Step 1: Setup Google Cloud Project

✅ In Google Cloud Console:

1. Go to **Cloud Console** → **Project Selector** → **New Project**

Name it, for example:

perl
CopyEdit
```
my-rag-agent-project
```

2.
3. Click **Create**.

Step 2: Enable Required APIs

In **Cloud Console** → **APIs & Services** → **Enabled APIs**:

- Enable these APIs:
    - Vertex AI API
    - Discovery Engine API.
    - Cloud Storage API
    - Identity and Access Management API

If you don't see "Agentspace", you may have to request allowlist from Google sales, as it is enterprise-featured.

Step 3: Create a Cloud Storage Bucket

✅ In **Cloud Storage**:

1. Go to **Storage** → **Buckets** → **Create**

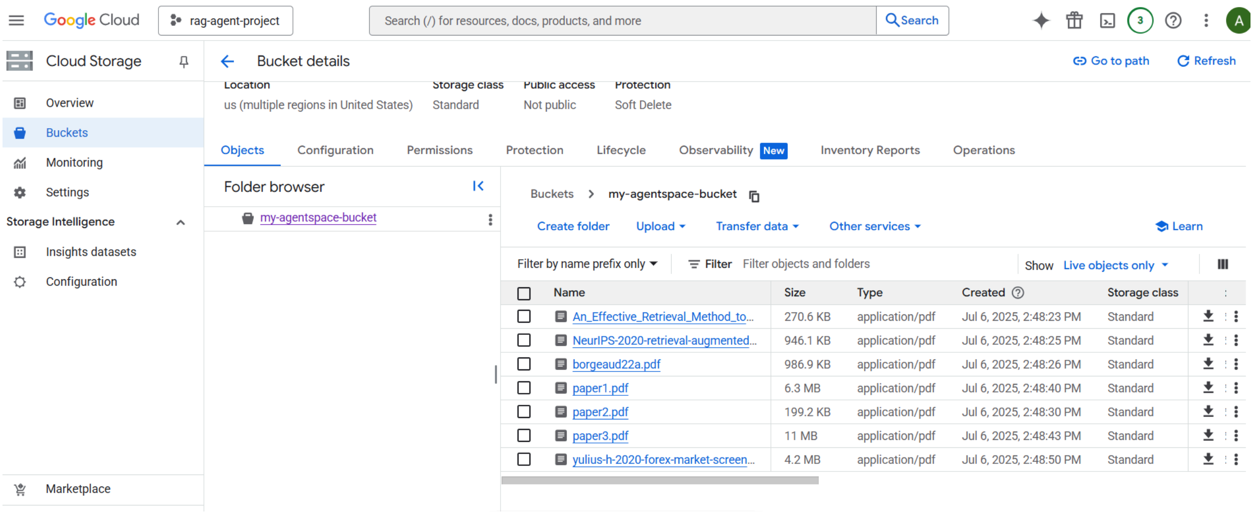
Name it for example:

perl
CopyEdit
`my-agentspace-bucket`

2.
3. Choose a region (say, `us-central1`)
4. Click **Create**.

✅ Upload your PDF:

- In that bucket, click **Upload File** → pick your `employee_policy.pdf`.



Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

☐	Name	Size	Type	Created	Storage class
☐	An_Effective_Retrieval_Method_to...	270.6 KB	application/pdf	Jul 6, 2025, 2:48:23 PM	Standard
☐	NeurIPS-2020-retrieval-augmented...	946.1 KB	application/pdf	Jul 6, 2025, 2:48:25 PM	Standard
☐	borgeaud22a.pdf	986.9 KB	application/pdf	Jul 6, 2025, 2:48:26 PM	Standard
☐	paper1.pdf	6.3 MB	application/pdf	Jul 6, 2025, 2:48:40 PM	Standard
☐	paper2.pdf	199.2 KB	application/pdf	Jul 6, 2025, 2:48:30 PM	Standard
☐	paper3.pdf	11 MB	application/pdf	Jul 6, 2025, 2:48:43 PM	Standard
☐	yulius-h-2020-forex-market-screen...	4.2 MB	application/pdf	Jul 6, 2025, 2:48:50 PM	Standard

Go to Discovery Engine

In Google Cloud Console, go to the left side menu:

nginx
CopyEdit

```
Vertex AI → Discovery Engine (or Vertex AI → Agent Builder /
AgentSpace, depending on your console name)
```

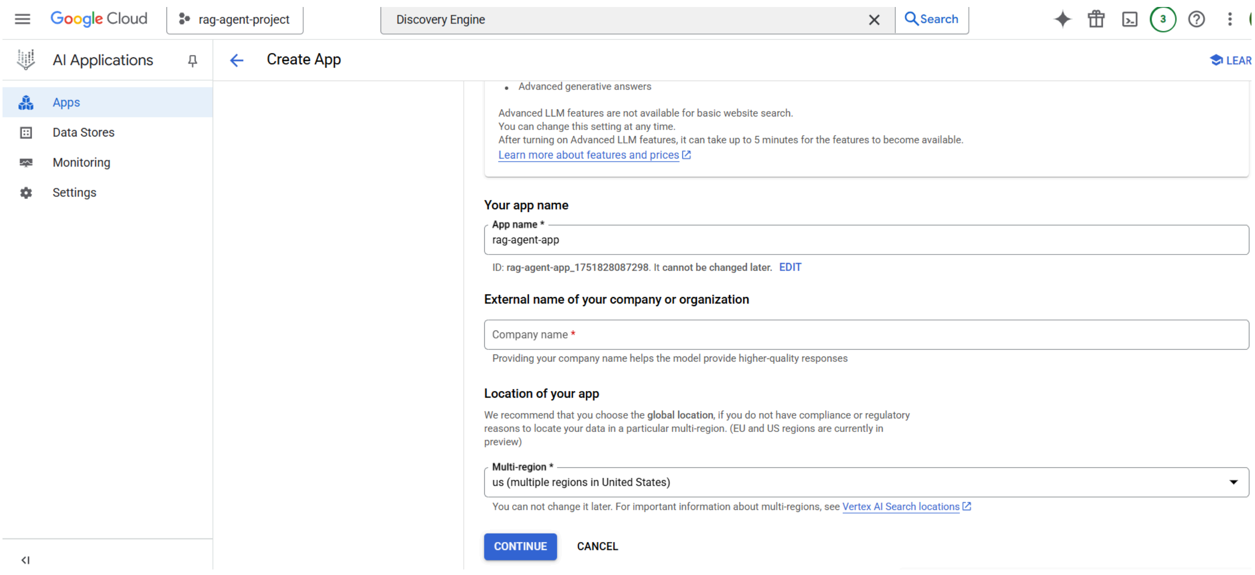
1. If you do not see Discovery Engine, you might need to enable *Discovery Engine API* in APIs & Services.

2. In Discovery Engine, choose:
 Data Stores → Create Data Store
 Since you want to do RAG over PDF documents (unstructured knowledge base), you should select:

➡️ Custom search (general)

This one is designed for:

- **Unstructured content (like your PDFs)**
- **Cloud Storage connectors**
- **Generative AI mode with chunk + rerank**
- **And flexible chunking**



- Advanced generative answers

Advanced LLM features are not available for basic website search.
You can change this setting at any time.
After turning on Advanced LLM features, it can take up to 5 minutes for the features to become available.
Learn more about features and prices

Your app name

App name *: rag-agent-app

ID: rag-agent-app_1751828087298. It cannot be changed later. EDIT

External name of your company or organization

Company name *

Providing your company name helps the model provide higher-quality responses

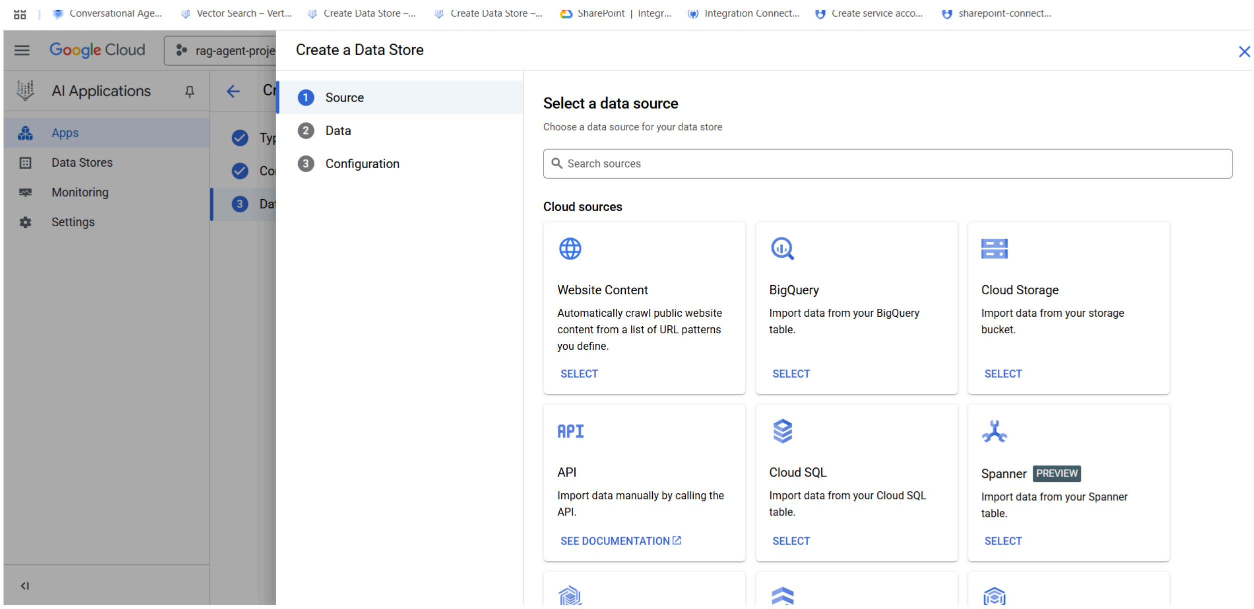
Location of your app

We recommend that you choose the **global location**, if you do not have compliance or regulatory reasons to locate your data in a particular multi-region. (EU and US regions are currently in preview)

Multi-region *: us (multiple regions in United States)

You can not change it later. For important information about multi-regions, see Vertex AI Search locations

lick CREATE DATA STORE (the blue button at the top).



