Chat with your PDF - Gen AI App - With Amazon Bedrock, RAG, S3, Langchain and Streamlit

we will build a CHATBOT like application with AWS Amazon Bedrock, docker, python, Langchain, and Streamlit. We will use Retrieval-Augmented generation concept to provide context to the Large Language model along with user query to generate response from our Knowledgebase

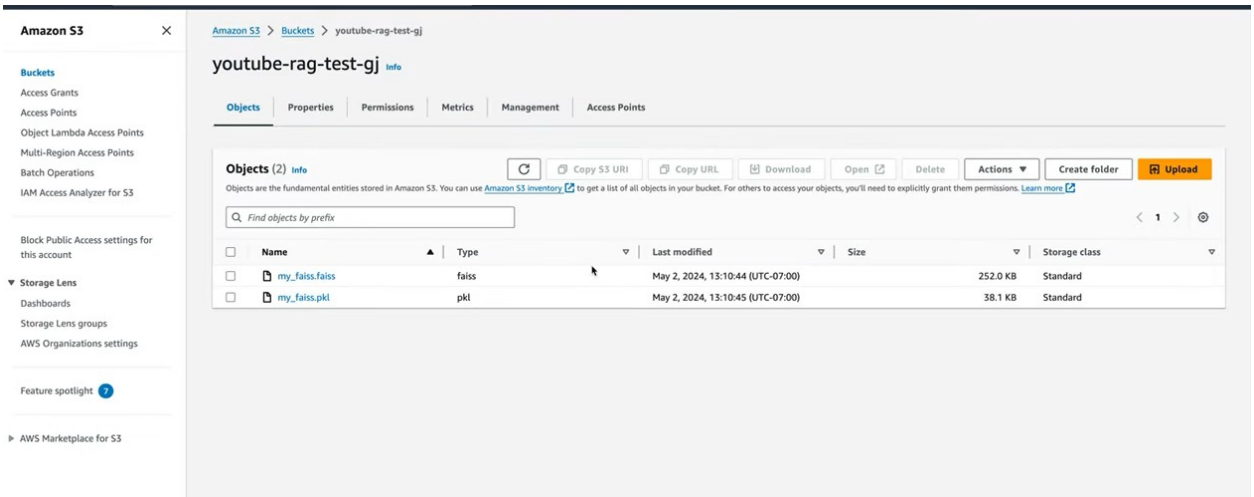
I will demonstrate the following: -

Architecture of the applications - Build 2 applications (ADMIN and USER) and create DOCKER images –

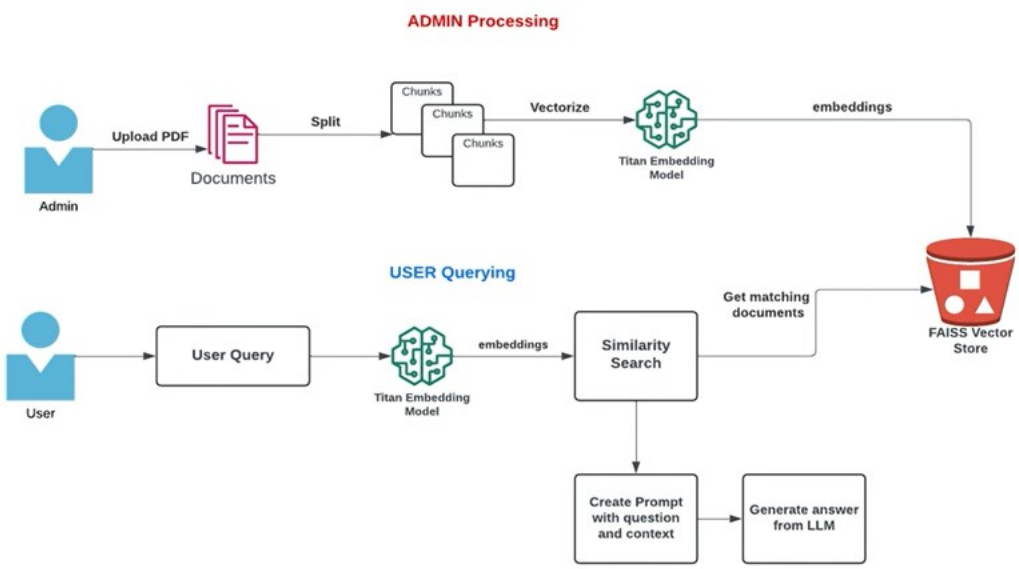
ADMIN Application: - Build Admin Web application where AdminUser can upload the pdf. - The PDF text is split into chunks - Using the Amazon Titan Embedding Model, create the vector representation of the chunks - Using FAISS, save the vector index locally - Upload the index to Amazon S3 bucket (You can use other vector stores like OpenSearch, Pinecone, PgVector etc., but for this demo, I chose cost effective S3) –

USER Application: - Build User Web application where users can query / chat with the pdf. - At the application start, download the index files from S3 to build local FAISS index (vector store) - Langchain's RetrievalQA, does the following: - Convert the User's query to vector embedding using Amazon Titan Embedding Model (Make sure to use the same model that was used for creating the chunk's embedding on the Admin side) - Do similarity search to the FAISS index and retrieve 5 relevant documents pertaining to the user query to build the context - Using Prompt template, provide the question and context to the Large Language Model. We are using Claude model from Anthropic. - Display the LLM's response to the user.

PDF .. Loaded in this folder- https://www.versusarthritis.org/media/22726/what-is-arthritis-information-booklet.pdf



What will we build?



