quantiphi
Solving What Matters

Health Canada

Health Canada DocAI Warehouse POC

RUN BOOK

Author: Quantiphi Team

Prepared for: Health Canada

Document type: Run Book

Date: 06th Apr 2023

Table of Contents

1. Solution Architecture

The main component of this solution is Google Cloud's Document AI Warehouse (DocAI Warehouse), which is a document management solution to store, manage, govern, and search documents at scale.

- The user uploads the pdf to the input GCS storage to trigger the Cloud Function.

- Cloud Function separates the first page of the document and sends it to the Document AI CDE processor for extraction of the 6 labeled entities and remaining pages to the OCR processor to extract raw text .

- The PDF is then added in the DocAI Warehouse along with its extracted entities as properties.

- The PDFs are displayed in the Document AI Warehouse and users can conduct keyword or faceted search via filtering document properties.

Google Cloud Platform (GCP) services used for this solution are as follows:

Document AI Warehouse
It is used to search, store, govern, and manage documents and their AI-extracted data and metadata in a single platform.Document AI Warehouse provides a high-throughput pipeline to transfer, extract using DocAI and ingest or index data from Google Cloud Storage and from on-premises file systems via Storage Transfer Service

Cloud Functions

Google Cloud Functions is a serverless execution environment for building and connecting cloud services. With Cloud Functions you write simple, single-purpose functions that are attached to events emitted from your cloud infrastructure and services.

Google Cloud Storage

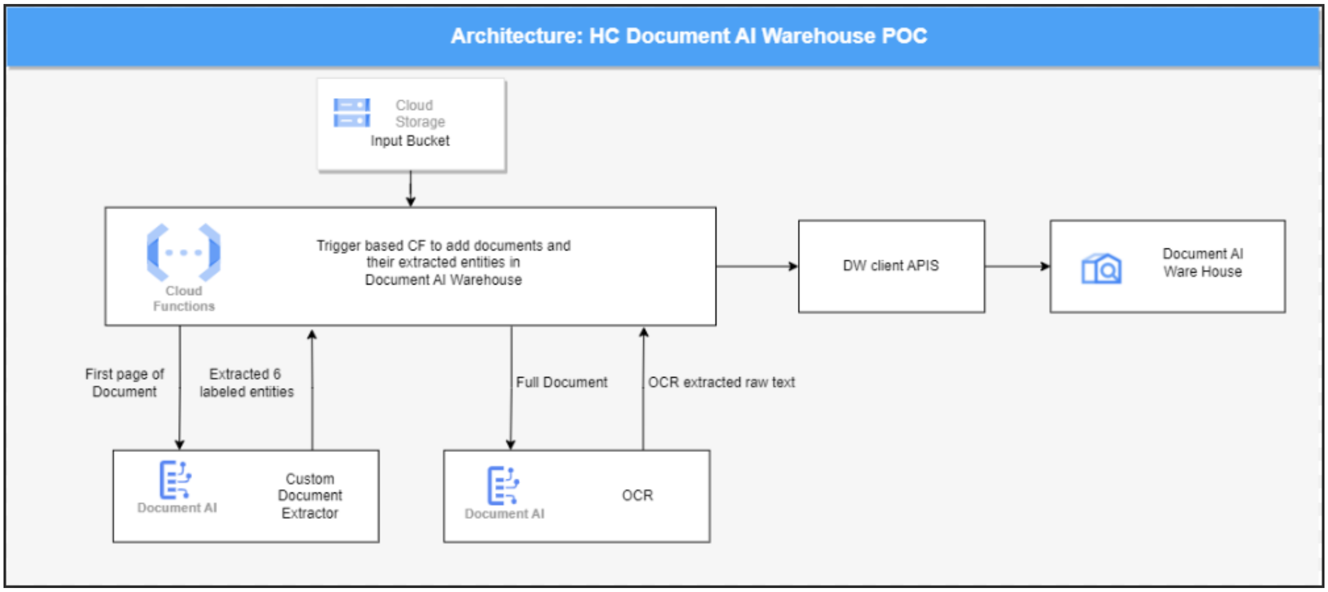
Google Cloud Storage (GCS) allows world-wide storage and retrieval of any amount of data at any time. Cloud Storage can be used for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

Document AI

- Document AI OCR
  This processor allows you to identify and extract text, including handwritten text, from documents in over 200 languages.

- Document AI Custom Document Extractor (CDE)
  This processor helps to build and train customized entity extractor for new document types for which no pre-trained processors are available

The Solution Architecture Diagram is as follows:


Architecture: HC Document AI Warehouse POC
Cloud Storage
Input Bucket
Cloud Functions
Trigger based CF to add documents and their extracted entities in Document AI Warehouse
DW client APIS
Document AI Ware House
First page of Document
Extracted 6 labeled entities
Full Document
OCR extracted raw text
Document AI
Custom Document Extractor
Document AI
OCR

1.1 Architecture Setup using Terraform

Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure in a declarative manner. We are using Terraform code to create the components of the Inferencing systems in the architecture diagram. Note that this is a one time task, when you are starting to build out the POC architecture.

- Terraform code link

Clone the repo using this command

1. Open Cloud shell

2. Clone the repo using the following command:

```
git clone https://gitlab.qdatalabs.com/applied-ai/canada/healthcanada/hc-docwarehouse.git
```

1.1.1 Create GCP Project and backend bucket

To start deploying infrastructure on the GCP platform, we need a GCP project and a GCS bucket for saving the Terraform remote state, Terraform should use a remote state to prevent race conditions and stale Terraform plans from corrupting/mismanaging resources..

To create a GCP project we can start Google Cloud Shell and run the following command.

```
gcloud projects create PROJECT_ID
```

To create GCS bucket for terraform which will store the terraform state file in the project, run the following command

```
gcloud storage buckets create gs://terraform_state_BUCKET_NAME
--project=PROJECT_ID  --location=northamerica-northeast1
```

Reference link for project creation
https://cloud.google.com/resource-manager/docs/creating-managing-projects

Reference link for cloud storage bucket creation
https://cloud.google.com/storage/docs/creating-buckets

1.1.2 Create Terraform runner service-account

```
gcloud iam service-accounts create SA_NAME --description="DESCRIPTION"
--display-name="DISPLAY_NAME"
```

To deploy infrastructure using Terraform to GCP, we can authenticate either using a GCP user account or, preferably, by using a service account. Especially in the case of deploying Terraform through a pipeline, it is required to have the means to authenticate as a service account such as by storing the service account JSON key file as a secret, or by using service account impersonation where we use short-lived credentials to authenticate calls to Google Cloud APIs. The user who will run the Terraform requires the Service Account Token Creator IAM role.

Reference link for service account creation
https://cloud.google.com/iam/docs/service-accounts-create#iam-service-accounts-create-gcloud

IAM roles needed for Terraform runner service-account are listed below

- Cloud Functions Admin
- Document AI Administrator
- Project IAM Admin
- Role Administrator
- Service Management Administrator
- Storage Admin
- Editor

gcloud command to add IAM Roles to terraform runner service account

```
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/cloudfunctions.admin"
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/roles/documentai.admin"
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/resourcemanager.projectIamAdmin"
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/roles/iam.roleAdmin"
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/roles/servicemanagement.admin"
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/roles/storage.admin
gcloud projects add-iam-policy-binding PROJECT_ID --member="serviceAccount:terraform_runner_SA_NAME@PROJECT_ID.iam.gserviceaccount.com"  --role="roles/roles/editor
```