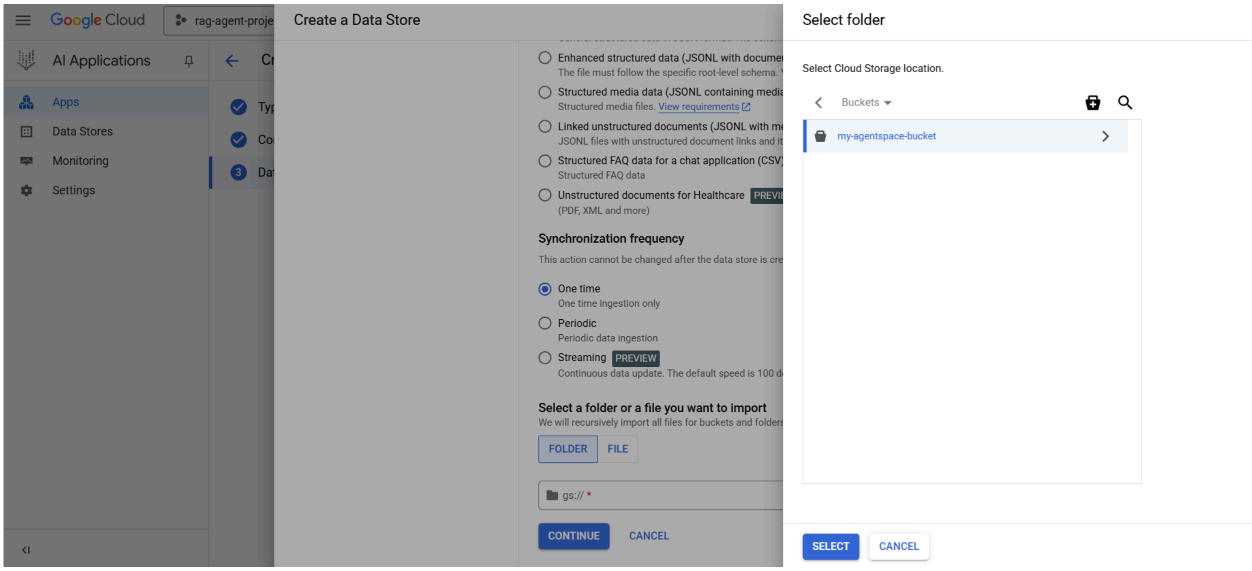
Create a Data Store

Select a data source

Choose a data source for your data store

Cloud sources

Website Content Automatically crawl public website content from a list of URL patterns you define.	BigQuery Import data from your BigQuery table.	Cloud Storage Import data from your storage bucket.
API Import data manually by calling the API.	Cloud SQL Import data from your Cloud SQL table.	Spanner Import data from your Spanner table.



Create a Data Store

○ Enhanced structured data (JSONL with documen
The file must follow the specific root-level schema.

○ Structured media data (JSONL containing media
Structured media files. View requirements

○ Linked unstructured documents (JSONL with me
JSONL files with unstructured document links and it

○ Structured FAQ data for a chat application (CSV)
Structured FAQ data

○ Unstructured documents for Healthcare PREVIE
(PDF, XML and more)

Synchronization frequency

This action cannot be changed after the data store is cre

● One time
One time ingestion only

○ Periodic
Periodic data ingestion

○ Streaming PREVIEW
Continuous data update. The default speed is 100 d

Select a folder or a file you want to import

We will recursively import all files for buckets and folders

gs:// *

Select folder

Select Cloud Storage location.

my-agentspace-bucket

✅ Default document parser

- **leave as `Layout Parser` (good for PDFs with headings and sections)**

✅ Layout parser settings

- ***Optional*:**

  - Enable table annotation → check this if your PDFs have important tables you want the LLM to reason over
  - Enable image annotation → only check if you have images with text you want to extract (OCR).
- If your PDFs are mostly text, you can leave both unchecked for now.

✅ Document chunking

- ✔️ *Enable advanced chunking configuration* (already checked in your screenshot)
- Chunk size limit: 500 tokens → perfect
- Include ancestor headings in chunks → you can check this if you want the chunk to also include its parent headings (like a section name) for more context. Recommended for policy documents.

So I'd suggest:

- check "Include ancestor headings in chunks" ✅
 (this helps provide extra context to the LLM)

✅ GENERATIVE AI OPTIONS (expand the accordion):

- Enable advanced generative answers
- Enable reranking

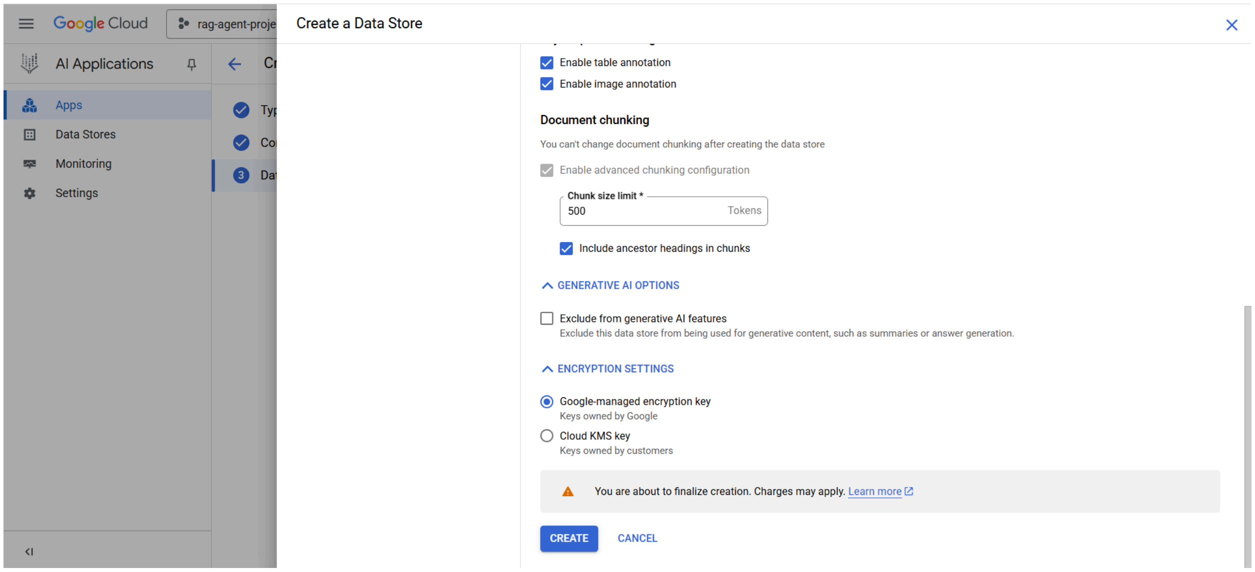
✅ ENCRYPTION SETTINGS

- you can leave default unless you have your own CMEK key

✅ Double-check:

- Default document parser: `Layout Parser`

- **Chunk size: `500`**
- **Include ancestor headings: ✅ checked**
- **Table + image annotations: ✅ checked**
- **Advanced generative answers: active (leave “exclude” unchecked)**
- **Encryption: Google-managed**



☑ Enable table annotation
☑ Enable image annotation

Document chunking

You can't change document chunking after creating the data store

☑ Enable advanced chunking configuration

Chunk size limit * 500 Tokens

☑ Include ancestor headings in chunks

GENERATIVE AI OPTIONS

☐ Exclude from generative AI features
Exclude this data store from being used for generative content, such as summaries or answer generation.

ENCRYPTION SETTINGS

● Google-managed encryption key
Keys owned by Google
○ Cloud KMS key
Keys owned by customers

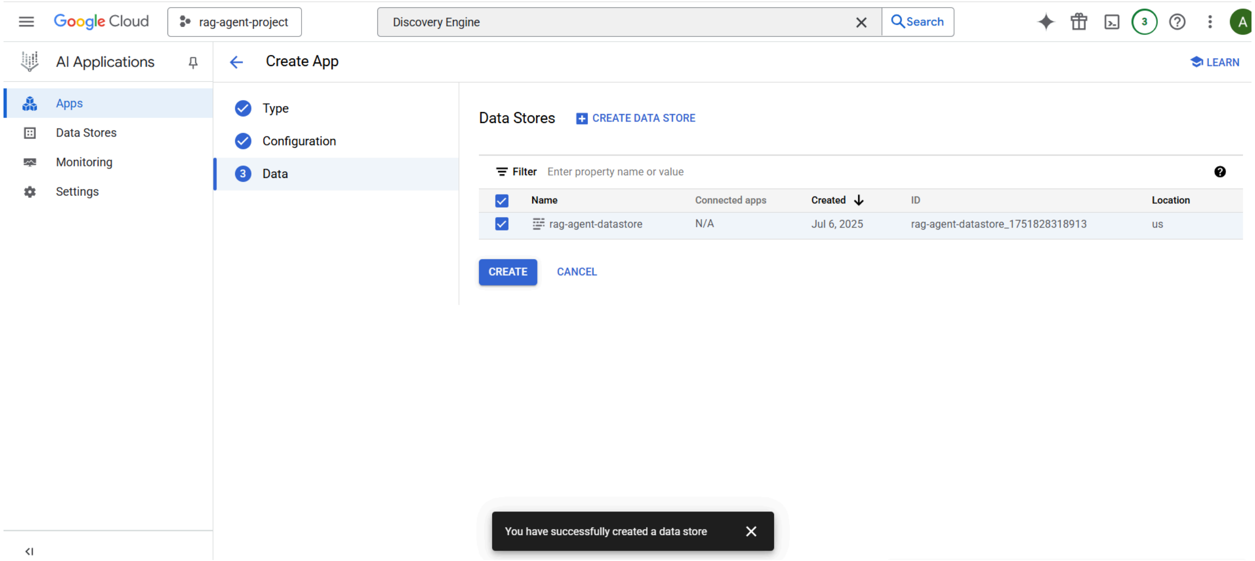
You are about to finalize creation. Charges may apply. Learn more

✅ Chunk size = 500 tokens (you set it)

✅ Overlap = 25% default (no box needed, built into the parser)

✅ Layout-aware = Layout Parser selected → ON

👉 This will:
✅ finalize linking your app with the data store
✅ kick off ingestion (you may see the status as `Ingesting` for a bit)
✅ make your PDFs fully discoverable for questions