What is RAG

- ✓ RAG stands for Retrieval-Augmented Generation
- ✓ Process of optimizing the output of Large Language Model
- ✓ LLM uses knowledge base outside its training data source
- ✓ To query LLM: Provide question and context in the Prompt
- ✓ Depending on use-case it can be Cost Effective compared to Fine Tuning

Build Admin Site

- ✓ Create S3 Bucket
- ✓ Create requirements.txt
- ✓ Create Python App
- ✓ Create Dockerfile and build docker image
- ✓ Access the application from Browser
- ✓ Upload a PDF
- ✓ Confirm the FAISS vector index files are uploaded to S3

Build User Site

- ✓ Create requirements.txt
- ✓ Create Python App
- ✓ Create Dockerfile and build docker image
- ✓ Access the application from Browser
- ✓ Ask a question
- ✓ Check the response

Go to aws console - s3- create S3 bucket

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US West (Oregon) us-west-2

Bucket type Info

- General purpose
  Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
- Directory - *New*
  Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended)
- ACLs enabled

Bucket name Info

youtube-demo-bedrock-chat-pdf

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

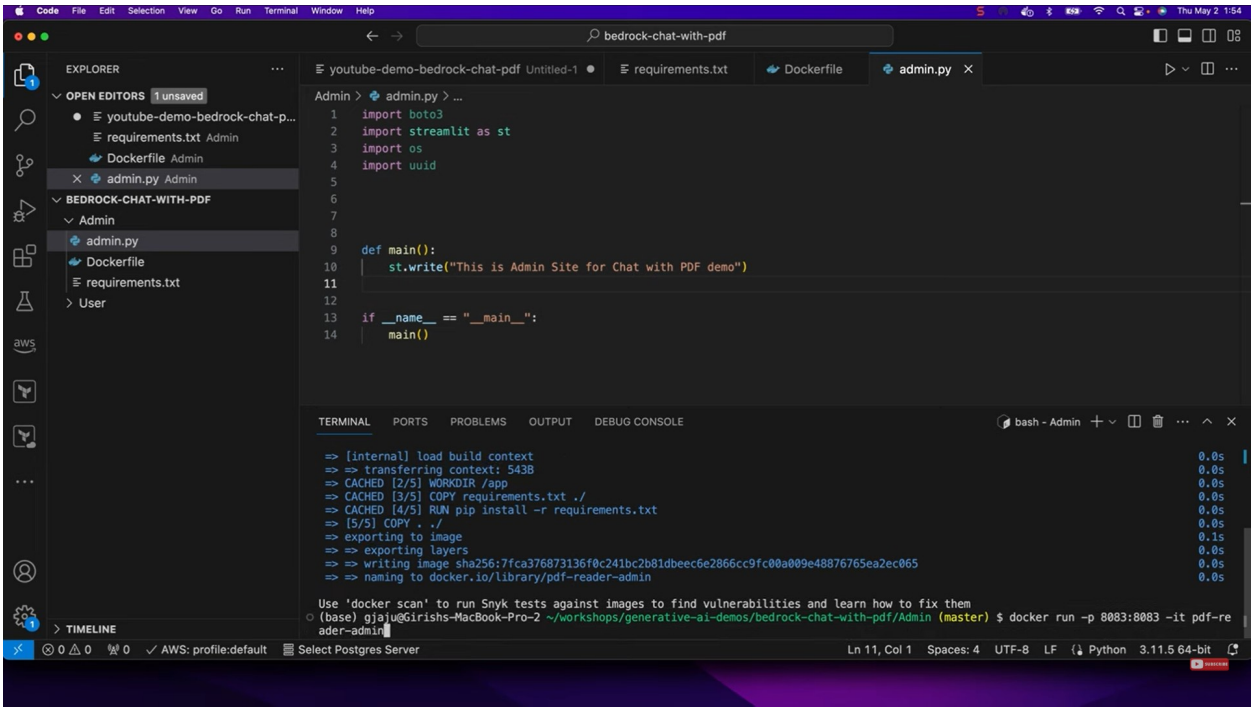
Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

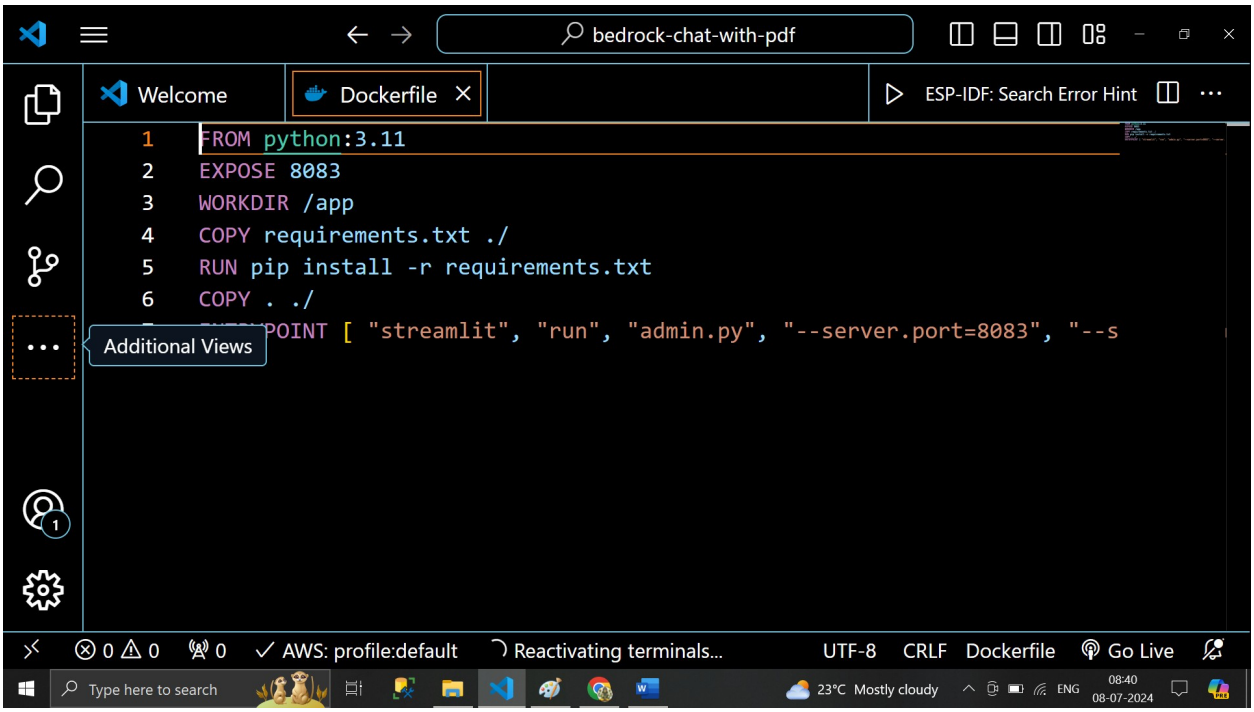
- ACLs disabled (recommended)
  All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- ACLs enabled
  Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

