Automate Data Capture at Scale with Document AI: Challenge Lab

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

In this task, you enable the Cloud Document AI API and copy your starter files into Cloud Shell.

Enable the Cloud Document AI API

- Enable the Cloud Document AI API.

You must enable the APIs for Document AI, Cloud Run functions, Cloud Build, and Geocoding for this lab, then create the API key that is required by the Geocoding Cloud Run function.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.
2. In Cloud Shell, enter the following commands to enable the APIs required by the lab:

```
gcloud services enable documentai.googleapis.com
gcloud services enable cloudfunctions.googleapis.com
gcloud services enable cloudbuild.googleapis.com
gcloud services enable geocoding-backend.googleapis.com
```

3. In the console, in the **Navigation menu** (☰), click **APIs & services > Credentials**.

4. Select **Create credentials**, then select **API key** from the dropdown menu.

The API key created dialog box displays your newly created key. An API key is a long string containing upper and lower case letters, numbers, and dashes. For example, `a4db08b757294ea94c08f2df493465a1`.

5. Click three dots under **Actions** then click **Edit API key** in the dialog box.

6. Select **Restrict key** in the **API restrictions** section to add API restrictions for your new API key.

7. Click in the filter box and type **Geocoding API**.

8. Select **Geocoding API** and click **OK**.

9. Click the **Save** button.

Copy the lab source files into your Cloud Shell

The Cloud Run functions with predefined code is hosted on a remote Cloud Storage bucket. Copy these source files into your Cloud Shell. These files include the source code for the Cloud Run functions and the schema for the BigQuery table that you will create in the lab.

- In Cloud Shell, enter the following command to clone the source repository for the lab:

```
mkdir ./document-ai-challenge
gsutil -m cp -r gs://spls/gsp367/* \
  ~/document-ai-challenge/
```

Task 2. Create a form processor

Create an instance of the general form processor using the Document AI **Form Parser** processor in the **General** (non-specialized) section. The general form processor will process any type of document and extract all the text content it can identify in the document as well as form information that it infers from the layout.

- Create the processor using the following configuration details:

Property	**Value**
Processor Type	Form Parser
Processor Name	`finance-processor`
Region	US

Note: You will configure a Cloud Run functions later in this lab with the **PROCESSOR ID** and **PARSER LOCATION** of this processor so that the Cloud Run functions will use this specific processor to process invoices. Click on the created processor and note the PROCESSOR ID. However, the processor region is the PARSER LOCATION.

1. In the Google Cloud Console, in the search bar, type `Document AI` and click the product page result.
2. Click **Explore Processors** and click **Create Processor** for Form Parser.
3. Specify the processor name as `finance-processor` and select the region **US (United States)** from the list.
4. Click **Create** to create your processor.

You will configure a Cloud Run function later in this lab with the processor ID and location of this processor so that the Cloud Run function will use this specific processor to process sample invoices.

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
`qwiklabs-gcp-02-afbb12f7cec5-input-invoices`	For input invoices	Standard	`us-west1`
`qwiklabs-gcp-02-afbb12f7cec5-output-invoices`	For storing processed data	Standard	`us-west1`
`qwiklabs-gcp-02-afbb12f7cec5-archived-invoices`	For archiving invoices	Standard	`us-west1`

Note: A bucket can be created using the gsutil tool with the *-mb* parameter, along with *-c* parameter to set the storage class, *-l* to set the (regional) location and the *-b* flag with the value of on or off to set uniform bucket level access. Read the mb - Make buckets reference for more about creating buckets using gsutil.

```
# Input bucket

gsutil mb -c standard -l us-west1 -b on gs://qwiklabs-gcp-02-afbb12f7cec5-input-invoices/


# Output bucket

gsutil mb -c standard -l us-west1 -b on gs://qwiklabs-gcp-02-afbb12f7cec5-output-invoices/
```

Archive bucket

gsutil mb -c standard -l us-west1 -b on gs://qwiklabs-gcp-02-afbb12f7cec5-archived-invoices/

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

Note: Use **bq mk** to create BigQuery resources. The command line switch parameter *-d* is used to create a dataset and *--location* is used to set the location of the resource. Read the Create datasets guide for more information about creating datasets using the bq command-line tool

Table

The table schema for the extracted information has been provided for you in the JSON file `document-ai-challenge/scripts/table-schema/doc_ai_extracted_entities.json`. Use this schema to create a table named **doc_ai_extracted_entities** in the **invoice_parser_results** dataset.

You need a dataset named `invoice_parser_results` in the **US** multi-region.

```
bq mk --location=US -d invoice_parser_results
```

Assuming the schema file is located at:

```
document-ai-challenge/scripts/table-schema/doc_ai_extracted_entities.json
```

Run this command to create the table:

```
bq mk --table invoice_parser_results.doc_ai_extracted_entities document-ai-
challenge/scripts/table-schema/doc_ai_extracted_entities.json
```

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=us-west1
  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
```

Note: If you get an permission error while deploying function wait for 2-3 minutes and re-run the commands.

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 variables`PROCESSOR_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.

Wait for the function to be fully redeployed.

Replace the following with **your actual values** (get these from your Document AI processor):

- `PROCESSOR_ID`: Your Form Parser Processor ID.
- `PARSER_LOCATION`: Region where your processor is deployed (e.g. `us`, `us-west1`). Must be **lowercase**.

Open the environment file and edit it:

```
nano cloud-functions/process-invoices/.env.yaml
```

Update the following variables in the file:

```
PROJECT_ID: your-project-id
PROCESSOR_ID: your-processor-id
PARSER_LOCATION: us-west1
```

Press `Ctrl+O` then `Enter` to save, then `Ctrl+X` to exit.

---

5. Deploy the function

```
export CLOUD_FUNCTION_LOCATION=us-west1

gcloud functions deploy process-invoices \
  --gen2 \
  --region=${CLOUD_FUNCTION_LOCATION} \
  --entry-point=process_invoice \
  --runtime=python39 \
  --source=cloud-functions/process-invoices \
  --timeout=400 \
  --env-vars-file=cloud-functions/process-invoices/.env.yaml \
  --trigger-resource=gs://${PROJECT_ID}-input-invoices \
  --trigger-event=google.storage.object.finalize \
  --service-account ${PROJECT_NUMBER}-compute@developer.gserviceaccount.com
\
  --allow-unauthenticated
```

Upload invoices to the input bucket

```
gsutil cp ~/document-ai-challenge/invoices/* gs://${PROJECT_ID}-input-
invoices
```

Monitor the Cloud Function

- Go to **Cloud Console > Cloud Functions > process-invoices**
- Click on **Logs**
- Look for logs showing status `OK`, and check for BigQuery insert statements.

🔄 Logs do not auto-refresh. Click **Refresh** to update.

Validate BigQuery Results

Go to **BigQuery Console** and run:

```
SELECT * FROM `PROJECT_ID.invoice_parser_results.doc_ai_extracted_entities`
LIMIT 10;
```

Replace `PROJECT_ID` with your actual project ID.

You should see rows corresponding to the extracted invoice data