Create and Test a Document AI Processor

Overview

This project involved leveraging Google Cloud's **Document AI API** to create and test a **general form processor** capable of extracting structured data (key-value pairs) from unstructured documents. The lab covered:

- Enabling the Document AI API.
- Creating a **Form Parser** processor.
- Testing the processor via the **Google Cloud Console** and **command line**.
- Making **synchronous API calls** using Python.

Objectives

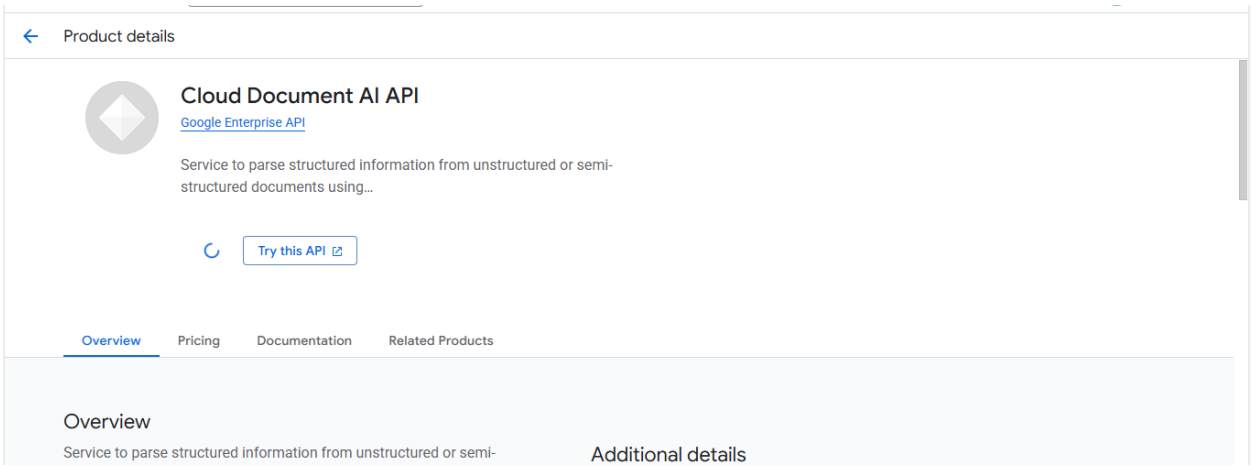
✅ **Enable the Document AI API**
✅ **Create and test a general form processor**
✅ **Authenticate API requests & download a sample form**
✅ **Process a document via** curl
✅ **Test the processor using Python client libraries**

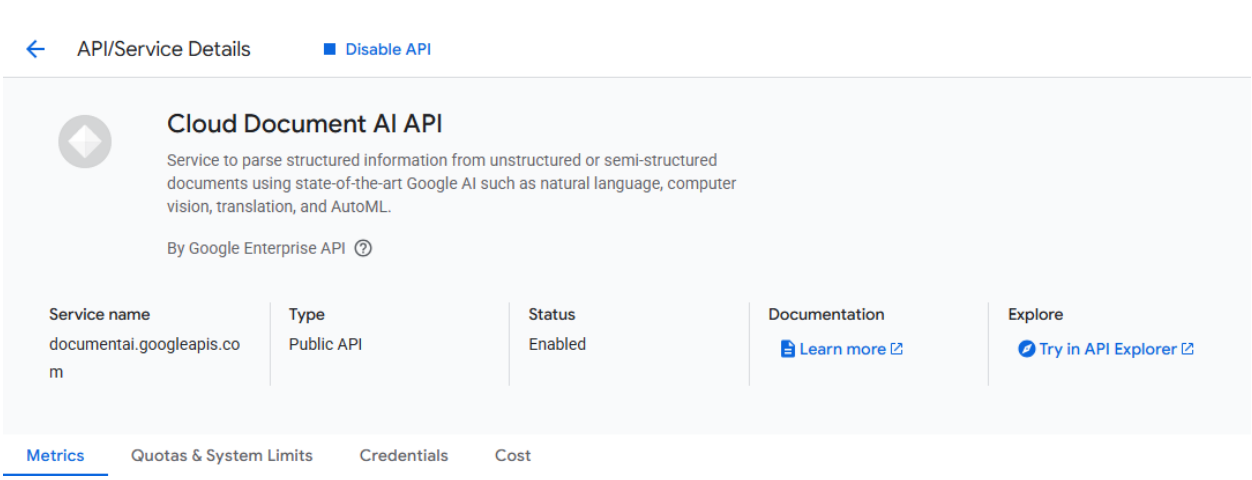
Lab Setup

- **Google Cloud Project**: Used a temporary project with pre-configured credentials.
- **Cloud Shell & VM Instance**: Accessed via SSH for command-line tasks.
- **Document AI API**: Enabled via Google Cloud Console.

Task 1: Enable the Cloud Document AI API

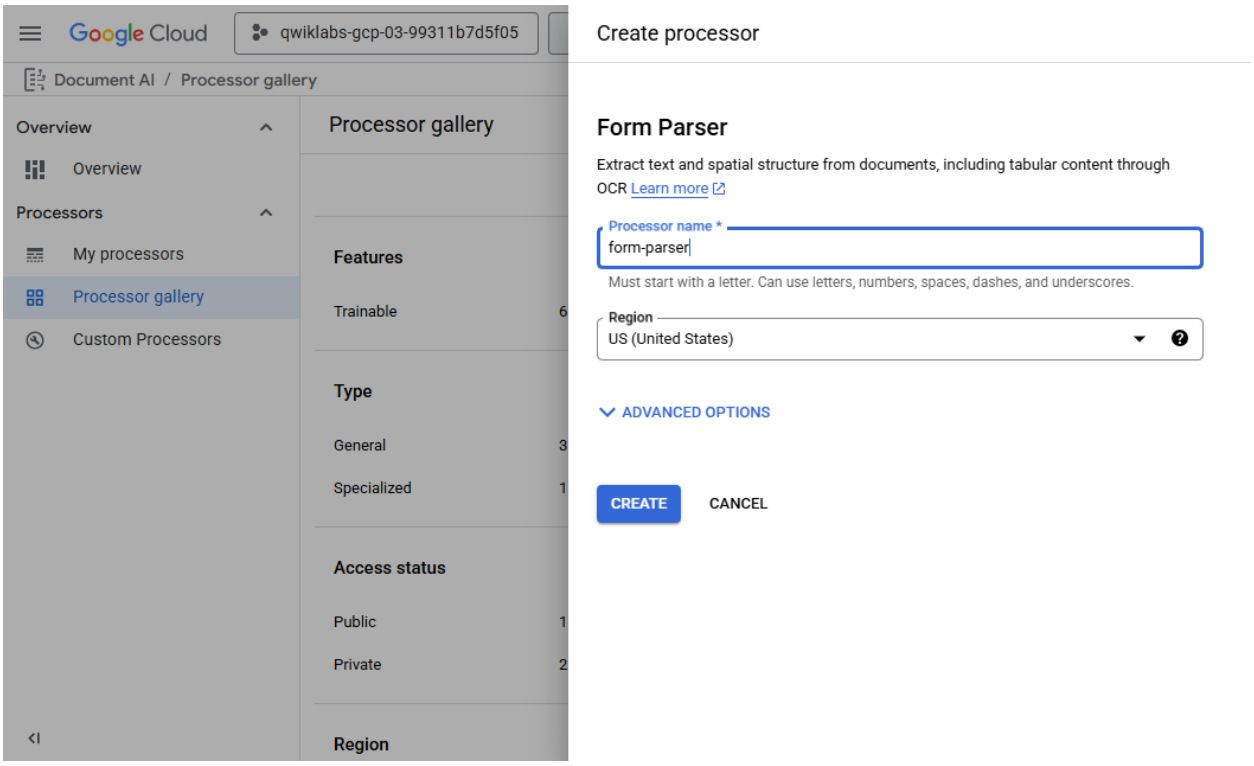
- Navigated to **APIs & Services > Library**.
- Searched for **"Cloud Document AI API"** and enabled it.
- Verified API activation.