1.1.3 Enable The Apis

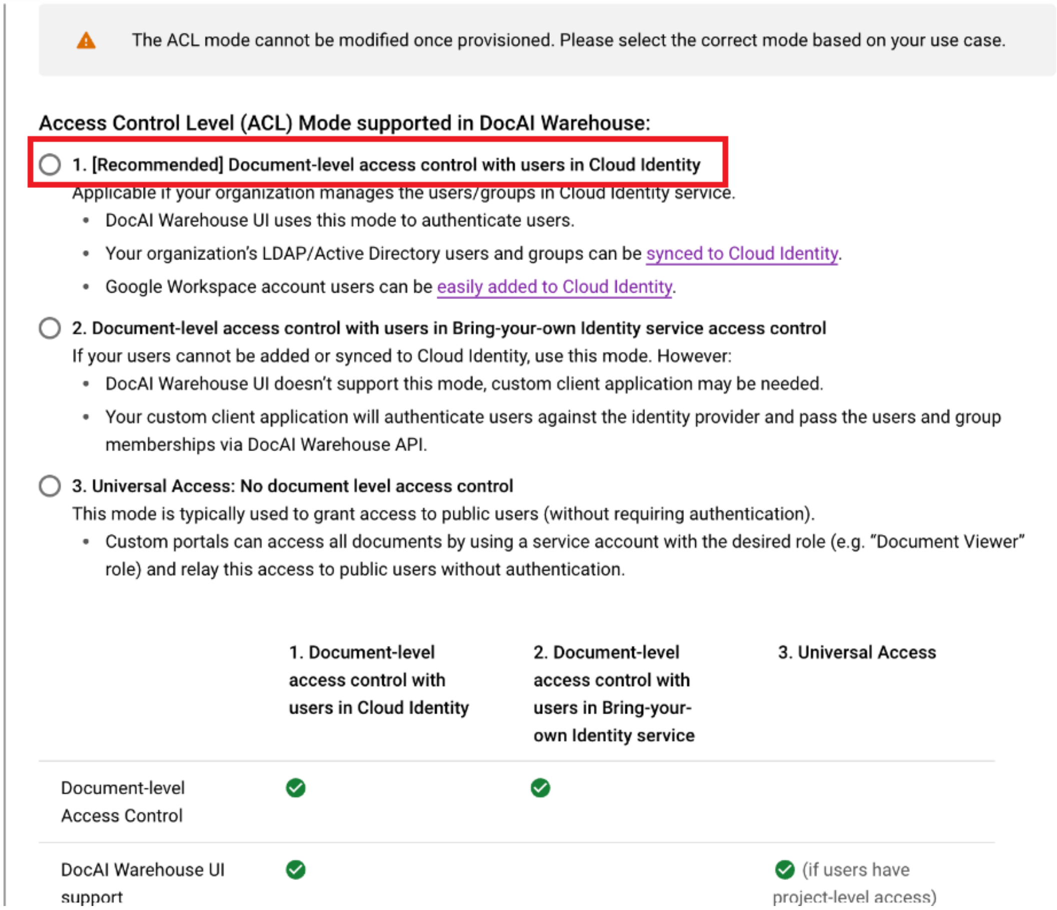
Go to the API and services from the cloud console click on enable API and services search below APIs and click on enable

- cloudresourcemanager.googleapis.com
- Serviceusage.googleapis.com
- App Engine
- IAM Service Account Credentials API

Note: These APIs are to be created manually

1.1.4 Document AI Warehouse creation

Refer to this link to create an instance of Document AI Warehouse in the project
Note:- Choose the ACL Mode as [Recommended] Document-level access control with users in Cloud Identity



⚠ The ACL mode cannot be modified once provisioned. Please select the correct mode based on your use case.

Access Control Level (ACL) Mode supported in DocAI Warehouse:

○ **1. [Recommended] Document-level access control with users in Cloud Identity**
Applicable if your organization manages the users/groups in Cloud Identity service.

- DocAI Warehouse UI uses this mode to authenticate users.
- Your organization's LDAP/Active Directory users and groups can be synced to Cloud Identity.
- Google Workspace account users can be easily added to Cloud Identity.

○ **2. Document-level access control with users in Bring-your-own Identity service access control**
If your users cannot be added or synced to Cloud Identity, use this mode. However:

- DocAI Warehouse UI doesn't support this mode, custom client application may be needed.
- Your custom client application will authenticate users against the identity provider and pass the users and group memberships via DocAI Warehouse API.

○ **3. Universal Access: No document level access control**
This mode is typically used to grant access to public users (without requiring authentication).

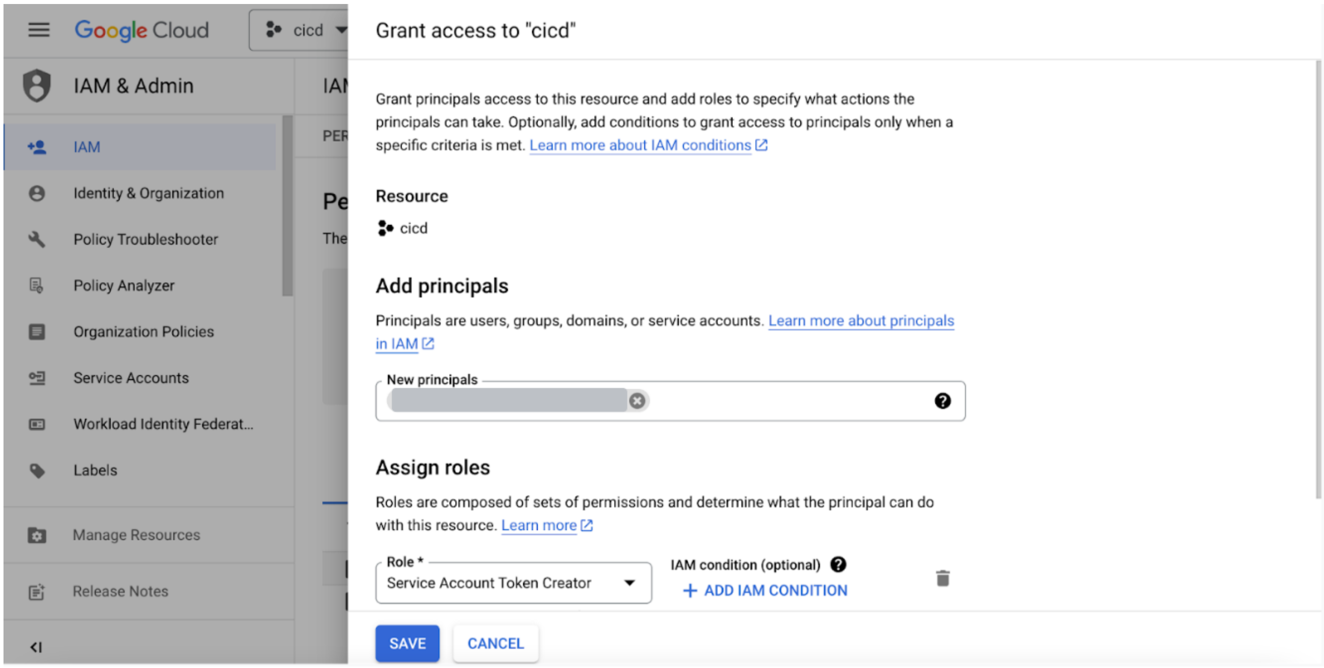
- Custom portals can access all documents by using a service account with the desired role (e.g. "Document Viewer" role) and relay this access to public users without authentication.

	1. Document-level access control with users in Cloud Identity	**2. Document-level access control with users in Bring-your-own Identity service**	**3. Universal Access**
Document-level Access Control	✓	✓	
DocAI Warehouse UI support	✓		✓ (if users have project-level access)

1.1.5 Grant SA token creator role

Provide service Account Token Creator role to the user who will run the Terraform requires the Service Account Token Creator IAM role.

