1.

LangChain`

- PyPDFLoader/ Unstructured PDFLoader.
- text_spiltter
- RetrievalQA

Sentence Transformer

- Create embedding usingall-minilm-l6-v2

FAISS

- Creates vector stores and performs similarity search i.e., querying.

OLLAMA

- Llama3 used as the LLM to generate response.

2.

AWS CLI

2.

FastAPI

FAST API

- Create an API endpoint for the RAG model created

ngrok

Ngrok

- API gateway to exposer local host to internet

3.

Amazon EC2

Streamlit

S3

- Used to store embeddings of documents

App exposed using public IP of EC2 instance

RAG application to answer questions related to document uploaded by the user.

This application is created to explore RAG and run it on Amazon Web Services to get familiar with its ecosystem.

The intention with this project was to create a RAG application completely on AWS platform. But as I could not avail enough resources to run a LLM on AWS in free tier plan, I had to resort to running the LLM on local machine and connect it to an EC2 instance. Though this is not scalable, it serves the purpose of understanding pipelines required to connect AWS to external APIs well.

Here we have created an application which has end user run point on EC2 instance, stored embeddings in S3, creation and querying of vector database and a LLM on a local machine to give back the response to end user.

Diagram:

In the diagram above Block labeled

'1' runs on the local machine,

'2' are connectors of local to AWS machine and

'3' runs on AWS.

The arrows show the flow of data from one service to another and the orange arrows show how the request from end user is passed to local machine. S3 component sends and receives data via AWS CLI, while it is not directly connected to EC2 instance.

Below is the brief description of components of the application created:

1. **LangChain:**

LangChain is a unified interface that makes it easy to create LLM applications using multiple opensource libraries maintained by it like PyPDFLoader, text splitter, RetreivalQA and many other APIs integrated with it. This saves the user from learning to use different APIs and integrating them manually.

LangChain allows user to create chains joining different parts of a LLM application like chunking of documents, embedding creation, storing vectors, querying the vector database , prompting LLM etc which when executed performs all the steps in a sequence.

In this application, loading of a document, chunking it and converting it into embeddings, creating a vector store, querying it and returning an answer from LLM are completely facilitated by LangChain framework

2. FastAPI:

FastAPI is a web framework that supports RESTful architecture and also provides extra features like automatic API documentation, asynchronous support, WebSocket capabilities etc.

This is used to create the API endpoints for interaction between client and server. Client being the EC2 machine and server being the local machine in our case.

3. Ngrok:

This is used to create a reverse proxy to a port on the machine ngrok is run on. Allowing user to expose ports to web traffic securely and with minimal changes in the local network configuration.

4. Docker:

This is used to build, deploy, run and manage applications using containers which contain all the necessary tools like environment variables, libraries, system configuration files, code etc., to run an application on any machine.

Docker requires us to build the DockerFile which contains the commands telling which image to copy to build the container, which files to copy to container from local folder and which command to run in bash once the docker container is run. Thus, we can set it up so the streamlit application runs as soon as the docker container is run.

In order to run an application in Amazon EC2 instance the application built locally is containerized and the docker image is built and pushed to docker hub. The uploaded docker image is pulled in EC2 instance and run. Furthermore, while running the docker app, we can connect the port on which the app is running in container to any port of the machine on which container is run, allowing us to access the port from internet through the machine's IP address.

5. S3:

S3 is an object storing service which allows us to store data as objects in buckets. This also integrates into other AWS service.

This is used in this project to store the embeddings created for a file so that when the same file is uploaded, we don't need to create embeddings again. The embeddings are saved to S3 from local using AWS CLI which requires an IAM user creation and getting AWS secret key to be given as environment variables or used in code directly.

6. EC2:

This is a virtual server provided by Amazon which here is used to run the streamlit app docker container. To access the stremlit application in this EC2 instance, you have to create a new security group and set custom TCP allowing traffic from 0.0.0.0 and the required port.

GITHUB LINK : https://github.com/wasif20875/RAG_AWS