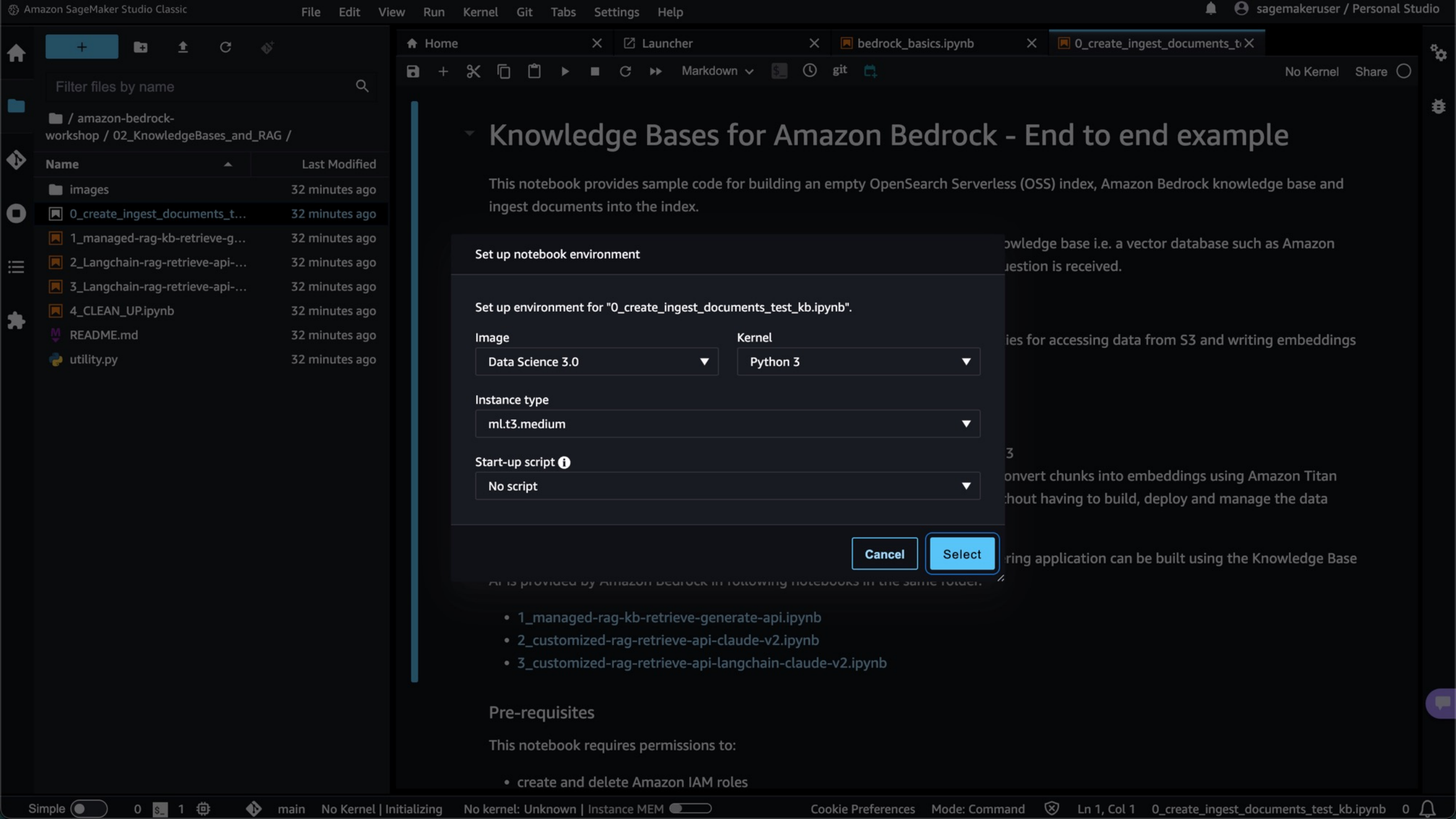
Use RAG to improve responses in generative AI applications

Michael Lin

Sr. Solutions Architect
Amazon Web Services

Hands-On Lab


Amazon SageMaker Studio Classic
File Edit View Run Kernel Git Tabs Settings Help
sagemakeruser / Personal Studio
Filter files by name
/ amazon-bedrock-workshop / 02_KnowledgeBases_and_RAG /
Name
Last Modified
images 32 minutes ago
0_create_ingest_documents_t... 32 minutes ago
1_managed-rag-kb-retrieve-g... 32 minutes ago
2_Langchain-rag-retrieve-api-... 32 minutes ago
3_Langchain-rag-retrieve-api-... 32 minutes ago
4_CLEAN_UP.ipynb 32 minutes ago
README.md 32 minutes ago
utility.py 32 minutes ago
Home
Launcher
bedrock_basics.ipynb
0_create_ingest_documents_t
Markdown
git
No Kernel
Share
Knowledge Bases for Amazon Bedrock - End to end example
This notebook provides sample code for building an empty OpenSearch Serverless (OSS) index, Amazon Bedrock knowledge base and ingest documents into the index.
Set up notebook environment
Set up environment for "0_create_ingest_documents_test_kb.ipynb".
Image
Data Science 3.0
Kernel
Python 3
Instance type
ml.t3.medium
Start-up script
No script
Cancel
Select
1_managed-rag-kb-retrieve-generate-api.ipynb
2_customized-rag-retrieve-api-claude-v2.ipynb
3_customized-rag-retrieve-api-langchain-claude-v2.ipynb
Pre-requisites
This notebook requires permissions to:
create and delete Amazon IAM roles
Simple
main
No Kernel | Initializing
No kernel: Unknown | Instance MEM
Cookie Preferences
Mode: Command
Ln 1, Col 1
0_create_ingest_documents_test_kb.ipynb

Why customize?

Adapt to domain-specific language

E.g., Healthcare – Understand medical terminology and provide accurate responses related to patient's health

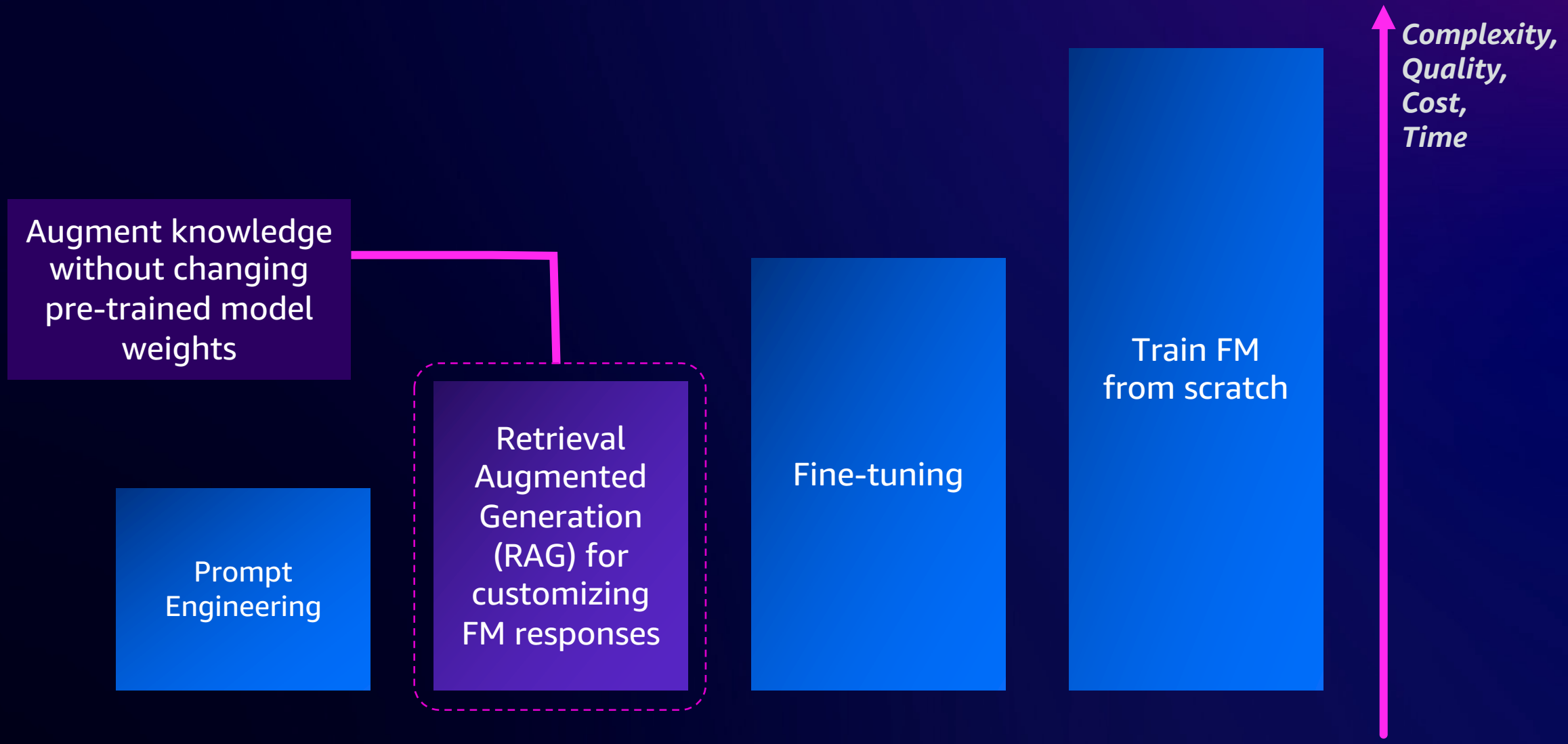
Enhance performance for specific tasks

E.g., Finance – Teach financial & accounting terms to provide good analysis for earnings reports

Improve context-awareness in responses

E.g., Customer Service – Improve ability to understand and respond to customer's inquires and complaints

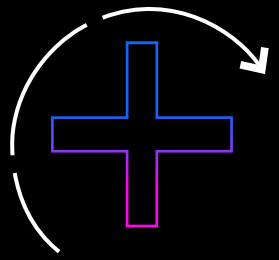
Common approaches for customizing foundation models (FMs)



What is Retrieval Augmented Generation?

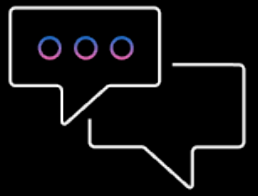
Retrieval

Fetches the relevant content from the external knowledge base or data sources based on a user query



Augmentation

Adding the retrieved relevant context to the user prompt, which goes as an input to the foundation model



Generation

Response from the foundation model based on the augmented prompt.

RAG use cases

Improved content quality

E.g., helps in reducing hallucinations and connecting with recent knowledge including enterprise data

Contextual chatbots and question answering

E.g., enhance chatbot capabilities by integrating with real-time data

Personalized search

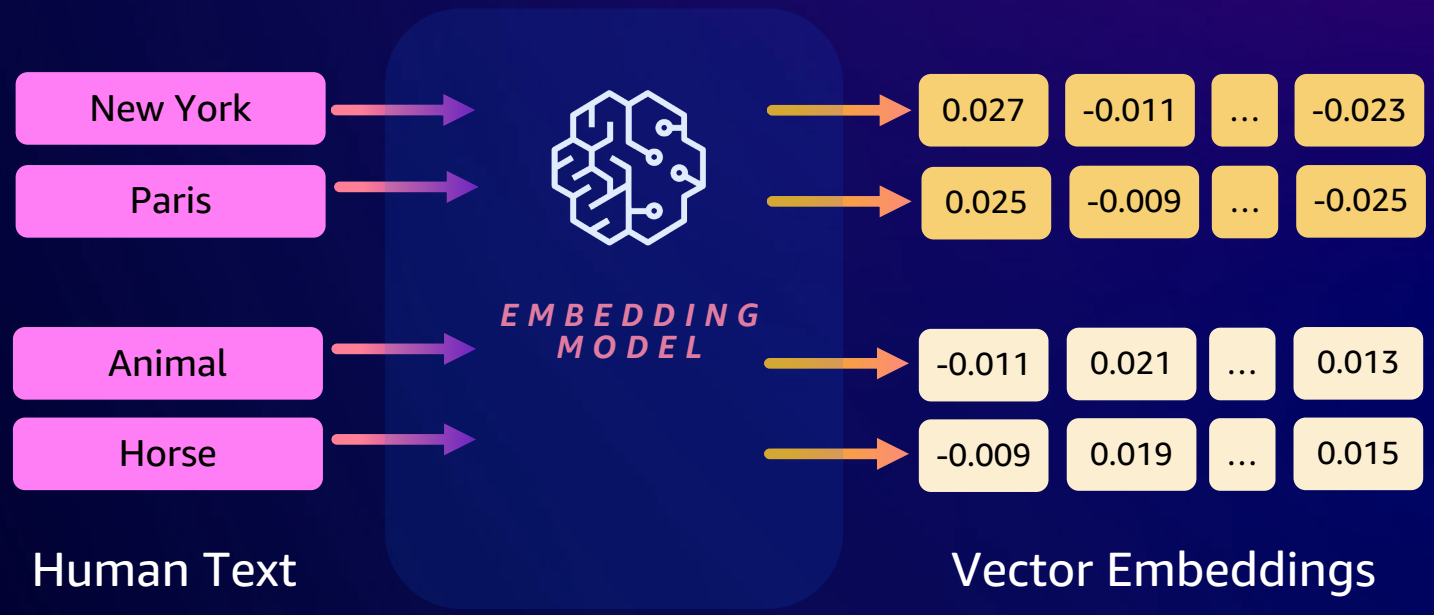
E.g., searching based on user previous search history and persona

Real-time data summarization

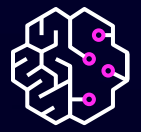
E.g., retrieving and summarizing transactional data from databases, or API calls

What are embeddings?

- Numerical representation of text (vectors) that captures semantics and relationships between words.
- Embedding models capture features and nuances of the text.
- Rich embeddings can be used to compare text similarity.
- Multilingual Text Embeddings can identify meaning in different languages.



Titan text embeddings model



Amazon Titan Text Embeddings

V*2.0*

Translates text inputs (words, phrases) into numerical representations (embeddings). Comparing embeddings produces more relevant and contextual responses than word matching.

Max Tokens: ***8,000***
Output Vectors***: 1,536***
Language: ***Multilingual*** (25 languages)

Model ID: ***amazon.titan-embed-g1-text-02***

Highlights

- Titan Text Embeddings offers fast, cost effective, high-performance, accurate embeddings in 25 languages.
- Optimized for text retrieval tasks, semantic similarity and clustering.
- Applications of this model includes semantic search and personalization.

Knowledge Bases for Amazon Bedrock

Gives FMs and agents contextual information from your private data sources for Retrieval Augmented Generation (RAG) to deliver more relevant, accurate, and customized responses.

