ETL Processing on Google Cloud Using Dataflow and BigQuery

Overview

In this lab you will build several data pipelines that will ingest data from a publicly available dataset into BigQuery, using these Google Cloud services:

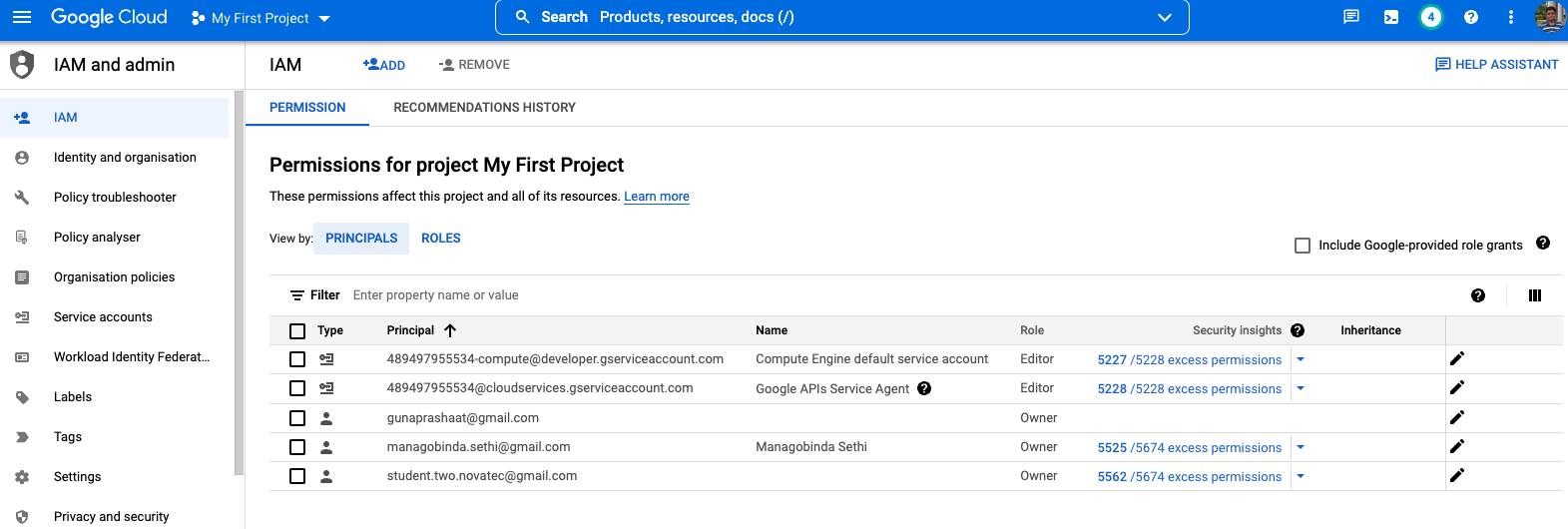
* **Cloud Storage**
* **Dataflow**
* **BigQuery tables**

You will create your own data pipeline, including the design considerations, as well as implementation details, to ensure that your prototype meets the requirements. Be sure to open the Python files and read the comments when instructed to.

Check project permissions

Before you begin your work on Google Cloud, you need to ensure that your project has the correct permissions within Identity and Access Management (IAM).

1. In the Google Cloud console, on the **Navigation menu** (), click **IAM & Admin** > **IAM**.
2. Confirm that the default compute Service Account {project-number}-compute@developer.gserviceaccount.com is present and has the editor role assigned. The account prefix is the project number, which you can find on **Navigation menu** > **Home**.



If the account is not present in IAM or does not have the editor role, follow the steps below to assign the required role.

* In the Google Cloud console, on the **Navigation menu**, click **Home**.

Copy the project number (e.g. 489497955534).

* On the **Navigation menu**, click **IAM & Admin** > **IAM**.
* At the top of the **IAM** page, click **Add**.
* For **New principals**, type:

{project-number}-compute@developer.gserviceaccount.com

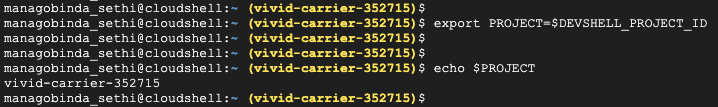
Replace {project-number} with your project number.

* For **Role**, select **Project** (or Basic) > **Editor**. Click **Save**.

Download the Starter Code

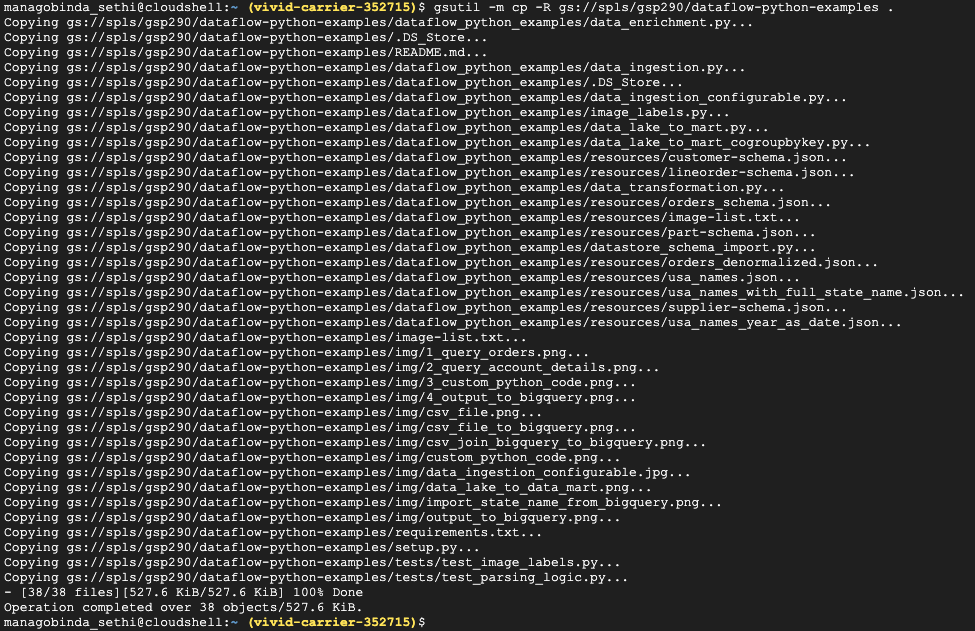
Open a session in Cloud Shell and store the **GCP\_PROJECT\_ID** in the **PROJECT** environment variable :

export PROJECT=$DEVSHELL\_PROJECT\_ID



Run the following command to copy and expand the source code for the lab into your Cloud Shell session:

gsutil -m cp -R gs://spls/gsp290/dataflow-python-examples .



**Create a Cloud Storage bucket**

Use the make bucket command to create a new regional bucket in the us-central1 region within your project:

gsutil mb -c regional -l us-central1 gs://$PROJECT

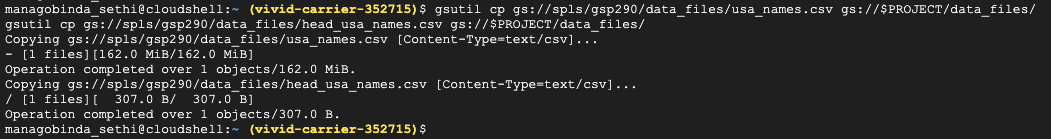


Copy files to your bucket

Use the gsutil command to copy sample data files to the Cloud Storage bucket you just created:

gsutil cp gs://spls/gsp290/data\_files/usa\_names.csv gs://$PROJECT/data\_files/

gsutil cp gs://spls/gsp290/data\_files/head\_usa\_names.csv gs://$PROJECT/data\_files/



Create the BigQuery dataset

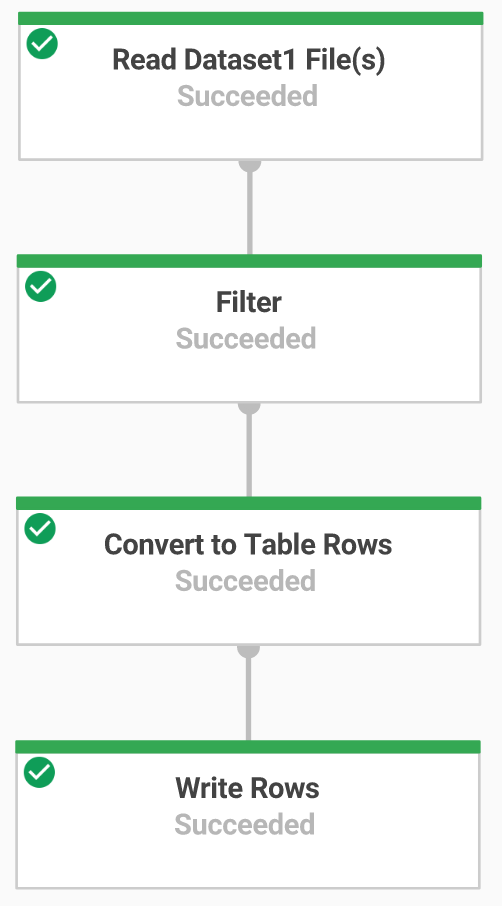
Create a dataset in BigQuery called lake. This is where all of your tables will be loaded in BigQuery:

bq mk lake



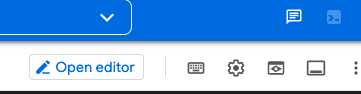
Build a Dataflow pipeline

In this section you will create an append-only Dataflow which will ingest data into the BigQuery table. You can use the built-in code editor which will allow you to view and edit the code in the Cloud Console.



Step 1 - Open the Cloud Shell code editor

Navigate to the source code by clicking on the **Open Editor** icon in Cloud Shell:



**Note:** If you cannot see this icon, close the Navigation menu. Click in the upper left corner.

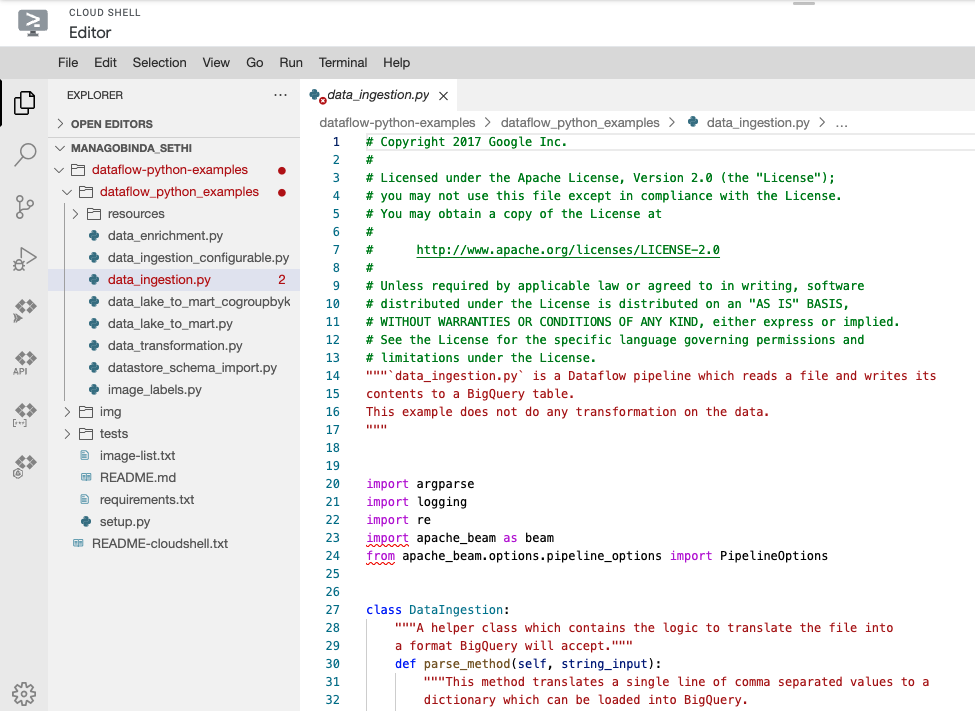
Step 2 - Data ingestion

You will now build a Dataflow pipeline with a TextIO source and a BigQueryIO destination to ingest data into BigQuery. More specifically, it will:

* Ingest the files from Cloud Storage.
* Filter out the header row in the files.
* Convert the lines read to dictionary objects.
* Output the rows to BigQuery.