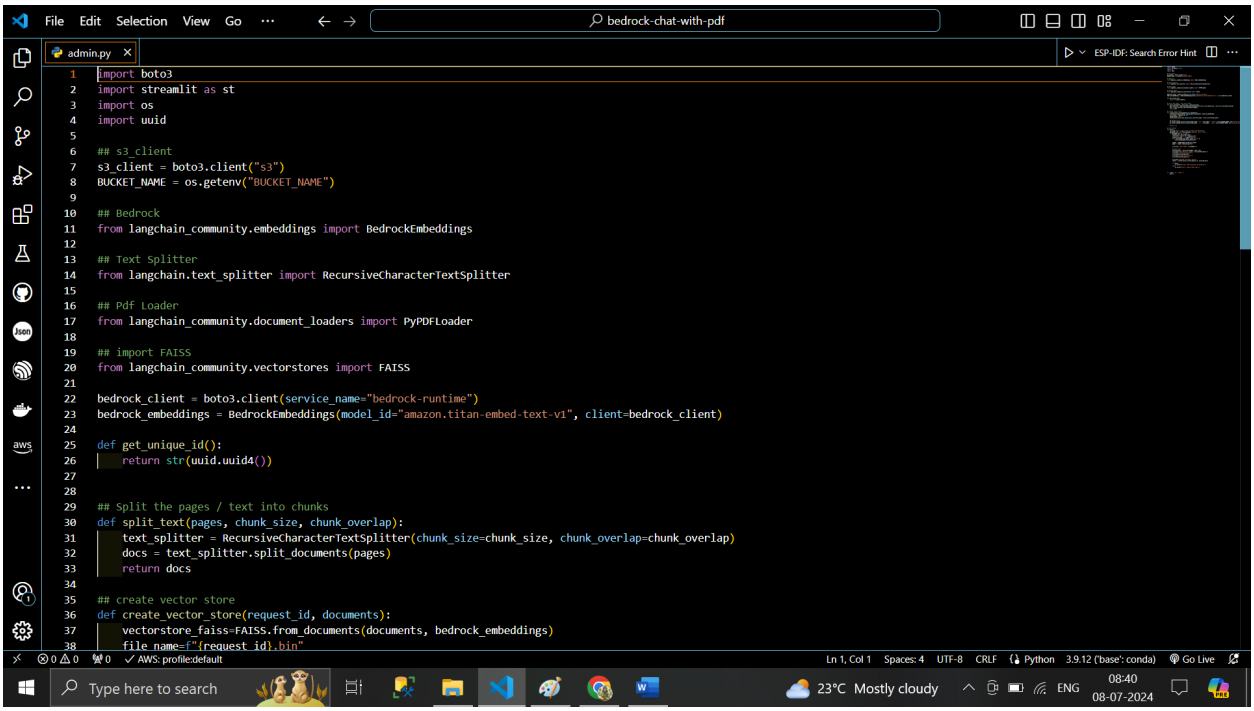
Object Ownership

Bucket owner enforced


bedrock-chat-with-pdf
EXPLORER
OPEN EDITORS 1 unsaved
youtube-demo-bedrock-chat-p...
requirements.txt Admin
Dockerfile Admin
admin.py Admin
BEDROCK-CHAT-WITH-PDF
Admin
admin.py
Dockerfile
requirements.txt
User
TIMELINE
youtube-demo-bedrock-chat-pdf Untitled-1
requirements.txt
Dockerfile
admin.py
import boto3
import streamlit as st
import os
import uuid
def main():
st.write("This is Admin Site for Chat with PDF demo")
if __name__ == "__main__":
main()
TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE
bash - Admin
=> [internal] load build context 0.0s
=> => transferring context: 543B 0.0s
=> CACHED [2/5] WORKDIR /app 0.0s
=> CACHED [3/5] COPY requirements.txt ./ 0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt 0.0s
=> [5/5] COPY . ./ 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:7fca376873136f0c241bc2b81dbeec6e2866cc9fc00a009e48876765ea2ec065 0.0s
=> => naming to docker.io/library/pdf-reader-admin 0.0s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) gjaju@Girishs-MacBook-Pro-2 ~/workshops/generative-ai-demos/bedrock-chat-with-pdf/Admin (master) $ docker run -p 8083:8083 -it pdf-reader-admin
AWS: profile:default
Select Postgres Server
Ln 11, Col 1 Spaces: 4 UTF-8 LF Python 3.11.5 64-bit

