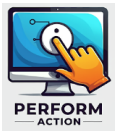
Vertex AI SDK for ABAP

- A quick hands-on guide

Iconography

Please note the below symbol while reading the guide:

	**Read-Only**: This section provides information for understanding. No action required.
	**Action**: Perform the steps described in this section on your system.
	**Validation**: Verify that the results match the expected outcome described in this section.
	**Observation**: Pay close attention to the details highlighted in this section.

Chapter 1: Introduction

	This chapter introduces the Vertex AI SDK for ABAP Handbook, a guide on how to integrate Vertex AI into SAP ABAP environments, with a focus on creating a generative AI application using Retrieval Augmented Generation (RAG).

Welcome to the **Vertex AI SDK for ABAP Handbook!** This comprehensive guide is designed to empower SAP customers like you to seamlessly integrate the power of Vertex AI into your ABAP environment, regardless of your chosen orchestration flavor. Whether you're on-premises, in the cloud, or leveraging a hybrid approach, this handbook will provide you with the knowledge and tools to unlock the full potential of AI within your SAP landscape.

Crafting a Vertex AI Application: A Step-by-Step Guide

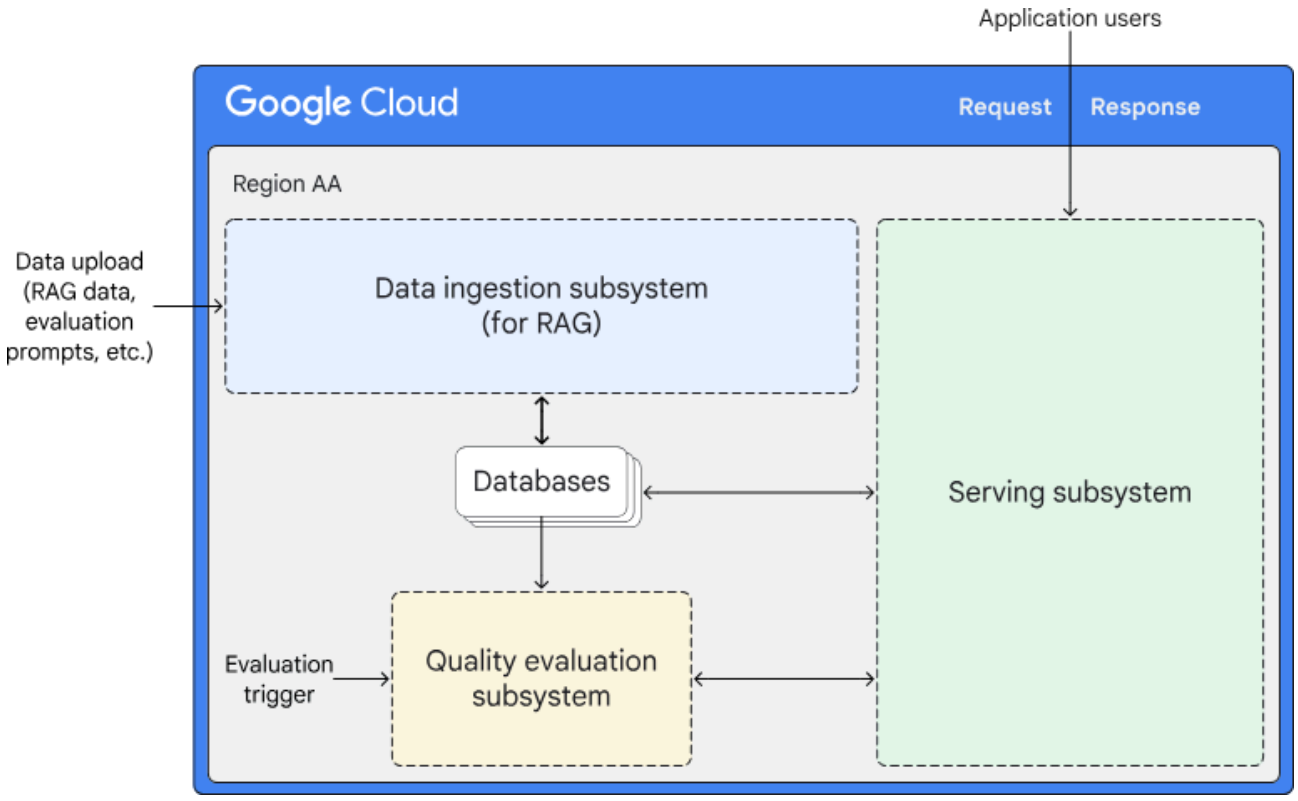
The goal is to equip you with the expertise to harness Vertex AI's capabilities for a wide range of SAP use cases. First focus will be on helping you build a RAG-capable generative AI application all using ABAP.

If you're familiar with RAG (Retrieval Augmented Generation), you can skip ahead and start building the HAZMAT prototype.

RAG-Enabled Architecture: The Key Components

A typical RAG (Retrieval Augmented Generation) architecture consists of three interconnected subsystems. Let's break them down:

- **Data Ingestion Subsystem**: This component handles the gathering and preparation of your data, ensuring it's ready for use by the AI model.
- **Serving Subsystem**: This subsystem manages the deployment and serving of your AI model, allowing it to interact with users and process their requests.
- **Quality Evaluation Subsystem**: This vital component continuously monitors and assesses the performance of your AI model, helping you identify areas for improvement and maintain optimal results.



Dive Deeper: To gain a comprehensive understanding of these subsystems and their roles within a RAG architecture, you can explore this informative link: [Infrastructure for a RAG-capable generative AI application using Vertex AI]

Before we dive deep into creating an application, let's quickly understand the developer workflow for building RAG-enabled applications.

Chapter 2: Pricing

	This chapter outlines the pricing details for utilizing Gemini API and Vertex AI, clarifying that while the Vertex AI SDK for ABAP is free, users will incur costs for using Google Cloud services like Gemini API (which offers both free and paid tiers with regional variations) and Vertex AI (which operates on a pay-as-you-go model).

The Vertex AI SDK for ABAP is offered at no cost. However, you are responsible for the charges that result from your use of Google Cloud services, such as Gemini API or Vertex AI API.

For quick prototyping with Gemini, you use the Gemini API and Google AI Studio. When you access the Gemini API through Google AI Studio, the Gemini API has both free and paid pricing tiers. However, the free tier of Gemini API is not available in all regions. For information about the regions where you can access the free and paid tiers of Gemini API, see Available regions for Google AI Studio and Gemini API. For information about other restrictions, see Use restrictions. For information about pricing and rate limits, see Pricing models.

Vertex AI pricing follows a pay-as-you-go model, which means that you're charged based on the resources you consume. For information about pricing, see Vertex AI pricing.

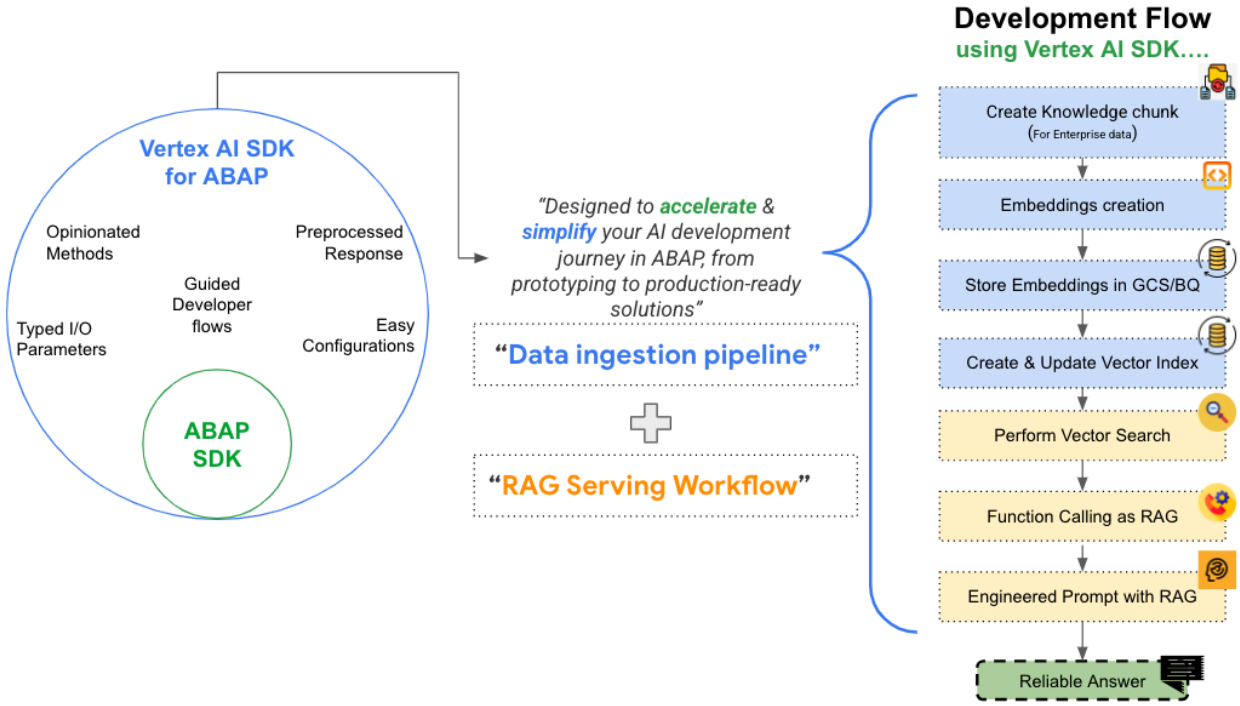
For information about how Gemini with Vertex AI is different from Gemini with Google AI Studio, see Google AI versus Vertex AI differences.

Chapter 3: Development Flow

	This chapter explains how the Vertex AI SDK for ABAP simplifies the integration of Vertex AI into SAP applications, focusing on an opinionated development flow for building Data Ingestion Pipelines and RAG Serving Workflows using ABAP.

The Vertex AI SDK for ABAP is a powerful tool that simplifies integrating Google's Vertex AI with SAP environments using ABAP programming. It streamlines the development of AI-powered enterprise solutions by providing ABAP classes, methods, and AI-centric data types for easy interaction with Vertex AI features. This SDK enables developers to efficiently prepare inputs, interpret outputs, and configure parameters, accelerating the creation of AI-driven applications within the SAP ecosystem.

Using the Vertex AI SDK, developers can follow an opinionated development flow to create Data Ingestion Pipelines and RAG Serving Workflows by using ABAP as the key orchestrator, simplifying the integration of AI capabilities into SAP applications.



Key components of the flow:

1. **Data Ingestion Pipeline:**
    - **Create Knowledge Chunks:** Process enterprise data to generate knowledge chunks.
    - **Embedding Creation:** Convert knowledge chunks into numerical representations (embeddings).
    - **Store Embeddings:** Store embeddings in Google Cloud Storage (GCS) or BigQuery (BQ) or Send to PubSub(PS).
    - **Create & Update Vector Index:** Build and maintain a vector index for efficient similarity search.
2. **RAG Serving Workflow:**
    - **Perform Vector Search:** Retrieve relevant information from the vector index based on user queries.

- **Function Calling as RAG:** Use external functions to augment the RAG's capabilities.
- **Engineered Prompt with RAG:** Craft effective prompts that incorporate retrieved information for accurate and contextually relevant responses.
- **Reliable Answer:** Generate a final answer based on the processed information.

Chapter 4: Overview: HAZMAT Pro (HPro) - Empowering Warehouse Safety with AI

	This chapter introduces HPro, an AI-powered prototype for enhancing warehouse safety, and outlines how to build it using Vertex AI SDK for ABAP.

Disclaimer: This is a prototype application primarily designed to guide ABAP developers in building end-to-end applications using the Vertex AI SDK. Its functionality may be limited and not intended for production use.

In today's fast-paced warehouse environments, ensuring the safe handling and management of hazardous materials (HAZMAT) is paramount. HPro, an AI-powered solution, revolutionizes HAZMAT management by providing real-time, actionable insights to warehouse operators and supervisors.

Key Features and Benefits:

- **Natural Language Interface:** Interact with HPro using simple, everyday language to get instant answers about HAZMAT handling, storage, and emergency procedures.
- **Real-Time Information:** Access up-to-date safety data sheets (SDS) and regulatory information, ensuring compliance and accuracy.
- **AI-Powered Pictogram Recognition:** Instantly identify and understand hazard symbols using your device's camera.

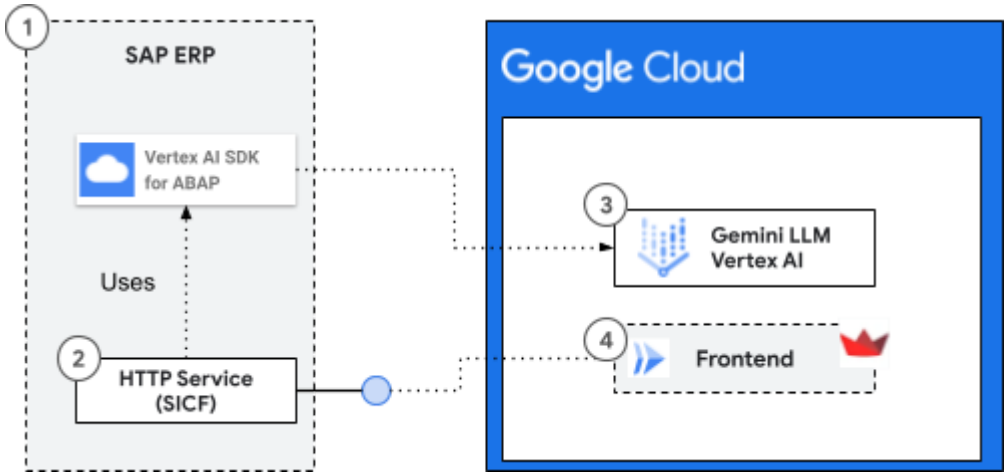
- **Inventory Integration:** Seamlessly connect with your existing inventory management system for real-time stock visibility and material-specific instructions.
- **Emergency Response Guidance:** Receive clear, step-by-step instructions for handling HAZMAT incidents and emergencies.
- **Risk Assessment and Mitigation:** Proactively identify and mitigate potential risks using AI-powered analysis of SDS and operational data.
- **Training and Education:** Enhance HAZMAT knowledge through personalized learning paths, interactive quizzes, and simulations.

