Building a RAG pipeline

Estimated Duration: 45 minutes

Introduction

Retrieval-Augmented Generation (RAG) is a technique in generative AI that combines the strengths of language models and information retrieval systems. RAG first retrieve relevant information from a knowledge base, then use this retrieved information to inform and guide the generation of relevant and coherent text. This approach allows the model to draw upon external knowledge beyond what is contained in its training data, leading to more informative and contextual responses.

If you are already using Amazon Bedrock, it offers native RAG capabilities with it's knowledge base feature. Sometimes, you want more control and flexibility with how you create and store your vector data. This lab enables you to implement your custom knowledge base using Serverless services.

Use Case:

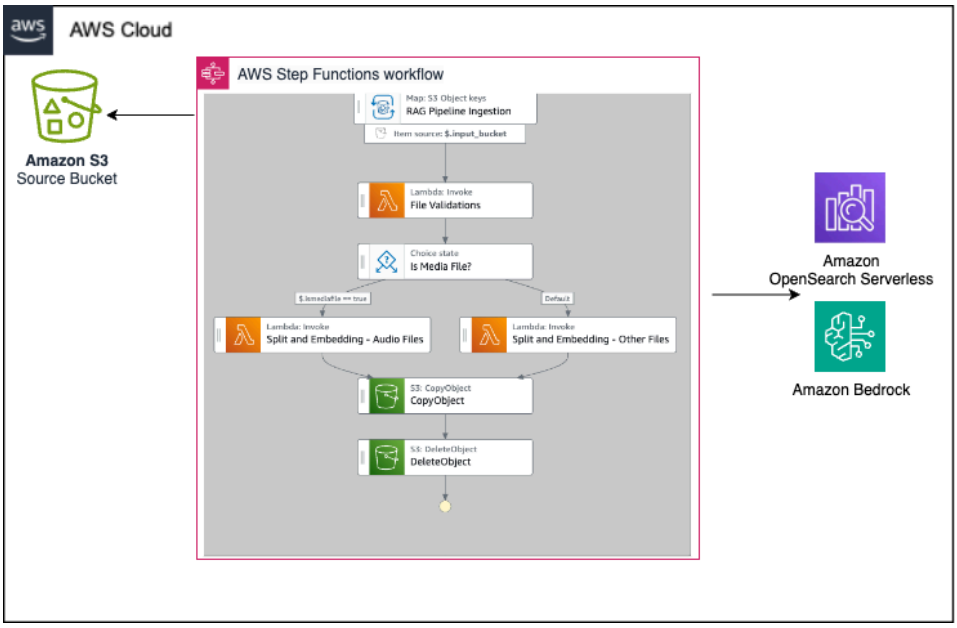
Imagine you have a vast repository of documents covering a specific domain or industry, stored in various file formats like PowerPoint, PDF, and audio. To find the information you need, you currently have to search each document individually, a time-consuming and inefficient process.

The goal of this lab is to develop a solution that allows users to seamlessly interact with this knowledge base and retrieve the relevant information they're interested in, without the need for manually searching through each file.

Architecture:

Data Ingestion:

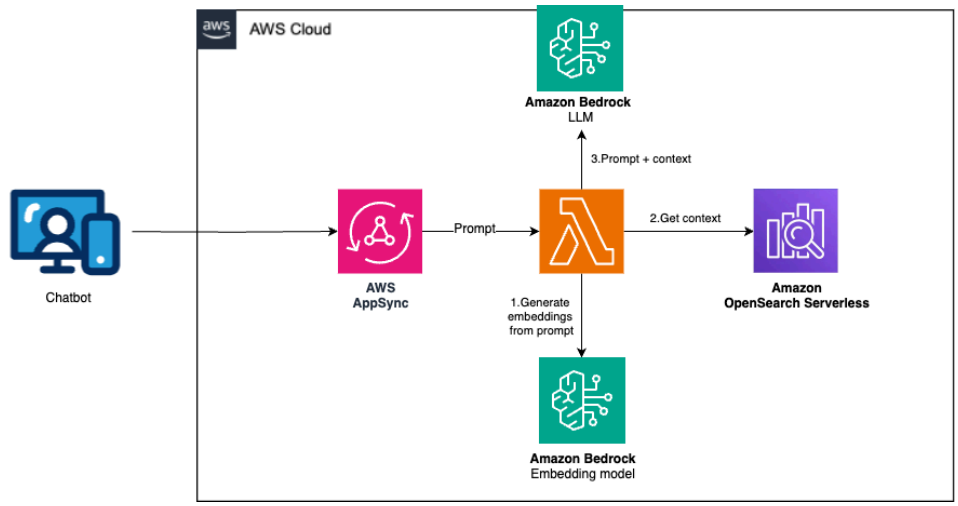
To enable RAG, you will first need to make the knowledge base available in embeddings, stored in a vector database - a process known as 'data ingestion'. You will start by building a data ingestion pipeline implemented as an AWS Step Functions workflow, which will read data from an S3 bucket, apply relevant filters, use an embedding model to generate vector representations of the data, and store the vectors in an Amazon OpenSearch Serverless database. Once this data ingestion process is complete, the workflow will archive the original files for future reference.



Data Retrieval or Inference:

To respond to the user's query, the system will first create embeddings based on the user's input prompt. It will then use these embeddings to retrieve the relevant details from the Amazon OpenSearch Serverless database, which stores the vector representations of the knowledge base. Finally, the system will send all the retrieved relevant files to a Large Language Model, which will then generate the response to be returned to the user.

> This solution is prebuilt and deployed in the account. You will use the chatbot UI to test the RAG functionality.



What you will accomplish

- Learn how to build a data ingestion pipeline that generates vector embedding of knowlege base at large scale.

Services in this module

- Amazon S3 - Object storage built to retrieve any amount of data from anywhere.
- AWS Step Functions - Visual workflows for distributed applications.
- AWS Lambda - Serverless compute service; Run code without thinking about servers or clusters.
- Amazon Bedrock - The easiest way to build and scale generative AI applications with foundation models.
- Amazon OpenSearch - Securely unlock real-time search, monitoring, and analysis of business and operational data.

- AWS AppSync - Connect apps to data and events with secure, serverless, and performant GraphQL and Pub/Sub APIs.

LLMs in this module

- Anthropic Claude 3 Haiku - Used for inference.
- Amazon Titan Embeddings V2 - Used to convert text to embedding data.

What's included in this module

Data ingestion and retrieval code in the following AWS Lambda functions:

- **rag-pipeline-ingestion-file-validation**: Lambda function to identify the format of the file such as audio, PPT, video etc.
- **rag-pipeline-ingestion-process-files**: Lambda function to process non-media files. It will read the file, split the data, embed the data, and index the information in a vector store.
- **rag-pipeline-ingestion-process-audio-files**: Lambda function to process media files. It will read the file, split the data, embed the data, and index the information in a vector store.
- **rag-pipeline-data-retrieval**: Lambda function connected to AppSync api to get domain specific information. Based on the prompt it will retrieve the data from the vector store and LLM.

Knowledge base for RAG in the Amazon OpenSearch Serverless Collection vector store:

- **rag-pipeline-collection**: A OpenSearch serverless collection will be used as the knowledge base.

Previous Next