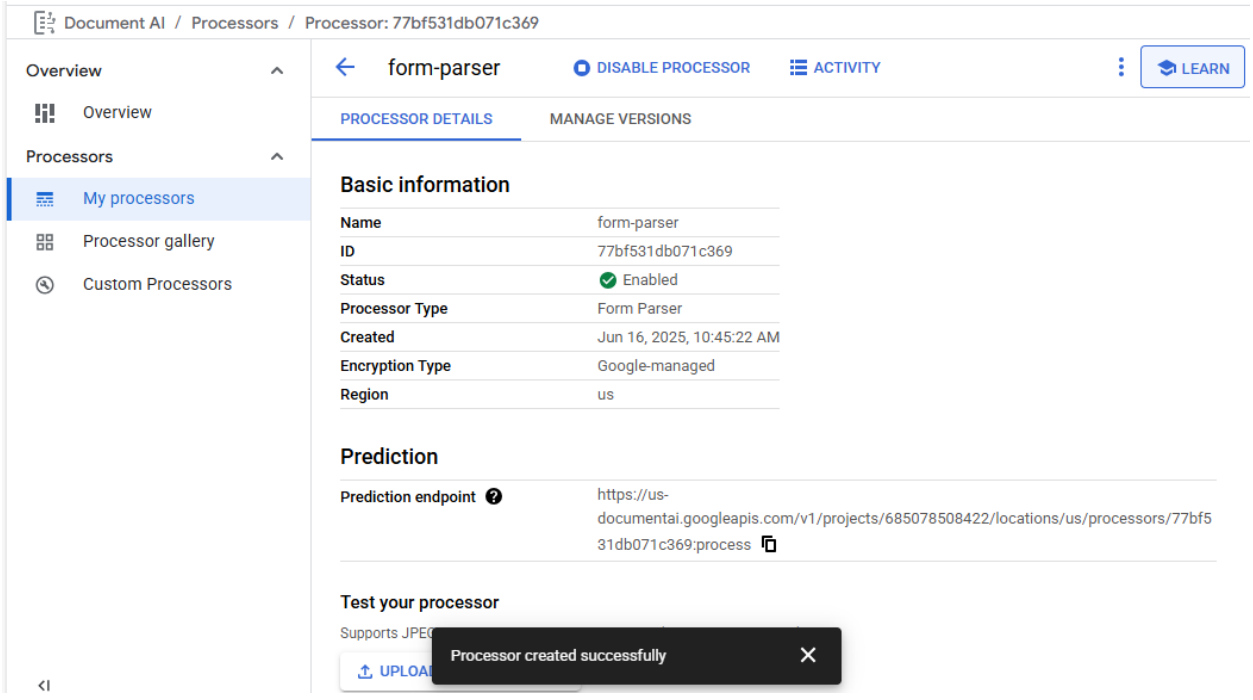


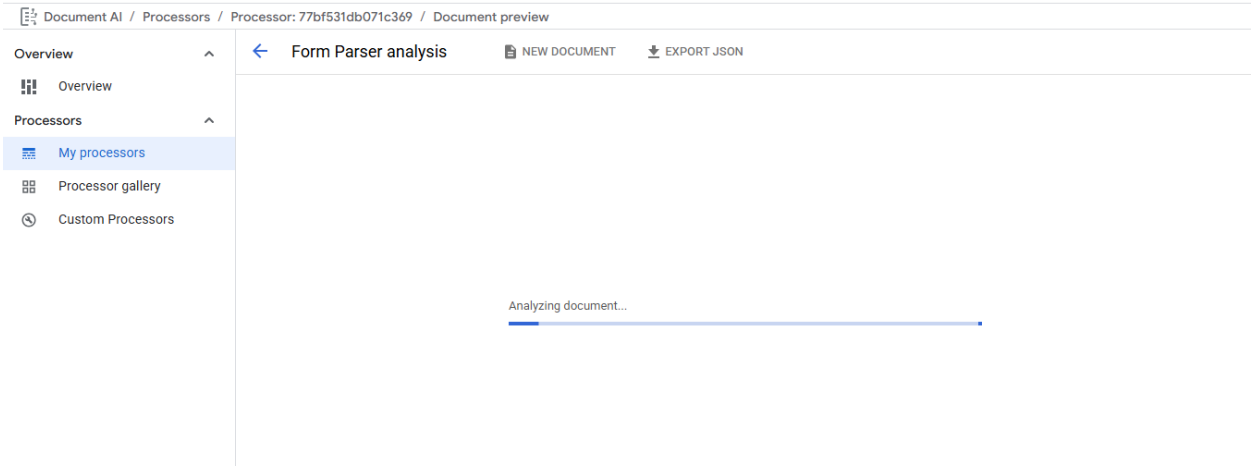


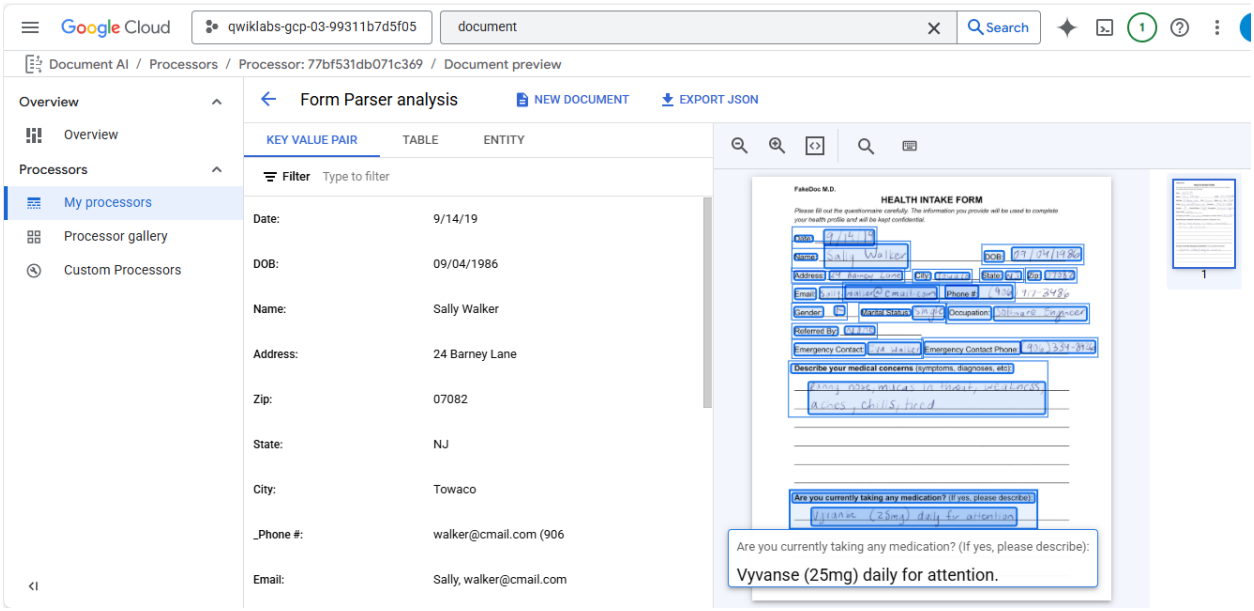
Task 2: Create and Test a General Form Processor

- Created a **Form Parser** processor named form-parser in the **US region**.
- Downloaded a sample **health intake form (**form.pdf**)** from Cloud Storage.
- Uploaded the form to the processor via the **Google Cloud Console**.
- Verified extraction of **key-value pairs** (e.g., Name: Sally Walker, Date: 9/14/19).


Google Cloud
qwiklabs-gcp-03-99311b7d5f05
Document AI / Processor gallery
Overview
Overview
Processors
My processors
Processor gallery
Custom Processors
Processor gallery
Features
Trainable 6
Type
General 3
Specialized 1
Access status
Public 1
Private 2
Region
Create processor
Form Parser
Extract text and spatial structure from documents, including tabular content through OCR Learn more
Processor name *
form-parser
Must start with a letter. Can use letters, numbers, spaces, dashes, and underscores.
Region
US (United States)
ADVANCED OPTIONS
CREATE
CANCEL

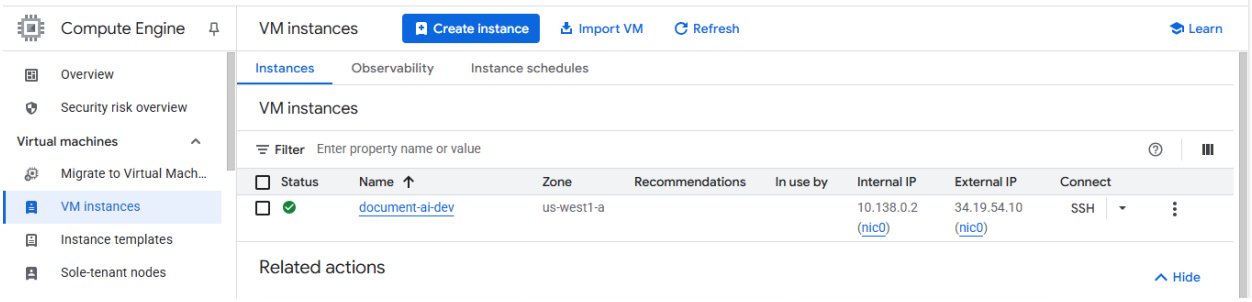