Vertex AI SDK: Your Development Toolkit

We'll leverage the powerful Vertex AI SDK to bring HPro to life. This toolkit provides a comprehensive suite of tools and APIs for building, deploying, and managing machine learning models on Google Cloud.

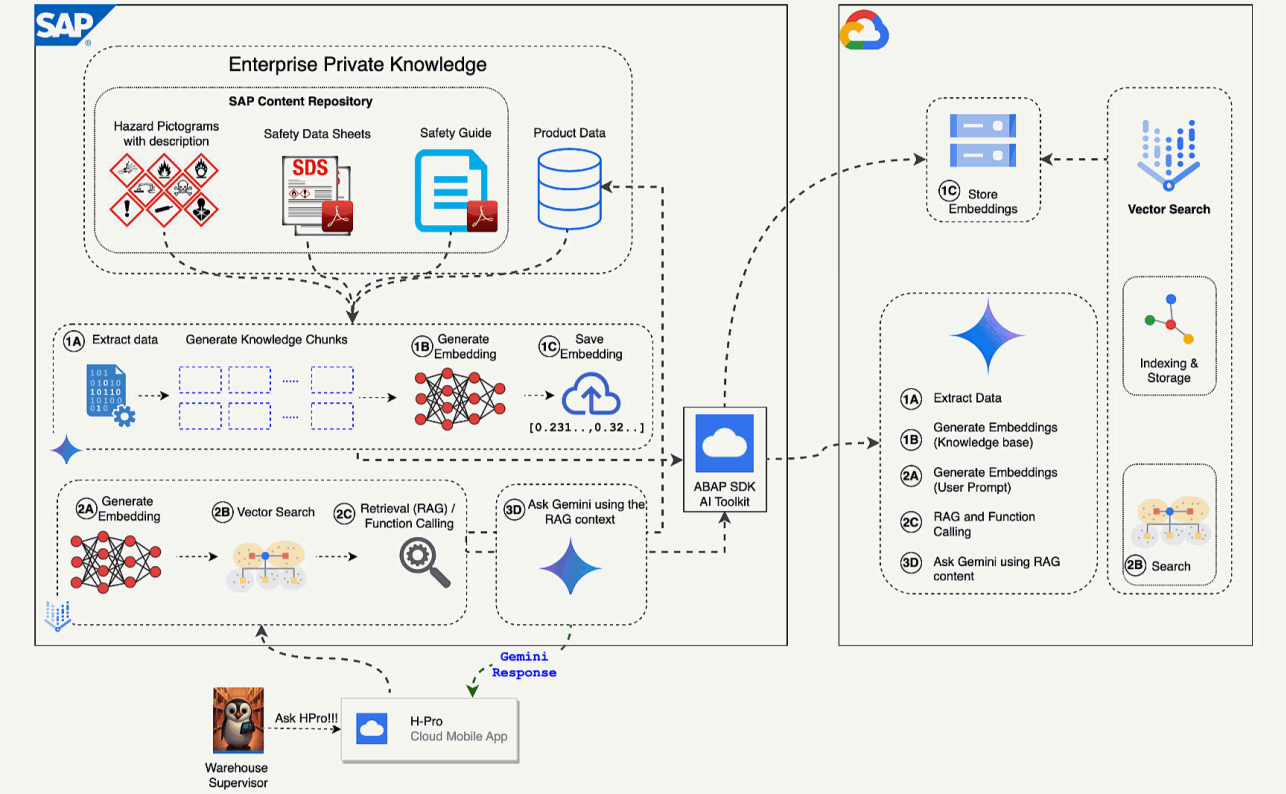
Architecture

The following diagram shows a reference architecture for the application:



Component	Subsystem	Details
1	SAP ERP	SAP ERP system such as S/4HANA, which uses **Vertex AI SDK** to develop Gemini-powered business applications.
2	HTTP Service (SICF)	A HTTP Service which will take the request for any frontend and use Vertex AI SDK for ABAP to integrate with Google Vertex AI. **Note**: Developers can explore alternative integration methods, such as using an OData service, but for simplicity this guide focuses on the ABAP service approach.
3	Gemini	Gemini Flash will be used to process the query and context.
4	Frontend	A lightweight Streamlit application will be deployed on Cloud run to interact with ABAP HTTP service. This will act as an UI for Warehouse workers. **Note**: Developers can explore alternative UI, such as a Fiori app using an OData service.

The following shows the overall flow across the components.



Next Steps: Let's Build!

Ready to embark on this exciting journey? In the following sections, we'll guide you through the step-by-step process of developing your HPro prototype using Vertex AI SDK. From setting up your development environment to deploying your model, you'll gain hands-on experience with the latest AI technologies.

Let's get started!

Chapter 5: Prepare GCP Project and SAP system

	This chapter outlines the prerequisites for starting the HAZMAT project, including setting up a Google Cloud project and an SAP sandbox system.

Before you start, make sure you have the following:

- **A Google Cloud project with billing enabled** or **Sign up for a 90-Day Free Trial of Google Cloud Platform.**
- In the Google Cloud Console, on the project selector page, select or create a Google Cloud project (For example: abap-sdk-poc).
- Please make a note of the project id which is also available in Google Cloud Console, we will use it in future steps.

- **A SAP sandbox system**:
  - **Recommended:** If you don't have one, follow the "Install ABAP Platform Trial on Google Cloud Platform and Install ABAP SDK" codelab to set up a sandbox.
  - **Existing SAP System:** If you already have one, consult the official documentation on setting up the ABAP SDK.

Please note: The instructions in the guide assume you're using the ABAP Platform Trial

(A4H) for development. If you're using a different SAP system, the overall process remains similar, though some specific steps or system configurations might vary.

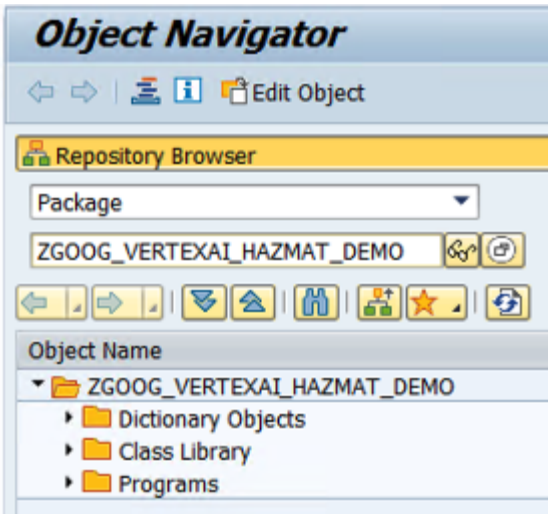
Chapter 6: Clone the HPro repository using ABAPGIT

	This chapter provides instructions on how to clone the HPro repository using ABAPGIT and verify the successful creation of the required package.

Step1: Execute the ABAP program `ZABAPGIT_STANDALONE` using transaction SE38 to clone the **TBD** repository.

TBD screenshot

Step2: Open transaction SE80 (Object Navigator) and check if the package `ZGOOG_VERTEXAI_HAZMAT_DEMO` has been created successfully.



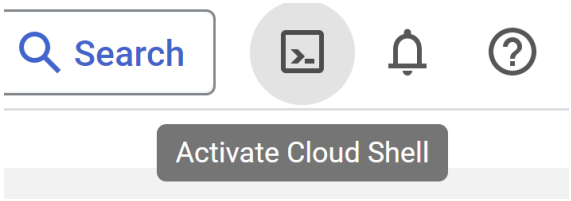
With the HAZMAT repository cloned and the ZGOOG_VERTEXAI_HAZMAT_DEMO package verified, the necessary components are in place to proceed with the next steps.

Chapter 7: ABAP SDK Configuration

	This chapter provides step-by-step instructions on configuring the ABAP SDK for the HAZMAT project, including creating service accounts, assigning roles, configuring client keys, RFC destinations, service mapping, and Vertex AI SDK Model Generation Parameters.

Please proceed with the below steps once you have installed ABAP SDK and configured the trust certificates.

Step1: From the Cloud Console, click Activate Cloud Shell on the top right corner:



Step2: Create Service Account

Run the following command to create a service account. Ignore this step if the service account already exists. Please change the project name if it's different from `abap-sdk-poc`.

```
gcloud config set project abap-sdk-poc
gcloud iam service-accounts create abap-sdk-dev \
        --description="ABAP SDK Dev Account" \
        --display-name="ABAP SDK Dev Account"
```

Step3: Assign the Vertex AI User role to the service account

Run the below command to apply the required IAM roles to the service account. Please change the project name if it's different from `abap-sdk-poc`.

```
gcloud projects add-iam-policy-binding abap-sdk-poc \
  --member "serviceAccount:abap-sdk-dev@abap-sdk-poc.iam.gserviceaccount.com" \
  --role "roles/aiplatform.user" \
  --role "roles/storage.objectAdmin" \
  --role "roles/iam.serviceAccountTokenCreator"
```

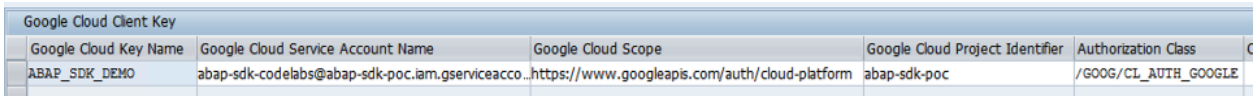
Step4: Configure client key

Log in to the SAP Trial A4H system with the user name DEVELOPER and password ABAPtr2022#00 and follow these steps to configure the client key:

1. In the SAP GUI, enter transaction code **SPRO**.
2. Click **SAP Reference IMG**.
3. Click **ABAP SDK for Google Cloud > Basic Settings > Configure Client Key**.
4. Click **New Entries**.
5. Enter values for the following fields:

Field	Values
Google Cloud Key Name	DEMO_AIPLATFORM
Google Cloud Service Account Name	abap-sdk-dev@abap-sdk-poc.iam.gserviceaccount.com
Google Cloud Scope	https://www.googleapis.com/auth/cloud-platform
Google Cloud Project Identifier	abap-sdk-poc
Authorization Class	/GOOG/CL_AUTH_GOOGLE.

** Leave the other fields blank*



Google Cloud Client Key

Google Cloud Key Name	Google Cloud Service Account Name	Google Cloud Scope	Google Cloud Project Identifier	Authorization Class
ABAP_SDK_DEMO	abap-sdk-codelabs@abap-sdk-poc.iam.gserviceacco..	https://www.googleapis.com/auth/cloud-platform	abap-sdk-poc	/GOOG/CL_AUTH_GOOGLE

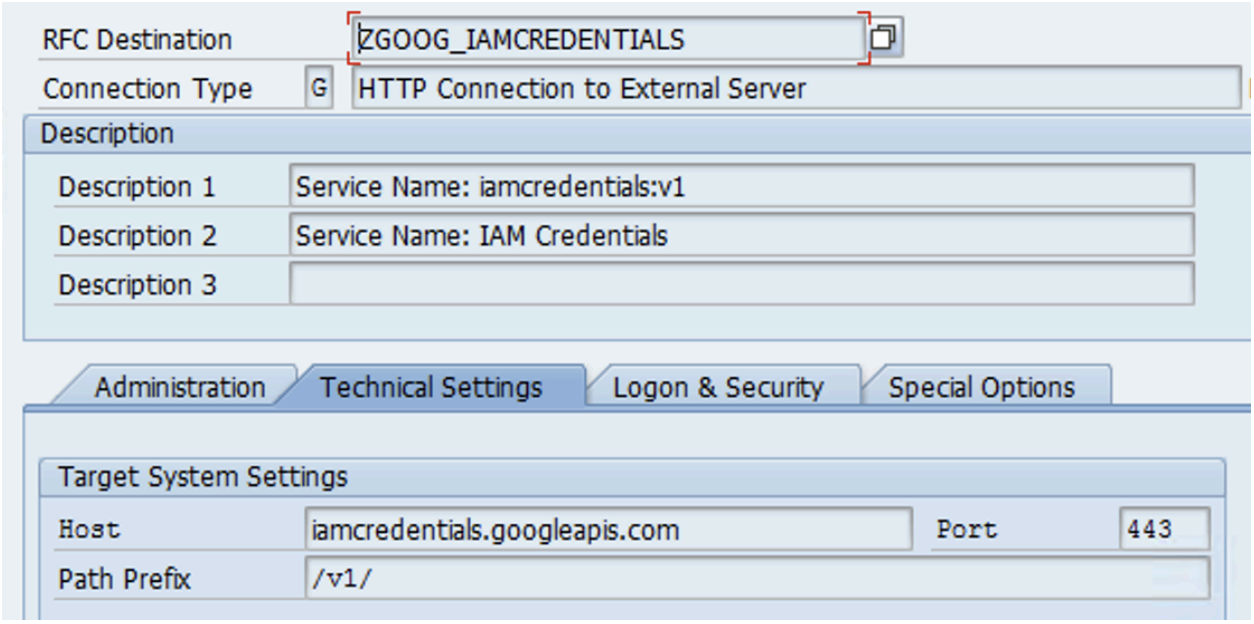
Note: The authentication class will and configuration will change if your SAP system is not deployed on Google Cloud. Please refer to this guide for different authentication options.

