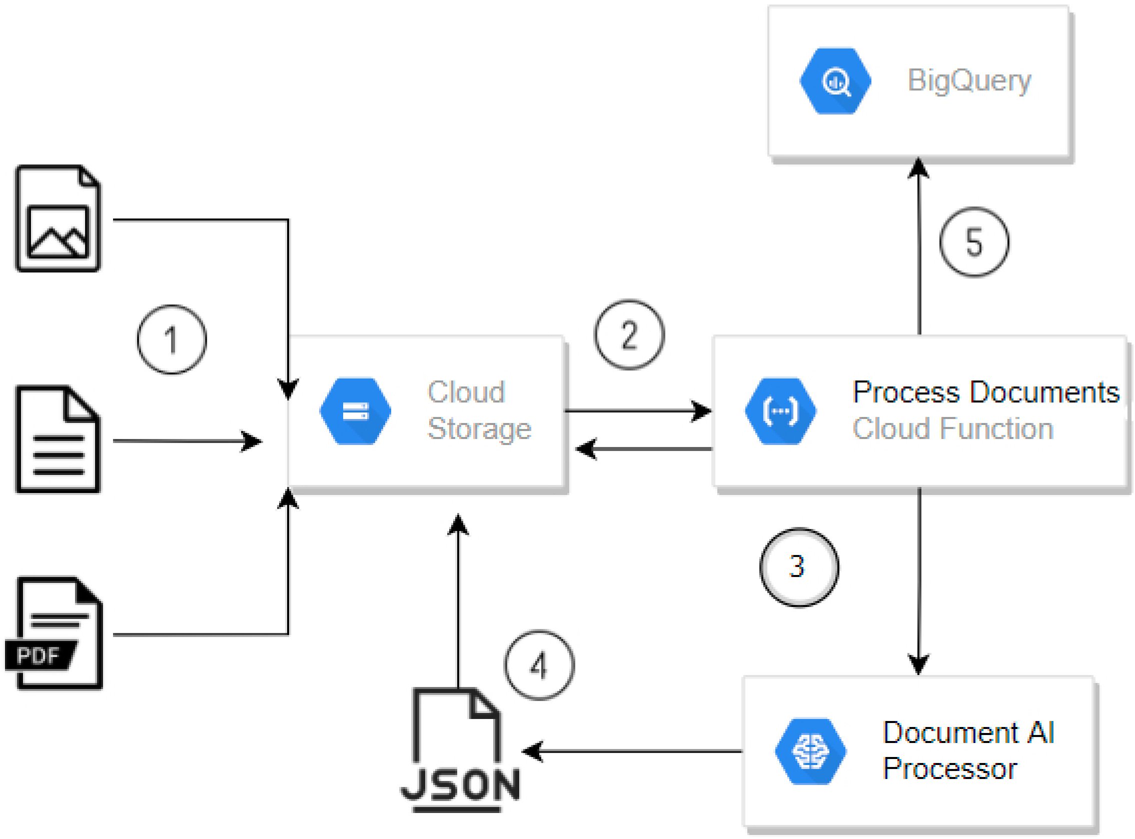
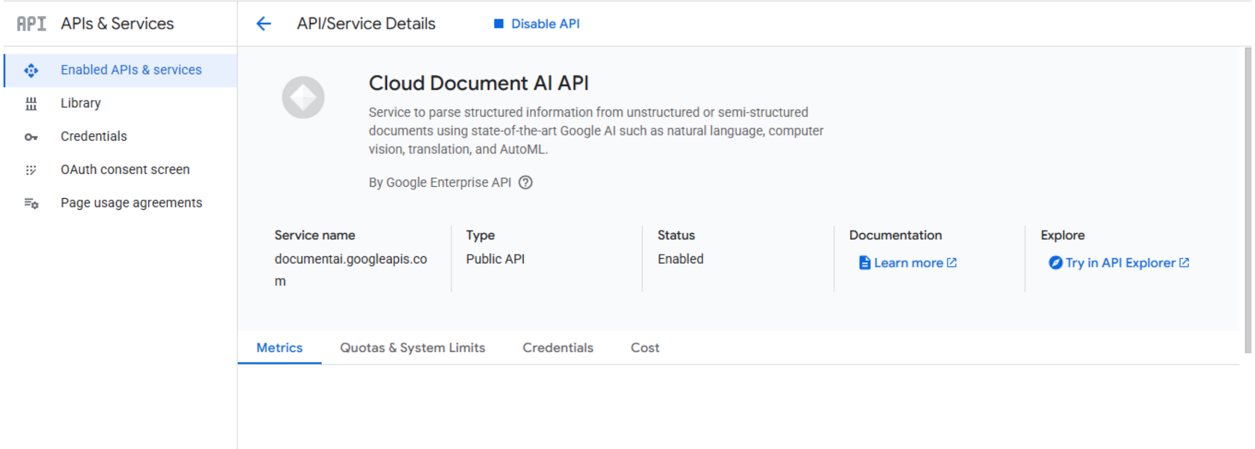
Automate Data Capture at Scale with Document AI: Challenge Lab



