Building an LLM and RAG-based chat application using AlloyDB AI and LangChain

ABOUT THIS CODELAB

schedule 92 minutes   subject Last updated January 24, 2024   account_circle Written by Gleb Otochkin

1. Introduction

In this codelab you will learn how to deploy the GenAI Databases Retrieval Service and create a sample interactive application using the deployed environment.

You can get more information about the GenAI Retrieval Service and the sample application here.

Prerequisites

- A basic understanding of the Google Cloud Console
- Basic skills in command line interface and Google Cloud shell

What you'll learn

- How to deploy AlloyDB Cluster
- How to connect to the AlloyDB
- How to configure and deploy GenAI Databases Retrieval Service
- How to deploy a sample application using the deployed service

What you'll need

- A Google Cloud Account and Google Cloud Project
- A web browser such as Chrome

2. Setup and Requirements

Self-paced environment setup

1. Sign-in to the Google Cloud Console and create a new project or reuse an existing one. If you don't already have a Gmail or Google Workspace account, you must create one.

- The **Project name** is the display name for this project's participants. It is a character string not used by Google APIs. You can always update it.
- The **Project ID** is unique across all Google Cloud projects and is immutable (cannot be changed after it has been set). The Cloud Console auto-generates a unique string; usually you don't care what it is. In most codelabs, you'll need to reference your Project ID (typically identified

as PROJECT_ID). If you don't like the generated ID, you might generate another random one. Alternatively, you can try your own, and see if it's available. It can't be changed after this step and remains for the duration of the project.

- For your information, there is a third value, a **Project Number**, which some APIs use. Learn more about all three of these values in the documentation.

Caution: A project ID is globally unique and can't be used by anyone else after you've selected it. You are the only user of that ID. Even if a project is deleted, the ID can't be used again

Note: If you use a Gmail account, you can leave the default location set to **No organization**. If you use a Google Workspace account, choose a location that makes sense for your organization.

2. Next, you'll need to enable billing in the Cloud Console to use Cloud resources/APIs. Running through this codelab won't cost much, if anything at all. To shut down resources to avoid incurring billing beyond this tutorial, you can delete the resources you created or delete the project. New Google Cloud users are eligible for the $300 USD Free Trial program.

Start Cloud Shell

While Google Cloud can be operated remotely from your laptop, in this codelab you will be using Google Cloud Shell, a command line environment running in the Cloud.

From the Google Cloud Console, click the Cloud Shell icon on the top right toolbar:

It should only take a few moments to provision and connect to the environment. When it is finished, you should see something like this:

This virtual machine is loaded with all the development tools you'll need. It offers a persistent 5GB home directory, and runs on Google Cloud, greatly enhancing network performance and authentication. All of your work in this codelab can be done within a browser. You do not need to install anything.

3. Before you begin

Enable API

Output:

Please be aware that some resources you enable are going to incur some cost if you are not using the promotional tier. In normal circumstances if all the resources are destroyed upon completion of the lab the cost of all resources would not exceed $5. We recommend checking your billing and making sure the exercise is acceptable for you.

Inside Cloud Shell, make sure that your project ID is setup:

Usually the project ID is shown in parentheses in the command prompt in the cloud shell as it is shown in the picture:

```
gcloud config set project [YOUR-PROJECT-ID]
```

Then set the PROJECT_ID environment variable to your Google Cloud project ID:

```
PROJECT_ID=$(gcloud config get-value project)
```

Enable all necessary services:

```
gcloud services enable alloydb.googleapis.com \
                       compute.googleapis.com \
                       cloudresourcemanager.googleapis.com \
                       servicenetworking.googleapis.com \
                       vpcaccess.googleapis.com \
                       aiplatform.googleapis.com \
                       cloudbuild.googleapis.com \
                       artifactregistry.googleapis.com \
                       run.googleapis.com \
                       iam.googleapis.com
```

