# Process Mining -Technical Documentation

[Process Mining -Technical Documentation 1](#_Toc15558)

[Step-by-Step Setup Guide 1](#_Toc5324)

[Section 1: Google Cloud Project Setup 2](#_Toc14379)

[1. Create a New Google Cloud Project: 2](#_Toc21097)

[2. Enable Required Google Cloud APIs: These APIs allow your project to use specific Google Cloud services. 2](#_Toc8846)

[Section 2: Cloud SQL PostgreSQL Database Setup 2](#_Toc4524)

[1. Create a Cloud SQL PostgreSQL Instance: 2](#_Toc12575)

[2. Note Down Database Connection Details: 3](#_Toc10771)

[3. Create the Database and Enable pgvector Extension: 3](#_Toc8257)

[Section 3: Local Development Environment Setup 3](#_Toc9676)

[1. Install Python: 4](#_Toc31580)

[2. Create Your Project Folder and Files: 4](#_Toc11463)

[3. Populate requirements.txt: 4](#_Toc5153)

[4. Populate Procfile: 4](#_Toc30062)

[5. Populate .env (for Local Testing Only): 4](#_Toc11498)

[6. Populate main.py (Your Application Code): 5](#_Toc20478)

[7. Set Up Python Virtual Environment and Install Dependencies: 5](#_Toc14109)

[Section 4: Local Testing with Cloud SQL Proxy (Optional but Recommended) 5](#_Toc18906)

[1. Download Cloud SQL Proxy: 5](#_Toc19908)

[2. Run the Cloud SQL Proxy: 5](#_Toc4179)

[3. Run Your FastAPI Application Locally: 6](#_Toc5552)

[4. Test Locally with curl: 6](#_Toc13998)

[Section 5: Deploy to Google Cloud Run 6](#_Toc14303)

[1. Install and Configure gcloud CLI: 6](#_Toc24956)

[2. Grant Cloud Run Service Account Permissions (CRITICAL!): 6](#_Toc4103)

[3. Deploy Your Application to Cloud Run: 7](#_Toc10614)

[Section 6: Test the Deployed Cloud Run Service 7](#_Toc14441)

[1. Test with curl: 7](#_Toc23166)

[2. Check Cloud Run Logs: 7](#_Toc9570)

This guide will walk you through setting up your semantic search application on Google Cloud, from project creation to deployment and testing. We'll ensure all necessary components, including Cloud SQL (PostgreSQL with pgvector) and Vertex AI (Gemini 1.5 Flash for LLM and text embeddings), are correctly configured.

**Assumptions:**

* You have a Google Cloud account.
* You are comfortable with basic command-line operations (PowerShell/CMD on Windows, Terminal on macOS/Linux).

## 

## Step-by-Step Setup Guide

### Section 1: Google Cloud Project Setup

This is where you'll create a new home for your project in Google Cloud.

#### Create a New Google Cloud Project:

* Go to the Google Cloud Console: [https://console.cloud.google.com/](https://console.cloud.google.com/" \t "_blank)
* Click on the project selector dropdown at the top (it usually shows your current project name or "My First Project").
* Click "New Project".
* Enter a "Project name" (e.g., IncidentSemanticSearch). The Project ID will be generated automatically (e.g., incident-semantic-search-123456). Note this Project ID down, as you'll use it frequently.
* Click "Create".
* Once the project is created, make sure you select your new project from the project selector dropdown.

#### Enable Required Google Cloud APIs: These APIs allow your project to use specific Google Cloud services.

* In the Google Cloud Console, navigate to "APIs & Services" > "Enabled APIs & Services".
* Click "+ ENABLE APIS AND SERVICES".
* Search for and enable the following APIs, one by one:
  + Cloud SQL Admin API
  + Vertex AI API
  + Cloud Run Admin API
  + Cloud Build API
  + Artifact Registry API

### Section 2: Cloud SQL PostgreSQL Database Setup

Here, you'll set up your PostgreSQL database, which will store your incident data and embeddings.

#### Create a Cloud SQL PostgreSQL Instance:

In the Google Cloud Console, navigate to **"SQL"**.

* Click **"+ CREATE INSTANCE"**.
* Choose **"PostgreSQL"**.
* **Choose a password** for the default postgres user. **SAVE THIS PASSWORD SECURELY.**
* Set an **"Instance ID"** (e.g., my-incident-db). This is the name of your database server.
* Choose a **"Region"** (e.g., us-central1). **Make sure this matches the region you will deploy your application to later.**
* Under "Choose a database version," keep the latest stable version (e.g., PostgreSQL 14 or 15).
* Under "Configuration options" > "Connectivity":
  + Select **"Public IP"**.
  + To allow your local machine to connect for setup, under "Authorized networks," click **"ADD NETWORK"**.
  + For "Network name," type My Home IP (or similar).
  + For "Network," enter your current public IP address (you can find this by searching "What is my IP" on Google, or using curl checkip.amazonaws.com in your terminal). Add /32 at the end (e.g., 203.0.113.42/32).
  + Click **"DONE"**.
* **Note:** For increased security, you should remove this 0.0.0.0/0 access after initial setup or if you're only deploying to Cloud Run (Cloud Run connects internally).
  + Review other settings (e.g., machine type, storage) and adjust if needed, but defaults are often fine for testing.
  + Click **"CREATE INSTANCE"**. This will take a few minutes.