Step5: Create RFC destination

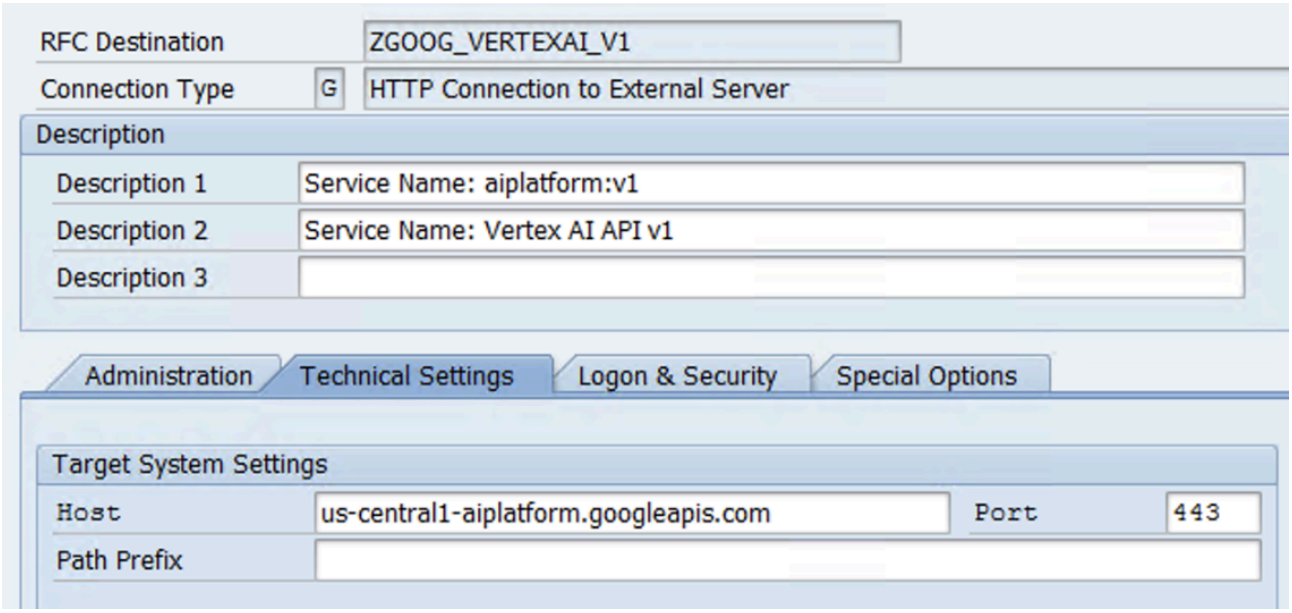
Create RFC destination for IAM credential and Vertex AI API using the transaction code SM59. If needed, please refer here for detailed steps on creating an RFC destination.

RFC destination name	Target host (API endpoint)	Notes
ZGOOG_IAMCREDENTIALS	• Host: iamcredentials.googleapis.com • Path: Prefix: /v1/ • Port: 443 • SSL: Active	This RFC destination targets the IAM API.
ZGOOG_VERTEX_AI	• Host: us-central1-aiplatform.googleapis.com • Port: 443 • SSL: ACTIVE	This RFC destination targets Vertex AI API us-central1 endpoint.

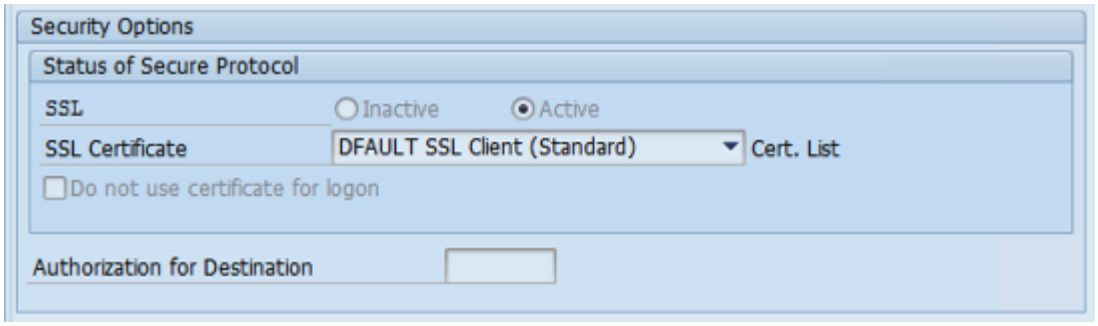
- Under the Technical Settings tab, enter the following details for the ZGOOG_IAMCREDENTIALS destination.



- Under the **Technical Settings** tab, enter the following details for the ZGOOG_VERTEX_AI destination.



- For the **SSL Certificate** field, make sure that the option **DEFAULT SSL Client (Standard)** is selected for both the RFC destinations.



Step6: Configure service mapping

To configure the service mapping table for IAM API, and Vertex AI API, perform the following steps:

1. In the SAP GUI, enter transaction code **SPRO**.
2. Click **SAP Reference IMG**.
3. Click **ABAP SDK for Google Cloud > Basic Settings > Configure Service Mapping**.
4. Click **New Entries** for IAM Credential and Vertex AI API and update the RFC destinations as shown below.

Field	Record 1 (IAM Credential)	Record 2 (Vertex AI)
Google Cloud Key Name	DEMO_AIPLATFORM	DEMO_AIPLATFORM
Google Service Name	iamcredentials:v1	aiplatform:v1
RFC Destination	ZGOOG_IAMCREDENTIALS	ZGOOG_VERTEX_AI

Step7: Configure Vertex AI SDK Model Generation Parameters

To configure the generation parameter for models related to Text embedding, Multimodal embeddings and Gemini-flash, perform the following

1. In the SAP GUI, enter transaction code **SPRO**.

2. Click **SAP Reference IMG**.
3. Click **ABAP SDK for Google Cloud > Basic Settings > Vertex AI SDK: Configure Model Generation Parameters**.
4. Click **New Entries** for model configurations and update the entries as shown below.

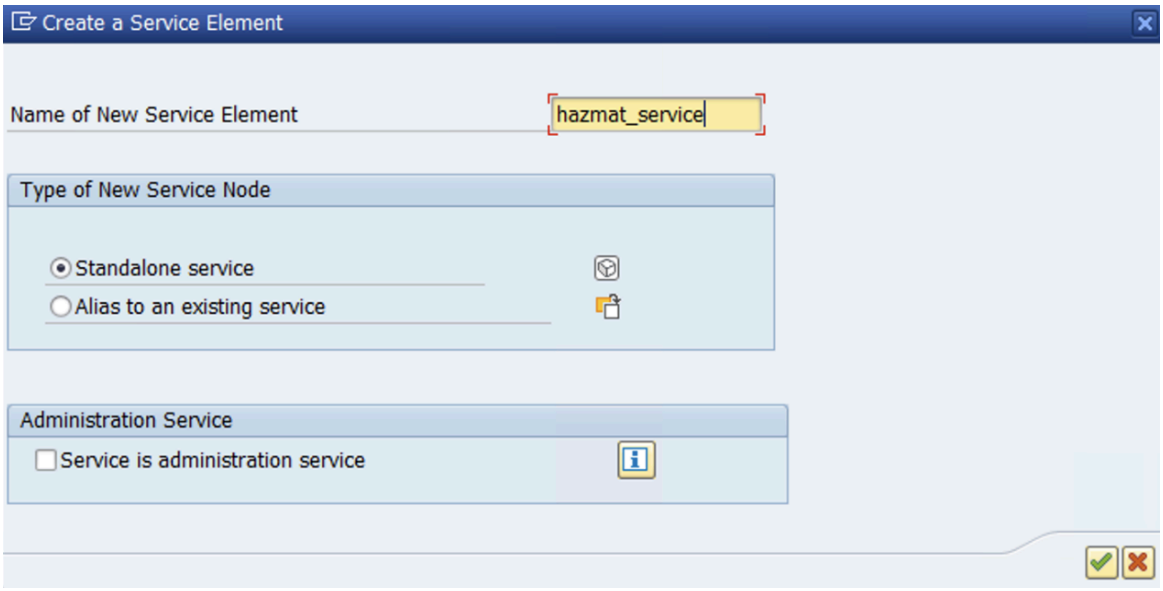
Field	Record 1 (Text Embeddings)	Record 2 (Multi Modal embeddings)	Record 3 (Gemini-Flash)	Record 4 (Gemini-Pro-Vision)
Model Key	Text-Embeddings	Multimodal-Embedding	Gemini-Flash	Gemini-Pro-Vision
Model ID	text-embedding-004	multimodalembedding@001	gemini-1.5-flash-001	gemini-1.0-pro-vision-001
Google Cloud Key Name	DEMO_AIPLATFORM	DEMO_AIPLATFORM	DEMO_AIPLATFORM	DEMO_AIPLATFORM
Google Cloud Region Location ID	us-central1	us-central1	us-central1	us-central1
Publisher ID for LLM	google	google	google	google

* Leave blank for remaining attributes

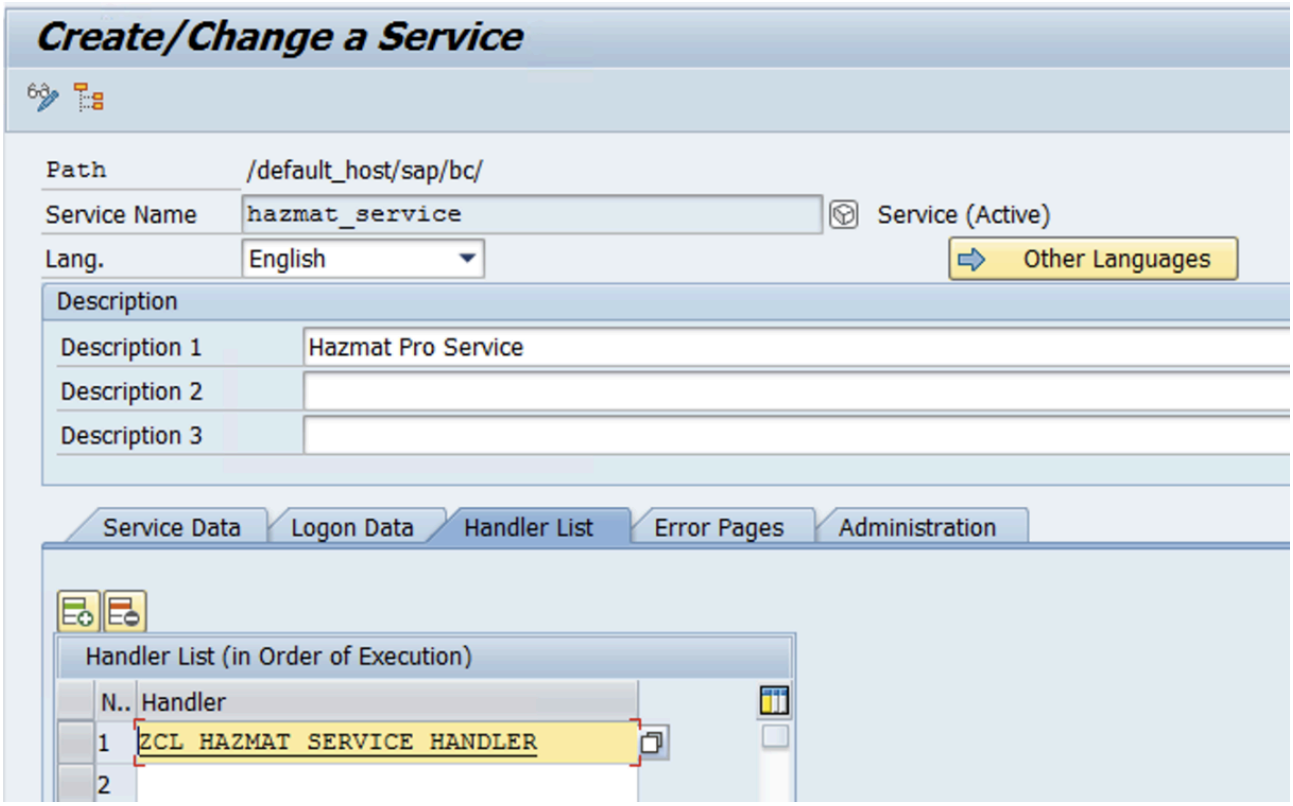
Step8: Create a SICF node for HPro service

To create a SICF node and configure the handler class, perform the following

1. In the SAP GUI, enter transaction code **SICF** and click **Execute(F8)**.
2. Goto **default_host > SAP > bc**. Select **bc** and click on the button 'Create Host/Service' and create service with name 'hazmat_service'



3. Configure the Handler class as ZCL_HAZMAT_SERVICE_HANDLER



Chapter 8: Prepare Enterprise Data

	This chapter guides you through preparing the necessary enterprise data for the HAZMAT Pro prototype, including setting up unstructured data in Google Cloud Storage and loading sample product information into your SAP system.

TBD-diagram

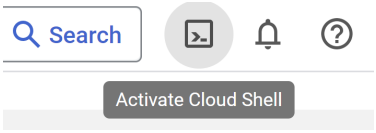
Data Preparation: Unstructured Data for Simulation

For HAZMAT Pro we will be using the following unstructured data to simulate enterprise

- A Safety Data Sheet (SDS) is a document that provides detailed information about the hazards of a chemical and how to handle it safely.
- A Hazard Pictogram is a graphical symbol that provides a quick visual warning about the potential hazards of a chemical or substance.
- A Warehouse Safety Guide(WSG) is a document that outlines the procedures and best practices for maintaining a safe working environment in a warehouse.

Run the following command in the Google Cloud Shell to set up the data requirement.

From the Cloud Console, click Activate Cloud Shell on the top right corner:



Ensure that the command is executed with a unique id e.g.: <unique_id> which will be used to name the bucket.

```
wget
https://raw.githubusercontent.com/google-cloud-abap/ai4sap/main/setup-scripts/setup-data-files.sh
chmod 755 setup-data-files.sh
./setup-data-files.sh <project_id> https://github.com/google-cloud-abap/ai4sap.git
<unique_id> us-central1
```

The script will perform these actions:

1. **Clone the GitHub repository:** Creates a local copy of the specified repository on your machine.
2. **Create a unique bucket:** Generates a Google Cloud Storage bucket named `hazmat-data-files-<unique-id>`, where you provide the `<unique-id>` when running the script.
3. **Upload data to the bucket:** Copies the contents of specific folders from the cloned repository into corresponding folders within the newly created bucket.

Here is a quick description of the bucket contents.