Task 1: Enable the Cloud Document AI API and copy lab source files

1. **Enable the Cloud Document AI API**:
   a. In the Google Cloud Console, navigate to "APIs & Services" > "Library"
   b. Search for "Cloud Document AI API"
   c. Click on the result and then click "Enable"



Cloud Document AI API

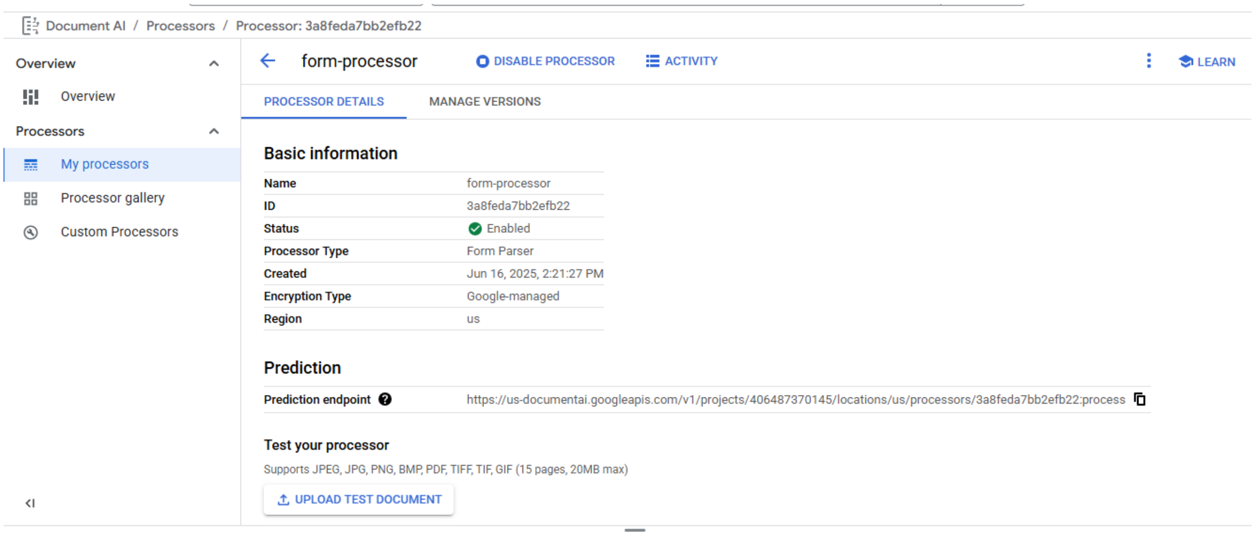
Service to parse structured information from unstructured or semi-structured documents using state-of-the-art Google AI such as natural language, computer vision, translation, and AutoML.

By Google Enterprise API

Service name	Type	Status	Documentation	Explore
documentai.googleapis.com	Public API	Enabled	Learn more	Try in API Explorer

Task 2: Create a form processor

1. In the Cloud Console, navigate to "Document AI" > "Processors"
2. Click "+ Create Processor"
3. Configure the processor:
    a. Processor Type: Select "Form Parser" under "General (non-specialized)"
    b. Processor Name: Enter any name (e.g., "invoice-processor")
    c. Region: Select "US"
4. Click "Create"
5. After creation, note down:
    a. The Processor ID (visible on the processor details page)
    b. The Parser Location is "us" (lowercase)



Basic information

Name	form-processor
ID	3a8feda7bb2efb22
Status	✓ Enabled
Processor Type	Form Parser
Created	Jun 16, 2025, 2:21:27 PM
Encryption Type	Google-managed
Region	us

Prediction

Prediction endpoint https://us-documentai.googleapis.com/v1/projects/406487370145/locations/us/processors/3a8feda7bb2efb22:process

Test your processor

Supports JPEG, JPG, PNG, BMP, PDF, TIFF, TIF, GIF (15 pages, 20MB max)

Task 3. Create Google Cloud resources

Prepare your environment by creating the Google Cloud Storage and BigQuery resources that are required for your document processing pipeline.