Fully managed support for end-to-end RAG workflow

Securely connect FMs and agents to data sources

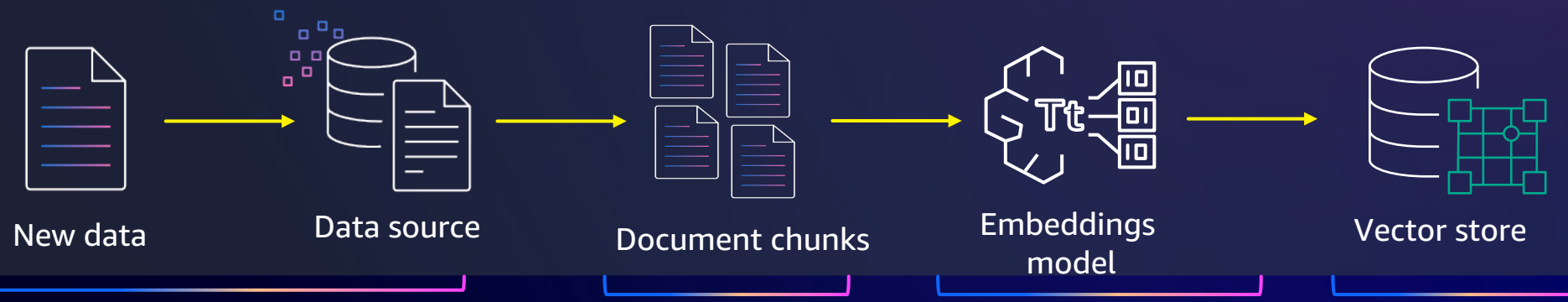
Easily retrieve relevant data and augment prompts

Provide source attribution

Data Ingestion Workflow

KNOWLEDGE BASES FOR AMAZON BEDROCK

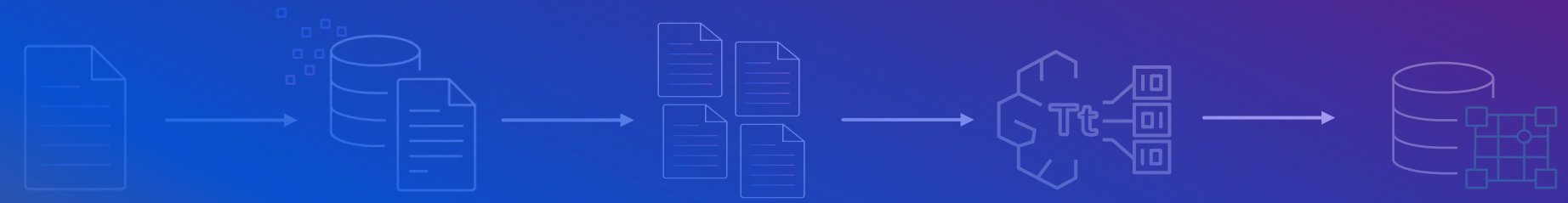
Fully managed data ingestion workflow



- Choose your data source (Amazon S3)
  - Support for incremental updates
  - Multiple data file formats supported
- Choose your chunking strategy
  - Fixed chunks
  - No chunking
  - Default (200 tokens)
- Choose your embedding model
  - Amazon Titan
- Choose your vector store
  - Open search serverless
  - Pinecone
  - Redis

Fully managed data ingestion

KNOWLEDGE BASES FOR AMAZON BEDROCK



Fully managed data ingestion workflow

- Choose your data source
- Support for incremental updates
- Multiple data file format supported

- Choose your chunking strategy
- Fixed chunks
- No chunking
- Default (200 tokens)

- Choose your embeddings model
- Amazon Titan

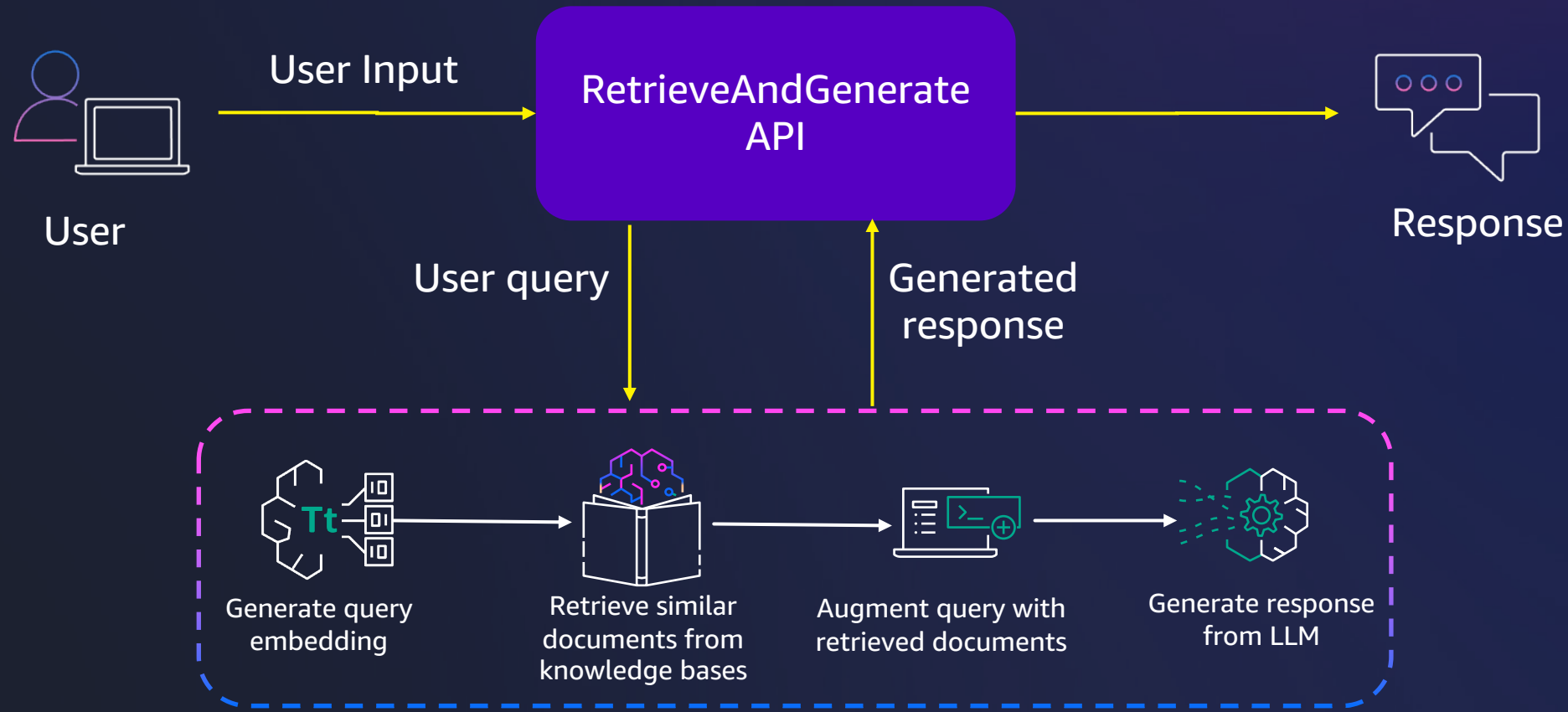
- Choose your vector store
- Open search serverless
- Pinecone
- Redis

Automated and fully managed data ingestion using Knowledge Bases for Amazon Bedrock

RetrieveAndGenerate API

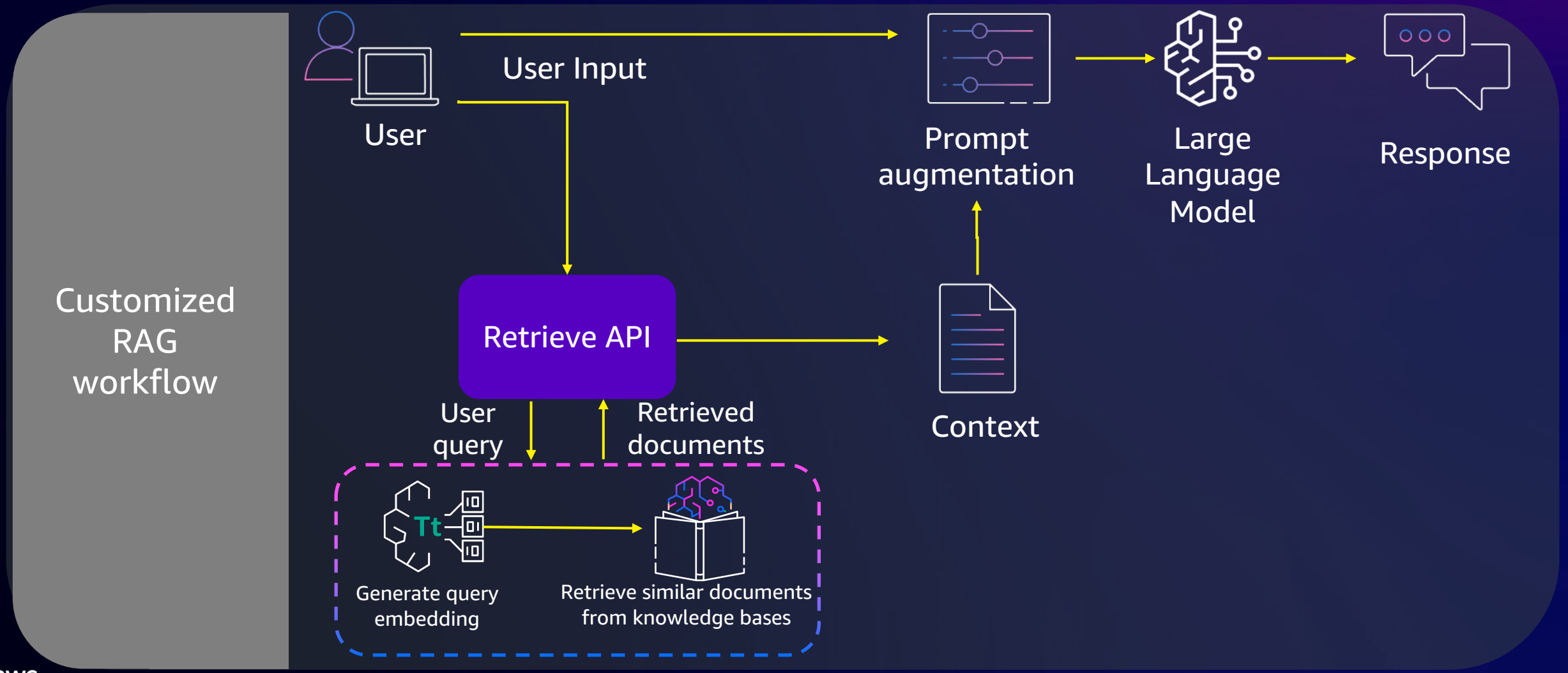
KNOWLEDGE BASES FOR AMAZON BEDROCK

Fully managed RAG



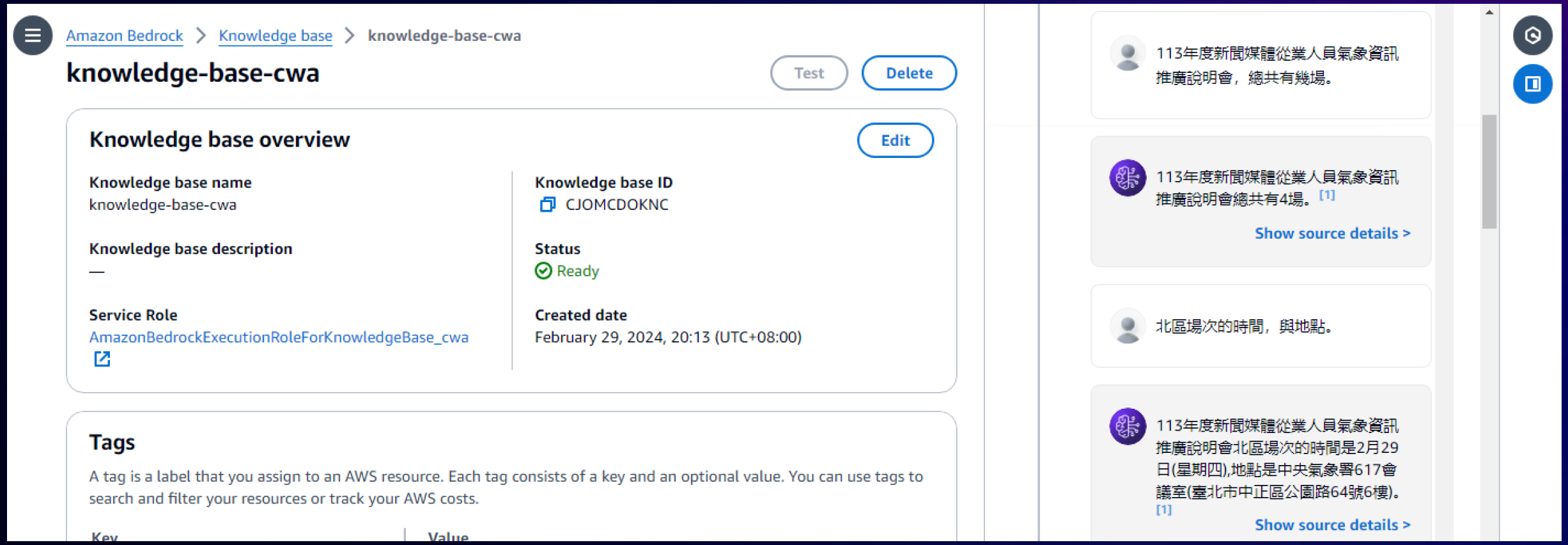
Customize RAG workflows using Retrieve API

KNOWLEDGE BASES FOR AMAZON BEDROCK

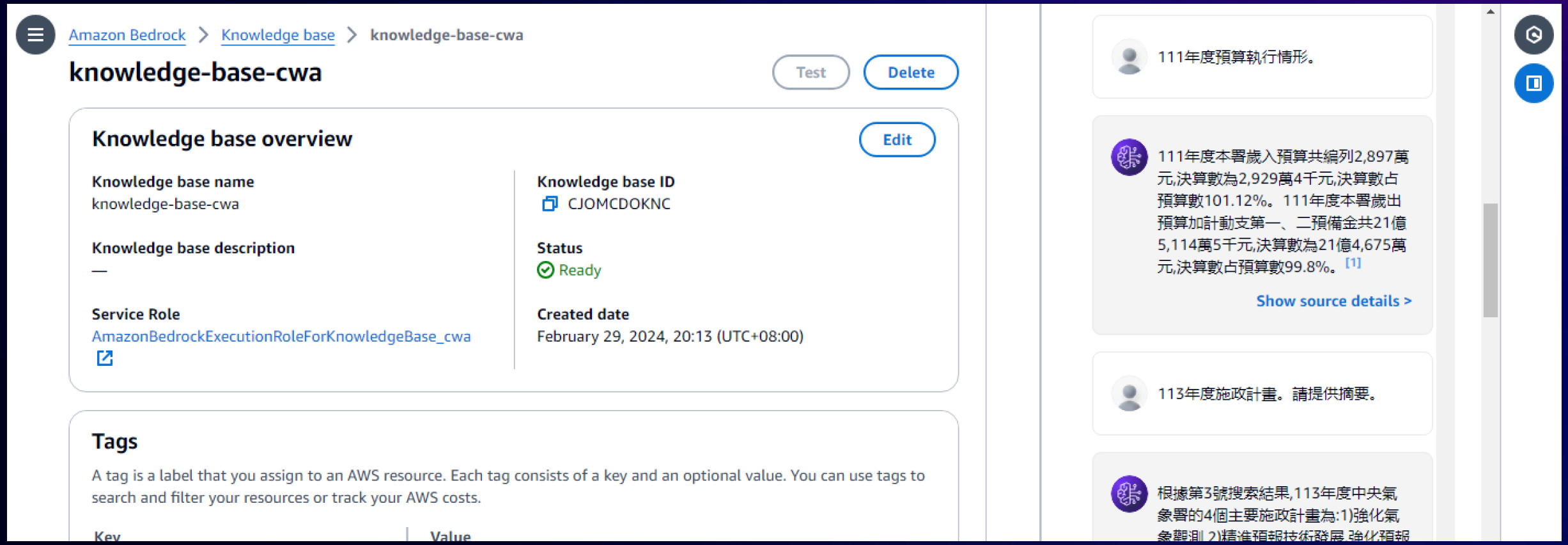


Demo – Fully managed RAG

Knowledge Base in Action



Knowledge Base in Action



交通部中央氣象署

「113年度新聞媒體從業人員氣象資訊推廣說明會」實施計畫

壹、目的

交通部中央氣象署為增進國內新聞媒體從業人員對本署氣象實務之認識，正確解讀氣象資訊及妥適應用，特以氣象防災及地震防災為主題，分別於北、東、中、南4區各舉辦1場本說明會，以深化氣象資訊之推廣與運用。

