Integrating the OpenAI API in Your ColdFusion Applications

Brian Klaas
@brian_klaas
brianklaas.net


YOU GET A CHATBOT!
YOU GET A CHATBOT!
YOU GET A CHATBOT!

https://venturebeat.com/ai/a-chevy-for-1-car-dealer-chatbots-show-perils-of-ai-for-customer-service/

https://x.com/chrisjbakke/status/1736533308849443121?s=61

Now?Soon?Never?

CFNext

Extract
Summarize
Auto-tag

Generate
Edit
OCR

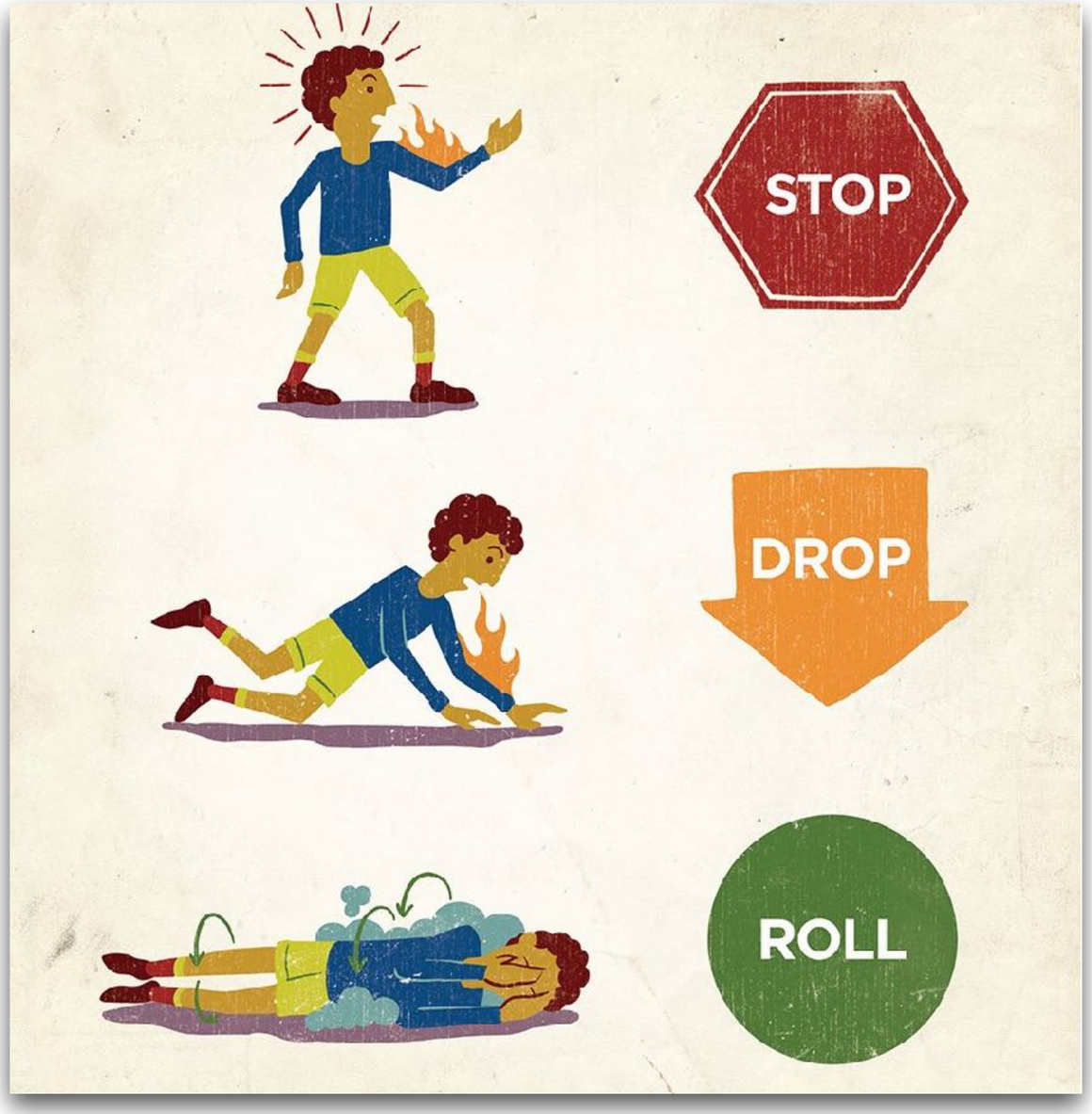
Embeddings
Analysis
Fine-tuning

What about solving problems now?

ANTHROP\C

HELLO GPTS!