Go to the GCP IAM section click on grant access and provide Service Account Token Creator role The user who will run the Terraform



Grant access to "cicd"

Grant principals access to this resource and add roles to specify what actions the principals can take. Optionally, add conditions to grant access to principals only when a specific criteria is met. Learn more about IAM conditions

Resource

cicd

Add principals

Principals are users, groups, domains, or service accounts. Learn more about principals in IAM

New principals: [redacted]

Assign roles

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

Role *: Service Account Token Creator

IAM condition (optional)

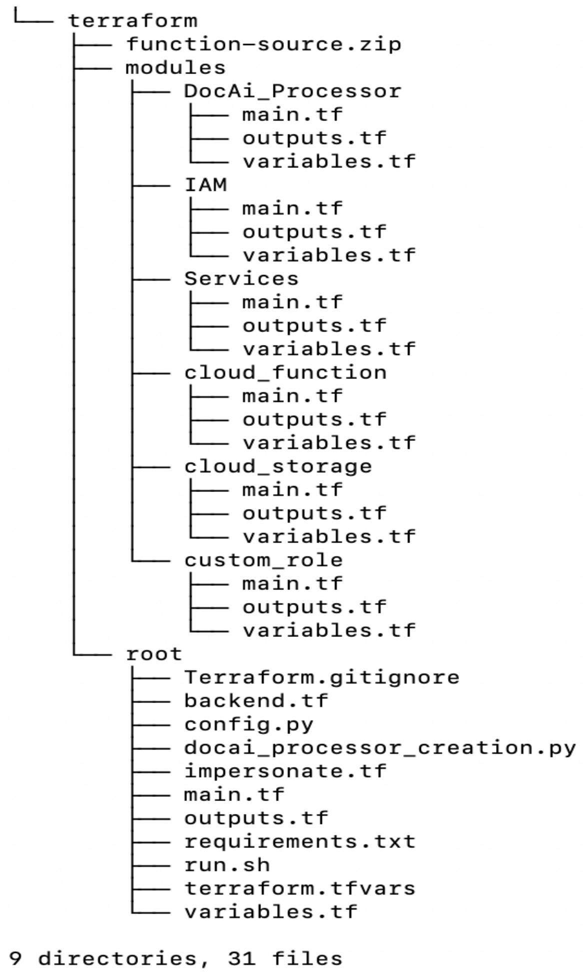
+ ADD IAM CONDITION

SAVE CANCEL

1.3 Deploying the Terraform code

Terraform structure

Note: function-source.zip contains the ML code that will reside in the Cloud function named **HC_cloud_fuction.**



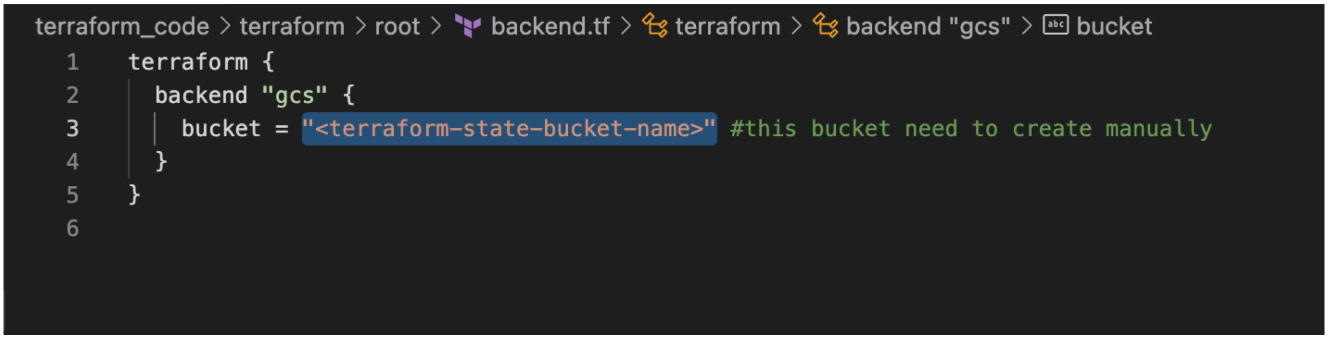
```
└── terraform
    ├── function-source.zip
    ├── modules
    │   ├── DocAi_Processor
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   ├── IAM
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   ├── Services
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   ├── cloud_function
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   ├── cloud_storage
    │   │   ├── main.tf
    │   │   ├── outputs.tf
    │   │   └── variables.tf
    │   └── custom_role
    │       ├── main.tf
    │       ├── outputs.tf
    │       └── variables.tf
    └── root
        ├── Terraform.gitignore
        ├── backend.tf
        ├── config.py
        ├── docai_processor_creation.py
        ├── impersonate.tf
        ├── main.tf
        ├── outputs.tf
        ├── requirements.txt
        ├── run.sh
        ├── terraform.tfvars
        └── variables.tf

9 directories, 31 files
```

1.3.1 Provide the values for Terraform variables

Go inside the terraform folder you will find root folder Inside the root folder there is a backend.tf file goes to this file you can find variable

bucket = "<terraform-state-bucket-name>"

replace **"<terraform-state-bucket-name>"** with your manually created bucket for terraform state bucket


terraform_code > terraform > root > backend.tf > terraform > backend "gcs" > bucket

```
terraform {
  backend "gcs" {
    bucket = "<terraform-state-bucket-name>" #this bucket need to create manually
  }
}
```

Go inside the terraform.tfvars folder and provide values for these variables

```
/* impersonate terraform service account */
service_account = "<terraform runner service_account>"

/* project_id */
project_id = "<project_id>"

/* Cloud Storage */
name     = "processor_training_bucket"
location = "us-central1"

/* IAM */
mode     = "additive"
projects = ["<project_id>"]
bindings = {
 "roles/bigquery.readSessionUser" = [
   "serviceAccount:<Doc Ai-service-account>",
 ]
 "roles/bigquery.user" = [
   "serviceAccount:<Doc Ai-service-account>",
 ]
```

```
"roles/contentwarehouse.admin" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>",
]
"roles/contentwarehouse.documentAdmin" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>",
]
"roles/contentwarehouse.documentCreator" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>",
]
"roles/contentwarehouse.serviceAgent" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/contentwarehouse.documentViewer" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/documentai.admin" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>"
]
"roles/contentwarehouse.documentAdmin" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/secretmanager.secretAccessor" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/iam.serviceAccountTokenCreator" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/storage.admin" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>"
```

```
]
"roles/storage.objectViewer" = [
  "serviceAccount:<Doc Ai-service-account>",
]
"roles/aiplatform.admin" = [
  "serviceAccount:<Doc Ai-service-account>",
  "user:<user_id>",
]
"roles/bigquery.admin" = [
  "user:<user_id>",
]
"roles/bigquery.dataEditor" = [
  "user:<user_id>",
]
"roles/errorreporting.admin" = [
  "user:<user_id>",
]
"roles/logging.admin" = [
  "user:<user_id>",
]
"roles/notebooks.admin" = [
  "user:<user_id>",
]
"roles/notebooks.viewer" = [
  "user:<user_id>",
]
"roles/iam.serviceAccountUser" = [
  "user:<user_id>",
]
"roles/contentwarehouse.documentAdmin" = [
  "user:<user_id>",
]
"roles/storage.admin" = [
  "user:<user_id>"
```

```
 ]
 "roles/storage.objectAdmin" = [
   "user:<user_id>",
 ]
}

/* custom role */
target_level = "project"
target_id    = "<project_id>"
role_id      = "warehouse_custom_role"
title        = "doc ai warehousecustom custom role "
description  = "custom role for doc ai warehouse"
permissions = ["contentwarehouse.documentSchemas.create",
"contentwarehouse.documentSchemas.delete", "contentwarehouse.documentSchemas.get",
"contentwarehouse.documentSchemas.list", "contentwarehouse.documentSchemas.update",
"contentwarehouse.documents.create", "contentwarehouse.documents.delete",
"contentwarehouse.documents.get", "contentwarehouse.documents.getIamPolicy",
"contentwarehouse.documents.update", "contentwarehouse.locations.initialize",
"contentwarehouse.operations.get", "contentwarehouse.rawDocuments.download",
"contentwarehouse.rawDocuments.upload", "contentwarehouse.synonymSets.get",
"contentwarehouse.synonymSets.list", "contentwarehouse.synonymSets.update"]
members     = ["serviceAccount:<Doc Ai-service-account>"]

/* cloud function */
cloud_function_name        = "HC_cloud_fuction"
cloud_function_desc        = "HC_cloud_function for Ml code"
runtime                    = "python310"
region                     = "us-central1"
timeout                    = 540
cloud_function_code_bucket  = "cloud_function_code_bucket"
cloud_function_event_bucket = "input-pdf-bucket"
source_code_name           = "function-source.zip"
source_code_path           = "../function-source.zip"
entry_point_function       = "main"
```

```
memory                  = 8192
# environment_variables for cloud function code #
project_number           = "<project_number>"
cloud_function_code_location = "us"
input_mime_type            = "application/pdf"
schema_id                   = "<doc ai warehouse schema_id>"
sa_user                   = "user:<doc ai warehouse service account>"


/* DocAi */
docai_location        = "us"
first_processor_type  = "OCR_PROCESSOR"
second_processor_type = "CUSTOM_EXTRACTION_PROCESSOR"
first_docai_name      = "ocr_processor"
second_docai_name     = "cde_processor"


/* Api And services */
gcp_service_list             =             ["contentwarehouse.googleapis.com",             "documentai.googleapis.com",
"cloudfunctions.googleapis.com", "cloudbuild.googleapis.com"]
```