#### Note Down Database Connection Details:

Once your Cloud SQL instance is created:

* Click on your instance ID (e.g., my-incident-db).
* On the "Overview" page, note the **"Public IP address"** (e.g., xx.xx.xx.xx). This is your DB\_HOST.
* The default database name is often postgres, but we'll create incidents\_db.
* Your user is postgres and you set the password earlier.
* The default port is 5432.

#### **Create the Database and Enable pgvector Extension:**

You'll need a tool to connect to PostgreSQL. The easiest is psql if you have it installed locally, or you can use the Cloud Shell.

* **Using Cloud Shell (easiest way from GCP):**
  + **Click** the **"Activate Cloud Shell"** icon in the top right corner of the Google Cloud Console (it looks like >\_).
* **Once** Cloud Shell initializes, connect to your database instance:
* **Bash :**
  + gcloud sql connect my-incident-db --user=postgres
* **Replace** my-incident-db with your instance ID. Enter the postgres user password when prompted.
* Once **connected** to the postgres prompt (postgres=>), run these SQL commands:
* SQL
  + CREATE DATABASE incidents\_db;
  + CREATE EXTENSION vector;
* This creates your incidents\_db database and enables the pgvector extension for similarity search.
* Type \q and press Enter to exit psql.

### Section 3: Local Development Environment Setup

This section prepares your computer to run and test the application code.

#### Install Python:

* Ensure you have Python 3.11 installed. Download it from [python.org](https://www.python.org/downloads/" \t "_blank). During installation, select "Add Python to PATH."

#### **Create Your Project Folder and Files:**

* Create a new folder on your computer for your project (e.g., Cloud-Functions-setup).
* Inside this folder, create the following empty files:
  + main.py
  + requirements.txt
  + Procfile
  + .env

#### **Populate requirements.txt:**

This file lists all the Python libraries your application needs.

* Open requirements.txt and add these lines:
  + fastapi
  + uvicorn[standard]
  + psycopg2-binary
  + pgvector
  + google-cloud-aiplatform[generative]==1.52.0
  + python-dotenv

#### **Populate Procfile:**

This file tells Cloud Run how to start your FastAPI application.

* Open Procfile and add this single line:
  + web: uvicorn main:app --host 0.0.0.0 --port $PORT

#### **Populate .env (for Local Testing Only):**

This file holds your database connection details for local running, so you don't hardcode them in your main.py. **This file should NOT be committed to public repositories.**

* Open .env and add your database details. Use localhost for DB\_HOST because you'll connect via the Cloud SQL Proxy.
  + DB\_HOST="localhost"
  + DB\_NAME="incidents\_db"
  + DB\_USER="postgres"
  + DB\_PASSWORD="YOUR\_POSTGRES\_PASSWORD" # Use the password you set in Cloud SQL
  + DB\_PORT="5432"
  + EMBEDDING\_MODEL\_NAME="text-embedding-004"
  + EMBEDDING\_DIMENSION="768"
  + GEMINI\_MODEL\_NAME="gemini-1.5-flash-001"

#### **Populate main.py (Your Application Code):**

This is the core Python code for your FastAPI application, including LLM interaction and database logic.

* Open main.py and copy-paste the entire code block below:
  + https://github.com/Varun2480/Process\_Mining/blob/main/Cloud-Functions-setup/main.py

#### **Set Up Python Virtual Environment and Install Dependencies:**

It's good practice to use a virtual environment to manage project dependencies.

* Open your terminal (PowerShell/CMD on Windows, Terminal on macOS/Linux).
* Navigate to your project folder (e.g., cd G:\POC's\Process Mining\Process\_Mining\Cloud-Functions-setup).
* Create a virtual environment:
* Bash:
  + python -m venv venv\_process\_mining
* Activate the virtual environment:
  + Windows: .\venv\_process\_mining\Scripts\activate
  + macOS/Linux: source venv\_process\_mining/bin/activate (You'll see (venv\_process\_mining) appear at the start of your terminal prompt.)
* Install the dependencies listed in requirements.txt:
* Bash:
  + pip install -r requirements.txt

### Section 4: Local Testing with Cloud SQL Proxy (Optional but Recommended)

This allows you to run your application on your computer and connect securely to your Cloud SQL database.

#### Download Cloud SQL Proxy:

* Go to: [https://cloud.google.com/sql/docs/postgres/connect-external-app#connect-proxy](https://www.google.com/search?q=https://cloud.google.com/sql/docs/postgres/connect-external-app%23connect-proxy" \t "_blank)
* Download the appropriate executable for your OS (e.g., cloud\_sql\_proxy.exe for Windows).
* Place the downloaded executable in your project folder (where main.py is).