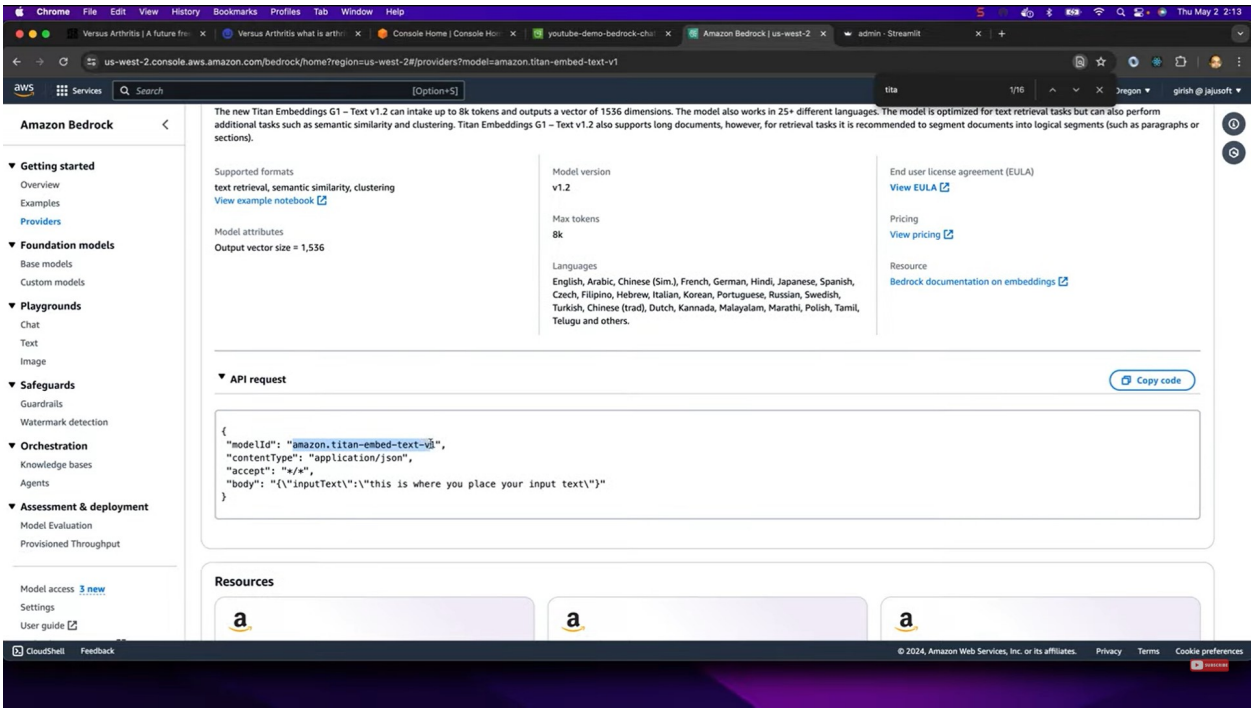
docker file

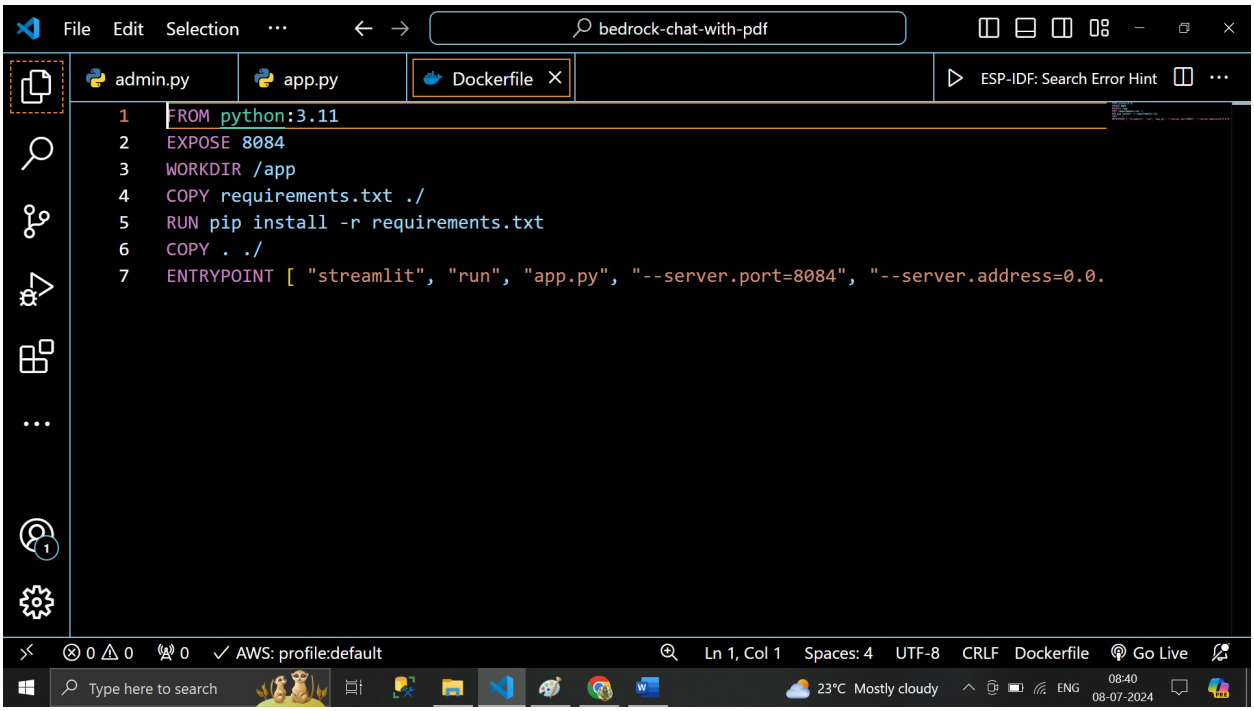

bedrock-chat-with-pdf
Welcome
Dockerfile
ESP-IDF: Search Error Hint
FROM python:3.11
EXPOSE 8083
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . ./
POINT ["streamlit", "run", "admin.py", "--server.port=8083", "--s
Additional Views
AWS: profile:default
Reactivating terminals...
UTF-8 CRLF Dockerfile
Go Live
Type here to search
23°C Mostly cloudy
ENG
08:40
08-07-2024

