Serverless Data Analysis with Dataflow - Lab 3 : Side Inputs (Python) v1.3

## Overview

In this lab, you learn how to load data into BigQuery and run complex queries. Next, you will execute a Dataflow pipeline that can carry out Map and Reduce operations, use side inputs and stream into BigQuery.

## Objective

In this lab, you learn how to use BigQuery as a data source into Dataflow, and how to use the results of a pipeline as a side input to another pipeline.

* Read data from BigQuery into Dataflow
* Use the output of a pipeline as a side-input to another pipeline

## Task 1. Preparation

For this lab, you will need the training-data-analyst files.

### **Verify that the repository files are in Cloud Shell**

1. Clone the repository from the Cloud Shell command line:

git clone https://github.com/GoogleCloudPlatform/training-data-analyst

1. Click on **File > Refresh** in the left navigator panel. You should see the **training-data-analyst** directory.

### **Verify that you have a Cloud Storage bucket**

If you don't have a bucket, you can follow these instructions to create a bucket.

1. In the Console, on the **Navigation menu** (7a91d354499ac9f1.png), click **Home**.
2. **Select and copy** the Project ID. For simplicity, you will use the Qwiklabs Project ID, which is already globally unique, as the bucket name.
3. In the Console, on the **Navigation menu** (7a91d354499ac9f1.png), click **Storage** > **Browser**.
4. Click **Create Bucket**.
5. Specify the following, and leave the remaining settings as their defaults:

|  |  |
| --- | --- |
| **Property** | **Value (type value or select option as specified)** |
| **Name** | <your unique bucket name (Project ID)> |
| **Default storage class** | Multi-Regional |
| **Location** | <Your location> |

1. Click **Create**.
2. Record the name of your bucket. You will need it in subsequent tasks.
3. In Cloud Shell enter the following to create an environment variable named "BUCKET" and verify that it exists with the echo command.

BUCKET="<your unique bucket name (Project ID)>"

echo $BUCKET

You can use $BUCKET in Cloud Shell commands. And if you need to enter the bucket name <your-bucket> in a text field in Console, you can quickly retrieve the name with "echo $BUCKET".

### **Verify environment variable for your Project ID**

1. Cloud Shell creates a default environment variable that contains the current Project ID.

echo $DEVSHELL\_PROJECT\_ID

### **Verify that Google Cloud Dataflow API is enabled for this project**

1. Return to the browser tab for Console. In the top search bar, enter **Google Dataflow API**. This will take you to the page, **Navigation menu > APIs & Services > Dashboard > Dataflow API**. It will either show a status information or it will give you the option to **Enable** the API.
2. If necessary, **Enable** the API.

### **Verify that Apache Beam is installed on Cloud Shell**

1. Return to Cloud Shell. Verify that Apache Beam is installed on Cloud Shell. If the Cloud Shell has timed out and was reconnected, it may have lost the in-memory components of Apache Beam. There is no harm in reinstalling. It will take the necessary steps.

cd ~/training-data-analyst/courses/data\_analysis/lab2/python

sudo ./install\_packages.sh

## Task 2. Try using BigQuery query

1. Return to the BigQuery web UI. If it is not already open, open [Console](http://console.cloud.google.com/). On the **Navigation menu** (7a91d354499ac9f1.png), click **BigQuery**.
2. Click **Compose Query** and type the following query.

SELECT

content

FROM

`fh-bigquery.github\_extracts.contents\_java\_2016`

LIMIT

10

1. Click on **Run**.

What is being returned?

The BigQuery table fh-bigquery.github\_extracts.contents\_java\_2016 contains the content (and some metadata) of all the Java files present in GitHub in 2016.

To find out how many Java files this table has, type the following query and click **Run**:

SELECT

COUNT(\*)

FROM

`fh-bigquery.github\_extracts.contents\_java\_2016`

Why do you think zero bytes of data were processed to return the result?



There were 0 records returned in the result.



BigQuery stores common metadata about the table (like row count). Querying metadata processes 0 bytes.



This dataset is not properly setup for billing.