1.3.2 Run terraform code

To run Terraform code run these commands

```
cd hc-docwarehouse/code/terraform-code/terraform/root

gcloud auth application-default login

terraform init

terraform plan

terraform apply
```

Note:-
After running the **gcloud auth application-default login** **please authenticate with user who already added to gcp project and have service account token creator role**

2. Batch Inference

2.1 Data Ingestion:

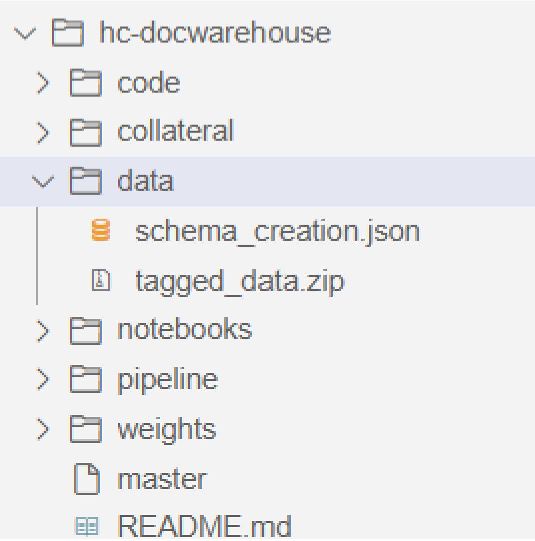
Through terraform 3 buckets will be created:

1. **<project-id>-cloud_function_code_bucket**: Contains the zip of Cloud function code
2. **<project-id>-input-pdf-bucket** : Landing bucket for the pdfs to trigger the pipeline that resides in cloud function

3. **<project-id>-processor_training_bucket** : Contains the exported jsons for train and test

2.1.1. CDE Data export management:

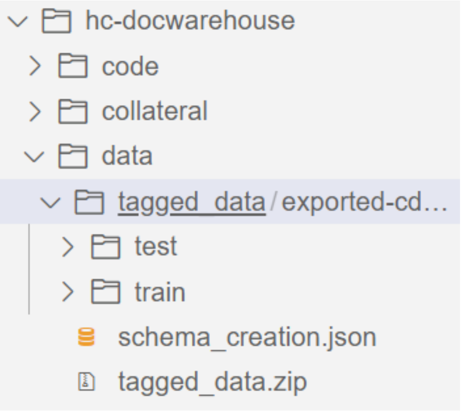
Since we have a trained model wrt to a specific training and test dataset, we need to move the training and testing dataset to the other environment to train the newly provisioned processor, Check for the data in the data folder.
Using command
cd hc-warehouse/data
The tagged data would be present in the zip file as shown below:



```
hc-docwarehouse
  code
  collateral
  data
    schema_creation.json
    tagged_data.zip
  notebooks
  pipeline
  weights
  master
  README.md
```

Now unzip the zip using the following command:
unzip tagged_data.zip

Once unzipped, the directory would look like this:



```
hc-docwarehouse
  code
  collateral
  data
    tagged_data / exported-cd…
      test
      train
    schema_creation.json
    tagged_data.zip
```

Go to tagged_data with command **cd tagged_data/**

Now, follow the following gsutil command to transfer the training files to the bucket

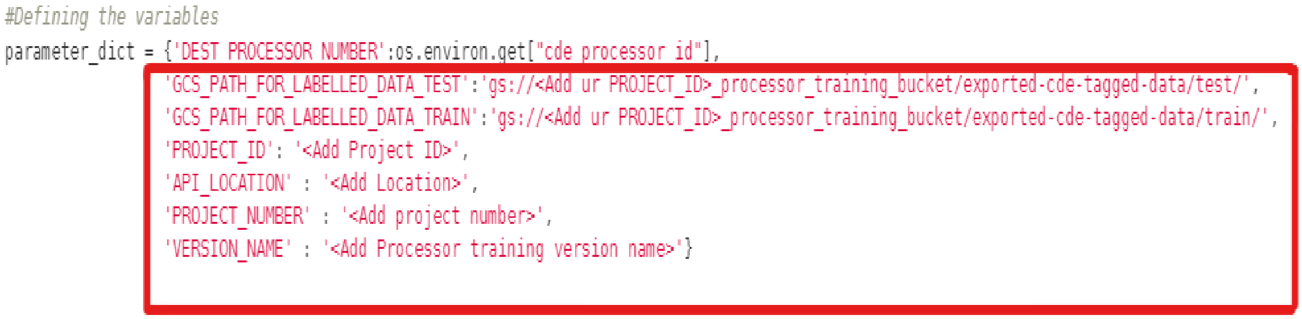
```
gsutil cp -r exported-cde-tagged-data gs://<project-id>_processor_training_bucket
```

Once the data transfer has been completed, the directory structure wrt train and test folder will be created in the target project which will have the exported jsons. The python code will automatically create a CDE processor by following ways as mentioned below:

Note: Before running the **run.sh** file, below steps need to be completed.

1. Get to the config file which is inside the root directory, directory sequence as follows:
   hc-docwarehouse→code→terraform-code→terraform→root→config.py

2. Open the config.py and fill in the placeholder places(i.e GCS path for train data, GCS path for test data, Project id, Project number, location, training version name(a string value for the display name)) as shown below.



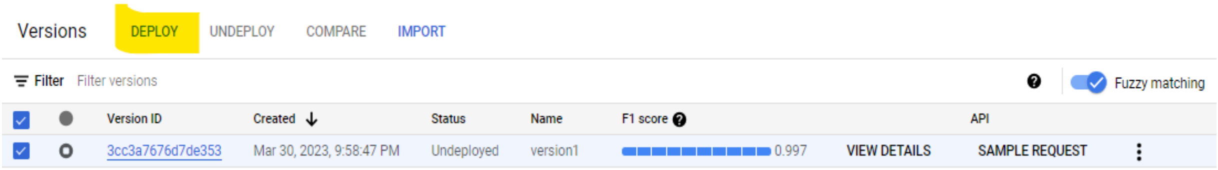
```
#Defining the variables
parameter_dict = {'DEST PROCESSOR NUMBER':os.environ.get["cde processor id"],
                  'GCS_PATH_FOR_LABELLED_DATA_TEST':'gs://<Add ur PROJECT_ID>_processor_training_bucket/exported-cde-tagged-data/test/',
                  'GCS_PATH_FOR_LABELLED_DATA_TRAIN':'gs://<Add ur PROJECT_ID>_processor_training_bucket/exported-cde-tagged-data/train/',
                  'PROJECT_ID': '<Add Project ID>',
                  'API_LOCATION' : '<Add Location>',
                  'PROJECT_NUMBER' : '<Add project number>',
                  'VERSION_NAME' : '<Add Processor training version name>'}
```

