

ETL Processing on Google Cloud Using Dataflow and BigQuery | Google Cloud Skills Boost

Qwiklabs : 15-19 minutes

GSP290



Google Cloud Self-Paced Labs

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.

Setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
 - The **Open Google Console** button
 - Time remaining
 - The temporary credentials that you must use for this lab
 - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.

Tip: Arrange the tabs in separate windows, side-by-side.

Note: If you see the **Choose an account** dialog, click **Use Another Account**.

3. If necessary, copy the **Username** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.
4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next**.

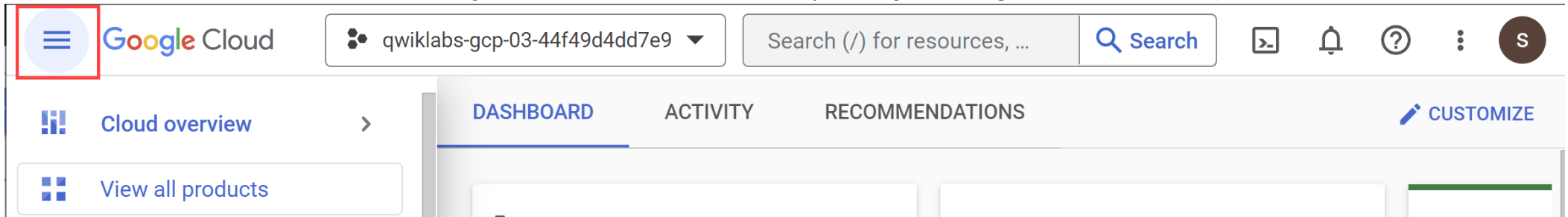
Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

5. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.


After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **PROJECT_ID**. The output contains a line that declares the **PROJECT_ID** for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

2. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3. Click **Authorize**.

4. Your output should now look like this:

Output:

```
ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net To set the active account, run: $ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core] project = <project_ID>
```

Example output:

```
[core] project = qwiklabs-gcp-44776a13dea667a6 Note: For full documentation of gcloud, in Google Cloud, refer to the gcloud CLI overview guide.
```

Task 1. 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.

Task 2. Download the starter code

1. Run the following command to get Dataflow Python Examples from [Google Cloud's professional services GitHub](#):

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

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

```
export PROJECT={{ project_0.project_id }} gcloud config set project $PROJECT
```

Task 3. Create Cloud Storage Bucket

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

```
gsutil mb -c regional -l {{{ project_0.default_region }}} gs://$PROJECT
```

Test completed task

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

Create a Cloud Storage Bucket

Task 4. Copy files to your bucket

- Use the `gsutil` command to copy files into the Cloud Storage bucket you just created:

```
gsutil cp gs://spl/gsp290/data_files/usa_names.csv gs://$PROJECT/data_files/ gsutil cp gs://spl/gsp290/data_files/head_usa_names.csv  
gs://$PROJECT/data_files/
```

Test completed task

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

Copy Files to Your Bucket

Task 5. 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
```

Test completed task

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

Create the BigQuery Dataset (name: lake)

Task 6. 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.



Read Dataset1 File(s)

Succeeded



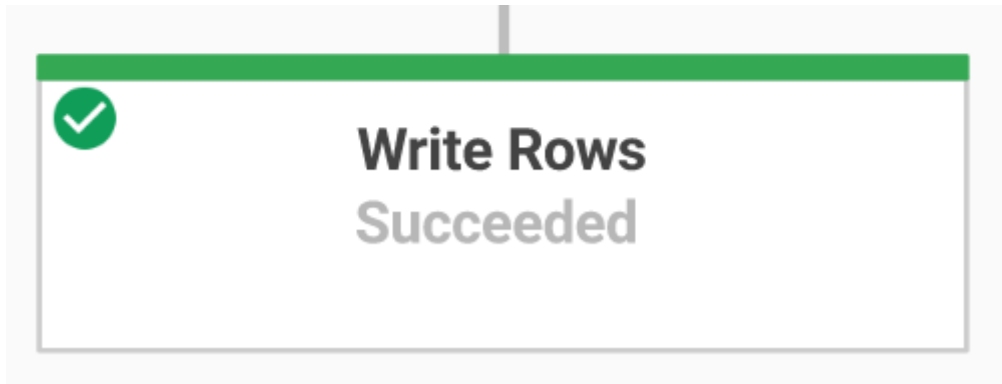
Filter

Succeeded



Convert to Table Rows

Succeeded



Open Code Editor

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



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

Task 7. 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.

Task 8. 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

1

CSV File

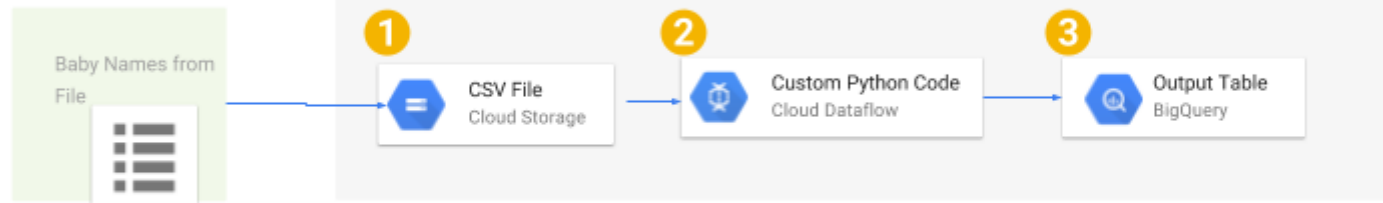
- Here we read the file. It is a Comma Separated Value File.
- We don't need to write any code. Just pass in the right arguments.
- The location on Google Cloud Storage is passed in from the command line, and defaulted with this location:

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_args.  
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

Task 9. Run the Apache Beam pipeline

1. 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.

2. 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.

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

