

University of Toronto  
School of Continuing Studies  
SCS 3547 Intelligent Agents

# Knowl-EDge

Serverless Learning Backend Service  
with Knowledge Base, Lambda and API Gateway

Student: Arjie Cristobal  
Professor: Larry Simon  
March 2024

## **Table of Contents**

<b>Objective</b>	<b>3</b>
Introduction	3
Purpose of Report	3
<b>Concept and Architecture</b>	<b>3</b>
Data Ingestion Workflow	3
Retrieval-Augmented Generation Process	4
Amazon Bedrock Knowledge Base	4
Knowl-EDge Architecture	5
Agents for Amazon Bedrock	5
<b>Conclusion</b>	<b>5</b>
Recommendation	5
<b>Appendix A. Concept and Architecture</b>	<b>6</b>
Figure 1 - Data Ingestion Workflow	6
Figure 2 - RAG Process	6
Figure 3 - Amazon Knowledge BaseFigure 4 - Architecture	7
<b>Appendix B. AWS Console</b>	<b>8</b>
Figure 1 - Identity Access Management (IAM)	8
Figure 2 - Amazon S3	8
<b>Appendix C. Code and Output</b>	<b>10</b>
Figure 1 - Lambda Function Code	10
Figure 2 - Test Output	11
<b>References</b>	<b>11</b>

# Objective

## Introduction

In today's fast paced technological landscape, developers and technical support face a common challenge: the overwhelming volume of information scattered across our documents from our different platforms. Finding relevant and accurate information is crucial for their productivity and problem-solving abilities. The manual scanning of each document often falls short and leads to inefficiency and frustration.

## Purpose of Report

The primary objective of this study is to develop Knowl-EDge, a Serverless Backend Learning Service tailored for developers and technical support.

The study aims to take advantage of Generative AI with Large Language Models (LLMs), including RAG (Retrieve-Augmented Generation) and integrate Agents for Amazon Bedrock Knowledge Base to enhance the overall functionality and effectiveness of the service.

Consumers of this new API will be able to efficiently inquire about and access information related to the applications within the organisation.

# Concept and Architecture

## Data Ingestion Workflow

The data ingestion workflow has four steps. The first step is to identify the data source and load all contents in that data source. In this study, we load all the documents to S3. The next step is to transform the document and that is through chunking. This process breaks down the document into 300 characters per chunk. Once the document is broken down into smaller

chunks it will be followed by the step that creates the vector embeddings using Amazon Titan Text Embedding. And finally, the last step is to store the vector embeddings to a vector store or database using Amazon OpenSearch. (See Appendix B, Figure 1 - Data Ingestion Workflow)

## Retrieval-Augmented Generation Process

Large Language Models are trained with large data and can answer any question using its memory. However the knowledge that LLMs have, depends on the cutoff date when they are last trained. When an LLM was asked about information after the cutoff date the model will most likely hallucinate. The researchers from Meta found that providing relevant information to the LLM improves the performance of the model. The process of providing information to LLM will eventually be known as Retrieval-Augmented Generation or in short - RAG.

The RAG Process starts with the user sending a Prompt to the model. The Prompt will undergo Text Embedding, a process that converts the characters into its numerical representation. Once the prompt is converted to numbers, a Similarity Search will happen and retrieve the relevant information in a Vector Store. The process of retrieval and returning a response or Context is the Retrieval in RAG.

The Context returned will then be augmented to the original Prompt. This is the Augment process of RAG.

The augmented Prompt with Context will then be passed to a Foundation Model to get the answer. This retrieval of the answer is the Generate portion of RAG. (See Appendix A, Figure 2 - RAG Process)

## Amazon Bedrock Knowledge Base

The Amazon Bedrock Knowledge Base (KB) automates all the four steps of Data Ingestion Workflow. The setting-up of Data Ingestion Workflow using Amazon Bedrock KB can be done through console or SDK. With Amazon Bedrock KB, the users just need to upload the documents to the data source and Knowledge Base is going to take care of all the four steps of Data Ingestion Workflow. (See Appendix A, Figure 3 - Amazon Knowledge Base)