貳、主(協)辦機關

一、主辦機關：中央氣象署

二、共同主辦：臺東縣政府、國立自然科學博物館

參、舉辦方式

一、參加對象：國內各媒體業者(報紙、廣播、電視等)

二、說明會共4場，日期時間及地點如下表：

場次	日期	地點
北區	2月29日 (星期四)	中央氣象署617會議室 (臺北市中正區公園路64號6樓)
南區	3月7日 (星期四)	中央氣象署臺灣南區氣象中心1樓視聽室 (臺南市中西區公園路21號)
東區	3月20日 (星期三)	中央氣象署臺東氣象站 (臺東市大同路106號)
中區	4月3日 (星期三)	國立自然科學博物館第二科學教室 (臺中市北區館前路1號)

中華民國 112 年 10 月
立法院第 10 屆第 8 會期

交通部中央氣象署 113 年度

施政計畫及預算編列口頭報告

一、111 年度預算執行情形

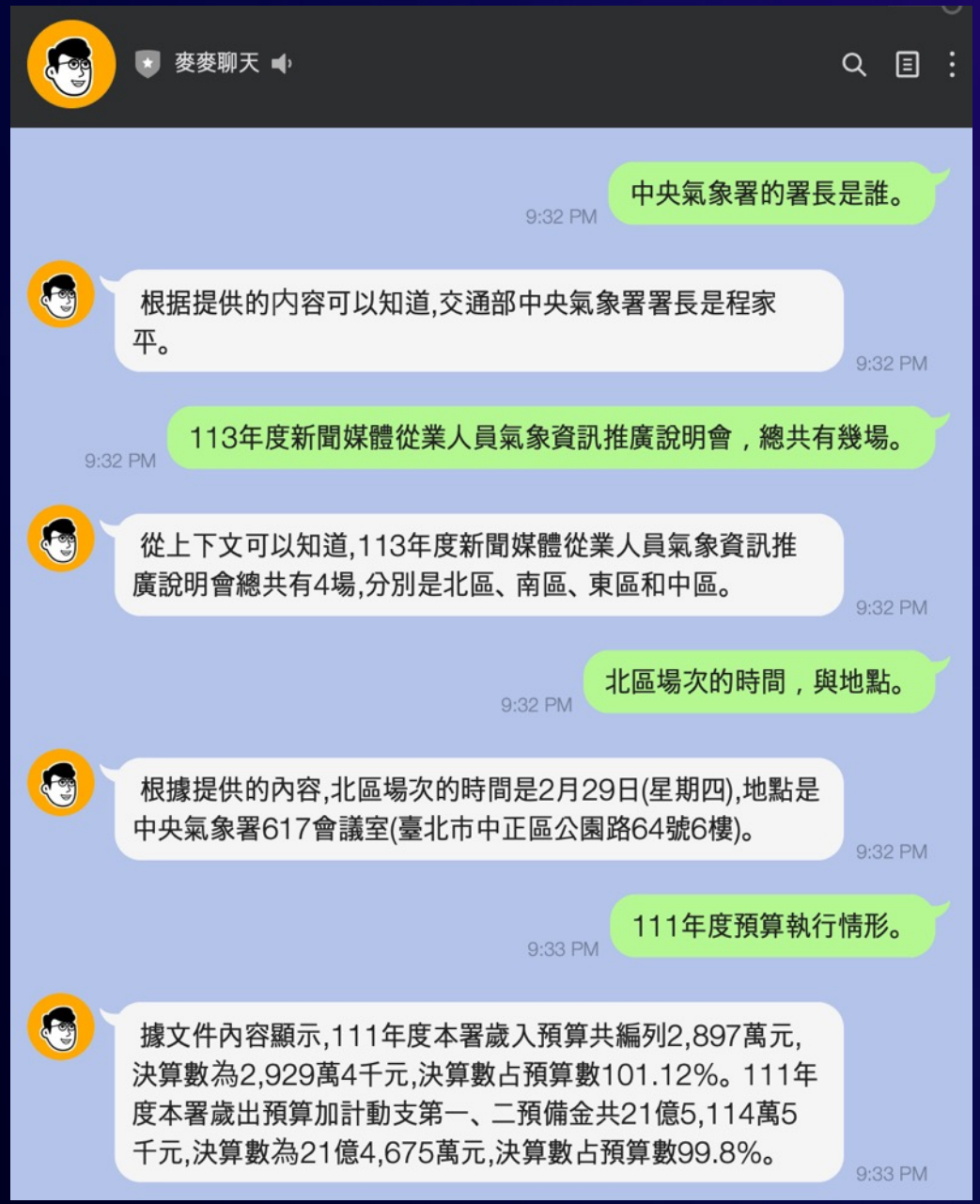
(一)111 年度本署歲入預算共編列 2,897 萬元，決算數為 2,929 萬 4 千元，決算數占預算數 101.12%。

(二)111 年度本署歲出預算加計動支第一、二預備金共 21 億 5,114 萬 5 千元，決算數為 21 億 4,675 萬元，決算數占預算數 99.8%。

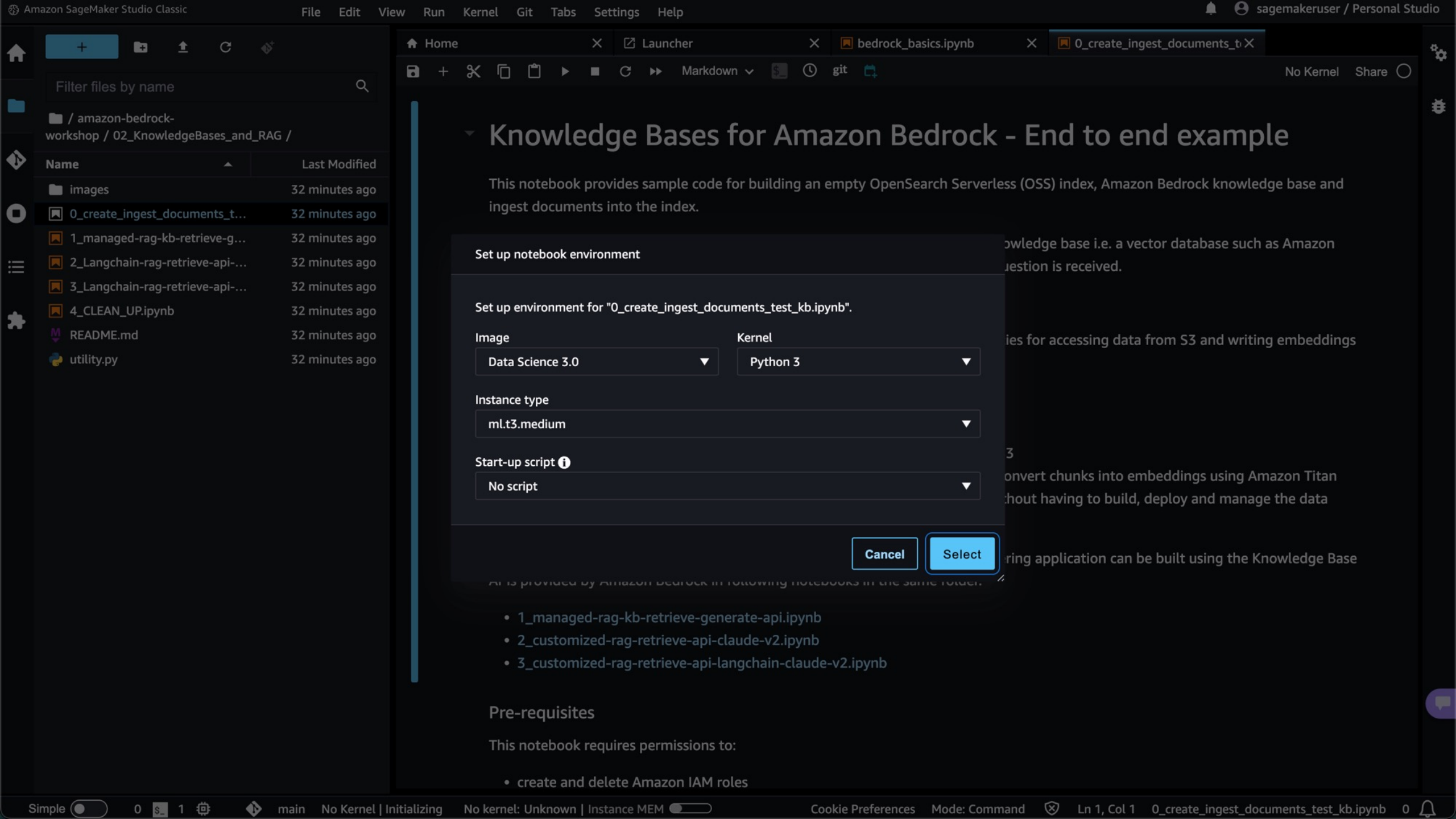
二、112 年度截至 9 月止預算執行情形

(一)112 年度本署歲入預算共編列 2,997 萬元，截至 9 月止預算分配數 1,782 萬 4 千元，實收數 2,548 萬 2 千元，占預算分配數 142.96%。

(二)112 年度本署歲出預算(含預備金動支數)共編列 19 億 692 萬 8 千元，截至 9 月止預算分配數 12 億 2,413 萬 6 千元，執行數 11 億 7,312 萬 7 千元，執行數占預算分配數 95.83%。



Hands-On Lab


Amazon SageMaker Studio Classic
File Edit View Run Kernel Git Tabs Settings Help
sagemakeruser / Personal Studio
Filter files by name
/ amazon-bedrock-workshop / 02_KnowledgeBases_and_RAG /
Name
Last Modified
images 32 minutes ago
0_create_ingest_documents_t... 32 minutes ago
1_managed-rag-kb-retrieve-g... 32 minutes ago
2_Langchain-rag-retrieve-api-... 32 minutes ago
3_Langchain-rag-retrieve-api-... 32 minutes ago
4_CLEAN_UP.ipynb 32 minutes ago
README.md 32 minutes ago
utility.py 32 minutes ago
Home
Launcher
bedrock_basics.ipynb
0_create_ingest_documents_t
Markdown
git
No Kernel
Share
Knowledge Bases for Amazon Bedrock - End to end example
This notebook provides sample code for building an empty OpenSearch Serverless (OSS) index, Amazon Bedrock knowledge base and ingest documents into the index.
Set up notebook environment
Set up environment for "0_create_ingest_documents_test_kb.ipynb".
Image
Data Science 3.0
Kernel
Python 3
Instance type
ml.t3.medium
Start-up script
No script
Cancel
Select
1_managed-rag-kb-retrieve-generate-api.ipynb
2_customized-rag-retrieve-api-claude-v2.ipynb
3_customized-rag-retrieve-api-langchain-claude-v2.ipynb
Pre-requisites
This notebook requires permissions to:
create and delete Amazon IAM roles
Simple
main
No Kernel | Initializing
No kernel: Unknown | Instance MEM
Cookie Preferences
Mode: Command
Ln 1, Col 1
0_create_ingest_documents_test_kb.ipynb

Knowledge Bases for Amazon Bedrock - End to end example

This notebook provides sample code for building an empty OpenSearch Serverless (OSS) index, Amazon Bedrock knowledge base and ingest documents into the index.

A data pipeline that ingests documents (typically stored in Amazon S3) into a knowledge base i.e. a vector database such as Amazon OpenSearch Service Serverless (AOSS) so that it is available for lookup when a question is received.

Steps:

- Create Amazon Bedrock Knowledge Base execution role with necessary policies for accessing data from S3 and writing embeddings into OSS.
- Create an empty OpenSearch serverless index.
- Download documents
- Create Amazon Bedrock knowledge base
- Create a data source within knowledge base which will connect to Amazon S3
- Start an ingestion job using KB APIs which will read data from s3, chunk it, convert chunks into embeddings using Amazon Titan Embeddings model and then store these embeddings in AOSS. All of this without having to build, deploy and manage the data pipeline.

Once the data is available in the Bedrock Knowledge Base then a question answering application can be built using the Knowledge Base APIs provided by Amazon Bedrock in following notebooks in the same folder.

- 1_managed-rag-kb-retrieve-generate-api.ipynb
- 2_customized-rag-retrieve-api-claude-v2.ipynb
- 3_customized-rag-retrieve-api-langchain-claude-v2.ipynb

Pre-requisites

This notebook requires permissions to:

- create and delete Amazon IAM roles
- create, update and delete Amazon S3 buckets
- access Amazon Bedrock
- access to Amazon OpenSearch Serverless

Setup

Before running the rest of this notebook, you'll need to run the cells below to (ensure necessary libraries are installed and) connect to Bedrock.