#### **Run the Cloud SQL Proxy:**

* Open a new terminal window (keep your existing one for your Python app).
* Navigate to your project folder.
* Run the proxy command. Replace YOUR\_PROJECT\_ID, YOUR\_REGION, and YOUR\_INSTANCE\_NAME with your actual details:
* Bash
  + # For Windows (PowerShell/CMD):
    - .\cloud\_sql\_proxy.exe -instances="YOUR\_PROJECT\_ID:YOUR\_REGION:YOUR\_INSTANCE\_NAME"=tcp:5432
* The proxy will start listening on localhost:5432. Do not close this terminal.

#### **Run Your FastAPI Application Locally:**

* In your **first terminal window** (where your virtual environment is activated), run:
* Bash:
  + python main.py
* You should see messages from Uvicorn indicating the app is running on http://127.0.0.1:8000.

#### **Test Locally with curl:**

* Open a **third terminal window** (or use a tool like Postman/Insomnia).
* Send a POST request to your local app:
* Bash
  + # For Windows (PowerShell/CMD)
    - curl -X POST "http://127.0.0.1:8000/query" -H "Content-Type: application/json" -d "{\"query\_text\": \"Tell me about incidents related to storage capacity alert.\"}"
* You should see a JSON response. Check the logs in your python main.py terminal for output from your print statements.

### Section 5: Deploy to Google Cloud Run

This is where you'll make your application available on Google Cloud.

#### Install and Configure gcloud CLI:

This command-line tool allows you to interact with Google Cloud services.

* Follow the official installation guide: [https://cloud.google.com/sdk/docs/install](https://cloud.google.com/sdk/docs/install" \t "_blank)
* Once installed, open a new terminal and initialize it:
* Bash:
  + gcloud init
* Follow the prompts to log in with your Google account and select your project.

#### **Grant Cloud Run Service Account Permissions (CRITICAL!):**

Your deployed application needs permission to talk to Vertex AI and Cloud SQL. Cloud Run services run under a special Google-managed service account.

* Go to the Google Cloud Console: [https://console.cloud.google.com/](https://console.cloud.google.com/" \t "_blank)
* Navigate to "IAM & Admin" > "IAM".
* Look for a service account named something like PROJECT\_NUMBER-compute@developer.gserviceaccount.com (e.g., 1065449696673-compute@developer.gserviceaccount.com). This is your default Compute Engine service account, often used by Cloud Run.
* Click the pencil icon (Edit principal) next to this service account.
* Click "+ ADD ANOTHER ROLE" and add the following roles:
* Cloud SQL Client
* Vertex AI User
* Click "SAVE".

#### **Deploy Your Application to Cloud Run:**

* Open your terminal and navigate to your project folder (where main.py, requirements.txt, and Procfile are).
* Ensure your venv\_process\_mining is deactivated for deployment (deactivate command). This ensures gcloud uses your system Python and doesn't get confused by the virtual env.
* Run the deployment command. Replace YOUR\_PROJECT\_ID with your actual Project ID.
* Bash
  + gcloud run deploy db-checker --source=. --region us-central1 --memory 512Mi --cpu 1 --timeout 60s --allow-unauthenticated --set-env-vars DB\_HOST="34.66.49.57" --set-env-vars DB\_NAME="incidents\_db" --set-env-vars DB\_USER="postgres" --set-env-vars DB\_PASSWORD="<your-db-password>" --project <your-project-name>
* The command will take a few minutes to build and deploy. Once complete, it will provide a **Service URL** (e.g., https://incident-query-endpoint-1065449696673.us-central1.run.app). **Note this URL down.**

### Section 6: Test the Deployed Cloud Run Service

Now, let's verify that your application is working correctly in the cloud.

#### Test with curl:

* **Open** your terminal (any terminal, no need for virtual env or proxy).
* Use the Service URL provided by the gcloud run deploy command and append /query to it.
* Bash
  + curl -X POST "https://incident-query-endpoint-1065449696673.us-central1.run.app/query" -H "Content-Type: application/json" -d "{\"query\_text\": \"Tell me about incidents related to storage capacity alert.\"}"
* **You** should receive a JSON response from your deployed FastAPI application.

#### **Check Cloud Run Logs:**

* In the Google Cloud Console, navigate to "Cloud Run".
* Select your incident-query-endpoint service.
* Click on the "Logs" tab.
* You should see detailed logs from your application, including messages about the generative model being initialized, the LLM interacting, and database queries being performed. This is your main debugging tool if you encounter any issues.

Congratulations! You have successfully set up and deployed your semantic search application on Google Cloud Run, utilizing Cloud SQL PostgreSQL with pgvector and Vertex AI's Gemini 1.5 Flash model for powerful AI capabilities.