**Review pipeline Python code**

In the Cloud Shell code editor navigate to training-data-analyst > courses > dw-pso > dataflow\_python\_examples and open the data\_ingestion.py file.



Read through the comments in the file, which explain what the code is doing. This code will populate the data in BigQuery.



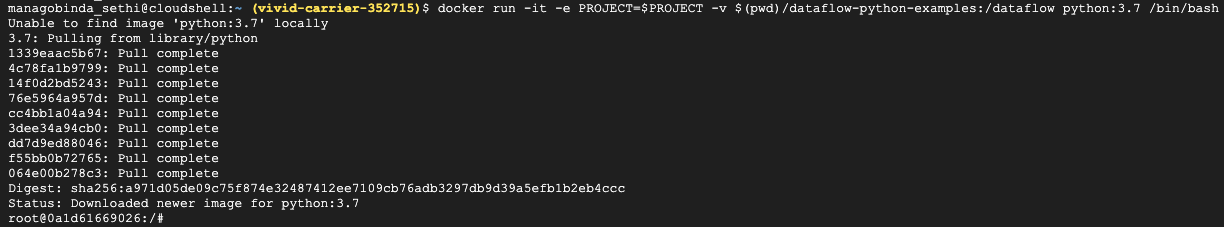
**Run the Apache Beam pipeline**

Return to your Cloud Shell session for this step. You will now do a bit of set up for the required python libraries.

The Dataflow job in this lab requires Python3.7. To ensure you're on the proper version, you will run the process on a Python 3.7 Docker container.

Run the following in Cloud Shell:

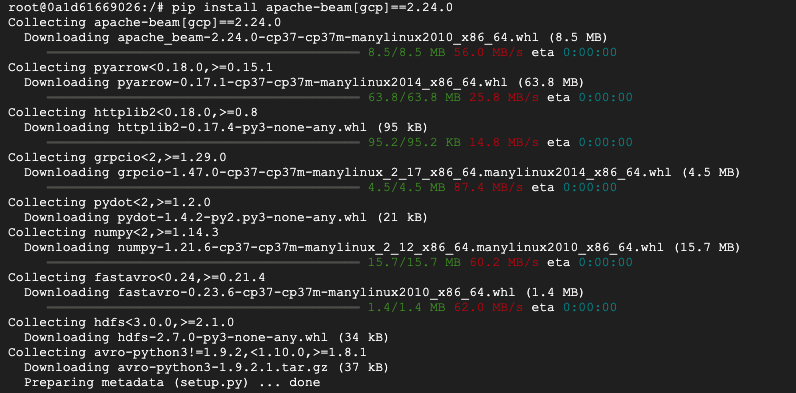
docker run -it -e PROJECT=$PROJECT -v $(pwd)/dataflow-python-examples:/dataflow python:3.7 /bin/bash

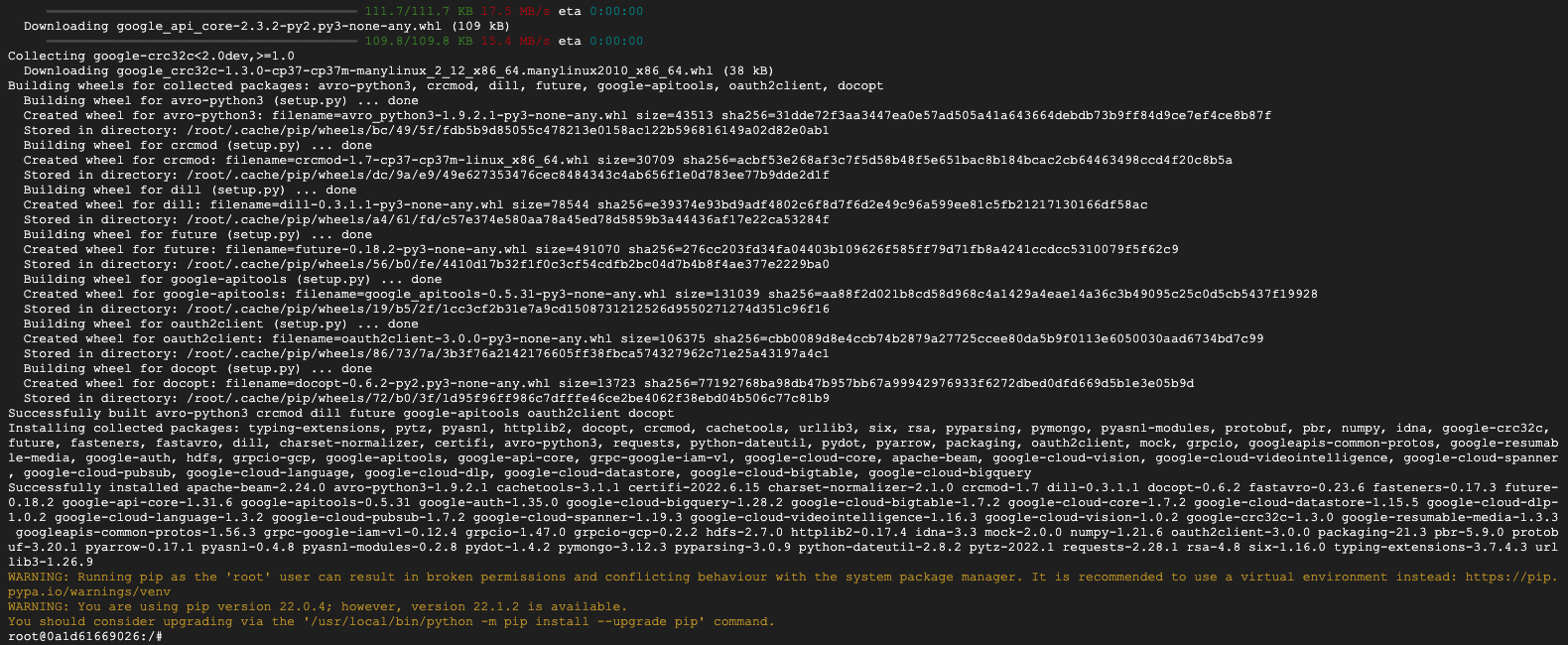


This command will pull a Docker container with the latest stable version of Python 3.7 and execute a command shell to run the next commands within the container. The -v flag provides the source code as a volume for the container so that we can edit in Cloud Shell editor and still access it within the container.

Once the container finishes pulling, run the following to install apache-beam:

pip install apache-beam[gcp]==2.24.0





Next, change directories into where you linked the source code:

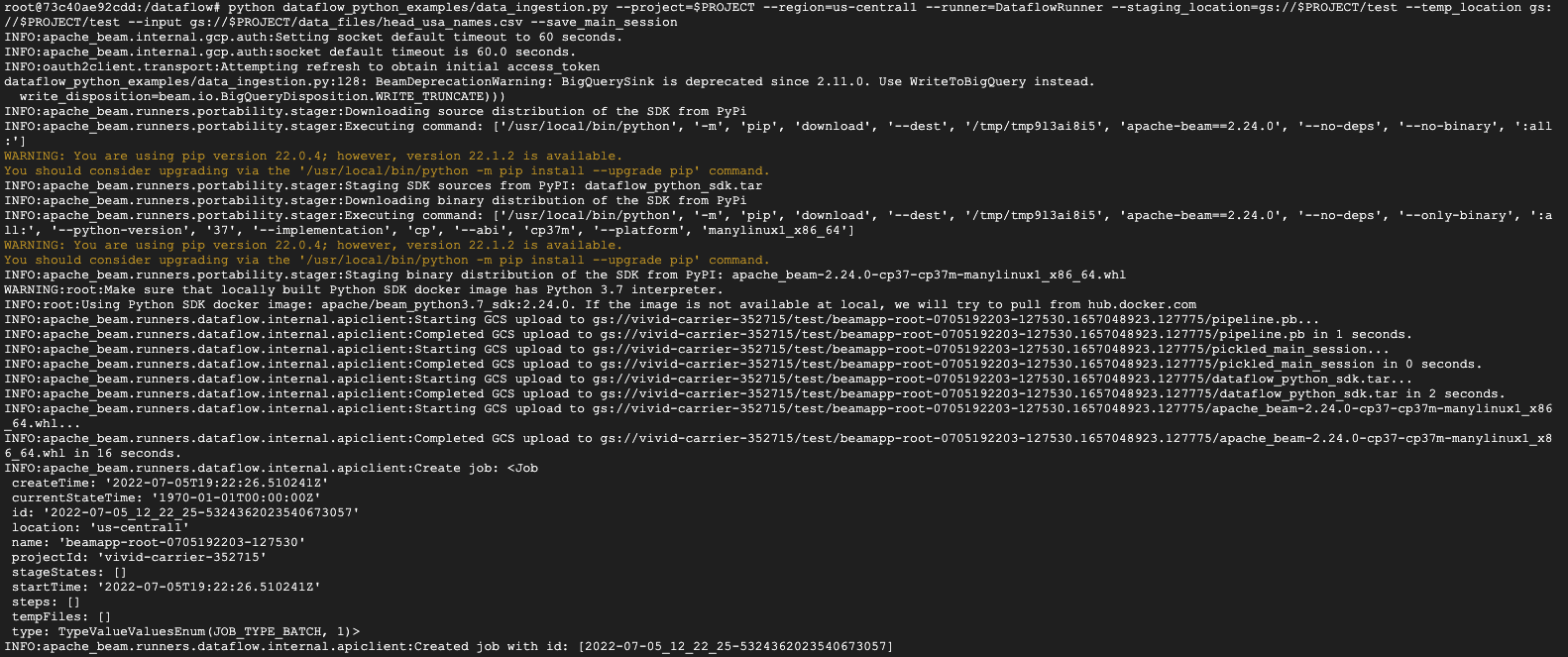
cd dataflow/

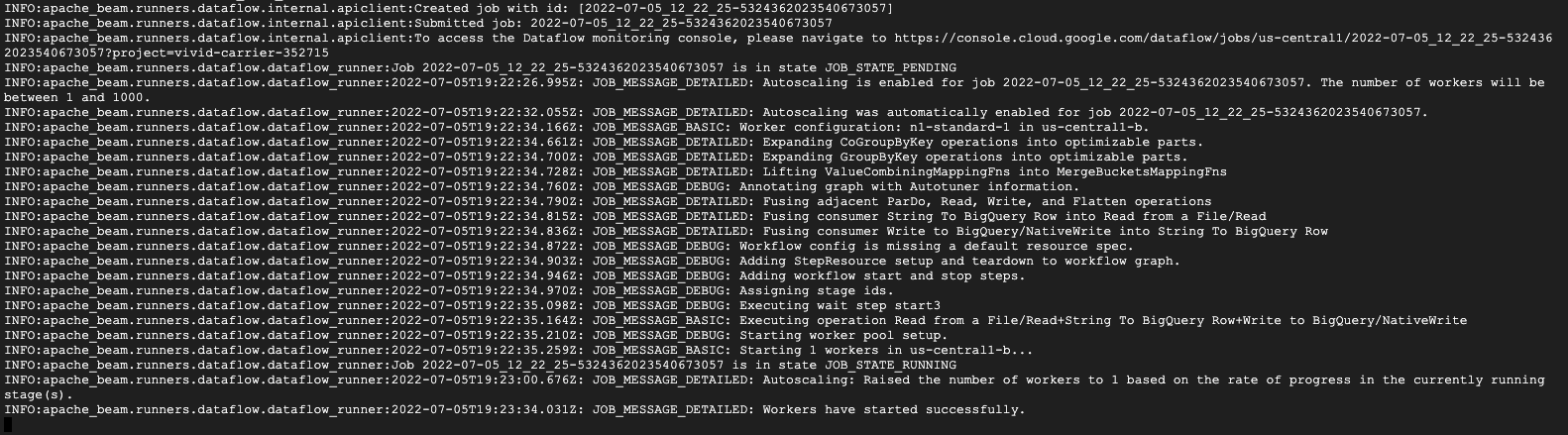


You will run the Dataflow pipeline in the cloud.