[2]:
```
%pip install -U opensearch-py==2.3.1
%pip install -U boto3==1.33.2
%pip install -U retrying==1.3.4
```

```
Collecting opensearch-py==2.3.1
  Downloading opensearch_py-2.3.1-py2.py3-none-any.whl.metadata (6.9 kB)
Collecting urllib3<2,>=1.21.1 (from opensearch-py==2.3.1)
  Downloading urllib3-1.26.18-py2.py3-none-any.whl.metadata (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 kB 511.6 kB/s eta 0:00:00a 0:00:01
Requirement already satisfied: requests<3.0.0,>=2.4.0 in /opt/conda/lib/python3.10/site-packages (from opensearch-py==2.3.1) (2.31.0)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages (from opensearch-py==2.3.1) (1.16.0)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.10/site-packages (from opensearch-py==2.3.1) (2.9.0.post0)
Requirement already satisfied: certifi>=2022.12.07 in /opt/conda/lib/python3.10/site-packages (from opensearch-py==2.3.1) (2023.11.17)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0,>=2.4.0->opensearch-py==2.3.1) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests<3.0.0,>=2.4.0->opensearch-py==2.3.1) (3.3)
Downloading opensearch_py-2.3.1-py2.py3-none-any.whl (327 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 327.3/327.3 kB 3.1 MB/s eta 0:00:00ta 0:00:01
Downloading urllib3-1.26.18-py2.py3-none-any.whl (143 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.8/143.8 kB 1.7 MB/s eta 0:00:00:00:01
Installing collected packages: urllib3, opensearch-py
  Attempting uninstall: urllib3
    Found existing installation: urllib3 2.2.1
    Uninstalling urllib3-2.2.1:
      Successfully uninstalled urllib3-2.2.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
distributed 2022.7.0 requires tornado<6.2,>=6.0.3, but you have tornado 6.4 which is incompatible.
Successfully installed opensearch-py-2.3.1 urllib3-1.26.18
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

[notice] A new release of pip is available: 23.3.1 -> 24.0
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
# restart kernel
from IPython.core.display import HTML
HTML("<script>Jupyter.notebook.kernel.restart()</script>")
```

[3]:

[4]:
```
import warnings
warnings.filterwarnings('ignore')
```

[5]:
```
import json
import os
import boto3
import pprint
from utility import create_bedrock_execution_role, create_oss_policy_attach_bedrock_execution_role, create_policies_in_oss
import random
from retrying import retry
suffix = random.randrange(200, 900)

sts_client = boto3.client('sts')
boto3_session = boto3.session.Session()
region_name = boto3_session.region_name
bedrock_agent_client = boto3_session.client('bedrock-agent', region_name=region_name)
service = 'aoss'
s3_client = boto3.client('s3')
account_id = sts_client.get_caller_identity()["Account"]
s3_suffix = f"{region_name}-{account_id}"
bucket_name = f'bedrock-kb-{s3_suffix}' # replace it with your bucket name.
pp = pprint.PrettyPrinter(indent=2)
```

[6]:
```
# Create S3 bucket for knowledge base data source
s3bucket = s3_client.create_bucket(
    Bucket=bucket_name,
    CreateBucketConfiguration={ 'LocationConstraint': region_name }
)
```

Create a vector store - OpenSearch Serverless index

Create a vector store - OpenSearch Serverless index

Step 1 - Create OSS policies and collection

Firt of all we have to create a vector store. In this section we will use *Amazon OpenSerach serverless.*

Amazon OpenSearch Serverless is a serverless option in Amazon OpenSearch Service. As a developer, you can use OpenSearch Serverless to run petabyte-scale workloads without configuring, managing, and scaling OpenSearch clusters. You get the same interactive millisecond response times as OpenSearch Service with the simplicity of a serverless environment. Pay only for what you use by automatically scaling resources to provide the right amount of capacity for your application—without impacting data ingestion.

[7]:
```
import boto3
import time
vector_store_name = f'bedrock-sample-rag-{suffix}'
index_name = f"bedrock-sample-rag-index-{suffix}"
aoss_client = boto3_session.client('opensearchserverless')
bedrock_kb_execution_role = create_bedrock_execution_role(bucket_name=bucket_name)
bedrock_kb_execution_role_arn = bedrock_kb_execution_role['Role']['Arn']
```

[8]:
```
# create security, network and data access policies within OSS
encryption_policy, network_policy, access_policy = create_policies_in_oss(vector_store_name=vector_store_name,
                      aoss_client=aoss_client,
                      bedrock_kb_execution_role_arn=bedrock_kb_execution_role_arn)
collection = aoss_client.create_collection(name=vector_store_name,type='VECTORSEARCH')
```

[9]:
```
pp.pprint(collection)
```

```
{ 'ResponseMetadata': { 'HTTPHeaders': { 'connection': 'keep-alive',
                                         'content-length': '314',
                                         'content-type': 'application/x-amz-json-1.0',
                                         'date': 'Tue, 26 Mar 2024 06:14:53 '
                                                 'GMT',
                                         'x-amzn-requestid': '6e67c99d-c29e-414b-9257-2b11ea0b1389'},
                        'HTTPStatusCode': 200,
                        'RequestId': '6e67c99d-c29e-414b-9257-2b11ea0b1389',
                        'RetryAttempts': 0},
  'createCollectionDetail': { 'arn': 'arn:aws:aoss:us-west-2:655380892071:collection/hnbf0ml4o1djqrjy1rcg',
                              'createdDate': 1711433692851,
```

[10]:
```
collection_id = collection['createCollectionDetail']['id']
host = collection_id + '.' + region_name + '.aoss.amazonaws.com'
print(host)
```

```
hnbf0ml4o1djqrjy1rcg.us-west-2.aoss.amazonaws.com
```

[11]:
```
# wait for collection creation
response = aoss_client.batch_get_collection(names=[vector_store_name])
# Periodically check collection status
while (response['collectionDetails'][0]['status']) == 'CREATING':
    print('Creating collection...')
    time.sleep(30)
    response = aoss_client.batch_get_collection(names=[vector_store_name])
print('\nCollection successfully created:')
print(response["collectionDetails"])
```

```
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...
Creating collection...

Collection successfully created:
[{'arn': 'arn:aws:aoss:us-west-2:655380892071:collection/hnbf0ml4o1djqrjy1rcg', 'collectionEndpoint': 'https://hnbf0ml4o1djqrjy1rcg.us-west-2.aoss.amazonaws.com', 'createdDate': 1711433692851, 'dashboardEndpoint': 'https://hnbf0ml4o1djqrjy1rcg.us-west-2.aoss.amazonaws.com/_dashboards', 'id': 'hnbf0ml4o1djqrjy1rcg', 'kmsKeyArn': 'auto', 'lastModifiedDate': 1711434081655, 'name': 'bedrock-sample-rag-408', 'standbyReplicas': 'ENABLED', 'status': 'ACTIVE', 'type': 'VECTORSEARCH'}]
```

[12]:
```
# create oss policy and attach it to Bedrock execution role
create_oss_policy_attach_bedrock_execution_role(collection_id=collection_id,
                                                bedrock_kb_execution_role=bedrock_kb_execution_role)
```

```
Opensearch serverless arn:  arn:aws:iam::655380892071:policy/AmazonBedrockOSSPolicyForKnowledgeBase_207
```

Step 2 - Create vector index

[13]:
```
from opensearchpy import OpenSearch, RequestsHttpConnection, AWSV4SignerAuth
credentials = boto3.Session().get_credentials()
awsauth = auth = AWSV4SignerAuth(credentials, region_name, service)

index_name = f"bedrock-sample-index-{suffix}"
body_json = {
   "settings": {
      "index.knn": "true",
       "number_of_shards": 1,
       "knn.algo_param.ef_search": 512,
       "number_of_replicas": 0,
   },
   "mappings": {
      "properties": {
         "vector": {
            "type": "knn_vector",
            "dimension": 1536,
             "method": {
                 "name": "hnsw",
                 "engine": "nmslib",
                 "space_type": "cosinesimil",
                 "parameters": {
                     "ef_construction": 512,
                     "m": 16
                 },
             },
         },
         "text": {
            "type": "text"
         },
         "text-metadata": {
            "type": "text"         }
      }
   }
}
# Build the OpenSearch client
oss_client = OpenSearch(
```

[14]:
```
# Create index
response = oss_client.indices.create(index=index_name, body=json.dumps(body_json))
print('\nCreating index:')
print(response)
time.sleep(60) # index creation can take up to a minute
```

```
Creating index:
{'acknowledged': True, 'shards_acknowledged': True, 'index': 'bedrock-sample-index-408'}
```

Download data

[15]:
```
# Download and prepare dataset
!mkdir -p ./data

from urllib.request import urlretrieve
urls = [
    'https://s2.q4cdn.com/299287126/files/doc_financials/2023/ar/2022-Shareholder-Letter.pdf',
    'https://s2.q4cdn.com/299287126/files/doc_financials/2022/ar/2021-Shareholder-Letter.pdf',
    'https://s2.q4cdn.com/299287126/files/doc_financials/2021/ar/Amazon-2020-Shareholder-Letter-and-1997-Shareholder-Letter.pdf',
    'https://s2.q4cdn.com/299287126/files/doc_financials/2020/ar/2019-Shareholder-Letter.pdf'
]

filenames = [
    'AMZN-2022-Shareholder-Letter.pdf',
    'AMZN-2021-Shareholder-Letter.pdf',
    'AMZN-2020-Shareholder-Letter.pdf',
    'AMZN-2019-Shareholder-Letter.pdf'
]

data_root = "./data/"

for idx, url in enumerate(urls):
    file_path = data_root + filenames[idx]
    urlretrieve(url, file_path)
```

Upload data to S3 Bucket

Upload data to S3 Bucket

[16]:
```
# Upload data to s3
s3_client = boto3.client("s3")
def uploadDirectory(path,bucket_name):
        for root,dirs,files in os.walk(path):
            for file in files:
                s3_client.upload_file(os.path.join(root,file),bucket_name,file)

uploadDirectory(data_root, bucket_name)
```

Create Knowledge Base ¶

Steps:

- initialize Open search serverless configuration which will include collection ARN, index name, vector field, text field and metadata field.
- initialize chunking strategy, based on which KB will split the documents into pieces of size equal to the chunk size mentioned in the `chunkingStrategyConfiguration` .
- initialize the s3 configuration, which will be used to create the data source object later.
- initialize the Titan embeddings model ARN, as this will be used to create the embeddings for each of the text chunks.

[17]:
```
opensearchServerlessConfiguration = {
            "collectionArn": collection["createCollectionDetail"]['arn'],
            "vectorIndexName": index_name,
            "fieldMapping": {
                "vectorField": "vector",
                "textField": "text",
                "metadataField": "text-metadata"
            }
        }

chunkingStrategyConfiguration = {
    "chunkingStrategy": "FIXED_SIZE",
    "fixedSizeChunkingConfiguration": {
        "maxTokens": 512,
        "overlapPercentage": 20
    }
```

Create Knowledge Base

Steps:

- initialize Open search serverless configuration which will include collection ARN, index name, vector field, text field and metadata field.
- initialize chunking strategy, based on which KB will split the documents into pieces of size equal to the chunk size mentioned in the `chunkingStrategyConfiguration` .
- initialize the s3 configuration, which will be used to create the data source object later.
- initialize the Titan embeddings model ARN, as this will be used to create the embeddings for each of the text chunks.

[17]:
```
opensearchServerlessConfiguration = {
            "collectionArn": collection["createCollectionDetail"]['arn'],
            "vectorIndexName": index_name,
            "fieldMapping": {
                "vectorField": "vector",
                "textField": "text",
                "metadataField": "text-metadata"
            }
        }

chunkingStrategyConfiguration = {
    "chunkingStrategy": "FIXED_SIZE",
    "fixedSizeChunkingConfiguration": {
        "maxTokens": 512,
        "overlapPercentage": 20
    }
}

s3Configuration = {
    "bucketArn": f"arn:aws:s3:::{bucket_name}",
    # "inclusionPrefixes":["*.*"] # you can use this if you want to create a KB using data within s3 prefixes.
}

embeddingModelArn = f"arn:aws:bedrock:{region_name}::foundation-model/amazon.titan-embed-text-v1"

name = f"bedrock-sample-knowledge-base-{suffix}"
description = "Amazon shareholder letter knowledge base."
roleArn = bedrock_kb_execution_role_arn
```

[18]:
```
# Create a KnowledgeBase
from retrying import retry

@retry(wait_random_min=1000, wait_random_max=2000,stop_max_attempt_number=7)
def create_knowledge_base_func():
    create_kb_response = bedrock_agent_client.create_knowledge_base(
        name = name,
        description = description,
        roleArn = roleArn,
        knowledgeBaseConfiguration = {
            "type": "VECTOR",
            "vectorKnowledgeBaseConfiguration": {
                "embeddingModelArn": embeddingModelArn
            }
        },
        storageConfiguration = {
            "type": "OPENSEARCH_SERVERLESS",
            "opensearchServerlessConfiguration":opensearchServerlessConfiguration
        }
    )
    return create_kb_response["knowledgeBase"]
```

[19]:
```
try:
    kb = create_knowledge_base_func()
except Exception as err:
    print(f"{err=}, {type(err)=}")
```

[20]:
```
pp.pprint(kb)
```

```
{ 'createdAt': datetime.datetime(2024, 3, 26, 6, 23, 28, 437073, tzinfo=tzlocal()),
  'description': 'Amazon shareholder letter knowledge base.',
  'knowledgeBaseArn': 'arn:aws:bedrock:us-west-2:655380892071:knowledge-base/CRXSFPF4H8',
  'knowledgeBaseConfiguration': { 'type': 'VECTOR',
                                  'vectorKnowledgeBaseConfiguration': { 'embeddingModelArn': 'arn:aws:bedrock:us-west-2::foundation-model/amazon.titan-embed-tex
t-v1'}},
  'knowledgeBaseId': 'CRXSFPF4H8',
  'name': 'bedrock-sample-knowledge-base-408',
  'roleArn': 'arn:aws:iam::655380892071:role/AmazonBedrockExecutionRoleForKnowledgeBase_207',
  'status': 'CREATING',
```

[21]:
```
# Get KnowledgeBase
get_kb_response = bedrock_agent_client.get_knowledge_base(knowledgeBaseId = kb['knowledgeBaseId'])
```

Next we need to create a data source, which will be associated with the knowledge base created above. Once the data source is ready, we can then start to ingest the documents.

[22]:
```
# Create a DataSource in KnowledgeBase
create_ds_response = bedrock_agent_client.create_data_source(
    name = name,
    description = description,
    knowledgeBaseId = kb['knowledgeBaseId'],
    dataSourceConfiguration = {
        "type": "S3",
        "s3Configuration":s3Configuration
    },
    vectorIngestionConfiguration = {
        "chunkingConfiguration": chunkingStrategyConfiguration
    }
)
ds = create_ds_response["dataSource"]
pp.pprint(ds)
```

```
{ 'createdAt': datetime.datetime(2024, 3, 26, 6, 23, 29, 351908, tzinfo=tzlocal()),
  'dataSourceConfiguration': { 's3Configuration': { 'bucketArn': 'arn:aws:s3:::bedrock-kb-us-west-2-655380892071'},
                               'type': 'S3'},
  'dataSourceId': 'CQKVIMQBBD',
  'description': 'Amazon shareholder letter knowledge base.',
  'knowledgeBaseId': 'CRXSFPF4H8',
  'name': 'bedrock-sample-knowledge-base-408',
  'status': 'AVAILABLE',
  'updatedAt': datetime.datetime(2024, 3, 26, 6, 23, 29, 351908, tzinfo=tzlocal()),
  'vectorIngestionConfiguration': { 'chunkingConfiguration': { 'chunkingStrategy': 'FIXED_SIZE',
                                                               'fixedSizeChunkingConfiguration': { 'maxTokens': 512,
                                                                                                   'overlapPercentage': 20}}}}
```

[23]:
```
# Get DataSource
bedrock_agent_client.get_data_source(knowledgeBaseId = kb['knowledgeBaseId'], dataSourceId = ds["dataSourceId"])
```

[23]:
```
{'ResponseMetadata': {'RequestId': '146c0a9e-b512-4e0c-853e-24a2c22ab687',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Tue, 26 Mar 2024 06:23:29 GMT',
```

[23]:
```
# Get DataSource
bedrock_agent_client.get_data_source(knowledgeBaseId = kb['knowledgeBaseId'], dataSourceId = ds["dataSourceId"])
```

[23]:
```
{'ResponseMetadata': {'RequestId': '146c0a9e-b512-4e0c-853e-24a2c22ab687',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Tue, 26 Mar 2024 06:23:29 GMT',
   'content-type': 'application/json',
   'content-length': '573',
   'connection': 'keep-alive',
   'x-amzn-requestid': '146c0a9e-b512-4e0c-853e-24a2c22ab687',
   'x-amz-apigw-id': 'VOV7RFSDPHcEgXg=',
   'x-amzn-trace-id': 'Root=1-660269e1-14b9e1aa5ac67dd0751f7794'},
  'RetryAttempts': 0},
 'dataSource': {'knowledgeBaseId': 'CRXSFPF4H8',
  'dataSourceId': 'CQKVIMQBBD',
  'name': 'bedrock-sample-knowledge-base-408',
  'status': 'AVAILABLE',
  'description': 'Amazon shareholder letter knowledge base.',
  'dataSourceConfiguration': {'type': 'S3',
   's3Configuration': {'bucketArn': 'arn:aws:s3:::bedrock-kb-us-west-2-655380892071'}},
  'vectorIngestionConfiguration': {'chunkingConfiguration': {'chunkingStrategy': 'FIXED_SIZE',
    'fixedSizeChunkingConfiguration': {'maxTokens': 512,
     'overlapPercentage': 20}}},
  'createdAt': datetime.datetime(2024, 3, 26, 6, 23, 29, 351908, tzinfo=tzlocal()),
  'updatedAt': datetime.datetime(2024, 3, 26, 6, 23, 29, 351908, tzinfo=tzlocal())}}
```

Start ingestion job

Once the KB and data source is created, we can start the ingestion job. During the ingestion job, KB will fetch the documents in the data source, pre-process it to extract text, chunk it based on the chunking size provided, create embeddings of each chunk and then write it to the vector database, in this case OSS.

