Processing Data with Google Cloud Dataflow

cloudskillsboost.google/games/2854/labs/17213

GSP198

Overview

In this lab you simulate a real-time, real world dataset from a historical dataset. You use Python and Dataflow to process a simulated dataset from a set of text files and then use BigQuery to store and analyze the resulting data.

The historical dataset this lab uses is from the US Bureau of Transport Statistics website, which provides historic information about internal flights in the United States.

Dataflow is a fully-managed service for transforming and enriching data in stream (real time) and batch (historical) modes via Java and Python APIs with the Apache Beam SDK. Dataflow provides a serverless architecture that can be used to shard and process very large batch datasets, or high volume live streams of data, in parallel.

BigQuery is a RESTful web service that enables interactive analysis of massive datasets working in conjunction with Cloud Storage.

Objectives

- Configure a Python application to create a simulated real-time data stream from historical data.
- Use Apache Beam locally to test Dataflow locally.
- Use Apache Beam to process data using Dataflow to create a simulated real-time dataset.
- Query the simulated real-time data stream using BiqQuery.

Setup and requirements

Qwiklabs 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.

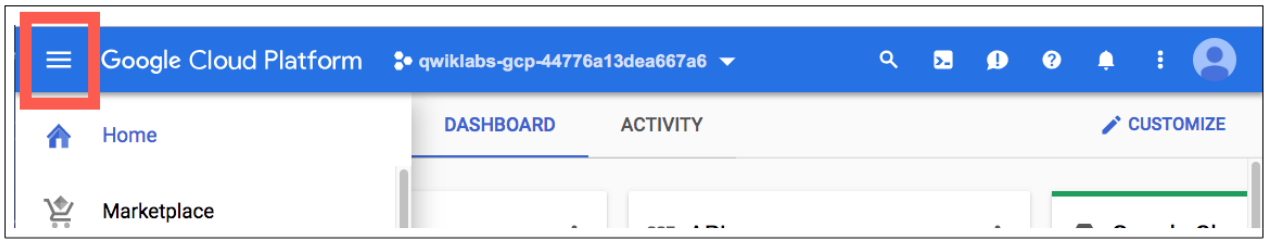
    Tip: Arrange the tabs in separate windows, side-by-side.

    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.



Task 1. Prepare your environment

This lab uses a set of code samples and scripts developed for the *Data Science on the Google Cloud Platform, 2nd Edition* book from O'Reilly Media, Inc. You clone the sample repository (used in Chapter 4) from Github to a VM and then carry out all lab tasks.

Clone the Data Science sample

1. In the Cloud Console, on the **Navigation menu** (≡), click **Compute Engine** > **VM instances**.

2. Click the **SSH** button next to `lab-vm-ql` VM to launch a terminal and connect.

3. In the terminal, enter the following command to clone the repository:

```
git clone https://github.com/GoogleCloudPlatform/data-science-on-gcp/
```

4. Change to the repository source directory for this lab:

```
cd ~/data-science-on-gcp/04_streaming/transform
```

5. Install required packages:

```
./install_packages.sh
```

Note: When you run these commands, ignore errors related to Google utilities and incompatible packages.

Test the Python data processing functions

1. In the `lab-vm-ql` terminal, use nano to examine the file `df05.py` :

```
nano df05.py
```

This file provides a number of functions to process the historical data file to generate corresponding departure and arrival event records, including accurate universal time-stamps, not just the time stamps provided in the source dataset. For more information on how these processing functions work, see Chapter 4 of *Data Science on the Google Cloud Platform, 2nd Edition*.

2. Press **CTRL+X** to exit the nano editor.

3. Execute the `df05.py` script. This script calls Apache Beam using the `Directrunner` parameter. This parameter executes at the VM, rather than using a configuration that deploys at scale to the cloud. This allows you to test and evaluate the accuracy of the code before committing significant resources to a processing task.

```
def run():
   with beam.Pipeline('DirectRunner') as pipeline:
```

4. Enter following command to run the script:

./df05.py
This may take up to 10 minutes to complete.