Expected output

```
student@cloudshell:~ (gleb-test-short-004)$ gcloud services enable alloydb.googleapis.com \
                       compute.googleapis.com \
                       cloudresourcemanager.googleapis.com \
                       servicenetworking.googleapis.com \
                       vpcaccess.googleapis.com \
                       aiplatform.googleapis.com \
                       cloudbuild.googleapis.com \
                       artifactregistry.googleapis.com \
                       run.googleapis.com \
                       iam.googleapis.com
Operation "operations/acf.p2-404051529011-664c71ad-cb2b-4ab4-86c1-1f3157d70ba1" finished succes
```

Configure your default region to us-central1 to use the Vertex AI models. Read more about regional restrictions.

```
gcloud config set compute/region us-central1
```

4. Deploy AlloyDB Cluster

Before creating an AlloyDB cluster we need to allocate a private IP range in our VPC to be used by the future AlloyDB instance, after that we will be able to create the cluster and instance.

Create private IP range

We need to configure Private Service Access configuration in our VPC for AlloyDB. The assumption here is that we have the "default" VPC network in the project and it is going to be used for all actions.

Create the private IP range:

```
gcloud compute addresses create psa-range \
    --global \
    --purpose=VPC_PEERING \
    --prefix-length=16 \
    --description="VPC private service access" \
    --network=default
```

Create private connection using the allocated IP range:

```
gcloud services vpc-peerings connect \
    --service=servicenetworking.googleapis.com \
    --ranges=psa-range \
    --network=default
```

Note: The second command takes a couple of minutes to execute

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ gcloud compute addresses create psa-range \
  --global \
  --purpose=VPC_PEERING \
  --prefix-length=16 \
  --description="VPC private service access" \
  --network=default
Created [https://www.googleapis.com/compute/v1/projects/test-project-402417/global/addresses/psa-rar

student@cloudshell:~ (test-project-402417)$ gcloud services vpc-peerings connect \
  --service=servicenetworking.googleapis.com \
  --ranges=psa-range \
  --network=default
Operation "operations/pssn.p24-4470404856-595e209f-19b7-4669-8a71-cbd45de8ba66" finished succe

student@cloudshell:~ (test-project-402417)$
```

Create AlloyDB Cluster

Create an AlloyDB cluster in the default region:

```
export PGPASSWORD=`openssl rand -hex 12`
export REGION=us-central1
export ADBCLUSTER=alloydb-aip-01
gcloud alloydb clusters create $ADBCLUSTER \
    --password=$PGPASSWORD \
    --network=default \
    --region=$REGION
```

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ export PGPASSWORD=`openssl rand -base64 12`
export REGION=us-central1
export ADBCLUSTER=alloydb-aip-01
```

```
gcloud alloydb clusters create $ADBCLUSTER \
   --password=$PGPASSWORD \
   --network=default \
   --region=$REGION
Operation ID: operation-1697655441138-6080235852277-9e7f04f5-2012fce4
Creating cluster...done.
```

Note the PostgreSQL password for future use:

```
echo $PGPASSWORD
```

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ echo $PGPASSWORD
bbefbfde7601985b0dee5723
```

Create AlloyDB Primary Instance

Create an AlloyDB primary instance for our cluster:

Note: The execution can take 10-15 minutes to complete

```
export REGION=us-central1
gcloud alloydb instances create $ADBCLUSTER-pr \
    --instance-type=PRIMARY \
    --cpu-count=2 \
    --region=$REGION \
    --cluster=$ADBCLUSTER
```

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ gcloud alloydb instances create $ADBCLUSTER-pr \
   --instance-type=PRIMARY \
   --cpu-count=2 \
   --region=$REGION \
   --availability-type ZONAL \
   --cluster=$ADBCLUSTER
Operation ID: operation-1697659203545-6080315c6e8ee-391805db-25852721
Creating instance...done.
```

5. Prepare GCE Virtual Machine

Create Service Account

Since we will use our VM to deploy our GenAI Databases Retrieval service and host a sample application, the first step is to create a Google Service Account (GSA). The GSA will be used by the GCE VM, and we will need to grant it the necessary privileges to work with other services.