[24]:
```
# Start an ingestion job
start_job_response = bedrock_agent_client.start_ingestion_job(knowledgeBaseId = kb['knowledgeBaseId'], dataSourceId = ds["dataSourceId"])
```

[25]:
```
job = start_job_response["ingestionJob"]
pp.pprint(job)
```

```
{ 'dataSourceId': 'CQKVIMQBBD',
```

Start ingestion job

Once the KB and data source is created, we can start the ingestion job. During the ingestion job, KB will fetch the documents in the data source, pre-process it to extract text, chunk it based on the chunking size provided, create embeddings of each chunk and then write it to the vector database, in this case OSS.

[24]:
```
# Start an ingestion job
start_job_response = bedrock_agent_client.start_ingestion_job(knowledgeBaseId = kb['knowledgeBaseId'], dataSourceId = ds["dataSourceId"])
```

[25]:
```
job = start_job_response["ingestionJob"]
pp.pprint(job)
```

```
{ 'dataSourceId': 'CQKVIMQBBD',
  'ingestionJobId': 'KLQYCO0QKN',
  'knowledgeBaseId': 'CRXSFPF4H8',
  'startedAt': datetime.datetime(2024, 3, 26, 6, 23, 31, 575806, tzinfo=tzlocal()),
  'statistics': { 'numberOfDocumentsDeleted': 0,
                  'numberOfDocumentsFailed': 0,
                  'numberOfDocumentsScanned': 0,
                  'numberOfModifiedDocumentsIndexed': 0,
                  'numberOfNewDocumentsIndexed': 0},
  'status': 'STARTING',
  'updatedAt': datetime.datetime(2024, 3, 26, 6, 23, 31, 575806, tzinfo=tzlocal())}
```

[26]:
```
# Get job
while(job['status']!='COMPLETE' ):
  get_job_response = bedrock_agent_client.get_ingestion_job(
      knowledgeBaseId = kb['knowledgeBaseId'],
        dataSourceId = ds["dataSourceId"],
        ingestionJobId = job["ingestionJobId"]
  )
  job = get_job_response["ingestionJob"]
pp.pprint(job)
time.sleep(40)
```

```
{ 'dataSourceId': 'CQKVIMQBBD',
  'ingestionJobId': 'KLQYCO0QKN',
  'knowledgeBaseId': 'CRXSFPF4H8',
  'startedAt': datetime.datetime(2024, 3, 26, 6, 23, 31, 575806, tzinfo=tzlocal()),
  'statistics': { 'numberOfDocumentsDeleted': 0,
                  'numberOfDocumentsFailed': 0,
```

[27]:
```
kb_id = kb["knowledgeBaseId"]
pp.pprint(kb_id)
```

```
'CRXSFPF4H8'
```

[28]:
```
%store kb_id
```

```
Stored 'kb_id' (str)
```

Test the knowledge base

Using RetrieveAndGenerate API

Behind the scenes, RetrieveAndGenerate API converts queries into embeddings, searches the knowledge base, and then augments the foundation model prompt with the search results as context information and returns the FM-generated response to the question. For multi-turn conversations, Knowledge Bases manage short-term memory of the conversation to provide more contextual results.

The output of the RetrieveAndGenerate API includes the generated response, source attribution as well as the retrieved text chunks.

[34]:
```
# try out KB using RetrieveAndGenerate API
bedrock_agent_runtime_client = boto3.client("bedrock-agent-runtime", region_name=region_name)
model_id = "anthropic.claude-instant-v1" # try with both claude instant as well as claude-v2. for claude v2 - "anthropic.claude-v2"
model_arn = f'arn:aws:bedrock:{region_name}::foundation-model/{model_id}'
```

[35]:
```
time.sleep(5)
query = "What is Amazon's doing in the field of generative AI?"
response = bedrock_agent_runtime_client.retrieve_and_generate(
    input={
        'text': query
    },
    retrieveAndGenerateConfiguration={
        'type': 'KNOWLEDGE_BASE',
        'knowledgeBaseConfiguration': {
            'knowledgeBaseId': kb_id,
            'modelArn': model_arn
        }
    },
```

Test the knowledge base

Using RetrieveAndGenerate API

Behind the scenes, RetrieveAndGenerate API converts queries into embeddings, searches the knowledge base, and then augments the foundation model prompt with the search results as context information and returns the FM-generated response to the question. For multi-turn conversations, Knowledge Bases manage short-term memory of the conversation to provide more contextual results.

The output of the RetrieveAndGenerate API includes the generated response, source attribution as well as the retrieved text chunks.

[34]:
```
# try out KB using RetrieveAndGenerate API
bedrock_agent_runtime_client = boto3.client("bedrock-agent-runtime", region_name=region_name)
model_id = "anthropic.claude-instant-v1" # try with both claude instant as well as claude-v2. for claude v2 - "anthropic.claude-v2"
model_arn = f'arn:aws:bedrock:{region_name}::foundation-model/{model_id}'
```

[35]:
```
time.sleep(5)
query = "What is Amazon's doing in the field of generative AI?"
response = bedrock_agent_runtime_client.retrieve_and_generate(
    input={
        'text': query
    },
    retrieveAndGenerateConfiguration={
        'type': 'KNOWLEDGE_BASE',
        'knowledgeBaseConfiguration': {
            'knowledgeBaseId': kb_id,
            'modelArn': model_arn
        }
    },
)

generated_text = response['output']['text']
pp.pprint(generated_text)
```

```
('Amazon has been working on their own large language models (LLMs) for '
 'generative AI and believes it will transform and improve virtually every '
 'customer experience. They are continuing to invest substantially in these '
 'models across all of their consumer, seller, brand, and creator experiences. '
 "Additionally, as they've done for years in AWS, they're democratizing this "
```

[36]:
```
## print out the source attribution/citations from the original documents to see if the response generated belongs to the context.
citations = response["citations"]
contexts = []
for citation in citations:
    retrievedReferences = citation["retrievedReferences"]
    for reference in retrievedReferences:
        contexts.append(reference["content"]["text"])

pp.pprint(contexts)
```

```
[ 'This shift was driven by several factors, including access to higher '
  'volumes of compute capacity at lower prices than was ever available. Amazon '
  'has been using machine learning extensively for 25 years, employing it in '
  'everything from personalized ecommerce recommendations, to fulfillment '
  'center pick paths, to drones for Prime Air, to Alexa, to the many machine '
  'learning services AWS offers (where AWS has the broadest machine learning '
  'functionality and customer base of any cloud provider). More recently, a '
  'newer form of machine learning, called Generative AI, has burst onto the '
  'scene and promises to significantly accelerate machine learning adoption. '
  'Generative AI is based on very Large Language Models (trained on up to '
  'hundreds of billions of parameters, and growing), across expansive '
  'datasets, and has radically general and broad recall and learning '
  'capabilities. We have been working on our own LLMs for a while now, believe '
  'it will transform and improve virtually every customer experience, and will '
  'continue to invest substantially in these models across all of our '
  'consumer, seller, brand, and creator experiences. Additionally, as we've '
  'done for years in AWS, we're democratizing this technology so companies of '
  'all sizes can leverage Generative AI. AWS is offering the most '
  'price-performant machine learning chips in Trainium and Inferentia so small '
  'and large companies can afford to train and run their LLMs in production. '
  'We enable companies to choose from various LLMs and build applications with '
  'all of the AWS security, privacy and other features that customers are '
  'accustomed to using. And, we're delivering applications like AWS's '
  'CodeWhisperer, which revolutionizes       developer productivity by '
  'generating code suggestions in real time. I could write an entire letter on '
  'LLMs and Generative AI as I think they will be that transformative, but '
  'I'll leave that for a future letter. Let's just say that LLMs and '
  'Generative AI are going to be a big deal for customers, our shareholders, '
  'and Amazon.   So, in closing, I'm optimistic that we'll emerge from this '
  'challenging macroeconomic time in a stronger position than when we entered '
  'it. There are several reasons for it and I've mentioned many of them above. '
```

Retrieve API

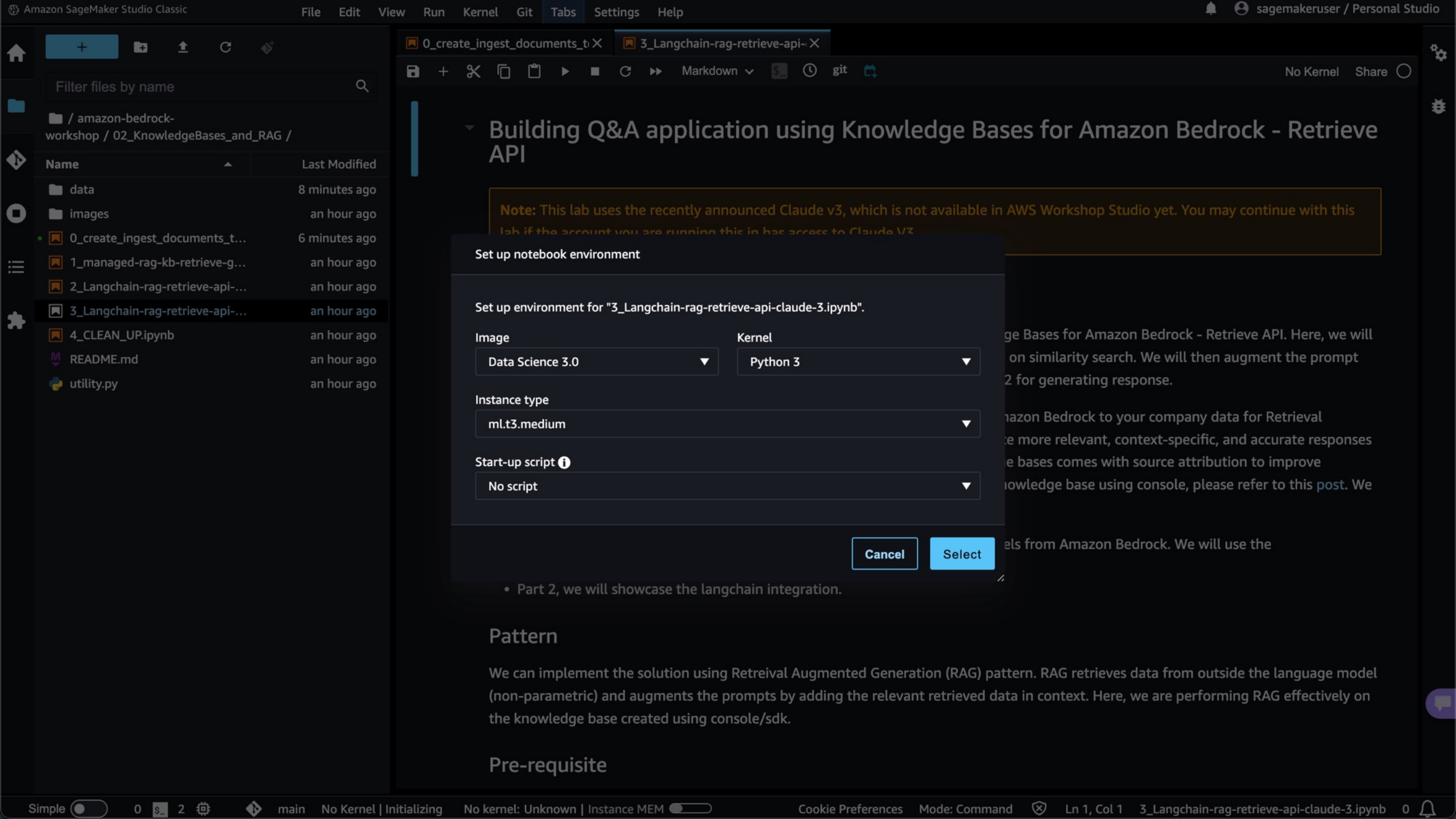
Retrieve API converts user queries into embeddings, searches the knowledge base, and returns the relevant results, giving you more control to build custom workflows on top of the semantic search results. The output of the Retrieve API includes the the retrieved text chunks, the location type and URI of the source data, as well as the relevance scores of the retrievals.

[37]:
```
# retreive api for fetching only the relevant context.
relevant_documents = bedrock_agent_runtime_client.retrieve(
    retrievalQuery= {
        'text': query
    },
    knowledgeBaseId=kb_id,
    retrievalConfiguration= {
        'vectorSearchConfiguration': {
            'numberOfResults': 3 # will fetch top 3 documents which matches closely with the query.
        }
    }
)
```

[38]:
```
pp.pprint(relevant_documents["retrievalResults"])
```

```
[ { 'content': { 'text': 'This shift was driven by several factors, including '
                         'access to higher volumes of compute capacity at '
                         'lower prices than was ever available. Amazon has '
                         'been using machine learning extensively for 25 '
                         'years, employing it in everything from personalized '
                         'ecommerce recommendations, to fulfillment center '
                         'pick paths, to drones for Prime Air, to Alexa, to '
                         'the many machine learning services AWS offers (where '
                         'AWS has the broadest machine learning functionality '
                         'and customer base of any cloud provider). More '
                         'recently, a newer form of machine learning, called '
                         'Generative AI, has burst onto the scene and promises '
                         'to significantly accelerate machine learning '
                         'adoption. Generative AI is based on very Large '
                         'Language Models (trained on up to hundreds of '
                         'billions of parameters, and growing), across '
                         'expansive datasets, and has radically general and '
                         'broad recall and learning capabilities. We have been '
                         'working on our own LLMs for a while now, believe it '
```

Hands-On Lab


Amazon SageMaker Studio Classic
File Edit View Run Kernel Git Tabs Settings Help
sagemakeruser / Personal Studio
0_create_ingest_documents_t
3_Langchain-rag-retrieve-api-
Markdown
git
No Kernel
Share
Filter files by name
/ amazon-bedrock-workshop / 02_KnowledgeBases_and_RAG /
Name
Last Modified
data 8 minutes ago
images an hour ago
0_create_ingest_documents_t... 6 minutes ago
1_managed-rag-kb-retrieve-g... an hour ago
2_Langchain-rag-retrieve-api-... an hour ago
3_Langchain-rag-retrieve-api-... an hour ago
4_CLEAN_UP.ipynb an hour ago
README.md an hour ago
utility.py an hour ago
Building Q&A application using Knowledge Bases for Amazon Bedrock - Retrieve API
Note: This lab uses the recently announced Claude v3, which is not available in AWS Workshop Studio yet. You may continue with this lab if the account you are running this in has access to Claude V3
Set up notebook environment
Set up environment for "3_Langchain-rag-retrieve-api-claude-3.ipynb".
Image
Data Science 3.0
Kernel
Python 3
Instance type
ml.t3.medium
Start-up script
No script
Cancel
Select
ge Bases for Amazon Bedrock - Retrieve API. Here, we will
on similarity search. We will then augment the prompt
2 for generating response.
nazon Bedrock to your company data for Retrieval
e more relevant, context-specific, and accurate responses
e bases comes with source attribution to improve
owledge base using console, please refer to this post. We
els from Amazon Bedrock. We will use the
Part 2, we will showcase the langchain integration.
Pattern
We can implement the solution using Retreival Augmented Generation (RAG) pattern. RAG retrieves data from outside the language model (non-parametric) and augments the prompts by adding the relevant retrieved data in context. Here, we are performing RAG effectively on the knowledge base created using console/sdk.
Pre-requisite
Simple
main
No Kernel | Initializing
No kernel: Unknown | Instance MEM
Cookie Preferences
Mode: Command
Ln 1, Col 1
3_Langchain-rag-retrieve-api-claude-3.ipynb

Building Q&A application using Knowledge Bases for Amazon Bedrock - Retrieve API

Note: This lab uses the recently announced Claude v3, which is not available in AWS Workshop Studio yet. You may continue with this lab if the account you are running this in has access to Claude V3.