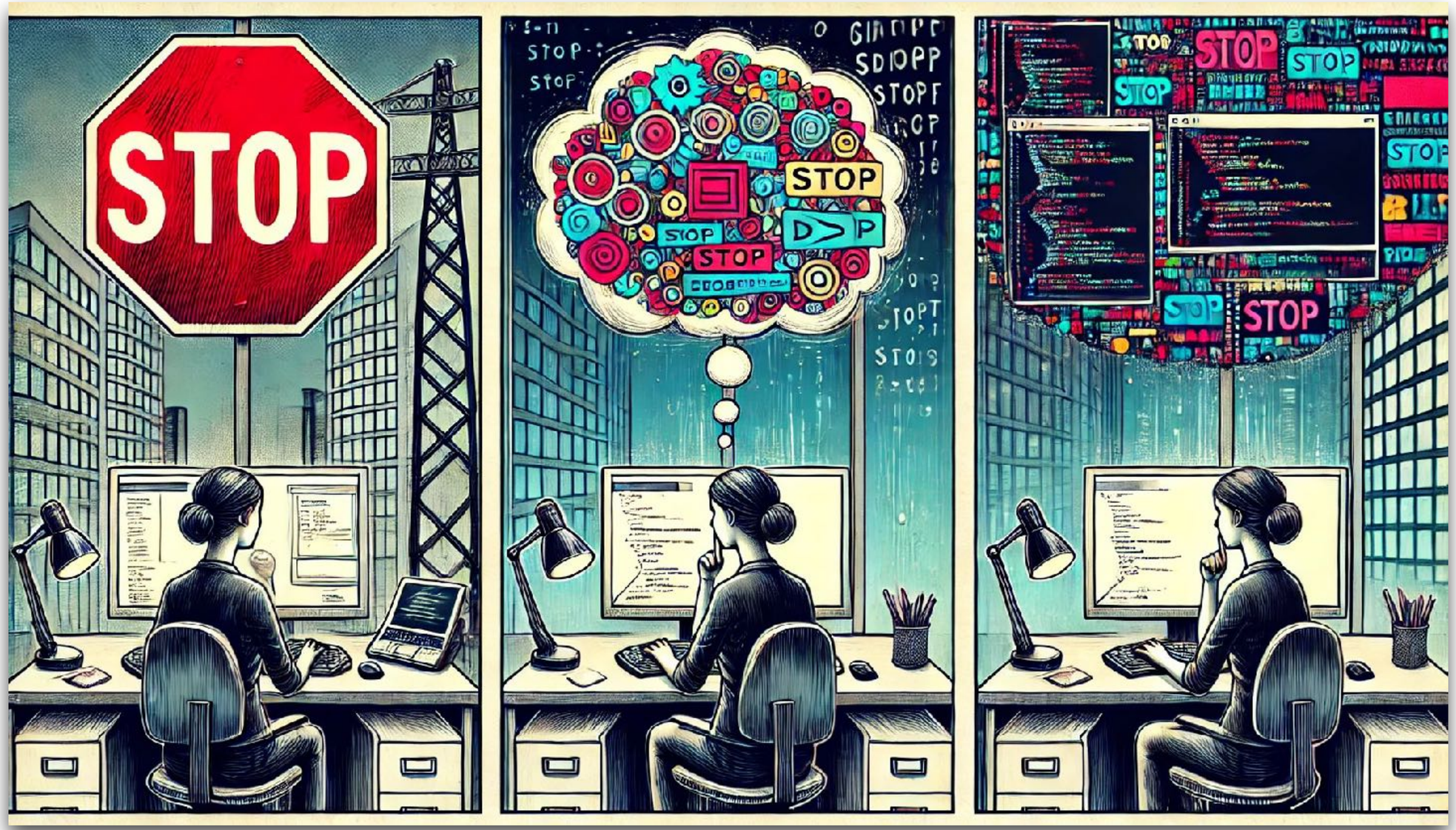
GPT-4o Prompt: “Create an image of a large pile of cash being set on fire, and the software developer setting the cash on fire is very, very, very happy about it.”


STOP
DROP
ROLL

GPT-4o Prompt: “Create an image of three business people abusing a sweet, innocent generative AI model.”

Don't allow free-range access to large language models.

Use an API instead.



HELLO GPTS!

GPT-4o Prompt: “Create a triptych image. Panel one is a big stop sign. Panel two is of a software developer thinking with a big thought bubble above her head. Panel three is the same developer programming at their desk.”

The OpenAI API

platform.openai.com

Completions

Chat Completions

```
{
   "model": "gpt-4o-mini",
   "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant."
    },
    {
      "role": "user",
      "content": "Who won the world series in 2020?"
    },
    {
      "role": "assistant",
      "content": "The Los Angeles Dodgers won the World Series in 2020."
    }
}
```

Legacy Completions	Modern Completions
Access to GPT 3.5 and lower only	Access to latest models
Can use file attachments	Cannot use file attachments

Assistants

Augments a basic chat completion with:

File Search (RAG)

Code Interpreter

Function Calling

Allow for more complex processes

Have conversations, just like in ChatGPT

Run code, generate files

Persistent threads of conversation

Auto-creation of embeddings = better results

Want to keep your data private?

You’re going to need to use the API.