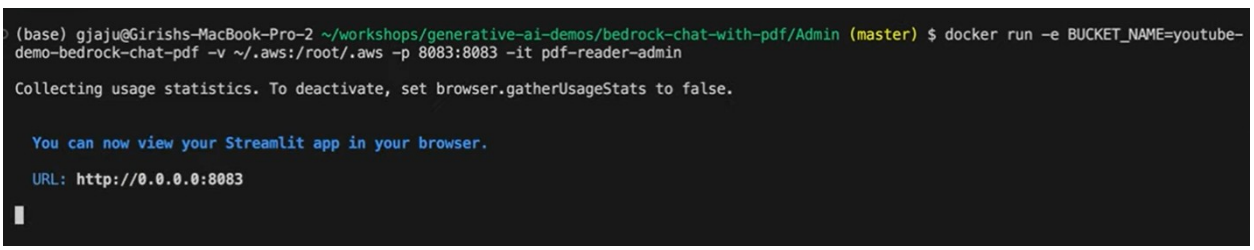


```python
import boto3
import streamlit as st
import os
import uuid

## s3_client
s3_client = boto3.client("s3")
BUCKET_NAME = os.getenv("BUCKET_NAME")

## Bedrock
from langchain_community.embeddings import BedrockEmbeddings

## Text Splitter
from langchain.text_splitter import RecursiveCharacterTextSplitter

## Pdf Loader
from langchain_community.document_loaders import PyPDFLoader

## import FAISS
from langchain_community.vectorstores import FAISS

bedrock_client = boto3.client(service_name="bedrock-runtime")
bedrock_embeddings = BedrockEmbeddings(model_id="amazon.titan-embed-text-v1", client=bedrock_client)

def get_unique_id():
    return str(uuid.uuid4())


## Split the pages / text into chunks
def split_text(pages, chunk_size, chunk_overlap):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=chunk_overlap)
    docs = text_splitter.split_documents(pages)
    return docs

## create vector store
def create_vector_store(request_id, documents):
    vectorstore_faiss=FAISS.from_documents(documents, bedrock_embeddings)
    file_name=f"{request_id}.bin"
```

Admin -side .. AWS Bedrock







User- Data- app.py file

File Edit Selection View Go ··· bedrock-chat-with-pdf

app.py

```python
import boto3
import streamlit as st
import os
import uuid

## s3_client
s3_client = boto3.client("s3")
BUCKET_NAME = os.getenv("BUCKET_NAME")

## Bedrock
from langchain_community.embeddings import BedrockEmbeddings
from langchain.llms.bedrock import Bedrock

## prompt and chain
from langchain.prompts import PromptTemplate
from langchain.chains import RetrievalQA

## Text Splitter
from langchain.text_splitter import RecursiveCharacterTextSplitter

## Pdf Loader
from langchain_community.document_loaders import PyPDFLoader

## import FAISS
from langchain_community.vectorstores import FAISS
```

0 0 0 AWS: profile:default Ln 107, Col 11 Spaces: 4 UTF-8 CRLF Python 3.9.12 ('base': conda) Go Live

Type here to search 23°C Mostly cloudy ENG 08:48 08-07-2024

Code File Edit Selection View Go Run Terminal Window Help Thu May 2 2:26

bedrock-chat-with-pdf

EXPLORER

OPEN EDITORS 1 unsaved
- youtube-demo-bedrock-chat-p...
- requirements.txt Admin
- Dockerfile Admin
- admin.py Admin
- requirements.txt User
- Dockerfile User
- app.py User

BEDROCK-CHAT-WITH-PDF
- Admin
  - admin.py
  - Dockerfile
  - requirements.txt
- User
  - app.py
  - Dockerfile
  - requirements.txt

TIMELINE

youtube-demo-bedrock-chat-pdf Untitled-1 | requirements.txt Admin | Dockerfile Admin | admin.py | requirements.txt

