[Workshop Studio](https://catalog.workshops.aws/)

[**Using generative AI on AWS for diverse content types**](https://catalog.workshops.aws/genai-on-aws/en-US)

* [Generative AI on AWS](https://catalog.workshops.aws/genai-on-aws/en-US/01-introduction)
* [Environment Setup](https://catalog.workshops.aws/genai-on-aws/en-US/02-start-workshop)
* [Stream 1: Getting started with generative AI](https://catalog.workshops.aws/genai-on-aws/en-US/03-getting-started)
* [Stream 2: Working with private documents](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents)
  + [Using Amazon Bedrock and Amazon OpenSearch Service](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/01-using-opensearch)
  + [Using Amazon Bedrock and Amazon RDS for PostgreSQL](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/02-using-postgresql)
    - [Configure the application](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/02-using-postgresql/01-configure-application)
    - [Update the code](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/02-using-postgresql/02-update-code)
    - [Run the application](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/02-using-postgresql/03-run-application)
  + [Using Amazon Bedrock and Amazon Kendra](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/03-using-kendra)
* [Stream 3: Working with image files](https://catalog.workshops.aws/genai-on-aws/en-US/05-working-with-image-files)
* [Stream 4: Working with video files](https://catalog.workshops.aws/genai-on-aws/en-US/06-working-with-video-files)
* [Stream 5: Working with audio files](https://catalog.workshops.aws/genai-on-aws/en-US/07-working-with-audio-files)
* [Stream 6: Working with SQL database](https://catalog.workshops.aws/genai-on-aws/en-US/08-working-with-databases)
* [Stream 7: Working with graphs](https://catalog.workshops.aws/genai-on-aws/en-US/09-working-with-graphs)
* [Stream 8: Working with business intelligence](https://catalog.workshops.aws/genai-on-aws/en-US/10-working-with-bi)
* [Stream 9: Working with logs](https://catalog.workshops.aws/genai-on-aws/en-US/11-working-with-logs)
* [Stream 10: Working with streaming data](https://catalog.workshops.aws/genai-on-aws/en-US/12-working-with-streaming-data)
* [Stream 11: Working with CSV files](https://catalog.workshops.aws/genai-on-aws/en-US/13-working-with-csv-files)
* [Cleanup](https://catalog.workshops.aws/genai-on-aws/en-US/14-cleanup)
* [Authors](https://catalog.workshops.aws/genai-on-aws/en-US/15-release)

Content preferences

Language

English

1. [Using generative AI on AWS for diverse content types](https://catalog.workshops.aws/genai-on-aws/en-US)
2. [Stream 2: Working with private documents](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents)
3. [Using Amazon Bedrock and Amazon RDS for PostgreSQL](https://catalog.workshops.aws/genai-on-aws/en-US/04-working-with-private-documents/02-using-postgresql)
4. **Configure the application**

Configure the application

In this section, you will be configuring [Knowledge-Base for Amazon Bedrock](https://aws.amazon.com/bedrock/knowledge-bases/). With Knowledge Base for Amazon Bedrock, you can give FMs and agents contextual information from your company’s private data sources for Retrieval Augmented Generation (RAG) to deliver more relevant, accurate, and customized responses

1. In the **Cloud9** terminal, enter the following command to go to the bedrock\_and\_pgvector directory.

1

cd ~/environment/stream2\_working\_with\_private\_documents/bedrock\_and\_pgvector/

1. To download required dependencies for the application, run the following command in the terminal.

1

python3.9 -m pip install -r requirements.txt

1. Run the following command in the terminal to install postgres and psql on this Cloud9 instance.

1

2

3

4

sudo amazon-linux-extras enable postgresql12

sudo yum clean metadata

sudo yum install postgresql -y

psql --version

1. Now, run the below command to set the required environment variables.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

ROLE\_ARN=$(aws cloudformation describe-stacks \

--stack-name 'using-genai-for-private-files-workshop' \

--query 'Stacks[0].Outputs[?OutputKey==`AmazonBedrockExecutionRoleARN`].OutputValue' \

--output text)

echo $ROLE\_ARN

REGION=$(aws configure get region)

echo $REGION

Aurora\_Cluster\_Writer\_Endpoint=$(aws cloudformation describe-stacks \

--stack-name 'using-genai-for-private-files-workshop' \

--query 'Stacks[0].Outputs[?OutputKey==`AuroraDBWriterEndpoint`].OutputValue' \

--output text)

echo $Aurora\_Cluster\_Writer\_Endpoint

Aurora\_Cluster\_Arn=$(aws cloudformation describe-stacks \

--stack-name 'using-genai-for-private-files-workshop' \

--query 'Stacks[0].Outputs[?OutputKey==`AuroraDBClusterARN`].OutputValue' \

--output text)

echo $Aurora\_Cluster\_Arn

Postgres\_Bucket\_Name=$(aws cloudformation describe-stacks \

--stack-name 'using-genai-for-private-files-workshop' \

--query 'Stacks[0].Outputs[?OutputKey==`WorkshopPostgreS3BucketName`].OutputValue' \

--output text)

echo $Postgres\_Bucket\_Name

Postgres\_Bucket\_Arn=$(aws cloudformation describe-stacks \

--stack-name 'using-genai-for-private-files-workshop' \

--query 'Stacks[0].Outputs[?OutputKey==`WorkshopPostgreS3BucketARN`].OutputValue' \

--output text)

echo $Postgres\_Bucket\_Arn

DB\_Password=$(aws secretsmanager get-secret-value --secret-id postgres\_vectors\_secrets --query SecretString --output text | jq -r .password)

echo "DB\_Password: $DB\_Password"

1. Use the following command in the terminal to connect to the DB cluster and use the use the password printed in the terminal above.