Context

In this notebook, we will dive deep into building Q&A application using Knowledge Bases for Amazon Bedrock - Retrieve API. Here, we will query the knowledge base to get the desired number of document chunks based on similarity search. We will then augment the prompt with relevant documents and query which will go as input to Anthropic Claude V2 for generating response.

With a knowledge base, you can securely connect foundation models (FMs) in Amazon Bedrock to your company data for Retrieval Augmented Generation (RAG). Access to additional data helps the model generate more relevant, context-specific, and accurate responses without continuously retraining the FM. All information retrieved from knowledge bases comes with source attribution to improve transparency and minimize hallucinations. For more information on creating a knowledge base using console, please refer to this post. We will cover 2 parts in the notebook:

- Part 1, we will share how you can use `RetrieveAPI` with foundation models from Amazon Bedrock. We will use the `anthropic.claude-3-sonnet-20240229-v1:0` model.
- Part 2, we will showcase the langchain integration.

Pattern

We can implement the solution using Retreival Augmented Generation (RAG) pattern. RAG retrieves data from outside the language model (non-parametric) and augments the prompts by adding the relevant retrieved data in context. Here, we are performing RAG effectively on the knowledge base created using console/sdk.

Pre-requisite

Before being able to answer the questions, the documents must be processed and ingested in vector database.

1. Load the documents into the knowledge base by connecting your s3 bucket (data source).
2. Ingestion - Knowledge bases will split them into smaller chunks (based on the strategy selected), generate embeddings and store it in the associated vectore store and notebook 0_create_ingest_documents_test_kb.ipynb takes care of it for you.

Setup

To run this notebook you would need to install following packages.

[2]:
```
%pip install --upgrade pip
%pip install boto3 --force-reinstall --quiet
%pip install botocore --force-reinstall --quiet
%pip install langchain --force-reinstall --quiet
```

```
Requirement already satisfied: pip in /opt/conda/lib/python3.10/site-packages (23.3.1)
Collecting pip
  Downloading pip-24.0-py3-none-any.whl.metadata (3.6 kB)
Downloading pip-24.0-py3-none-any.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 11.1 MB/s eta 0:00:0000:0100:01
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.3.1
    Uninstalling pip-23.3.1:
      Successfully uninstalled pip-23.3.1
Successfully installed pip-24.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a v
irtual environment instead: https://pip.pypa.io/warnings/venv
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following depende
ncy conflicts.
spyder 5.3.3 requires pyqt5<5.16, which is not installed.
spyder 5.3.3 requires pyqtwebengine<5.16, which is not installed.
distributed 2022.7.0 requires tornado<6.2,>=6.0.3, but you have tornado 6.4 which is incompatible.
jupyterlab 3.4.4 requires jupyter-server~=1.16, but you have jupyter-server 2.12.1 which is incompatible.
jupyterlab-server 2.10.3 requires jupyter-server~=1.4, but you have jupyter-server 2.12.1 which is incompatible.
notebook 6.5.6 requires jupyter-client<8,>=5.3.4, but you have jupyter-client 8.6.0 which is incompatible.
notebook 6.5.6 requires pyzmq<25,>=17, but you have pyzmq 25.1.2 which is incompatible.
opensearch-py 2.3.1 requires urllib3<2,>=1.21.1, but you have urllib3 2.2.1 which is incompatible.
panel 0.13.1 requires bokeh<2.5.0,>=2.4.0, but you have bokeh 3.3.2 which is incompatible.
sagemaker 2.199.0 requires urllib3<1.27, but you have urllib3 2.2.1 which is incompatible.
sagemaker-datawrangler 0.4.3 requires sagemaker-data-insights==0.4.0, but you have sagemaker-data-insights 0.3.3 which is incompatible.
spyder 5.3.3 requires ipython<8.0.0,>=7.31.1, but you have ipython 8.18.1 which is incompatible.
spyder 5.3.3 requires pylint<3.0,>=2.5.0, but you have pylint 3.0.2 which is incompatible.
spyder-kernels 2.3.3 requires ipython<8,>=7.31.1; python_version >= "3", but you have ipython 8.18.1 which is incompatible.
spyder-kernels 2.3.3 requires jupyter-client<8,>=7.3.4; python_version >= "3", but you have jupyter-client 8.6.0 which is incompatible.
```

[5]:
```
store -r kb_id
```

[6]:
```
# store -r kb_id
# kb_id = "<knowledge base id>" If you have already created knowledge base, comment the `store -r kb_id` and provide knowledge base id here.
```

Follow the steps below to initiate the bedrock client:

1. Import the necessary libraries, along with langchain for bedrock model selection, llama index to store the service context containing the llm and embedding model instances. We will use this service context later in the notebook for evaluating the responses from our Q&A application.

2. Initialize `anthropic.claude-3-sonnet-20240229-v1:0` as our large language model to perform query completions using the RAG pattern with the given knowledge base, once we get all text chunk searches through the `retrieve` API.

[7]:
```
import boto3
import pprint
from botocore.client import Config
import json

pp = pprint.PrettyPrinter(indent=2)
session = boto3.session.Session()
region = session.region_name
bedrock_config = Config(connect_timeout=120, read_timeout=120, retries={'max_attempts': 0})
bedrock_client = boto3.client('bedrock-runtime', region_name = region)
bedrock_agent_client = boto3.client("bedrock-agent-runtime",
                        config=bedrock_config, region_name = region)
print(region)
```

```
us-west-2
```

Part 1 - Retrieve API with foundation models from Amazon Bedrock

Define a retrieve function that calls the `Retreive API` provided by Knowledge Bases for Amazon Bedrock which converts user queries into embeddings, searches the knowledge base, and returns the relevant results, giving you more control to build custom workflows on top of the semantic search results. The output of the `Retrieve API` includes the the `retrieved text chunks` , the `location type` and `URI` of the source data, as well as the relevance `scores` of the retrievals. You can also use the `overrideSearchType` option in `retrievalConfiguration` which offers the choice to use either `HYBRID` or `SEMANTIC` . By default, it will select the right strategy for you to give you most relevant results, and if you want

[8]:
```
def retrieve(query, kbId, numberOfResults=5):
    return bedrock_agent_client.retrieve(
        retrievalQuery= {
            'text': query
        },
        knowledgeBaseId=kbId,
        retrievalConfiguration= {
            'vectorSearchConfiguration': {
                'numberOfResults': numberOfResults,
                'overrideSearchType': "HYBRID", # optional
            }
        }
    )
```

Initialize your Knowledge base id before querying responses from the initialized LLM

Next, we will call the `retreive API` , and pass `knowledge base id` , `number of results` and `query` as paramters.

`score` : You can view the associated score of each of the text chunk that was returned which depicts its correlation to the query in terms of how closely it matches it.

[9]:
```
query = "What is Amazon doing in the field of Generative AI?"
response = retrieve(query, kb_id, 5)
retrievalResults = response['retrievalResults']
pp.pprint(retrievalResults)
```

```
[ { 'content': { 'text': 'This shift was driven by several factors, including '
                         'access to higher volumes of compute capacity at '
                         'lower prices than was ever available. Amazon has '
                         'been using machine learning extensively for 25 '
                         'years, employing it in everything from personalized '
                         'ecommerce recommendations, to fulfillment center '
                         'pick paths, to drones for Prime Air, to Alexa, to '
                         'the many machine learning services AWS offers (where '
                         'AWS has the broadest machine learning functionality '
                         'and customer base of any cloud provider). More '
                         'recently, a newer form of machine learning, called '
                         'Generative AI, has burst onto the scene and promises '
                         'to significantly accelerate machine learning '
```

Extract the text chunks from the retrieveAPI response

In the cell below, we will fetch the context from the retrieval results.

[10]:
```
# fetch context from the response
def get_contexts(retrievalResults):
    contexts = []
    for retrievedResult in retrievalResults:
        contexts.append(retrievedResult['content']['text'])
    return contexts
```

[11]:
```
contexts = get_contexts(retrievalResults)
pp.pprint(contexts)
```

```
[ 'This shift was driven by several factors, including access to higher '
  'volumes of compute capacity at lower prices than was ever available. Amazon '
  'has been using machine learning extensively for 25 years, employing it in '
  'everything from personalized ecommerce recommendations, to fulfillment '
  'center pick paths, to drones for Prime Air, to Alexa, to the many machine '
  'learning services AWS offers (where AWS has the broadest machine learning '
  'functionality and customer base of any cloud provider). More recently, a '
  'newer form of machine learning, called Generative AI, has burst onto the '
  'scene and promises to significantly accelerate machine learning adoption. '
  'Generative AI is based on very Large Language Models (trained on up to '
  'hundreds of billions of parameters, and growing), across expansive '
  'datasets, and has radically general and broad recall and learning '
  'capabilities. We have been working on our own LLMs for a while now, believe '
  'it will transform and improve virtually every customer experience, and will '
  'continue to invest substantially in these models across all of our '
  'consumer, seller, brand, and creator experiences. Additionally, as we've '
  'done for years in AWS, we're democratizing this technology so companies of '
  'all sizes can leverage Generative AI. AWS is offering the most '
  'price-performant machine learning chips in Trainium and Inferentia so small '
  'and large companies can afford to train and run their LLMs in production. '
  'We enable companies to choose from various LLMs and build applications with '
  'all of the AWS security, privacy and other features that customers are '
  'accustomed to using. And, we're delivering applications like AWS's '
  'CodeWhisperer, which revolutionizes developer productivity by '
  'generating code suggestions in real time. I could write an entire letter on '
```

Prompt specific to the model to personalize responses

Here, we will use the specific prompt below for the model to act as a financial advisor AI system that will provide answers to questions by using fact based and statistical information when possible. We will provide the `Retrieve API` responses from above as a part of the `{contexts}` in the prompt for the model to refer to, along with the user `query`.

[12]:
```
prompt = f"""
Human: You are a financial advisor AI system, and provides answers to questions by using fact based and statistical information when possible.
Use the following pieces of information to provide a concise answer to the question enclosed in <question> tags.
If you don't know the answer, just say that you don't know, don't try to make up an answer.
<context>
{contexts}
</context>

<question>
{query}
</question>

The response should be specific and use statistics or numbers when possible.

Assistant:"""
```

Invoke foundation model from Amazon Bedrock

In this example, we will use `anthropic.claude-3-sonnet-20240229-v1:0` foundation model from Amazon Bedrock.

- It offers maximum utility at a lower price than competitors, and is engineered to be the dependable, high-endurance workhorse for scaled AI deployments. Claude 3 Sonnet can process images and return text outputs, and features a 200K context window.
- Model attributes
  - Image to text & code, multilingual conversation, complex reasoning & analysis

[13]:
```
# payload with model paramters
messages=[{ "role":'user', "content":[{'type':'text','text': prompt.format(contexts, query)}]}]
sonnet_payload = json.dumps({
    "anthropic_version": "bedrock-2023-05-31",
    "max_tokens": 512,
    "messages": messages,
```

Invoke foundation model from Amazon Bedrock

In this example, we will use `anthropic.claude-3-sonnet-20240229-v1:0` foundation model from Amazon Bedrock.

- It offers maximum utility at a lower price than competitors, and is engineered to be the dependable, high-endurance workhorse for scaled AI deployments. Claude 3 Sonnet can process images and return text outputs, and features a 200K context window.
- Model attributes
  - Image to text & code, multilingual conversation, complex reasoning & analysis

[13]:
```
# payload with model paramters
messages=[{ "role":'user', "content":[{'type':'text','text': prompt.format(contexts, query)}]}]
sonnet_payload = json.dumps({
    "anthropic_version": "bedrock-2023-05-31",
    "max_tokens": 512,
    "messages": messages,
    "temperature": 0.5,
    "top_p": 1
        } )
```

[19]:
```
# modelId = 'anthropic.claude-3-sonnet-20240229-v1:0' # change this to use a different version from the model provider
modelId = 'anthropic.claude-instant-v1' # change this to use a different version from the model provider
accept = 'application/json'
contentType = 'application/json'
response = bedrock_client.invoke_model(body=sonnet_payload, modelId=modelId, accept=accept, contentType=contentType)
response_body = json.loads(response.get('body').read())
response_text = response_body.get('content')[0]['text']

pp.pprint(response_text)
```

```
('According to the context provided:\n'
 '\n'
 '- Amazon has been working on their own Large Language Models (LLMs) for '
 'Generative AI for a while and believes it will transform and improve '
 'virtually every customer experience. They plan to continue investing '
 'substantially in these models across all of their consumer, seller, brand, '
 'and creator experiences. \n'
 '\n'
 '- Amazon is offering LLMs and enabling companies to build applications using '
```

Part 2 - LangChain integration

In this notebook, we will dive deep into building Q&A application using Retrieve API provided by Knowledge Bases for Amazon Bedrock and LangChain. We will query the knowledge base to get the desired number of document chunks based on similarity search, integrate it with LangChain retriever and use `Anthropic Claude 3 Sonnet` model for answering questions.

[20]:
```
# from langchain.llms.bedrock import Bedrock
from langchain_community.chat_models.bedrock import BedrockChat
from langchain.retrievers.bedrock import AmazonKnowledgeBasesRetriever

llm = BedrockChat(model_id=modelId,
                  client=bedrock_client)
```

Create a `AmazonKnowledgeBasesRetriever` object from LangChain which will call the `Retreive API` provided by Knowledge Bases for Amazon Bedrock which converts user queries into embeddings, searches the knowledge base, and returns the relevant results, giving you more control to build custom workflows on top of the semantic search results. The output of the `Retrieve API` includes the the `retrieved text chunks` , the `location type` and `URI` of the source data, as well as the relevance `scores` of the retrievals.