(or at least the Playground)

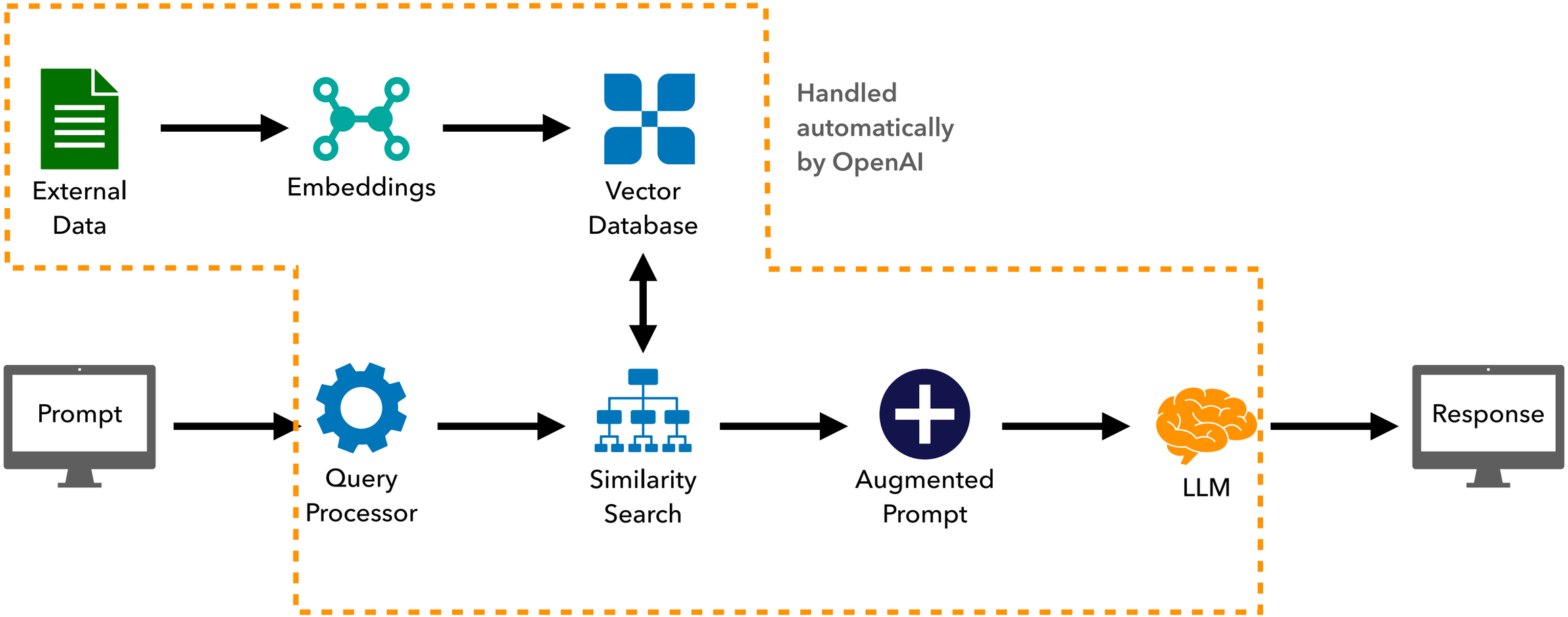
Assistants are the future of OpenAI's API offerings

Retrieval Augmented Generation

Got documents or data?

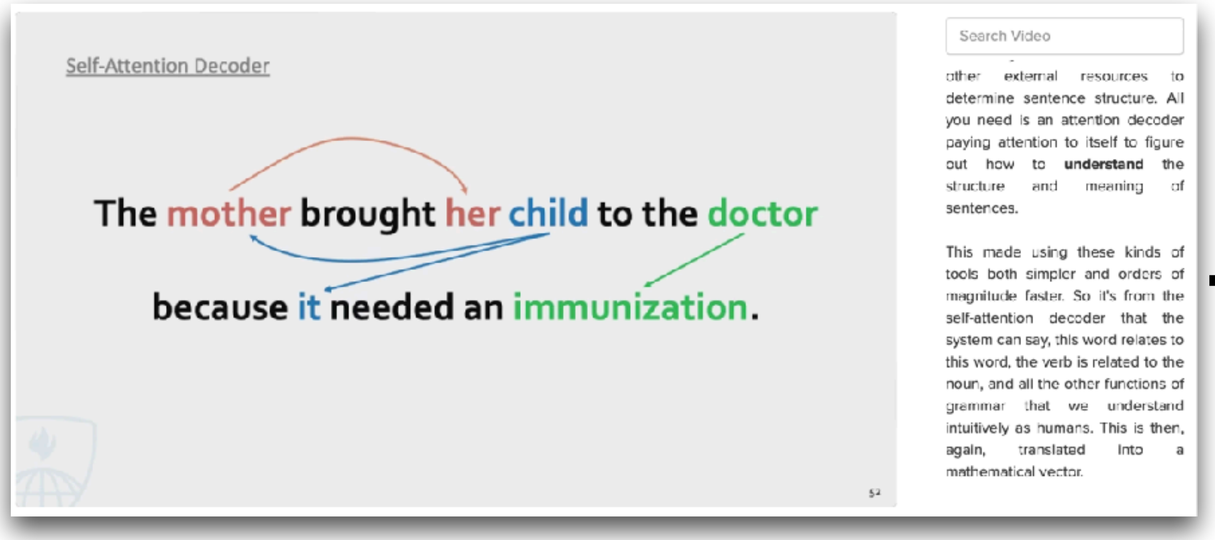
You’re going to need some RAG.