Cache is enabled so all queries process 0 bytes.

Submit

How many files are there in this dataset?

Is this a dataset you want to process locally or on the cloud?

## Task 3. Explore the pipeline code

1. In Cloud Shell editor, or in Cloud Shell, navigate to the lab directory:

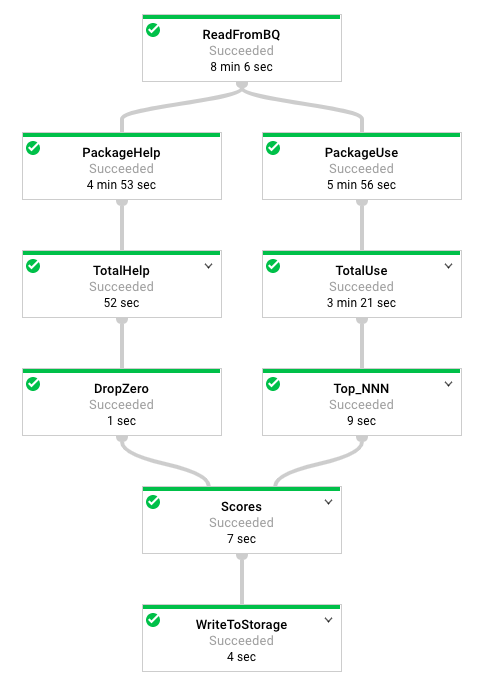
cd ~/training-data-analyst/courses/data\_analysis/lab2/python

1. View the pipeline code using Cloud Shell editor or nano. **Do not make any changes to the code.**

cd ~/training-data-analyst/courses/data\_analysis/lab2/python

nano JavaProjectsThatNeedHelp.py

Refer to this diagram as you read the code. The pipeline looks like this:



1. Answer the following questions:

* Looking at the class documentation at the very top, what is the purpose of this pipeline?
* Where does the content come from?
* What does the left side of the pipeline do?
* What does the right side of the pipeline do?
* What does ToLines do? (Hint: look at the content field of the BigQuery result)
* Why is the result of ReadFromBQ stored in a named PCollection instead of being directly passed to another step?
* What are the two actions carried out on the PCollection generated from ReadFromBQ?
* If a file has 3 FIXMEs and 2 TODOs in its content (on different lines), how many calls for help are associated with it?
* If a file is in the package com.google.devtools.build, what are the packages that it is associated with?
* popular\_packages and help\_packages are both named PCollections and both used in the Scores (side inputs) step of the pipeline. Which one is the main input and which is the side input?
* What is the method used in the Scores step?
* What Python data type is the side input converted into in the Scores step?

The Java version of this program is slightly different from the Python version. The Java SDK supports AsMap and the Python SDK doesn't. It supports AsDict instead. In Java, the PCollection is converted into a View as a preparatory step before it is used. In Python, the PCollection conversion occurs in the step where it is used.

## Task 4. Execute the pipeline

1. Change into the directory:

cd ~/training-data-analyst/courses/data\_analysis/lab2/python

1. The program requires BUCKET and PROJECT values and choosing whether to run the pipeline locally using --DirectRunner or on the cloud using --DataFlowRunner
2. Execute the pipeline locally by typing the following into Cloud Shell.

python JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $DEVSHELL\_PROJECT\_ID --DirectRunner

1. Once the pipeline has finished executing, On the **Navigation menu** (7a91d354499ac9f1.png), click **Storage > Browser** and click on your bucket. You will find the results in the **javahelp** folder. Click on the **Result** object to examine the output.
2. Execute the pipeline on the cloud by typing the following into Cloud Shell.

python JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $DEVSHELL\_PROJECT\_ID --DataFlowRunner

1. Return to the browser tab for Console. On the **Navigation menu** (7a91d354499ac9f1.png), click **Dataflow** and click on your job to monitor progress.
2. Once the pipeline has finished executing, On the **Navigation menu** (7a91d354499ac9f1.png) click **Storage > Browser** and click on your bucket. You will find the results in the **javahelp** folder. Click on the **Result** object to examine the output.