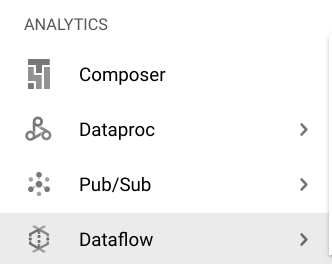
The following will spin up the workers required, and shut them down when complete:

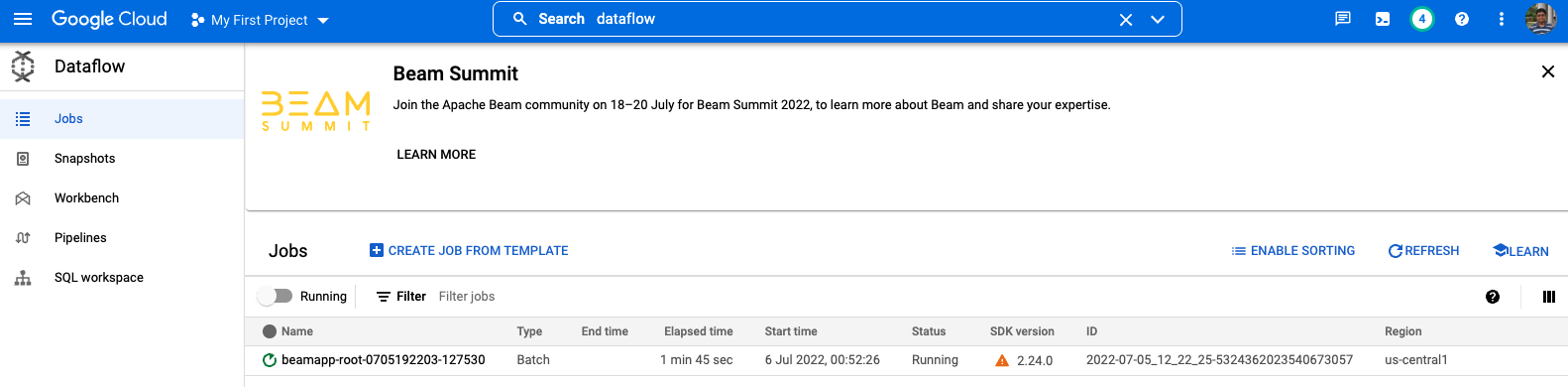
python dataflow\_python\_examples/data\_ingestion.py --project=$PROJECT --region=us-central1 --runner=DataflowRunner --staging\_location=gs://$PROJECT/test --temp\_location gs://$PROJECT/test --input gs://$PROJECT/data\_files/head\_usa\_names.csv --save\_main\_session

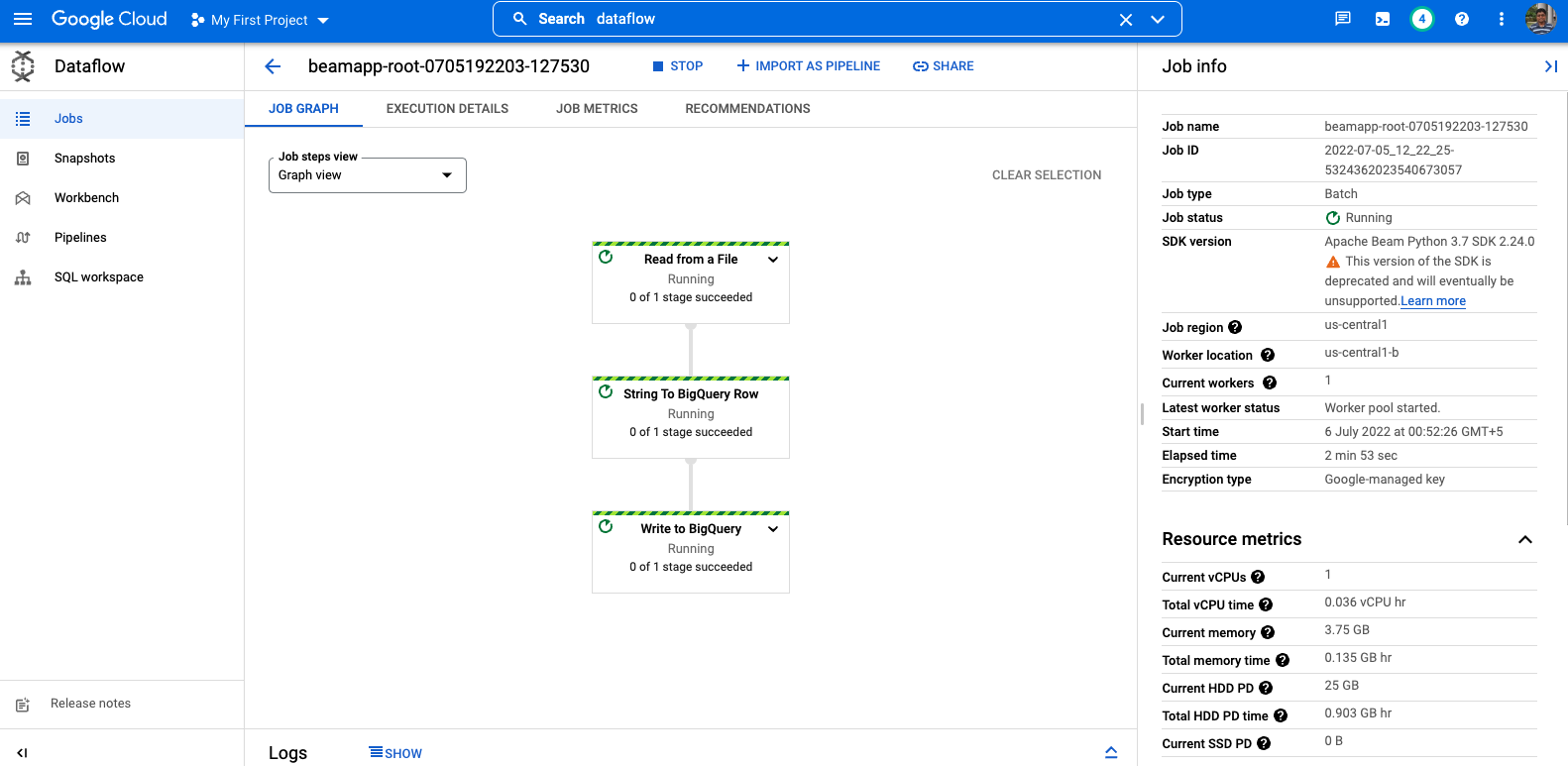


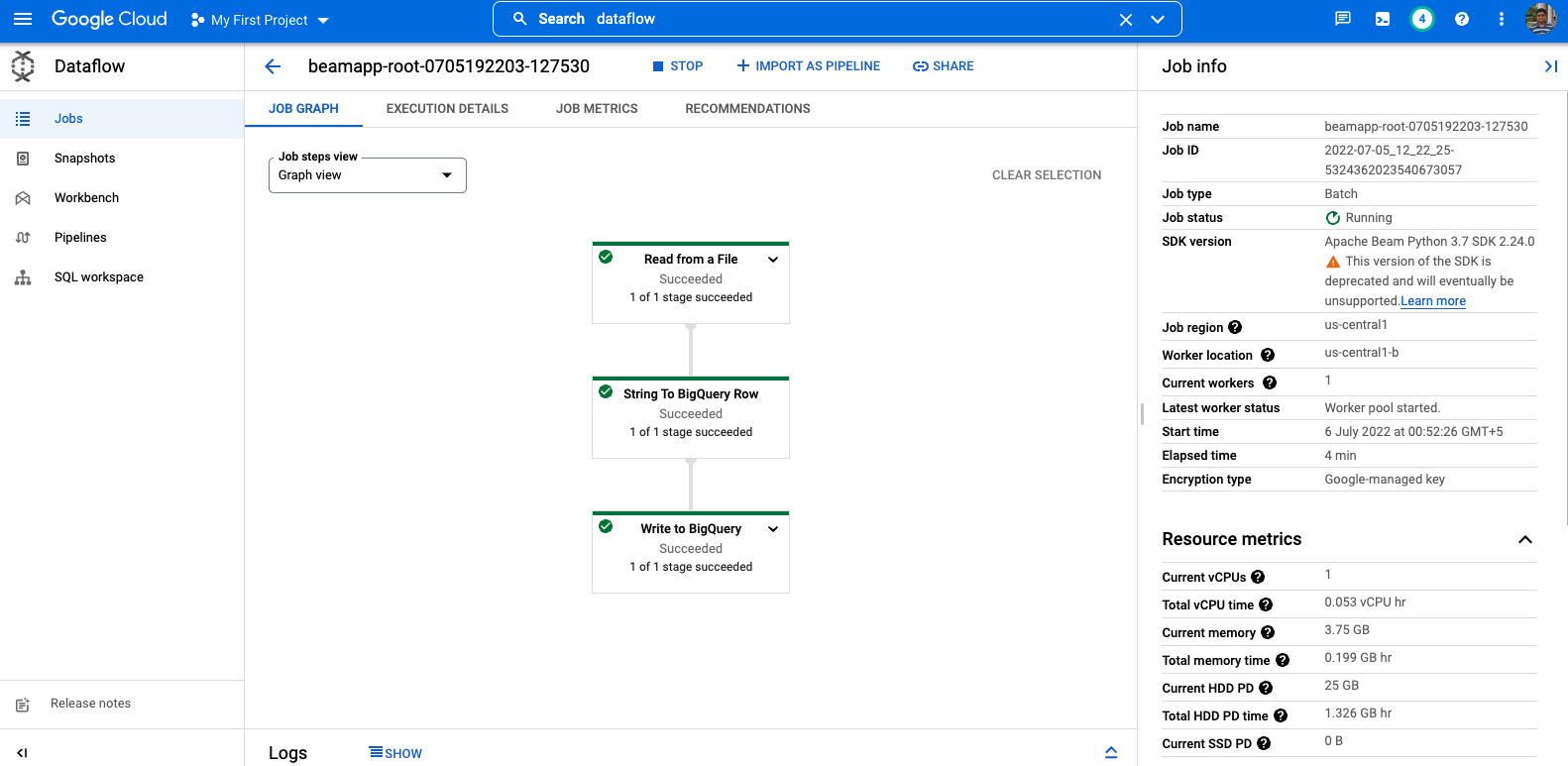


Return to the Cloud Console and open the **Navigation menu** > **Dataflow** to view the status of your job.







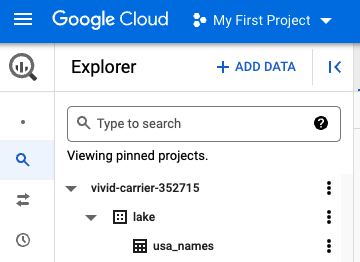


Click on the name of your job to watch it's progress. Once your **Job Status** is **Succeeded**.

Navigate to BigQuery (**Navigation menu** > **BigQuery**) see that your data has been populated.



Click on your project name to see the **usa\_names** table under the lake dataset.



Click on the table then navigate to the **Preview** tab to see examples of the usa\_names data.

**Note:** If you don't see the usa\_names table, try refreshing the page or view the tables using the classic BigQuery UI.