When it completes, you see two events for each flight. Check the events file by running:

head -3 all_events-00000*
Note: There are now departure and arrival events interleaved. The departure events do not contain any arrival information, and look like this:

{"FL_DATE": "2015-04-28", "UNIQUE_CARRIER": "EV", "ORIGIN_AIRPORT_SEQ_ID": "1013503", "ORIGIN": "ABE", "DEST_AIRPORT_SEQ_ID": "1039705", "DEST": "ATL", "CRS_DEP_TIME": "2015-04-28 20:00:00", "DEP_TIME": "2015-04-28 19:55:00", "DEP_DELAY": -5, "CRS_ARR_TIME": "2015-04-28 22:09:00", "CANCELLED": false, "DIVERTED": false, "DEP_AIRPORT_LAT": 40.65222222, "DEP_AIRPORT_LON": -75.44055556,"DEP_AIRPORT_TZOFFSET": -14400.0, "ARR_AIRPORT_LAT": 33.63666667, "ARR_AIRPORT_LON": -84.42777778, "ARR_AIRPORT_TZOFFSET": -14400.0, "EVENT_TYPE": "departed", "EVENT_TIME": "2015-04-28 19:55:00"}
{"FL_DATE": "2015-04-28", "UNIQUE_CARRIER": "EV", "ORIGIN_AIRPORT_SEQ_ID": "1013503", "ORIGIN": "ABE", "DEST_AIRPORT_SEQ_ID": "1039705", "DEST": "ATL", "CRS_DEP_TIME": "2015-04-28 20:00:00", "DEP_TIME": "2015-04-28 19:55:00", "DEP_DELAY": -5, "TAXI_OUT": 7, "WHEELS_OFF": "2015-04-28 20:02:00", "WHEELS_ON": "2015-04-28 21:47:00", "TAXI_IN": 4, "CRS_ARR_TIME": "2015-04-28 22:09:00", "ARR_TIME": "2015-04-28 21:51:00", "ARR_DELAY": -18, "CANCELLED": false, "DIVERTED": false, "DISTANCE": "692.00", "DEP_AIRPORT_LAT": 40.65222222, "DEP_AIRPORT_LON": -75.44055556, "DEP_AIRPORT_TZOFFSET": -14400.0, "ARR_AIRPORT_LAT": 33.63666667, "ARR_AIRPORT_LON": -84.42777778, "ARR_AIRPORT_TZOFFSET": -14400.0, "EVENT_TYPE": "arrived", "EVENT_TIME": "2015-04-28 21:51:00"}
The first event is a `departed` event and is published at the departure time. The second is an `arrived` event and is published at the arrival time. The departed event has a number of missing fields corresponding to data that is not known at that time.

This task may take up to 10 minutes to process a small sample of a thousand or so events. The code that identifies the timezones is very compute heavy as it has to check against a large number of intersecting polygons to locate each airport.

Task 2. Read and write to the Cloud

Configure BigQuery and Dataflow for your project.

So far, you've been reading and writing files from one location. Once you start to run your code in production, in a serverless environment, the concept of a “location” no longer makes sense. You have to read and write from Cloud Storage. Also, because this is structured data, it is preferable to read and write to BigQuery.

Now copy the airport geolocation file to your Cloud Storage bucket. This file identifies the physical location of each airport in order to convert the local time fields to universal time.

To save you some time:

- The Cloud Storage bucket was pre-created for you.
- The historical flights text data files are already in the bucket.

1. Run the following command to make sure you are in the working directory:

```
cd ~/data-science-on-gcp/04_streaming/transform
```

2. Run these commands:

```
export PROJECT_ID=$(gcloud info --format='value(config.project)') export
BUCKET=${PROJECT_ID}-ml ./stage_airports_file.sh $BUCKET ./df06.py --project
$PROJECT_ID --bucket $BUCKET
```

This may take up to 10 minutes to complete.