Folder name	What do they have?
hazmat-sds	Safety Data Sheets (SDS) for various hazardous materials (PDF format)
hazmat-wsg	Warehouse Safety Guide (PDF format)
hazmat-pictogram	Collection of Hazard Pictogram images
hazmat-pictogram-descriptions	Enterprise description for Pictograms
hazmat-prod	Sample material data in CSV format
hazmat-sds-chunks	Safety Data Sheets (SDS) knowledge chunks
hazmat-wsg-chunks	Warehouse Safety Guide knowledge chunks
hazmat-sds-embeddings	Pre-computed embeddings for SDS knowledge chunks
hazmat-wsg-embeddings	Pre-computed embeddings for the Warehouse Safety Guide

hazmat-pictogram-embeddings	Pre-computed embeddings for Hazard Pictograms
hazmat-prod-embeddings	Pre-computed embeddings for product data
hazmat-prompts	Pre-created prompts repository

Data Preparation: Adding Product Information to SAP

The SAP sandbox environment lacks the necessary hazardous material data for our purposes. To simulate integration with SAP and demonstrate real-time inventory retrieval using Gemini's functional calling feature, we'll load sample material data into the `ZHAZMAT_PRODUCTS` table, which is already present within the `ZGOOG_VERTEXAI_HAZMAT_DEMO` package.

Follow these steps:

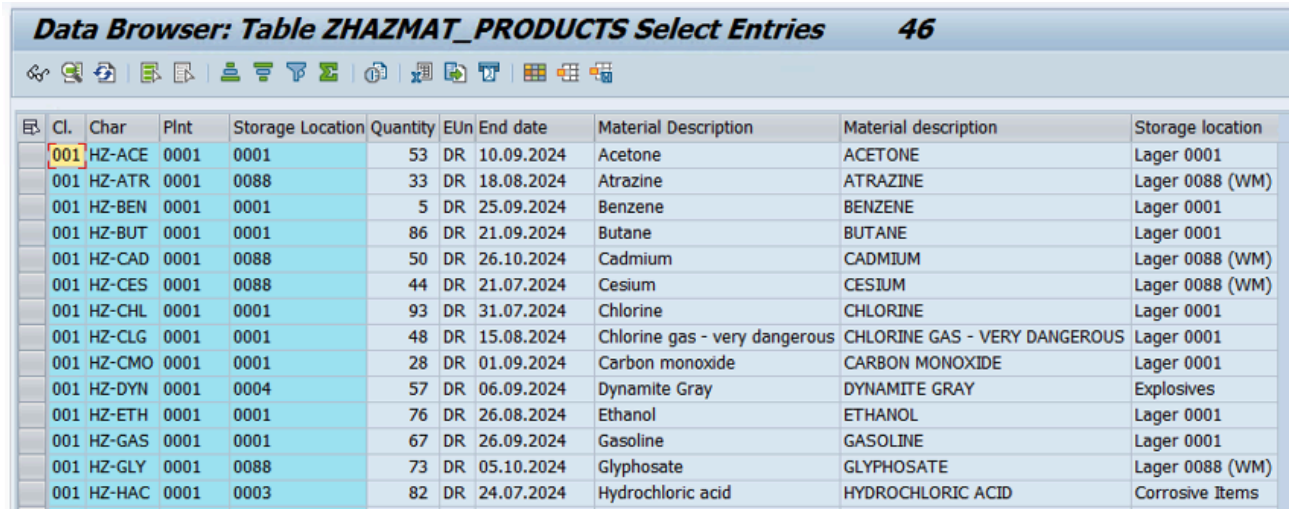
Step1: Download Sample Data: Obtain the `product.csv` file from the designated GitHub folder (link to be provided) and save it to your desktop.

	MATNR	WERKS	LGORT	ERFMG	ERFME	EXPDT	MAKTX	MAKTG	LGOBE
2	HZ-ACE	0001	0001	53	DR	10.09.2024	Acetone	ACETONE	Lager 0001
3	HZ-ATR	0001	0088	33	DR	18.08.2024	Atrazine	ATRAZINE	Lager 0088 (WM)
4	HZ-BEN	0001	0001	5	DR	25.09.2024	Benzene	BENZENE	Lager 0001
5	HZ-BUT	0001	0001	86	DR	21.09.2024	Butane	BUTANE	Lager 0001
6	HZ-CAD	0001	0088	50	DR	26.10.2024	Cadmium	CADMIUM	Lager 0088 (WM)
7	HZ-CES	0001	0088	44	DR	21.07.2024	Cesium	CESIUM	Lager 0088 (WM)
8	HZ-CHL	0001	0001	93	DR	31.07.2024	Chlorine	CHLORINE	Lager 0001

Step2: Load Data into SAP: Execute the ABAP program `ZR_HAZMAT_LOAD_PRODUCT_DATA` using transaction SE38. Select the downloaded `product.csv` file to initiate the data loading process.

Step3: Verify Data Load: Access the `ZHAZMAT_PRODUCTS` table using transaction SE16.

Confirm that the sample product records have been successfully loaded.

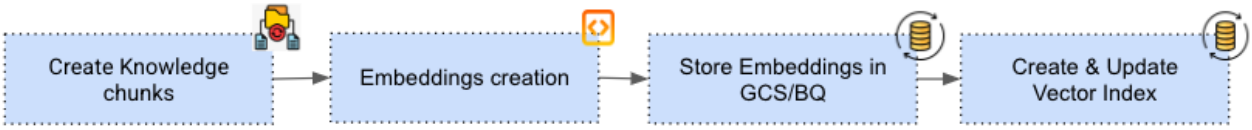


Cl.	Char	Plnt	Storage Location	Quantity	EUn	End date	Material Description	Material description	Storage location
001	HZ-ACE	0001	0001	53	DR	10.09.2024	Acetone	ACETONE	Lager 0001
001	HZ-ATR	0001	0088	33	DR	18.08.2024	Atrazine	ATRAZINE	Lager 0088 (WM)
001	HZ-BEN	0001	0001	5	DR	25.09.2024	Benzene	BENZENE	Lager 0001
001	HZ-BUT	0001	0001	86	DR	21.09.2024	Butane	BUTANE	Lager 0001
001	HZ-CAD	0001	0088	50	DR	26.10.2024	Cadmium	CADMIUM	Lager 0088 (WM)
001	HZ-CES	0001	0088	44	DR	21.07.2024	Cesium	CESIUM	Lager 0088 (WM)
001	HZ-CHL	0001	0001	93	DR	31.07.2024	Chlorine	CHLORINE	Lager 0001
001	HZ-CLG	0001	0001	48	DR	15.08.2024	Chlorine gas - very dangerous	CHLORINE GAS - VERY DANGEROUS	Lager 0001
001	HZ-CMO	0001	0001	28	DR	01.09.2024	Carbon monoxide	CARBON MONOXIDE	Lager 0001
001	HZ-DYN	0001	0004	57	DR	06.09.2024	Dynamite Gray	DYNAMITE GRAY	Explosives
001	HZ-ETH	0001	0001	76	DR	26.08.2024	Ethanol	ETHANOL	Lager 0001
001	HZ-GAS	0001	0001	67	DR	26.09.2024	Gasoline	GASOLINE	Lager 0001
001	HZ-GLY	0001	0088	73	DR	05.10.2024	Glyphosate	GLYPHOSATE	Lager 0088 (WM)
001	HZ-HAC	0001	0003	82	DR	24.07.2024	Hydrochloric acid	HYDROCHLORIC ACID	Corrosive Items

Chapter 9: Importance of Data Ingestion Pipeline and Vector Index

	This chapter emphasizes the importance of data ingestion pipelines and knowledge chunks, even with large language models, highlighting their role in cost efficiency, relevance, customization, and data freshness.

Let's quickly talk about the Importance of Data Ingestion Pipelines & Knowledge Chunks, Even with Large Context Models.



While powerful models like Gemini 1.5 Pro with their vast context window offer the tempting allure of bypassing RAG based solutions entirely - providing a rapid path to harnessing LLM potential - establishing a dedicated Data Ingestion Pipeline and structuring your data into Knowledge Chunks remains essential for several key reasons:

1. Cost Efficiency:

- Processing large volumes of data directly through a model like Gemini can be computationally expensive.
- A data ingestion pipeline preprocesses and optimizes your data, reducing the amount of information the model needs to handle, leading to significant cost savings.

2. Relevance and Performance:

- Enterprise (unstructured) data is often vast and contains a lot of irrelevant information for specific queries.
- A pipeline can filter, clean, and structure your data, ensuring only relevant knowledge chunks are presented to the model, improving response relevance and performance.

3. Customization and Control:

- A pipeline lets you tailor data processing to your specific use case, applying domain-specific transformations and enriching data with metadata.
- This granularity allows you to optimize the model's output for your specific requirements, which is impossible when feeding raw data directly into a large model.

4. Data Freshness and Updates:

- An ingestion pipeline enables you to keep your knowledge base up-to-date by incorporating new data and updates efficiently.
- This ensures that your AI system remains relevant and provides accurate responses based on the latest information.

What are Knowledge Chunks?

- Knowledge chunks are smaller, semantically meaningful units of information extracted from your enterprise data.
- These chunks can be paragraphs, sentences, or even phrases, depending on the nature of your data and use case.

- By dividing data into chunks, you enable the model to focus on relevant information and avoid processing the entire dataset for each query.
- Each chunk is then converted into a numerical representation called an **embedding** using advanced language models. These embeddings capture the semantic meaning of the text, allowing for efficient comparison and retrieval based on similarity.
- These embeddings are then organized into a **vector index**, a specialized data structure optimized for similarity search. This enables rapid retrieval of the most relevant knowledge chunks given a user query, significantly enhancing the performance and responsiveness of your AI system.

Best Practices for Data Ingestion Pipelines (Cheatsheet):

1. **Clean & Prep Data:** Remove noise, standardize formats, and fix errors.
2. **Transform & Enrich:** Extract key info, add metadata for context.
3. **Chunk Strategically:** Divide data into meaningful units (paragraphs, sentences).
4. **Manage Metadata:** Track source, date, etc., for better search & filtering.
5. **Store & Index:** Use suitable storage & indexing for quick retrieval.
6. **Monitor & Maintain:** Regularly check data quality & pipeline health.

Conclusion:

Investing in a Data Ingestion Pipeline and creating Knowledge Chunks provides significant benefits in terms of cost efficiency, relevance, customization, and maintainability, even when using powerful large language models. The Vertex AI SDK for ABAP further empowers developers to build ABAP-centric components that contribute to the

enterprise data pipeline, efficiently handling both structured and unstructured SAP-specific data. By following best practices, you can create a robust pipeline that maximizes the value of your enterprise data and enables your AI systems to deliver accurate and insightful responses.

Chapter 10: Intelligent Knowledge Chunking

	This chapter explains how to create knowledge chunks from enterprise documents using Gemini's capabilities and the Vertex AI SDK, simplifying the document chunking process for developers.

TBD-diagram

There are several ways to create knowledge chunks from enterprise documents like PDFs. For example, you can use Python libraries like **PyMuPDF, Langchain, or Tiktoken** to extract text and split it into manageable chunks. Alternatively, you can leverage **Google Cloud Document AI** to automatically process and analyze the documents, generating structured data and identifying key entities and concepts within the text.

Given that the Safety Data Sheets (SDSs) will comprise sixteen sections, and the formatting of these sections will vary across different manufacturers, this prototype leverages Gemini's capabilities to efficiently generate knowledge chunks for SDSs. You can refer to the `ZCL_HAZMAT_GCS_DATA_INGESTER` class and the `SPLIT_WSG_IN_CHUNKS` method, where we craft a specific prompt to instruct Gemini to split the SDS document into sections and return a well-structured JSON array.

This prompt includes:

- **Clear instructions:** Gemini is explicitly told to identify all section headings, match them to a predefined list, extract the content of each matched section, handle any unmatched sections, and present the extracted information in a structured format sorted by section ID.
- **Predefined section headings:** A list of possible section headings (with

corresponding IDs) is provided to guide Gemini's extraction process.

- **Desired output format:** The prompt specifies that the results should be in JSON format, with a specific structure including the chemical name and an array of section details (ID, header, and content).

This approach demonstrates how we're using Gemini's powerful language understanding and generation capabilities to automate the knowledge chunking process within our prototype.

Furthermore, the SDK simplifies the process for developers by providing methods like `SET_FILE_DATA`, allowing them to directly point to files stored in cloud storage. The SDK then handles the transmission of the file to Gemini along with the splitting instructions.

```
...->set_file_data( iv_mime_type = 'application/pdf' iv_file_uri = iv_file_gcs_uri)
```

This streamlined approach significantly reduces the complexity of integrating document chunking into applications.

Example code for splitting using Gemini:

```
. . .
        lv_instruction =  'You will receive a document containing sections with
headings. These sections correspond to a' &&
                          ' predefined list of possible section headings (Section ID:
Section Header):' &&
                          '"1": "Identification"' &&
                          '"2": "Hazard identification"' &&
                          '"3": "Composition / information on ingredients"' &&
                          '"4": "First-aid measures"' &&
                          '"5": "Fire-fighting measures"' &&
                          '"6": "Accidental release measures"' &&
                          '"7": "Handling and storage"' &&
                          '"8": "Exposure controls / personal protection"' &&
```

```
                          '"9": "Physical and chemical properties"' &&
                          '"10": "Stability and reactivity"' &&
                          '"11": "Toxicological information"' &&
                          '"12": "Ecological information"' &&
                          '"13": "Disposal considerations"' &&
                          '"14": "Transport information"' &&
                          '"15": "Regulatory information"' &&
                          '"16": "Other information"' &&

                          'task is to:' &&
                          'all section headings within the document.' &&
                          'each identified heading to the predefined list.' &&
                          'the content within each matched section.' &&
                          'any sections not present in the predefined list.' &&
                          'a structured representation of the extracted information.'
&&
                          'the result sorted by section id.'.


. . .

        lv_prompt = 'Analyze the following document and extract section-specific
content:' &&
                    'Output the results in JSON format with the following structure:'
&&
                    'JSON' &&
                    '{' &&
                    '"CHEMICAL_NAME": "[Name of the chemical extracted from the
document]",' &&
                    '"SECTION_DETAILS": [' &&
                    '{' &&
                    '"SECTION_ID": "[ID of the section from the predefined list]",'
&&
                    '"SECTION_HEADER": "[Matched section header from the predefined
list]",' &&
                    '"SECTION_CONTENT": "[Extracted content from the section, If the
document has more than 8 pages, limit this section length to maximum of 75 tokens]"'
&&
                    '},' &&
                    '// ... (Repeat for each extracted section)' &&
                    ']' &&
                    '}' &&
                    'no format or pretty print the json data'.

        DATA(lv_response) = CONV string( mo_model->clear_file_data(
                                        )->set_file_data( iv_mime_type          =
'application/pdf'
```

```
                                                    iv_file_uri           =
iv_file_gcs_uri
                                    )->generate_content( lv_prompt
                                    )->get_text( ) ).
. . .
```

Similar approach can be used to create knowledge chunks for other Enterprise documents like the Warehouse Safety Guide. You can refer to the `ZCL_HPRO_INGEST_DOCUMENTS` class and the `SPLIT_WSG_IN_CHUNKS` method.