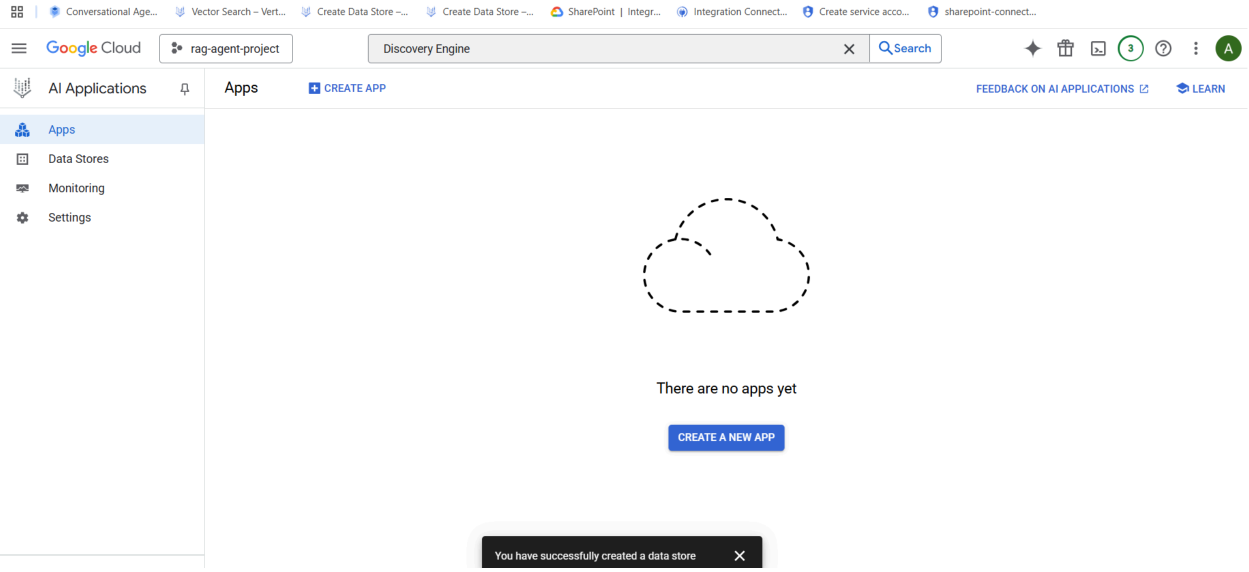


Create App

Data Stores

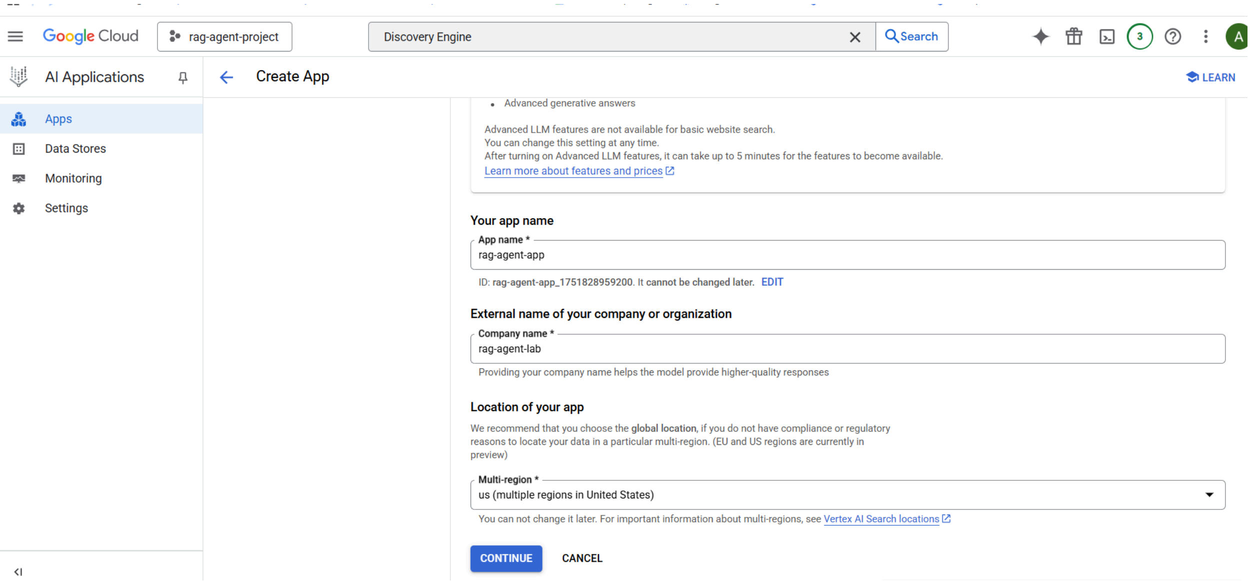
☑	Name	Connected apps	Created ↓	ID	Location
☑	rag-agent-datastore	N/A	Jul 6, 2025	rag-agent-datastore_1751828318913	us

You have successfully created a data store



There are no apps yet

You have successfully created a data store



- Advanced generative answers

Advanced LLM features are not available for basic website search.
You can change this setting at any time.
After turning on Advanced LLM features, it can take up to 5 minutes for the features to become available.
Learn more about features and prices

Your app name

App name *: rag-agent-app

ID: rag-agent-app_1751828959200. It cannot be changed later. EDIT

External name of your company or organization

Company name *: rag-agent-lab

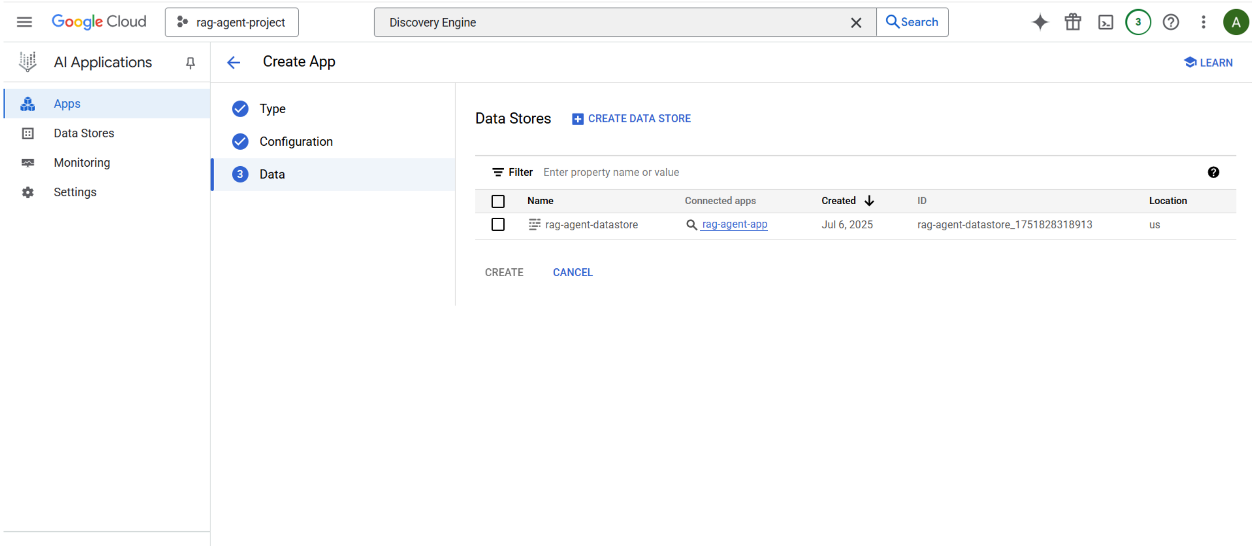
Providing your company name helps the model provide higher-quality responses

Location of your app

We recommend that you choose the **global location**, if you do not have compliance or regulatory reasons to locate your data in a particular multi-region. (EU and US regions are currently in preview)

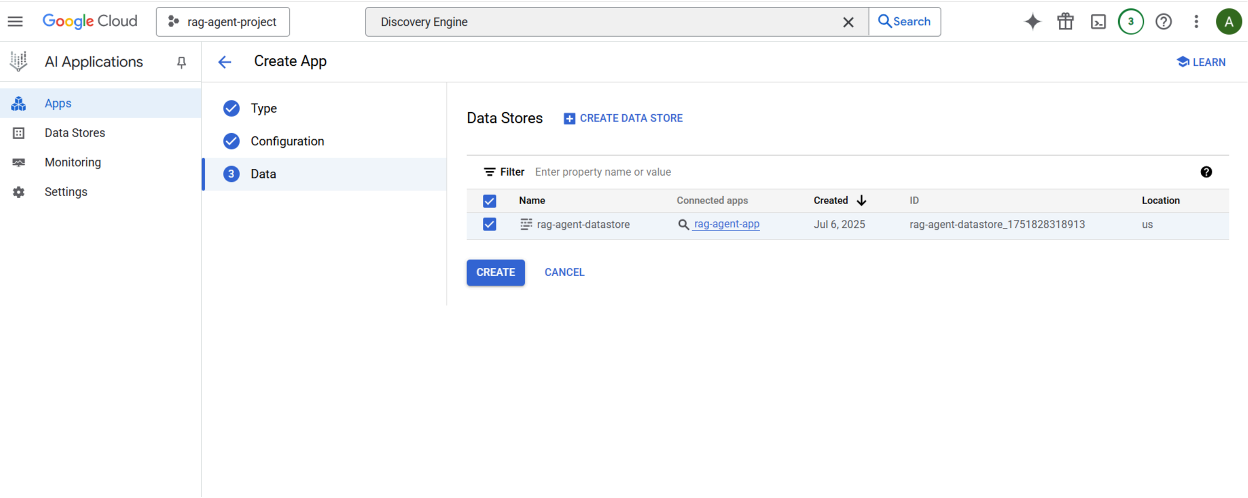
Multi-region *: us (multiple regions in United States)

You can not change it later. For important information about multi-regions, see Vertex AI Search locations



