Serverless Data Analysis with Dataflow: Side Inputs (Python)

## 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

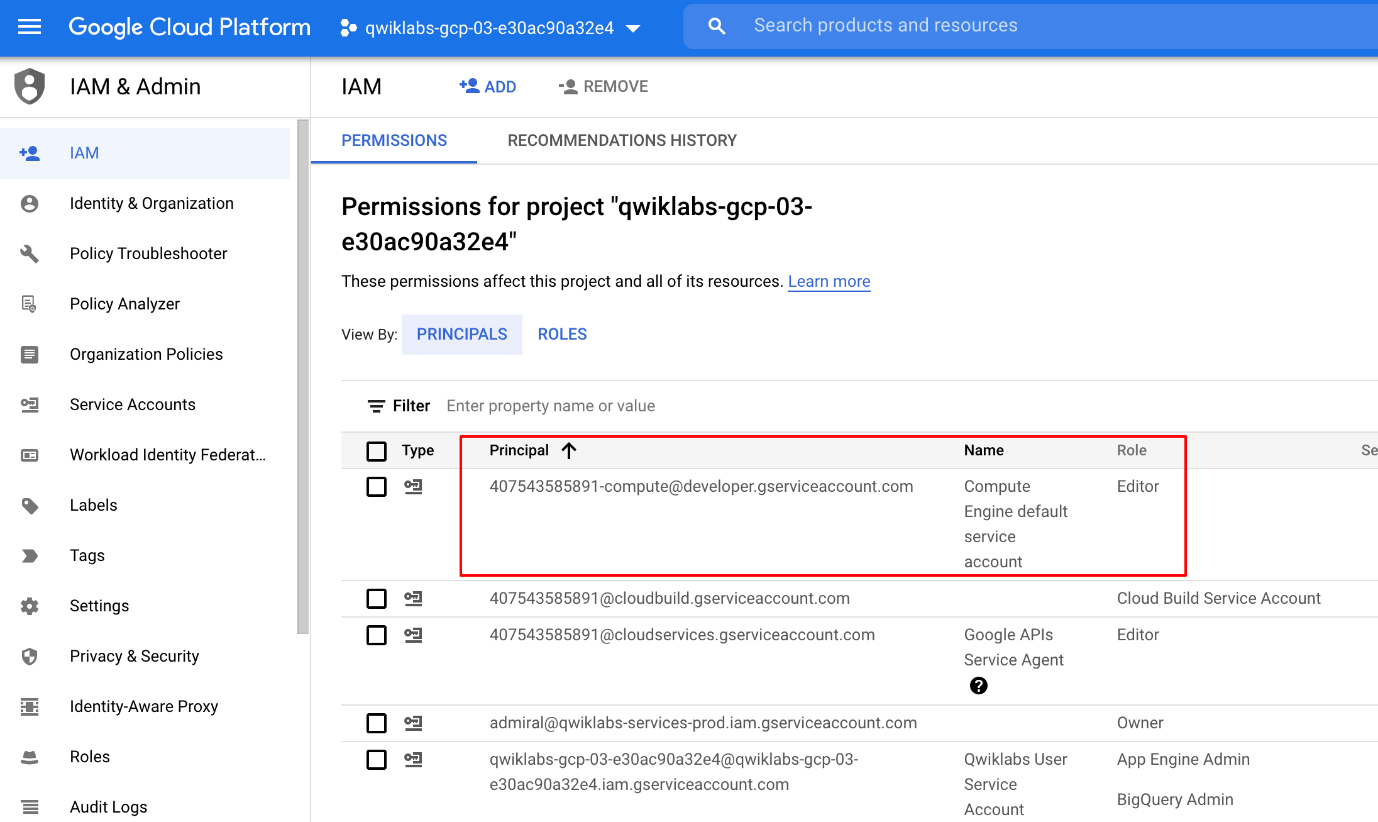
In this lab, you learn how to use BigQuery as a data source into Dataflow, and how to use the results of a pipeline as a side input to another pipeline.

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

### **Check project permissions**

Before you begin your work on Google Cloud, you need to ensure that your project has the correct permissions within Identity and Access Management (IAM).

1. In the Google Cloud console, on the **Navigation menu** (Navigation menu icon), click **IAM & Admin** > **IAM**.
2. Confirm that the default compute Service Account {project-number}-compute@developer.gserviceaccount.com is present and has the editor role assigned. The account prefix is the project number, which you can find on **Navigation menu** > **Home**.