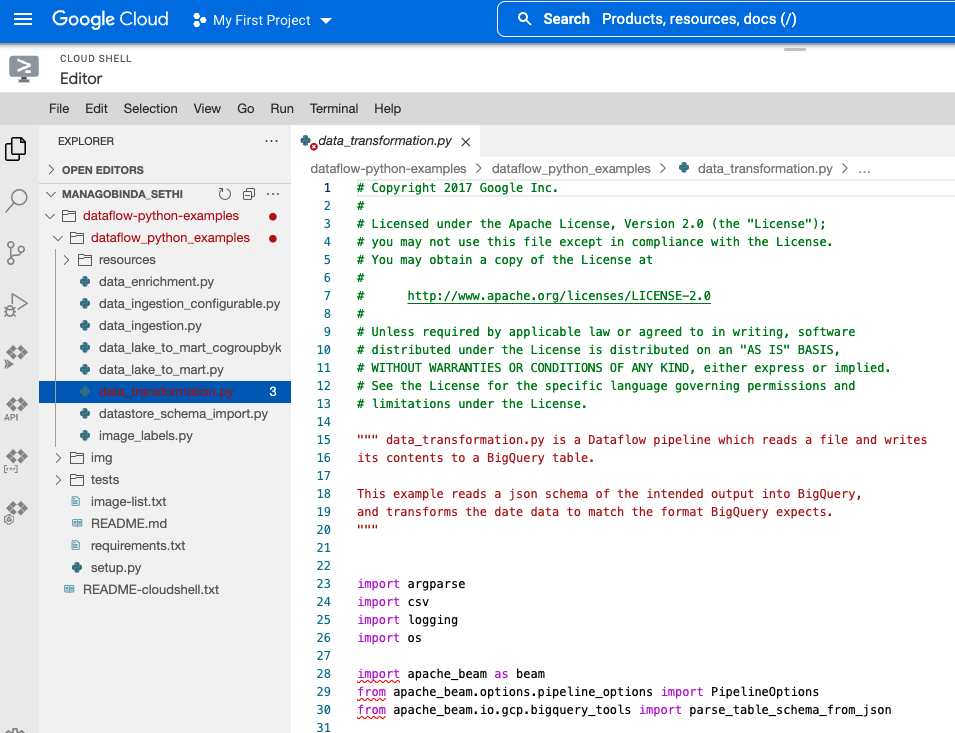
Step 3 - Data transformation

You will now build a Dataflow pipeline with a TextIO source and a BigQueryIO destination to ingest data into BigQuery. More specifically, you will:

* Ingest the files from Cloud Storage.
* Convert the lines read to dictionary objects.
* Transform the data which contains the year to a format BigQuery understands as a date.
* Output the rows to BigQuery.

**Review pipeline Python code**

Navigate to data\_transformation.py and open it in the Cloud Shell code editor. Read through the comments in the file which explain what the code is doing.

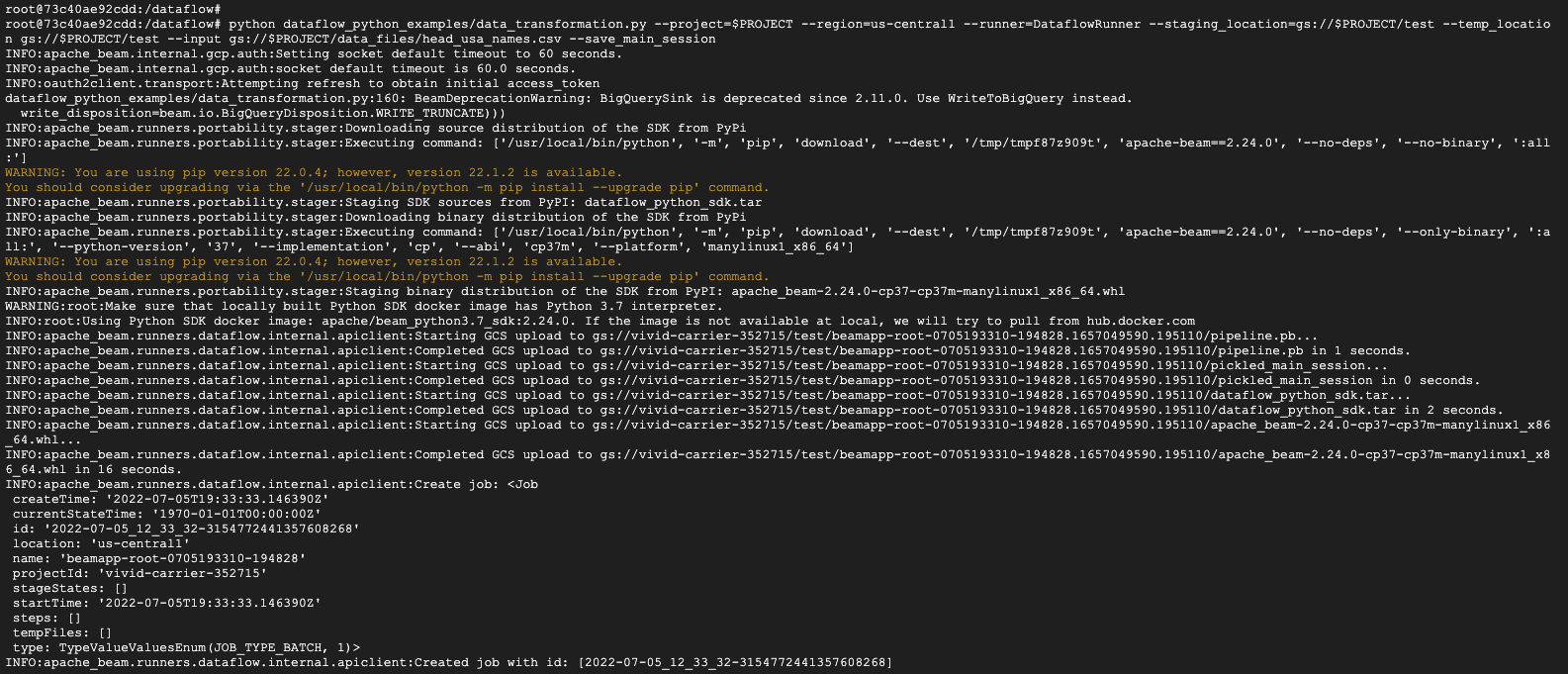


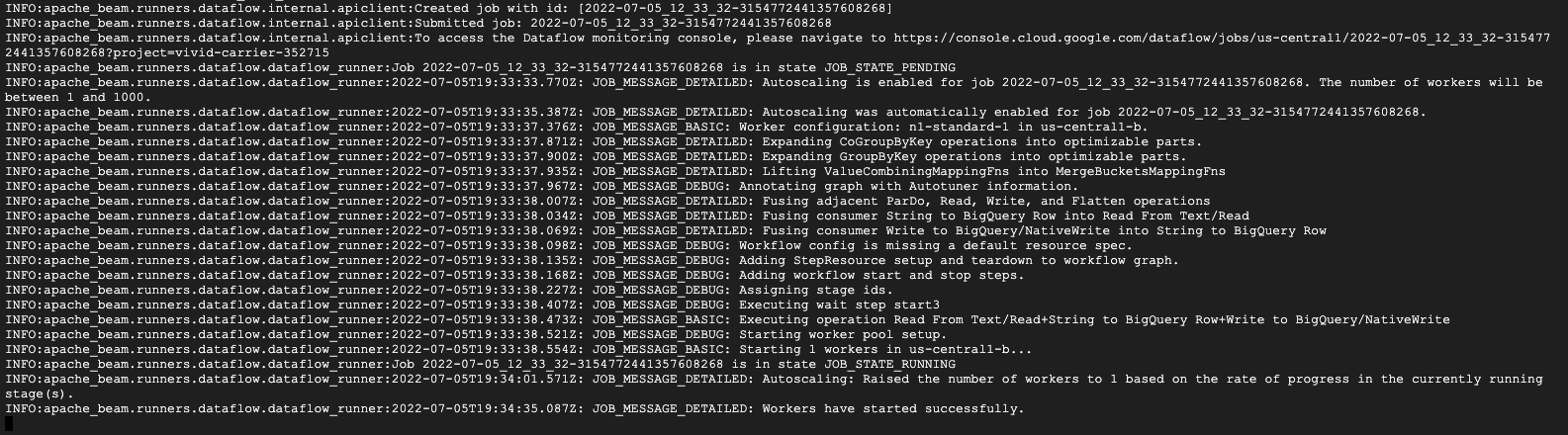
**Run the Apache Beam pipeline**

You will run the Dataflow pipeline in the cloud. This will spin up the workers required, and shut them down when complete.

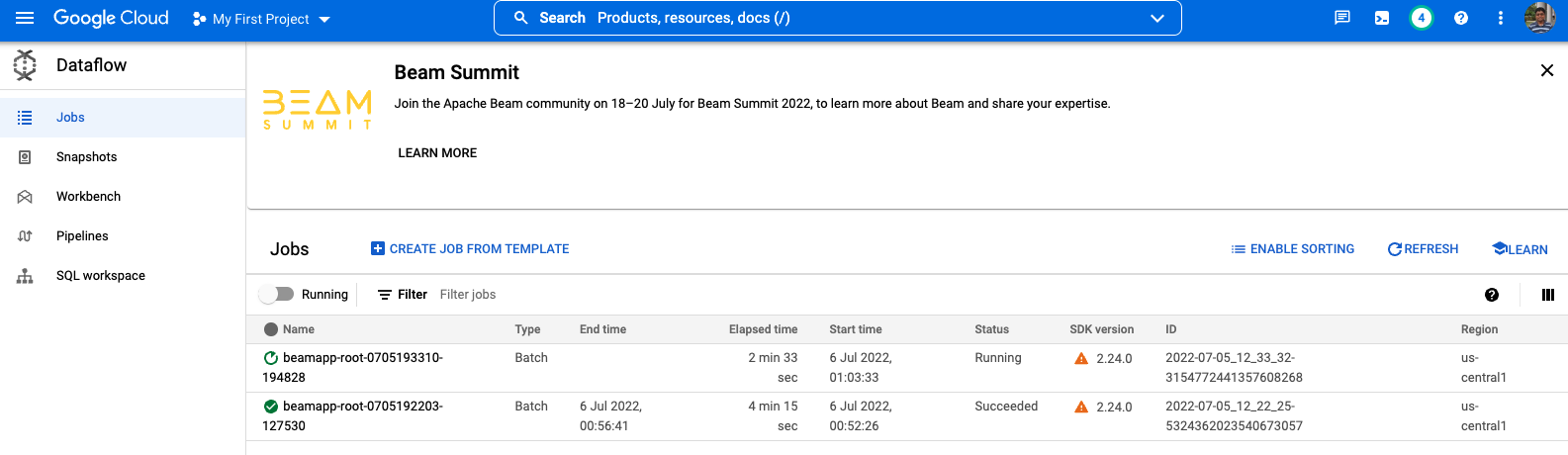
Run the following commands to do so:

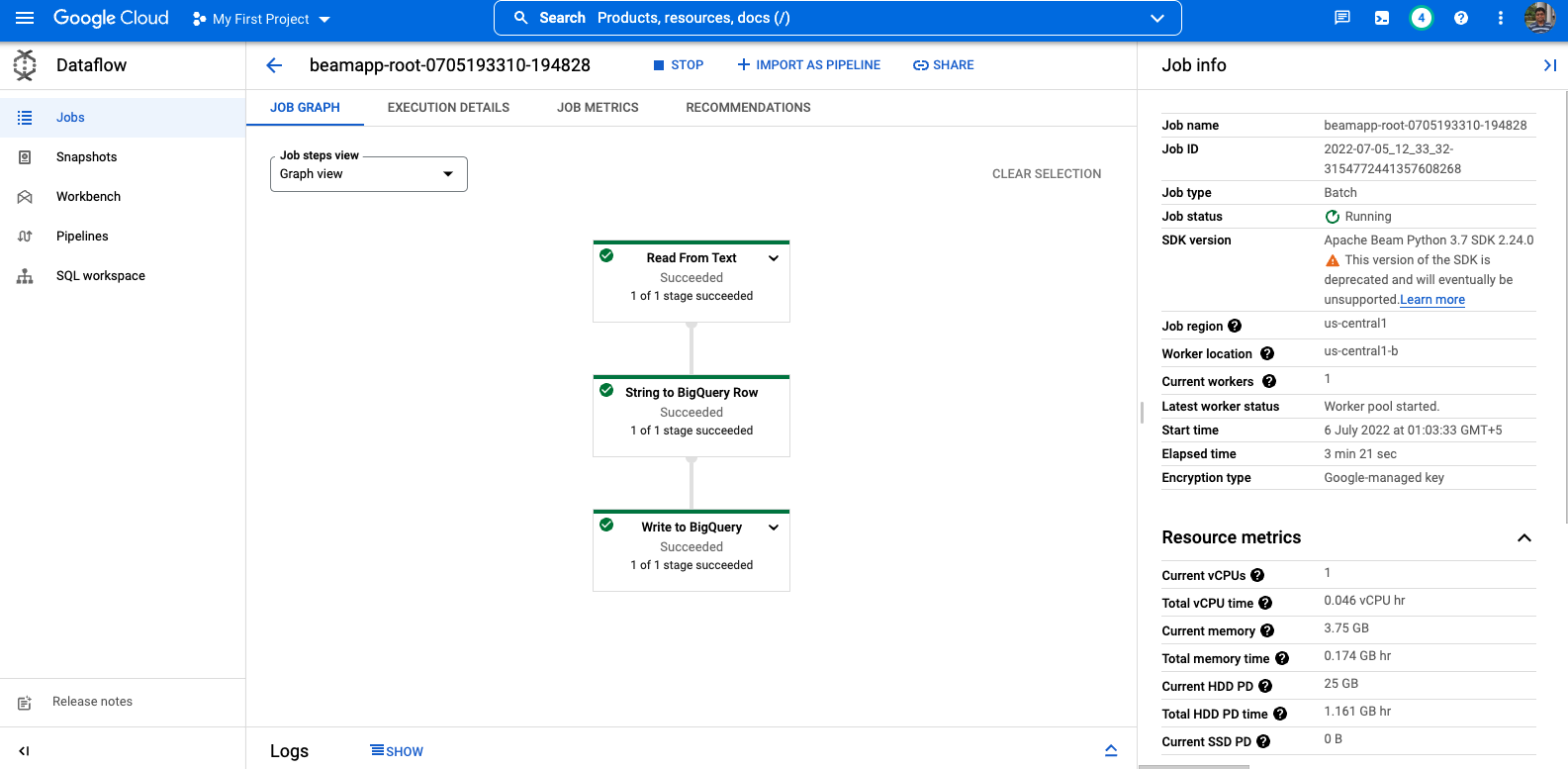
python dataflow\_python\_examples/data\_transformation.py --project=$PROJECT --region=us-central1 --runner=DataflowRunner --staging\_location=gs://$PROJECT/test --temp\_location gs://$PROJECT/test --input gs://$PROJECT/data\_files/head\_usa\_names.csv --save\_main\_session