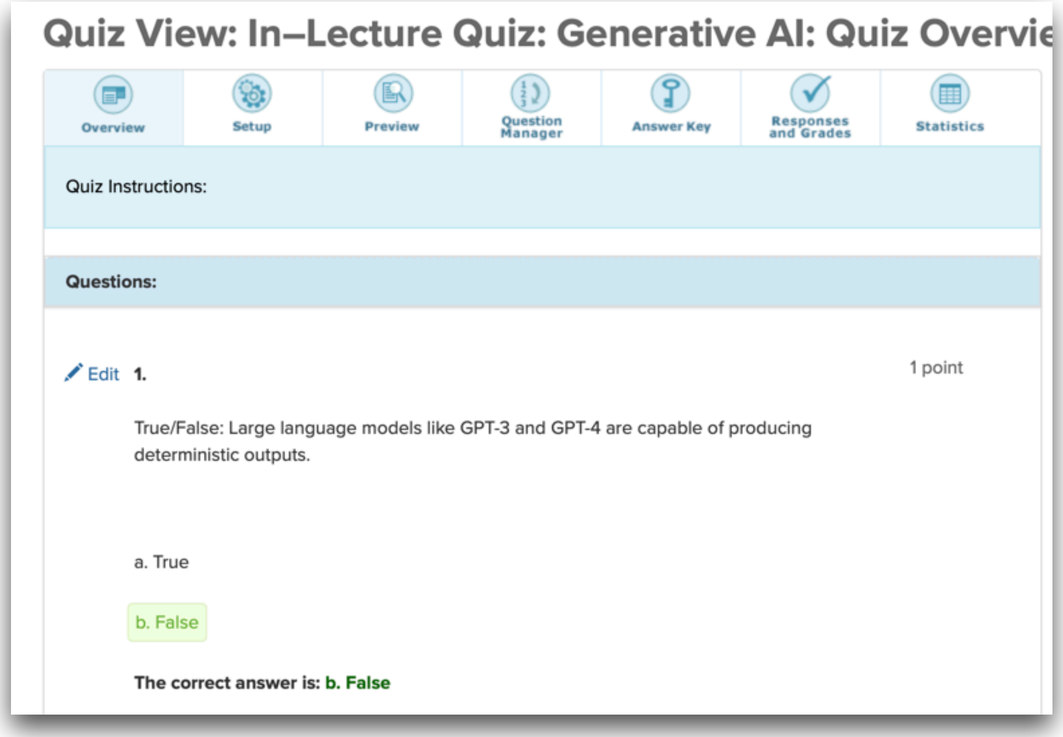
How does RAG work?


External Data
Embeddings
Vector Database
Handled automatically by OpenAI
Prompt
Query Processor
Similarity Search
Augmented Prompt
LLM
Response

No support for parsing images in documents.

The API Workflow


Self-Attention Decoder
The mother brought her child to the doctor
because it needed an immunization.
Search Video
other external resources to determine sentence structure. All you need is an attention decoder paying attention to itself to figure out how to understand the structure and meaning of sentences.
This made using these kinds of tools both simpler and orders of magnitude faster. So it's from the self-attention decoder that the system can say, this word relates to this word, the verb is related to the noun, and all the other functions of grammar that we understand intuitively as humans. This is then, again, translated into a mathematical vector.


Quiz View: In–Lecture Quiz: Generative AI: Quiz Overvie
Overview
Setup
Preview
Question Manager
Answer Key
Responses and Grades
Statistics
Quiz Instructions:
Questions:
Edit 1.
1 point
True/False: Large language models like GPT-3 and GPT-4 are capable of producing deterministic outputs.
a. True
b. False
The correct answer is: b. False

Steps in the Assistants Workflow

1. Upload files
2. Create vector store
3. Add files to vector store
4. Create a thread
5. Add messages to thread
6. Create a run
7. Get the generated content from the run

Things will go wrong

Design for resilience.