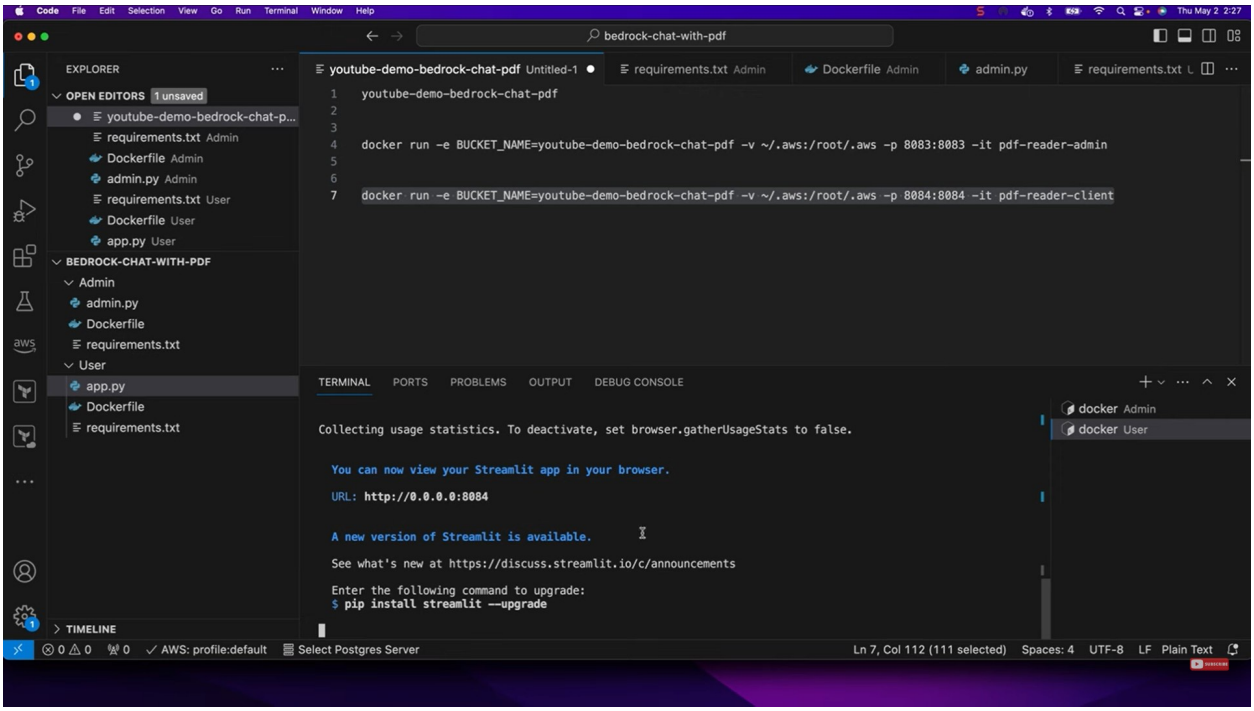
```
youtube-demo-bedrock-chat-pdf


docker run -e BUCKET_NAME=youtube-demo-bedrock-chat-pdf -v ~/.aws:/root/.aws -p 8083:8083 -it pdf-reader-admin


docker run -e BUCKET_NAME=youtube-demo-bedrock-chat-pdf -v ~/.aws:/root/.aws -p 8084:8084 -it pdf-reader-client
```

TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE

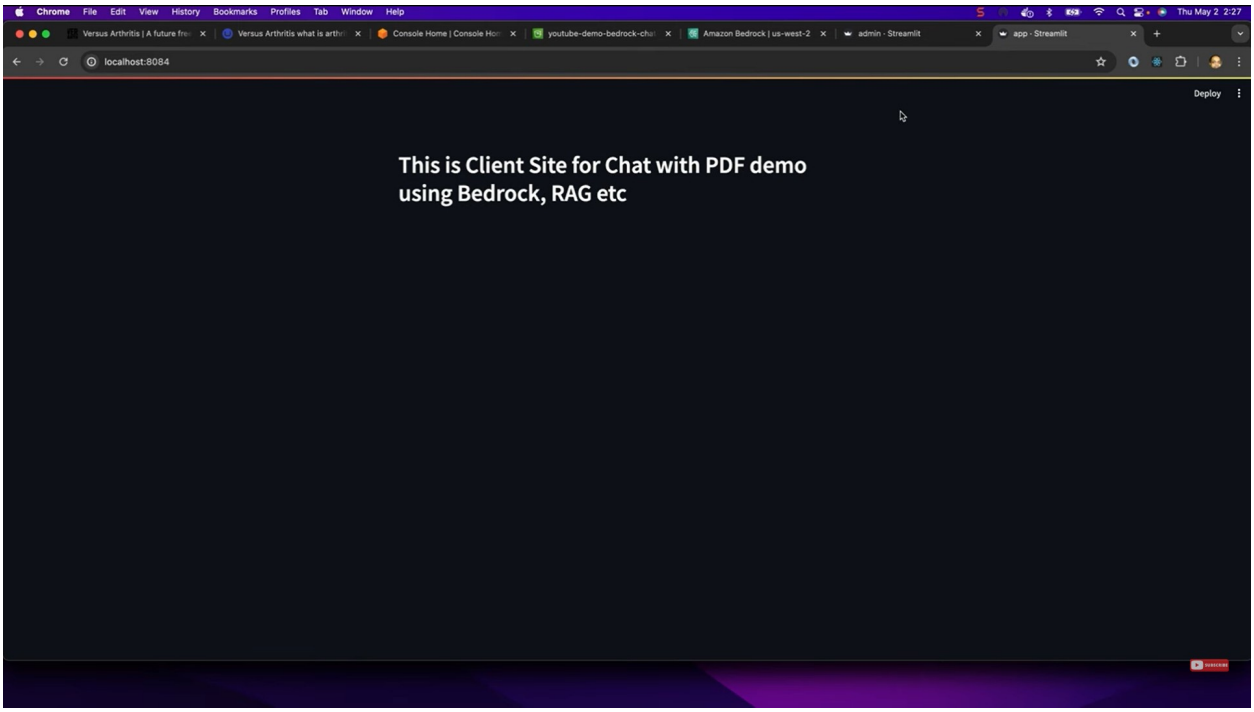
```
 => [internal] load build context                                                    0.0s
 => => transferring context: 1.18kB                                                  0.0s
 => [1/5] FROM docker.io/library/python:3.11@sha256:e453eb723bc8ecac7a797498f9a5915d13e567620d48dcd356  0.0s
 => CACHED [2/5] WORKDIR /app                                                        0.0s
 => CACHED [3/5] COPY requirements.txt ./                                            0.0s
 => CACHED [4/5] RUN pip install -r requirements.txt                                 0.0s
 => [5/5] COPY . ./                                                                  0.0s
 => exporting to image                                                               0.1s
 => => exporting layers                                                              0.0s
 => => writing image sha256:06edd29077c4697bad0dc5dc0757eafe80fe16ab61b03222332c02b36d9cbee7  0.0s
 => => naming to docker.io/library/pdf-read-client                                   0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
(base) gjaju@Girishs-MacBook-Pro-2 ~/workshops/generative-ai-demos/bedrock-chat-with-pdf/User (master) $ dock
er build -t pdf-reader-client .
[+] Building 0.0s (0/1)
```

