Quick tip: Review the prerequisites before you run the lab

End Lab 00:18:12

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

Open GCP Console

GCP Username

student-00-a606a398702d

GCP Password

9cFLTiefsfUZ

GCP Project

qwiklabs-gcp-02-9dfd5d3!

AI Applications Data Store Status Checker

Lab 1 hour No cost Intermediate

★★★★☆

This lab may incorporate AI tools to support your learning.

100/100

Previous Next

Overview

What is a Data Store?

Data Stores are the fundamental building block behind AI Applications.

Data Store Indexing Time

With each website or set of documents added, the Data Store needs to index the site and/or documents in order for them to be searchable. This can take up to 4 hours for new data store web content to be indexed.

Using the attached example notebook, you can query your Data Store ID to see if indexing is complete. Once complete, you can additionally use the notebook to search your Data Store for specific pages or documents.

Objectives

This lab uses the Cloud Discovery Engine API to check a Data Store for indexed docs.

This lab utilizes the `google-cloud-discoveryengine` Python library and allows the

- Check Indexing Status of given Data Store ID.
- List all documents in a given Data Store ID.
- List all indexed URLs for a given Data Store ID
- Search all indexed URLs for a specific URL within a given Data Store ID.

Setup and requirements

Before you click the Start Lab button

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

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new,

What you need

To complete this lab, you need:

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

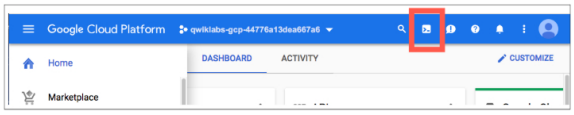
Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

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

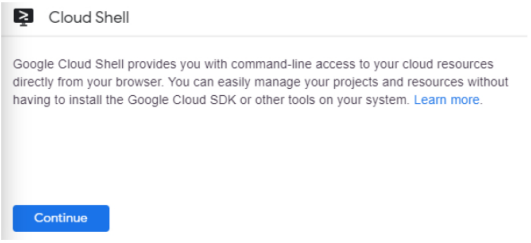
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

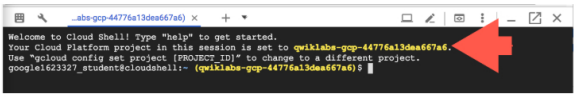
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your PROJECT_ID



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

(Example output)

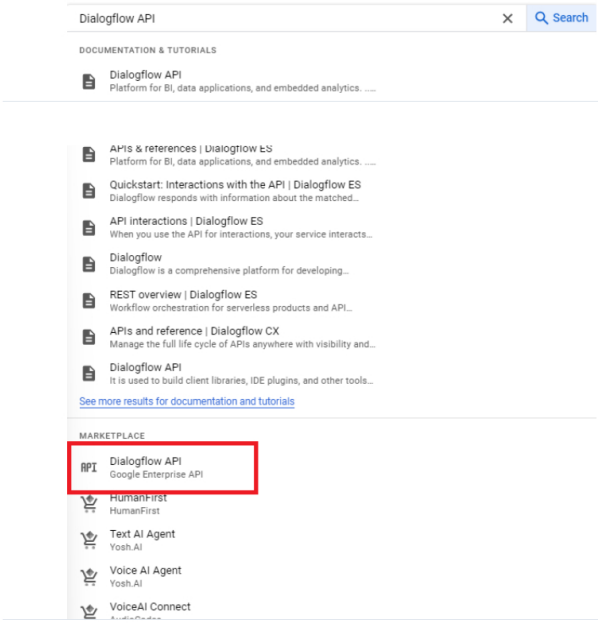
```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

Task 1. Enable APIs

In this task, you need to enable the Dialogflow API.

To enable the Dialogflow API, follow these steps:

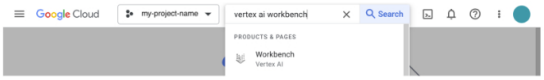
1. Type **Dialogflow API** into the top search bar of the Google Cloud console, choose the selected result as shown below, and continue.



2. Click the **Enable** button.

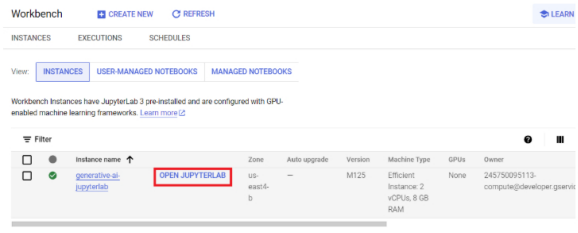
Task 2. Install prerequisites

1. In the Google Cloud console, navigate to **Vertex AI Workbench**. In the top search bar of the Google Cloud console, enter **Vertex AI Workbench**, and click on the first result.



2. Under **Instances**, click on **Open JupyterLab** for **generative-ai-jupyterlab** notebook.

The JupyterLab will run in a new tab.



3. On the **Launcher**, under **Notebook**, click on **Python 3** to open a new python notebook.

4. In the first cell, enter the following command to install the latest version of Google Cloud Discovery Engine. To run the command, hit **SHIFT+ENTER**, or click on play button at the top.