Navigate to **Navigation menu** > **Dataflow** and click on the name of this job view the status.

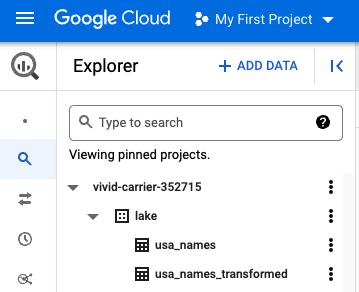




When your **Job Status** is **Succeeded** in the Dataflow Job status screen, navigate to BigQuery to check to see that your data has been populated.

You should see the **usa\_names\_transformed** table under the lake dataset.

Click on the table and navigate to the **Preview** tab to see examples of the usa\_names\_transformed data.



**Note:** If you don't see the usa\_names\_transformed table, try refreshing the page or view the tables using the classic BigQuery UI.

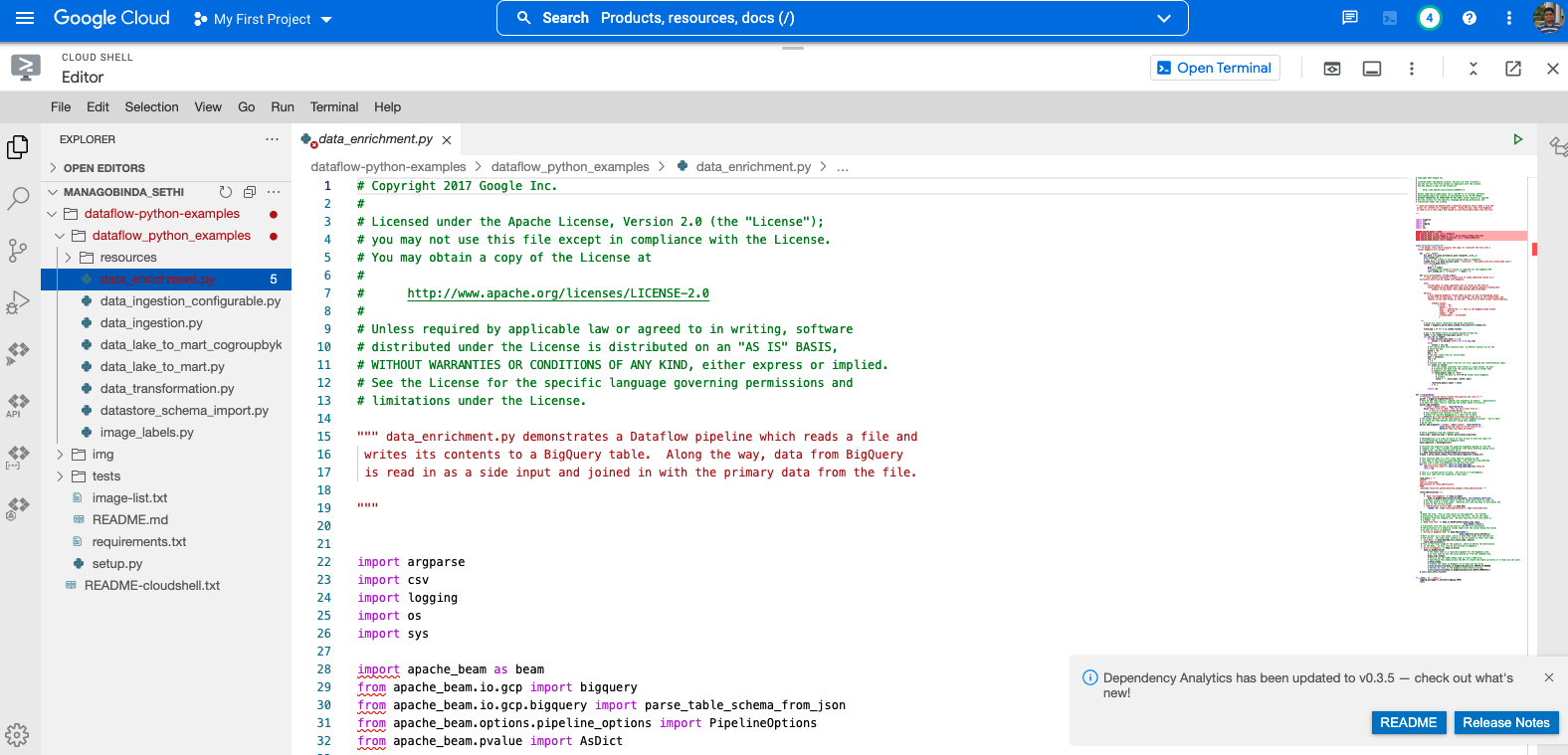
Step 4 - Data enrichment

You will now build a Dataflow pipeline with a TextIO source and a BigQueryIO destination to ingest data into BigQuery. More specifically, you will:

* Ingest the files from Cloud Storage.
* Filter out the header row in the files.
* Convert the lines read to dictionary objects.
* Output the rows to BigQuery.

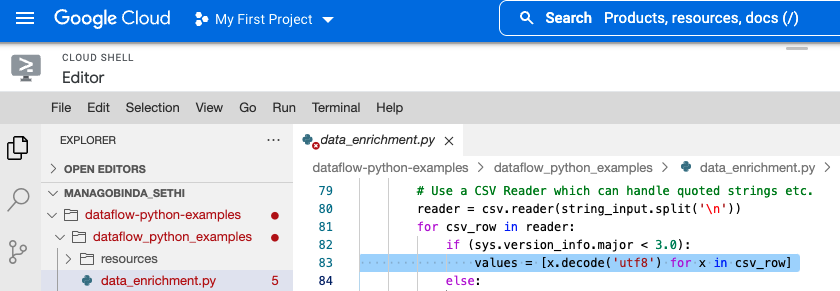
**Review pipeline Python code**

Navigate to data\_enrichment.py and open it in the Cloud Shell code editor. Check out the comments which explain what the code is doing. This code will populate the data in BigQuery.



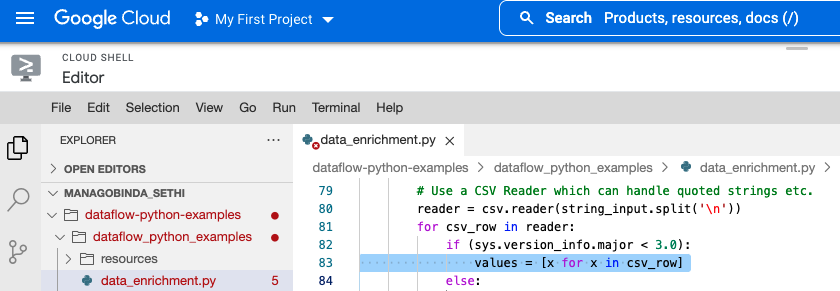
Line 83 currently looks like:

values = [x.decode('utf8') for x in csv\_row]



Edit it so it looks like the following:

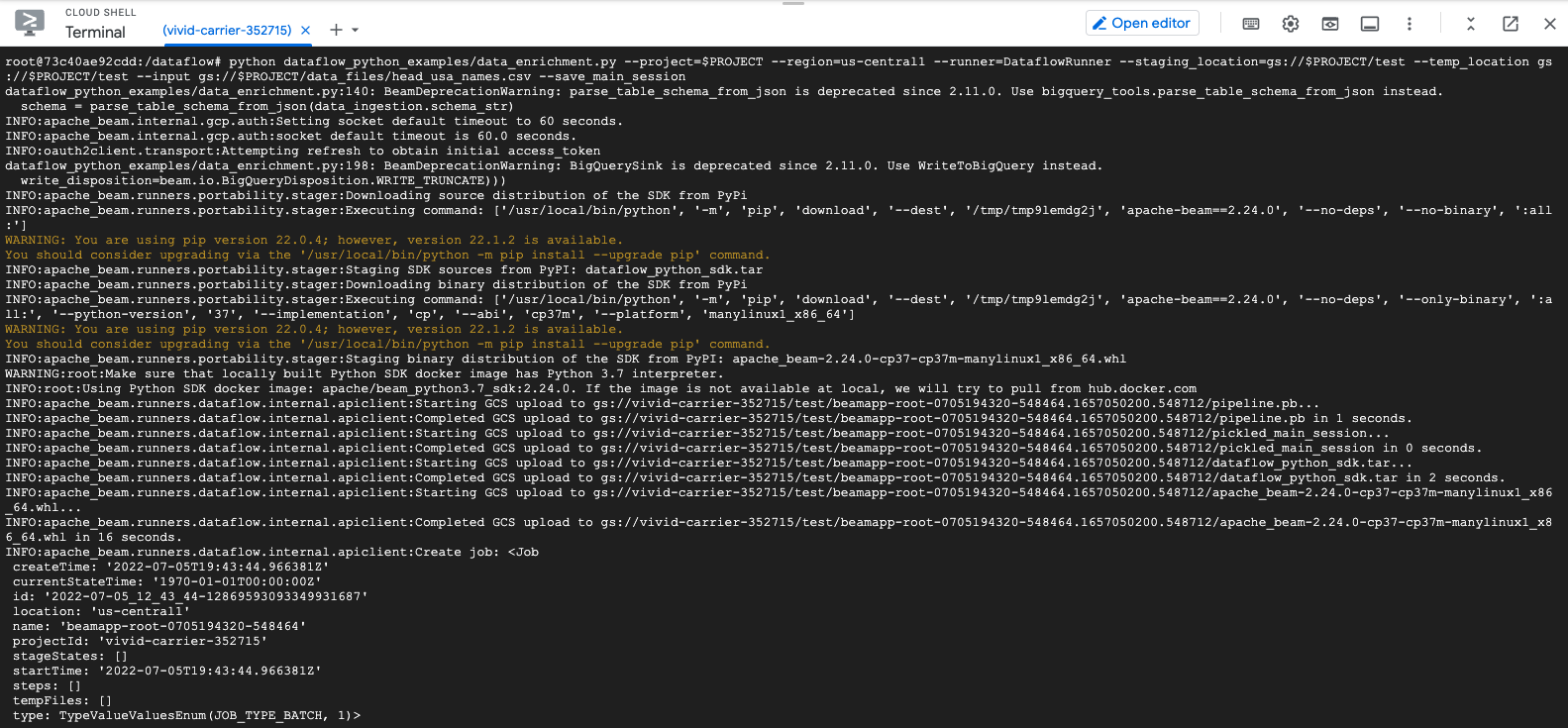
values = [x for x in csv\_row]