docker Admin
docker User

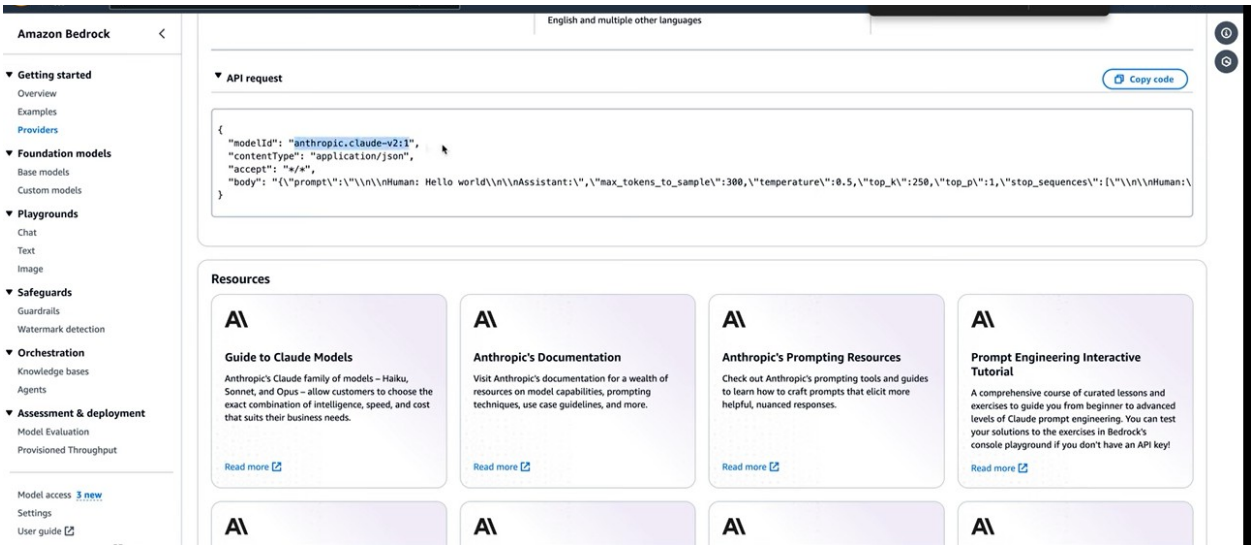
0 0 0 AWS: profile:default Select Postgres Server Ln 7, Col 112 (111 selected) Spaces: 4 UTF-8 LF Plain Text

