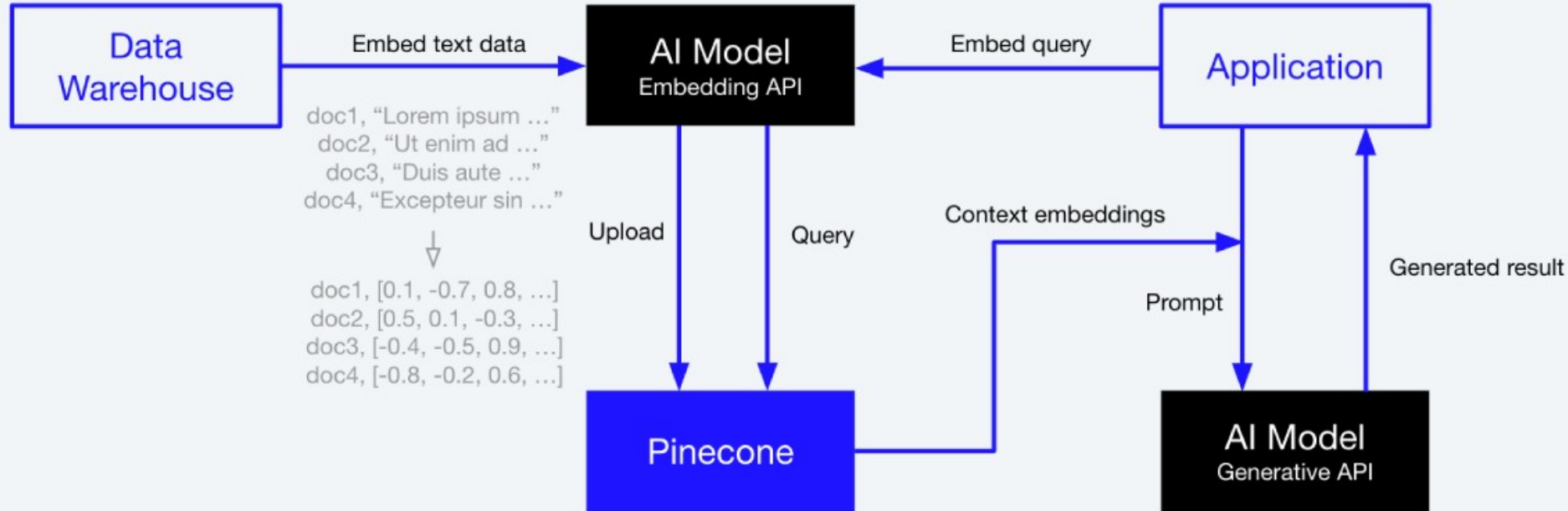
What is pinecone

Pinecone is the vector database for machine learning applications. Build vector-based personalization, ranking, and search systems that are accurate, fast, and scalable. Use simple APIs with zero maintenance.

In this embeddings of documents or content are stored and retrieve embedding by cosine similarity.



For chatbot



Important libraries to convert document to embeddings

For langchain: `!pip install --upgrade langchain openai -q`

For embeddings: `!pip install sentence_transformers -q`

Other : `!pip install unstructured -q`

`!pip install unstructured[local-inference] -q`

`!pip install detectron2@git+https://github.com/facebookresearch/detectron2.git@v0.6#egg=detectron2 -q`

`!apt-get install poppler-utils`

Load directory in which are saved in of readable extension

```
from langchain.document_loaders import DirectoryLoader
```

```
directory = '/content/data/'
```

```
def load_docs(directory):
```

```
    loader = DirectoryLoader(directory)
```

```
    documents = loader.load()
```

```
    return documents
```

```
documents = load_docs(directory)
```

Split document into chunks

```
from langchain.text_splitter import RecursiveCharacterTextSplitter

def split_docs(documents, chunk_size=500, chunk_overlap=20):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=chunk_overlap)
    docs = text_splitter.split_documents(documents)
    return docs

docs = split_docs(documents)
print(len(docs))
```

Embeddings and Initialize Pinecone

pinecone

```
from langchain.embeddings import SentenceTransformerEmbeddings
embeddings = SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")
```

```
import pinecone
from langchain.vectorstores import Pinecone
pinecone.init(
    api_key="0f0a35be-e8ea-4068-9876-0017b4e4c2cb",
    environment="us-west1-gcp-free"
)
index_name = "longchain-chatbot"
index = Pinecone.from_documents(docs, embeddings, index_name=index_name)
```


Search query in vector DB

```
query = "famous places in Los Angeles?"  
query_result = embeddings.embed_query(query)  
index.query(query_result, top_k=4, include_metadata=True)
```

In this we can see that query is written then query is converted into embeddings now index.query will search nearest or similar embeddings related to query.

Thank You

