



Serverless x GenAI BKK Workshop

Generative AI at scale: Serverless workflows for enterprise-ready apps

[Pre-requisite] Enable foundation model access in Amazon Bedrock

[Pre-requisite] Configuring the front-end application

► Playground

▼ Use cases

▼ Building a RAG pipeline

Testing the need for RAG

Building the RAG Pipeline

Running the Pipeline

Testing the RAG Inference

▼ AWS account access

[Open AWS console \(us-west-2\)](#)

[Get AWS CLI credentials](#)

Exit event

[Event dashboard](#) > [Use cases](#) > [Building a RAG pipeline](#) > High level Code Walkthrough

High level Code Walkthrough

The Lambda functions uses [Llama Index](#) to create and store vector index. **Llama Index** is a leading data framework for building LLM applications. It can load data from variety of data sources, index the data and query the data stored in indexes. In this section, you will do a walk through of the code to understand some core functionalities

Lambda function code

```
1  from llama_index.core import (SimpleDirectoryReader, StorageContext,
2                                VectorStoreIndex)
3  from llama_index.core.settings import Settings
4  from llama_index.embeddings.bedrock import BedrockEmbedding
5  from llama_index.llms.bedrock import Bedrock
6  from llama_index.vector_stores.opensearch import (OpensearchVectorClient,
7                                                    OpensearchVectorStore)
8  from opensearchpy import AsyncHttpConnection, AWSV4SignerAsyncAuth
9  from s3fs import S3FileSystem
10
11
12  # http endpoint for your OpenSearch cluster
13  endpoint = os.getenv("OS_COLLECTION_ENDPOINT")
14  # index to demonstrate the VectorStore impl
15  idx = os.getenv("OS_INDEX_NAME")
16
```



```

17 # model used to create embedding
18 embed_model = BedrockEmbedding(model_name=os.getenv("BEDROCK_EMBEDDING_MODEL"))
19 Settings.embed_model = embed_model
20
21 def lambda_handler(event, context):

```


© 2008 - 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

```


24     s3_fs = S3FileSystem(anon=False, endpoint_url=None)
25     if event.get("Key"):
26         files = bucket_name + event.get("Key")
27     ### LOAD THE FILES USING LLAMA INDEX
28     documents = SimpleDirectoryReader(input_files=[files], fs=s3_fs,
29         recursive=True).load_data()
30
31     ### Authenticate using SIGV4
32     auth = AWSV4SignerAsyncAuth(credentials, region, service)
33     client = OpensearchVectorClient(
34         endpoint = endpoint,
35         index = idx,
36         embedding_field= embedding_field,
37         text_field=text_field,
38         dim = 1536,
39         engine="faiss",
40         http_auth=auth,
41         use_ssl=True,
42         verify_certs=True,
43         connection_class=AsyncHttpConnection
44     )
45
46     # initialize vector store
47     vector_store = OpensearchVectorStore(client)
48     res = "Indexed"
49
50     try:
51         storage_context = StorageContext.from_defaults(vector_store=vector_store)

```

```
52
53     index = VectorStoreIndex.from_documents(
54         documents=documents, storage_context=storage_context
55     )
56     logger.info("Indexing completed")
57
58     return {
59         'status': res,
60         'message': "Successfully indexed the file",
61         'processedfiles': files,
62         'archive': {
63             'sourcebucket': files.split('/')[0],
64             'archivebucket': archive_bucket,
65             'filename': files.split('/')[1]
66         }
67     }
68 except Exception as e:
69     logger.error(f"Error: {e}")
70     return {
71         'status': 'Error',
72         'message': 'Error occurred while indexing the files',
73         'errorfiles': files
74     }
```

1. Walk through the code to understand different functionalities.
2. The code uses LLama index document reader to load the data
3. Llama index then uses Bedrock titan model to create the vector embeddings
4. You will also notice [SIGV4](#)  auth with OpenSearch vector store and the storage of data in the database

Packaging for Lambda function

When you bundle the Llama index libraries with your code, you will likely exceed the Lambda zip archive format 250MB limitation. Like we did in the workshop, we recommend you to use [container format to bundle the Lambda code](#) .

[Previous](#)[Next](#)