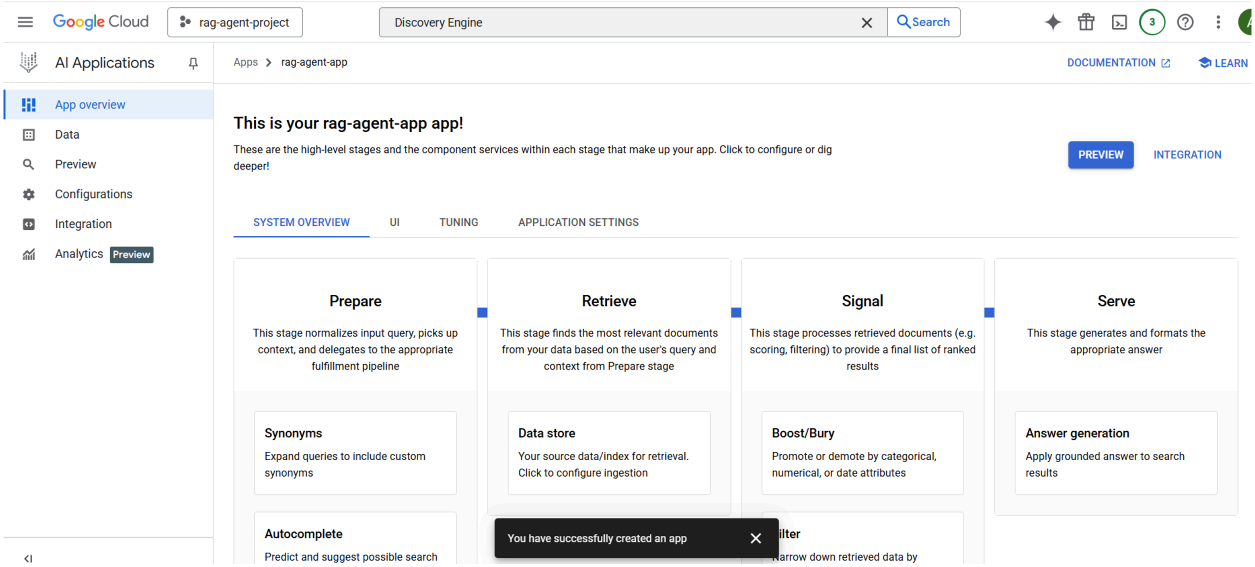
Data Stores

☐	Name	Connected apps	Created ↓	ID	Location
☐	rag-agent-datastore	rag-agent-app	Jul 6, 2025	rag-agent-datastore_1751828318913	us



Data Stores

☑	Name	Connected apps	Created ↓	ID	Location
☑	rag-agent-datastore	rag-agent-app	Jul 6, 2025	rag-agent-datastore_1751828318913	us



This is your rag-agent-app app!

These are the high-level stages and the component services within each stage that make up your app. Click to configure or dig deeper!

Prepare

This stage normalizes input query, picks up context, and delegates to the appropriate fulfillment pipeline

Retrieve

This stage finds the most relevant documents from your data based on the user's query and context from Prepare stage

Signal

This stage processes retrieved documents (e.g. scoring, filtering) to provide a final list of ranked results

Serve

This stage generates and formats the appropriate answer

Synonyms

Expand queries to include custom synonyms

Data store

Your source data/index for retrieval. Click to configure ingestion

Boost/Bury

Promote or demote by categorical, numerical, or date attributes

Answer generation

Apply grounded answer to search results

Autocomplete

Predict and suggest possible search

ilter

arrow down retrieved data by

You have successfully created an app

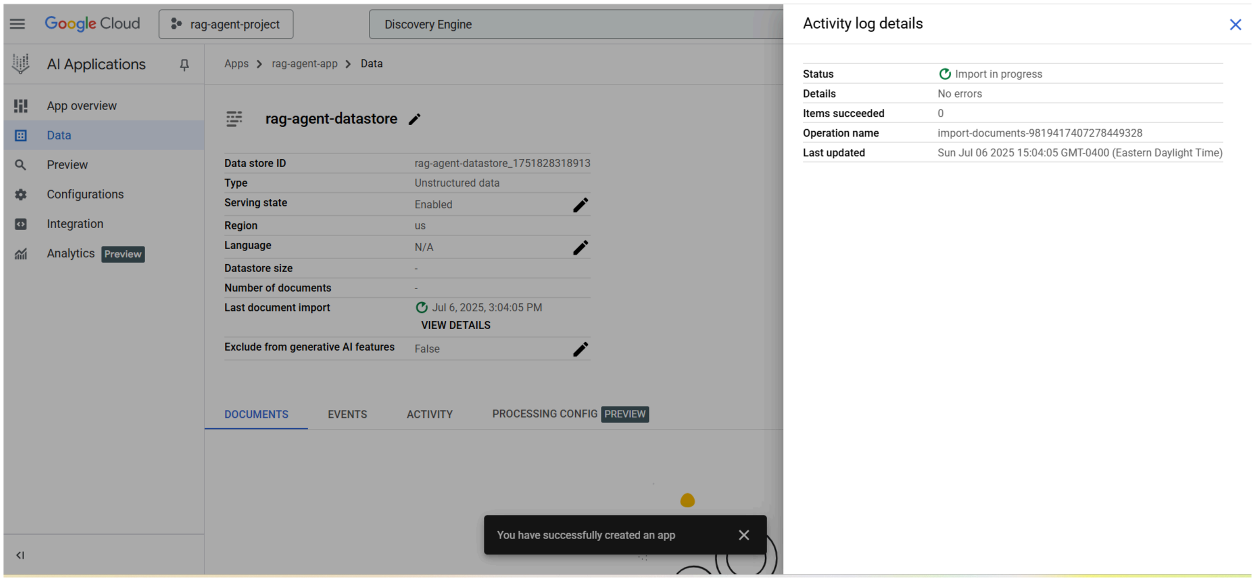
Test your app

👉 **Click the blue PREVIEW button (top right of your screenshot).**

This will open the *Test UI* where you can send sample queries.

✅ **The system will:**

- **run semantic search over your chunked documents**
- **rerank the top chunks**
- **send them to the LLM (Gemini or PaLM)**
- **generate an answer**
- **and highlight which chunks from your documents were used**



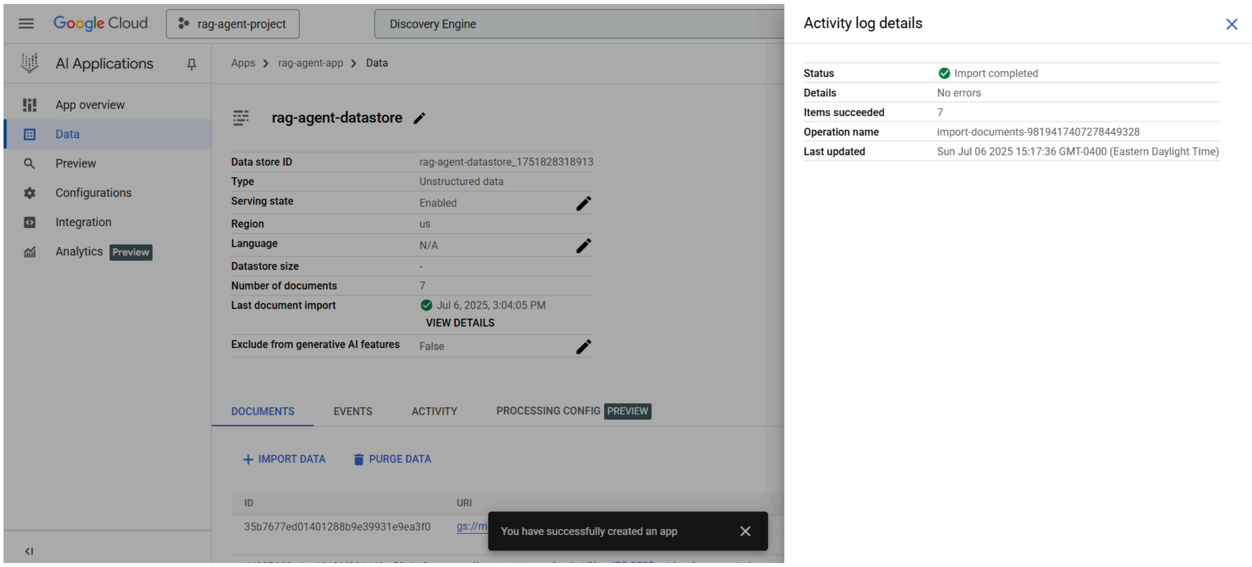
rag-agent-datastore

Data store ID	rag-agent-datastore_1751828318913
Type	Unstructured data
Serving state	Enabled
Region	us
Language	N/A
Datastore size	-
Number of documents	-
Last document import	Jul 6, 2025, 3:04:05 PM
Exclude from generative AI features	False

You have successfully created an app

Activity log details

Status	Import in progress
Details	No errors
Items succeeded	0
Operation name	import-documents-9819417407278449328
Last updated	Sun Jul 06 2025 15:04:05 GMT-0400 (Eastern Daylight Time)



rag-agent-datastore

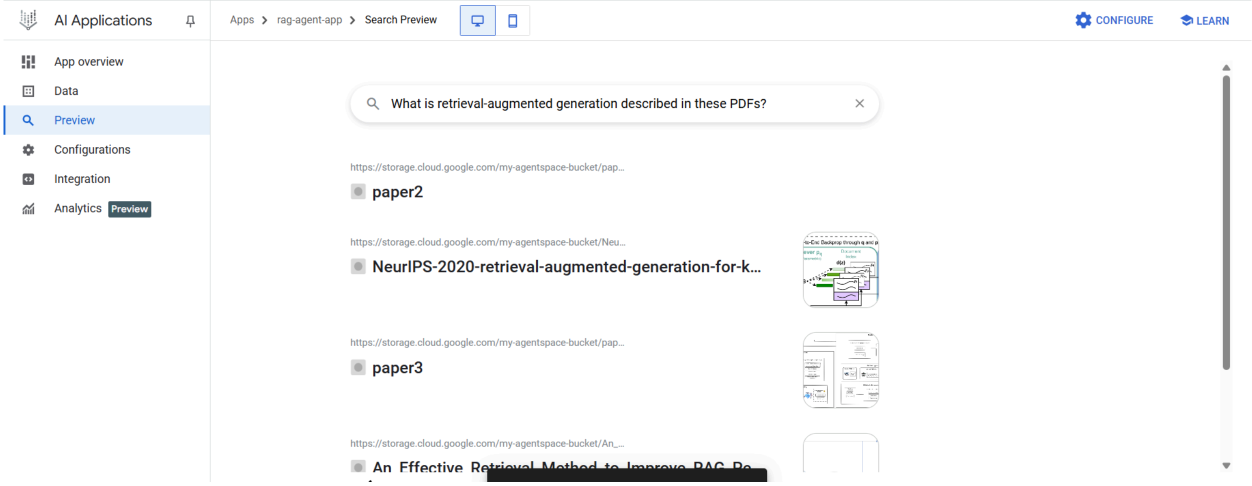
Data store ID	rag-agent-datastore_1751828318913
Type	Unstructured data
Serving state	Enabled
Region	us
Language	N/A
Datastore size	-
Number of documents	7
Last document import	Jul 6, 2025, 3:04:05 PM
Exclude from generative AI features	False