Test completed task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Copy the airport geolocation file to your Cloud Storage bucket. **Note:** The Dataflow API is enabled by default for Qwiklabs lab accounts. For your own projects you will have to enable the Dataflow API explicitly by going to **APIs and Services** > **Library** and searching for **Dataflow**.

Task 3. Process the data using Dataflow

Now deploy the functions using the full python script, `df07.py`, that is configured to run the tasks using Dataflow. The script calls Apache Beam using the `DataflowRunner` to execute the processing tasks in parallel using Dataflow. This substantially speeds up the processing of the data.

1. Use nano to reinspect the source code:

```
nano df07.py
```

Notice the `run` function has been changed to use Dataflow. It also points to the pre-configured Cloud Storage bucket that contains the historical flight data text files.

2. Press **CTRL+X** to exit the nano editor.

3. Execute the script:

./df07.py -p $PROJECT_ID -b $BUCKET -r us-central1 **Note:** You can ignore warnings related to Apache Beam for Python3 and missing metadata.
This script completes in a few seconds. The complete set of tasks has been handed off to Dataflow and will take about 30 minutes to complete. While you wait, continue to the next section. The initial startup phase for Dataflow typically takes about 7 or 8 minutes before it starts producing the simulated event data.

The lab has been configured to provide 2 months of source data, covering just under 900,000 flights that took place between Jan 1 2015 and Feb 28 2015. The resulting simulated dataset will have 2.6 million records as each non-cancelled flight that departs and arrives in that time window has three events generated: Departure, Wheels Off, and Arrived.

Dataflow autoscales the number of workers based on throughput – the more lines in the input data files, the more workers needed. To constrain the number of workers used, specify the maximum number of workers (`max_num_workers` parameter).

Test completed task

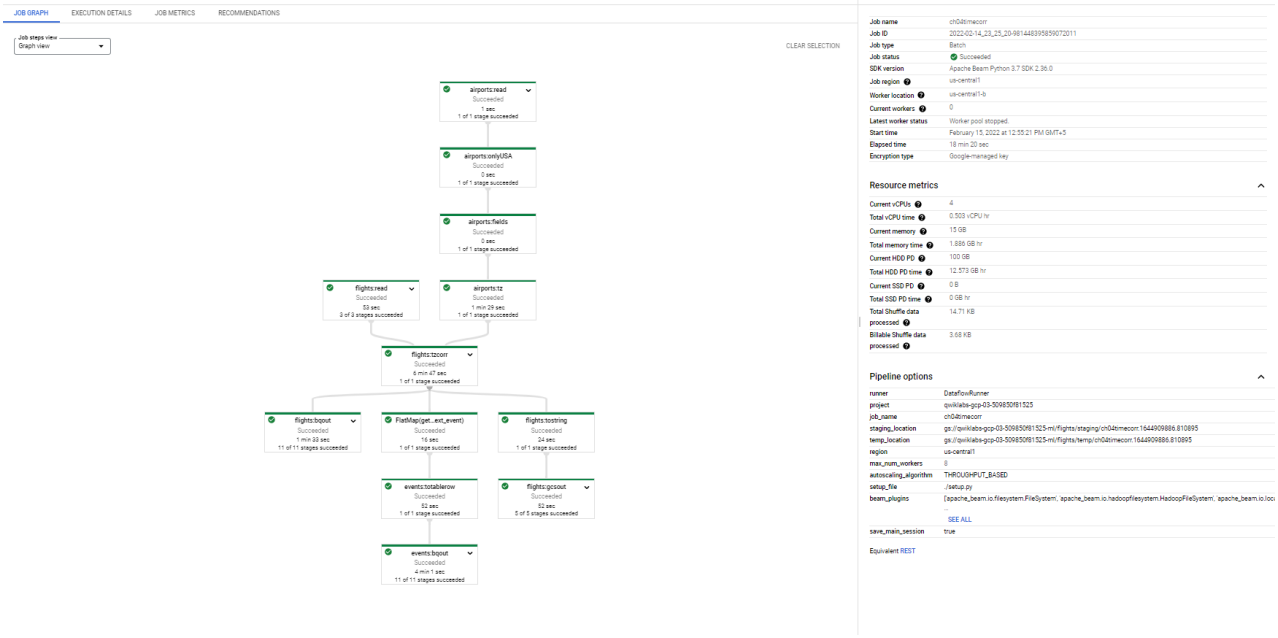
Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Process the Data using Dataflow (submit Dataflow job). If this step fails to score, wait a minute and resubmit the job by rerunning the previous command.