```
pip install apache-beam[gcp]==2.24.0
```

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

```
cd dataflow/
```

You will run the Dataflow pipeline in the cloud.

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

```
python dataflow_python_examples/data_ingestion.py \ --project=$PROJECT --region={{ project_0.default_region }} \ --runner=DataflowRunner \ --staging_location=gs://$PROJECT/test \ --temp_location gs://$PROJECT/test \ --input gs://$PROJECT/data_files/head_usa_names.csv \ --save_main_session
```

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

ANALYTICS



Composer



Dataproc



Pub/Sub

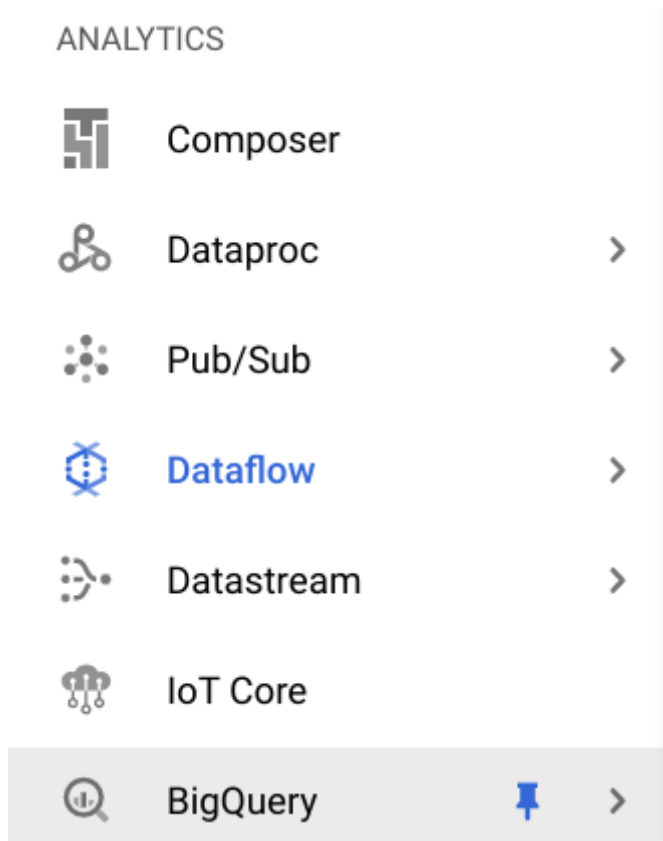


Dataflow



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

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



9. Click on your project name to see the **usa_names** table under the lake dataset.



10. 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.

Test completed task

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

Build a Data Ingestion Dataflow Pipeline

Task 10. 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.

Task 11. 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.

1. Run the following commands to do so:

```
python dataflow_python_examples/data_transformation.py \
--project=$PROJECT \
--region={{ project_0.default_region }} \
--runner=DataflowRunner \
--staging_location=gs://$PROJECT/test \
--temp_location gs://$PROJECT/test \
--input gs://$PROJECT/data_files/head_usa_names.csv \
--save_main_session
```

2. Navigate to **Navigation menu > Dataflow** and click on the name of this job to view the status of your job.
3. 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.
4. You should see the **usa_names_transformed** table under the `lake` dataset.

5. 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.

Test completed task

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

Build a Data Transformation Dataflow Pipeline

Task 12. 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.

Task 13. Review pipeline python code

1. In the Code Editor, open `data_enrichment.py` file.
2. 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]
```

3. Edit it so it looks like the following:

```
values = [x for x in csv_row]
```

Task 14. Run the Apache Beam pipeline

Here you'll run the Dataflow pipeline in the cloud.

1. Run the following to spin up the workers required, and shut them down when complete:

```
python dataflow_python_examples/data_enrichment.py \ --project=$PROJECT \ --region={{ project_0.default_region }} \ --runner=DataflowRunner \ --staging_location=gs://$PROJECT/test \ --temp_location gs://$PROJECT/test \ --input gs://$PROJECT/data_files/head_usa_names.csv \ --save_main_session
```

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

3. 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.

4. 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.

Test completed task

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

Build a Data Enrichment Dataflow Pipeline

Task 15. 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.

Task 16. 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.

Task 17. Run the Apache Beam Pipeline

Now you'll run the Dataflow pipeline in the cloud.

1. 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={{ project_0.default_region }}
```

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

3. 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.

4. 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.

Test completed task

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

Build a Data lake to Mart Dataflow Pipeline

Test your understanding

Below are multiple choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

Congratulations!

You have used python files to ingest data into BigQuery using Dataflow.

Finish your quest

This self-paced lab is part of the [Data Engineering](#) quest. A quest is a series of related labs that form a learning path. Completing this quest earns you the badge to recognize your achievement. You can make your badge public and link to it in your online resume or social media account. [Enroll in this quest](#) and get immediate completion credit if you've taken this lab. [See other available quests](#).

Take your next lab

Continue your quest with [Predict Visitor Purchases with a Classification Model in BQML](#), or check out these suggestions:

- [Predict Housing Prices with Tensorflow and AI Platform](#)
- [Cloud Composer: Copy BigQuery Tables Across Different Locations](#)

Next steps / learn more

Looking for more? Check out official documentation on:

- [Google Dataflow](#)
- [BigQuery](#)
- Also review the [Apache Beam Programming Guide](#) for more advanced concepts.

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated January 12, 2022

Lab Last Tested January 12, 2022

Copyright 2023 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.