ETL Processing on Google Cloud Using Dataflow and BigQuery

Overview

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

- Cloud Storage
- Dataflow
- BigQuery

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.

Ensure that the Dataflow API is successfully enabled

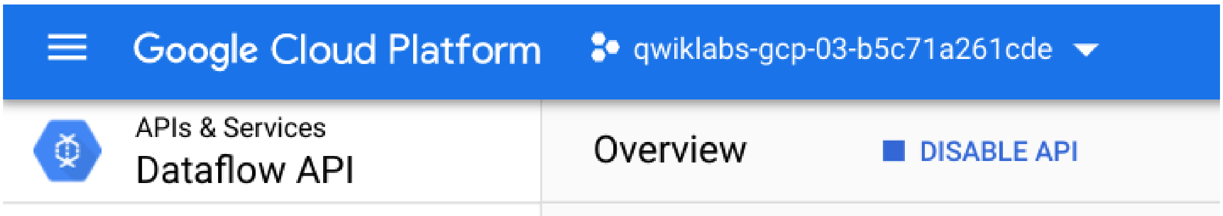
To ensure access to the necessary API, restart the connection to the Dataflow API.

1. In the Cloud Console, enter "Dataflow API" in the top search bar. Click on the result for **Dataflow API**.
2. Click **Manage**.
3. Click **Disable API**.

If asked to confirm, click **Disable**.

4. Click **Enable**.

When the API has been enabled again, the page will show the option to disable.



Download the starter code

Run the following command to get Dataflow Python Examples from Google Cloud's professional services GitHub:

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

Now set a variable equal to your project id, replacing <YOUR-PROJECT-ID> with your lab Project ID:

```
export PROJECT=<YOUR-PROJECT-ID>
gcloud config set project $PROJECT
```

Create 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 files into 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/
```

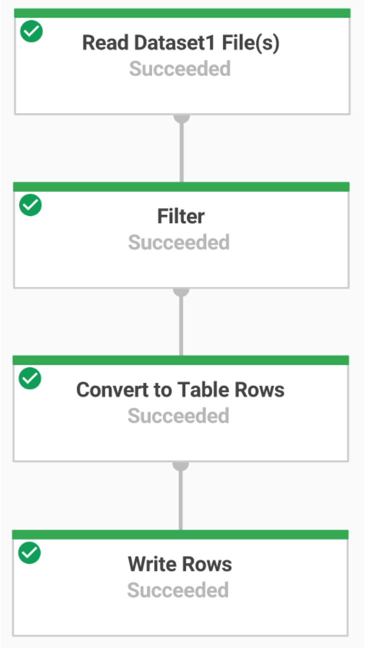
Create the BigQuery Dataset

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

```
bq mk lake
```

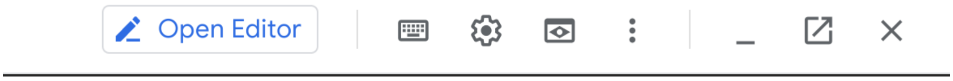
Build a Dataflow Pipeline

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 Google Cloud console.



Open Code Editor

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



If prompted click on **Open in a New Window**. It will open the code editor in new window.

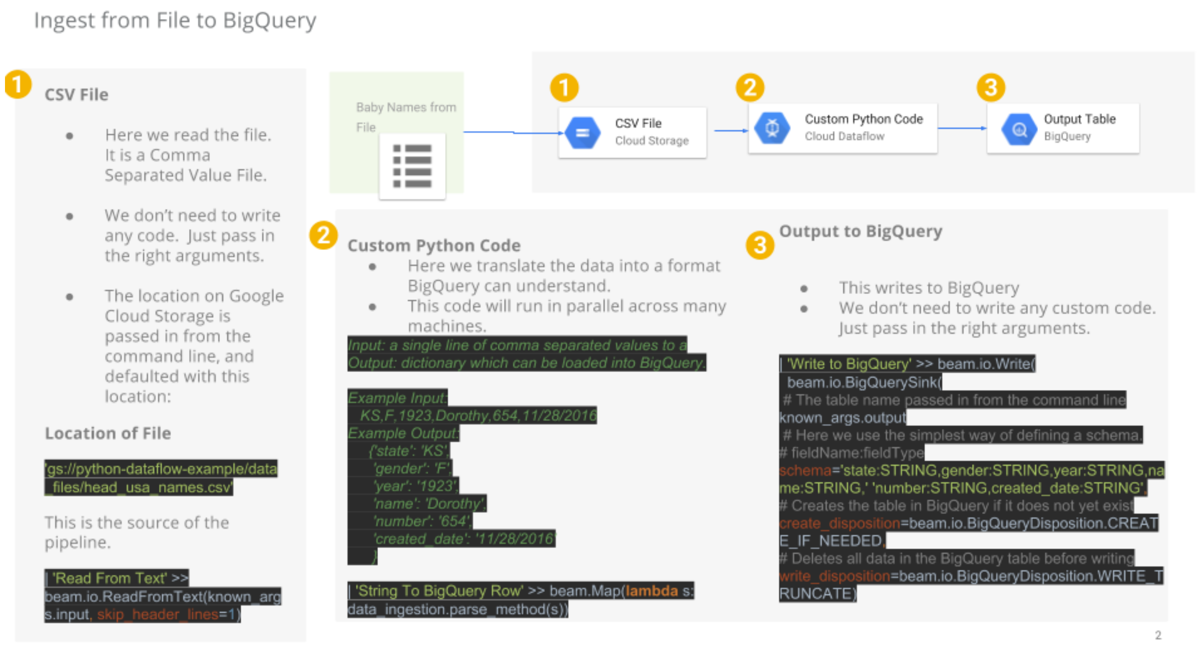
Data Ingestion

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 Code Editor navigate to `dataflow-python-examples` > `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.



Ingest from File to BigQuery

Baby Names from File

1 CSV File Cloud Storage → 2 Custom Python Code Cloud Dataflow → 3 Output Table BigQuery

1 **CSV File**

- Here we read the file. It is a Comma Separated Value File.
- We don't need to write any code. Just pass in the right arguments.
- The location on Google Cloud Storage is passed in from the command line, and defaulted with this location:

Location of File

```
'gs://python-dataflow-example/data
_files/head_usa_names.csv'
```

This is the source of the pipeline.