## Knowl-EDge Architecture

The architecture starts with the Client sending a Prompt to a URL provided by Amazon API Gateway() then the API Gateway will pass the prompt as an Event to AWS Lambda(). The Lambda function will invoke the RetrieveAndGenerate API () which is provided by Knowledge Base and will return a response which will be the contextual information. This Context will then be passed to the Claude Foundation model and returned a response. The said response from the model will be sent back by the Lambda function to the user via the API Gateway. (See Appendix A, Figure 4 - Knowl-EDge Architecture)

## Agents for Amazon Bedrock

The agents for Amazon Bedrock offer the ability to build and configure autonomous agents in the Knowl-EDge application. The agent helps end-users complete actions based on organisation data and user input. The agent orchestrates interactions between foundation models (FMs), data source, software applications, and user conversations.

## Conclusion

In summary, our study successfully achieved its primary objective: the development of Knowl-EDge, a Serverless Backend Learning Service designed for developers and technical support. By using the power of Generative AI with LLM including RAG, we are able to improve the functionality and effectiveness of the service. The integration of Agents for Amazon Bedrock Knowledge Base ensures that consumers of this API can efficiently inquire about and access crucial information about our applications.

## Recommendation

The study only used Claude, Anthropic's Large Language Model (LLM), which prompted the need for a foundation model selection process. To improve the results, we also need to consider employing prompt engineering techniques. Additionally, evaluating other vector stores that align better with the use case is highly desirable.