Task 4. Monitor the Dataflow job and inspect the processed data

1. In the Google Cloud Console, click **Navigation menu**, and in the Analytics section click on **Dataflow**.

2. Click the name of the Dataflow job to open the job details page for the events simulation job. This lets you monitor the progress of your job.



3. In the Cloud Console, on the **Navigation menu** (☰), click **BigQuery**. If prompted, click **Done**.

4. Copy the following script and paste it in the **Query editor** field:

```
SELECT ORIGIN, DEP_TIME, DEST, ARR_TIME, ARR_DELAY, EVENT_TIME, EVENT_TYPE FROM dsongcp.flights_simevents WHERE (DEP_DELAY > 15 and ORIGIN = 'SEA') or (ARR_DELAY > 15 and DEST = 'SEA') ORDER BY EVENT_TIME ASC LIMIT 5
```

5. Wait for it to validate then click **RUN**.

Test completed task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Compose Queries.
If you see less than 5 rowas of data, the Dataflow job is still running. Wait a few minutes and try again.

When the Dataflow job completes, you see 5 rows of data. It's now easier to see some of the differences between arrival and departure events. You will also see that there is now a third type of event, `wheelsoff`, that describes the moment when the plane actually leaves the ground.

BigQuery is a columnar database which makes it inefficient when you need to query all of the fields associated with a specific event. For the event simulation task that, you need to be able to retrieve all of the data fields for each event as efficiently as possible. The processing script addresses this problem by adding a field to the table that includes all of the original event record in text format. This trade-off between storage and speed allows higher performance querying at the expense of having a larger database table.

5. Click **Compose New Query** in the top left of the BigQuery console, and then click **Compose Query** to confirm.

6. Copy this new query and paste it into the **Query editor**:

```
SELECT EVENT_TYPE, EVENT_TIME, EVENT_DATA FROM
`dsongcp.flights_simevents` WHERE EVENT_TIME >= TIMESTAMP('2015-01-01
00:00:00 UTC') AND EVENT_TIME < TIMESTAMP('2015-01-03 00:00:00 UTC')
ORDER BY EVENT_TIME ASC LIMIT 2
```

Wait for it to validate then click **RUN**.

You can see that the table now includes a field that contains all of the event data.

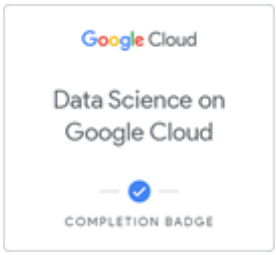
Test your understanding

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

Congratulations!

You used Dataflow to process historical batch data using Python and Apache Beam. You also used BigQuery to analyse a database that contains simulated real-time event data.

Finish your Quest



This self-paced lab is part of the Quest, Data Science on Google Cloud. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badges public and link to them 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 Visualize Real Time Geospatial Data with Google DataStudio, or check out these suggestions:

- Interactive Data Exploration with Vertex AI Workbench
- Evaluating a Data Model

Next steps / learn More

The source of this lab:

> Data Science on the Google Cloud Platform, 2nd Edition: O'Reilly Media, Inc.

Google Cloud Training & 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 June 6, 2022

Lab Last Tested June 6, 2022