```
| 'Read From Text' >>
beam.io.ReadFromText(known_arg
s.input, skip_header_lines=1)
```

2 **Custom Python Code**

- Here we translate the data into a format BigQuery can understand.
- This code will run in parallel across many machines.

```
Input: a single line of comma separated values to a
Output: dictionary which can be loaded into BigQuery.

Example Input:
  KS,F,1923,Dorothy,654,11/28/2016
Example Output:
    {'state': 'KS',
     'gender': 'F',
     'year': '1923',
     'name': 'Dorothy',
     'number': '654',
     'created_date': '11/28/2016'
     }

| 'String To BigQuery Row' >> beam.Map(lambda s:
data_ingestion.parse_method(s))
```

3 **Output to BigQuery**

- This writes to BigQuery
- We don't need to write any custom code. Just pass in the right arguments.

```
| 'Write to BigQuery' >> beam.io.Write(
 beam.io.BigQuerySink(
 # The table name passed in from the command line
known_args.output,
 # Here we use the simplest way of defining a schema.
# fieldName:fieldType
schema='state:STRING,gender:STRING,year:STRING,na
me:STRING,' 'number:STRING,created_date:STRING',
# Creates the table in BigQuery if it does not yet exist
create_disposition=beam.io.BigQueryDisposition.CREAT
E_IF_NEEDED,
# Deletes all data in the BigQuery table before writing
write_disposition=beam.io.BigQueryDisposition.WRITE_T
RUNCATE)
```

2

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.

Run the following to set up the python environment:

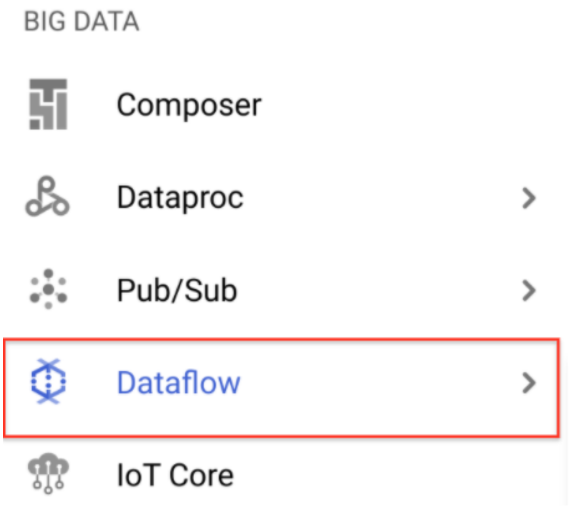
```
cd dataflow-python-examples/
# Here we set up the python environment.
# Pip is a tool, similar to maven in the java world
sudo pip install virtualenv
#Dataflow requires python 3.7
virtualenv -p python3 venv
source venv/bin/activate
pip install apache-beam[gcp]==2.24.0
```

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.

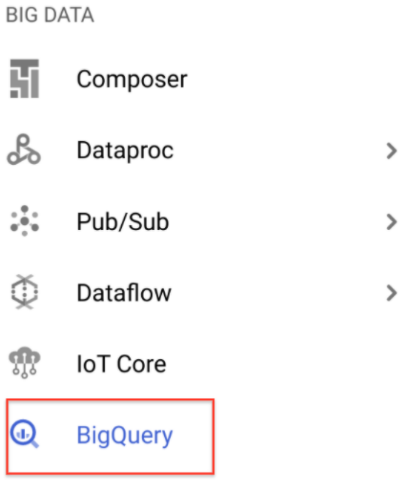
Data Transformation

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

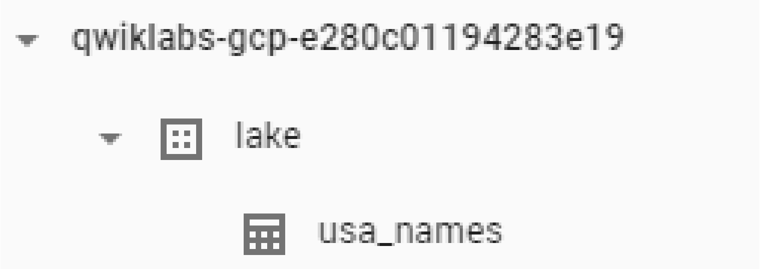
- Ingest the files from Cloud Storage.
- Convert the lines read to dictionary objects.
- Transform the data which contains the year to a format BigQuery understands as a date.
- Output the rows to BigQuery.

Review pipeline python code

In the Code Editor open `data_transformation.py` file. Read through the comments in the file which explain what the code is doing.



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.

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 to view the status of your job.

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.

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

In the Code Editor open `data_enrichment.py` file. 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 the `usa_names_enriched` 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.

Data lake to Mart

Now build a Dataflow pipeline that reads data from 2 BigQuery data sources, and then joins the data sources. Specifically, you:

- Ingest files from 2 BigQuery sources.
- Join the 2 data sources.
- 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 Code Editor open `data_lake_to_mart.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

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 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 **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.*