## Appendix A. Concept and Architecture

Figure 1 - Data Ingestion Workflow

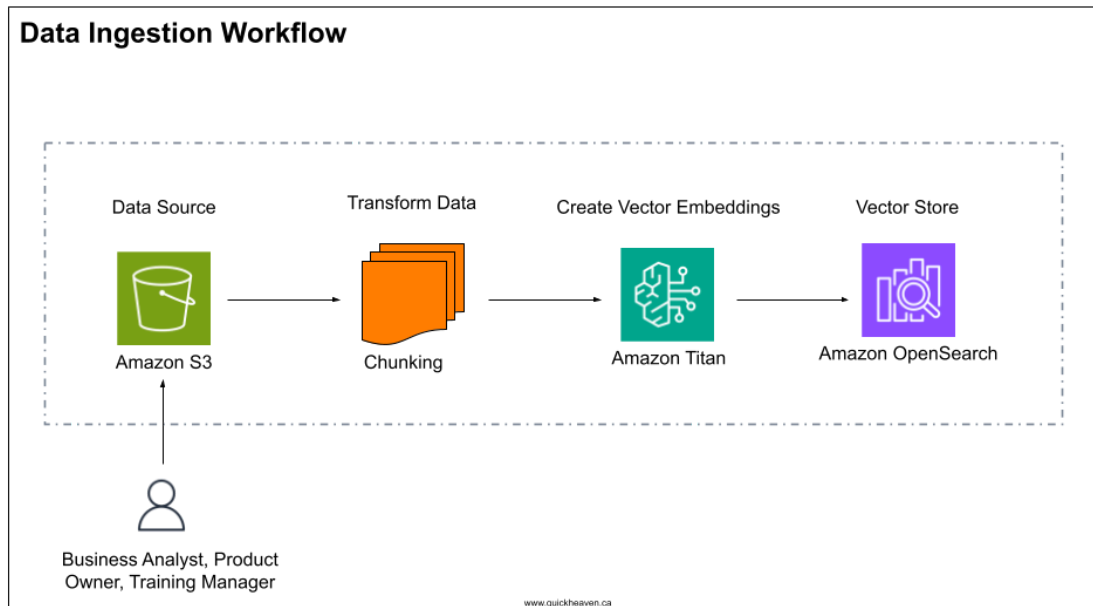


Figure 2 - RAG Process

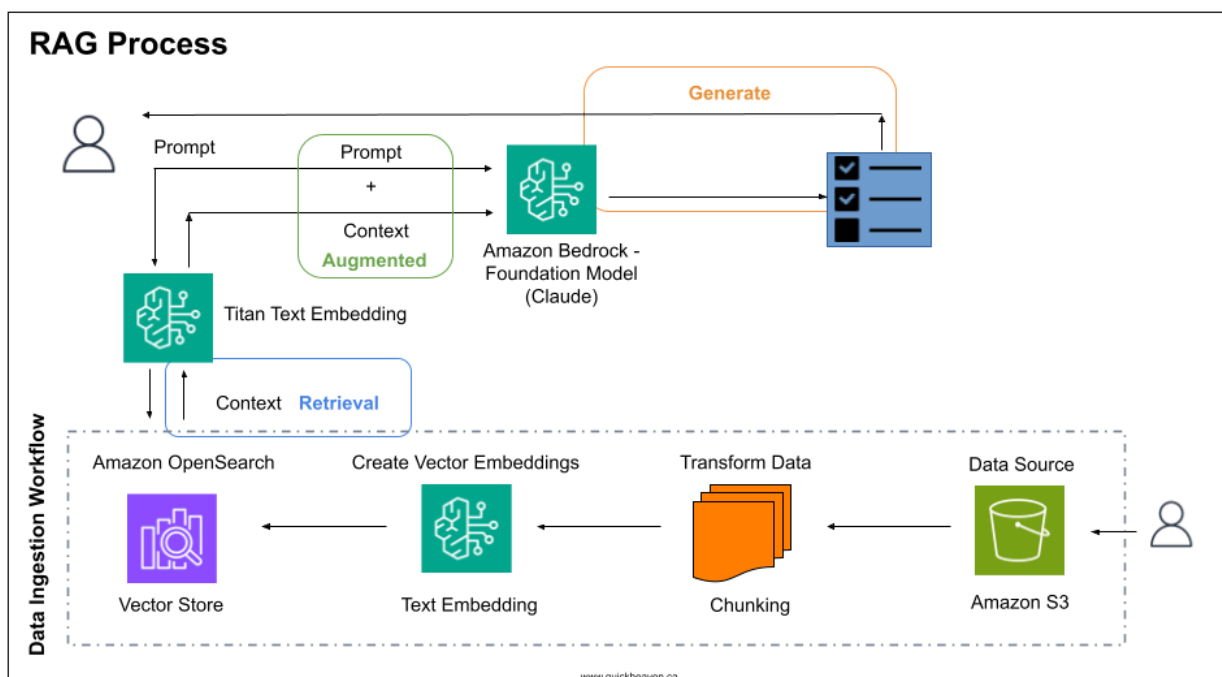