Document AI / Processors / Processor: 77bf531db071c369
Overview
Overview
Processors
My processors
Processor gallery
Custom Processors
form-parser
DISABLE PROCESSOR
ACTIVITY
LEARN
PROCESSOR DETAILS
MANAGE VERSIONS
Basic information
Name form-parser
ID 77bf531db071c369
Status Enabled
Processor Type Form Parser
Created Jun 16, 2025, 10:45:22 AM
Encryption Type Google-managed
Region us
Prediction
Prediction endpoint
https://us-documentai.googleapis.com/v1/projects/685078508422/locations/us/processors/77bf531db071c369:process
Test your processor
Supports JPEG
UPLOAD
Processor created successfully



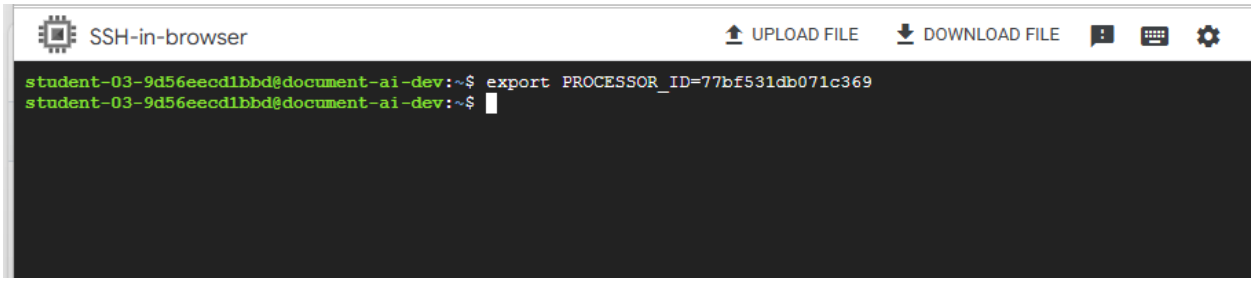


Task 3: Set Up the Lab Instance

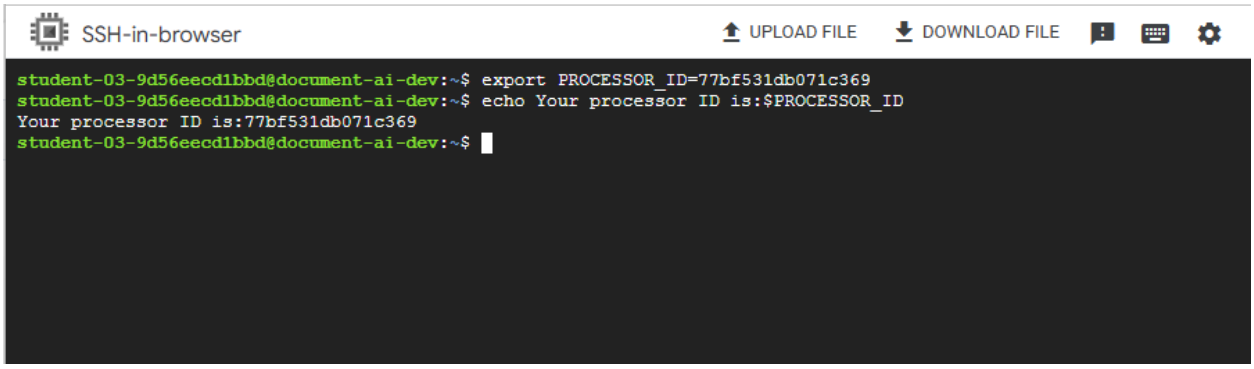
- Connected to the document-ai-dev **VM** via SSH.
- Set environment variables for **Processor ID**.
- Created a **service account** (document-ai-service-account) with roles/documentai.apiUser permissions.
- Generated **service account credentials** (key.json).
- Downloaded the sample form (health-intake-form.pdf) and encoded it in **Base64** for API submission.