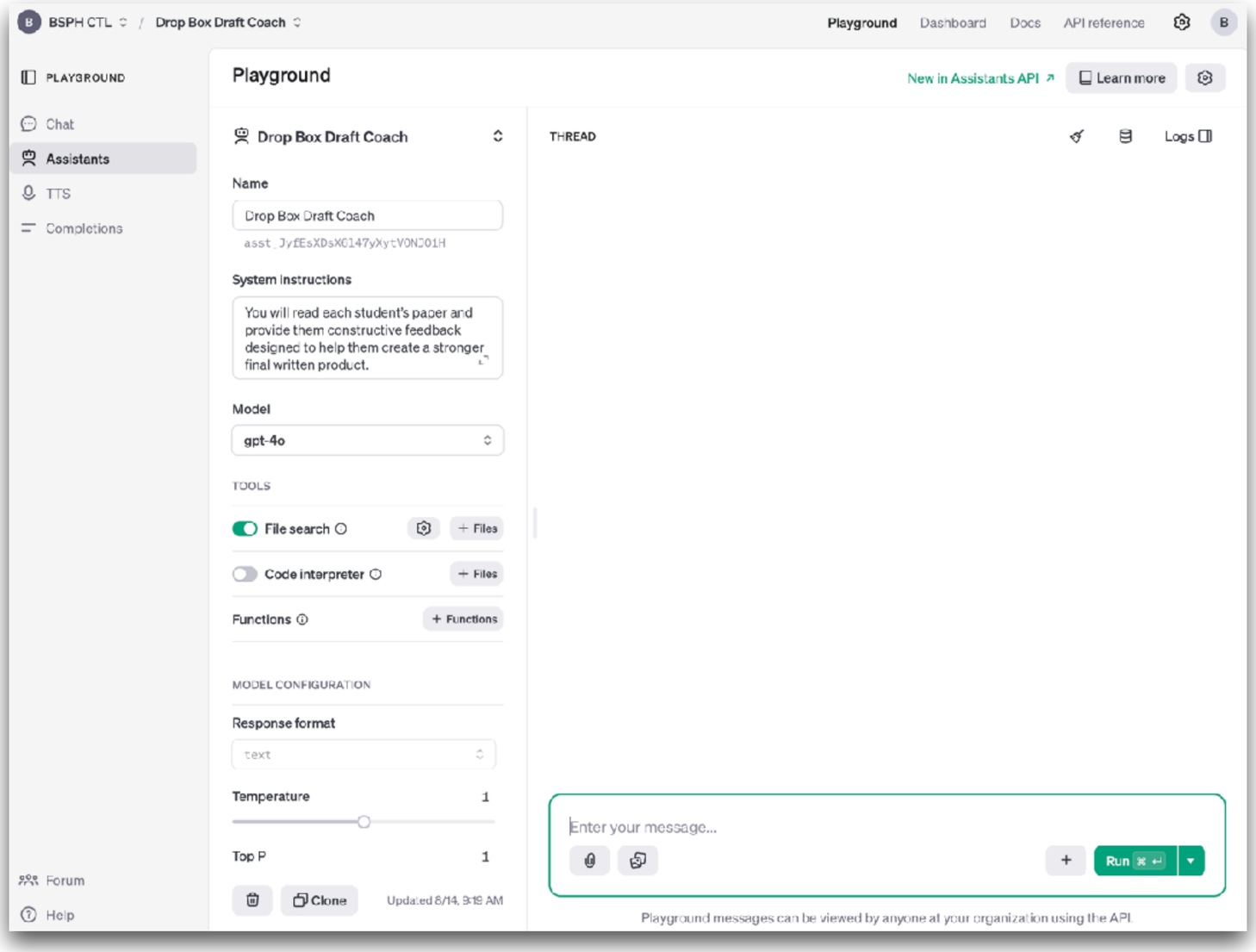
(Hint: )

