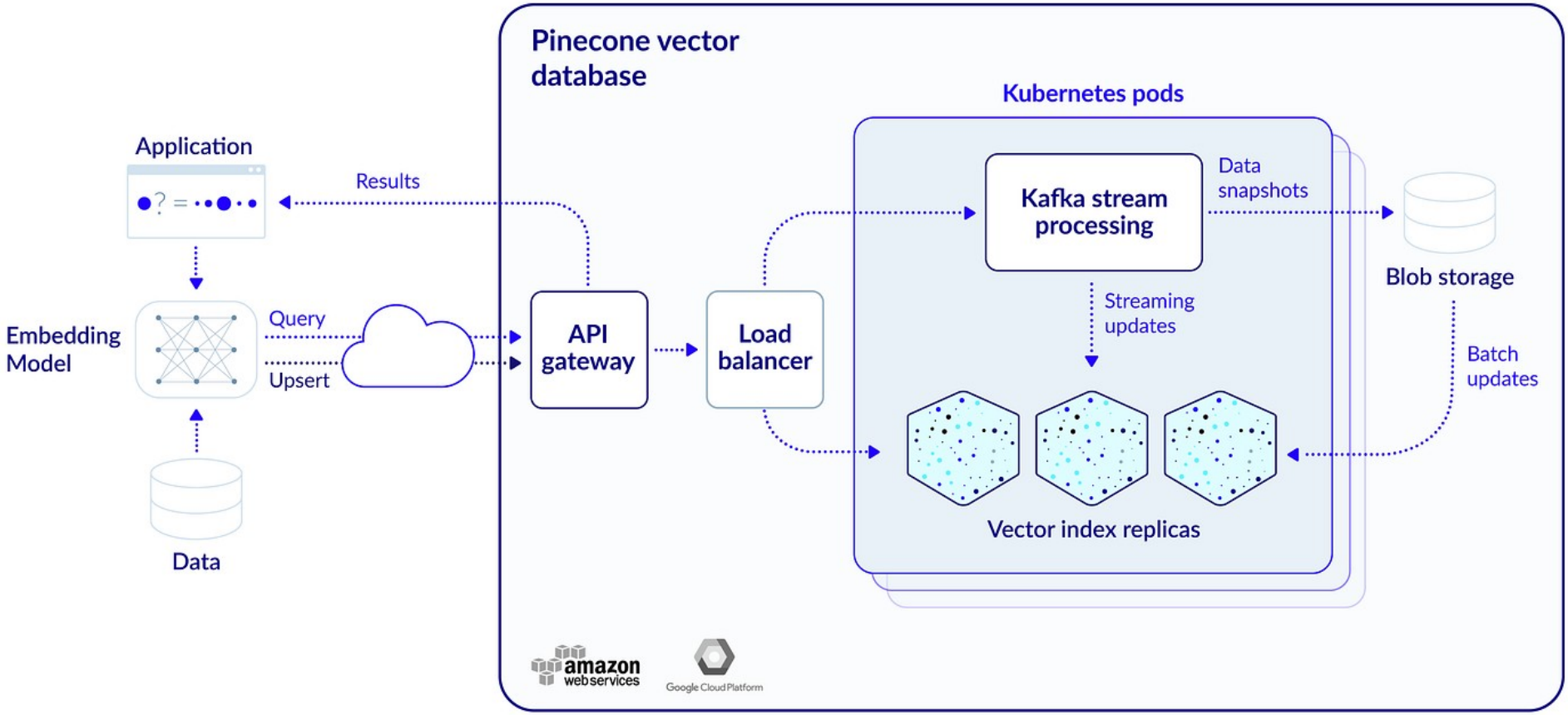
Pinecone

Introduction

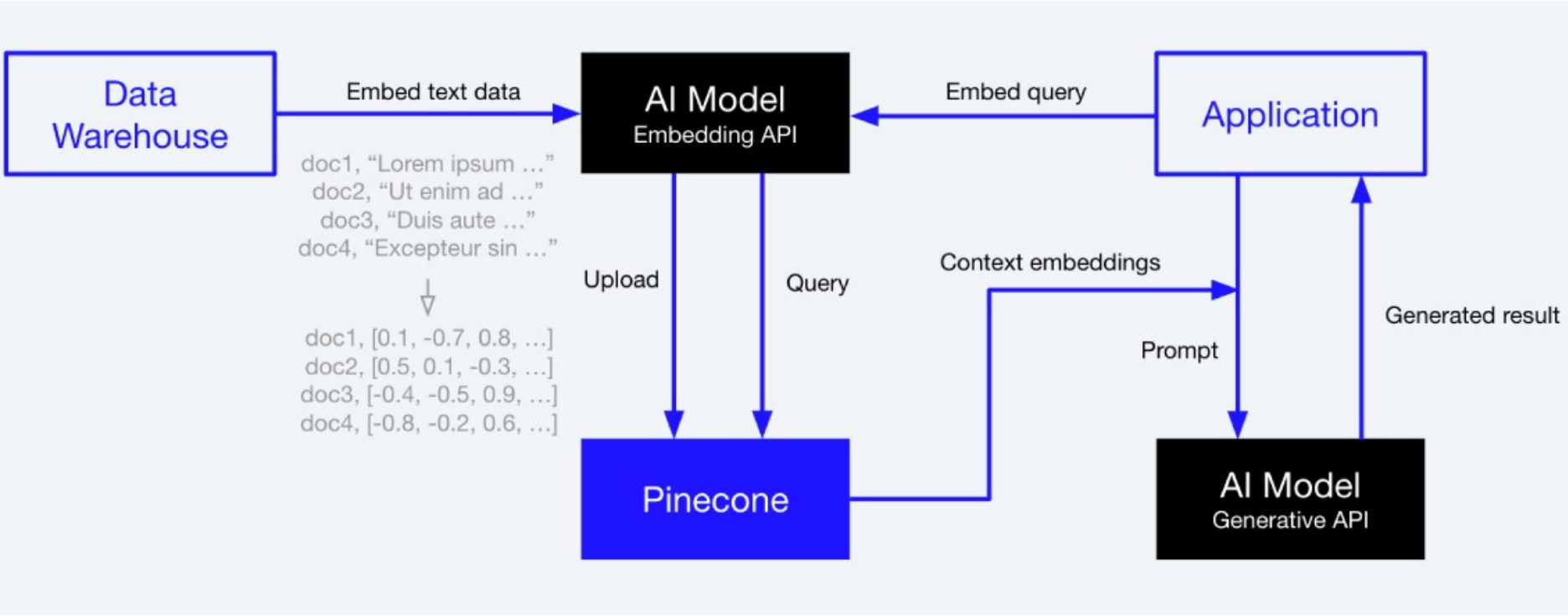
What is pinecone

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.


Pinecone vector database
Kubernetes pods
Application
Results
Embedding Model
Query
Upsert
Data
API gateway
Load balancer
Kafka stream processing
Data snapshots
Blob storage
Streaming updates
Batch updates
Vector index replicas
amazon web services
Google Cloud Platform

For chatbot


Data Warehouse
Embed text data
doc1, “Lorem ipsum …”
doc2, “Ut enim ad …”
doc3, “Duis aute …”
doc4, “Excepteur sin …”
doc1, [0.1, -0.7, 0.8, …]
doc2, [0.5, 0.1, -0.3, …]
doc3, [-0.4, -0.5, 0.9, …]
doc4, [-0.8, -0.2, 0.6, …]
AI Model
Embedding API
Upload
Query
Pinecone
Embed query
Application
Context embeddings
Prompt
Generated result
AI Model
Generative API

Important llibraries 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(documents))
```

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