1

psql -h $Aurora\_Cluster\_Writer\_Endpoint -d postgres -U workshopUser -p 3306

1. To create an extension named "vector" in a PostgreSQL database, run the following command in Cloud9 terminal.

1

CREATE EXTENSION IF NOT EXISTS vector;

1. Use the following command to retrieve the version number of the installed 'vector' extension in a PostgreSQL database.

1

SELECT extversion FROM pg\_extension WHERE extname='vector';

1. Run the following command in Cloud9 terminal to create a new schema named bedrock\_integration in a PostgreSQL database.

1

CREATE SCHEMA bedrock\_integration;

1. In the **Cloud9** terminal, enter the following command to create a new user and grant required permissions

1

2

CREATE ROLE bedrock\_user WITH PASSWORD 'workshoppassword@1' LOGIN;

GRANT ALL ON SCHEMA bedrock\_integration to bedrock\_user;

1. Then log in using the new user and password you just created.

1

2

\q

psql -h $Aurora\_Cluster\_Writer\_Endpoint -d postgres -U bedrock\_user -p 3306

1. Now, create a table with the right schema using the below command in the Cloud9 terminal.

1

CREATE TABLE bedrock\_integration.bedrock\_kb (id uuid PRIMARY KEY, embedding vector(1536), chunks text, metadata json);

1. Create an index named bedrock\_kb on the embedding column of a table within the bedrock\_integration schema, using the vector\_cosine\_ops operator class in PostgreSQL

1

2

CREATE INDEX on bedrock\_integration.bedrock\_kb USING hnsw (embedding vector\_cosine\_ops);

\q

1. Add the secrets to secrets manager

* Go to the [AWS Secrets Manager Console](https://console.aws.amazon.com/secretsmanager/listsecrets)
* Click the button Store a new secret
* Select secret type as Credentials for Amazon RDS database
* Provide the username bedrock\_user and password workshoppassword@1
* Under Database, select workshopdbcluster
* Click Next and provide secret name postgres\_kb\_secret
* Click Next > Next > Store

1. To retrieve the ARN of the secrets you created above, use the following command in your Cloud9 terminal.

1

2

SECRET\_ARN=$(aws secretsmanager list-secrets --query 'SecretList[?Name==`postgres\_kb\_secret`].ARN' --output text)

echo $SECRET\_ARN

1. To create knowledge base in Amazon Bedrock, use the following command in Cloud9 terminal.

1

2

3

4

5

6

7

Knowledge\_Base\_Id=$(aws bedrock-agent create-knowledge-base \

--name workshop-aurora-knowledge-base \

--role-arn $ROLE\_ARN \

--knowledge-base-configuration 'type=VECTOR,vectorKnowledgeBaseConfiguration={embeddingModelArn="arn:aws:bedrock:'"$REGION"'::foundation-model/amazon.titan-embed-text-v1"}' \

--storage-configuration 'type=RDS,rdsConfiguration={resourceArn='"$Aurora\_Cluster\_Arn"',credentialsSecretArn='"$SECRET\_ARN"',databaseName=postgres,tableName=bedrock\_integration.bedrock\_kb,fieldMapping={primaryKeyField=id,vectorField=embedding,textField=chunks,metadataField=metadata}}'\

| jq -r '.knowledgeBase.knowledgeBaseId')

echo $Knowledge\_Base\_Id

1. Follow the steps below to add a secret value in secrets manager
   * Go to the [AWS Secrets Manager Console](https://console.aws.amazon.com/secretsmanager/listsecrets)
   * Click on **postgres\_vectors\_secrets** secret
   * Under **Secret value** click **Retrieve secret value**
   * Then, click **Edit**
   * Click **+ Add row**
   * For key type **KNOWLEDGE\_BASE\_ID** and for the value paste the value of KNOWLEDGE\_BASE\_ID printed in the terminal above
   * Click **Save**
2. To create a data store within the knowledge base, use the following command in Cloud9 terminal.

1

2

3

4

5

6

7

Data\_Source\_Id=$(aws bedrock-agent create-data-source \

--knowledge-base-id $Knowledge\_Base\_Id \

--name workshop-aurora-kb-data-source \

--data-source-configuration "type=S3,s3Configuration={bucketArn='"$Postgres\_Bucket\_Arn"'}" \

--vector-ingestion-configuration "chunkingConfiguration={chunkingStrategy=NONE}" \

| jq -r '.dataSource.dataSourceId')

echo $Data\_Source\_Id

Previous

Next

© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.[Privacy policy](http://aws.amazon.com/privacy)[Terms of use](http://aws.amazon.com/terms)Cookie preferences

Update the code

In this section, you will be upading code to o use the [RetrieveAndGenerate](https://docs.aws.amazon.com/bedrock/latest/APIReference/API_agent-runtime_RetrieveAndGenerate.html) API to query your knowledge base and generate responses from the information it retrieves.