Three prerequisites

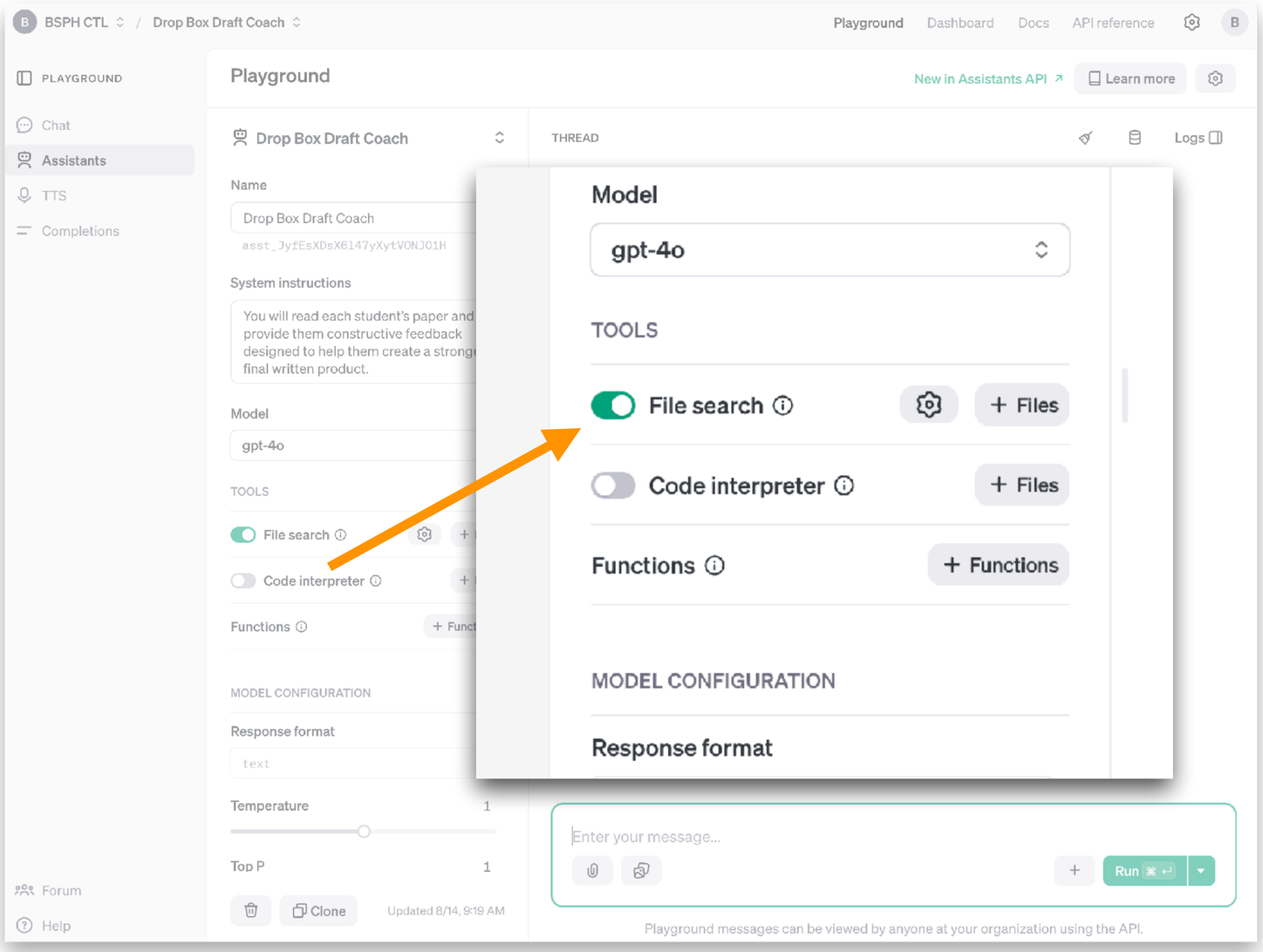
You need an assistant.

platform.openai.com

Three Prerequisites



Three Prerequisites



Test your prompts in the Playground.

Lots and lots and **LOTS** of tests.

You need a service account.

(One service account per project is safer.)

You need headers.

```
cfhttpparam(name="Authorization", type="header", value="Bearer YOUR TOKEN");
cfhttpparam(name="OpenAI-Beta", type="header", value="assistants=v2");
```

Current API is beta 2

Steps in the Assistants Workflow

1. Upload files
2. Create vector store
3. Add files to vector store
4. Create a thread
5. Add messages to thread
6. Create a run
7. Get the generated content from the run

1 Upload files

POST https://api.openai.com/v1/files


node
JS


Cf

?

openAI-coldfusion-utils

https://github.com/brianklaas/openAI-coldfusion-utils

1 Upload files

```
cfhttp(method=“POST”, charset="utf-8", url=“https://api.openai.com/v1/files", result=“result")
   cfhttpparam(name="Authorization", type="header", value=“Bearer YOUR TOKEN”);
   cfhttpparam(name="OpenAI-Beta", type="header", value="assistants=v2");
   cfhttpparam(name="purpose", type="formfield", value="assistants");
   cfhttpparam(type="file", file=“PATH TO FILE”, name=“file");
}
```

1 Upload files

openAI-coldfusion-utils

uploadFile(string pathToFile)

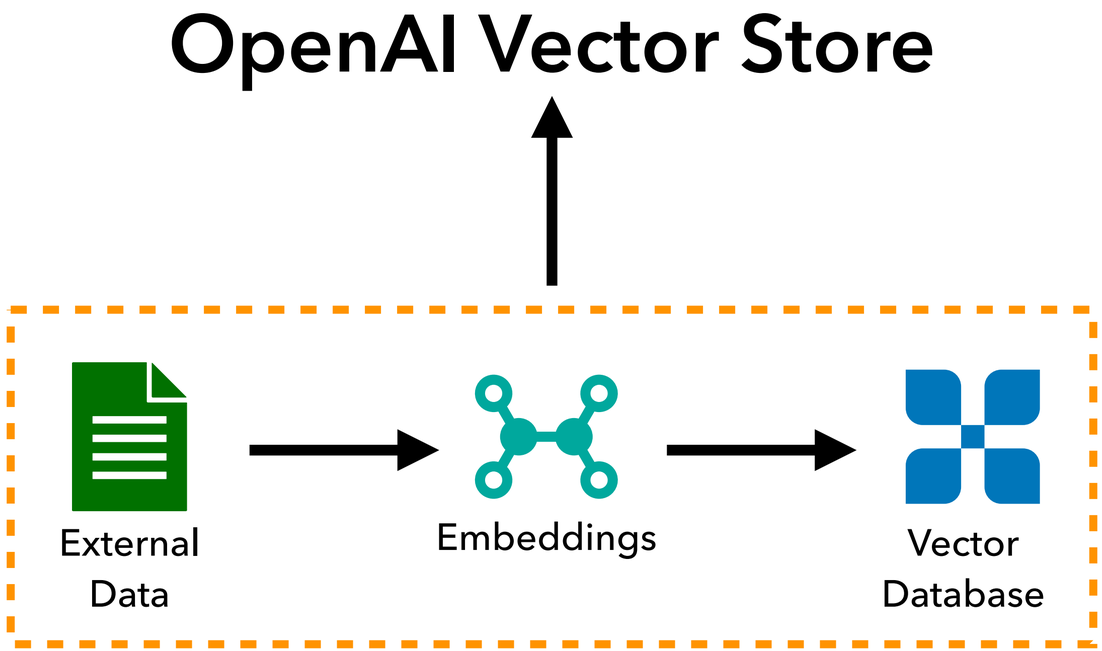
1 Upload files


```
{
  "id": "file-abc123",
  "object": "file",
  "bytes": 120000,
  "created_at": 1677610602,
  "filename": “myDocument.pdf"
}
```