[21]:
```
query = "What is Amazon doing in the field of Generative AI?"
retriever = AmazonKnowledgeBasesRetriever(
        knowledge_base_id=kb_id,
        retrieval_config={"vectorSearchConfiguration":
                          {"numberOfResults": 4,
                           'overrideSearchType': "SEMANTIC", # optional
                           }
                          },
        # endpoint_url=endpoint_url,
        # region_name=region,
        # credentials_profile_name="<profile_name>",
    )
docs = retriever.get_relevant_documents(
        query=query
    )
pp.pprint(docs)
```

```
[ Document(page_content='This shift was driven by several factors, including access to higher volumes of compute capacity at lower prices than was ever availabl
e. Amazon has been using machine learning extensively for 25 years, employing it in everything from personalized ecommerce recommendations, to fulfillment cente
r pick paths, to drones for Prime Air, to Alexa, to the many machine learning services AWS offers (where AWS has the broadest machine learning functionality and
customer base of any cloud provider). More recently, a newer form of machine learning, called Generative AI, has burst onto the scene and promises to significan
tly accelerate machine learning adoption. Generative AI is based on very Large Language Models (trained on up to hundreds of billions of parameters, and growin
```

Prompt specific to the model to personalize responses

Here, we will use the specific prompt below for the model to act as a financial advisor AI system that will provide answers to questions by using fact based and statistical information when possible. We will provide the Retrieve API responses from above as a part of the `{context}` in the prompt for the model to refer to, along with the user `query`.

[22]:
```
from langchain.prompts import PromptTemplate

PROMPT_TEMPLATE = """
Human: You are a financial advisor AI system, and provides answers to questions by using fact based and statistical information when possible.
Use the following pieces of information to provide a concise answer to the question enclosed in <question> tags.
If you don't know the answer, just say that you don't know, don't try to make up an answer.
<context>
{context}
</context>

<question>
{question}
</question>

The response should be specific and use statistics or numbers when possible.

Assistant:"""
claude_prompt = PromptTemplate(template=PROMPT_TEMPLATE,
                               input_variables=["context","question"])
```

Integrating the retriever and the LLM defined above with RetrievalQA Chain to build the Q&A application.

[23]:
```
from langchain.chains import RetrievalQA

qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    return_source_documents=True,
    chain_type_kwargs={"prompt": claude_prompt}
)
```

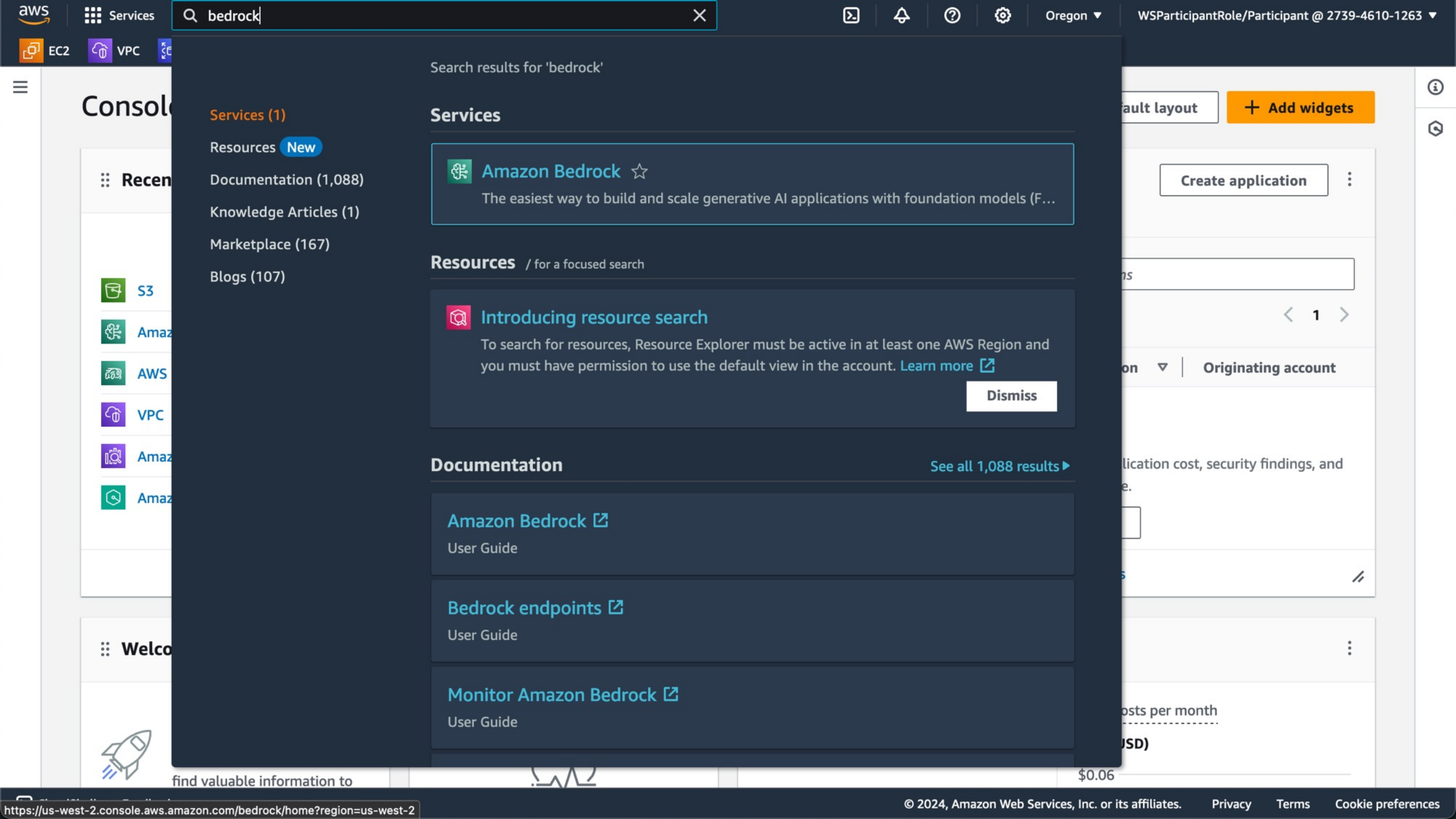
[23]:
```
from langchain.chains import RetrievalQA

qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    return_source_documents=True,
    chain_type_kwargs={"prompt": claude_prompt}
)
```

[24]:
```
answer = qa.invoke(query)
pp.pprint(answer)
```

```
{ 'query': 'What is Amazon doing in the field of Generative AI?',
  'result': 'From the context provided:\n'
            '\n'
            'Amazon has been working on their own large language models (LLMs) '
            'for Generative AI for a while now. They believe Generative AI, '
            'based on LLMs trained on up to hundreds of billions of parameters '
            'across expansive datasets, will transform and improve virtually '
            'every customer experience. Amazon is continuing to invest '
            'substantially in these models across all of their consumer, '
            'seller, brand, and creator experiences. Additionally, Amazon is '
            'democratizing this technology through AWS so companies of all '
            'sizes can leverage Generative AI. AWS offers various LLMs and '
            'enables companies to build applications using Generative AI with '
            "AWS's security, privacy and other features. One example "
            "application mentioned is AWS's CodeWhisperer, which uses "
            'Generative AI to generate code suggestions in real time to '
            'revolutionize developer productivity.',
  'source_documents': [ Document(page_content='This shift was driven by several factors, including access to higher volumes of compute capacity at lower prices
than was ever available. Amazon has been using machine learning extensively for 25 years, employing it in everything from personalized ecommerce recommendation
s, to fulfillment center pick paths, to drones for Prime Air, to Alexa, to the many machine learning services AWS offers (where AWS has the broadest machine lea
rning functionality and customer base of any cloud provider). More recently, a newer form of machine learning, called Generative AI, has burst onto the scene an
d promises to significantly accelerate machine learning adoption. Generative AI is based on very Large Language Models (trained on up to hundreds of billions of
parameters, and growing), across expansive datasets, and has radically general and broad recall and learning capabilities. We have been working on our own LLMs
for a while now, believe it will transform and improve virtually every customer experience, and will continue to invest substantially in these models across all
of our consumer, seller, brand, and creator experiences. Additionally, as we've done for years in AWS, we're democratizing this technology so companies of all s
izes can leverage Generative AI. AWS is offering the most price-performant machine learning chips in Trainium and Inferentia so small and large companies can af
ford to train and run their LLMs in production. We enable companies to choose from various LLMs and build applications with all of the AWS security, privacy and
```

Hands-On Lab


aws
Services
bedrock
Oregon
WSParticipantRole/Participant @ 2739-4610-1263
EC2
VPC
Search results for 'bedrock'
Services (1)
Resources New
Documentation (1,088)
Knowledge Articles (1)
Marketplace (167)
Blogs (107)
Services
Amazon Bedrock
The easiest way to build and scale generative AI applications with foundation models (F…
Resources / for a focused search
Introducing resource search
To search for resources, Resource Explorer must be active in at least one AWS Region and you must have permission to use the default view in the account. Learn more
Dismiss
Documentation
See all 1,088 results
Amazon Bedrock
User Guide
Bedrock endpoints
User Guide
Monitor Amazon Bedrock
User Guide
Add widgets
Create application
Originating account
S3
VPC
$0.06
https://us-west-2.console.aws.amazon.com/bedrock/home?region=us-west-2
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Machine Learning

Amazon Bedrock

The easiest way to build and scale generative AI applications with foundation models (FMs)

Try Bedrock

Get started

Overview

Amazon Bedrock is a fully managed service that makes FMs from leading AI startups and Amazon available via an API, so you can choose from a wide range of FMs to find the model that is best suited for your use case. With Bedrock's serverless experience, you can get started quickly, privately customize FMs with your own data, and easily integrate and deploy them into your applications using the AWS tools without having to manage any infrastructure.

Benefits

Accelerate development of generative

Choose FMs from AI21 Labs,

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Machine Learning

Amazon Bedrock

The easiest way to build and scale generative AI applications with foundation models (FMs)

Try Bedrock

Get started

Overview

Amazon Bedrock is a fully managed service that makes FMs from leading AI startups and Amazon available via an API, so you can choose from a wide range of FMs to find the model that is best suited for your use case. With Bedrock's serverless experience, you can get started quickly, privately customize FMs with your own data, and easily integrate and deploy them into your applications using the AWS tools without having to manage any infrastructure.

Benefits

Accelerate development of generative AI applications using FMs

Choose FMs from AI21 Labs, Anthropic, Stability AI, and Amazon

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Knowledge bases

How it works

Create a knowledge base

To create a knowledge base, specify the location of your data, select an embedding model, and configure a vector store for Bedrock to store and update your embeddings.

Test the knowledge base

Query your knowledge base in the test window. You can either retrieve source text chunks or generate responses using the chunks by connecting to a foundation model.

Use the knowledge base

Integrate your knowledge base into your application as is or add it to agents.

Knowledge bases (0)

Edit | Delete | Test knowledge base | Create knowledge base

Find knowledge base

1

Name	Status	Description	Source files	Creation time	Last sync warnings	Last sync

No knowledge base

No knowledge base to display

Create knowledge base

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Provide knowledge base details

Knowledge base details

Knowledge base name

knowledge-base-quick-start-oghwe

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 50 characters.

Knowledge base description - *optional*

Enter description

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 200 characters.

IAM permissions

IAM roles are used to access other services on your behalf.

Runtime role

- (●) Create and use a new service role
- ( ) Use an existing service role

Service role name

AmazonBedrockExecutionRoleForKnowledgeBase_oghwe

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Provide knowledge base details

Knowledge base details

Knowledge base name

knowledge-base-mfg-day-workshop

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 50 characters.

Knowledge base description - *optional*

Enter description

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 200 characters.

IAM permissions

IAM roles are used to access other services on your behalf.

Runtime role

- (●) Create and use a new service role
- ( ) Use an existing service role

Service role name

AmazonBedrockExecutionRoleForKnowledgeBase_oghwe

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Amazon Bedrock > Knowledge bases > Create knowledge base

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Provide knowledge base details

Knowledge base details

Knowledge base name

knowledge-base-mfg-day-workshop

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 50 characters.

Knowledge base description - *optional*

Enter description

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 200 characters.

IAM permissions

IAM roles are used to access other services on your behalf.

Runtime role

- (●) Create and use a new service role
- ( ) Use an existing service role

Service role name

AmazonBedrockExecutionRoleForKnowledgeBase_mfg-day-workshop

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Enter description

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 200 characters.

IAM permissions

IAM roles are used to access other services on your behalf.

Runtime role

- (•) Create and use a new service role
- ( ) Use an existing service role

Service role name

AmazonBedrockExecutionRoleForKnowledgeBase_mfg-day-workshop

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Next

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Set up data source

Set up your data source by specifying the S3 location of your data.

Data source: knowledge-base-quick-start-q5idm-data-source

Data source name

knowledge-base-quick-start-q5idm-data-source

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 100 characters.

S3 URI

Choose a s3 location | View | Browse S3

Add customer-managed KMS key for S3 data - *optional*
If you encrypted your S3 data, provide the KMS key here so that Bedrock can decrypt it.

Advanced settings - *optional*

Cancel Previous Next

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Step 1
Provide knowledge base details

Step 2
Set up data source

Step 3
Select embeddings model and configure vector store

Step 4
Review and create

Set up data source

Set up your data source by specifying the S3 location of your data.

Data source: knowledge-base-mfg-day-workshop

Data source name

knowledge-base-mfg-day-workshop

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 100 characters.

