Main menu

Deploy a RAG application with vector search in Firestore

Course · 2 hours 30 minutes · < 1% complete

Course overview

Deploy a RAG application with vector search in Firestore | Challenge Lab

Deploy a RAG application with vector search in Firestore: Challenge Lab

Your Next Steps

Course Badge

Course Survey

Course > Deploy a RAG application with vector search in Firestore >

Quick tip: Review the prerequisites before you run the lab

End Lab 01:01:41

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Open Google Cloud Console

Username
student-00-4eeb390ef734

Password
nxKF0fprC7tF

GCP Project ID
qwiklabs-gcp-01-4db5c0a

Deploy a RAG application with vector search in Firestore: Challenge Lab

Lab · 2 hours 30 minutes · No cost · Intermediate

★★★★☆

This lab may incorporate AI tools to support your learning.

GENAI069

Scoring: You must score **80% or higher** to pass this Challenge lab.

Challenge Lab Overview

This lab will challenge you to perform actions and automation across products. Instead of following step-by-step instructions, you are given a common business scenario and a set of tasks - you figure out how to complete them on your own! An automated scoring system (shown on this page) provides feedback on whether you have completed your tasks correctly.

When you take a Challenge Lab, you will not be taught Google Cloud concepts. You will need to use your skills to assess how to build the solution to the challenge presented. This lab is only recommended for students who have those skills. Are you up for the challenge?

Objective

This lab tests your ability to develop a real-world Generative AI Q&A solution using a RAG framework. You will use Firestore as a vector database and deploy a Flask app as a user interface to query a food safety knowledge base.

This lab uses the following technologies and Google Cloud services:

- Vertex AI
- Vertex AI Colab Enterprise
- Vertex AI Embeddings API
- Gemini 2.0 Flash
- Cloud Firestore

In this challenge lab, you will demonstrate your ability to load a text document and split it into chunks, generate embeddings for each chunk, store the text chunks and their embeddings, conduct vector search to return similar documents to a query document, complete a RAG framework by having Gemini generate a response based on a context of similar documents to a query.

Related learning materials

If you are looking for resources that will help you develop the skills required to complete

this lab successfully, consider reviewing relevant sections from the following:

Category	Resources
Learning Path	Build with Vertex AI: Building RAG Applications with Vertex AI

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

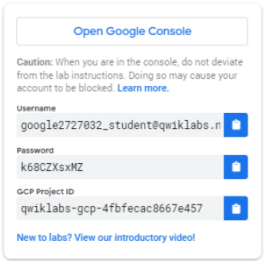
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

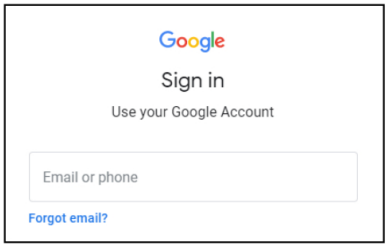
Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.

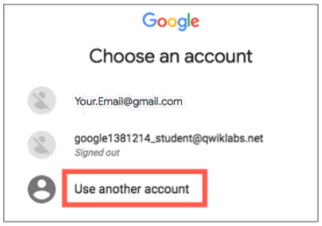


2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.



3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Important: You must use the credentials from the Connection Details panel. Do not use

your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.

Challenge Scenario

Here is a company overview as provided on Cymbal Shops's website.

Cymbal Shops is an American retail chain headquartered in Minneapolis that sells homewares, electronics, and clothing.

Founded in 1974, Cymbal Shops started out as Cymbal Air, selling AC systems manufactured by Cymbal Group in Minnesota and neighboring states. The company quickly expanded into domestic merchandise in order to satisfy the need for quality products at an affordable price in the midst of a recession.

Cymbal Shops' broad product assortment was once a benefit - capturing planned purchases and last minute splurges. In recent years, however, the company has struggled to adapt to the acceleration in e-commerce. Digitally native companies are on the rise and Cymbal Shops must implement significant changes in order to keep pace and maintain relevance. Today, Cymbal Shops operates 714 stores across North America and reported $15 billion in revenue in 2019. They currently employ 80,400 employees across the United States and Canada.