ID	URI
35b7677ed01401288b9e39931e9ea3f0	gs://m

You have successfully created an app

Activity log details

Status	Import completed
Details	No errors
Items succeeded	7
Operation name	import-documents-9819417407278449328
Last updated	Sun Jul 06 2025 15:17:36 GMT-0400 (Eastern Daylight Time)



What is retrieval-augmented generation described in these PDFs?

https://storage.cloud.google.com/my-agentspace-bucket/pap...
paper2

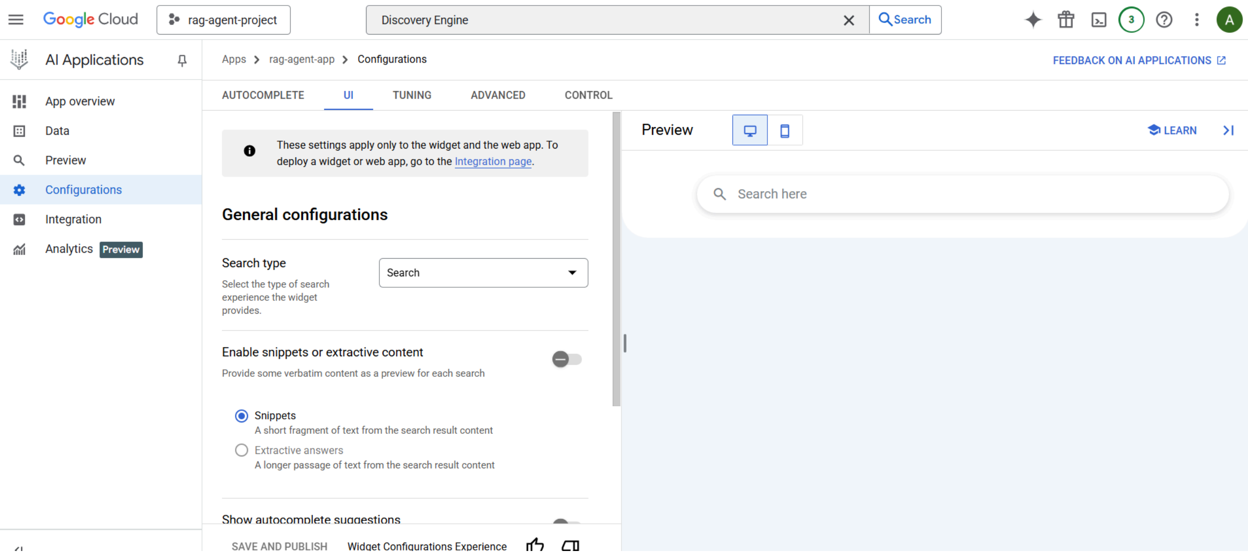
https://storage.cloud.google.com/my-agentspace-bucket/Neu...
NeurIPS-2020-retrieval-augmented-generation-for-k...

https://storage.cloud.google.com/my-agentspace-bucket/pap...
paper3

https://storage.cloud.google.com/my-agentspace-bucket/An_...
An Effective Retrieval Method to Improve RAG Pe...

Go to your app configuration

- Click on **App overview** → **Serve** → **Answer generation**
- Make sure "Grounded Answer Generation" is turned **on** (it usually is by default, but check).



These settings apply only to the widget and the web app. To deploy a widget or web app, go to the Integration page.

General configurations

Search type: Search
Select the type of search experience the widget provides.

Enable snippets or extractive content
Provide some verbatim content as a preview for each search

● Snippets
A short fragment of text from the search result content

○ Extractive answers
A longer passage of text from the search result content

Show autocomplete suggestions

To achieve your goal — specific answer with references + section names

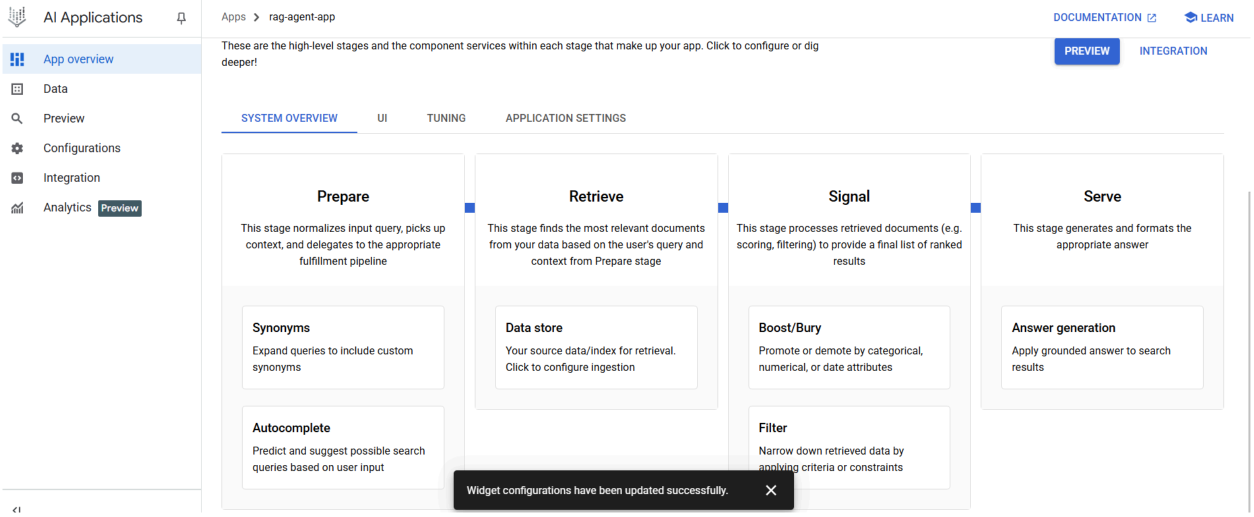
DO THIS INSTEAD:

✅ **Stay with Snippets** (required with advanced chunking)
✅ Then use the **Grounded Answer Generation** capability:

1. Go to **App Overview > Serve > Answer Generation**
    - enable *Grounded answers*
    - that lets the LLM combine chunks into a single natural-language answer
    - with references to each chunk (containing subtitle, heading, page etc)
2. Save and deploy

✅ *Result:*

- The system will use advanced chunking (as you set)
- then let the LLM *generate* a final answer with:
    - supporting chunks
    - and metadata (section names, page numbers, etc)



These are the high-level stages and the component services within each stage that make up your app. Click to configure or dig deeper!

Prepare

This stage normalizes input query, picks up context, and delegates to the appropriate fulfillment pipeline

Synonyms
Expand queries to include custom synonyms

Autocomplete
Predict and suggest possible search queries based on user input

Retrieve

This stage finds the most relevant documents from your data based on the user's query and context from Prepare stage

Data store
Your source data/index for retrieval. Click to configure ingestion

Signal

This stage processes retrieved documents (e.g. scoring, filtering) to provide a final list of ranked results

Boost/Bury
Promote or demote by categorical, numerical, or date attributes

Filter
Narrow down retrieved data by applying criteria or constraints

Serve

This stage generates and formats the appropriate answer

Answer generation
Apply grounded answer to search results

Widget configurations have been updated successfully.

✅ **1. Save and publish**
Click the blue **Save and Publish** button at the bottom left, so these snippet settings take effect.

✅ 2. Go to the App Overview

- In the left menu, click **App overview**
- You’ll see the four stages: *Prepare*, *Retrieve*, *Signal*, *Serve*
- In the **Serve** stage, click **Answer generation**

✅ 3. Enable Grounded Answer Generation

- Inside the *Serve* stage, there should be a toggle for **Answer Generation** (sometimes called “grounded answers” or “generative answers” in Google Discovery Engine)
- Enable it
- This feature allows the LLM to:
    - Take the chunks retrieved from your vector DB
    - Combine them
    - Generate a final answer
    - Include references to the chunk metadata (like heading, paragraph name, or page number)

👉 “Search with an answer”

Why?

- *Search with an answer* will generate a grounded answer above the search results, **and** it will still list the sources (chunks/paragraphs) with references.
- This mode allows you to get the best of both worlds:
    - A generative summary of the answer
    - While still surfacing the original passages with their metadata (section, page, heading)
    - And enables highlighted tokens in the result view

Recommendation: Select “Gemini 2.0 Flash 1”
because:

✅ tuned for Q&A and summarization
✅ fast (Flash)
✅ optimized for grounded answers
✅ supports multi-passage summarization with references

Next steps after selecting Gemini 2.0 Flash 1

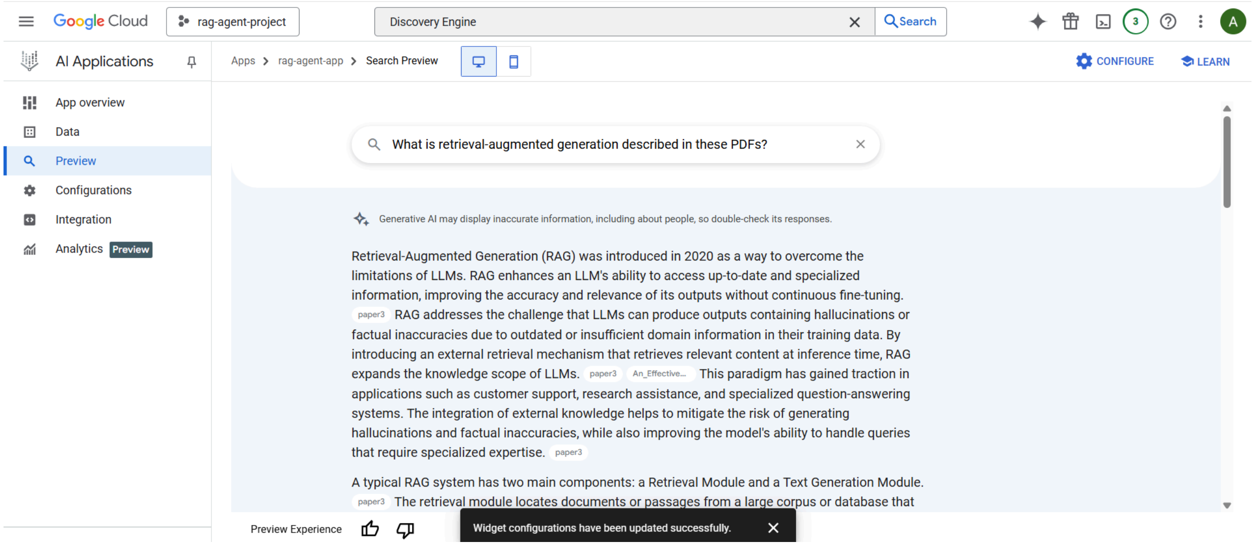
1. Save and Publish
2. Test in Preview
3. If needed, enrich your document metadata further (section, page, heading) during chunking
4. Verify highlighted references are showing up as expected

setting “English” might be safest, to keep all summaries uniform, unless you expect mixed languages.