In the Cloud Shell execute:

```
PROJECT_ID=$(gcloud config get-value project)
gcloud iam service-accounts create compute-aip --project $PROJECT_ID
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/cloudbuild.builds.editor"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/artifactregistry.admin"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/storage.admin"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/run.admin"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/iam.serviceAccountUser"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/alloydb.viewer"
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:compute-aip@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/aiplatform.user"
```

Deploy GCE VM

Create a GCE VM in the same region and VPC as the AlloyDB cluster.

In Cloud Shell execute:

```
export ZONE=us-central1-a
gcloud compute instances create instance-1 \
  --zone=$ZONE \
  --scopes=https://www.googleapis.com/auth/cloud-platform \
  --service-account=compute-aip@$PROJECT_ID.iam.gserviceaccount.com
```

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ export ZONE=us-central1-a
student@cloudshell:~ (test-project-402417)$ gcloud compute instances create instance-1 \
  --zone=$ZONE \
  --scopes=https://www.googleapis.com/auth/cloud-platform
  --service-account=compute-aip@$PROJECT_ID.iam.gserviceaccount.com
Created [https://www.googleapis.com/compute/v1/projects/test-project-402417/zones/us-central1-a/insta
NAME: instance-1
ZONE: us-central1-a
```

```
MACHINE_TYPE: n1-standard-1
PREEMPTIBLE:
INTERNAL_IP: 10.128.0.2
EXTERNAL_IP: 34.71.192.233
STATUS: RUNNING
```

Install Postgres Client

Install the PostgreSQL client software on the deployed VM

Connect to the VM:

Note: First time the SSH connection to the VM can take longer since the process includes creation of RSA key for secure connection and propagating the public part of the key to the project

```
gcloud compute ssh instance-1 --zone=us-central1-a
```

Expected console output:

```
student@cloudshell:~ (test-project-402417)$ gcloud compute ssh instance-1 --zone=us-central1-a
Updating project ssh metadata...working..Updated [https://www.googleapis.com/compute/v1/projects/tes
Updating project ssh metadata...done.
Waiting for SSH key to propagate.
Warning: Permanently added 'compute.5110295539541121102' (ECDSA) to the list of known hosts.
Linux instance-1 5.10.0-26-cloud-amd64 #1 SMP Debian 5.10.197-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
student@instance-1:~$
```

Install the software running command inside the VM:

```
sudo apt-get update
sudo apt-get install --yes postgresql-client
```

Expected console output:

```
student@instance-1:~$ sudo apt-get update
sudo apt-get install --yes postgresql-client
Get:1 https://packages.cloud.google.com/apt google-compute-engine-bullseye-stable InRelease [5146 E
Get:2 https://packages.cloud.google.com/apt cloud-sdk-bullseye InRelease [6406 B]
```

```
Hit:3 https://deb.debian.org/debian bullseye InRelease
Get:4 https://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:5 https://packages.cloud.google.com/apt google-compute-engine-bullseye-stable/main amd64 Pack
Get:6 https://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:7 https://deb.debian.org/debian bullseye-backports InRelease [49.0 kB]
...redacted...
update-alternatives: using /usr/share/postgresql/13/man/man1/psql.1.gz to provide /usr/share/man/man1
Setting up postgresql-client (13+225) ...
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.31-13+deb11u7) ...
```

Connect to the Instance

Connect to the primary instance from the VM using psql.

Continue with the opened SSH session to your VM. If you have been disconnected then connect again using the same command as above.

Use the previously noted $PGASSWORD and the cluster name to connect to AlloyDB from the GCE VM:

```
export PGPASSWORD=<Noted password>
```

```
export PROJECT_ID=$(gcloud config get-value project)
export REGION=us-central1
export ADBCLUSTER=alloydb-aip-01
export INSTANCE_IP=$(gcloud alloydb instances describe $ADBCLUSTER-pr --cluster=$ADBCLUSTER --r
psql "host=$INSTANCE_IP user=postgres sslmode=require"
```

Expected console output:

```
student@instance-1:~$ export PGPASSWORD=P9...
student@instance-1:~$ export REGION=us-central1
student@instance-1:~$ export ADBCLUSTER=alloydb-aip-01
student@instance-1:~$ export INSTANCE_IP=export INSTANCE_IP=$(gcloud alloydb instances descrit
student@instance-1:~$ psql "host=$INSTANCE_IP user=postgres sslmode=require"
psql (13.11 (Debian 13.11-0+deb11u1), server 14.7)
WARNING: psql major version 13, server major version 14.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=>
```

Exit from the psql session keeping the SSH connection up:

```
exit
```

Expected console output:

```
postgres=> exit
student@instance-1:~$
```

6. Initialize the database

We are going to use our client VM as a platform to populate our database with data and host our application. The first step is to create a database and populate it with data.

Create Database

Create a database with name "assistantdemo".

In the GCE VM session execute:

Note: If your SSH session was terminated you need to reset your environment variables such as:

export PGPASSWORD=<Noted password>

export INSTANCE_IP=<AlloyDB Omni Instance IP>

```
psql "host=$INSTANCE_IP user=postgres" -c "CREATE DATABASE assistantdemo"
```

Expected console output:

```
student@instance-1:~$ psql "host=$INSTANCE_IP user=postgres" -c "CREATE DATABASE assistantde
CREATE DATABASE
student@instance-1:~$
```

Enable pgVector extension.

```
psql "host=$INSTANCE_IP user=postgres dbname=assistantdemo" -c "CREATE EXTENSION vector"
```

Expected console output:

```
student@instance-1:~$ psql "host=$INSTANCE_IP user=postgres dbname=assistantdemo" -c "CREATE
CREATE EXTENSION
student@instance-1:~$
```

Install Python

To continue we are going to use prepared Python scripts from GitHub repository but before doing that we need to install the required software.

In the GCE VM execute:

```
sudo apt install -y git build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev curl \
libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
curl https://pyenv.run | bash
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
exec "$SHELL"
```

Expected console output:

```
student@instance-1:~$ sudo apt install -y git build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev curl \
libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev
curl https://pyenv.run | bash
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(pyenv init -)"' >> ~/.bashrc
exec "$SHELL"
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
xz-utils is already the newest version (5.2.5-2.1~deb11u1).
The following additional packages will be installed:
...
```

Install Python 3.11.x.

In the GCE VM execute:

```
pyenv install 3.11.6
pyenv global 3.11.6
python -V
```

Expected console output:

```
student@instance-1:~$ pyenv install 3.11.6
pyenv global 3.11.6
python -V
Downloading Python-3.11.6.tar.xz...
-> https://www.python.org/ftp/python/3.11.6/Python-3.11.6.tar.xz
```

```
Installing Python-3.11.6...
Installed Python-3.11.6 to /home/student/.pyenv/versions/3.11.6
Python 3.11.6
student@instance-1:~$
```

Populate Database

Clone the GitHub repository with the code for the retrieval service and sample application.

In the GCE VM execute:

```
git clone https://github.com/GoogleCloudPlatform/genai-databases-retrieval-app.git
```

Expected console output:

```
student@instance-1:~$ git clone https://github.com/GoogleCloudPlatform/genai-databases-retrieval-app
Cloning into 'genai-databases-retrieval-app'...
remote: Enumerating objects: 525, done.
remote: Counting objects: 100% (336/336), done.
remote: Compressing objects: 100% (201/201), done.
remote: Total 525 (delta 224), reused 179 (delta 135), pack-reused 189
Receiving objects: 100% (525/525), 46.58 MiB | 16.16 MiB/s, done.
Resolving deltas: 100% (289/289), done.
```

Prepare configuration file

In the GCE VM execute:

Note: If your SSH session was terminated you need to set your environment variables such as:

export PGPASSWORD=<Noted password>

REGION=us-central1

INSTANCE_IP=$(gcloud alloydb instances describe $ADBCLUSTER-pr –cluster=$ADBCLUSTER –region=$REGION –format="value(ipAddress)")