Make sure you stay within the context window

2 Create vector store

POST https://api.openai.com/v1/vector_stores


OpenAI Vector Store
External Data
Embeddings
Vector Database

2 Create vector store

```
cfhttp(method=“POST”, charset="utf-8", url=“https://api.openai.com/v1/vector_store”, result=“result")
    cfhttpparam(name="Authorization", type="header", value=“Bearer YOUR TOKEN”);
    cfhttpparam(name="OpenAI-Beta", type="header", value="assistants=v2");
    cfhttpparam(name="content-type", type="header", value="application/json");
    cfhttpparam(type="body", value=“#requestBodyAsJSON#");
}
```

2 Create vector store

openAI-coldfusion-utils

createVectorStore()

2 Create vector store

Set expires_after **value**

2 Create vector store


```
{
  "id": "vs_abc123",
  "object": "vector_store",
  "created_at": 1699061776,
  "name": "Support FAQ",
  "bytes": 139920,
  "file_counts": {
    "in_progress": 0,
    "completed": 3,
    "failed": 0,
    "cancelled": 0,
    "total": 3
  }
}
```

3 Add files to vector store

POST https://api.openai.com/v1/{vector_store_id}/files

3 Add files to vector store

openAI-coldfusion-utils

```
for (fileID in filesArray) {
  addFileToVectorStore(fileID, vectorStoreID);
}
```

3 Add files to vector store


```
{
  "id": "file-abc123",
  "object": "vector_store.file",
  "created_at": 1699061776,
  "usage_bytes": 1234,
  "vector_store_id": "vs_abcd",
  "status": "completed",
  "last_error": null
}
```

How do you know if it’s done?

openAI-coldfusion-utils

hasFileBeenProcessed(fileID, vectorStoreID)

GET https://api.openai.com/v1/vector_stores/{vector_store_id}/files/{file_id}

while() loop

Database rows + scheduled task

Strategies for Waiting

SQS + CF = ❤️

```
sqsService=getCloudService(credentials,configuration);

sqsService.sendMessage(string sqsQueueURL, struct message);

sqsService.receiveMessage(string sqsQueueURL);

sqsService.deleteMessage(string sqsQueueURL, string receiptHandle);
```

SQS + CF = ❤️

SQS Provides:

Message ordering
Fault-tolerant resilience
Easy retries in case of API failure

SQS + CF = ❤️

You'll need SQS queues for:

Checking file processing completion
Checking run completion

SQS + CF = ❤️

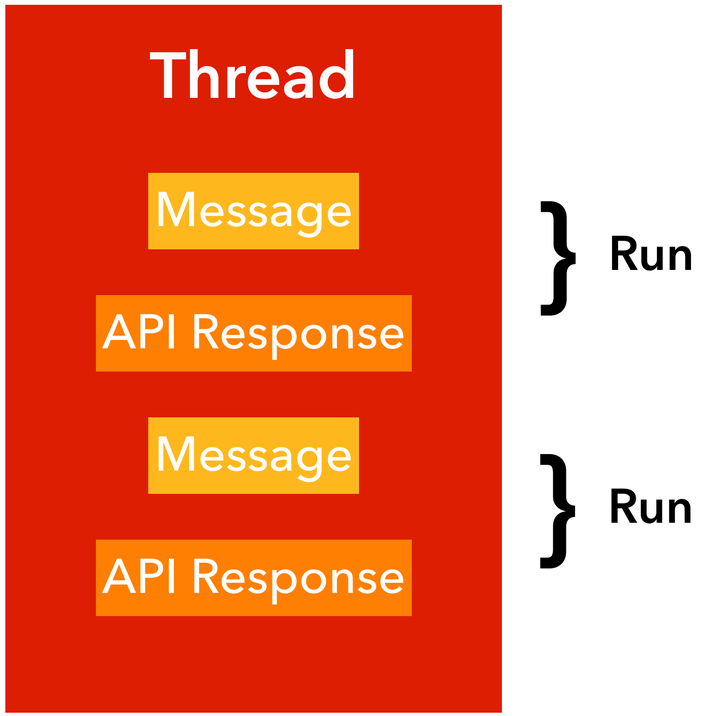
You'll need SQS queues for:

Uploading files
Checking file processing completion
Checking run completion
File and vector store cleanup

4 Create a thread

POST https://api.openai.com/v1/threads

Conversing with GPT4o



4 Create a thread

openAI-coldfusion-utils

createThread(vectorStoreID)

4 Create a thread


```
{
  "id": "thread_abc123",
  "object": "thread",
  "created_at": 1699012949,
  "metadata": {},
  "tool_resources": {
    [ "file_search" ]
  }
}
```

5 Add messages to thread

POST https://api.openai.com/v1/threads/{thread_id}/messages

5 Add messages to thread

5 Add messages to thread

openAI-coldfusion-utils

createMessageInThread(threadID, messageBody)

Assistant Instructions + Message Body = “Prompt”

Main prompt → instructions

Custom content/context → message body

Conversing with GPT4o

Assistant Instructions + Message Body + Files in Vector Store → 

LLM

5 Add messages to thread

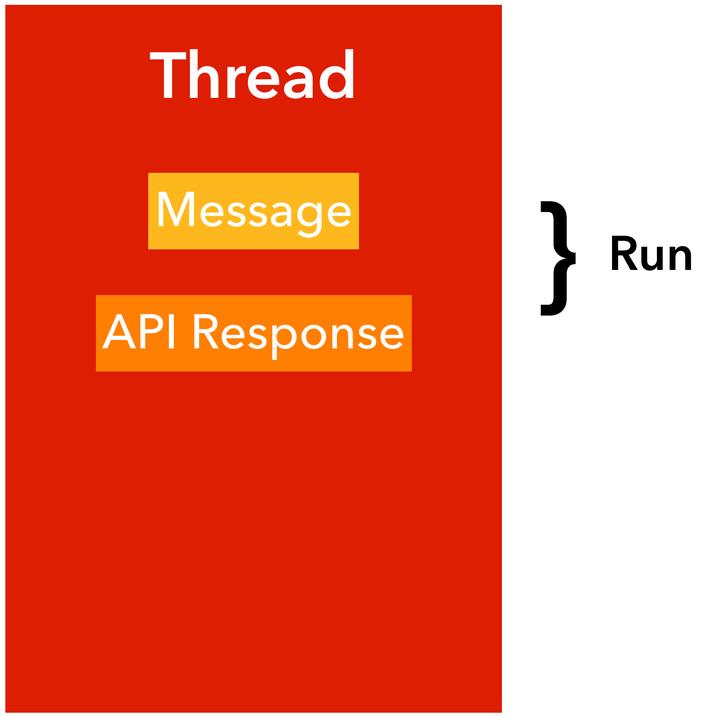
openAI-coldfusion-utils

createMessageInThread(threadID, messageBody)

6 Create a run

POST https://api.openai.com/v1/threads/{thread_id}/runs

6 Create a run



6 Create a run

openAI-coldfusion-utils

createRun(threadID, assistantID)

6 Create a run


```
{
  "id": "run_abc123",
  "created_at": 1699063290,
  "assistant_id": "asst_abc123",
  "thread_id": "thread_abc123",
  "status": "queued",
  "started_at": 1699063290,
  "expires_at": null,
  "cancelled_at": null,
  "failed_at": null,
  "completed_at": null,
  "last_error": null,
  "tools": [
    {
      "type": “file_search”
    }
  ]
}
```

6 Create a run

Runs can take a while

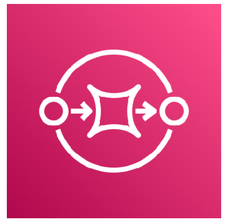
6* Check for run completion

openAI-coldfusion-utils

isRunComplete(threadID, runID)

6* Check for run completion

Failed runs?



7 Get the generated content from the run

GET https://api.openai.com/v1/threads/{thread_id}/runs/{runID}

7 Get the generated content from the run

In the cfhttp **call response:**

response.callResult.data[1].content[1].text.value

7 Get the generated content from the run

Check that the response is useful!

7 Get the generated content from the run

Output defaults to Markdown

https://www.raymondcamden.com/2024/09/16/parsing-markdown-in-coldfusion

Steps in the Assistants Workflow

1. Upload files
2. Create vector store
3. Add files to vector store
4. Create a thread
5. Add messages to thread
6. Create a run
7. Get the generated content from the run

Don’t forget to clean up the files!

Painful Lessons

Refine your prompt in the Playground before building anything.

Don't ask for references.

Pass in dynamic instructions in a thread/message.

Really structured output?
Use Structured Outputs (JSON Schema).

Requests fail.
Build for retries.

Multi-step prompts?
Write options into your app.

Friends don’t let friends build open-ended chatbots.

Completions may be just fine!

Go Do!

brian.klaas@gmail.com
brianklaas.net
CFML Slack

https://github.com/brianklaas/openAI-coldfusion-utils