👉 **My recommended setup for your case (testing + highlighting tokens + paragraph metadata):**

- **Enable related questions: ON** ✅
- **Ignore no answer summary for query: OFF** ⚪ (so you see when no relevant answer exists)
- **Ignore Adversarial Query: OFF** ⚪ (OK while testing)

When you go to production, you might revisit “Ignore Adversarial Query” and turn it ON.



What is retrieval-augmented generation described in these PDFs?

Generative AI may display inaccurate information, including about people, so double-check its responses.

Retrieval-Augmented Generation (RAG) was introduced in 2020 as a way to overcome the limitations of LLMs. RAG enhances an LLM's ability to access up-to-date and specialized information, improving the accuracy and relevance of its outputs without continuous fine-tuning. paper3 RAG addresses the challenge that LLMs can produce outputs containing hallucinations or factual inaccuracies due to outdated or insufficient domain information in their training data. By introducing an external retrieval mechanism that retrieves relevant content at inference time, RAG expands the knowledge scope of LLMs. paper3 An_Effective... This paradigm has gained traction in applications such as customer support, research assistance, and specialized question-answering systems. The integration of external knowledge helps to mitigate the risk of generating hallucinations and factual inaccuracies, while also improving the model's ability to handle queries that require specialized expertise. paper3

A typical RAG system has two main components: a Retrieval Module and a Text Generation Module. paper3 The retrieval module locates documents or passages from a large corpus or database that

Widget configurations have been updated successfully.

✅ you have configured **Classic RAG**
✅ you have also configured **Chunk-and-ReRank**
✅ with correct chunk size and overlap
✅ connected to GCS
✅ indexing is currently in progress

$$
$$$$

yes i want: ✅ answers pointing to specific paragraphs/subtitles ✅ highlighted tokens ✅ and paragraph metadata (like “section name”, “page”, “heading”)

What you will do:

- When ingesting documents, *chunk* them with a *custom chunker* that adds metadata for each chunk, like:
  - `page_number`
  - `section_heading`
  - `paragraph_index`

    - `original_text`

- Store these fields as metadata in the vector database (Discovery’s vector store).

✅ Then, in your retrieval or answer generation step, you will:

- pull back the top relevant chunks
- include their metadata in the LLM prompt (so it can cite sections)
- and optionally **highlight** the best matching tokens in the UI, if you build a frontend over the API.

🚀2️⃣How to inject metadata during chunking

If you are chunking a PDF with `Layout Parser` in Discovery, you cannot easily add advanced metadata in the console — so you’ll want to do **custom ingestion** with something like a Python script.

Sample chunk + metadata:

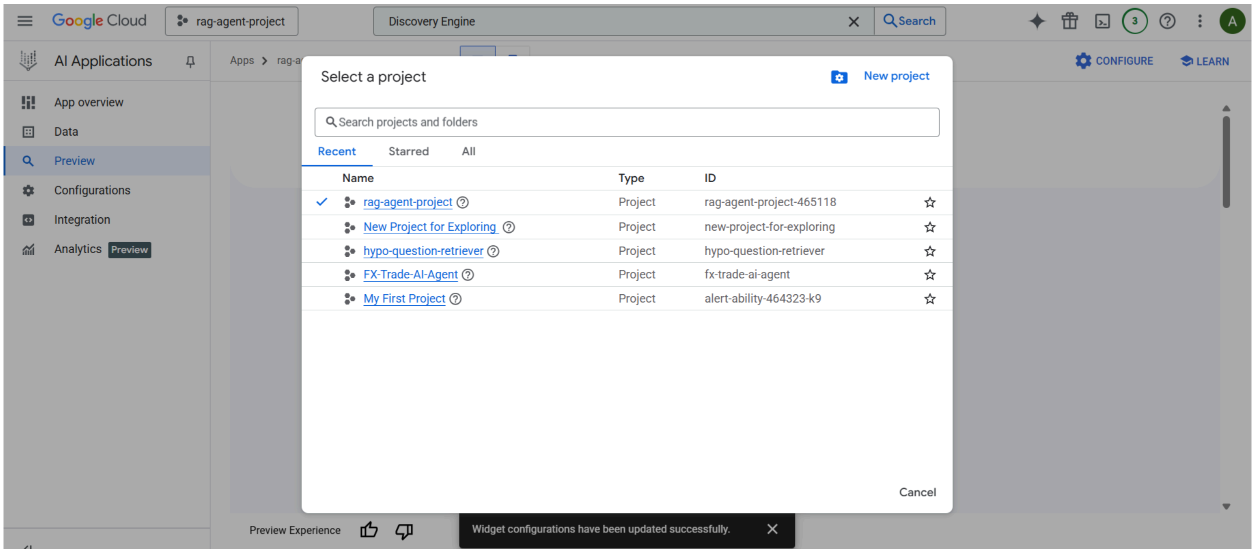
python
CopyEdit

```
{
  "id": "doc_123_page5_para2",
  "embedding": [0.45, 0.78, ...],
  "text": "Our employee vacation policy covers full-time
employees...",
  "metadata": {
      "source": "wells_handbook.pdf",
      "page": 5,
      "section": "Time Off and Leave",
      "heading": "Vacation Policy",
      "paragraph_index": 2
  }
}
```

You can do this chunking with:
✅ PyPDF2 / pdfminer to split PDF
✅ add headings from layout parsing
✅ embed with Gemini Embeddings API
✅ store with Discovery's `dataConnector` API (or even store in a hybrid vector db like Pinecone if you prefer, and connect Discovery on top)

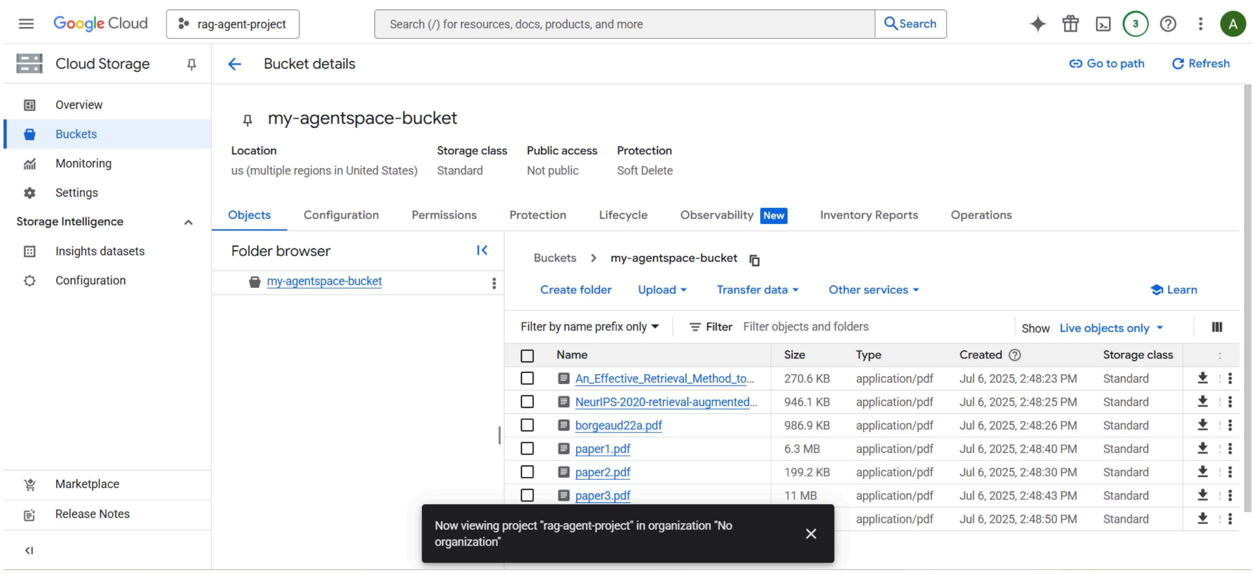
Let me be very clear: **most** of these advanced metadata chunking steps happen *outside* the Google Cloud Console, because the console itself does not do "paragraph-level metadata enrichment" automatically. But I will explain **where** the console fits in each step so you have the full picture.

✔ parse a PDF
✔ extract headings
✔ chunk with metadata
✔ embed with Gemini
✔ upload to Discovery



	Name	Type	ID	
✓	rag-agent-project	Project	rag-agent-project-465118	☆
	New Project for Exploring	Project	new-project-for-exploring	☆
	hypo-question-retriever	Project	hypo-question-retriever	☆
	FX-Trade-AI-Agent	Project	fx-trade-ai-agent	☆
	My First Project	Project	alert-ability-464323-k9	☆

Widget configurations have been updated successfully.