Figure 3 - Amazon Knowledge Base

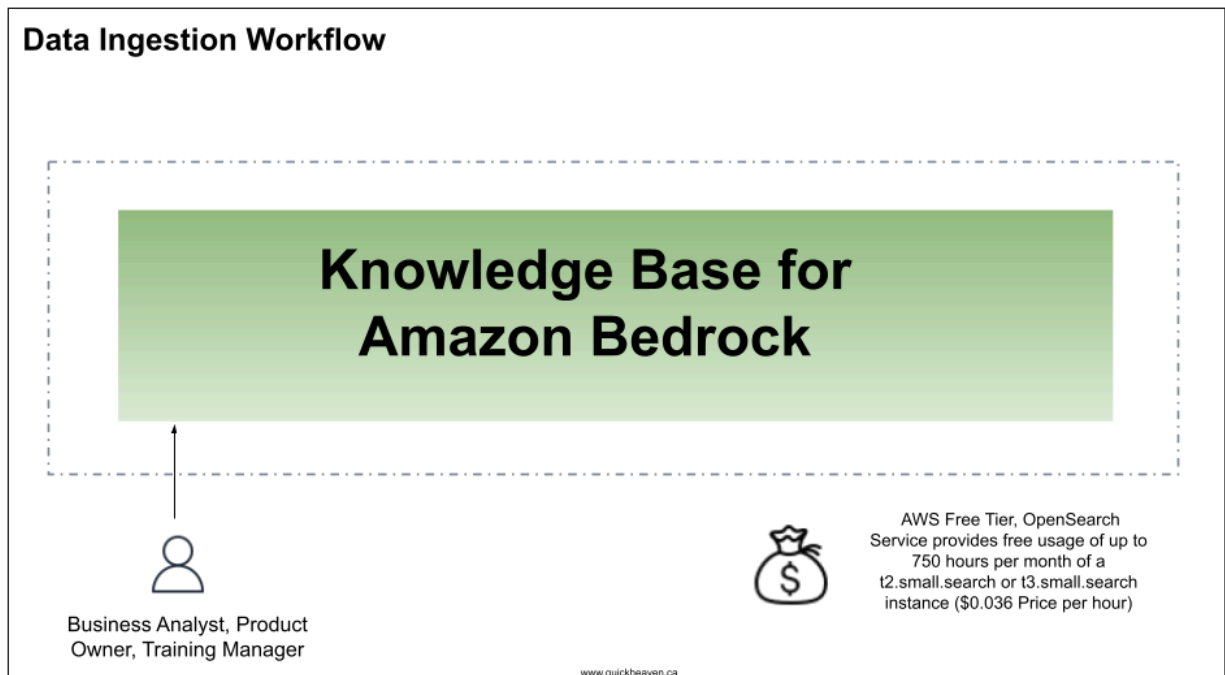
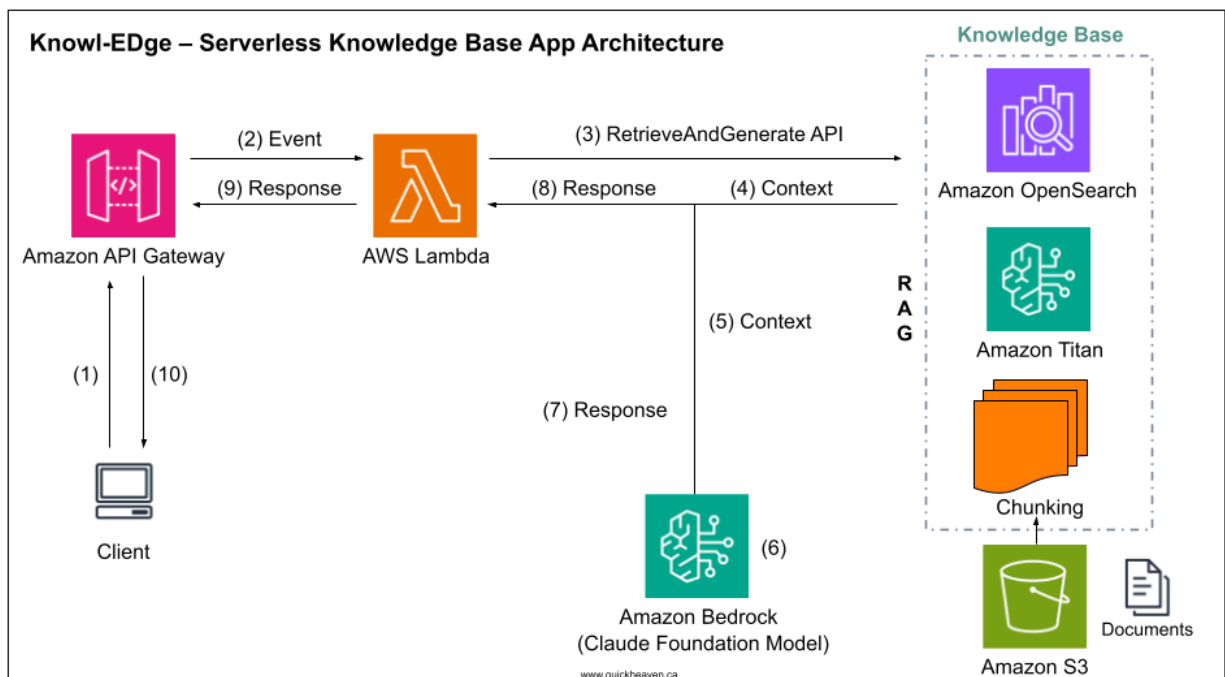


Figure 4 - Architecture



# Appendix B. AWS Console

Figure 1 - Identity Access Management (IAM)