```
pip install --upgrade google-cloud-discoveryengine humanize
```

Output:

```
                                        832.0/832.0 kB 14.3 MB/s eta 0:00:00a 0:00:01
Downloading humanize-4.8.0-py3-none-any.whl (117 kB)
                                        117.1/117.1 kB 22.0 MB/s eta 0:00:00
Installing collected packages: humanize, google-cloud-discoveryengine
```

```
Successfully installed google-cloud-discoveryengine-0.11.2 humanize-4.8.0
Note: you may need to restart the kernel to use updated packages.
```

5. Restart the kernel by clicking the refresh button at the top, followed by clicking **Restart** button.

Task 3. Helper Methods

1. Run the following code in the next cell to setup the helper methods.

```
import humanize
```

```
from typing import List, Optional

from google.api_core.client_options import ClientOptions
from google.cloud import discoveryengine_v1beta as
discoveryengine


def _call_list_documents(
    project_id: str, location: str, datastore_id: str,
page_token: Optional[str] = None
) -> discoveryengine.ListDocumentsResponse:
    """Build the List Docs Request payload."""
    client_options = (
        ClientOptions(
            api_endpoint=f"{location}-
discoveryengine.googleapis.com")
        if location != "global"
        else None
    )
    client = discoveryengine.DocumentServiceClient(
        client_options=client_options)
```

```
            project_id, location, datastore_id, "default_branch"
        ),
        page_size=1000,
        page_token=page_token,
    )

    return client.list_documents(request=request)


def list_documents(
    project_id: str, location: str, datastore_id: str,
rate_limit: int = 1
) -> List[discoveryengine.Document]:
    """Gets a list of docs in a datastore."""

    res = _call_list_documents(project_id, location,
datastore_id)

    # setup the list with the first batch of docs
    docs = res.documents

    while res.next_page_token:
```

```
        res = _call_list_documents(
            project_id, location, datastore_id,
res.next_page_token
        )
        docs.extend(res.documents)

    return docs


def list_indexed_urls(
    docs: Optional[List[discoveryengine.Document]] = None,
    project_id: str = None,
    location: str = None,
    datastore_id: str = None,
) -> List[str]:
    """Get the list of docs in data store, then parse to only
urls."""
    if not docs:
        docs = list_documents(project_id, location, datastore_id)
    urls = [doc.content.uri for doc in docs]
```

```
def search_uri(uris: List[str], uri: str) -> None:
    """Searches a url in a list of urls."""
    for item in urls:
        if url in item:
            print(item)


def search_doc_id(
    doc_id: str,
    docs: Optional[List[discoveryengine.Document]] = None,
    project_id: str = None,
    location: str = None,
    datastore_id: str = None,
) -> None:
    """Searches a doc_id in a list of docs."""
    if not docs:
        docs = list_documents(project_id, location, datastore_id)

    doc_found = False
    for doc in docs:
        if doc.parent_document_id == doc_id:
```

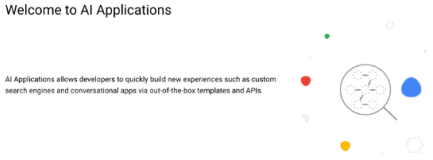
```
    if not doc_found:
        print(f"Document not found for provided Doc ID:
`{doc_id}`")


def estimate_data_store_size(
    urls: Optional[List[str]] = None,
    docs: Optional[List[discoveryengine.Document]] = None,
    project_id: str = None,
    location: str = None,
    datastore_id: str = None,
) -> None:
    """For Advanced Website Indexing data stores only."""
    if not urls:
        if not docs:
            docs = list_documents(project_id, location,
datastore_id)
        urls = list_indexed_urls(docs=docs)