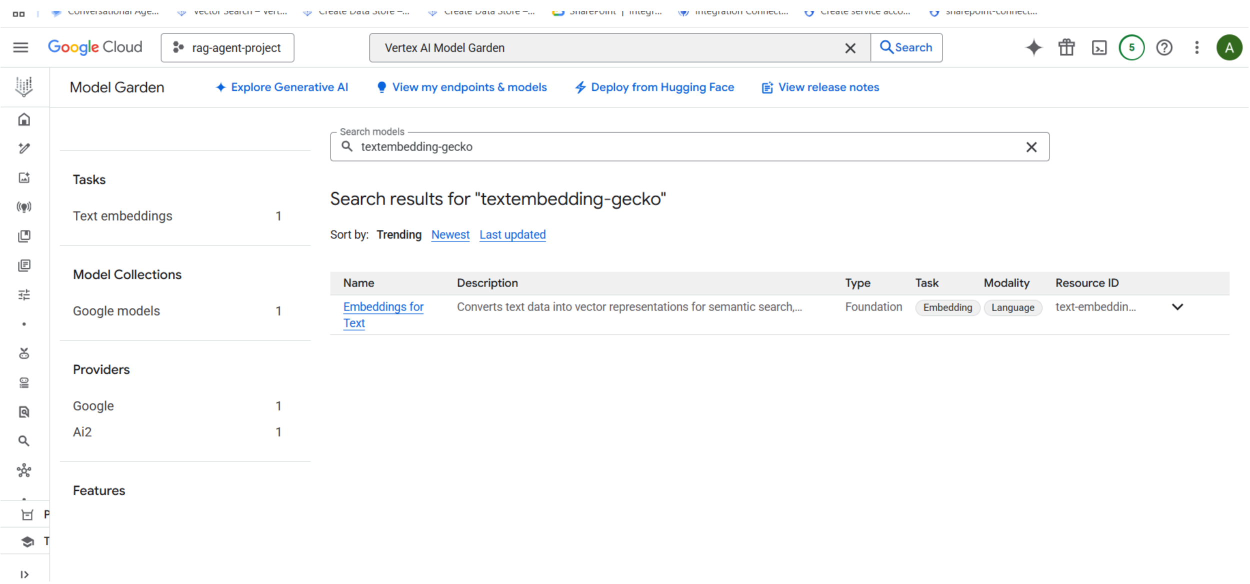
my-agentspace-bucket

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	Soft Delete

Buckets > my-agentspace-bucket

☐	Name	Size	Type	Created	Storage class
☐	An_Effective_Retrieval_Method_to...	270.6 KB	application/pdf	Jul 6, 2025, 2:48:23 PM	Standard
☐	NeurIPS-2020-retrieval-augmented...	946.1 KB	application/pdf	Jul 6, 2025, 2:48:25 PM	Standard
☐	borgeaud22a.pdf	986.9 KB	application/pdf	Jul 6, 2025, 2:48:26 PM	Standard
☐	paper1.pdf	6.3 MB	application/pdf	Jul 6, 2025, 2:48:40 PM	Standard
☐	paper2.pdf	199.2 KB	application/pdf	Jul 6, 2025, 2:48:30 PM	Standard
☐	paper3.pdf	11 MB	application/pdf	Jul 6, 2025, 2:48:43 PM	Standard
			application/pdf	Jul 6, 2025, 2:48:50 PM	Standard

Now viewing project "rag-agent-project" in organization "No organization"



textembedding-gecko

Tasks	
Text embeddings	1

Model Collections	
Google models	1

Providers	
Google	1
Ai2	1

Features

Search results for "textembedding-gecko"

Sort by: **Trending** Newest Last updated

Name	Description	Type	Task	Modality	Resource ID
Embeddings for Text	Converts text data into vector representations for semantic search,...	Foundation	Embedding	Language	text-embeddin...

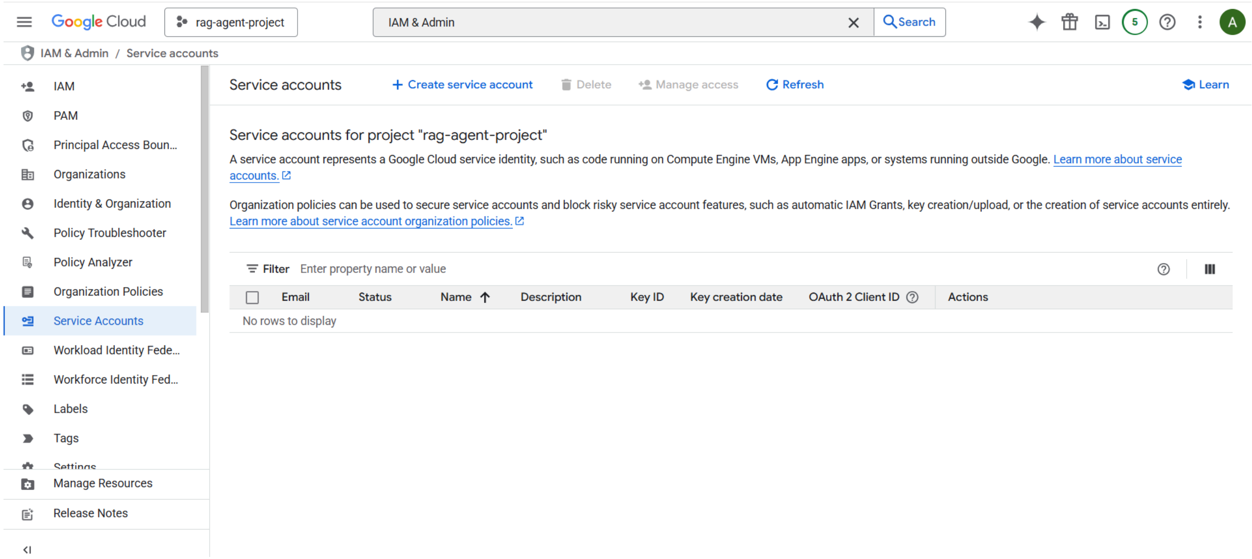
https://cloud.google.com/python/docs/reference/aiplatform/1.28.0/vertexai.language_models.TextEmbeddingModel#:~:text=Dismiss%20View,:%20str)%20%2D%3E%20vertexai.

You obtain the `/path/to/your/service_account.json` file when you create and download a service account key from your Google Cloud project.

Here's a step-by-step guide:

1. **Go to the Google Cloud Console:**
    - Open your web browser and navigate to https://console.cloud.google.com/.
    - Make sure you select the correct Google Cloud project where you want to use Vertex AI.
2. **Navigate to IAM & Admin -> Service Accounts:**
    - In the Google Cloud Console, use the navigation menu (usually on the left) and go to "IAM & Admin" > "Service Accounts."
3. **Create a New Service Account (if you don't have one):**
    - Click on "+ CREATE SERVICE ACCOUNT" at the top of the page.
    - **Service account name:** Give it a descriptive name (e.g., `vertex-ai-user`, `my-app-service-account`).
    - **Service account ID:** This will be automatically generated based on the name.
    - **Service account description:** (Optional) Add a brief description.
    - Click "CREATE AND CONTINUE."
4. **Grant Permissions/Roles:**
    - This is the crucial step for giving your service account the necessary access to Vertex AI.
    - In the "Grant this service account access to project" section, click on the "Select a role" dropdown.
    - Search for and add the following roles (at a minimum):
        - `Vertex AI User`
        - `Service Usage Consumer` (this allows the service account to use APIs enabled in your project)
    - Depending on your specific needs, you might also consider:
        - `Vertex AI Developer` (broader permissions for development)
        - `Storage Object Viewer` or `Storage Object Creator` (if your application needs to read/write from Cloud Storage buckets)
    - **Principle of Least Privilege:** Always grant only the necessary permissions. Avoid giving roles like "Owner" or "Editor" unless absolutely required, especially in production environments.
    - Click "CONTINUE."
5. **Grant users access to this service account (Optional):**
    - You can skip this step unless you need to grant other users or service accounts the ability to *impersonate* this new service account.
    - Click "DONE."
6. **Create and Download the JSON Key:**
    - Now that the service account is created, you need to generate a key for it.

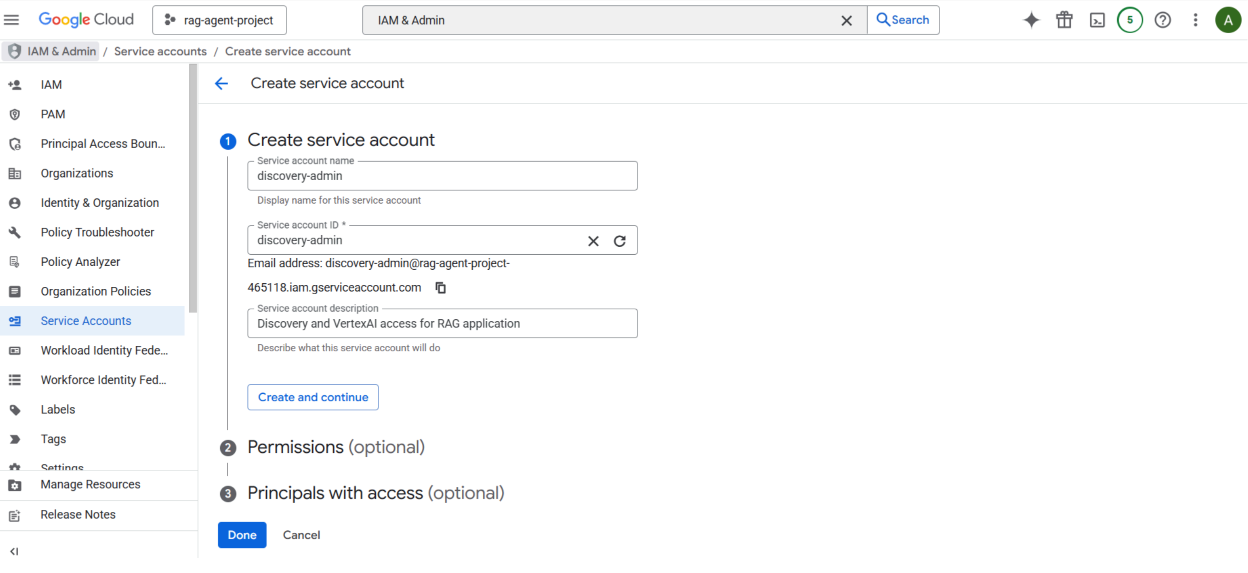
    - On the Service Accounts page, find the service account you just created.
    - Click on the three vertical dots (Actions menu) under the "Actions" column for your service account.
    - Select "Manage keys."
    - Click on "ADD KEY" > "Create new key."
    - Select "JSON" as the key type.
    - Click "CREATE."
7. Your browser will automatically download a JSON file. This file contains the private key for your service account and is what you'll use for authentication.
8. **Store the JSON Key Securely:**
    - **This file is highly sensitive.** Treat it like a password. Anyone who has this file can authenticate as your service account and access resources it has permissions for.
    - **Do NOT** commit it to version control (Git, etc.).
    - Store it in a secure location on your machine or server where your code will run.
9. **Set the Environment Variable:**
    - Once you have the JSON file, replace `/path/to/your/service_account.json` in your code with the actual path to the downloaded JSON file on your system.
10. For example, if you downloaded `m`



Service accounts for project "rag-agent-project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.