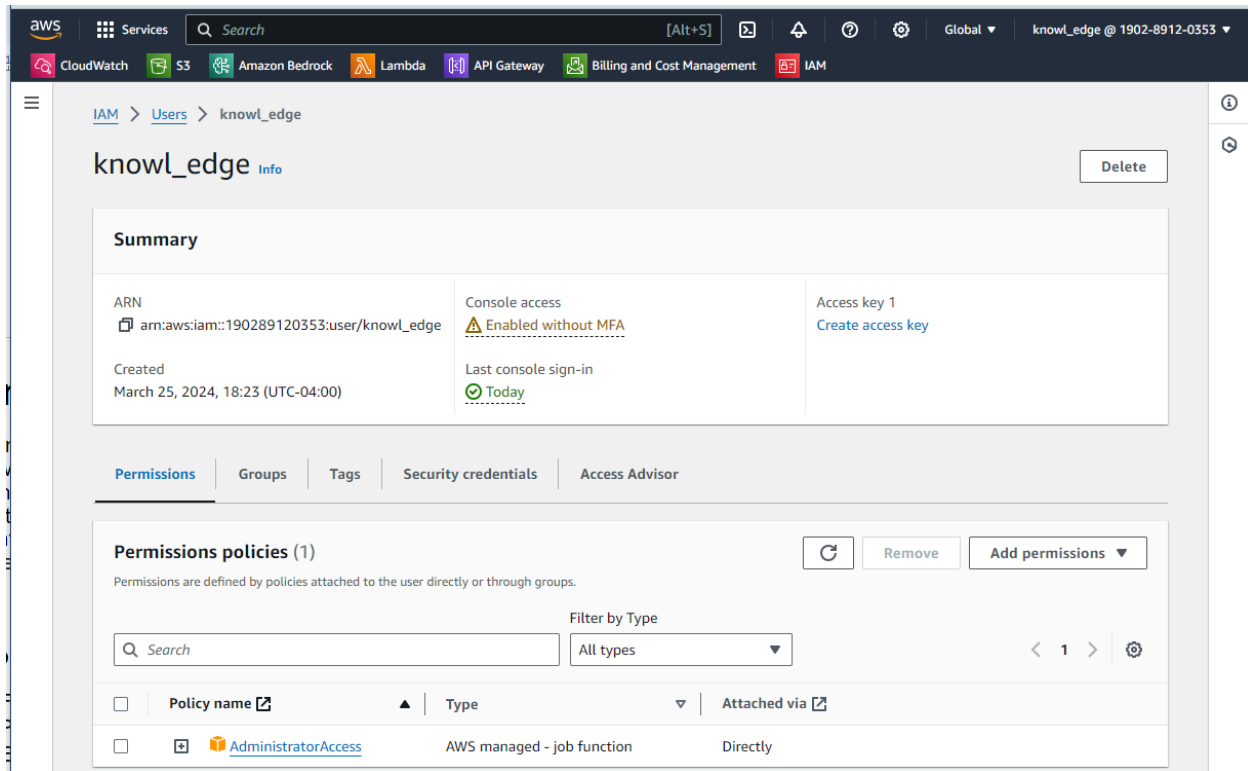


Figure 2 - Amazon S3

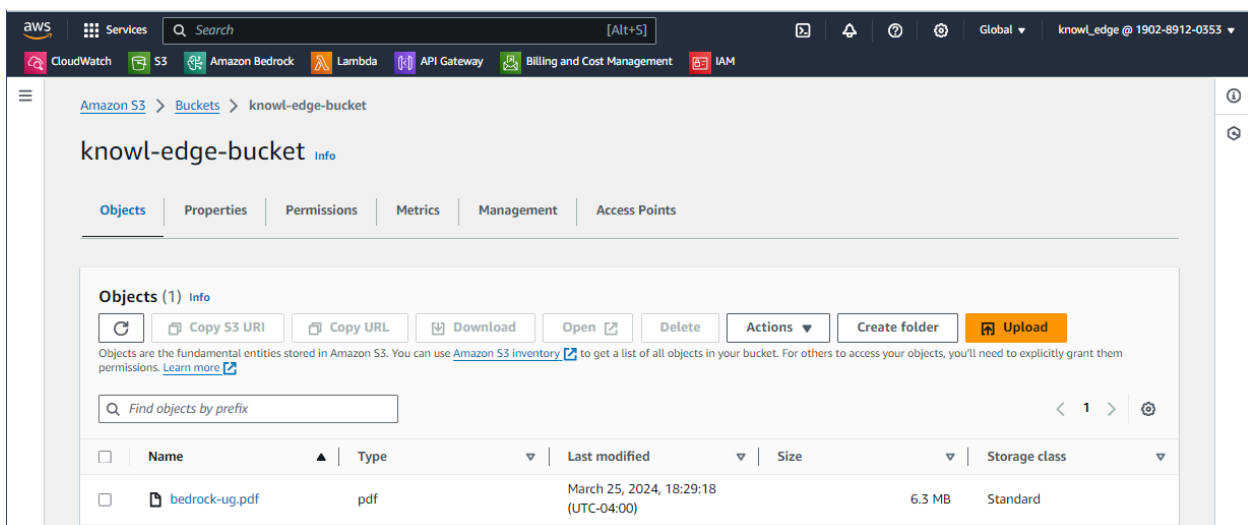