Chapter 11: Transforming Knowledge into Searchable Vectors

	This chapter explains how to transform knowledge chunks into searchable numerical representations (embeddings) for both text and images using the Vertex AI SDK for ABAP, which simplifies the integration of Vertex AI and offers flexibility in storing the embeddings.

After breaking down your knowledge base into manageable chunks, the next step is to make this information easily searchable. This involves creating embeddings – numerical representations of each chunk – which will be used to build a vector index in subsequent stages.

This prototype utilizes the `text-embedding-004` model from Vertex AI to transform textual data into numerical vectors. This captures the essence of the information in a machine-readable format. However, for pictograms, we use the `multimodalembedding@001` model to generate embeddings that capture visual information.

Leveraging the Vertex AI SDK for ABAP simplifies this process significantly. With just a few lines of code, you can generate embeddings for your knowledge chunks. The SDK also provides pre-built ABAP structures to create JSON Lines (JSONL) files, which are essential for fine-tuning Large Language Models (LLMs). These built-in features streamline the integration of Vertex AI within your ABAP environment.

For a practical example of embedding generation:

- **Safety Data Sheets (SDS):** Refer to the `ZCL_HPRO_INGEST_DOCUMENTS` class and the `CREATE_SDS_EMBED_SEND_TO_GCS` method.

- **Warehouse Safety Guides:** See the `CREATE_WSG_EMBED_SEND_TO_GCS` method within the same class.
- **SAP Product data:** Explore the `CREATE_PROD_EMBED_SEND_TO_GCS` method.
- **Pictograms:** The `CREATE_PICT_EMBED_SEND_TO_GCS` method provides a clear example of how to generate image embeddings using the SDK's simplified approach.

Following is an example code for SDS:

```
. . .
    LOOP AT mt_embedding_data ASSIGNING FIELD-SYMBOL(<ls_emdedding>).
      "Additional optional parameters
      ls_addln_params-task_type = /goog/cl_embeddings_model=>c_retrieval_document.
      ls_addln_params-title = |{ <ls_emdedding>-matnr }:{ <ls_emdedding>-section_id
}:{ <ls_emdedding>-section_header }|.

      TRY.
          DATA(lo_client) = NEW /goog/cl_embeddings_model( "iv_key_name  = mv_ai_key
                                                           iv_model_key =
'Text-Embeddings' ).

          CLEAR: ls_embedding_template.
          ls_embedding_template-id = <ls_emdedding>-guid.
          ls_embedding_template-content = <ls_emdedding>-revised_content.
          ls_embedding_template-source =  'SAP-DOC-ZCL_HPRO_INGEST_DOCUMENTS'.

          GET TIME STAMP FIELD ls_embedding_template-feature_timestamp.

          "Create embedding with template record
          lo_client->gen_text_embeddings_by_struct( is_input        =
ls_embedding_template
                                                    is_addln_params = ls_addln_params
).

          DATA(lv_msg) = |Product: { <ls_emdedding>-matnr }, LGORT: {
<ls_emdedding>-lgort }, Section: { <ls_emdedding>-section_id } sent to GCS!|.
          MESSAGE lv_msg TYPE 'I'.
          DATA(lv_filename) = |{ <ls_emdedding>-guid }.json|.
          lo_client->send_struct_to_gcs( iv_bucket_name = mv_tgt_bucket_name
iv_file_name = lv_filename ).
```

```
        CATCH /goog/cx_sdk INTO DATA(lo_exception).
          ev_err_text = lo_exception->get_text( ).
          lv_msg = |Product: { <ls_emdedding>-matnr }, LGORT: { <ls_emdedding>-lgort
}, Section: { <ls_emdedding>-section_id } failed:{ ev_err_text }|.
          MESSAGE lv_msg TYPE 'S' DISPLAY LIKE 'E'.
      ENDTRY.
    ENDLOOP.
. . .
```

Furthermore, the SDK simplifies the transfer of these embeddings to various destinations for downstream processing. Methods like `SEND_STRUCT_TO_GCS`, `SEND_STRUCT_TO_BQ`, and `SEND_STRUCT_TO_PUBSUB` enable you to seamlessly store the generated embeddings in your preferred target, such as Google Cloud Storage, BigQuery, or even send them to Pub/Sub for further integrations. This flexibility allows for a smooth and adaptable workflow within your existing data infrastructure.

Chapter 12: Create and Update Vector Index

	This chapter guides you through creating and configuring four vector indexes to power semantic search within the HAZMAT application's RAG workflow.

Given that we have the embeddings available, we can progress to the subsequent phase of Vector Index creation. For a thorough understanding of the various configuration options available for index management, please refer to this guide.

For the HAZMAT application we will create the following 4 indexes.

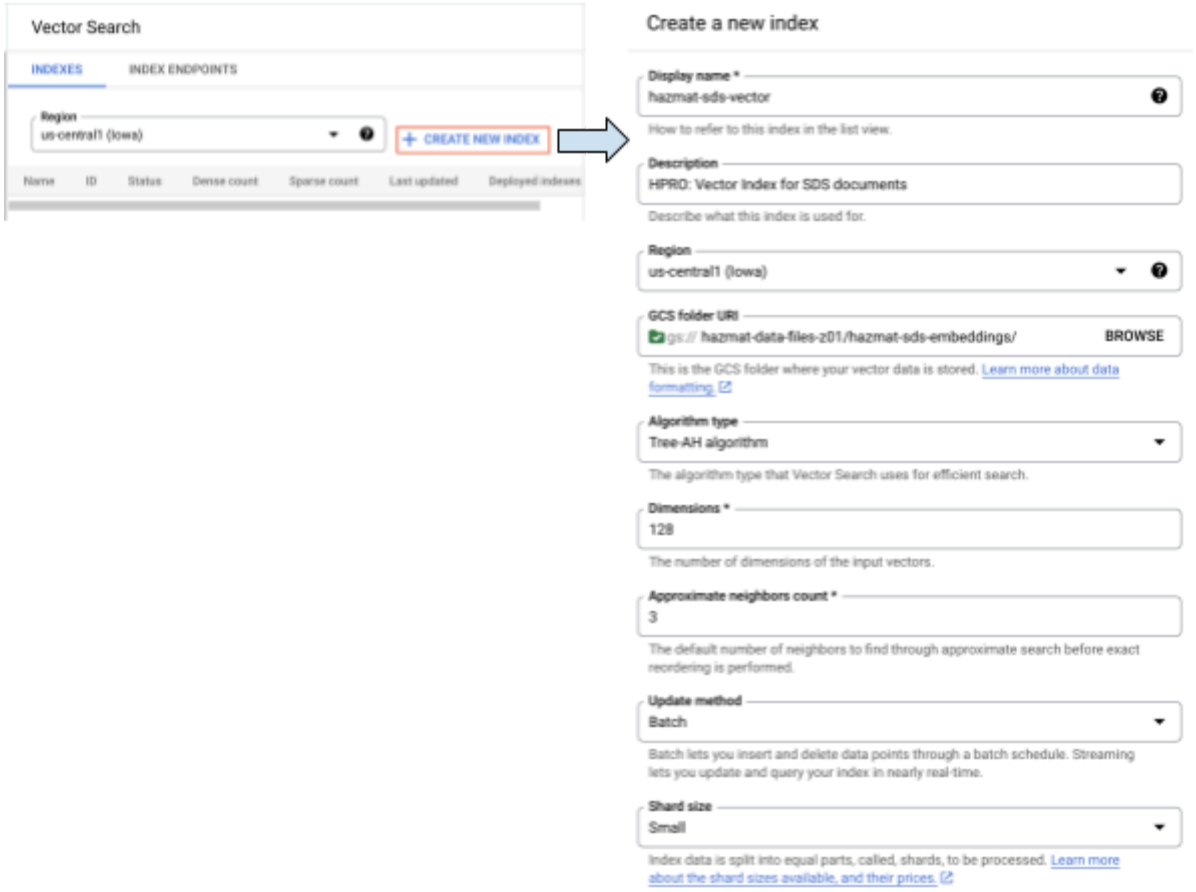
Index Name	Descriptions
hazmat-sds-vector	Index for SDS knowledge chunks
hazmat-wsg-vector	Index for Warehouse Safety Guide knowledge chunks

hazmat-pictogram-vector	Index for hazardous material pictograms
hazmat-prod-vector	Index for SAP materials data

Use the below instructions to create the index `hazmat-sds-vector` in the Google Cloud console:

Step1: Goto Google Cloud console and search for Vertex Search or click on this link.

Step2: Select region as 'us-central1' and click on 'Create new index' to create the following 5 indexes with the listed values.



Index attributes:

Refer to the below table for attribute values:

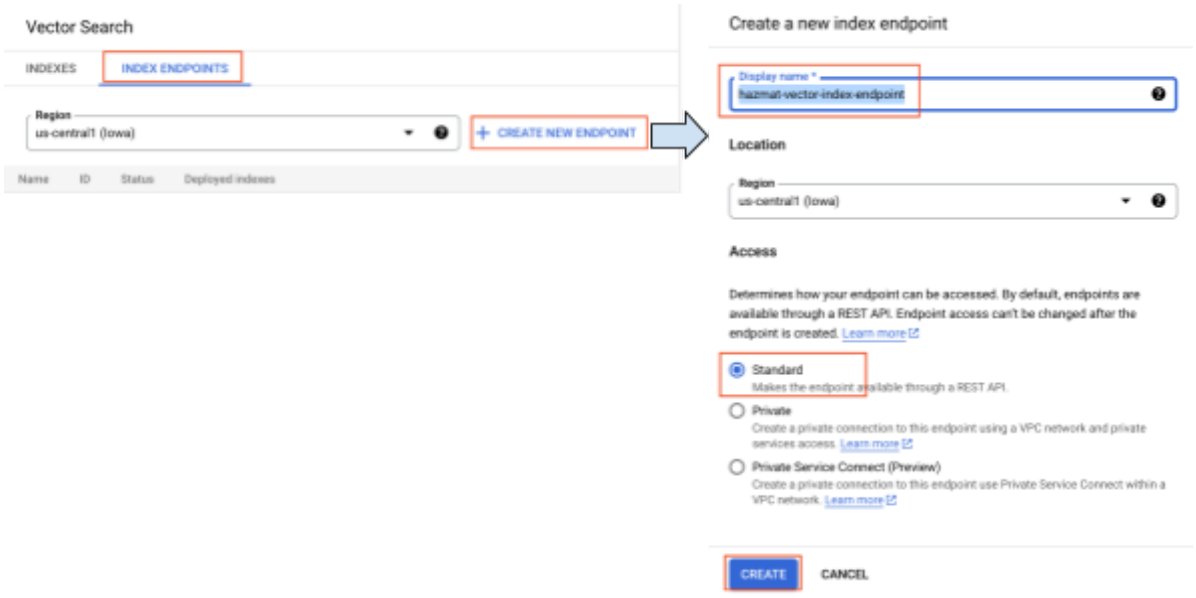
	Index 1	**Index 2**	**Index 3**	**Index 4**
Display Name	hazmat-sds-vector	hazmat-wsg-vector	hazmat-pictogram-vector	hazmat-prod-vector
Description	HPRO: Vector Index for SDS documents	HPRO: Vector Index for Warehouse safety	HPRO: Vector Index for Pictograms	HPRO: Vector Index for Products

		guide		
Region	us-central1	us-central1	us-central1	us-central1
GCS folder URI (Select folder)	hazmat-sds-embeddings	hazmat-wsg-embeddings	hazmat-pictogram-embeddings	hazmat-prod-embeddings
Algorithm Type	Tree-AH algorithm	Tree-AH algorithm	Tree-AH algorithm	Tree-AH algorithm
Dimension	768	768	1408	768
Approximate neighbors count	3	3	3	3
Update method	Batch	Batch	Batch	Batch
Shard size	Small	Small	Small	Small

Step3: Create an Index Endpoint to deploy the index for serving.

An index endpoint in Google Vector is a server that accepts query requests for an index. Multiple indexes can be deployed to the same index endpoint.

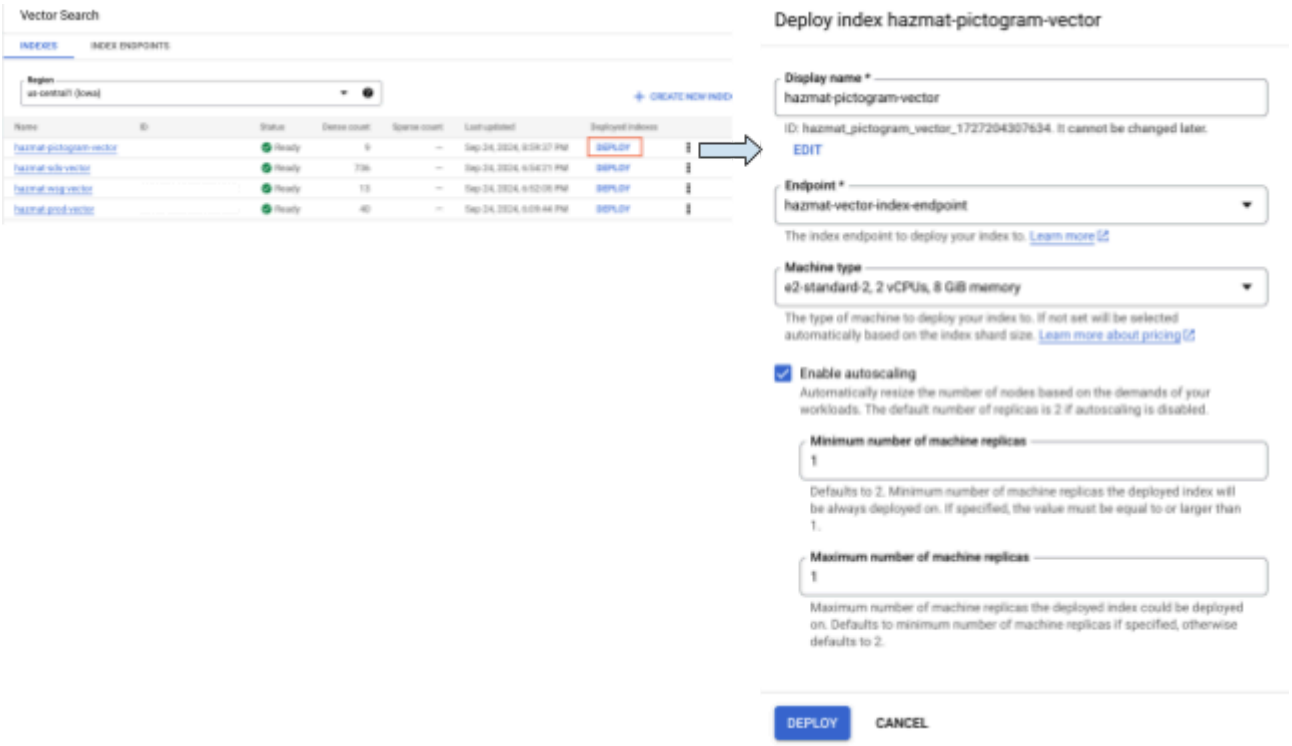
Keep the region as ‘us-central1’ and click on ‘Create new endpoint’. Enter the display name as ‘hazmat-vector-index-endpoint’ and access as ‘Standard’ and click Create.