```
cd genai-databases-retrieval-app/retrieval_service
cp example-config.yml config.yml
sed -i s/127.0.0.1/$INSTANCE_IP/g config.yml
sed -i s/my-password/$PGPASSWORD/g config.yml
sed -i s/my_database/assistantdemo/g config.yml
sed -i s/my-user/postgres/g config.yml
cat config.yml
```

Expected console output:

```
student@instance-1:~$ cd genai-databases-retrieval-app/retrieval_service
cp example-config.yml config.yml
sed -i s/127.0.0.1/$INSTANCE_IP/g config.yml
sed -i s/my-password/$PGPASSWORD/g config.yml
sed -i s/my_database/assistantdemo/g config.yml
sed -i s/my-user/postgres/g config.yml
cat config.yml
host: 0.0.0.0
# port: 8080
datastore:
  # Example for AlloyDB
  kind: "postgres"
  host: 10.65.0.2
  # port: 5432
  database: "assistantdemo"
  user: "postgres"
  password: "P9..."
```

Populate database with the sample dataset.

In the GCE VM execute:

```
pip install -r requirements.txt
python run_database_init.py
```

Expected console output(redacted):

```
student@instance-1:~/genai-databases-retrieval-app/retrieval_service$ pip install -r requirements.txt
python run_database_init.py
Collecting asyncpg==0.28.0 (from -r requirements.txt (line 1))
  Obtaining dependency information for asyncpg==0.28.0 from https://files.pythonhosted.org/packages/7
  Downloading asyncpg-0.28.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.meta
Collecting fastapi==0.101.1 (from -r requirements.txt (line 2))
...
database init done.
student@instance-1:~/genai-databases-retrieval-app/retrieval_service$
```

7. Deploy the Extension Service to Cloud Run

Now we can deploy the extension service to Cloud Run.

Create Service Account

Create a service account for the extension service and grant necessary privileges.

Open another Cloud Shell tab using the sign "+" at the top.

In the new cloud shell tab execute:

```
export PROJECT_ID=$(gcloud config get-value project)
gcloud iam service-accounts create retrieval-identity
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:retrieval-identity@$PROJECT_ID.iam.gserviceaccount.com" \
  --role="roles/aiplatform.user"
```

Expected console output:

```
student@cloudshell:~ (gleb-test-short-003)$ gcloud iam service-accounts create retrieval-identity
Created service account [retrieval-identity].
```

Close the tab by either execution command "exit" in the tab:

```
exit
```

Deploy the extension service

Continue in the first tab where you are connected to the VM through SSH by deploying the service.

In the VM SSH session execute:

```
cd ~/genai-databases-retrieval-app
gcloud alpha run deploy retrieval-service \
    --source=./retrieval_service/\
    --no-allow-unauthenticated \
    --service-account retrieval-identity \
    --region us-central1 \
    --network=default \
    --quiet
```

Expected console output:

```
student@instance-1:~/genai-databases-retrieval-app$ gcloud alpha run deploy retrieval-service \
  --source=./retrieval_service/\
  --no-allow-unauthenticated \
  --service-account retrieval-identity \
  --region us-central1 \
  --network=default
This command is equivalent to running `gcloud builds submit --tag [IMAGE] ./retrieval_service/` and `gcl

Building using Dockerfile and deploying container to Cloud Run service [retrieval-service] in project [gleb
X Building and deploying... Done.
  ✓ Uploading sources...
  ✓ Building Container... Logs are available at [https://console.cloud.google.com/cloud-build/builds/6ebe
  ✓ Creating Revision...
  ✓ Routing traffic...
    Setting IAM Policy...
Completed with warnings:
  Setting IAM policy failed, try "gcloud beta run services remove-iam-policy-binding --region=us-central1
Service [retrieval-service] revision [retrieval-service-00002-4pl] has been deployed and is serving 100 pe
Service URL: https://retrieval-service-onme64eorq-uc.a.run.app
student@instance-1:~/genai-databases-retrieval-app$
```

Verify The Service

Now we can check if the service runs correctly and the VM has access to the endpoint.

In the VM SSH session execute:

```
curl -H "Authorization: Bearer $(gcloud auth print-identity-token)" $(gcloud  run services list
```

Expected console output:

```
student@instance-1:~/genai-databases-retrieval-app$ curl -H "Authorization: Bearer $(gcloud auth print-
{"message":"Hello World"}student@instance-1:~/genai-databases-retrieval-app$
```

If we see the "Hello World" message it means our service is up and serving the requests.
8. Deploy Sample Application

Now we have the extension service up and running and can deploy a sample application which is going to use the service. The application can be deployed on the VM or any other service like Cloud Run, Kubernetes or even locally on a laptop. Here we are going to show how to deploy it on the VM.

Prepare the environment

We continue to work on our VM. We need to add necessary modules to Python.

In the VM SSH session execute:

```
cd ~/genai-databases-retrieval-app/langchain_tools_demo
pip install -r requirements.txt
```

Expected output (redacted):

```
student@instance-1:~$ cd ~/genai-databases-retrieval-app/langchain_tools_demo
pip install -r requirements.txt
Collecting fastapi==0.104.0 (from -r requirements.txt (line 1))
```

```
  Obtaining dependency information for fastapi==0.104.0 from https://files.pythonhosted.org/packages/dl
  Downloading fastapi-0.104.0-py3-none-any.whl.metadata (24 kB)
...
```

Run Assistant Application

Now we can start our application

In the VM SSH session execute:

```
export BASE_URL=$(gcloud  run services list --filter="(retrieval-service)" --format="value(URL)
python main.py
```

Expected output (redacted):

```
student@instance-1:~/genai-databases-retrieval-app/langchain_tools_demo$ export BASE_URL=$(gclc
student@instance-1:~/genai-databases-retrieval-app/langchain_tools_demo$ python main.py
INFO:     Started server process [28565]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8081 (Press CTRL+C to quit)
```

Connect to the Application

You have several ways to connect to the application running on the VM. For example you can open port 8081 on the VM using firewall rules in the VPC or create a load balancer with public IP. Here we are going to use a SSH tunnel to the VM translating the local port 8081 to the VM port 8081.

Open another Cloud Shell tab using the sign "+" at the top.

In the new cloud shell tab start the tunnel to your VM by executing the gcloud command:

```
gcloud compute ssh instance-1 --zone=us-central1-a -- -NL 8080:localhost:8081
```

It will show an error "Cannot assign requested address" - please ignore it.

Here is the expected output:

```
student@cloudshell:~ gcloud compute ssh instance-1 --zone=us-central1-a -- -NL 8080:localhost:8081
bind [::1]:8081: Cannot assign requested address
```

It opens port 8081 on your cloud shell which can be used for the "Web preview".

Click on the "Web preview" button on the right top of your Cloud Shell and from the drop down menu choose "Preview on port 8080"

It opens a new tab in your web browser with the application interface. You should be able to see the "SFO Airport Assistant" page and you can post your question to the assistant at the bottom of the page.

This application showcases an "SFO Airport Assistant": a San Francisco Airport-based AI assistant that has access to information about airports, flights, and amenities. It can help answer users questions like:

When is the next flight to Denver?

Are there any luxury shops around gate D50?

Where can I get coffee near gate A6?

Where can I buy a gift?

The application uses the latest Google foundation models to generate responses and augment it by information about flights and amenities from the operational AlloyDB database. You can read more about this demo application on the Github page of the project.

9. Clean up environment

Now when all tasks are completed we can clean up our environment

Delete Cloud Run Service

In Cloud Shell execute:

```
gcloud run services delete retrieval-service --region us-central1
```

Expected console output:

```
student@cloudshell:~ (gleb-test-short-004)$ gcloud run services delete retrieval-service --region us-cent
Service [retrieval-service] will be deleted.