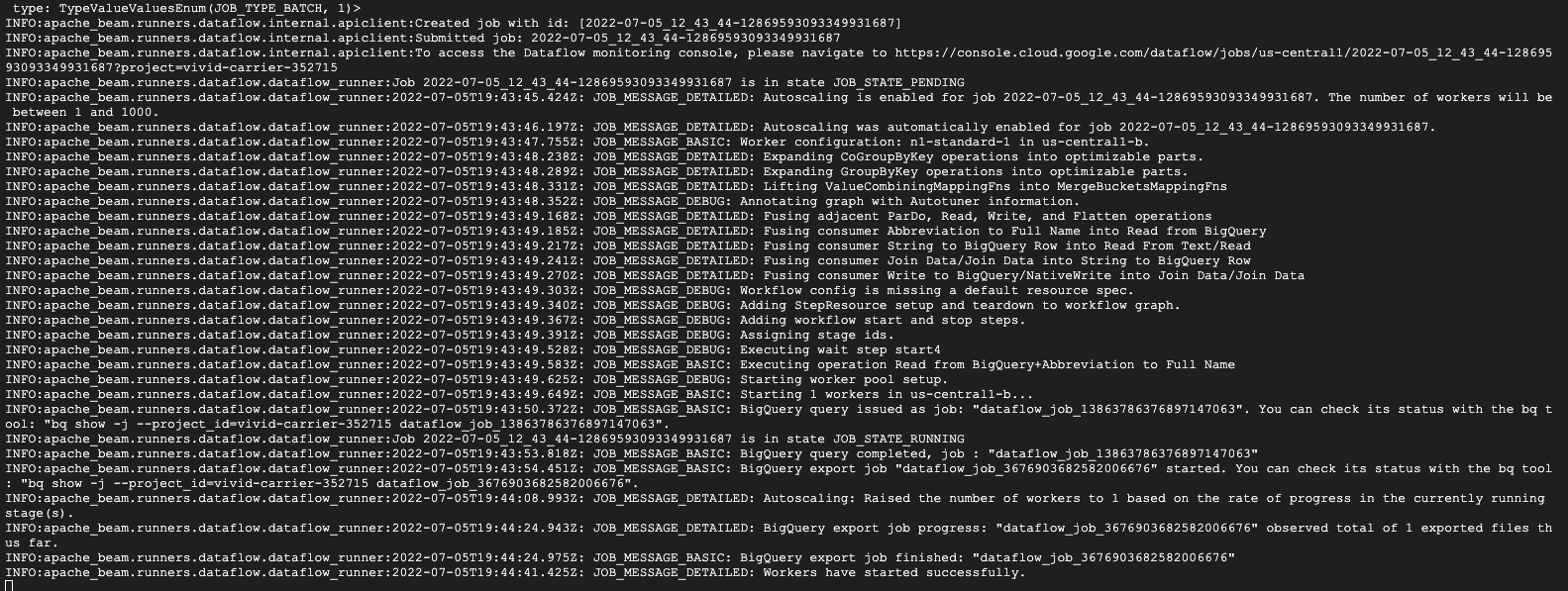


**Run the Apache Beam pipeline**

Here you'll run the Dataflow pipeline in the cloud. Run the following to spin up the workers required, and shut them down when complete:

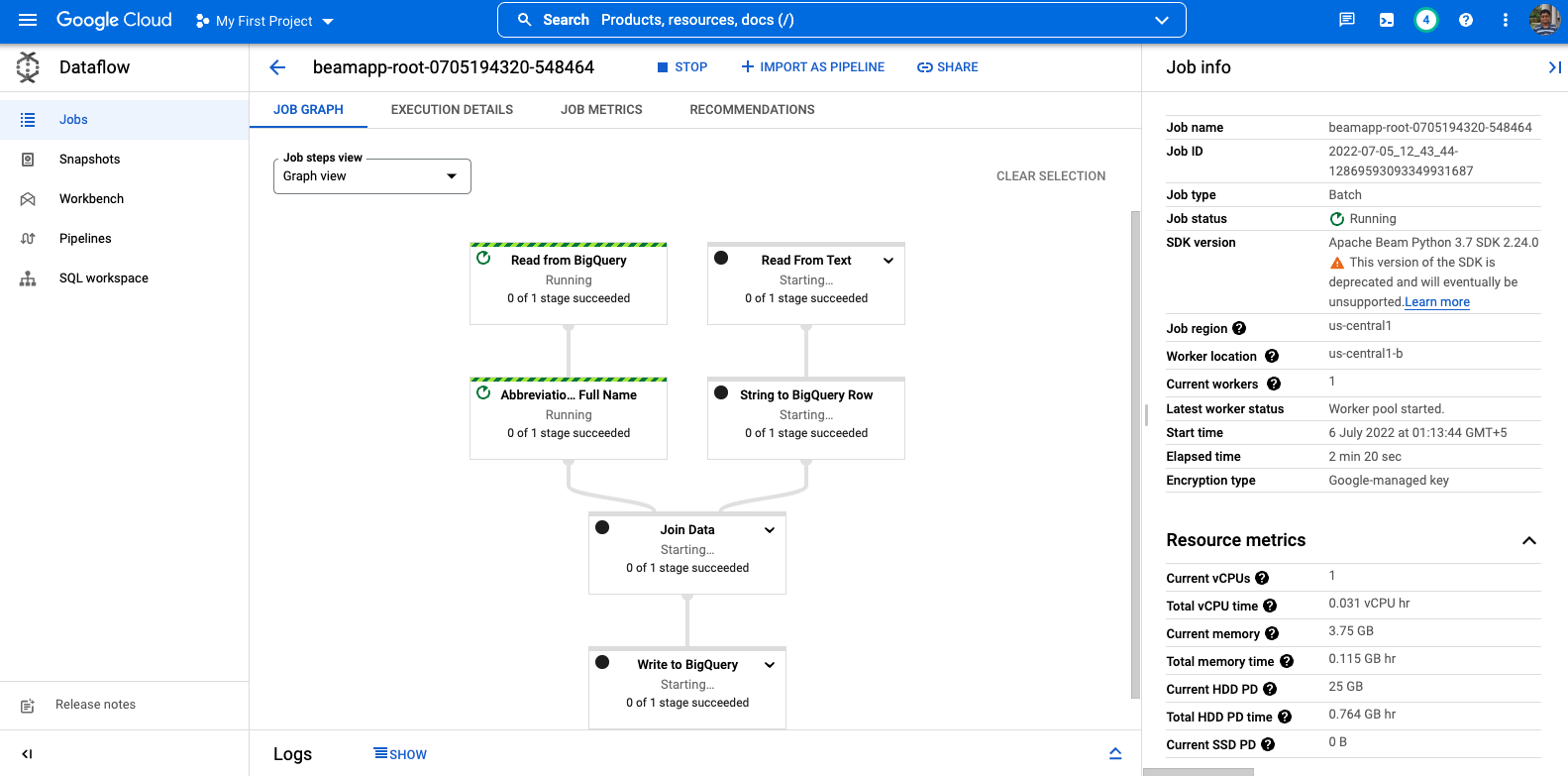
python dataflow\_python\_examples/data\_enrichment.py --project=$PROJECT --region=us-central1 --runner=DataflowRunner --staging\_location=gs://$PROJECT/test --temp\_location gs://$PROJECT/test --input gs://$PROJECT/data\_files/head\_usa\_names.csv --save\_main\_session

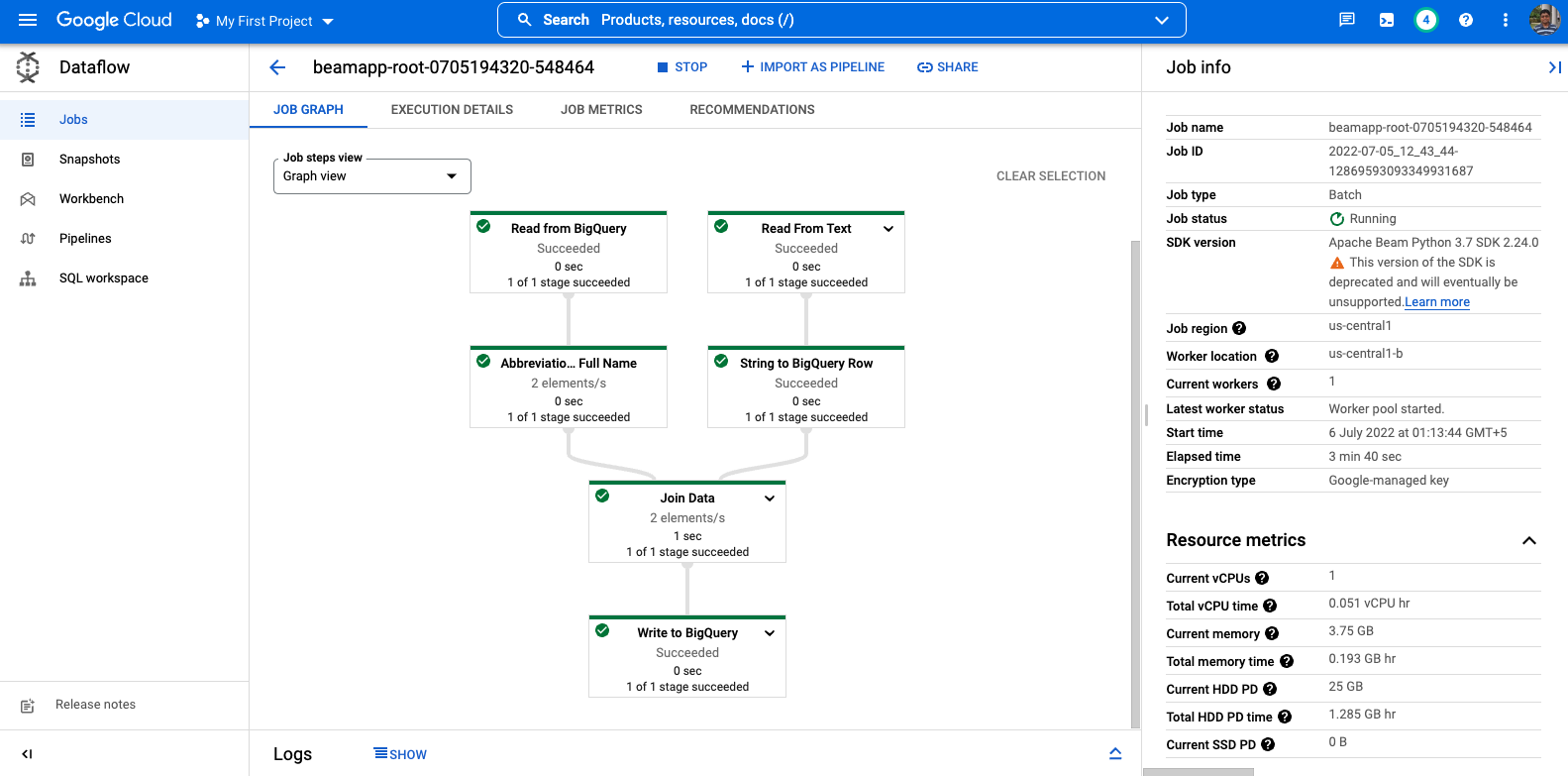




Navigate to **Navigation menu** > **Dataflow** to view the status of your job.



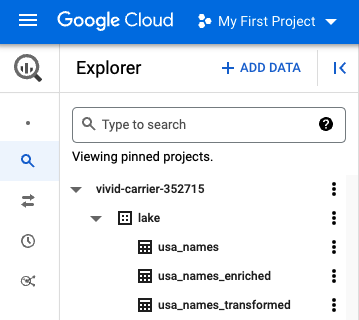




Once your **Job Status** is **Succeed** in the Dataflow Job status screen, navigate to BigQuery to check to see that your data has been populated.

You should see the **usa\_names\_enriched** table under the lake dataset.

Click on the table and navigate to the Preview tab to see examples of data for the data.