Figure 3 - Amazon Bedrock Knowledge Bases

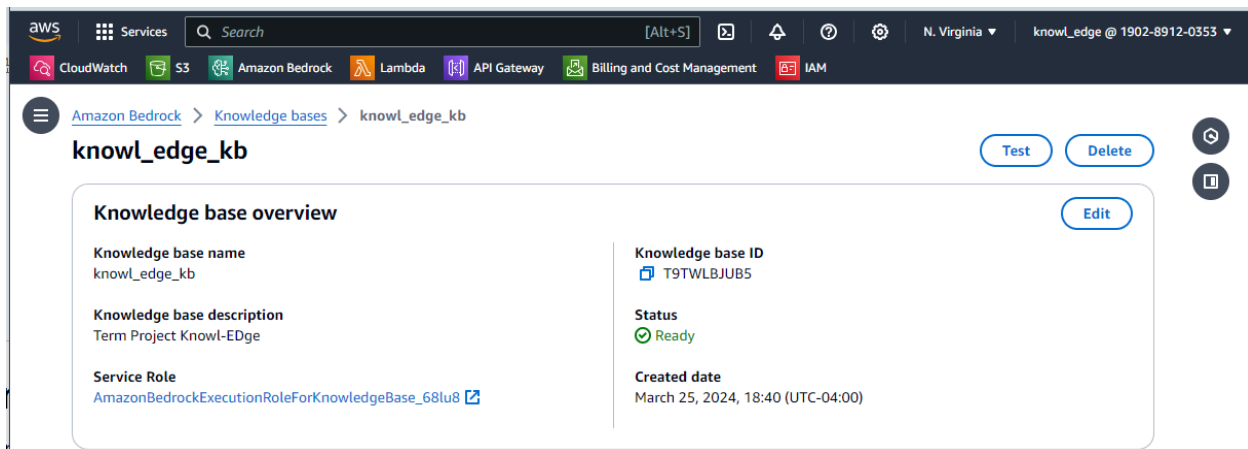