If the account is not present in IAM or does not have the editor role, follow the steps below to assign the required role.

* In the Google Cloud console, on the **Navigation menu**, click **Home**.
* Copy the project number (e.g. 729328892908).
* On the **Navigation menu**, click **IAM & Admin** > **IAM**.
* At the top of the **IAM** page, click **Add**.
* For **New principals**, type:

{project-number}-compute@developer.gserviceaccount.com

Copied!

content\_copy

Replace {project-number} with your project number.

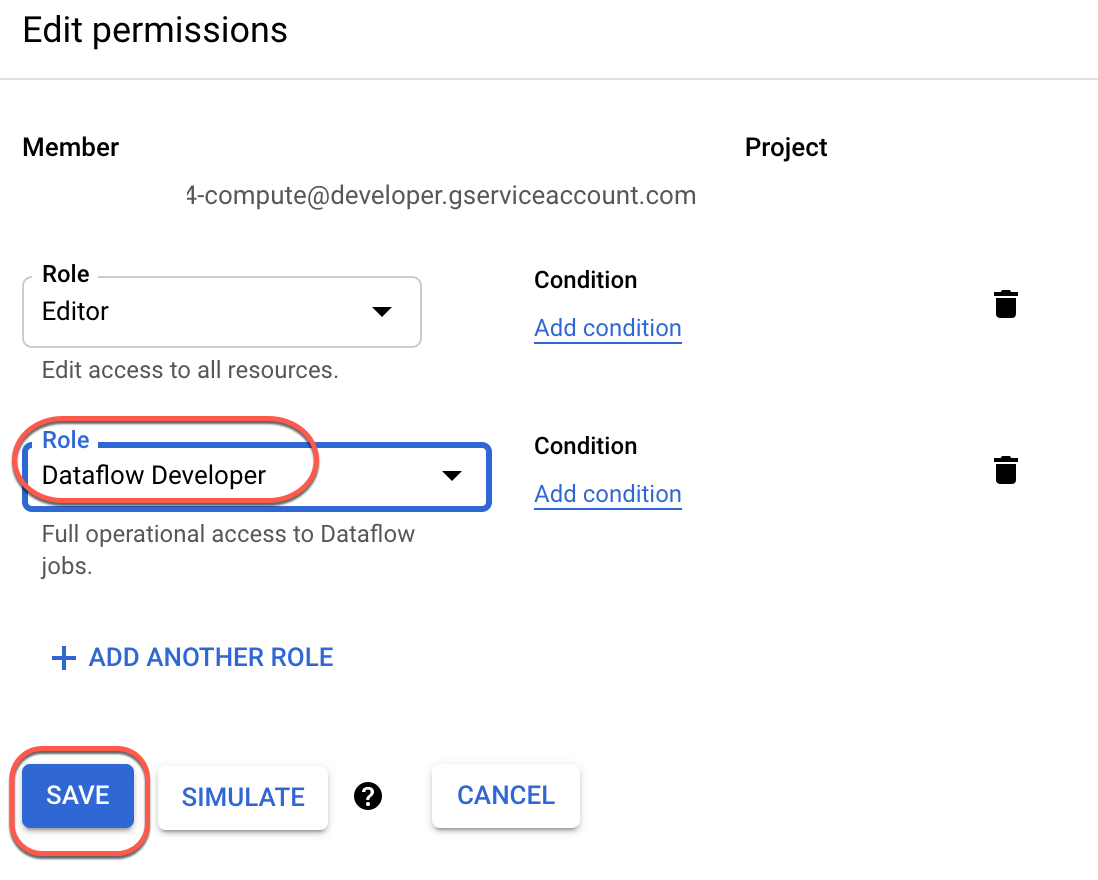
* For **Role**, select **Project** (or Basic) > **Editor**. Click **Save**.

## Task 1. Preparation

### **Assign the Dataflow Developer Role**

If the account does not have the Dataflow Developer role, follow the steps below to assign the required role.

* On the **Navigation menu**, click **IAM & Admin** > **IAM**.
* Select the default compute Service Account {project-number}-compute@developer.gserviceaccount.com.
* Select the **Edit** option (the pencil on the far right).
* Click **Add Another Role**
* Click inside the box for **Select a Role**. In the **Type to filter** selector, type and choose **Dataflow Developer**.
* Click **Save**.