Create service account

1 Create service account

Service account name: discovery-admin

Display name for this service account

Service account ID *: discovery-admin

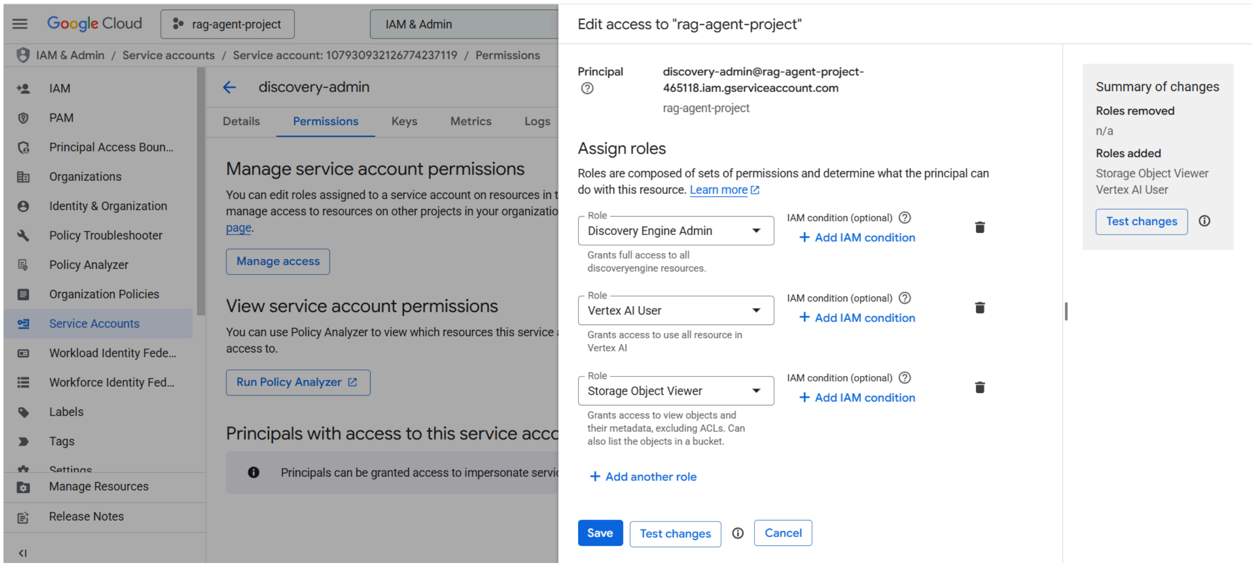
Email address: discovery-admin@rag-agent-project-465118.iam.gserviceaccount.com

Service account description: Discovery and VertexAI access for RAG application

Describe what this service account will do

2 Permissions (optional)

3 Principals with access (optional)



Manage service account permissions

You can edit roles assigned to a service account on resources in t manage access to resources on other projects in your organizatio page.

View service account permissions

You can use Policy Analyzer to view which resources this service access to.

Principals with access to this service acc

Principals can be granted access to impersonate servic

Edit access to "rag-agent-project"

Principal: discovery-admin@rag-agent-project-465118.iam.gserviceaccount.com
rag-agent-project

Assign roles

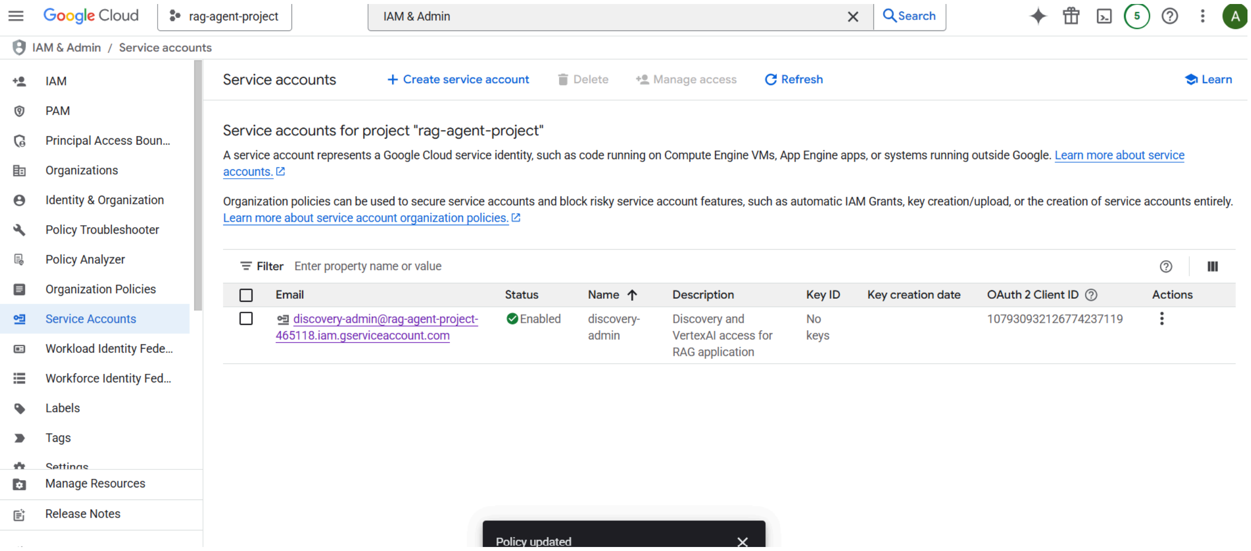
Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role: Discovery Engine Admin
Grants full access to all discoveryengine resources.
IAM condition (optional)

Role: Vertex AI User
Grants access to use all resource in Vertex AI
IAM condition (optional)

Role: Storage Object Viewer
Grants access to view objects and their metadata, excluding ACLs. Can also list the objects in a bucket.
IAM condition (optional)

Summary of changes
Roles removed
n/a
Roles added
Storage Object Viewer
Vertex AI User



Service accounts

Service accounts for project "rag-agent-project"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.

Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.

☐	Email	Status	Name ↑	Description	Key ID	Key creation date	OAuth 2 Client ID	Actions
☐	discovery-admin@rag-agent-project-465118.iam.gserviceaccount.com	✓ Enabled	discovery-admin	Discovery and VertexAI access for RAG application	No keys		107930932126774237119	⋮

Policy updated

Perfect — you are **almost there**, but notice it says:

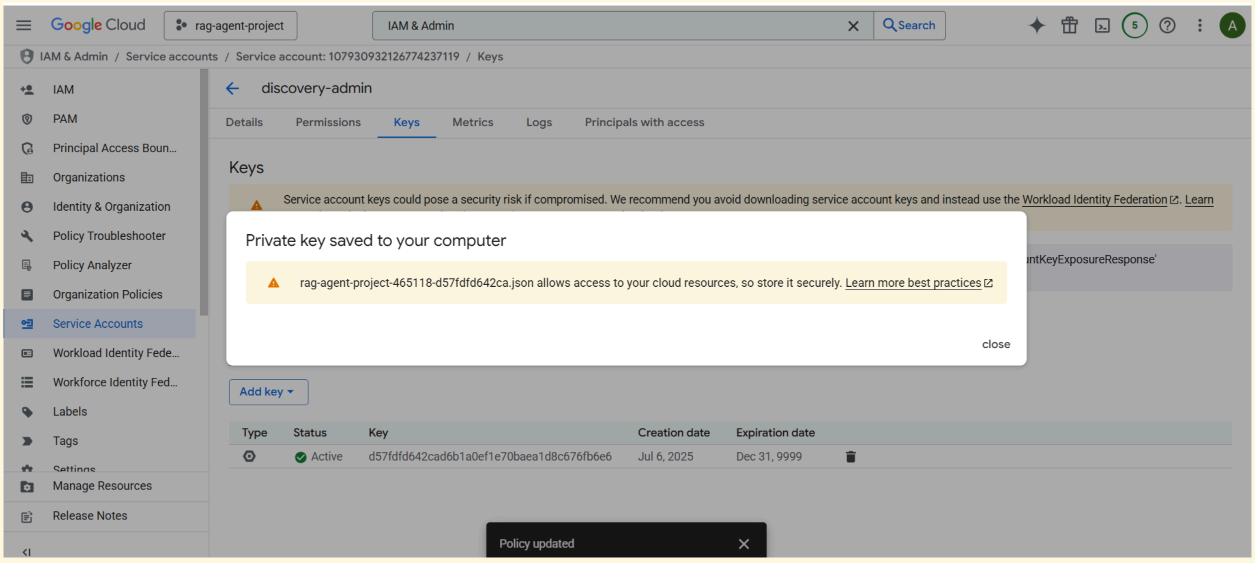
> **Key ID** → *No keys*

✅ That means you still need to generate a key file so you can authenticate from Colab or local.

Do this next (exact steps):

1️⃣Click the **three dots** on the right of `discovery-admin@...`
2️⃣Select **Manage keys**
3️⃣Click **Add Key** → **Create new key** → **JSON**
4️⃣Download the JSON key file (for example: `discovery-admin-key.json`)
5️⃣Upload that JSON file to Colab
6️⃣In Colab, set:

rag-agent-project-465118-d57fdfd642ca.json



Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use the Workload Identity Federation. Learn

ıntKeyExposureResponse'

Private key saved to your computer

rag-agent-project-465118-d57fdfd642ca.json allows access to your cloud resources, so store it securely. Learn more best practices

Type	Status	Key	Creation date	Expiration date
	Active	d57fdfd642cad6b1a0ef1e70baea1d8c676fb6e6	Jul 6, 2025	Dec 31, 9999

Policy updated

🚀 Awesome — let's go step by step to **push your chunks to Discovery**.

✅ **High-level steps:**

1. You already have `all_chunks` with text + metadata + embeddings
2. Now we'll use the Discovery Engine Data API to push these chunks
3. Then you'll verify the chunks are indexed and ready for RAG