Cymbal Shops is a digitally transforming legacy retailer.

It is inspired by clients like: Bed Bath & Beyond, Best Buy, Home Depot, and Nordstrom.

Your Challenge

You work for Cymbal Shops, a chain offering prepared meals to-go in busy downtown areas.

The company's employees in the New York area needs to meet the New York City Department of Health and Mental Hygiene's food safety guidelines as provided in this Food Protection Training Manual.

To make it easier for them, your manager has asked you to prepare an application where employees can ask questions and a bot will return the answers based only on what is found in this document.

Task 1. Create a Colab Enterprise Notebook

In this section, you will set up a **Colab Enterprise** notebook environment in the **Google Cloud Console**.

1. In the **Google Cloud Console**, navigate to **Vertex AI > Colab Enterprise**.

2. When prompted to enable APIs, click **ENABLE**.

3. Within the Colab Enterprise panel in the console, click on **Create Notebook**. Rename the notebook to `cymbal_vector_database.ipynb`.

4. Paste the following code into the top cell of the notebook and run the cell.

```
!pip install --quiet --upgrade google-cloud-logging
google_cloud_firestore google_cloud_aiplatform langchain
langchain-google-vertexai langchain_community
langchain_experimental pymupdf
```

5. After the cell completes running, indicated by a checkmark to the left of the cell, the packages should be installed. To use them, restart the runtime.

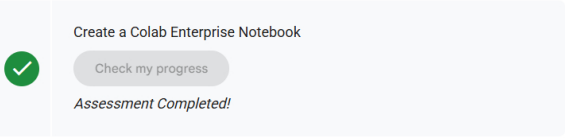
6. Import the following packages by running the following command:

```
import vertexai
import logging
import google.cloud.logging
from vertexai.language_models import TextEmbeddingModel
from vertexai.generative_models import GenerativeModel

import pickle
from IPython.display import display, Markdown

from langchain_google_vertexai import VertexAIEmbeddings
from langchain_community.document_loaders import PyMuPDFLoader
from langchain_experimental.text_splitter import SemanticChunker

from google.cloud import firestore
from google.cloud.firestore_v1.vector import Vector
from google.cloud.firestore_v1.base_vector_query import
DistanceMeasure
```

7. Next, initialize Vertex AI with your project-id `qwiklabs-gcp-01-4db5c0a69bf4` and a location of `us-central1`.

8. Populate a variable named `embedding_model` with an instance of the `langchain_google_vertexai` class VertexAIEmbeddings. Pass it a parameter `model_name` set to the text embedding model version of `text-embedding-005`. You will use this LangChain class for your embedding model so that you can use a LangChain semantic chunker to chunk your dataset.

Click **Check my progress** to verify the objective.



Task 2. Download, process and chunk data semantically

In this section, you will prepare the `NYC Food Safety Manual` for **Retrieval-Augmented Generation (RAG)**. Clean the PDF content and split it into meaningful chunks based on `semantic similarity` using sentence embeddings and generate numerical representations (embeddings) for each identified text chunk.

1. Download the **New York City Department of Health and Mental Hygiene's Food Protection Training Manual**. This document will serve as your `Retrieval-Augmented Generation` source content.

```
!gcloud storage cp gs://partner-genai-
bucket/genai069/nyc_food_safety_manual.pdf .
```

2. Use the LangChain class PyMuPDFLoader to load the contents of the PDF to a variable named `data`.

3. The following function is provided to do some basic cleaning on artifacts found in this particular document. Create a variable called `cleaned_pages` that is a list of strings, with each string being a page of content cleaned by this function.

```
def clean_page(page):
```

```
  return page.page_content.replace("-\n","")\
                          .replace("\n"," ")\
                          .replace("\x02","")\
                          .replace("\x03","")\
                          .replace("fo d P R O T E C T I O N  T R
A I N I N G  M A N U A L","")\
                          .replace("N E W  Y O R K  C I T Y  D E
P A R T M E N T  O F  H E A L T H  &  M E N T A L  H Y G I E N
E","")
```