**Note:** If you don't see the usa\_names\_enriched table, try refreshing the page or view the tables using the classic BigQuery UI.

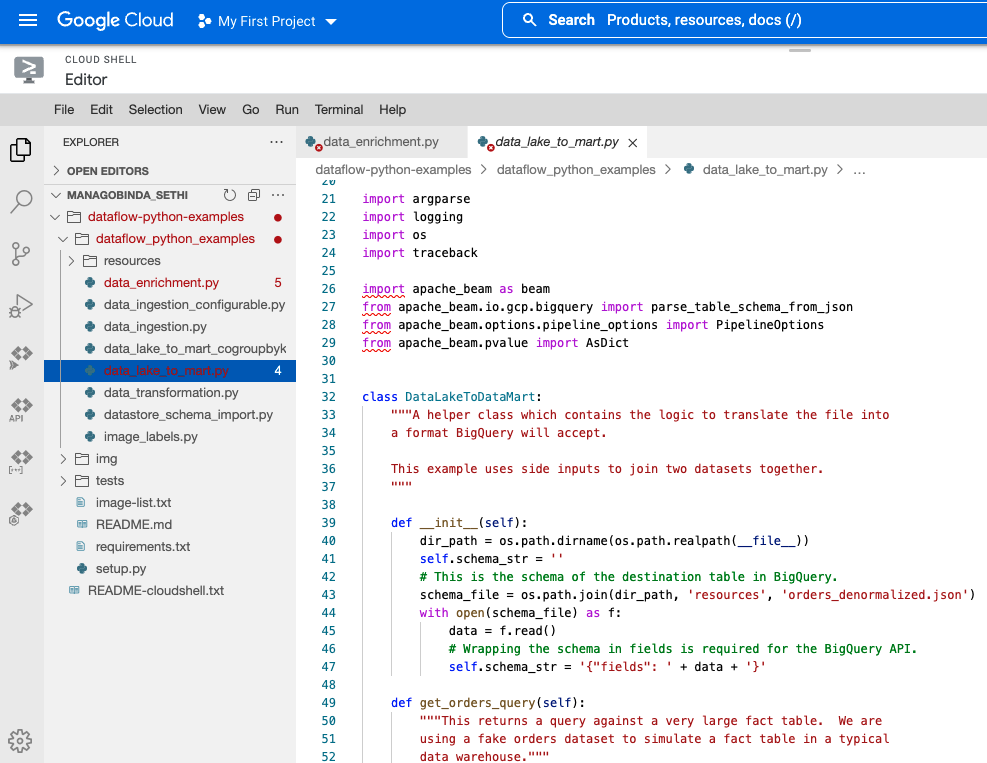
Step 5 - Data lake to Mart

You will now build a Dataflow pipeline with a TextIO source and a BigQueryIO destination to ingest data into BigQuery. More specifically, you will:

* Ingest the files from Cloud Storage.
* Filter out the header row in the files.
* Convert the lines read to dictionary objects.
* Output the rows to BigQuery.

**Review pipeline Python code**

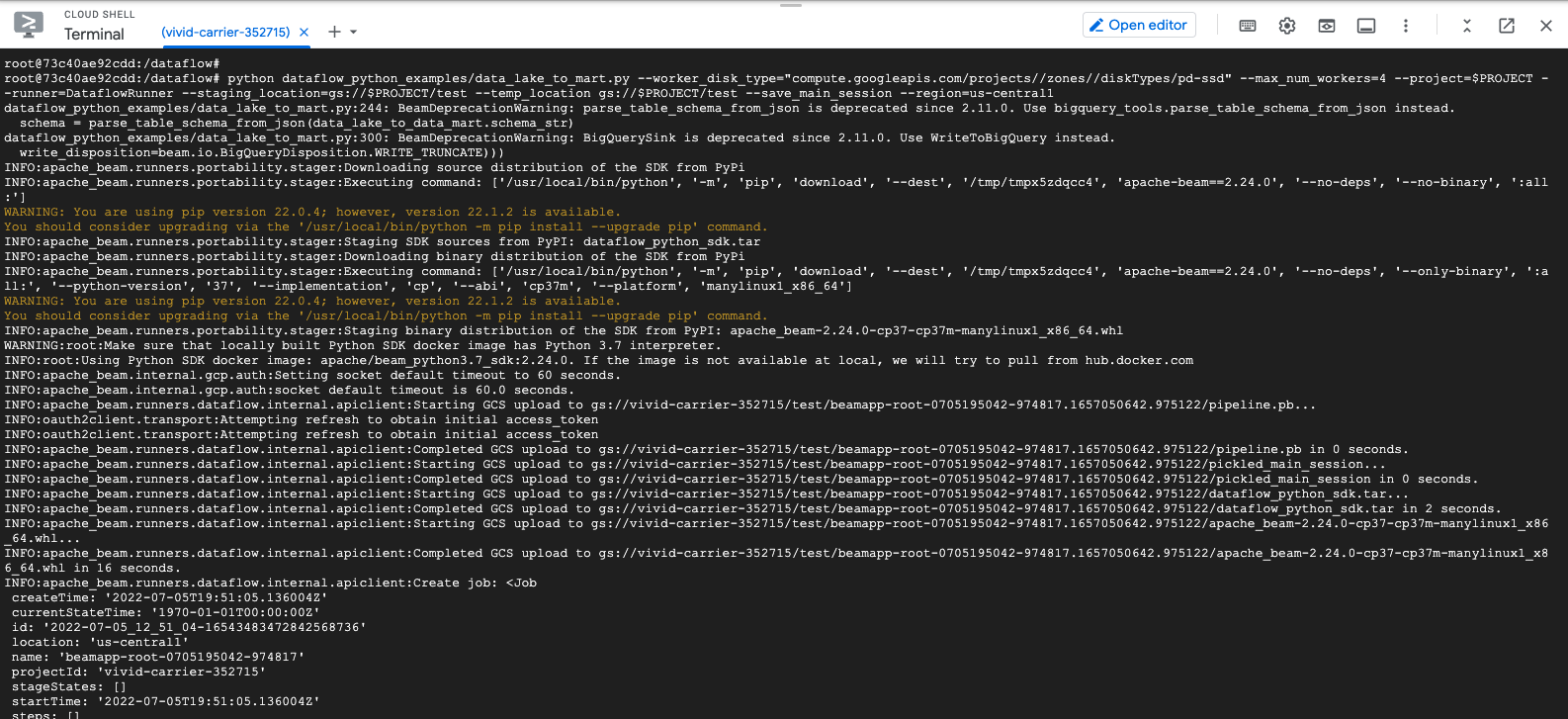
Navigate to data\_lake\_to\_mart.py and open it in the Cloud Shell code editor. Read through the comments in the file which explain what the code is doing. This code will populate the data in BigQuery.

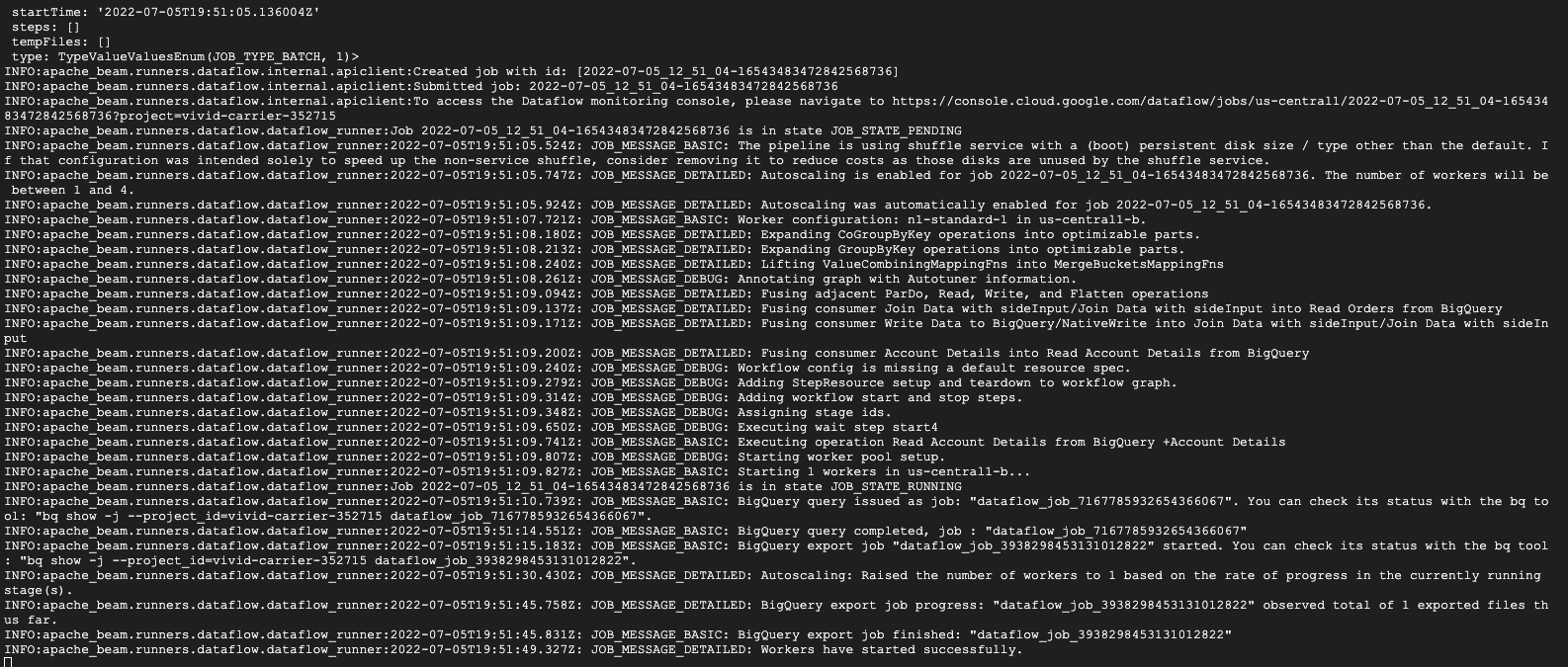


**Run the Apache Beam pipeline**

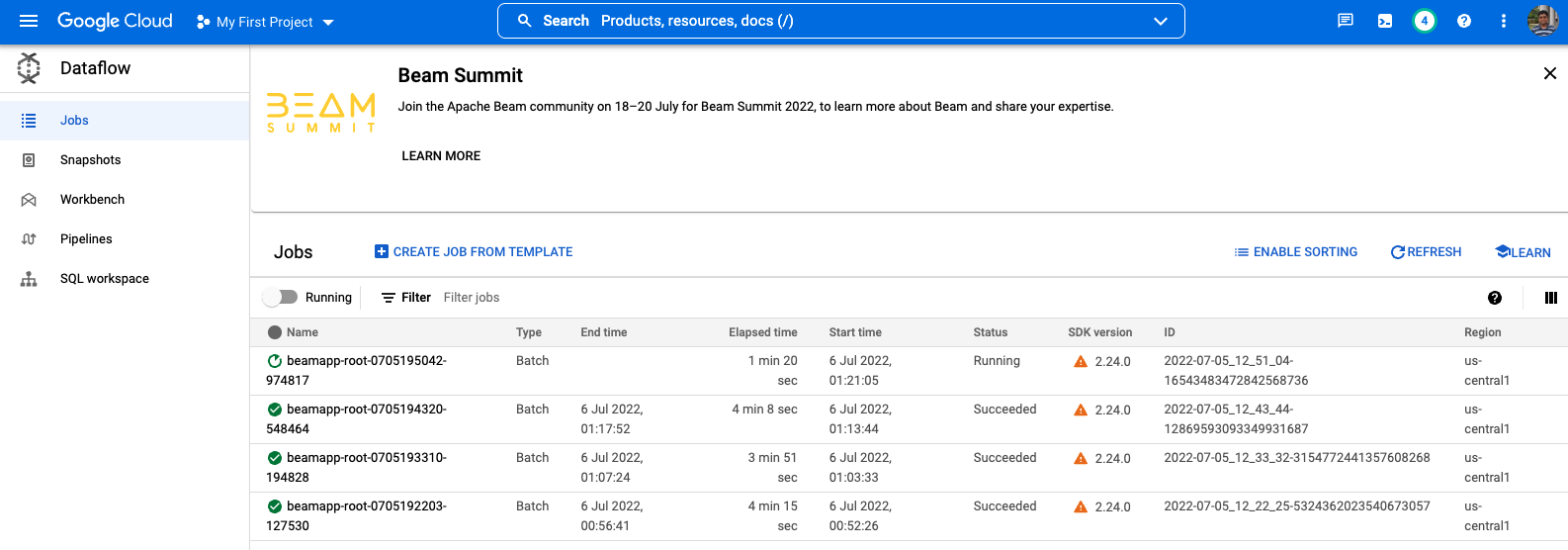
Now you'll run the Dataflow pipeline in the cloud. Run the following to spin up the workers required, and shut them down when complete:

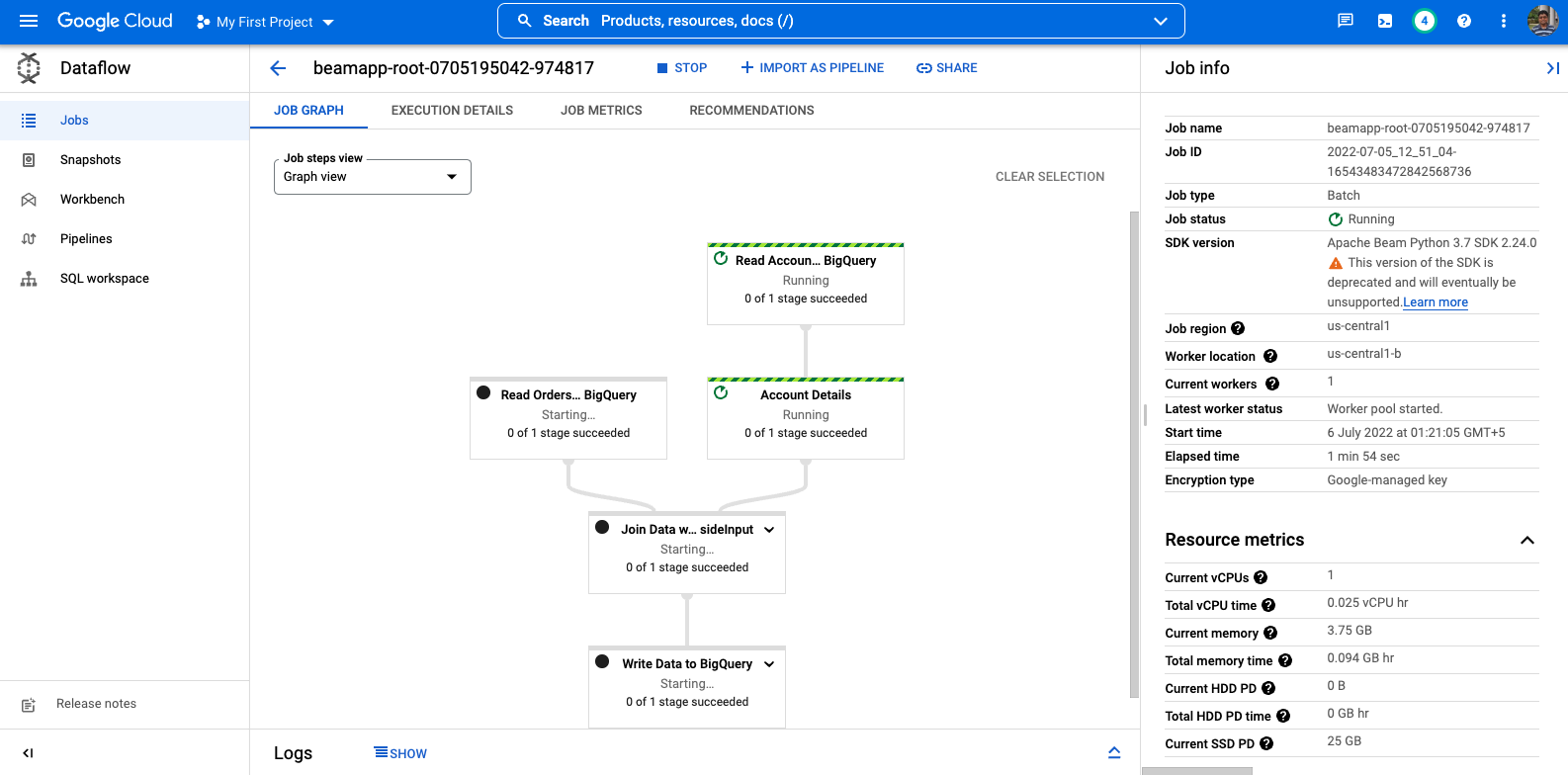
python dataflow\_python\_examples/data\_lake\_to\_mart.py --worker\_disk\_type="compute.googleapis.com/projects//zones//diskTypes/pd-ssd" --max\_num\_workers=4 --project=$PROJECT --runner=DataflowRunner --staging\_location=gs://$PROJECT/test --temp\_location gs://$PROJECT/test --save\_main\_session --region=us-central1

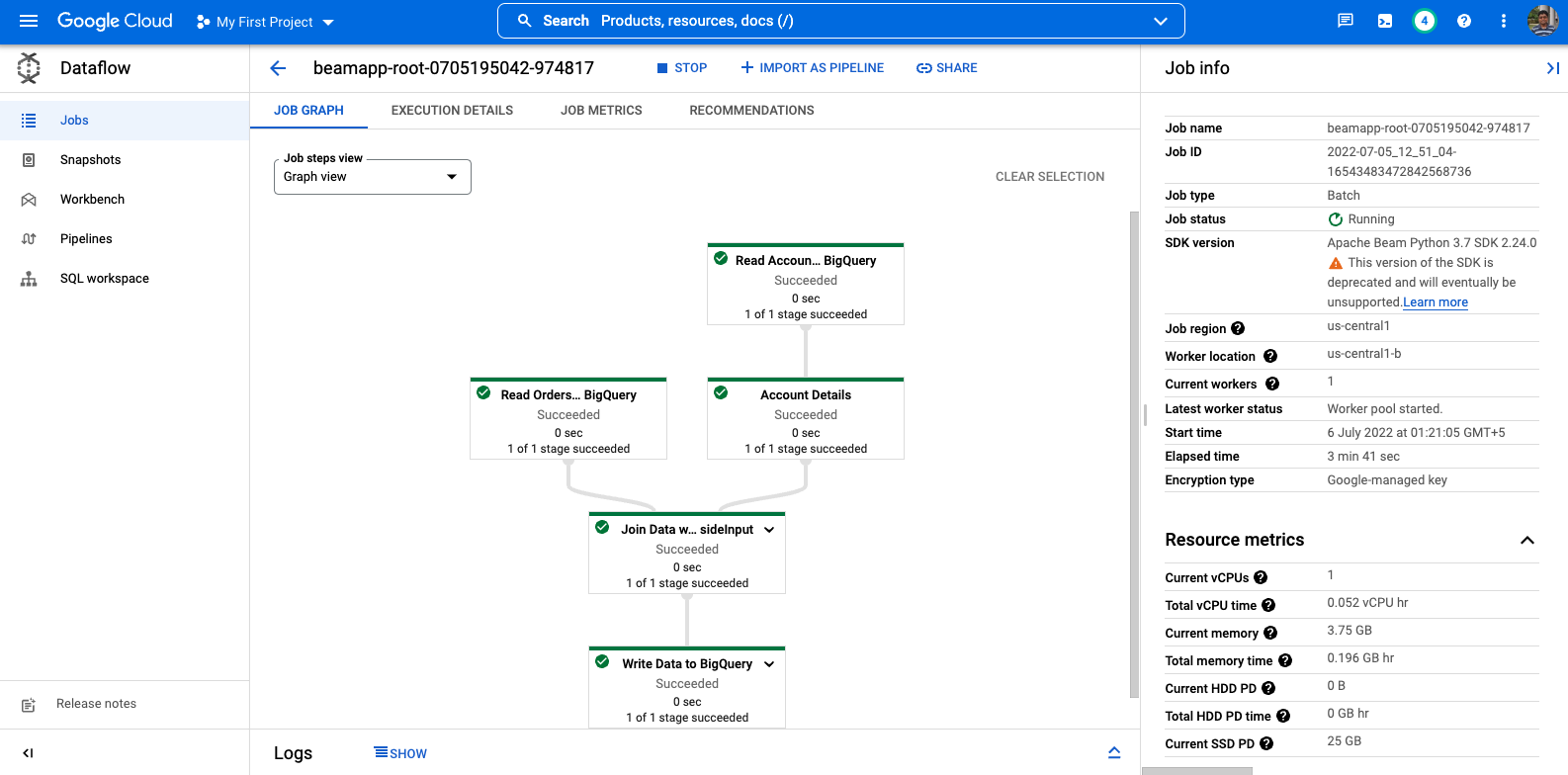




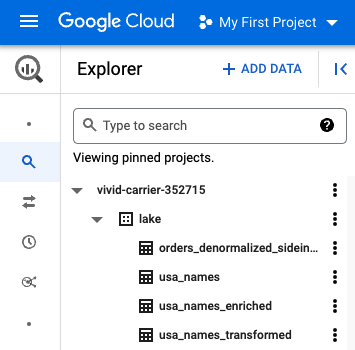
Navigate to **Navigation menu** > **Dataflow** and click on the name of this new job to view the status.



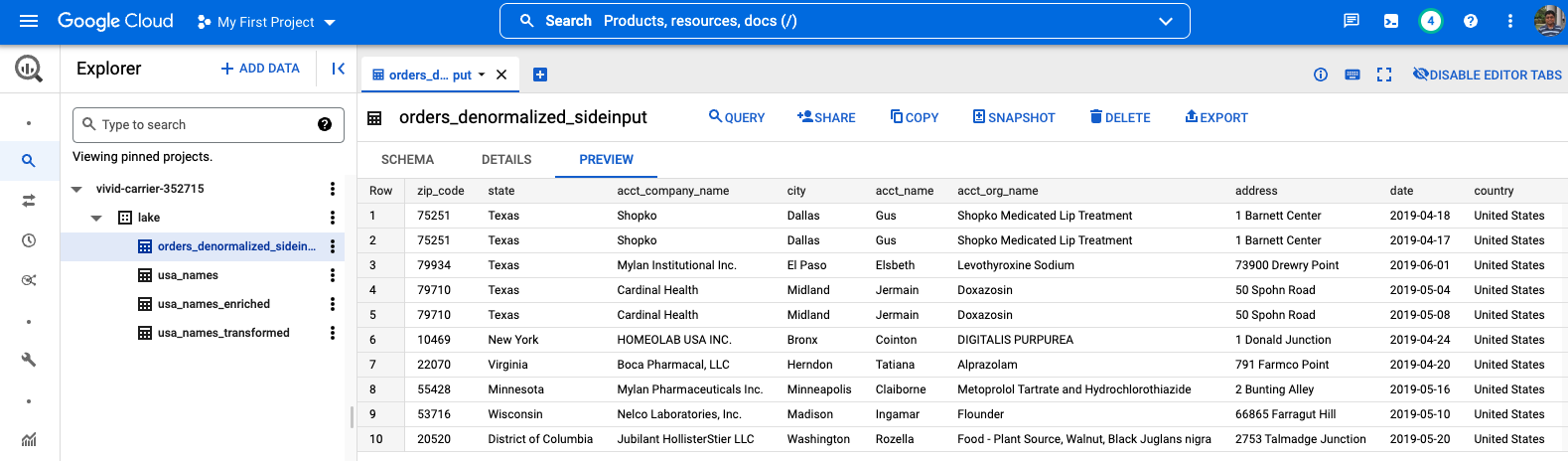




Once you've **Job Status** is **Succeeded** in the Dataflow Job status screen, navigate to BigQuery to check to see that your data has been populated.



You should see the **orders\_denormalized\_sideinput** table under the lake dataset.



Click on the table and navigate to the **Preview** section to see examples of orders\_denormalized\_sideinput data.

**Note:** If you don't see the orders\_denormalized\_sideinput table, try refreshing the page or view the tables using the classic BigQuery UI.