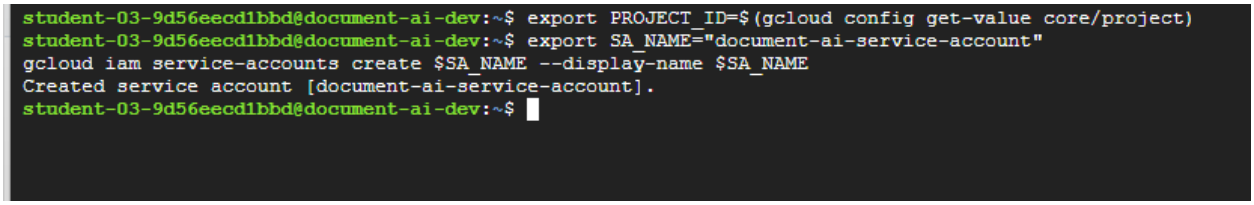
3. In the SSH session, create an environment variable to contain the Document AI processor ID. You must replace the placeholder for [your processor id]:



4. In the SSH session confirm that the environment variable contains the Document AI processor ID:



5. This should print out the Processor ID similar to the following:



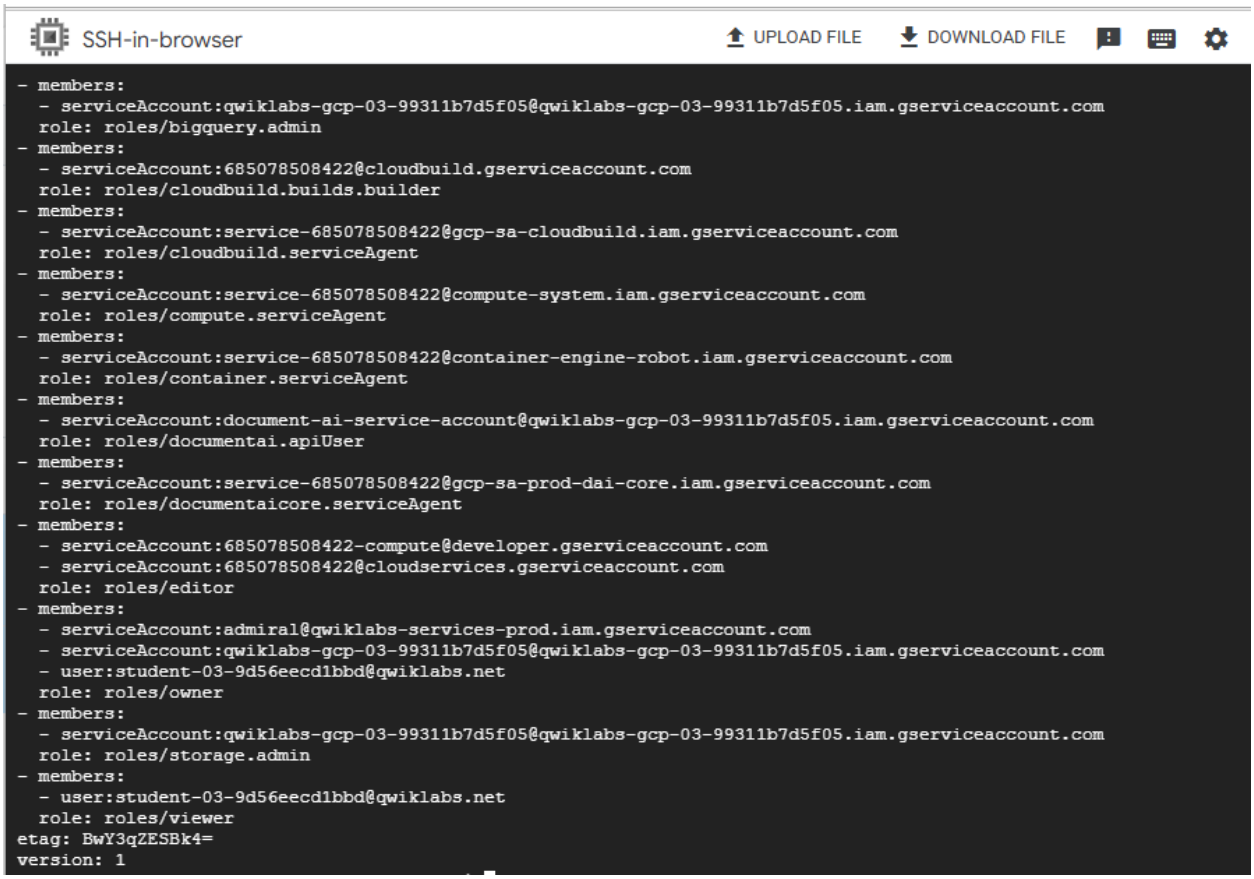
Authenticate API requests

In order to make requests to the Document AI API, you need to provide a valid credential. In this task create a service account, limit the permissions granted to

that service account to those required for the lab, and then generate a credential for that account that can be used to authenticate Document AI API requests.

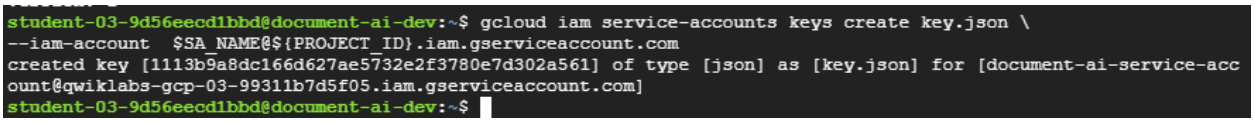
1. Set an environment variable with your Project ID, which you will use throughout this lab:

   export PROJECT_ID=$(gcloud config get-value core/project)



2. Create a new service account to access the Document AI API by using:

export SA_NAME="document-ai-service-account" gcloud iam service-accounts create $SA_NAME --display-name $SA_NAME



3. Bind the service account to the Document AI API user role:

gcloud projects add-iam-policy-binding ${PROJECT_ID}

--member="serviceAccount:$SA_NAME@${PROJECT_ID}.iam.gserviceaccount.com"

--role="roles/documentai.apiUser"

4. Create the credentials that will be used to log in as your new service account and save them in a JSON file called key.json in your working directory:

gcloud iam service-accounts keys create key.json

--iam-account $SA_NAME@${PROJECT_ID}.iam.gserviceaccount.com

```
student-03-9d56eecd1bbd@document-ai-dev:~$ export GOOGLE_APPLICATION_CREDENTIALS="$PWD/key.json"
student-03-9d56eecd1bbd@document-ai-dev:~$ echo $GOOGLE_APPLICATION_CREDENTIALS
/home/student-03-9d56eecd1bbd/key.json
student-03-9d56eecd1bbd@document-ai-dev:~$
```

5. Set the GOOGLE_APPLICATION_CREDENTIALS environment variable, which is used by the library to find your credentials, to point to the credentials file:

export GOOGLE_APPLICATION_CREDENTIALS="$PWD/key.json"

6. Check that the GOOGLE_APPLICATION_CREDENTIALS environment variable is set to the full path of the credentials JSON file you created earlier:

echo $GOOGLE_APPLICATION_CREDENTIALS

Download the sample form to the VM instance

Now you can download a sample form and then base64 encode it for submission to the Document AI API.

