MIDSHIPS
READY TO INTEGRATE

Intro to LLM

MARCH 2024

Say goodbye to the old way of
doing things. **Say hello to Midship**

Architecture Consideration

- Type of Document – PDF,WORD,WEB
- Data chunking strategy - Small/Large (Depends on Embedding Model)
- Embedding model selection – Open source/Closed source/Dimensions
- Vector Database – Open Source/Closed Source/SAAS
- Search retrieval strategy – Hybrid/Vector-Only-Searc
- Token and Context usage – Optimal Token Size per u
- Type of LLM – Open Source/ Closed Source (SAAS) / (SAAS) / On-premise

Type Of Document

What does the private knowledge base looks like ? Does it have PDF, Word, Web etc. Depending on the ty
doc we use different modules to chunk the data.

Data Chunking Strategy

Use Smaller/Larger chunks? What should be the logical end point to create the chunk? The chunk size wil
depend on the embedding models dimension.

Embedding Model Selection

This is the module which will encode text to vector. There are both closed and open source available. Nee
Selected.

Vector Database

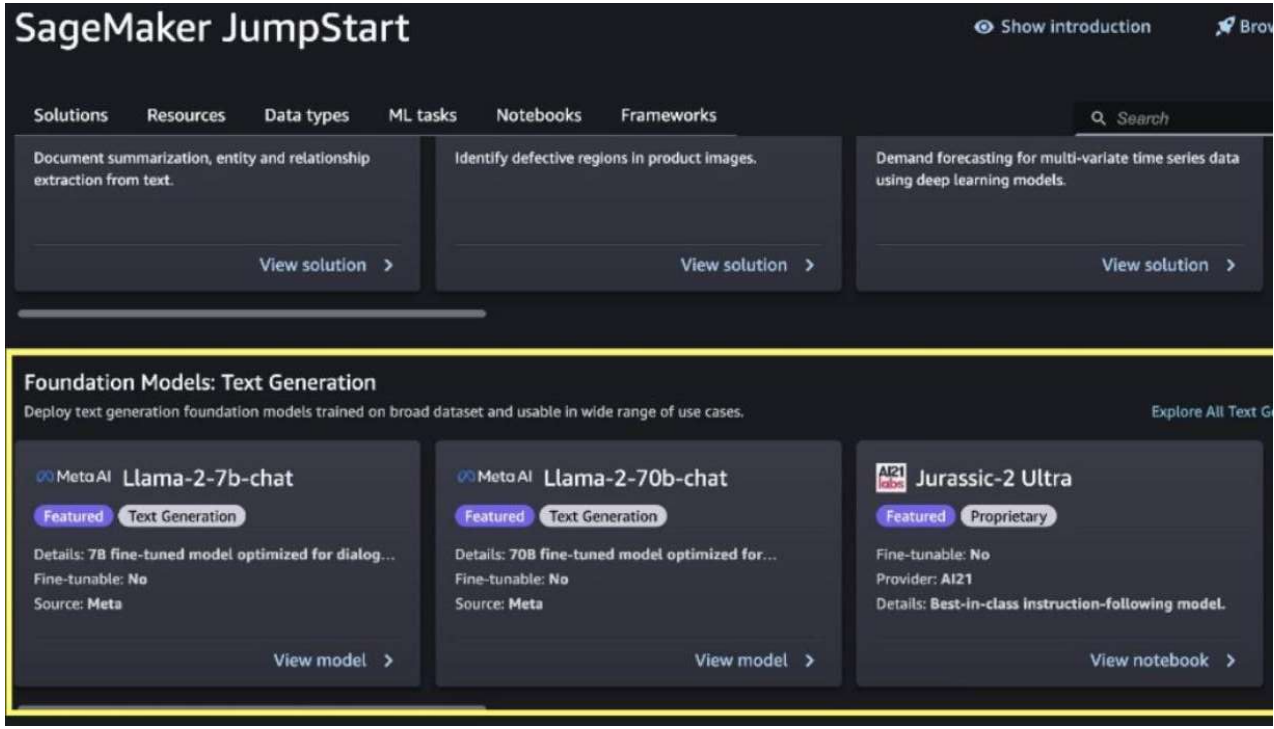
Are we going to use Open source/ Closed source/ SAAS