3. Once the changes in the config file has been made, run the following commands:

```
cd hc-docwarehouse/code/terraform-code/terraform/root
sudo chmod +x run.sh
./run.sh
```

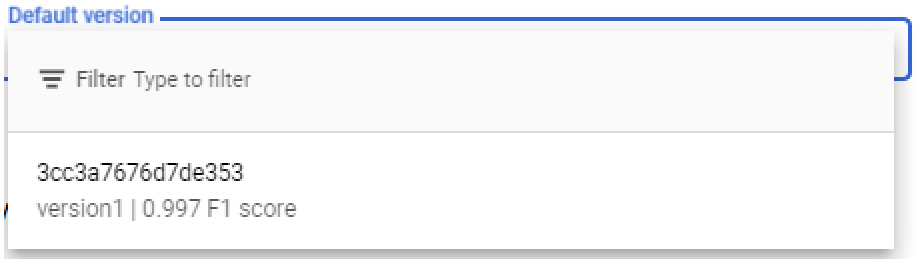
4. Now it will start initializing a CDE processor in the background and will perform the following sequence of steps: Data Import → Training → Deployment.

5. After the processor is trained, go to the managed versions tab in the CDE processor and click on deploy to deploy the trained version .



☑	●	Version ID	Created ↓	Status	Name	F1 score		API	
☑	○	3cc3a7676d7de353	Mar 30, 2023, 9:58:47 PM	Undeployed	version1	0.997	VIEW DETAILS	SAMPLE REQUEST	⋮

6. Assign that deployed version as default to process the requests by the respective version.



Default version

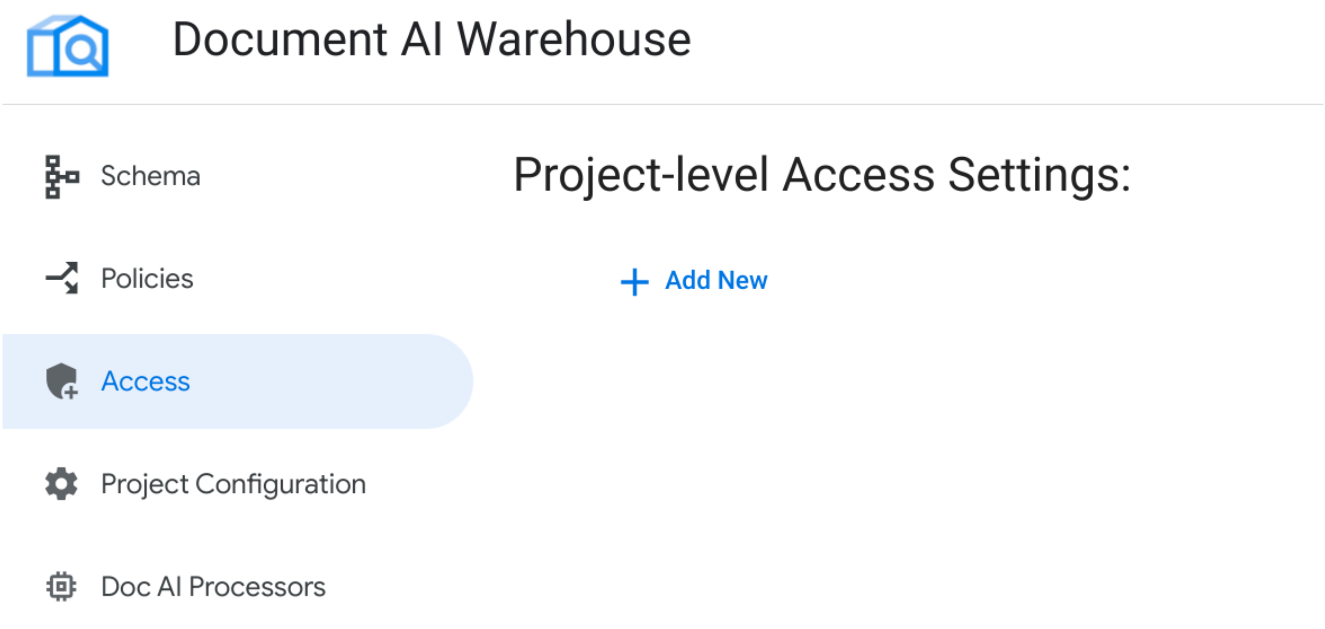
Filter Type to filter

3cc3a7676d7de353
version1 | 0.997 F1 score

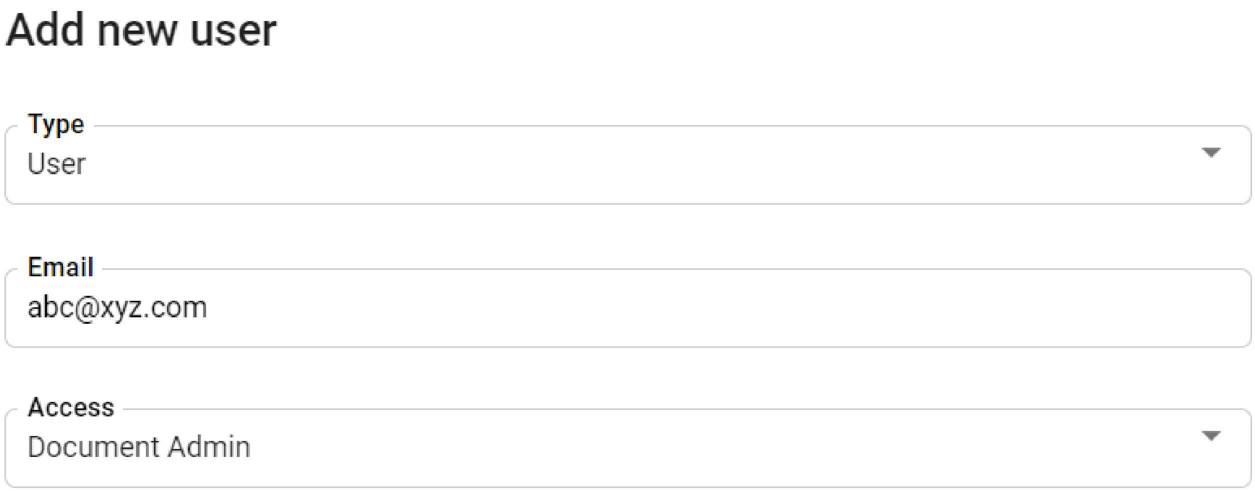
2.2 DocAI Warehouse

2.2.1 Steps to setup DocAI Warehouse

After creating the DocAI warehouse instance, one has to join the warehouse ui preview group. Link to the group. Once you are added to the group you can start viewing the UI of the warehouse.
At present, no-one has authority to work with any documents. We must set up some initial access permissions. Launch the UI and go to the Admin tab and click Access.



Click the Add New button and enter details for your identity:



Add new user

Type
User

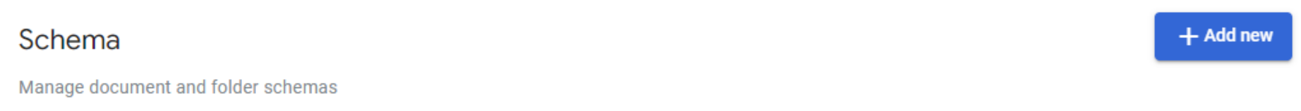
Email
abc@xyz.com

Access
Document Admin

We will find that our user now has permissions to work with document.

Note : Please make sure that the document ai warehouse service account is added as Document Admin.