    # Filter to only include website urls.
    urls = list(filter(lambda x: re.search(r"https?://", x),
```

```
        print(
            "No urls found. Make sure this data store is for
websites with advanced indexing."
        )
        return

    # For website indexing, each page is calculated as 500KB.
    size = len(urls) * 500_000
    print(f"Estimated data store size:
{humanize.naturalsize(size)}")


PENDING_MESSAGE = """
No docs found.\n\nIt\'s likely one of the following issues: \n
[1] Your data store is not finished indexing. \n  [2] Your data
store failed indexing. \n  [3] Your data store is for website
data without advanced indexing.\n\n
If you just added your data store, it can take up to 4 hours
before it will become available.
"""
```

Task 4. User Inputs

Creating a new data store.

1. Navigate to AI Applications and click on **CONTINUE AND ACTIVATE THE API** button.

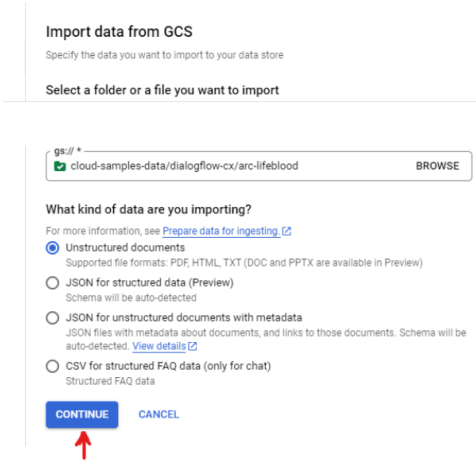


2. In the **Create App** page, select **Chat** as an App type.

3. For **Company Name** enter `Cymbal`. For **Agent Name** enter `cymbalagent` and click **CONTINUE**.

4. In the **Data Stores** page, click **+ CREATE NEW DATA STORE**.

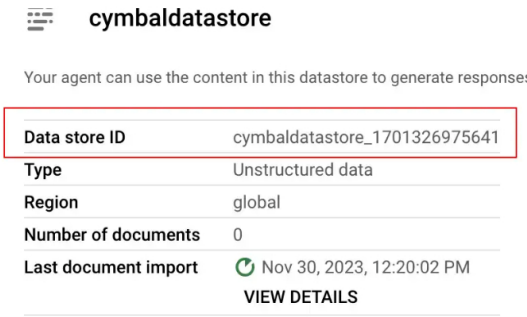
5. Select **Cloud Storage**, and enter the following Google Cloud Storage location `cloud-samples-data/dialogflow-cx/arc-lifeblood` to add the folder. Then select **Unstructured documents** under **What kind of data are you importing?** section and finally click **CONTINUE**.



6. For Data store name, enter **cymbaldatastore** and click **CREATE**. This creates a Data store.

7. Finally on the App's Data page, select **cymbaldatastore** and click on **CREATE**.

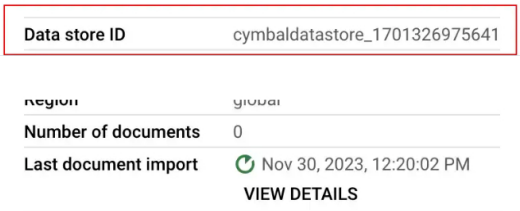
8. Click on `cymbaldatastore` and note down the Data store ID.



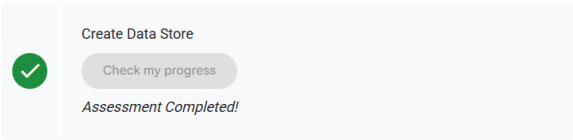
9. Alternatively, you can find the datastore_id by from **Navigation Menu** ☰ **> AI Applications**.

10. For `cymbalagent`, click on **View** under **Connected data stores**.

11. Finally, click on `cymbaldatastore` to view the Data store ID. Note down the Data store ID.



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



Let's use the `list_documents` method, to check if the data store has finished indexing.

1. Back in the Jupyter notebook, run the following command in the next cell and replace the `YOUR_DATA_STORE_ID` with the Data store ID noted down in earlier task.