Step4: Deploy Indexes to the Endpoint

Select the index created earlier and deploy **all 4 indexes** to the same endpoint, with the following values:

- **Display name**: Keep *same as Index name*
- **Machine type**: e2-standard-2
- **Enable autoscaling**: True
- **Minimum number of machine replicas**: 1
- **Maximum number of machine replicas**: 1



The deployment will take a few hours to complete. Once deployed each index will be associated with a unique deployment id, which we will use to configure in SAP using SDK in the later chapters.

Conclusion:

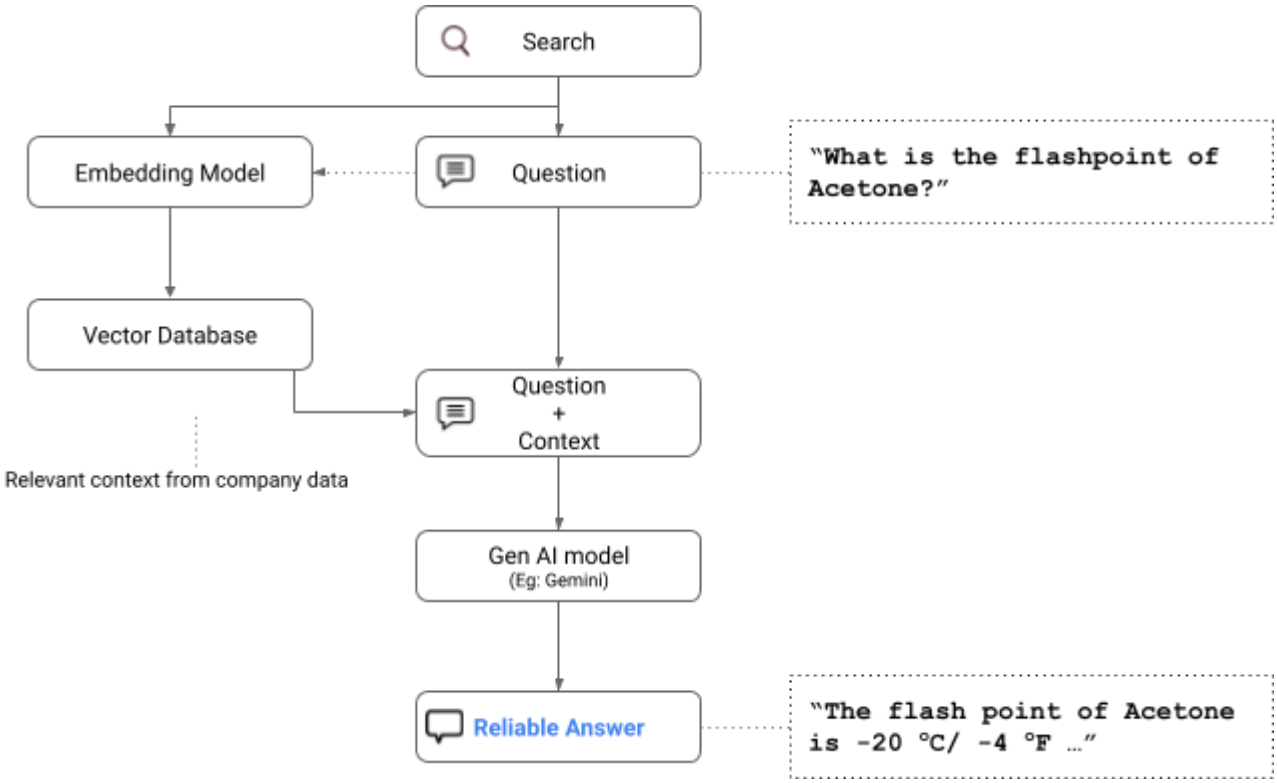
The HAZMAT application now has a fully operational vector search infrastructure. The next step is to integrate this with the RAG workflow, enabling the application to dynamically retrieve contextually relevant information from the knowledge base and generate more informed and accurate responses.

Chapter 13: RAG Serving workflow Overview

	This chapter dives into the architecture and components of a Retrieval Augmented Generation (RAG) serving workflow, providing a comprehensive understanding of how to leverage this powerful approach for enhanced LLM interactions.

What is a RAG Serving Workflow?

The following diagram illustrates the key stages and elements involved:



Retrieval Augmented Generation (RAG) is a framework that enhances Large Language Models (LLMs) by providing them with relevant external knowledge. This is crucial because LLMs, while powerful, have limitations:

- **Knowledge Cut-off:** LLMs are trained on a fixed dataset and may not have access to the latest information.
- **Hallucination:** LLMs can sometimes generate incorrect or nonsensical outputs, especially when faced with ambiguous queries or unfamiliar topics.

RAG addresses these limitations by retrieving relevant context from external knowledge sources (like your vector database) and feeding it to the LLM along with the user query. This allows the LLM to generate more accurate, informed, and contextually relevant responses.

Components of a RAG Serving Workflow

1. **Search:**
    - This stage involves receiving a user query and initiating the search process.
    - It often includes pre-processing steps like cleaning and formatting the query for optimal retrieval.
    - In HPro, this could be a user asking a question about a hazardous material or searching for safety procedures.
2. **Embedding Model:**
    - The query is converted into a numerical vector representation (embedding) using an embedding model.
    - This embedding captures the semantic meaning of the query.
    - HPro uses the `text-embedding-004` model for text and `multimodalembedding@001` for images (pictograms).
3. **Vector Database:**

- The embedding is used to search a vector database containing pre-computed embeddings of your knowledge base. For HPro
- The database utilizes similarity search to find the most relevant documents or information chunks related to the query.
- HPro uses a Vertex AI Vector Search index with four separate indexes for SDSs, Warehouse Safety Guides, pictograms, and product data.

4. **Context Retrieval:**
   - The top-matching documents or chunks from the vector database are retrieved as context.
   - This context provides the LLM with relevant background information to answer the query accurately.
5. **Gen AI Model (e.g., Gemini):**
   - The user query and the retrieved context are combined and fed as input to the LLM.
   - The LLM processes this information and generates a response.
   - HPro uses Gemini Flash to process the query and context.
6. **Reliable Answer:**
   - The LLM generates a response that is more likely to be accurate, relevant, and comprehensive due to the added context.

Conclusion

This chapter provided a foundational understanding of RAG serving workflows and their components. You also learned about the benefits of using RAG and how Vertex AI provides the necessary tools and infrastructure to implement RAG effectively. In the subsequent chapters, we will delve deeper into the practical implementation of these concepts using the Vertex AI SDK for ABAP.

Chapter 14: Configure Vector Search in SAP

	This chapter guides you through configuring SAP to connect to your Vector Search endpoint, enabling retrieval of relevant information from your indexed data.

To commence the initial phase of developing the RAG workflow, we shall configure the Vertex AI SDK for ABAP to utilize the Vector Search endpoint established in Chapter 12: Create and Update Vector Index. Please proceed with the below steps:

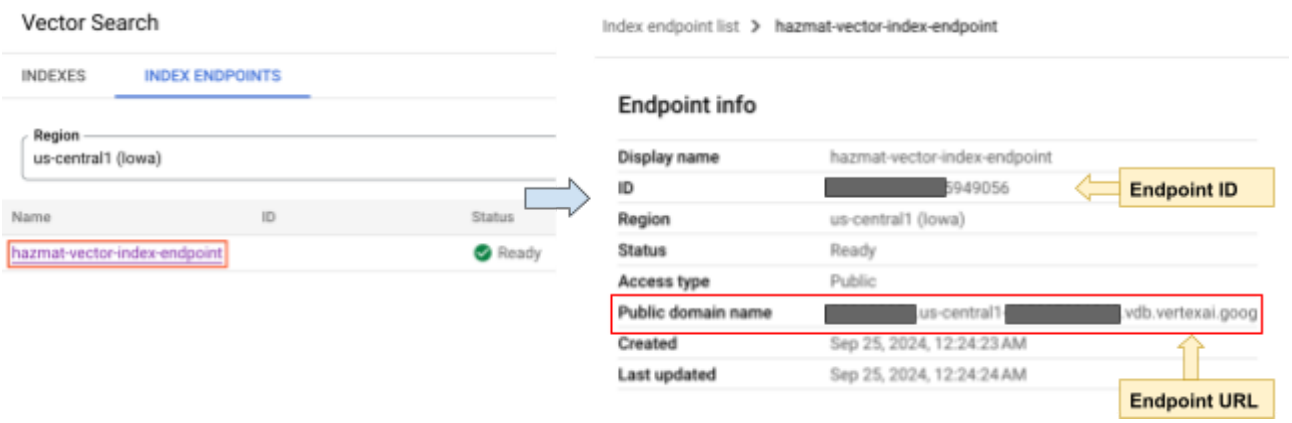
Step1: Gather the parameters to configure Vector search in SAP

In this step we will gather the following parameter for configuring Vector search

- Endpoint ID
- Endpoint URL
- Deployment ID for all below four indexes:
    - hazmat-sds-vector
    - hazmat-wsg-vector
    - hazmat-pictogram-vector
    - hazmat-prod-vector

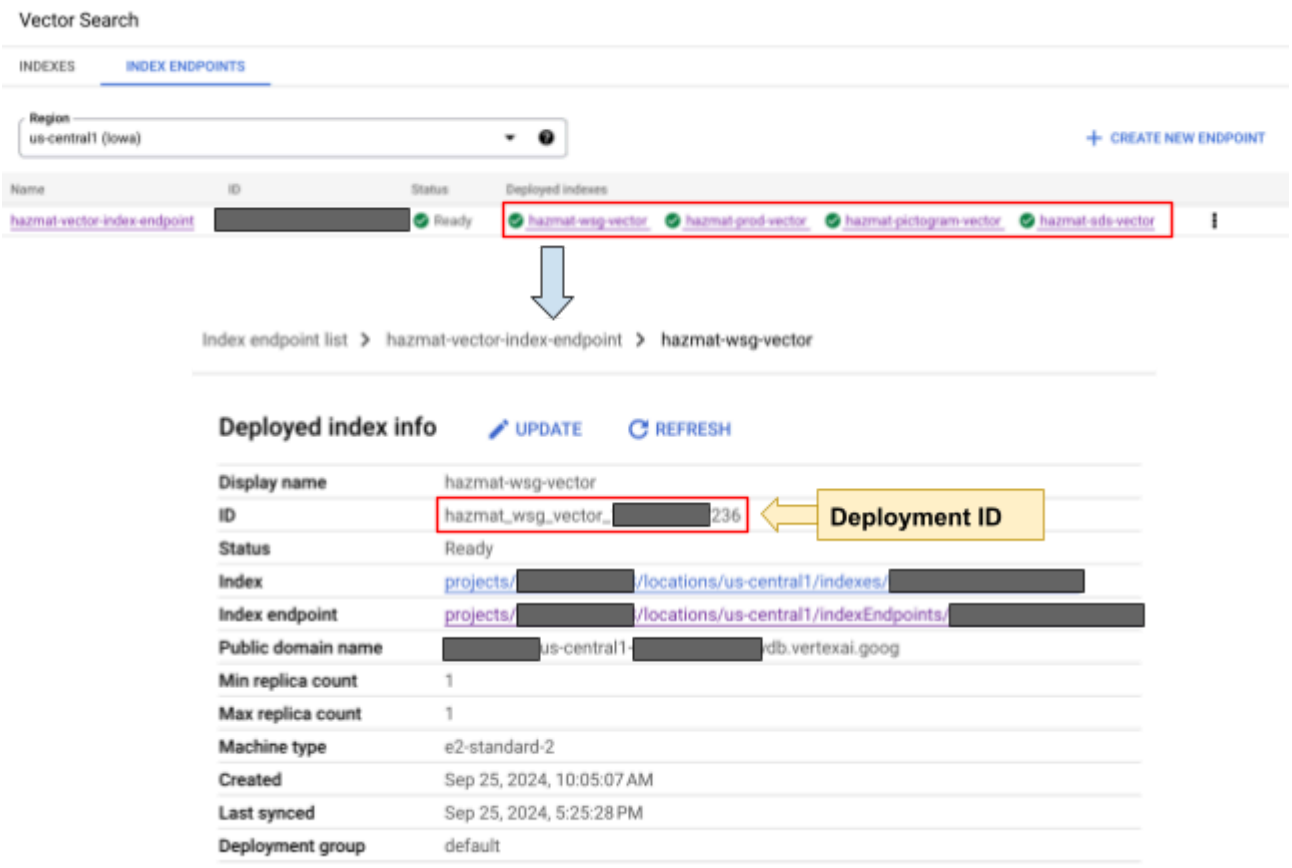
Get Endpoint ID and Endpoint URL:

Go to Vector Search in Google cloud console and click on the Index Endpoint. Copy the **endpoint id** which will be used for SDK configuration later and also **public domain name**, which will be used as a host url while configuring the RFC destination.