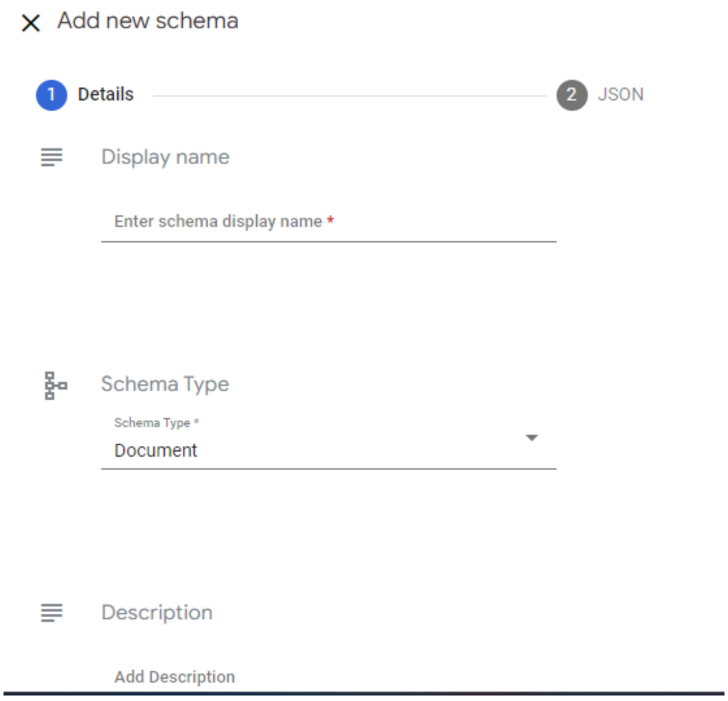
2.2.2 Steps to create schema through Warehouse UI



Schema

Manage document and folder schemas

Click on add new button and fill the details like schema display name,type and description.



Display name

Enter schema display name *

Schema Type

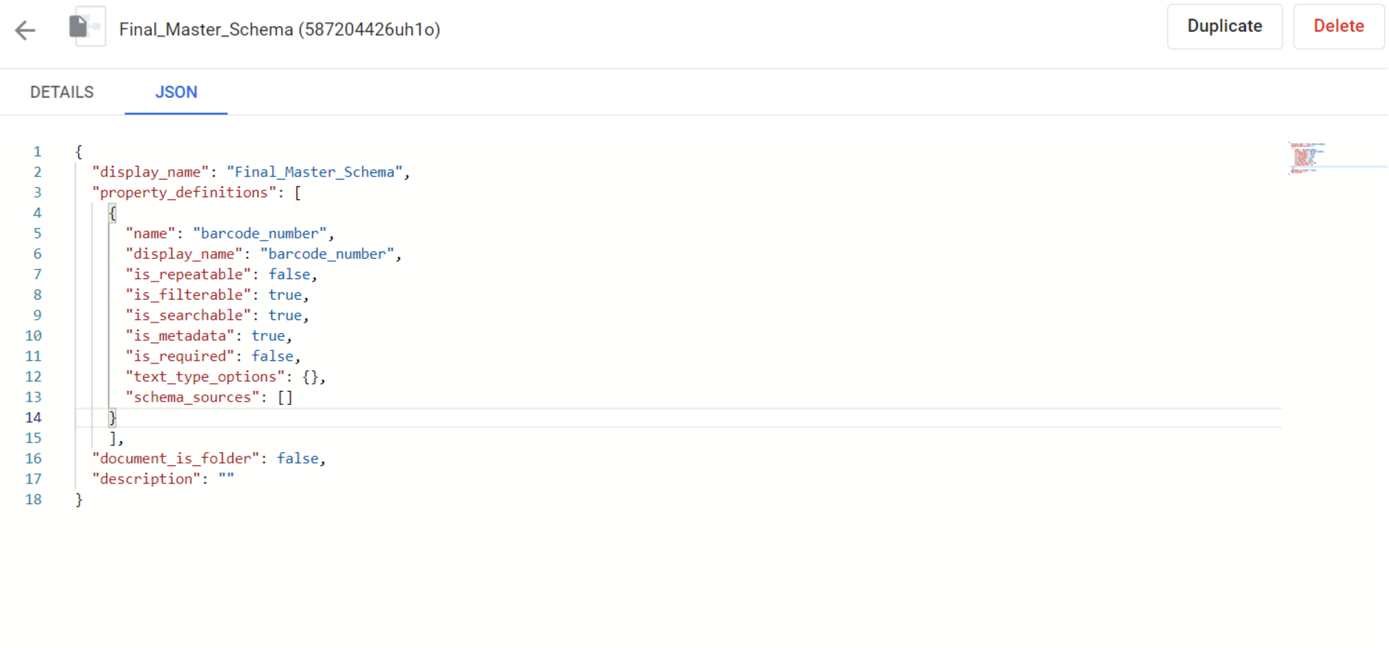
Schema Type *
Document

Description

Add Description

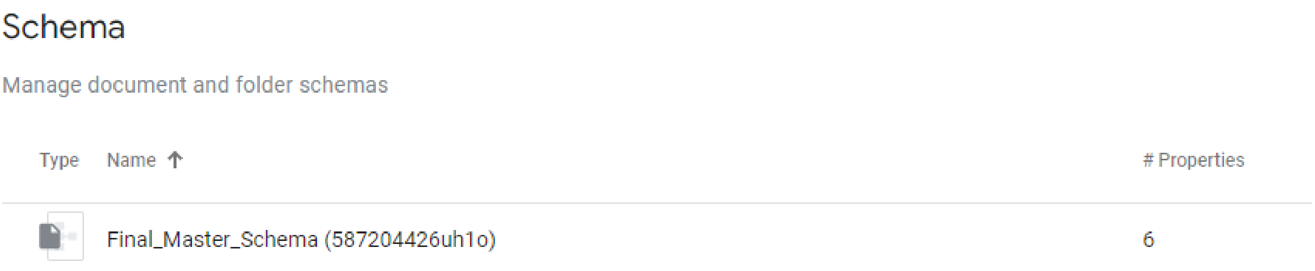
Add the required properties, a sample template of schema json attached here and click on Done.

Shown below creates a schema which has one attributed **barcode number**, which is filterable as well as searchable.



```
{
  "display_name": "Final_Master_Schema",
  "property_definitions": [
    {
      "name": "barcode_number",
      "display_name": "barcode_number",
      "is_repeatable": false,
      "is_filterable": true,
      "is_searchable": true,
      "is_metadata": true,
      "is_required": false,
      "text_type_options": {},
      "schema_sources": []
    }
  ],
  "document_is_folder": false,
  "description": ""
}
```

Once the schema is created, it would reflect in the Schema section along with its number of properties as shown below.



Type	Name ↑	# Properties
	Final_Master_Schema (587204426uh1o)	6

2.3 Cloud Function:

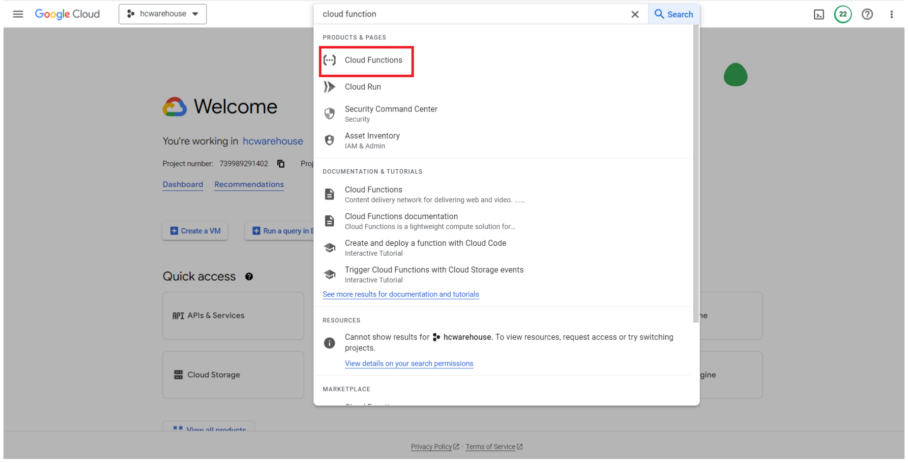
The ML pipeline code is running on a cloud function called **HC_cloud_function**. Following the deployment of the Cloud function using the Terraform script, the following prerequisites must be met:

Prerequisites:

1. Change the schema_id and sa_user under the Runtime environment variables of Cloud function to the desired values.

   Steps:

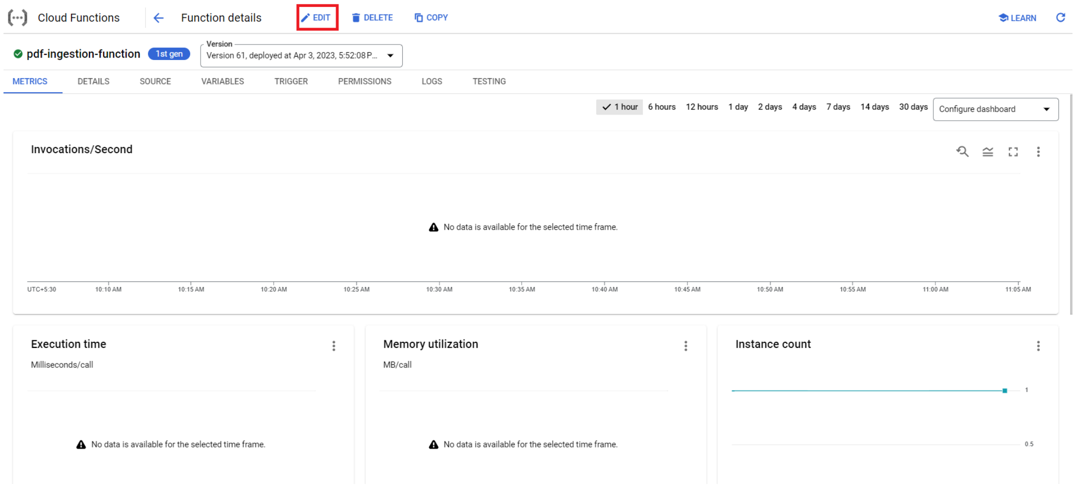
   a. Open the cloud functions page in the Google Cloud console as shown


Google Cloud
hcwarehouse
cloud function
Search
22
PRODUCTS & PAGES
Cloud Functions
Cloud Run
Security Command Center
Security
Asset Inventory
IAM & Admin
DOCUMENTATION & TUTORIALS
Cloud Functions
Content delivery network for delivering web and video. ......
Cloud Functions documentation
Cloud Functions is a lightweight compute solution for...
Create and deploy a function with Cloud Code
Interactive Tutorial
Trigger Cloud Functions with Cloud Storage events
Interactive Tutorial
See more results for documentation and tutorials
RESOURCES
View details on your search permissions
MARKETPLACE
Welcome
You're working in hcwarehouse
Project number: 739989291402
Dashboard
Recommendations
Create a VM
Run a query in
Quick access
APIs & Services
Cloud Storage
View all products
Privacy Policy · Terms of Service

Cannot show results for hcwarehouse. To view resources, request access or try switching projects.

   b. Head to the deployed Cloud function pdf-ingestion-function

   c. Click on edit



pdf-ingestion-function 1st gen

Version: Version 61, deployed at Apr 3, 2023, 5:52:08 P...

Invocations/Second

No data is available for the selected time frame.

Execution time
Milliseconds/call

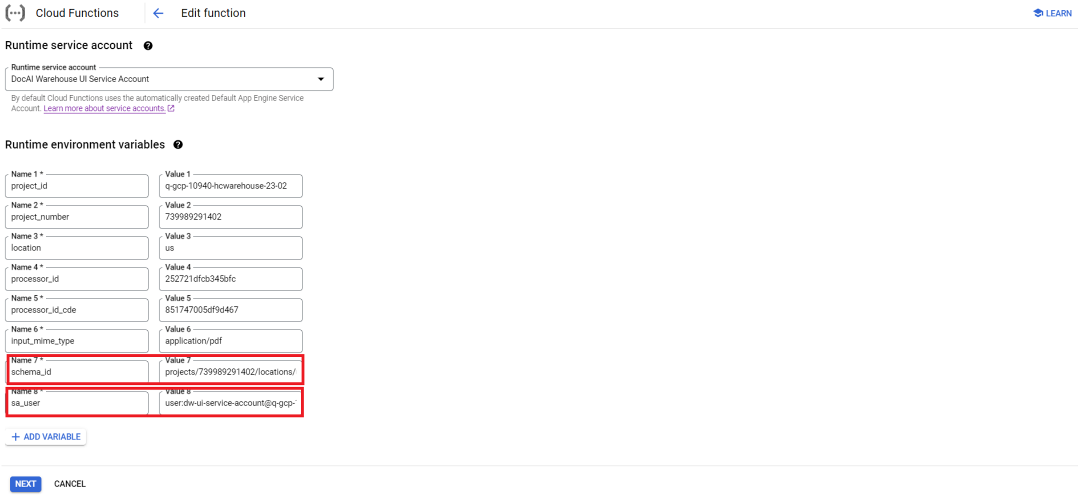
No data is available for the selected time frame.

Memory utilization
MB/call

No data is available for the selected time frame.

Instance count

d. Expand the Runtime, build, connections and security settings and find the variables under the section of Runtime environment variables



Runtime service account

Runtime service account: DocAI Warehouse UI Service Account

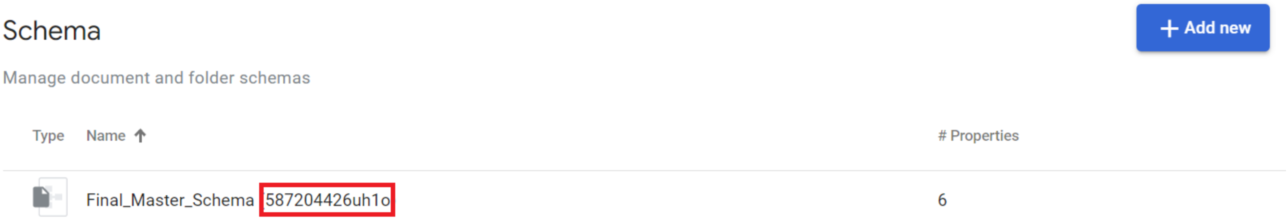
By default Cloud Functions uses the automatically created Default App Engine Service Account. Learn more about service accounts.

Runtime environment variables

Name	Value
Name 1 * project_id	Value 1 q-gcp-10940-hcwarehouse-23-02
Name 2 * project_number	Value 2 739989291402
Name 3 * location	Value 3 us
Name 4 * processor_id	Value 4 252721dfcb345bfc
Name 5 * processor_id_cde	Value 5 851747005df9d467
Name 6 * input_mime_type	Value 6 application/pdf
Name 7 * schema_id	Value 7 projects/739989291402/locations/
Name 8 * sa_user	Value 8 user:dw-ui-service-account@q-gcp-

Schema_id value needs to be changed according to the requirements. Schema id needs to follow this format:
projects/{project}/locations/{location}/documentSchemas/{schema_id}

schema_id can be found here as shown:


Schema

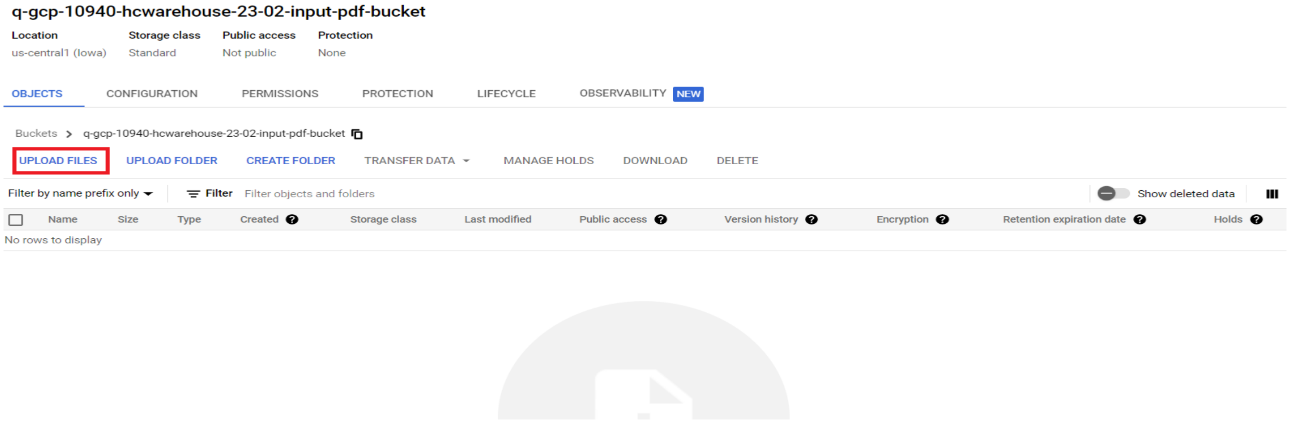
Manage document and folder schemas

Type	Name	# Properties
	Final_Master_Schema 587204426uh1o	6

e. Then click on next followed by deploy to deploy the cloud function with the changes.

2.4. Pipeline run:

We are using GCS for data ingestion and the name of the bucket goes by **<project-id>-input-pdf-bucket**. The solution supports both single and multiple file uploads.


q-gcp-10940-hcwarehouse-23-02-input-pdf-bucket

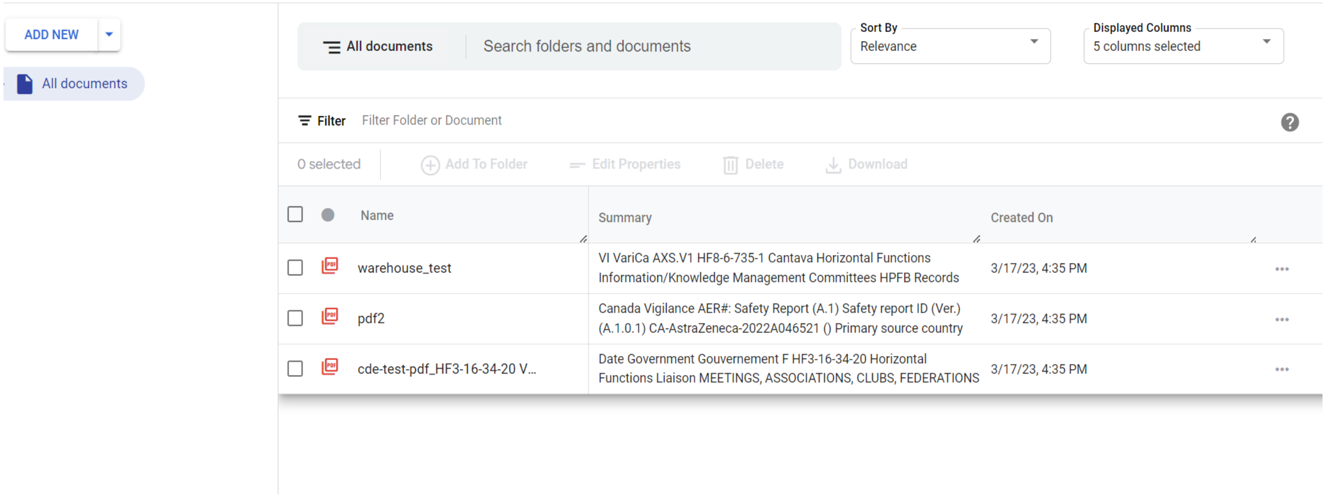
Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

□	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date	Holds

No rows to display

After uploading of documents to cloud storage, it triggers the cloud function named as **HC_cloud_function** which ocr the complete document and extract entities from the first page using Custom Document Extractor(CDE).

And after the extraction process, it pushed the pdf document to Warehouse with the extracted entities to be displayed as the properties wrt the respective document as shown below.

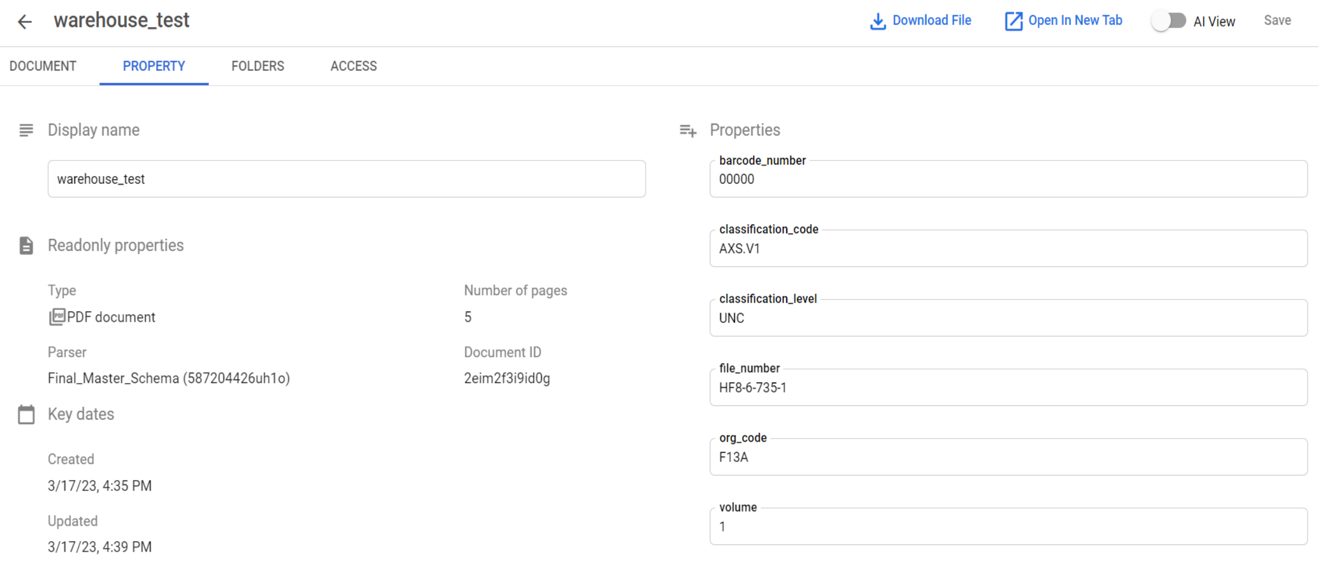


Sort By: Relevance

Displayed Columns: 5 columns selected

□	●	Name	Summary	Created On	
□		warehouse_test	VI VariCa AXS.V1 HF8-6-735-1 Cantava Horizontal Functions Information/Knowledge Management Committees HPFB Records	3/17/23, 4:35 PM	•••
□		pdf2	Canada Vigilance AER#: Safety Report (A.1) Safety report ID (Ver.) (A.1.0.1) CA-AstraZeneca-2022A046521 () Primary source country	3/17/23, 4:35 PM	•••
□		cde-test-pdf_HF3-16-34-20 V...	Date Government Gouvernement F HF3-16-34-20 Horizontal Functions Liaison MEETINGS, ASSOCIATIONS, CLUBS, FEDERATIONS	3/17/23, 4:35 PM	•••

The properties for a document are listed out in this fashion(shown below):



warehouse_test

Display name

warehouse_test

Readonly properties

Type

PDF document

Number of pages

5

Parser

Final_Master_Schema (587204426uh1o)

Document ID

2eim2f3i9id0g

Key dates

Created

3/17/23, 4:35 PM

Updated

3/17/23, 4:39 PM

Properties

barcode_number: 00000

classification_code: AXS.V1

classification_level: UNC

file_number: HF8-6-735-1

org_code: F13A

volume: 1