Figure 4 - AWS Lambda

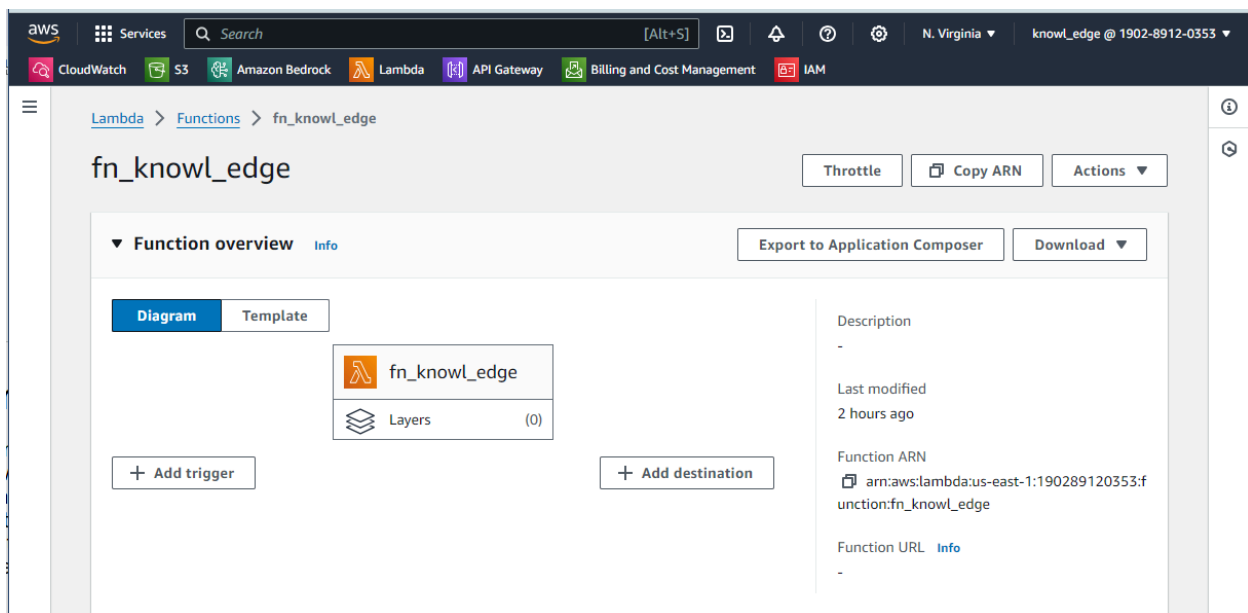
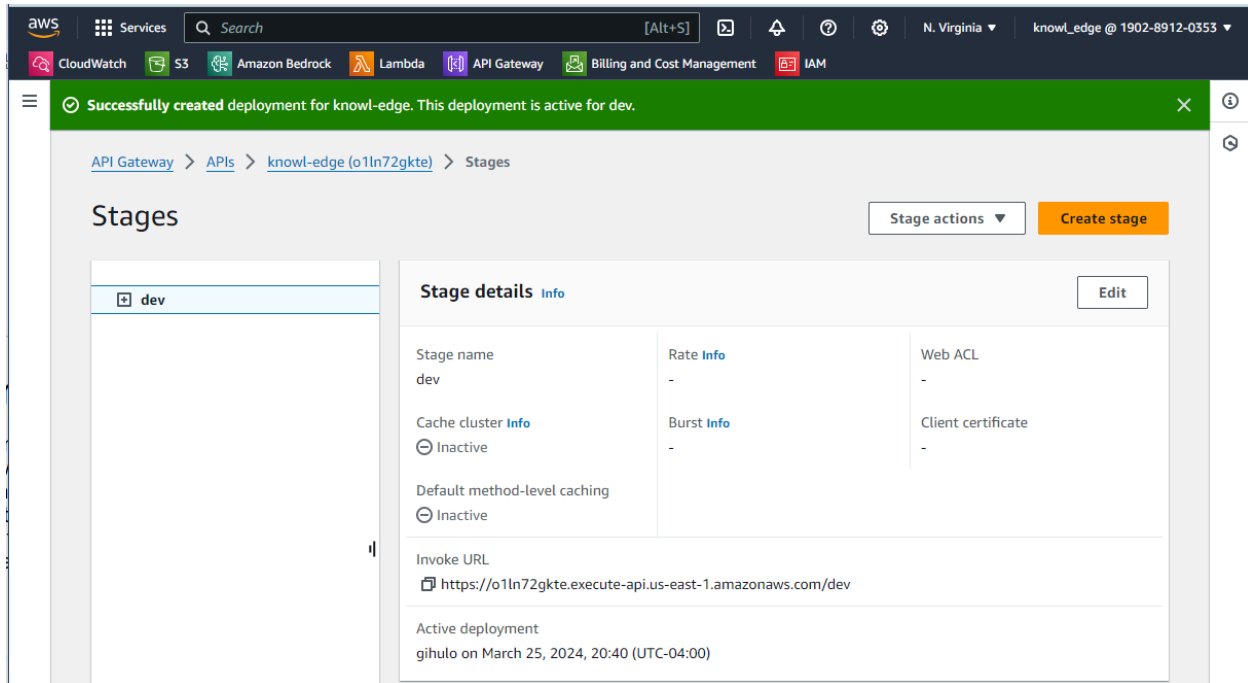