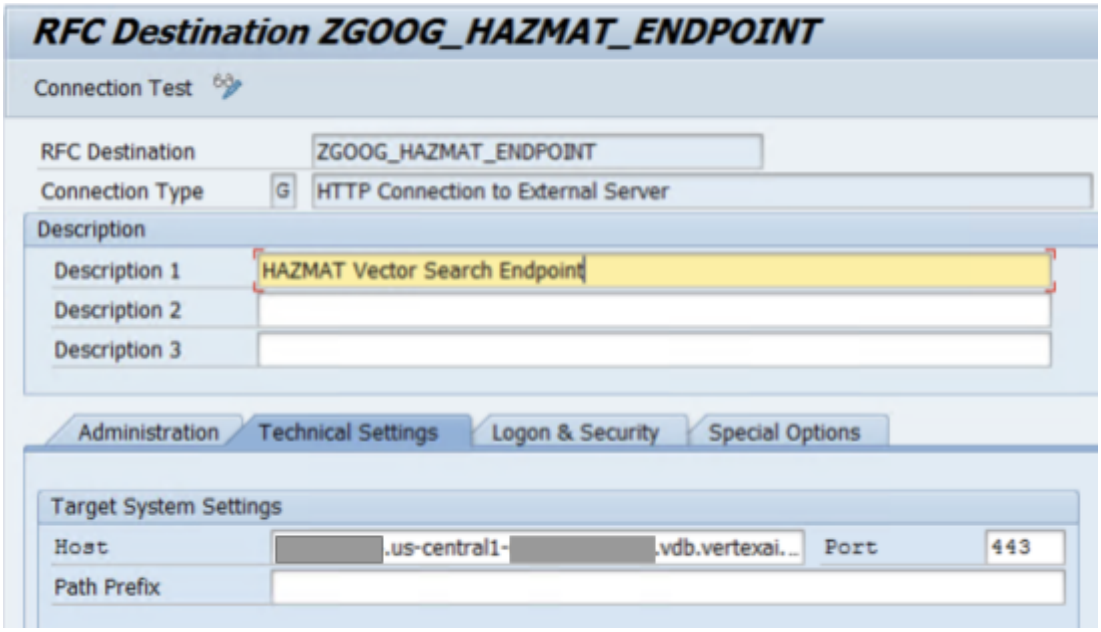
Get Deployment ID for indexes:

Go to Vector Search in Google cloud console and click on the Deployed Indexes to get the deployment ID for all four indexes as shown below.



Step2: Establish an RFC destination.

1. In the SAP GUI, enter transaction code **SM59**.
2. Create new RFC destinations with the name `ZGOOG_HAZMAT_ENDPOINT` of type G- **HTTP Connections to External Server**.
3. Go to the **Technical Settings** tab and enter the following details:
    - **Target Host**: enter the Vector search endpoint host URL copied from
    - **Port**.: enter 443. This port number is used for secure communication.
4. Go to the **Logon & Security** tab and make sure that the **SSL** is set as **Active** and **SSL Certificate** field is set with the option **DFAULT SSL Client (Standard)**.



Step3: Create a service mapping entry for the search invoker.

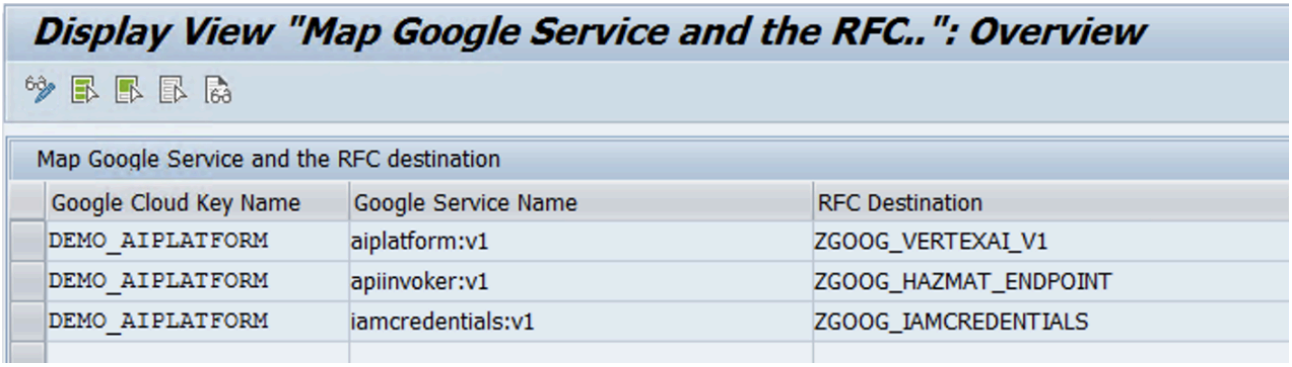
Configure the service mapping table for invoking the Vector Search Index endpoint using the ABAP SDK for Google Cloud.

- In SAP GUI, execute the transaction code `/GOOG/SDK_IMG`.

- Alternatively, execute the transaction code SPRO, and then click **SAP Reference IMG**.
- Click ABAP SDK for **Google Cloud > Basic Settings > Configure Service Map**.
- Click **New Entries** and create with the following values.

Field	Values
Google Cloud Key Name	DEMO_AIPLATFORM
Google Service Name	apiinvoker:v1
RFC Destination	ZGOOG_HAZMAT_ENDPOINT

After the successful completion of this step, the service map configuration table will contain the following three entries:



Step4: Configure Vector Search parameters.

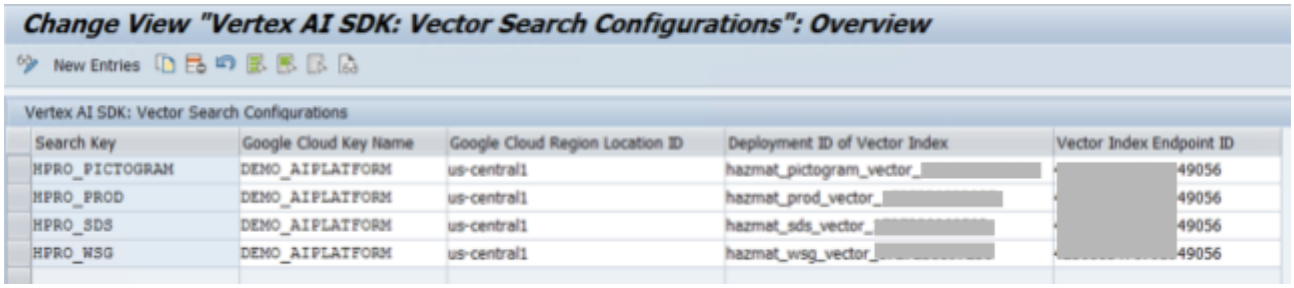
Configure the Vector search parameters for performing the Vector Search using the ABAP SDK for Google Cloud.

- In SAP GUI, go to SPRO, and then click **SAP Reference IMG**.
- Click ABAP SDK for **Google Cloud > Basic Settings > Vertex AI SDK: Configure Vector Search Parameters**.
- Click **New Entries** and create four entries with the following values.

	Index 1	Index 2	Index 3	Index 4

Search Key	HPRO_SDS	HPRO_WSG	HPRO_PICTOGRAM	HPRO_PROD
Google Cloud Key Name	DEMO_AIPLATFORM	DEMO_AIPLATFORM	DEMO_AIPLATFORM	DEMO_AIPLATFORM
Google Cloud Region Location ID	us-central1	us-central1	us-central1	us-central1
Deployment ID of Vector Index	hazmat_sds_vector_**xxxxx** *(Details From step1)*	hazmat_wsg_vector_**xxxxx** *(Details From step1)*	hazmat_pictogram_vector_**xxxxx** *(Details From step1)*	hazmat_prod_vector_**xxxxx** *(Details From step1)*
Vector Index Endpoint ID *(Same for all records)*	**xxxxx** *(Endpoint ID From step1)*	**xxxxx** *(Endpoint ID From step1)*	**xxxxx** *(Endpoint ID From step1)*	**xxxxx** *(Endpoint ID From step1)*

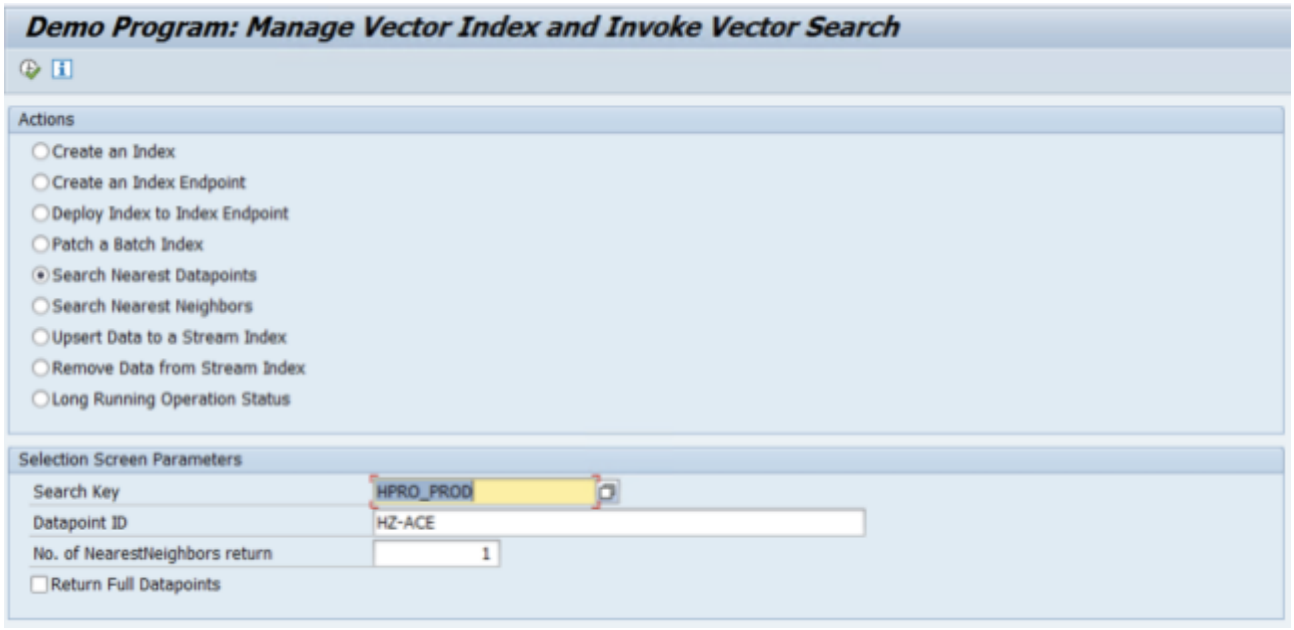
Once completed the table entries should look like this



Step5: Validate configuration by running the demo program.

Follow the below steps to validate the Vector search configuration

- In SAP GUI, go to SPRO, and then click **SAP Reference IMG**.
- Click ABAP SDK for **Google Cloud > Demos > Vertex AI SDK: Demo: Manage Vector Index and Invoke Vector Search.**
- Select **Search Nearest Neighbors** and enter the following values:
    - Search Key: HPRO_PROD
    - Datapoint ID: HZ-ACE
    - No. of NearestNeighbors return: 1



Successful Response:

Conclusion

By completing the steps in this chapter, you have successfully configured your SAP system to leverage the power of Vertex AI's Vector Search. This connection enables your ABAP applications to efficiently retrieve contextually relevant information from your indexed datasets, laying the groundwork for developing robust and intelligent applications, such as the HPro solution for HAZMAT handling. You're now well-equipped to move forward and implement the remaining components of your RAG workflow.

Chapter 15: Use Stremlit to to build the UI

	In this chapter we will deploy a Streamlit frontend app to Cloud Run and connect it to your ABAP backend HAZMAT service.

This section outlines the steps to deploy a Streamlit application to Cloud Run, which will interact with your SAP system using an ABAP service created within SICF (as described in Chapter 7). Your Streamlit application will communicate directly with this service.

(Note: Developers can explore alternative integration methods, such as using an OData service and a Fiori app, but for simplicity this guide focuses on the ABAP service approach.)

To deploy your Streamlit application to Cloud Run, follow these steps:

Step 1: Prepare your Streamlit Application

Clone the repository: Go to the Google Cloud shell and clone the GitHub repository containing your Streamlit frontend application code.

```
git clone https://github.com/google-cloud-abap/demo-hazmat-frontend.git
```

Step 2: Configure Your Application

Before deploying, you'll need to configure your Streamlit application to connect to your SAP system:

Open the config.toml file

```
cd demo-hazmat-frontend/config
vim config.toml
```

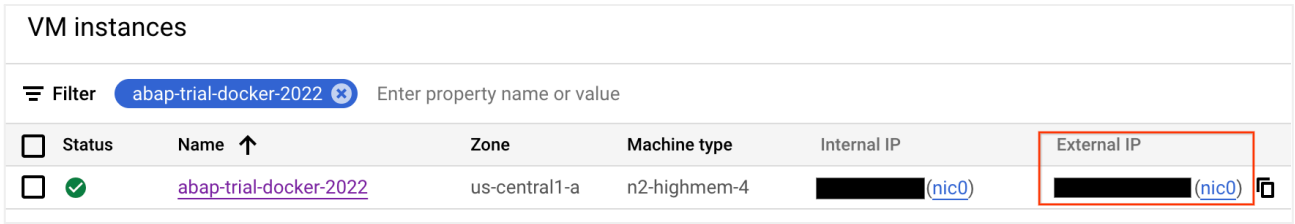
Modify the `base_url` in the config to include the External IP address of the SAP system.

Update the base URL: Replace the placeholder base URL with the external IP address of your SAP system. You can find this external IP address in your Google Cloud console where your SAP system is hosted.

```
[api]
base_url = "http://xx.xxx.xxx.xxx"  #  Modify this to SAP system External IP Address
port = 50000 #  Modify this to SAP system port if you are not using ABAP Trail
username = "DEVELOPER"  # Replace with your SAP username
password = "ABAPtr2022#00"  # Replace with your SAP password

get_prompt_api = "/sap/bc/hazmat_service/getPromptRepo"
get_product_api = "/sap/bc/hazmat_service/getProducts"
post_prompt_api = "/sap/bc/hazmat_service/processPrompt"
```

External IP address can be referred from the Google cloud console.