1. Enter the following command in the SSH window to download the sample form to your working directory:

   gsutil cp gs://cloud-training/gsp924/health-intake-form.pdf .

```
student-03-9d56eecd1bbd@document-ai-dev:~$ gsutil cp gs://cloud-training/gsp924/health-intake-form.pdf .
Copying gs://cloud-training/gsp924/health-intake-form.pdf...
/ [1 files][624.2 KiB/624.2 KiB]
Operation completed over 1 objects/624.2 KiB.
student-03-9d56eecd1bbd@document-ai-dev:~$
```

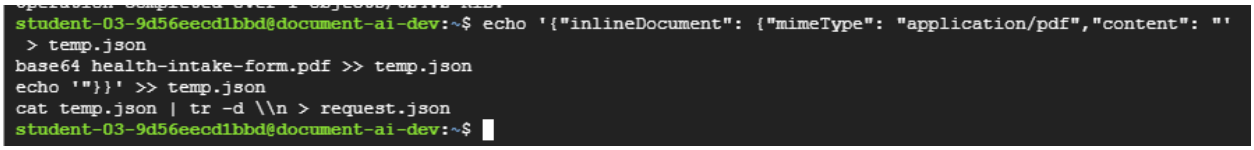
2. Create a JSON request file for submitting the base64 encoded form for processing:

   echo '{"inlineDocument": {"mimeType": "application/pdf","content": "' > temp.json

   base64 health-intake-form.pdf >> temp.json

   echo '"}}' >> temp.json

cat temp.json | tr -d \\n > request.json



Task 4. Make a synchronous process document request using curl

In this task you process the sample document by making a call to the synchronous Document AI API endpoint using curl.

1. Submit a form for processing via curl. The result will be stored in output.json:

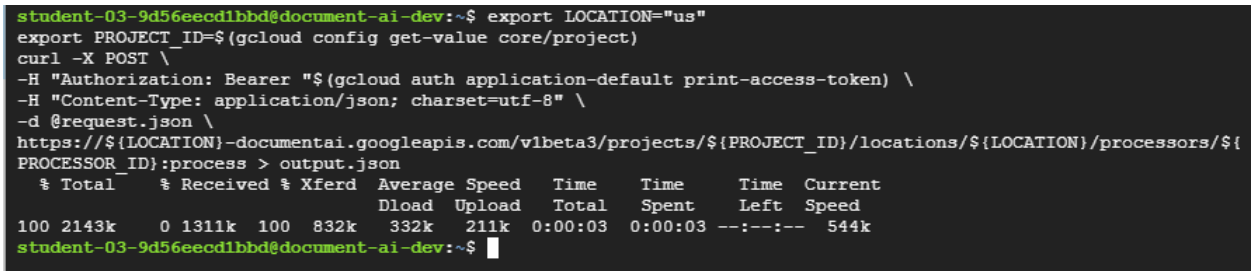
export LOCATION="us" export PROJECT_ID=$(gcloud config get-value core/project) curl -X POST

-H "Authorization: Bearer "$(gcloud auth application-default print-access-token)

-H "Content-Type: application/json; charset=utf-8"

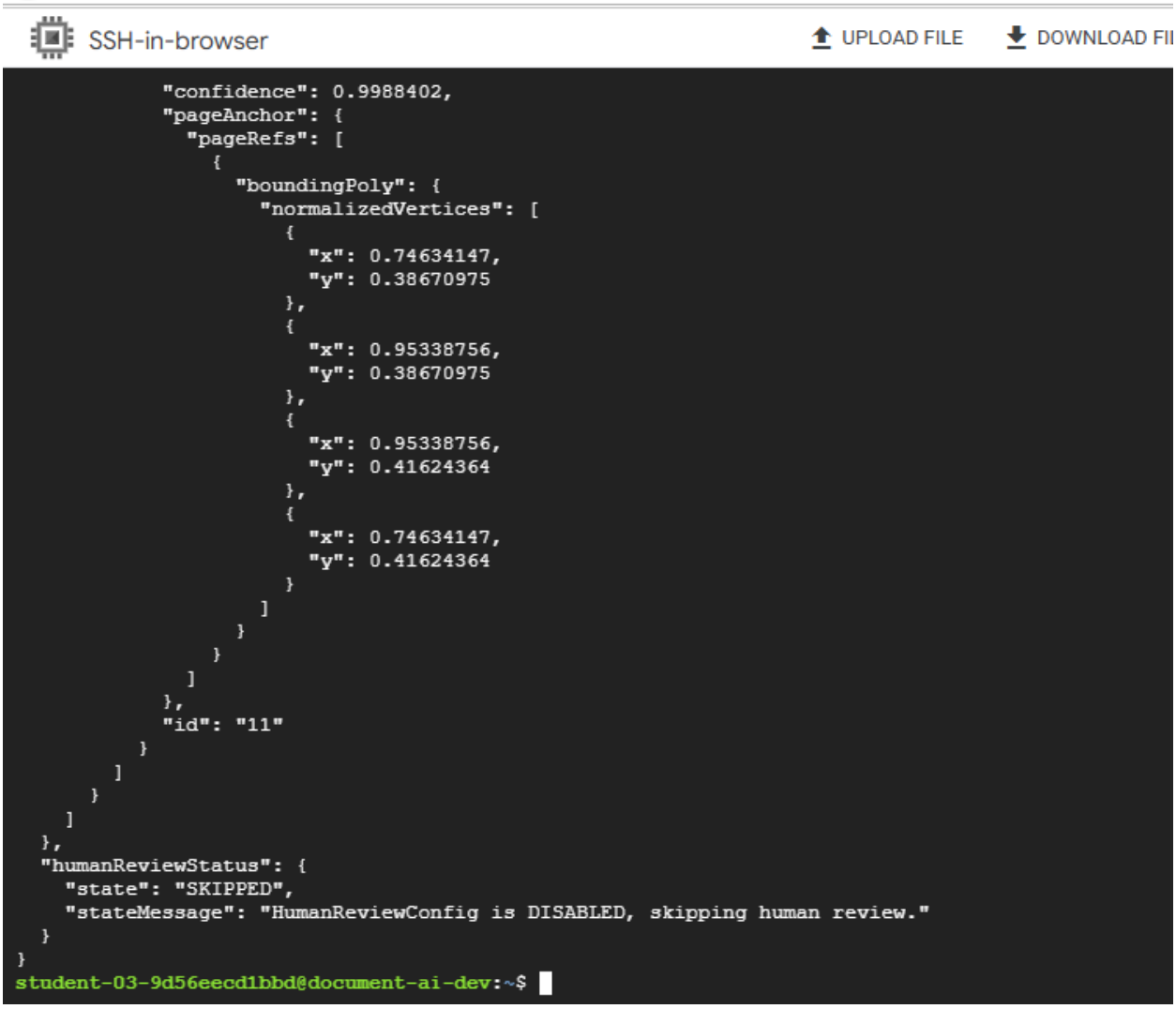
-d @request.json

https://${LOCATION}-documentai.googleapis.com/v1beta3/projects/${PROJECT_ID}/locations/${LOCATION}/processors/${PROCESSOR_ID}:process > output.json