Figure 5 - API Gateway



## Appendix C. Code and Output

Figure 1 - Lambda Function Code

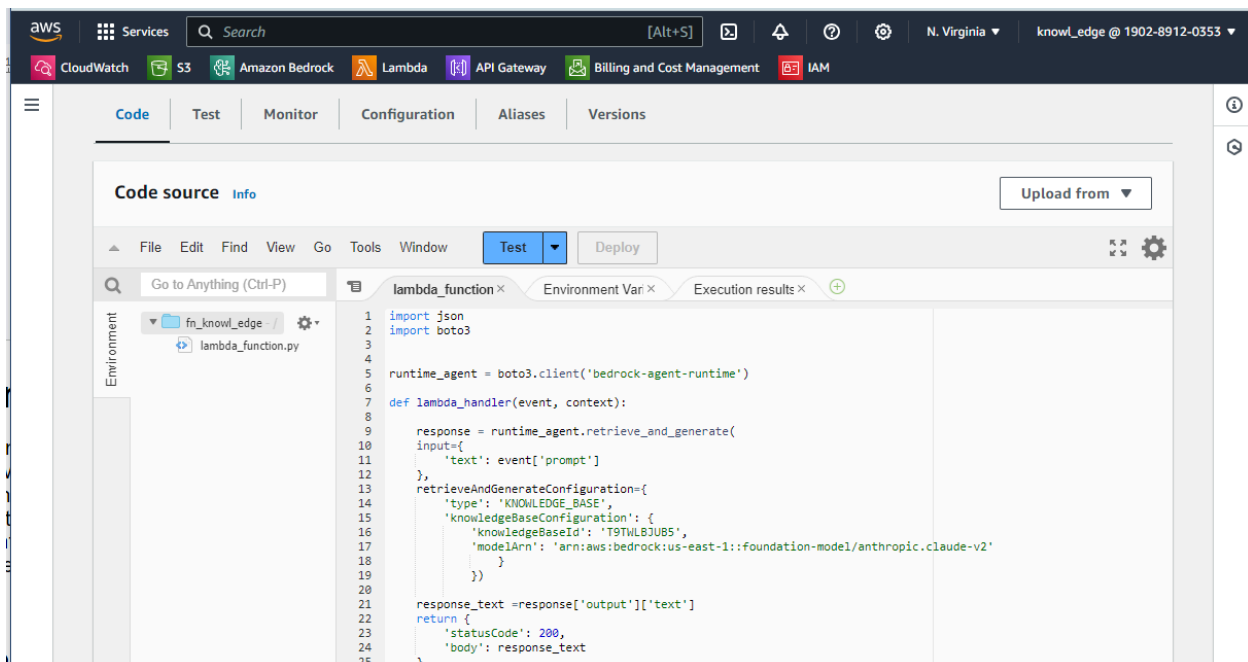



Figure 2 - Test Output

 / - GET method test results		
Request	Latency	Status
/?prompt="What is Foundation Model?"	6208	200
Response body		
<pre>{"statusCode": 200, "body": "A foundation model (FM) is an AI model with a large number of parameters and trained on a massive amount of diverse data. A foundation model can generate a variety of responses for a wide range of use cases."}</pre>		
Response headers		
<pre>{   "Content-Type": "application/json",   "X-Amzn-Trace-Id": "Root=1-66021d75-4d0c490591d102a9092c7e2d;Parent=1fa338d8b25bb48f;Sampled=0;lineage=6fe4fe36:0" }</pre>		

## References

1. Amazon S3  
<https://aws.amazon.com/s3/>
2. Amazon Titan  
<https://aws.amazon.com/bedrock/titan/>
3. Amazon Open Search  
<https://aws.amazon.com/what-is/opensearch/>
4. Retrieval-Augmented Generation Process  
<https://llmstack.ai/blog/retrieval-augmented-generation>
5. Amazon Knowledge Base  
<https://aws.amazon.com/bedrock/knowledge-bases/>
6. Amazon API Gateway  
<https://aws.amazon.com/api-gateway/>
7. AWS Lambda  
<https://aws.amazon.com/lambda/>

8. Agents for Bedrock Runtime

<https://docs.aws.amazon.com/bedrock/latest/userguide/agents.html>

9. RetrieveAndGenerateAPI

[https://docs.aws.amazon.com/bedrock/latest/APIReference/API\\_agent-runtime\\_RetrieveAndGenerate.html](https://docs.aws.amazon.com/bedrock/latest/APIReference/API_agent-runtime_RetrieveAndGenerate.html)

10. Anthropic Claude

<https://aws.amazon.com/bedrock/claude/>