4. Use LangChain's SemanticChunker with the `embedding_model` you created earlier to **split the first five pages** of `cleaned_pages` into text chunks. The `SemanticChunker` determines when to start a new chunk when it encounters a larger distance between sentence embeddings. Save the strings of page content from the resulting documents into a list of strings called `chunked_content`. Take a look at a few of the chunks to get familiar with the content.

5. Use the `embedding_model` to generate embeddings of the text chunks, saving them to a list called `chunked_embeddings`. To do so, pass your list of chunks to the VertexAIEmbeddings class's `embed_documents()` method.

6. You should have successfully chunked & embedded a short section of the document. To get the chunks & corresponding embeddings for the full document, run the following code:

```
!gcloud storage cp gs://partner-genai-
bucket/genai069/chunked_content.pkl .
!gcloud storage cp gs://partner-genai-
bucket/genai069/chunked_embeddings.pkl .

chunked_content = pickle.load(open("chunked_content.pkl", "rb"))
chunked_embeddings = pickle.load(open("chunked_embeddings.pkl",
"rb"))

# Do not delete this logging statement.
client = google.cloud.logging.Client()
client.setup_logging()
log_message = f"chunked contents are: {chunked_content[0][:20]}"
logging.info(log_message)
```

Click **Check my progress** to verify the objective.

Download, process and chunk data semantically

Check my progress

Assessment Completed!

Task 3. Prepare your vector database

In this section, you will set up a Firestore database to store the processed NYC Food Safety Manual chunks and their embeddings for efficient retrieval. You'll then build a search function to find relevant information based on a user query.

1. Create a Firestore database with the default name of `(default)` in Native Mode and leave the other settings to default.

2. Next, in your Colab Enterprise Notebook populate a `db` variable with a Firestore Client.

3. Use a variable called `collection` to create a reference to a collection named **food-safety**.

4. Using a combination of your lists `chunked_content` and `chunked_embeddings`, add a document to your collection for each of your chunked documents. Each document can be assigned a random ID, but it should have a field called `content` to store the chunk text and a field called `embedding` to store a Firestore Vector() of the associated embedding.

5. Create a vector index for your collection using your `embedding` field.

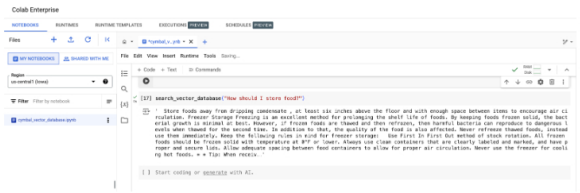
Note: A **find_nearest()** operation cannot be executed on a collection without an index. When attempted, the system will return an error message including instructions to create the index using a gcloud command.

6. Complete the function below to receive a query, get its embedding, and compile a context consisting of the text from the 5 documents with the most similar embeddings. This time, use the `embed_query()` method of the LangChain VertexAIEmbeddings `embedding_model` to embed the user's query.

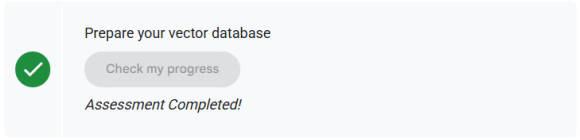
```
def search_vector_database(query: str):

  context = ""

  # 1. Generate the embedding of the query

  # 2. Get the 5 nearest neighbors from your collection.
  # Call the get() method on the result of your call to
  # find_nearest to retrieve document snapshots.

  # 3. Call to_dict() on each snapshot to load its data.
  # Combine the snapshots into a single string named context


  return context
```

7. Next, call the function with the query `How should I store food?` to confirm it's functionality.

```
search_vector_database("How should I store food?")
```



Click **Check my progress** to verify the objective.