2. Make sure your output.json file contains the results of the API call:

cat output.json



Extract the form entities

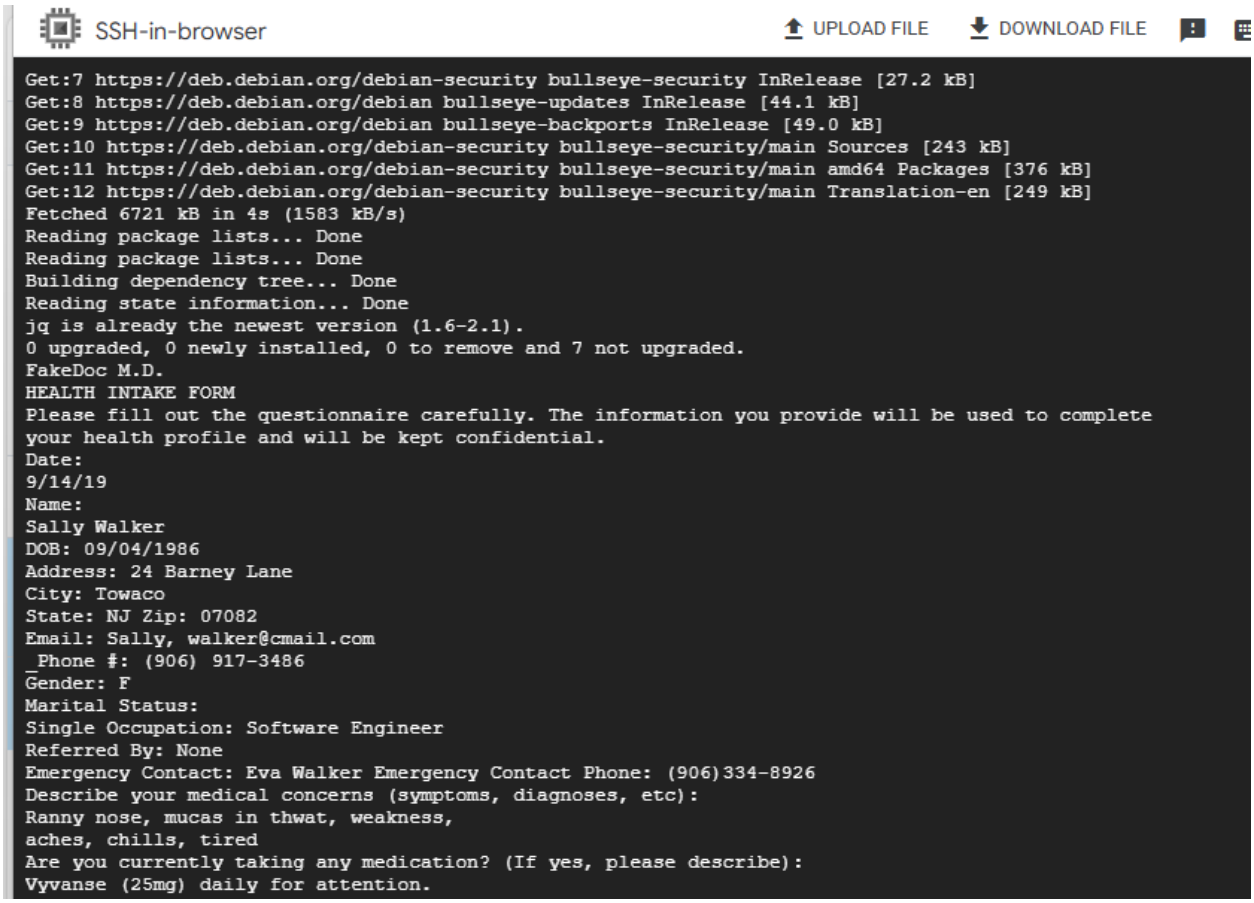
Next, explore some of the information extracted from the sample form.