Type of LLM

Are we going to use open source/ Closed source/ SAAS / On-premise

AWS SAAS OFFERING FOR LLM/Vector Database

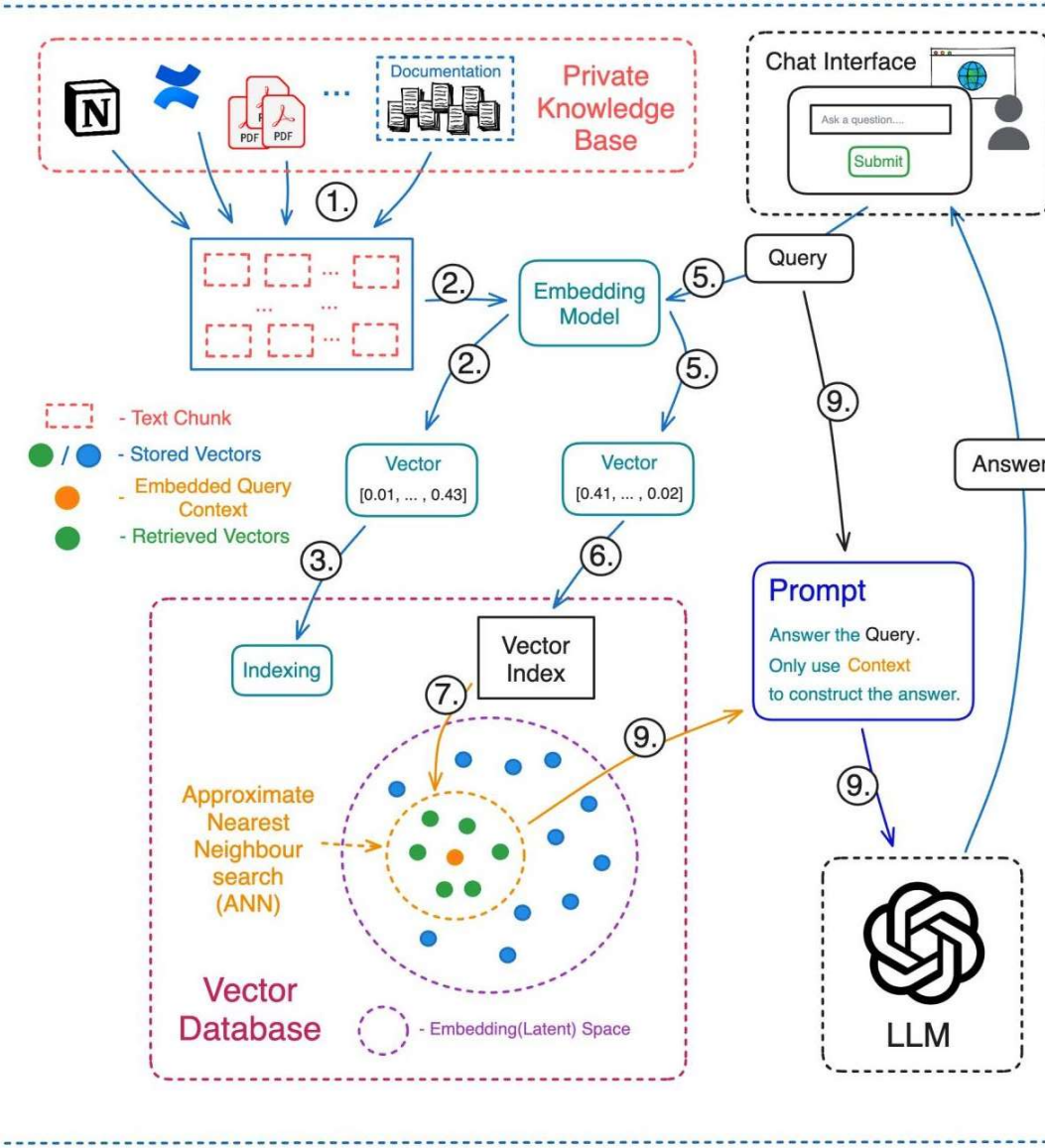
- Sagemaker – Provide various Open source LLM models as a SAAS offering
- Vector Database – Qdrant or similar SAAS offering available on AWS