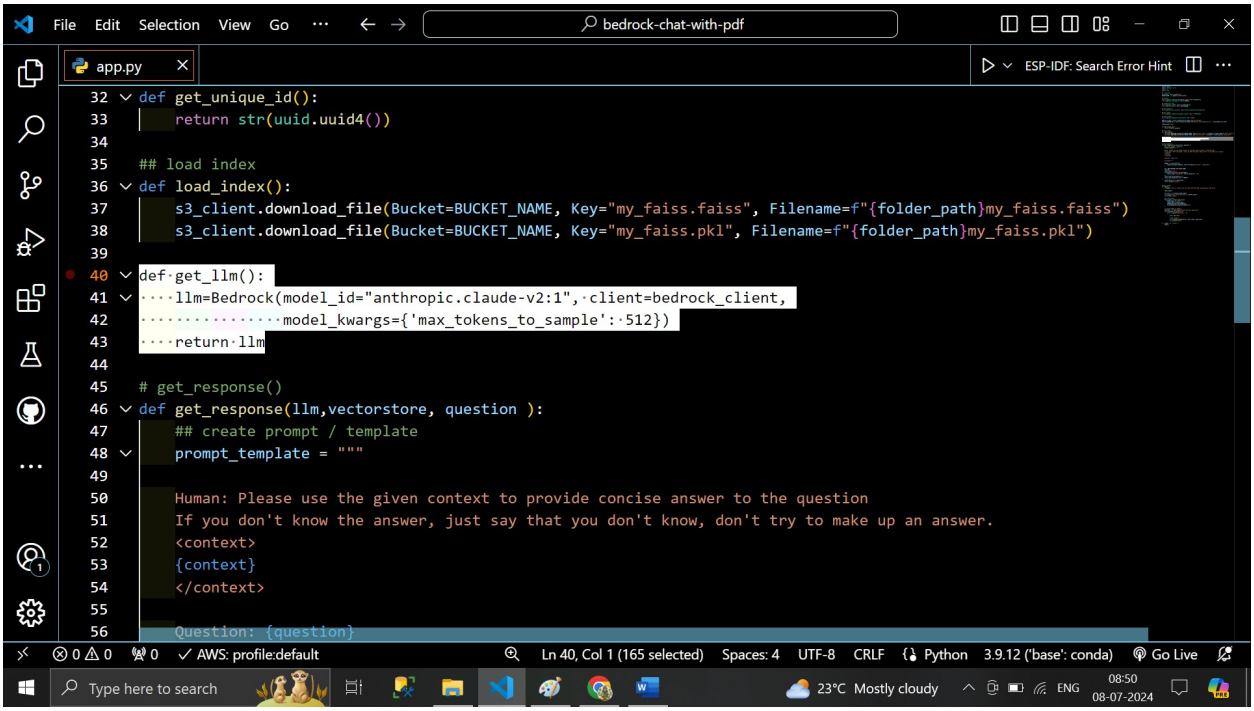


Loading app

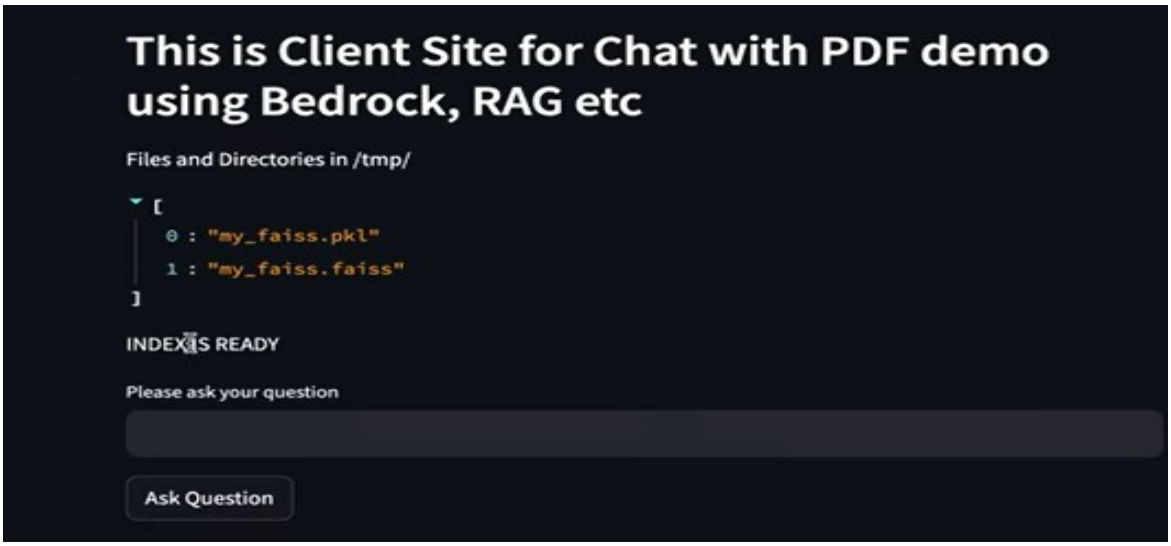


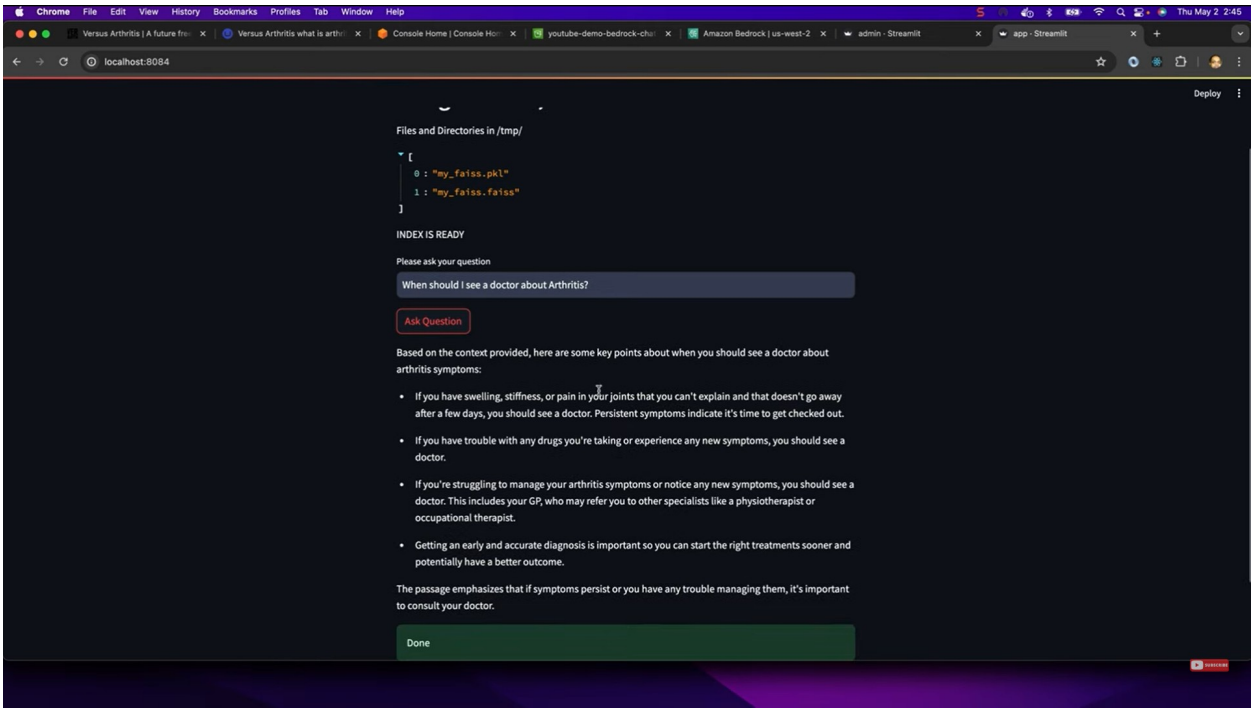
Go to bedrock base Models





Output .. index it ready





Ask question Outside PDF File

INDEX IS READY

Please ask your question

Who won the cricket world cup in 2023?

Ask Question

Unfortunately, I do not have enough context in the given information to determine who won the cricket world cup in 2023. The passage discusses arthritis and provides information about Versus Arthritis, an arthritis charity. It does not mention anything about cricket or the 2023 world cup. Since there is no relevant information provided to answer the question, I don't know who won the cricket world cup in 2023.

Done