Create input, output, and archive Cloud Storage buckets

- In this step, you must create the three Cloud Storage buckets listed below with uniform bucket level access enabled.

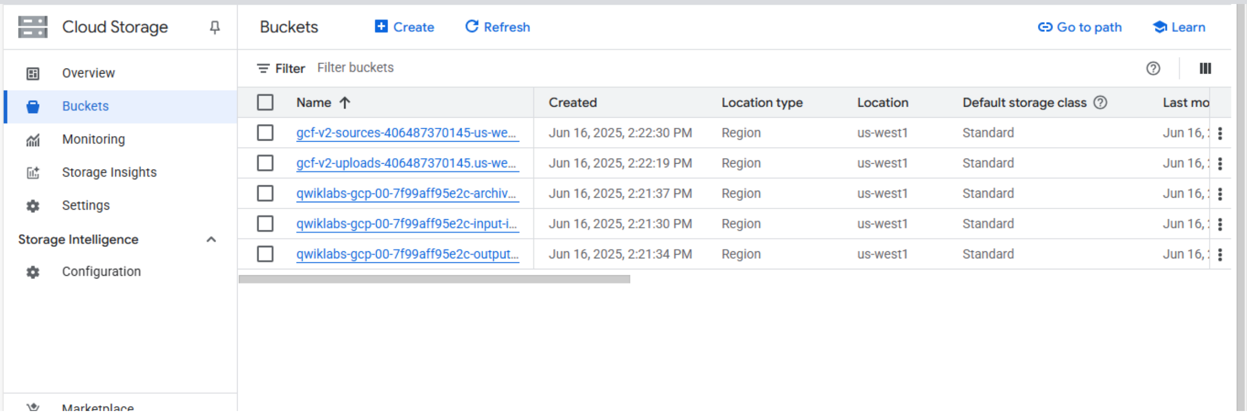
Bucket Name	Purpose	Storage class	Location
`input_bucket_name`	For input invoices	Standard	`REGION`
`output_bucket_name`	For storing processed data	Standard	`REGION`
`archive_bucket_name`	For archiving invoices	Standard	`REGION`

Create a BigQuery dataset and tables

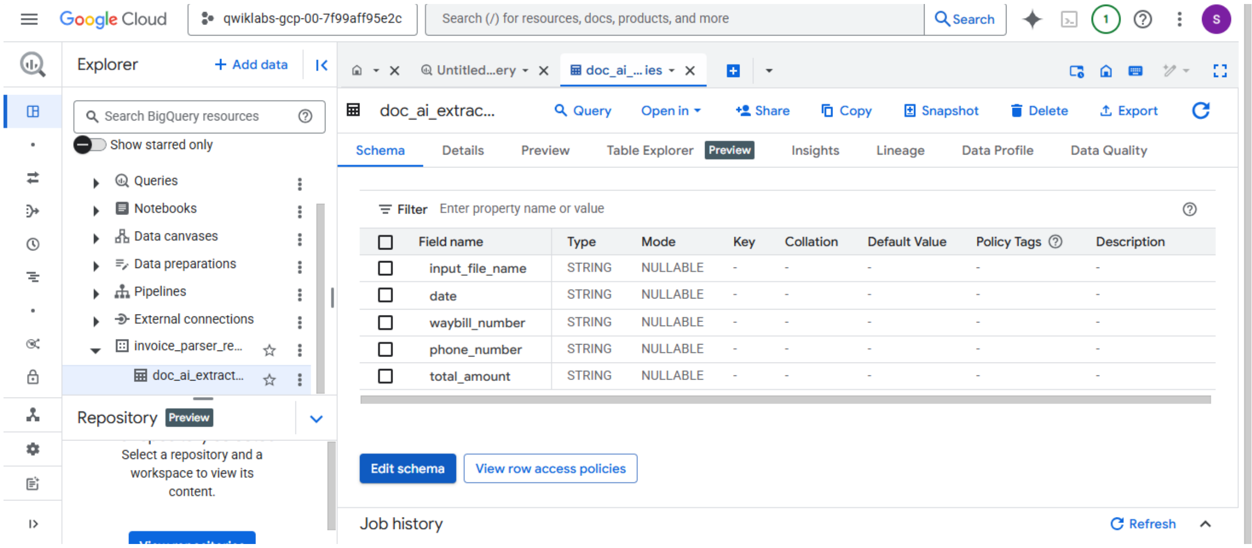
- In this step, you must create a BigQuery dataset and the output table required for your data processing pipeline.