1. Extract the raw text detected in the document as follows:

   sudo apt-get update

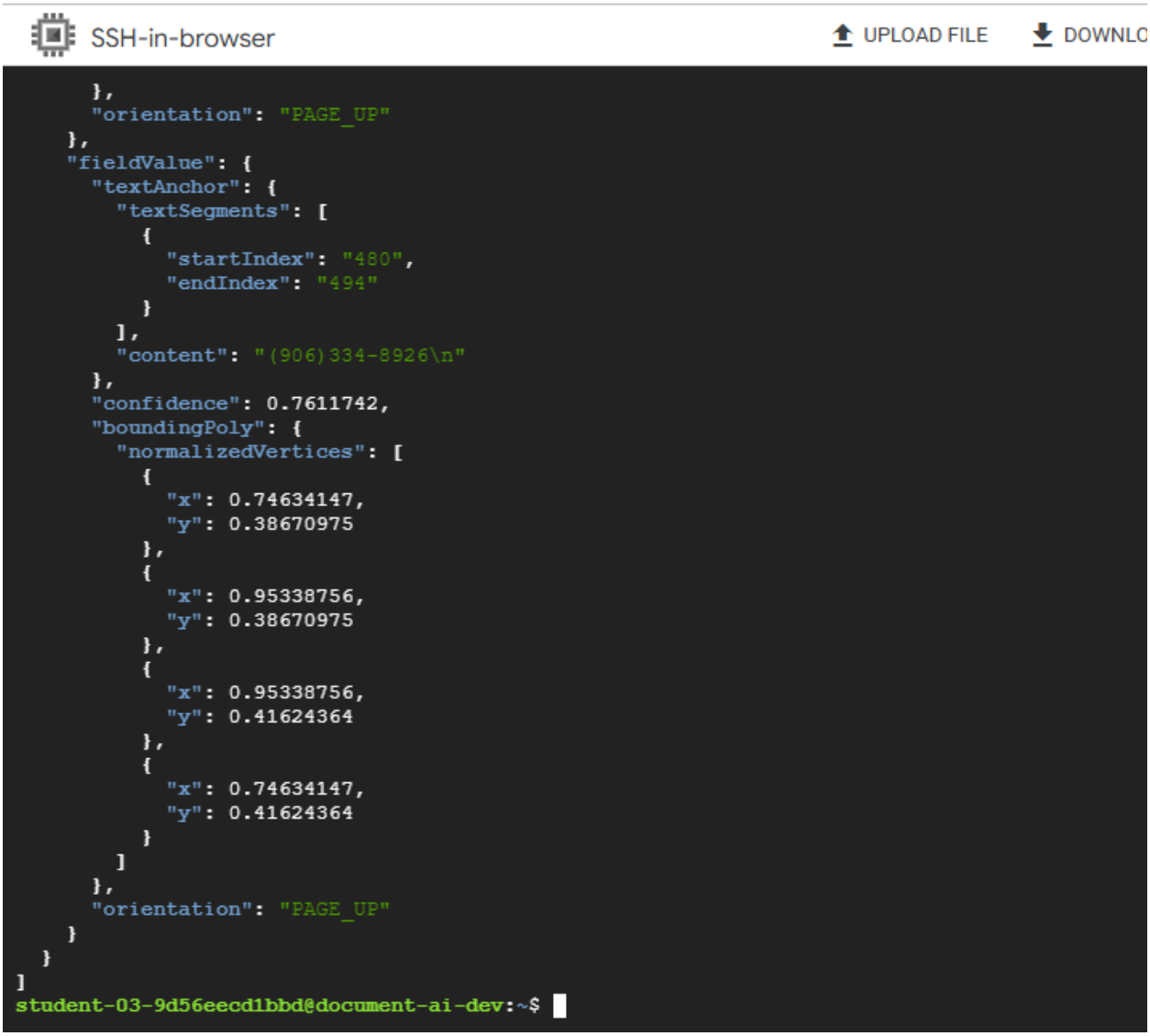
   sudo apt-get install jq

   cat output.json | jq -r ".document.text"



2. Extract the list of form fields detected by the form processor:

   cat output.json | jq -r ".document.pages[].formFields"



Task 5. Test a Document AI form processor using the Python client libraries

Make a synchronous call to the Document AI API using the Python Document AI client libraries.

Now you will process a document using the synchronous endpoint. For processing large amounts of documents at a time you can use the asynchronous API. To learn more about using the Document AI APIs, read the guide.

If you want to run Python scripts directly, you need to provide the appropriate credentials to those scripts, so that they can make calls to the API using a service account that has been configured with the correct permissions. To read more about how to configure this form of authentication, see the Authenticating as a service account documentation.

Configure your VM Instance to use the Document AI Python client

Now install the Python Google Cloud client libraries into the VM Instance.

1. Enter the following command in the SSH terminal shell to import the lab files into your VM Instance:

   gsutil cp gs://cloud-training/gsp924/synchronous_doc_ai.py .

```
student-03-9d56eecd1bbd@document-ai-dev:~$ gsutil cp gs://cloud-training/gsp924/synchronous_doc_ai.py .
Copying gs://cloud-training/gsp924/synchronous_doc_ai.py...
/ [1 files][  3.1 KiB/  3.1 KiB]
Operation completed over 1 objects/3.1 KiB.
student-03-9d56eecd1bbd@document-ai-dev:~$
```

2. Enter the following command to install the Python client libraries required for Document AI and the other libraries required for this lab:

   sudo apt install python3-pip

   python3 -m pip install --upgrade google-cloud-documentai google-cloud-storage prettytable

```
student-03-9d56eecd1bbd@document-ai-dev:~$ sudo apt install python3-pip
python3 -m pip install --upgrade google-cloud-documentai google-cloud-storage prettytable
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-pip is already the newest version (20.3.4-4+deb11u1).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
Collecting google-cloud-documentai
  Downloading google_cloud_documentai-3.5.0-py3-none-any.whl (299 kB)
     |████████████████████████████████| 299 kB 7.2 MB/s
Collecting google-cloud-storage
  Downloading google_cloud_storage-3.1.0-py2.py3-none-any.whl (174 kB)
     |████████████████████████████████| 174 kB 51.1 MB/s
Collecting prettytable
  Downloading prettytable-3.16.0-py3-none-any.whl (33 kB)
Collecting google-auth!=2.24.0,!=2.25.0,<3.0.0,>=2.14.1
  Downloading google_auth-2.40.3-py2.py3-none-any.whl (216 kB)
     |████████████████████████████████| 216 kB 47.4 MB/s
Collecting google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*
,!=2.9.*,<3.0.0,>=1.34.1
  Downloading google_api_core-2.25.1-py3-none-any.whl (160 kB)
     |████████████████████████████████| 160 kB 51.9 MB/s
Collecting proto-plus<2.0.0,>=1.22.3
  Downloading proto_plus-1.26.1-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 10.5 MB/s
```

Review the Document AI API Python code

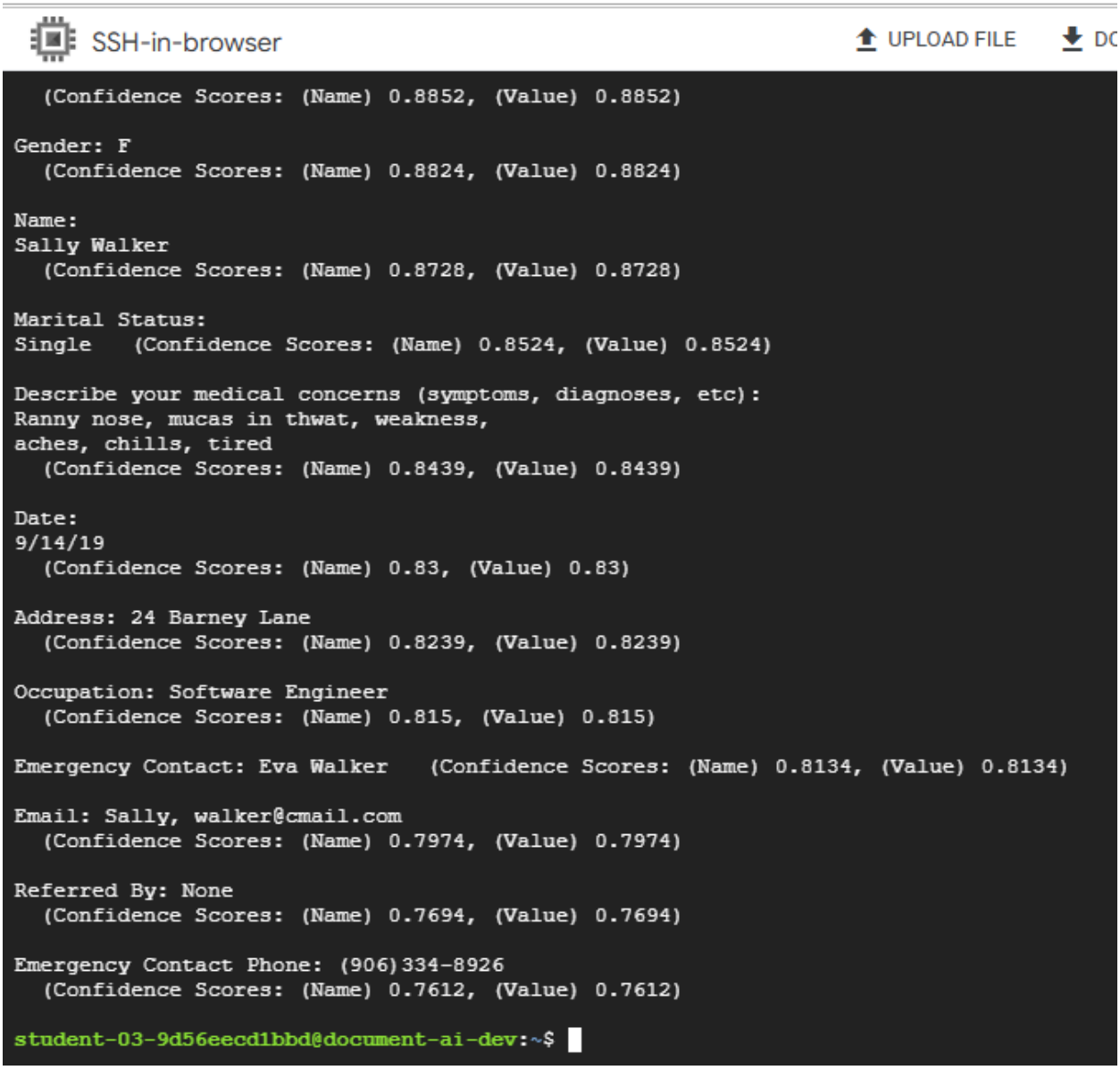
Take a minute to review the Python code in the sample file. You can use an editor of your choice, such as vi or nano, to review the code in the SSH session or you can use the command from the previous section to copy the example code into the Cloud Shell and use the Code Editor to view the source code if you prefer.