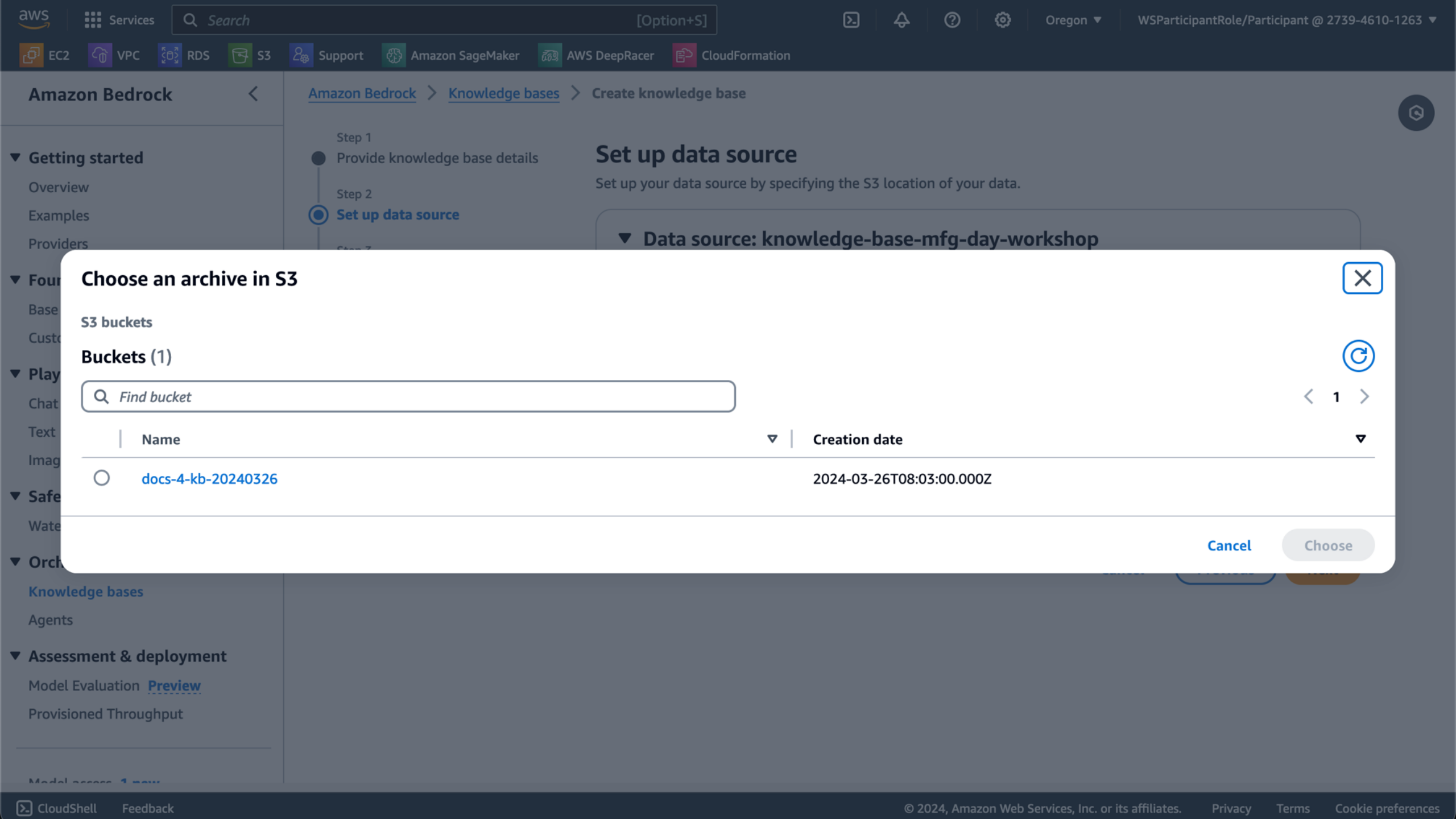
S3 URI

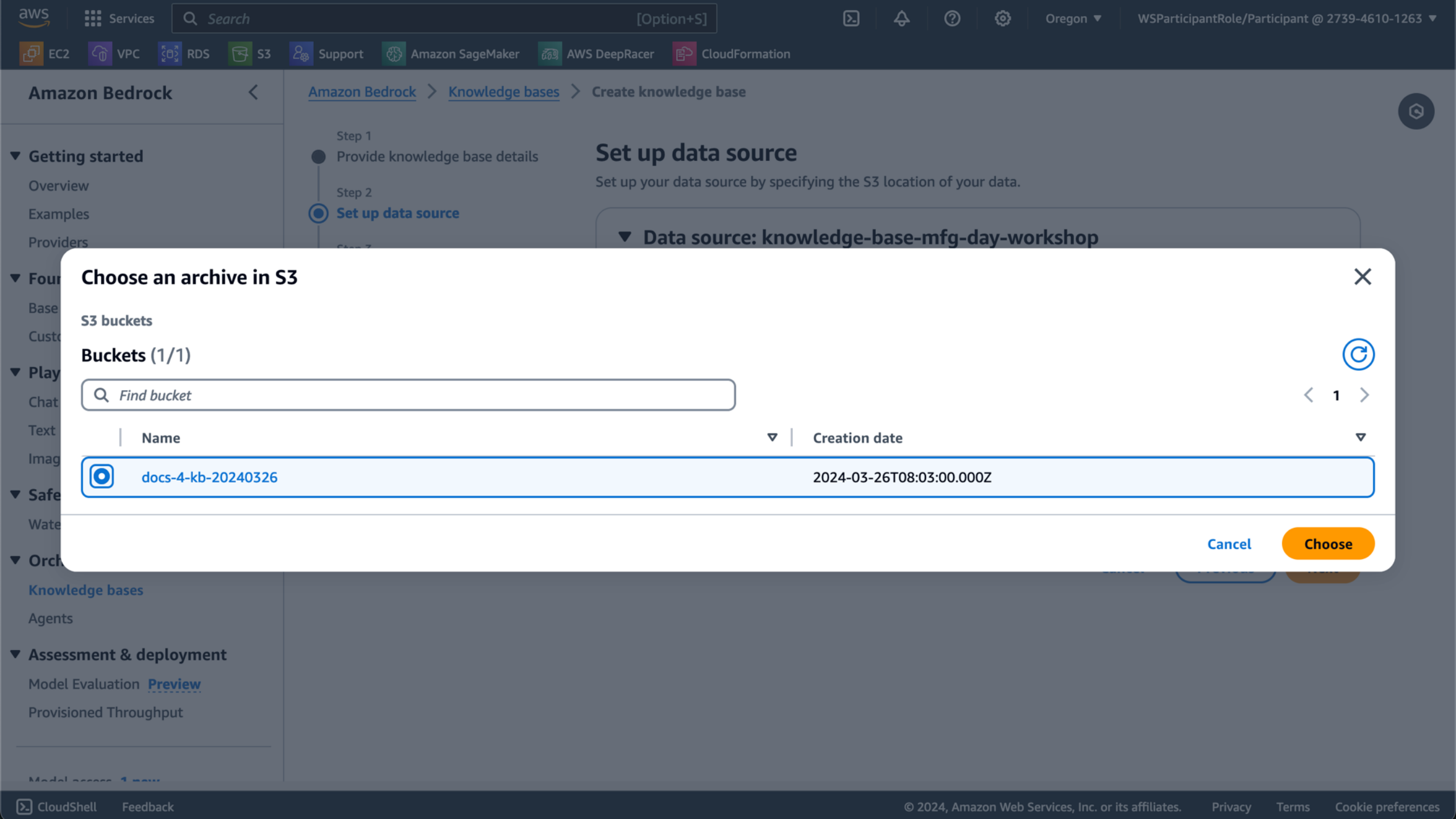
Choose a s3 location | View | Browse S3

Add customer-managed KMS key for S3 data - *optional*
If you encrypted your S3 data, provide the KMS key here so that Bedrock can decrypt it.

Advanced settings - *optional*

Cancel | Previous | Next


aws
Services
Search
[Option+S]
Oregon
WSParticipantRole/Participant @ 2739-4610-1263
EC2
VPC
RDS
S3
Support
Amazon SageMaker
AWS DeepRacer
CloudFormation
Amazon Bedrock
Getting started
Overview
Examples
Providers
Base
Chat
Text
Knowledge bases
Agents
Assessment & deployment
Model Evaluation
Preview
Provisioned Throughput
Amazon Bedrock > Knowledge bases > Create knowledge base
Step 1
Provide knowledge base details
Step 2
Set up data source
Set up data source
Set up your data source by specifying the S3 location of your data.
Data source: knowledge-base-mfg-day-workshop
Choose an archive in S3
S3 buckets
Buckets (1)
Find bucket
1
Name
Creation date
docs-4-kb-20240326
2024-03-26T08:03:00.000Z
Cancel
Choose
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Option+S]
Oregon
WSParticipantRole/Participant @ 2739-4610-1263
EC2
VPC
RDS
S3
Support
Amazon SageMaker
AWS DeepRacer
CloudFormation
Amazon Bedrock
Getting started
Overview
Examples
Providers
Knowledge bases
Agents
Assessment & deployment
Model Evaluation
Preview
Provisioned Throughput
Amazon Bedrock > Knowledge bases > Create knowledge base
Step 1
Provide knowledge base details
Step 2
Set up data source
Set up data source
Set up your data source by specifying the S3 location of your data.
Data source: knowledge-base-mfg-day-workshop
Choose an archive in S3
S3 buckets
Buckets (1/1)
Find bucket
1
Name
Creation date
docs-4-kb-20240326
2024-03-26T08:03:00.000Z
Cancel
Choose
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Amazon Bedrock

Getting started
Overview
Examples
Providers
Foundation models
Base models
Custom models
Playgrounds
Chat
Text
Image
Safeguards
Watermark detection Preview
Orchestration
Knowledge bases
Agents
Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Step 1
Provide knowledge base details

Step 2
Set up data source

Step 3
Select embeddings model and configure vector store

Step 4
Review and create

Set up data source

Set up your data source by specifying the S3 location of your data.

Data source: knowledge-base-mfg-day-workshop

Data source name

knowledge-base-mfg-day-workshop

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 100 characters.

S3 URI

s3://docs-4-kb-20240326

View | Browse S3

Add customer-managed KMS key for S3 data - *optional*
If you encrypted your S3 data, provide the KMS key here so that Bedrock can decrypt it.

Advanced settings - *optional*

Cancel | Previous | Next

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Amazon Bedrock > Knowledge bases > Create knowledge base

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Select embeddings model and configure vector store

Choose an embeddings model to convert the data that you will provide in the next step, and provide details for a vector data store in which Bedrock can store, manage, and update your embeddings. Embeddings model and vector store cannot be changed after creation of knowledge base.

Embeddings model

Select an embeddings model to convert your data into an embedding. Pricing depends on the model. Learn more

- Titan Embeddings G1 - Text v1.2
  By Amazon | Vector dimensions: 1536
- Embed English v3
  By Cohere | Vector dimensions: 1024
- Embed Multilingual v3
  By Cohere | Vector dimensions: 1024

Vector database

Let Amazon create a vector store on your behalf or select a previously created store to allow Bedrock to store, update and manage embeddings. You will be billed directly from the vector store provider. Learn more

Select how you want to create your vector store.

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Amazon Bedrock > Knowledge bases > Create knowledge base

Step 1
Provide knowledge base details
Step 2
Set up data source
Step 3
Select embeddings model and configure vector store
Step 4
Review and create

Select embeddings model and configure vector store

Choose an embeddings model to convert the data that you will provide in the next step, and provide details for a vector data store in which Bedrock can store, manage, and update your embeddings. Embeddings model and vector store cannot be changed after creation of knowledge base.

Embeddings model

Select an embeddings model to convert your data into an embedding. Pricing depends on the model. Learn more

- Titan Embeddings G1 - Text v1.2 — By Amazon | Vector dimensions: 1536 (selected)
- Embed English v3 — By Cohere | Vector dimensions: 1024
- Embed Multilingual v3 — By Cohere | Vector dimensions: 1024

Vector database

Let Amazon create a vector store on your behalf or select a previously created store to allow Bedrock to store, update and manage embeddings. You will be billed directly from the vector store provider. Learn more

Select how you want to create your vector store.

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Embed Multilingual v3

By Cohere | Vector dimensions: 1024

Vector database

Let Amazon create a vector store on your behalf or select a previously created store to allow Bedrock to store, update and manage embeddings. You will be billed directly from the vector store provider. Learn more

Select how you want to create your vector store.

- (•) Quick create a new vector store - *Recommended*
  We will create an Amazon OpenSearch Serverless vector store on your behalf. This cost-efficient option is intended only for development and can't be migrated to production workload later. Learn more
- ( ) Choose a vector store you have created
  Select Amazon OpenSearch Serverless, Amazon Aurora, Pinecone or Redis Enterprise Cloud and provide field mappings.

- [ ] Enable redundancy (active replicas) - *optional*
  The default configuration has active replicas disabled, which is optimal for development workloads. Enable this option if you want to enable redundant active replicas, which may increase storage costs.

- [ ] Add customer-managed KMS key for Amazon Opensearch Serverless vector - optional
  If you encrypted your Opensearch data, provide the KMS key here so that Bedrock can decrypt it.

Cancel Previous Next

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Quick create a new vector store - *Recommended*
We will create an Amazon OpenSearch Serverless vector store on your behalf. This cost-efficient option is intended only for development and can't be migrated to production workload later. Learn more

Choose a vector store you have created
Select Amazon OpenSearch Serverless, Amazon Aurora, Pinecone or Redis Enterprise Cloud and provide field mappings.

Select an existing database

Vector engine for Amazon OpenSearch Serverless
If you are a first time user, create a vector database by visiting OpenSearch Service

Amazon Aurora
If you are a first time user, create a vector database by visiting RDS Console

Pinecone
If you are a first time user, create a vector database by visiting Pinecone

Redis Enterprise Cloud
If you are a first time user, create a vector database by visiting Redis Enterprise Cloud

Collection ARN
Specify ARN of your OpenSearch Serverless Collection

Enter collection ARN. For example: "arn:aws:aoss:us-east-1:149020153069:collection/lgqni2uq61hckn6qv0le"

Vector index name

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Embed Multilingual v3

By Cohere | Vector dimensions: 1024

Vector database

Let Amazon create a vector store on your behalf or select a previously created store to allow Bedrock to store, update and manage embeddings. You will be billed directly from the vector store provider. Learn more

Select how you want to create your vector store.

- Quick create a new vector store - *Recommended*
  We will create an Amazon OpenSearch Serverless vector store on your behalf. This cost-efficient option is intended only for development and can't be migrated to production workload later. Learn more
- Choose a vector store you have created
  Select Amazon OpenSearch Serverless, Amazon Aurora, Pinecone or Redis Enterprise Cloud and provide field mappings.

- Enable redundancy (active replicas) - *optional*
  The default configuration has active replicas disabled, which is optimal for development workloads. Enable this option if you want to enable redundant active replicas, which may increase storage costs.

- Add customer-managed KMS key for Amazon Opensearch Serverless vector - optional
  If you encrypted your Opensearch data, provide the KMS key here so that Bedrock can decrypt it.

Cancel Previous Next

Amazon Bedrock

- Getting started
  - Overview
  - Examples
  - Providers
- Foundation models
  - Base models
  - Custom models
- Playgrounds
  - Chat
  - Text
  - Image
- Safeguards
  - Watermark detection Preview
- Orchestration
  - Knowledge bases
  - Agents
- Assessment & deployment
  - Model Evaluation Preview
  - Provisioned Throughput

Model access 1 new

Amazon Bedrock > Knowledge bases > Create knowledge base

Step 1 Provide knowledge base details
Step 2 Set up data source
Step 3 Select embeddings model and configure vector store
Step 4 Review and create

Review and create

Step 1: Provide details

Edit

Knowledge base details

Knowledge base name
knowledge-base-mfg-day-workshop

Knowledge base description
—

Service role
AmazonBedrockExecutionRoleForKnowledgeBase_mfg-day-workshop

Tags (0)

Key	Value

No tags to display

Step 2: Setup up data source

Edit

Data source: knowledge-base-mfg-day-workshop

Data source name

S3 URI

Amazon Bedrock

Getting started
Overview
Examples
Providers

Foundation models
Base models
Custom models

Playgrounds
Chat
Text
Image

Safeguards
Watermark detection Preview

Orchestration
Knowledge bases
Agents

Assessment & deployment
Model Evaluation Preview
Provisioned Throughput

Step 2: Setup up data source

Edit

Data source: knowledge-base-mfg-day-workshop

Data source name
knowledge-base-mfg-day-workshop

S3 URI
s3://docs-4-kb-20240326

Customer-managed KMS Key for S3
—

KMS key for transient data storage
—

Step 3: Select embeddings model and configure vector store

Edit

Embeddings model

Model
Titan Embeddings G1 - Text v1.2
1536

Vector store

Quick create vector store - *Recommended*
We will create an Amazon OpenSearch Serverless vector store in your account on your behalf.

Cancel Previous **Create knowledge base**

Thank you!

Please complete the session survey in the mobile app

Michael Lin

linmicht@amazon.com