**Overview**

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

* **GCS** - Google Cloud Storage
* **Dataflow** - Google Dataflow
* **BigQuery** - 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.

### **Download the Starter Code**

Open a session in Cloud Shell and run the following command to check out the code from [GCP's professional services github](https://github.com/GoogleCloudPlatform/professional-services/blob/master/examples/dataflow-python-examples/README.md):

git clone https://github.com/GoogleCloudPlatform/professional-services.git

Now set a variable equal to your project id, replacing <YOUR-PROJECT-ID> with your Qwiklabs GCP 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

Use the gsutil command to copy files to the GCS bucket you just created:

gsutil cp gs://python-dataflow-example/data\_files/usa\_names.csv gs://$PROJECT/data\_files/

gsutil cp gs://python-dataflow-example/data\_files/head\_usa\_names.csv gs://$PROJECT/data\_files/

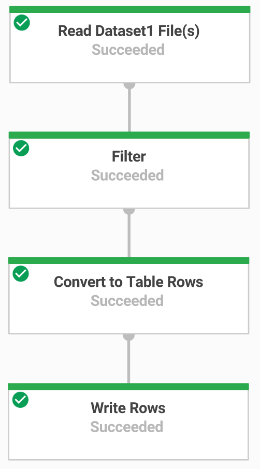
### **Create the BigQuery Dataset**

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 GCP console.



### **Step 1 - Open Code Editor**

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

aed20438abbffef3.png

**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 GCS.
* 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 professional-services > examples > 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.

#### **Run the Apache Beam Pipeline**

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

Run the following to set up the python environment:

cd professional-services/examples/dataflow-python-examples/

*# Here we set up the python environment.*

*# Pip is a tool, similar to maven in the java world*

sudo pip install --upgrade virtualenv

*#Dataflow requires python 2.7*

virtualenv -p `which python 2.7` dataflow-env

source dataflow-env/bin/activate

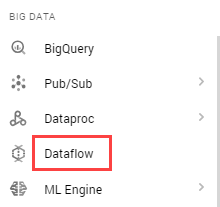
pip install apache-beam[gcp]

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=$PROJECT --runner=DataflowRunner --staging\_location=gs://$PROJECT/test --temp\_location gs://$PROJECT/test --input gs://$PROJECT/data\_files/head\_usa\_names.csv --save\_main\_session

Return to the GCP 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.

#### Test Completed Task

Click **Check my progress** to verify your performed task.

### **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 GCS.
* 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 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=$PROJECT --runner=DataflowRunner --staging\_location=gs://$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 GCS.
* 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 Code Editor. Check out the comments which explain what the code is doing. This code will populate the data in BigQuery.

#### **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=$PROJECT --runner=DataflowRunner --staging\_location=gs://$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 GCS.
* 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 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

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.

### **Step 6 - Data lake to Mart CoGroupByKey**

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 GCS.
* 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\_cogroupbykey.py and open it in Code Editor. Read through the comments which explain what the code is doing. Also review the [Apache Beam Programming Guide](https://beam.apache.org/documentation/programming-guide/)for more advanced concepts. This code will populate the data in BigQuery.

#### **Run the Apache Beam Pipeline**

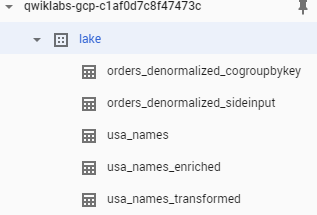
Here you'll run the Dataflow pipeline in the cloud. Run the following to spin up the workers required, and shut them down when complete:

python dataflow\_python\_examples/data\_lake\_to\_mart\_cogroupbykey.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

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\_cogroupbykey** table under the lake dataset.



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

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