Task 4. Deploy a Generative AI application to search your vector store

Now that your vector database is prepared, in this section you will work on the client application to query it and return answers generated by `Gemini`.

1. Activate **Cloud Shell** by selecting the icon in the upper right of the Cloud Console.

2. To set up the base application and install packages, run the following command in **Cloud Shell**:

```
gcloud storage cp -r gs://partner-genai-bucket/genai069/gen-ai-assessment .
cd gen-ai-assessment
python3 -m pip install -r requirements.txt
```

3. Run this base Flask application in **Cloud Shell** with:

```
python3 main.py
```

4. The app will run by default on Port `8080`. Preview the app by clicking the **Web Preview** icon at the top of the Cloud Shell panel, then **Preview on port 8080**.

5. You should be able to see the app and get a greeting, but if you try to ask `FreshBot` any questions, it will currently only reply **Not implemented**.

6. Back in the Cloud Console, you can press **CTRL+C** to stop the app from running. Click **Open Editor** at the top of the Cloud Shell panel.

7. Within the Explorer panel on the left of the Cloud Shell Editor, select the file **generative-ai-assessment/main.py**

8. Just below the initialization of the **Firestore database** client, find the first comment labeled `TODO`. Assign a reference to your **food-safety** collection to the `collection` variable.

Note: As you work through the next few steps, you can test your application by

running the command **python3 main.py**. Alternatively, you can also use the **Web Preview** option to preview the application.

9. Beneath the next two comments beginning `TODO`, assign the appropriate models to the variables `embedding_model` (using the same embedding model version of `text-embedding-005` you used earlier in the lab) and `gen_model` (using the generative model version "gemini-2.0-flash" and a temperature of 0).

10. Complete the function `search_vector_database` as you did above.

11. Complete the `ask_gemini` function to instruct your `gen_model` to answer a question based on the context retrieved from the vector database.

Since some food safety content involves knives and potential burns, set Gemini's Dangerous Content safety setting to block only high-probability dangerous content.

12. When you have completed these steps, make sure the Flask app is running in your Cloud Shell Terminal and test it by making sure it can answer the following question:

```
What temperature range do Mesophilic Bacteria grow best in?
```

You should receive a response with a temperature range of `50 to 110 degrees Fahrenheit` from the model.

Now, we will deploy the Flask application from this directory to **Cloud Run**.

13. Navigate in the Cloud Console to **Artifact Registry** and find the `cymbal-artifact-repo` repo that has been created for you in the `us-central1` location. Use the copy button to copy the repo URI.

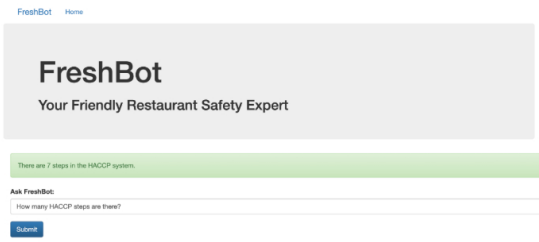
14. Create a Docker image named `cymbal-image` using `Dockerfile` by running the following command:

```
docker build -t cymbal-image -f Dockerfile .
```

15. Push the image to the Artifact Registry repository. Refer to this documentation if you need assistance.

16. Deploy the app to **Cloud Run** as a service named `cymbal-freshbot-service`. Allow the service to accept unauthenticated invocations.

17. Test your model through its **Cloud Run** service URL with the following question: `What is FIFO in food safety?`



Click **Check my progress** to verify the objective.

Deploy a Generative AI application to search your vector store

Check my progress

Assessment Completed!

Congratulations!

In this challenge lab, you have demonstrated your ability to load a text document and split it into chunks, generate embeddings for each chunk, store the text chunks and their embeddings, conduct vector search to return similar documents to a query document and complete a RAG framework by having Gemini generate a response based on a context of similar documents to a query.

Manual Last Updated April 09, 2025

Lab Last Tested April 09, 2025

the respective companies with which they are associated.