Step3: Deploy to cloud run

Navigate to the project directory `demo-hazmat-frontend` and execute the below deployment commands:

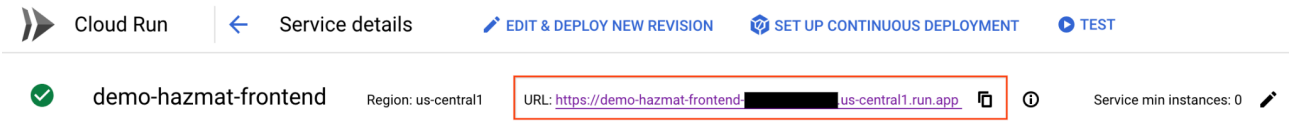
```
cd ..
export PROJECT_ID="abap-sdk-poc" # Your Project ID
export REGION="us-central1"
gcloud run deploy demo-hazmat-frontend --source . --region="$REGION"
--project="$PROJECT_ID" --allow-unauthenticated
```

Once the deployment is successful, you'll see a confirmation message in your terminal, including the service URL of your deployed application.

```
$> gcloud run deploy demo-hazmat-frontend --source . --region="$REGION" --project="$PROJECT_ID" --allow-unauthenticated
Building using Buildpacks and deploying container to Cloud Run service [demo-hazmat-frontend] in project [          ] region [us-central1]
OK Building and deploying new service... Done.
  OK Uploading sources...
  OK Building Container... Logs are available at [https://console.cloud.google.com/cloud-build/builds/e1d363be-4fd7-48e5-b8ca-caf7b8c2cc88?project=          ].
  OK Creating Revision...
  OK Routing traffic...
  OK Setting IAM Policy...
Done.
Service [demo-hazmat-frontend] revision [demo-hazmat-frontend-00001-hqm] has been deployed and is serving 100 percent of traffic.
Service URL: https://demo-hazmat-frontend-          .us-central1.run.app
```

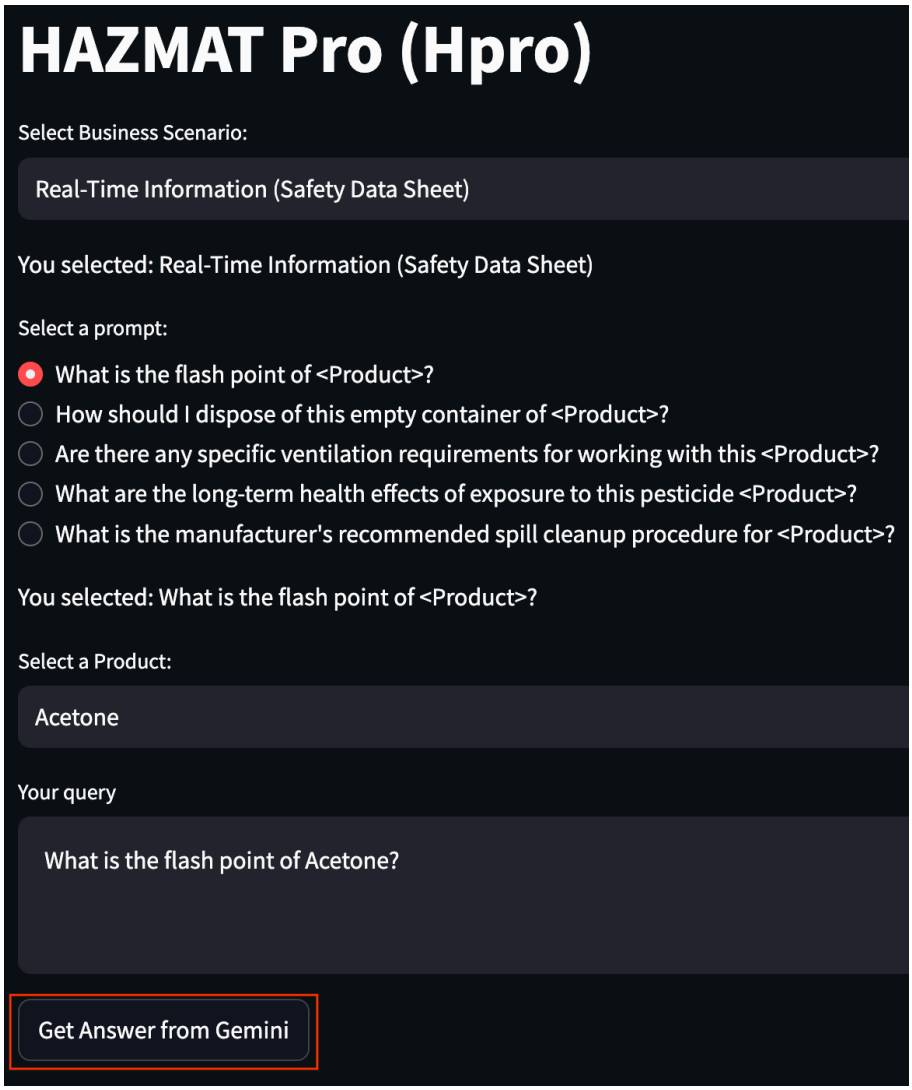
Step4: Test your application

Access the application: Click on the service URL from the deployment output message to open your Streamlit application in your web browser or open the app by navigating to cloud run in the Google Cloud Console.

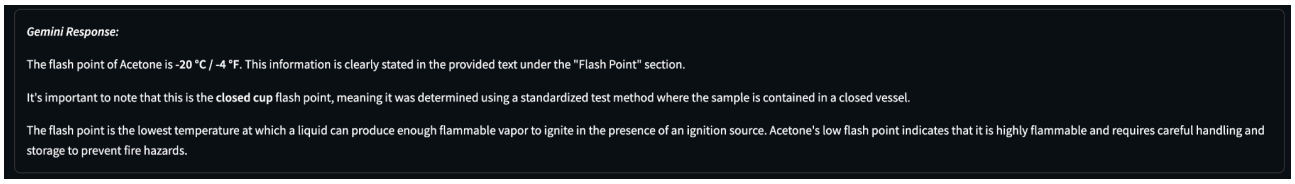


Run a test scenario:

- Select "Real-Time Information (Safety Data Sheet)" as the scenario.
- Select "What is the flash point of <Product>" as the prompt.
- Enter "Acetone" as the product.
- Click on "Get Answer from Gemini."



Verify the response: You should receive a response displaying the flash point of Acetone, retrieved from your SAP system.



Explore and Extend

Feel free to explore the application by selecting other business scenarios and prompts. Many of the processing workflows are intentionally left unimplemented, providing you with the opportunity to:

- **Code your own prompt processing workflows**: Implement the logic to handle different prompts and scenarios, integrating with your SAP system as needed.
- **Customize and enhance**: Add new features, improve the user interface, and tailor the application to your specific requirements.

Chapter 16: Under the Hood of HPro: Prompt Processing and Response Generation

This chapter explores the design of the HPro application, focusing on its prompt processing workflow. Building on the RAG concepts from Chapter 13, we'll detail how HPro handles user prompts and generates insightful responses.

Our application was designed to address the following key scenarios within the business:

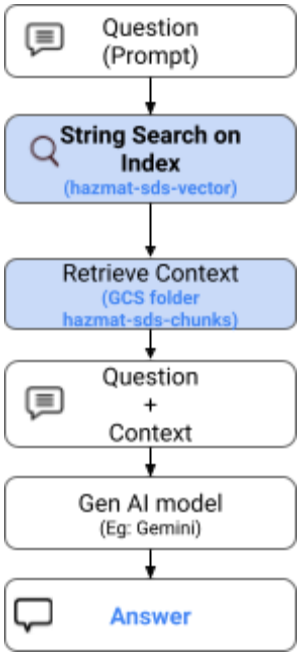
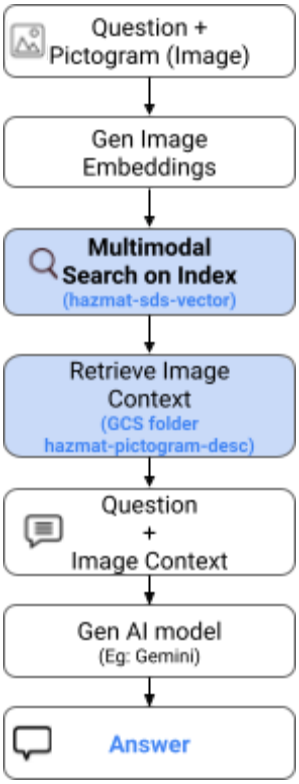
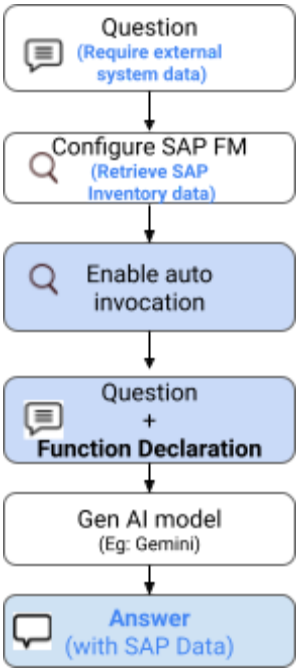
Scenario	Functionality	Enabled in HPro?
Real-Time Information (Safety Data Sheet)	Gemini leverages RAG to access up-to-date information from SDS documents provided by the manufacturer(example), ensuring accuracy and relevance.	Yes
AI-Powered Pictogram Recognition	Users can photograph hazard pictograms for instant identification and detailed information.	Yes
Inventory Integration	Seamlessly links to SAP inventory data for real-time stock visibility and material-specific instructions.	Yes
Emergency Response Guidance	Provides clear, step-by-step instructions for first aid and exposure control in critical situations.	Partially
Risk Assessment and Mitigation	Scenario-based guidance and proactive risk identification using LLM analysis of SDS and SAP EHS data.	No
Training and Education	Personalized learning paths, interactive quizzes, and simulations for enhanced HAZMAT knowledge.	No
Incident Reporting and Analysis	Guided incident reporting and LLM-powered root cause analysis.	No
Waste Management Optimization	Waste stream classification assistance and resource recovery recommendations.	No

Exploring the Potential of AI

To demonstrate the power of AI in real-time decision-making, we created a set of predefined prompts that simulated user queries. This allowed us to showcase how AI could be integrated into various aspects of the application, beyond traditional chatbot interactions.

Developing a Prompt Processing Framework

To effectively handle these diverse scenarios, we developed a flexible prompt processing workflow. As an example the following three workflows are implemented in the current version of prototype.

Real-Time Information (Safety Data Sheet)	AI-Powered Pictogram Recognition	Inventory Integration
		

By combining AI-powered capabilities with a robust prompt processing framework, we aim to streamline decision-making, and enhance overall operational efficiency.

Chapter 17: Conclusion

Summary of Key Takeaways

- **Retrieval Augmented Generation (RAG)** is a powerful technique for enhancing large language models (LLMs) by providing them with relevant external knowledge.
- The **Vertex AI SDK** for ABAP makes it easy to integrate Vertex AI services, such as Gemini and Vector Search, into your SAP applications.
- By using **RAG** and the **Vertex AI SDK** for ABAP, you can develop applications that can answer user questions accurately, reliably, and with context.

Potential Future Directions

- As LLMs continue to evolve, we can expect to see even more innovative applications of RAG in the future.
- The Vertex AI SDK for ABAP is a valuable tool for developers who want to build AI-powered applications on SAP.
- By staying up-to-date on the latest advancements in AI and SAP, you can ensure that your applications are always at the forefront of innovation.

Call to Action

- I encourage you to explore the resources mentioned in this book and to experiment with the Vertex AI SDK for ABAP.
- I believe that RAG has the potential to transform the way we develop and use enterprise applications.
- I look forward to seeing the innovative applications that you create.

Glossary

ABAPGIT: is a Git client for ABAP that allows you to manage ABAP source code in Git repositories.

BigQuery: is a fully managed, serverless data warehouse that enables scalable analysis over petabytes of data.

Cloud Run: is a fully managed serverless platform that automatically scales your stateless containers.

Cloud Shell: is a web-based, interactive shell environment that you can use to manage your Google Cloud resources.

Document AI: is a Google Cloud service that uses machine learning to extract information from documents.

Embedding: is a numerical representation of a piece of data, such as text or an image, that can be used for similarity search.

Gemini: is a family of large language models (LLMs) developed by Google.

Google Cloud Storage (GCS): is an object storage service for storing any kind of data.

HAZMAT (Hazardous Materials): are substances that can pose a risk to health, safety, or the environment.

HPro (HAZMAT Pro): is an AI-powered prototype application for enhancing warehouse safety.

JSONL (JSON Lines): is a newline-delimited JSON format for storing structured data.

Knowledge Chunk: is a small, semantically meaningful unit of information extracted from enterprise data.

Large Language Model (LLM): is a type of artificial intelligence (AI) that can understand and generate text in response to a wide range of prompts and questions.

Pub/Sub: is a real-time messaging service that allows you to send and receive messages between independent applications.

PyMuPDF: is a Python library for working with PDF files.

RAG (Retrieval Augmented Generation) is a technique for enhancing LLMs by providing them with relevant external knowledge.

Safety Data Sheet (SDS): is a document that provides detailed information about the hazards of a chemical and how to handle it safely.

SAP EHS (Environment, Health, and Safety): is a software solution that helps organizations manage environmental, health, and safety regulations.

SICF (Service Implementation Cockpit Framework): is a tool for configuring and managing ICF services in SAP.

Streamlit: is an open-source Python library that makes it easy to create and share custom web apps for machine learning and data science.

Tiktoken: is a tool for counting tokens in text, which is used to measure the input and output of LLMs.

Vector Database: is a type of database that is optimized for storing and searching vector embeddings.

Vector Index: is a data structure that is used to organize vector embeddings for efficient similarity search.

Vertex AI: is a Google Cloud service that provides a unified platform for building and deploying machine learning models.