Dataset

Dataset Name	Location
invoice_parser_results	US



Name	Created	Location type	Location	Default storage class	Last mo
gcf-v2-sources-406487370145-us-we...	Jun 16, 2025, 2:22:30 PM	Region	us-west1	Standard	Jun 16,
gcf-v2-uploads-406487370145.us-we...	Jun 16, 2025, 2:22:19 PM	Region	us-west1	Standard	Jun 16,
qwiklabs-gcp-00-7f99aff95e2c-archiv...	Jun 16, 2025, 2:21:37 PM	Region	us-west1	Standard	Jun 16,
qwiklabs-gcp-00-7f99aff95e2c-input-i...	Jun 16, 2025, 2:21:30 PM	Region	us-west1	Standard	Jun 16,
qwiklabs-gcp-00-7f99aff95e2c-output...	Jun 16, 2025, 2:21:34 PM	Region	us-west1	Standard	Jun 16,



Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
input_file_name	STRING	NULLABLE	-	-	-	-	-
date	STRING	NULLABLE	-	-	-	-	-
waybill_number	STRING	NULLABLE	-	-	-	-	-
phone_number	STRING	NULLABLE	-	-	-	-	-
total_amount	STRING	NULLABLE	-	-	-	-	-

Task 4. Deploy the document processing Cloud Run functions

To complete this task, you must deploy the Cloud Run functions that your data processing pipeline uses to process invoices uploaded to Cloud Storage. This function will use a Document AI API Generic Form processor to extract form data from the raw documents.

You can examine the source code of the Cloud Run functions using the Code Editor or any other editor of your choice. The Cloud Run functions is stored in the following folders in Cloud Shell:

- Process Invoices - scripts/cloud-functions/process-invoices

The Cloud Run functions, process-invoices, must be triggered when files are uploaded to the input files storage bucket you created earlier.

Deploy the Cloud Run functions to process documents uploaded to Cloud Storage

Deploy a Cloud Run functions that uses a Document AI form processor to parse form documents that have been uploaded to a Cloud Storage bucket.

1. Navigate to scripts directory:

```
cd ~/document-ai-challenge/scripts
```

2. Assign the Artifact Registry Reader role to the Compute Engine service account:

```
PROJECT_ID=$(gcloud config get-value project) PROJECT_NUMBER=$(gcloud projects list --filter="project_id:$PROJECT_ID" --format='value(project_number)')

SERVICE_ACCOUNT=$(gcloud storage service-agent --project=$PROJECT_ID)

gcloud projects add-iam-policy-binding $PROJECT_ID

--member serviceAccount:$SERVICE_ACCOUNT

--role roles/pubsub.publisher
```

3. Deploy the Cloud Run functions:

```
export CLOUD_FUNCTION_LOCATION="REGION" gcloud functions deploy process-invoices

--gen2

--region=${CLOUD_FUNCTION_LOCATION}

--entry-point=process_invoice
```

```
--runtime=python39

--service-account=${PROJECT_ID}@appspot.gserviceaccount.com

--source=cloud-functions/process-invoices

--timeout=400

--env-vars-file=cloud-functions/process-invoices/.env.yaml

--trigger-resource=gs://${PROJECT_ID}-input-invoices

--trigger-event=google.storage.object.finalize

 --service-account $PROJECT_NUMBER-compute@developer.gserviceaccount.com

--allow-unauthenticated
```

If you inspect the Cloud Run Functions source code you will see that the function gets the Document AI processor details via two runtime environment variables.

- You will have to **reconfigure** the Cloud Run functions deployment so that the environment variablesPROCESSOR_ID and PARSER_LOCATION contain the correct values for the **Form Parser** processor you deployed in a previous step.
- Make sure the PARSER_LOCATION value must be in lower case.
- Make sure to also update the PROJECT_ID environment variable with your project ID.

Task 5. Test and validate the end-to-end solution

For your final task you must successfully process the set of invoices that are available in the ~/document-ai-challenge/invoices folder using your pipeline.

1. Upload these invoices to the input Cloud Storage bucket and monitor the progress of the pipeline.
2. Watch the events until you see a final event indicating that the function execution finished with a status of **OK**.