### **Ensure that the Dataflow API is successfully enabled**

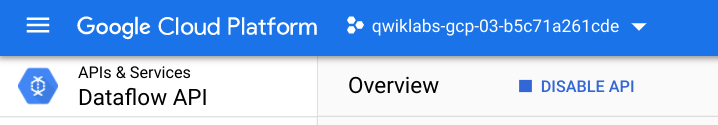
To ensure access to the necessary API, restart the connection to the Dataflow API.

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**.

1. Click **Enable**.

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



### **Open the SSH terminal and connect to the training VM**

You will be running all code from a curated training VM.

1. In the Console, on the **Navigation menu** (Navigation menu icon), click **Compute Engine** > **VM instances**.
2. Locate the line with the instance called **training-vm**.
3. On the far right, under **Connect**, click on **SSH** to open a terminal window.
4. In this lab, you will enter CLI commands on the **training-vm**.

### **Download Code Repository**

1. Next you will download a code repository for use in this lab. In the **training-vm** SSH terminal enter the following:

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

Copied!

content\_copy

### **Create a Cloud Storage bucket**

Follow these instructions to create a bucket.

1. In the Console, on the **Navigation menu**, 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.

1. In the Console, on the **Navigation menu**, click **Cloud Storage** > **Browser**.
2. Click **Create Bucket**.
3. 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)> |
| **Location type** | Multi-Region |
| **Location** | <Your location> |

1. Click **Create**.

Record the name of your bucket. You will need it in subsequent tasks.

1. In the **training-vm** SSH terminal enter the following to create two environment variables. One named "BUCKET" and the other named "PROJECT". Verify that each exists with the echo command.

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

echo $BUCKET

Copied!

content\_copy

PROJECT="<your unique project name (Project ID)>"

echo $PROJECT

Copied!

content\_copy

## Task 2. Try using BigQuery query

1. In the console, on the **Navigation menu** (), click **BigQuery**.
2. If prompted click **Done**.
3. Click **Compose new query** and type the following query.

SELECT

content

FROM

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

LIMIT

10

Copied!

content\_copy

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`

Copied!

content\_copy

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. Return to the **training-vm** SSH terminal and navigate to the directory /training-data-analyst/courses/data\_analysis/lab2/python and view the file JavaProjectsThatNeedHelp.py.

View the file with Nano. **Do not make any changes to the code.** Press **Ctrl+X** to exit Nano.

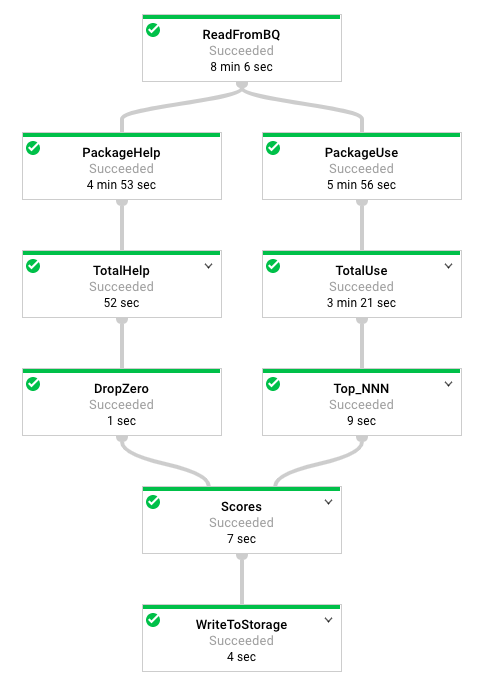
cd ~/training-data-analyst/courses/data\_analysis/lab2/python

nano JavaProjectsThatNeedHelp.py

Copied!

content\_copy

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. 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 the **training-vm** SSH terminal.

python3 JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $PROJECT --DirectRunner

Copied!

content\_copy

**Note:**Please ignore the warning if any and move forward.

1. Once the pipeline has finished executing, On the **Navigation menu** (), click **Cloud 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 the **training-vm** SSH terminal.

python3 JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $PROJECT --DataFlowRunner

Copied!

content\_copy

**Note:**Please ignore the warning if any and move forward.

1. Return to the browser tab for Console. On the **Navigation menu** (), click **Dataflow** and click on your job to monitor progress.
2. Once the pipeline has finished executing, On the **Navigation menu** () click **Cloud 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. The file name will be the same but you will notice that the file creation time is more recent.

Click Check my progress to verify the objective.

Execute the pipeline

Check my progress

## End your lab