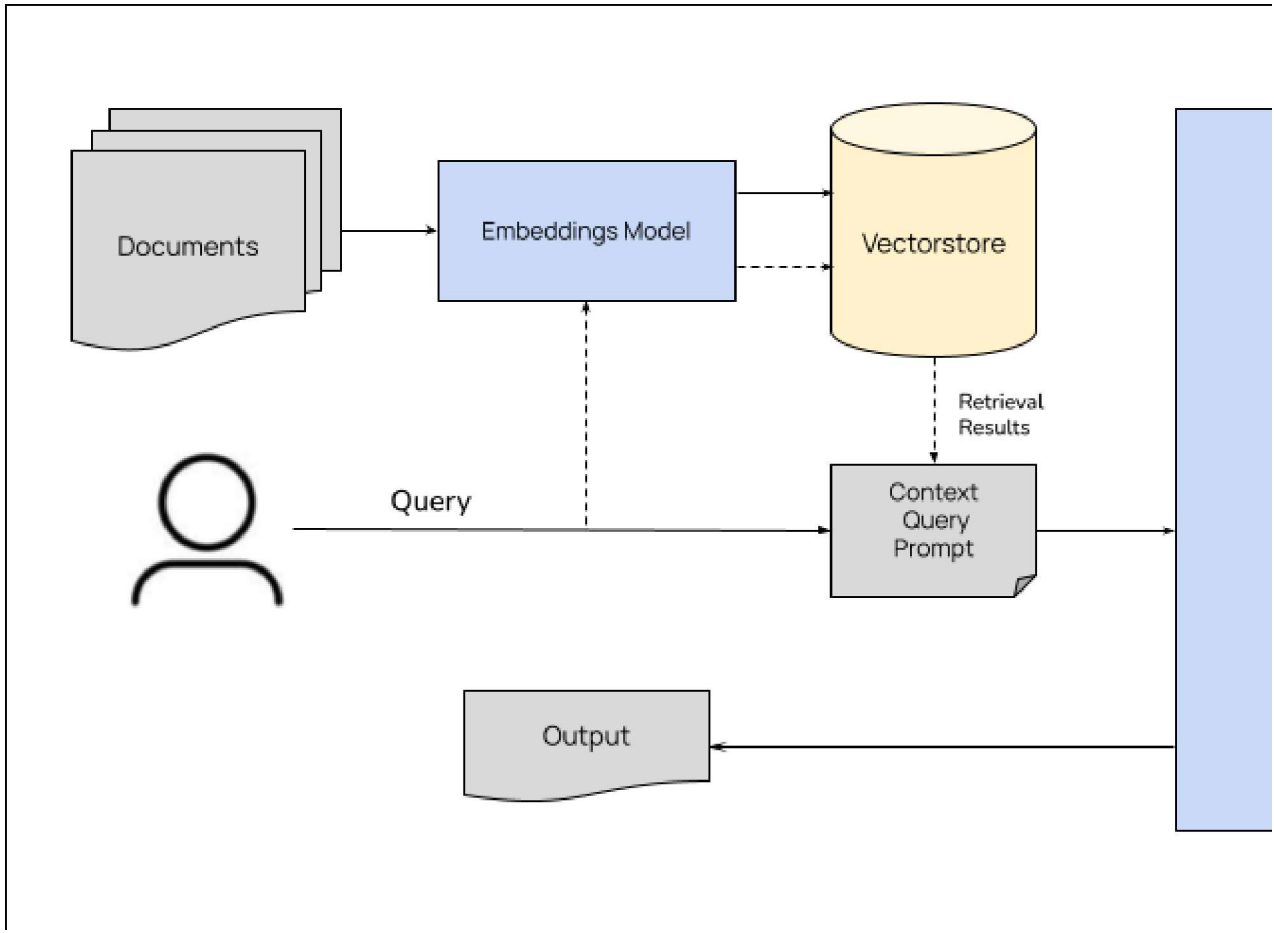
On-Premise Open Source LLM/Vector Database

- This VM size and specification will depend on t type of LLM (Size, Number of parameter)
- Typically for an LLM to provide good result for corporate enterprise should be at least of size [instance_type="ml.g5.12xlarge”]
- Apart from the LLM installation, VM also needs allocated for – Web application (front end - siz will depend on load); Embedding Model (depe on the type of embedding models and speed required – GPU’s might be needed to accelerat the processing); Vector Database (Size depend workload and size of private repository)

Overall Architecture



Frontend Flow



Backend Flow

Depending on the business requirement (CRON JOB TO BE TRIGGERED) - HOW UP-TODATE THE RESULT NEEDS TO BE ?

1. Define the **type of documents** to be used - PDF, WORD, Webpages. This will be your source of truth (**PRIVATE KNOWLEDGE BASE**). Whenever there is a new document added or an existing document modified, the job needs to to update the vector database.
2. Divide the document into smaller chunks. The **chunking strategy** depends on number of factors like - size of docu type of vector store, dimension of vector store etc.
3. Use an **Embedding Model** to convert/encode the data ch to vector.
4. Push the vector and its metadata (like actual document, source, size etc) into the vector store (to be used while performing user queries)

Frontend Flow

1. User query is passed to an embedding model to be encoded as a vector.
2. The embedded user query is passed to a **vector database**, where **vector search** is performed to compare the embeddings o user queries within the vectors of the database.
3. The original user query is then appended a LLM prompt with relevant context found/retrieved from the vector search.
4. The LLM generates a response with the newly provided context.