1. In Cloud9, go to the stream2\_working\_with\_private\_documents/bedrock\_and\_pgvector folder, open the app.py file and paste the below code.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

import streamlit as st

import boto3

import json

import os

# CSS for the chat interface and responses

st.markdown('''

<style>

.chat-message {padding: 1.5rem; border-radius: 0.5rem; margin-bottom: 1rem; display: flex}

.chat-message.user {background-color: #2b313e}

.chat-message.bot {background-color: #475063}

.chat-message .avatar {width: 20%}

.chat-message .avatar img {max-width: 78px; max-height: 78px; border-radius: 50%; object-fit: cover}

.chat-message .message {width: 80%; padding: 0 1.5rem; color: #fff}

.response, .url {background-color: #f0f0f0; padding: 1rem; border-radius: 0.5rem; margin-bottom: 1rem;}

</style>

''', unsafe\_allow\_html=True)

# Message templates

bot\_template = '''

<div class="chat-message bot">

<div class="avatar">

<img src="https://i.ibb.co/cN0nmSj/Screenshot-2023-05-28-at-02-37-21.png">

</div>

<div class="message">{{MSG}}</div>

</div>

'''

user\_template = '''

<div class="chat-message user">

<div class="avatar">

<img src="https://i.ibb.co/wRtZstJ/Aurora.png">

</div>

<div class="message">{{MSG}}</div>

</div>

'''

secret\_name = "postgres\_vectors\_secrets"

st.title("Chat with Bedrock Knowledge-Base")

session = boto3.session.Session()

region\_name = session.region\_name

bedrock\_client = boto3.client('bedrock-agent-runtime')

client = session.client(

service\_name='secretsmanager',

region\_name=region\_name

)

get\_secret\_value\_response = client.get\_secret\_value(

SecretId=secret\_name

)

secret = get\_secret\_value\_response['SecretString']

parsed\_secret = json.loads(secret)

knowledge\_base\_id = parsed\_secret["KNOWLEDGE\_BASE\_ID"]

# Initialize conversation history if not present

if 'conversation\_history' not in st.session\_state:

st.session\_state.conversation\_history = []

user\_input = st.text\_input("You: ")

if st.button("Send"):

# Retrieve and Generate call

response = bedrock\_client.retrieve\_and\_generate(

input={"text": user\_input},

retrieveAndGenerateConfiguration={

"knowledgeBaseConfiguration": {

"knowledgeBaseId": knowledge\_base\_id,

"modelArn": f"arn:aws:bedrock:{region\_name}::foundation-model/anthropic.claude-v2"

},

"type": "KNOWLEDGE\_BASE"

}

)

# Extract response

response\_text = response['output']['text']

# Check if there are any retrieved references

if not response['citations'][0]['retrievedReferences']:

# No references found, use the response text

display\_text = response\_text

else:

# Handle normal case with references

# Extract S3 URI (assuming references are present)

s3\_uri = response['citations'][0]['retrievedReferences'][0]['location']['s3Location']['uri']

display\_text = f"{response\_text}<br><br>Reference: {s3\_uri}"

# Insert the response at the beginning of the conversation history

st.session\_state.conversation\_history.insert(0, ("Assistant", f"<div class='response'>{display\_text}</div>"))

st.session\_state.conversation\_history.insert(0, ("You", user\_input))

# Display conversation history

for speaker, text in st.session\_state.conversation\_history:

if speaker == "You":

st.markdown(user\_template.replace("{{MSG}}", text), unsafe\_allow\_html=True)

else:

st.markdown(text, unsafe\_allow\_html=True)

Run the application

1. In the **Cloud9** terminal, enter the following command to go to the bedrock\_and\_pgvector directory.

1

cd ~/environment/stream2\_working\_with\_private\_documents/bedrock\_and\_pgvector/

1. In the **Cloud9** terminal, run the following command to download the dataset using LangChain wikipedia retriever.

1

python download\_dataset.py

1. In the **Cloud9** terminal, run the following command to upload a dataset to Amazon S3.

1

aws s3 cp ./docs s3://${Postgres\_Bucket\_Name}/ --recursive

1. In the **Cloud9** terminal, run the following command to sync data in knowledge-base

1

aws bedrock-agent start-ingestion-job --knowledge-base-id $Knowledge\_Base\_Id --data-source-id $Data\_Source\_Id

**Note:** Please wait for a few seconds for the file to be fully indexed before continuing.

1. In the **Cloud9** terminal, enter the following command to run the Streamlit app.

1

streamlit run app.py

1. The output will display two IP addressess. Open the endpoint with the **External IP** address.

A screen shot of a computer

Description automatically generated

1. In the browser, ask questions related to the document. For example,

* How many people live in New York City?
* When was the original name of Hong Kong?
* What is the history of Singapore?