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

```
from llama_index.core import (SimpleDirectoryReader, StorageContext,
                              VectorStoreIndex)
from llama_index.core.settings import Settings
from llama_index.embeddings.bedrock import BedrockEmbedding
from llama_index.llms.bedrock import Bedrock
from llama_index.vector_stores.opensearch import (OpensearchVectorClient,
                                                  OpensearchVectorStore)
from opensearchpy import AsyncHttpConnection, AWSV4SignerAsyncAuth
from s3fs import S3FileSystem


# http endpoint for your OpenSearch cluster
endpoint = os.getenv("OS_COLLECTION_ENDPOINT")
# index to demonstrate the VectorStore impl
idx = os.getenv("OS_INDEX_NAME")

```

```
# model used to create embedding
embed_model = BedrockEmbedding(model_name=os.getenv("BEDROCK_EMBEDDING_MODEL"))
Settings.embed_model = embed_model

def lambda_handler(event, context):
```

```
        s3_fs = S3FileSystem(anon=False, endpoint_url=None)
        if event.get("Key"):
            files = bucket_name + event.get("Key")
        ### LOAD THE FILES USING LLAMA INDEX
        documents = SimpleDirectoryReader(input_files=[files], fs=s3_fs,
            recursive=True).load_data()

        ### Authenticate using SIGV4
        auth = AWSV4SignerAsyncAuth(credentials, region, service)
        client = OpensearchVectorClient(
            endpoint = endpoint,
            index = idx,
            embedding_field= embedding_field,
            text_field=text_field,
            dim = 1536,
            engine="faiss",
            http_auth=auth,
            use_ssl=True,
            verify_certs=True,
            connection_class=AsyncHttpConnection
        )

        # initialize vector store
        vector_store = OpensearchVectorStore(client)
        res = "Indexed"

        try:
            storage_context = StorageContext.from_defaults(vector_store=vector_store
```

```python

            index = VectorStoreIndex.from_documents(
                documents=documents,storage_context=storage_context
            )
            logger.info("Indexing completed")

            return {
                'status': res,
                'message': "Successfully indexed the file",
                'processedfiles': files,
                'archive':{
                    'sourcebucket': files.split('/')[0],
                    'archivebucket': archive_bucket,
                    'filename': files.split('/')[1]
                }
            }
        except Exception as e:
            logger.error(f"Error: {e}")
            return {
                'status': 'Error',
                'message': 'Error occurred while indexing the files',
                'errorfiles': files
            }
```

1. Walk through the code to understand different functionalities.
2. The code uses LLama index document reader to load the data
3. Llambda 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