Do you want to continue (Y/n)?  Y

Deleting [retrieval-service]...done.
Deleted service [retrieval-service].
```

Delete the Service Account for cloud run service

In Cloud Shell execute:

```
PROJECT_ID=$(gcloud config get-value project)
gcloud iam service-accounts delete retrieval-identity@$PROJECT_ID.iam.gserviceaccount.com --qui
```

Expected console output:

```
student@cloudshell:~ (gleb-test-short-004)$ PROJECT_ID=$(gcloud config get-value project)
Your active configuration is: [cloudshell-222]
student@cloudshell:~ (gleb-test-short-004)$ gcloud iam service-accounts delete retrieval-identity@$PR
deleted service account [retrieval-identity@gleb-test-short-004.iam.gserviceaccount.com]
student@cloudshell:~ (gleb-test-short-004)$
```

Destroy the AlloyDB instances and cluster when you are done with the lab

Delete AlloyDB cluster and all instances

The cluster is destroyed with option force which also deletes all the instances belonging to the cluster.

In the cloud shell define the project and environment variables if you've been disconnected and all the previous settings are lost:

```
gcloud config set project <your project id>
```

```
gcloud config set project <your project id>
export REGION=us-central1
export ADBCLUSTER=alloydb-aip-01
export PROJECT_ID=$(gcloud config get-value project)
```

Delete the cluster:

```
gcloud alloydb clusters delete $ADBCLUSTER --region=$REGION --force
```

Note: The command takes 3-5 minutes to execute

Expected console output:

```
student@cloudshell:~ (test-project-001-402417)$ gcloud alloydb clusters delete $ADBCLUSTER --regio
All of the cluster data will be lost when the cluster is deleted.

Do you want to continue (Y/n)?  Y

Operation ID: operation-1697820178429-6082890a0b570-4a72f7e4-4c5df36f
Deleting cluster...done.
```

Delete AlloyDB Backups

Delete all AlloyDB backups for the cluster:

Note: The command will destroy all data backups for the cluster with name specified in environment variable

```
for i in $(gcloud alloydb backups list --filter="CLUSTER_NAME: projects/$PROJECT_ID/locations/$I
```

Expected console output:

```
student@cloudshell:~ (test-project-001-402417)$ for i in $(gcloud alloydb backups list --filter="CLUSTEF
Operation ID: operation-1697826266108-60829fb7b5258-7f99dc0b-99f3c35f
Deleting backup...done.
```

Now we can destroy our VM

Delete GCE VM

In Cloud Shell execute:

```
export GCEVM=instance-1
export ZONE=us-central1-a
gcloud compute instances delete $GCEVM \
    --zone=$ZONE \
    --quiet
```

Expected console output:

```
student@cloudshell:~ (test-project-001-402417)$ export GCEVM=instance-1
export ZONE=us-central1-a
gcloud compute instances delete $GCEVM \
   --zone=$ZONE \
   --quiet
Deleted
```

Delete the Service Account for GCE VM and The Retrieval service

In Cloud Shell execute:

```
PROJECT_ID=$(gcloud config get-value project)
gcloud iam service-accounts delete compute-aip@$PROJECT_ID.iam.gserviceaccount.com --quiet
```

Expected console output:

```
student@cloudshell:~ (gleb-test-short-004)$ PROJECT_ID=$(gcloud config get-value project)
gcloud iam service-accounts delete compute-aip@$PROJECT_ID.iam.gserviceaccount.com --quiet
Your active configuration is: [cloudshell-222]
deleted service account [compute-aip@gleb-test-short-004.iam.gserviceaccount.com]
student@cloudshell:~ (gleb-test-short-004)$
```

10. Congratulations

Congratulations for completing the codelab.

What we've covered

- How to deploy AlloyDB Cluster
- How to connect to the AlloyDB
- How to configure and deploy GenAI Databases Retrieval Service
- How to deploy a sample application using the deployed service