Task 6. Run the Document AI Python code

Execute the sample code and process the same file as before.

1. Create environment variables for the Project ID and the IAM service account credentials file:

   export PROJECT_ID=$(gcloud config get-value core/project) export GOOGLE_APPLICATION_CREDENTIALS="$PWD/key.json"



2. Call the synchronous_doc_ai.py python program with the parameters it requires:

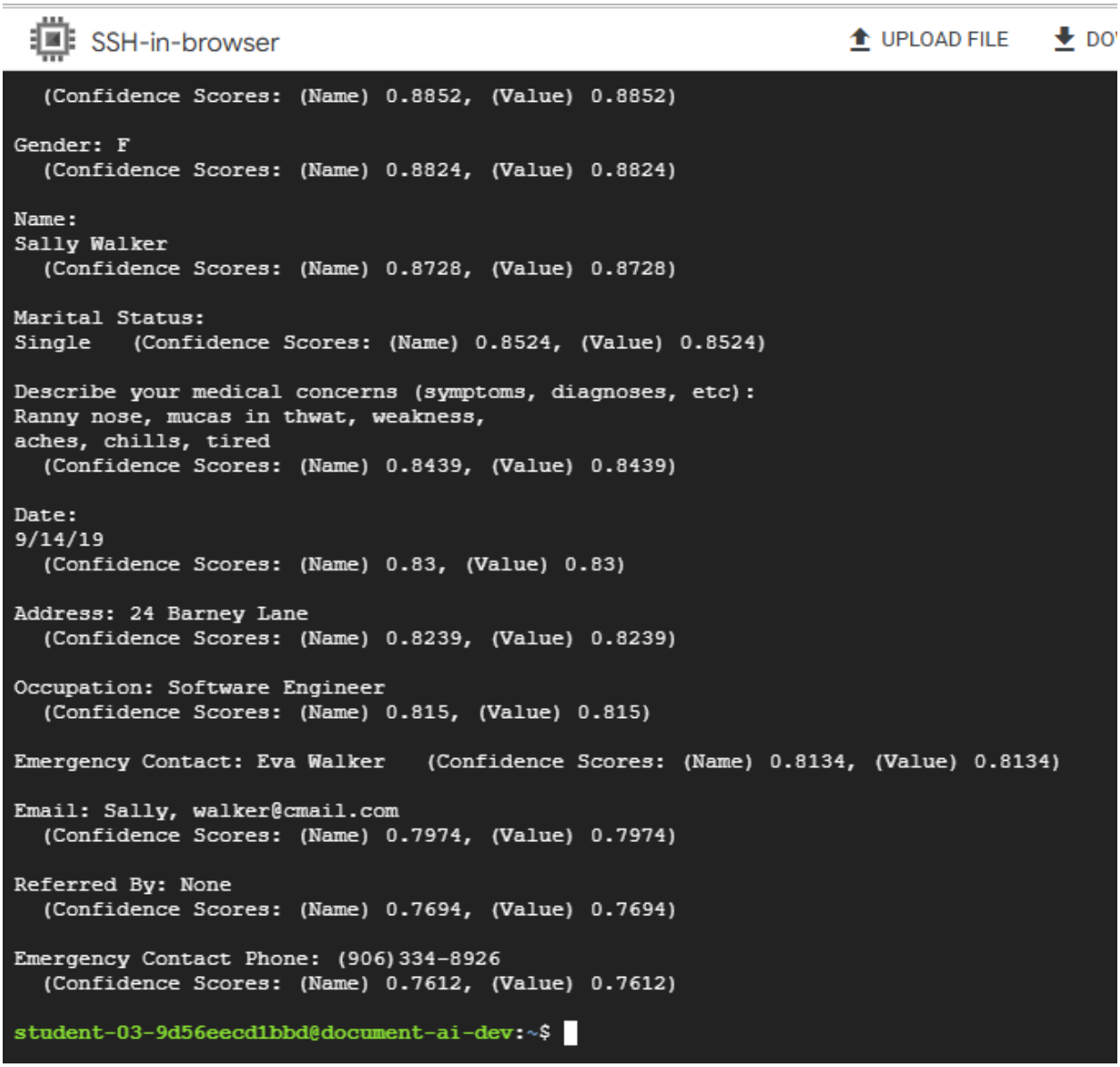
   python3 synchronous_doc_ai.py

   --project_id=$PROJECT_ID

   --processor_id=$PROCESSOR_ID

--location=us

--file_name=health-intake-form.pdf | tee results.txt



6. Conclusion

- Successfully **created and tested** a **Document AI processor**.
- Extracted **structured data** (key-value pairs) from an unstructured PDF.
- Demonstrated **three methods** of processing documents:
  - **Google Cloud Console** (UI-based testing).

- **Command Line (**curl**)** (API testing).
- **Python Client Libraries** (programmatic integration).

Future Improvements

- **Batch Processing**: Use **asynchronous API** for large document volumes.
- **Custom Processors**: Train a **custom processor** for domain-specific documents.
- **Post-Processing**: Integrate with **BigQuery** or **Cloud Storage** for analytics.