```
PROJECT = !gcloud config get-value project
project_id  = PROJECT[0]
location = "global"  # Options: "global", "us", "eu"
datastore_id = "YOUR_DATA_STORE_ID"
```

2. Run the following code snippet in the next notebook cell.

```
docs = list_documents(project_id, location, datastore_id)

if len(docs) == 0:
    print(PENDING_MESSAGE)
else:
    """
    Your indexing is complete.\n
    Your index contains {len(docs)} documents.
    """
    print(SUCCESS_MESSAGE)
```

Output:

```
Success!

Your indexing is complete.

Your index contains 79 documents.
```

Note If you just added your data store, it can take up to 10 to 15 mins before it will become available.

Task 6. List Documents

Let's list all the documents for a given Data Store ID.

1. Run the following code in the next notebook cell.

```
docs = list_documents(project_id, location, datastore_id)
docs[0]
```

Output:

```
name:
```

```
"projects/566064737691/locations/global/collections/default_collection/data
id: "01c3140622cfed9a86572e550ed049a0"
schema_id: "default_schema"
struct_data {
}
parent_document_id: "01c3140622cfed9a86572e550ed049a0"
```

```
  uri: "gs://cloud-samples-data/dialogflow-cx/arc-
lifeblood/testing.html"
}
```

Task 7. Search Data Store by Doc ID

Let's search through all docs in a given Data Store and find a specific Doc ID.

In the following command, replace the **placeholder_document_id** with the value of **parent_document_id** from the last output.

```
document_id = "placeholder_document_id"
search_doc_id(document_id, docs=docs)
```

Output:

```
name:
"projects/566064737691/locations/global/collections/default_collection/data
id: "01c3140622cfed9a86572e550ed049a0"
schema_id: "default_schema"
struct_data {
}
parent_document_id: "01c3140622cfed9a86572e550ed049a0"
content {
  mime_type: "text/html"
  uri: "gs://cloud-samples-data/dialogflow-cx/arc-
lifeblood/testing.html"
}
```

Task 8. List Indexed URLs

1. Run the following code snippet in the next notebook cell, to retrieve a indexed URL.

```
urls = list_indexed_urls(docs=docs)
urls[0]
```

Output:

```
'gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/testing.html'
```

Task 9. Search Indexed URLs

1. Run the following code snippet in the next notebook cell, to search for a specific URL within a list of indexed URLs.

```
search_url(urls, "gs://cloud-samples-data/dialogflow-cx/arc-
lifeblood")
```

Output:

```
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/testing.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/strategy.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/making-blood-
```

```
components.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/making-your-
donation.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/products.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/high-ferritin.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/forms.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/blood-for-
transfusion.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/how-you-can-give-
```

```
immunogenetics-services.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/blood.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/prepare-and-
aftercare.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-strategy.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-history.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-services.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/about.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/iron-deficiency.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/iron-health.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/donor-centre.html
```

2. Run the below code in the notebook cell, and should have similar output.

```
search_url(urls, "dialogflow-cx")
```

output

```
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/testing.html
```

```
components.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/making-your-
donation.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/products.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/high-ferritin.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/forms.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/blood-for-
transfusion.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/how-you-can-give-
life.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/transplantation-
immunogenetics-services.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/blood.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/prepare-and-
aftercare.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-strategy.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-history.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/our-services.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/about.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/iron-deficiency.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/iron-health.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/donor-centre.html
```

```
transfusion.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/sexual-activity.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/learn-about-
blood.html
gs://cloud-samples-data/dialogflow-cx/arc-lifeblood/avoid-a-
transfusion.html
......
so on
```

Congratulations!

You've successfully learned to utilize Vertex AI to verify Data store indexing status, document listing, and URL searches.

Manual Last Updated April 03, 2025

Lab Last Tested April 03, 2025