

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

Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, 1 : 15 : 00), and make sure you can finish within that time.


There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **Start lab**.
4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
If you use other credentials, you'll receive errors or **incur charges**.
7. Accept the terms and skip the recovery resource page.

Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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** () , select **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 > Cloud Overview > Dashboard**.

Permissions for project "qwiklabs-gcp-00-3f97701829bb"

These permissions affect this project and all of its resources. [Learn more](#)

☐ Include




VIEW BY PRINCIPALS

VIEW BY ROLES

+ GRANT ACCESS

- REMOVE ACCESS

Filter Enter property name or value

<input type="checkbox"/> Type	Principal ↑	Name	Role	Security insights ⓘ
<input type="checkbox"/> 	96496971506-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor Owner	
<input type="checkbox"/> 	admiral@qwiklabs-services-prod.iam.gserviceaccount.com		Owner	
<input type="checkbox"/> 	qwiklabs-gcp-00-3f97701829bb@qwiklabs-gcp-00-3f97701829bb.iam.gserviceaccount.com	Qwiklabs User Service Account	BigQuery Admin Owner	

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

1. In the Google Cloud console, on the **Navigation menu**, click **Cloud Overview > Dashboard**.
2. Copy the project number (e.g. 729328892908).
3. On the **Navigation menu**, select **IAM & Admin > IAM**.
4. At the top of the roles table, below **View by Principals**, click **Grant Access**.
5. For **New principals**, type:
`{project-number}-compute@developer.gserviceaccount.com`

6. Replace `{project-number}` with your project number.
7. For **Role**, select **Project (or Basic) > Editor**.
8. 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.

1. On the **Navigation menu**, click **IAM & Admin > IAM**.

2. Select the default compute Service Account `{project-number}-compute@developer.gserviceaccount.com`.
3. Select the **Edit** option (the pencil on the far right).
4. Click **Add Another Role**.
5. Click inside the box for **Select a Role**. In the **Type to filter** selector, type and choose **Dataflow Developer**.
6. Click **Save**.

Edit permissions

Member	Project
4-compute@developer.gserviceaccount.com	
<div><div>Role</div><div>Editor</div></div> <div>Edit access to all resources.</div>	<div><div>Condition</div><div>Add condition</div></div> <div></div>
<div><div>Role</div><div>Dataflow Developer</div></div> <div>Full operational access to Dataflow jobs.</div>	<div><div>Condition</div><div>Add condition</div></div> <div></div>
+ ADD ANOTHER ROLE	
<div><div>SAVE</div><div>SIMULATE</div><div>?</div><div>CANCEL</div></div>	

Ensure that the Dataflow API is successfully enabled


1. On the Google Cloud Console title bar, click **Activate Cloud Shell**. If prompted, click **Continue**.
2. Run the following commands to ensure that the Dataflow API is enabled cleanly in your project. If prompted, click **Authorize**:

```
gcloud services disable dataflow.googleapis.com
```

```
gcloud services enable dataflow.googleapis.com
```

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** () , 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. If prompted, click **Authorize**.
4. In this lab, you will enter CLI commands on the **training-vm**.

Download Code Repository

- 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
```

Create a Cloud Storage bucket

Follow these instructions to create a bucket.

1. In the Console, on the **Navigation menu**, click **Cloud Storage > Buckets**.
2. Click **+ Create**.
3. Specify the following, and leave the remaining settings as their defaults:

Property	Value (type value or select option as specified)
Name	qwiklabs-gcp-01-e823f76128b1
Location type > Region	us-central1

4. Click **Create**.
5. If you get the Public access will be prevented prompt, select Enforce public access prevention on this bucket and click **Confirm**.


6. In the **training-vm** SSH terminal enter the following to create three environment variables. One named "BUCKET", another named "PROJECT", and the last named "REGION". Verify that each exists with the echo command:

```
BUCKET="qwiklabs-gcp-01-e823f76128b1"  
echo $BUCKET
```

```
PROJECT="qwiklabs-gcp-01-e823f76128b1"  
echo $PROJECT
```

```
REGION="us-central1"  
echo $REGION
```

Task 2. Try using BigQuery query

1. In the console, on the **Navigation menu** () , click **BigQuery**.
2. If prompted click **Done**.
3. Click "+" (SQL Query) and type the following query:

```
SELECT  
  content  
FROM  
  `cloud-training-demos.github_repos.contents_java`  
LIMIT  
  10
```

4. Click on **Run**.

What is being returned?

The BigQuery table `cloud-training-demos.github_repos.contents_java` contains the content (and some metadata) of all the Java files present in GitHub in 2016.

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

```
SELECT  
  COUNT(*)  
FROM  
  `cloud-training-demos.github_repos.contents_java`
```

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

There were 0 records returned in the result.

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

This dataset is not properly set up for billing.
Cache is enabled so all queries process 0 bytes.
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

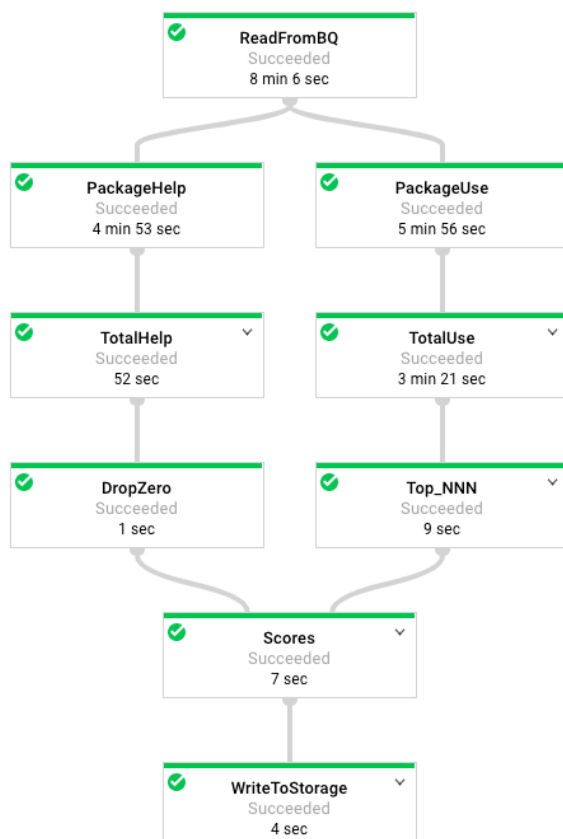
1. Return to the **training-vm** SSH terminal and navigate to the directory `/training-data-analyst/courses/data_analysis/lab2/python` and view the file `JavaProjectsThatNeedHelp.py`.

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
```

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



2. 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?


Note: 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, PROJECT, and REGION values and whether you want 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  
--region $REGION --DirectRunner
```



Note: Please ignore the warning if any, such as '**BeamDeprecationWarning**', and move forward.

3. Once the pipeline has finished executing, On the **Navigation menu** () , click **Cloud Storage > Buckets** and click on your bucket. You will find the results in the **javahelp** folder. Click on the **Result** object to examine the output.

4. Execute the pipeline on the cloud by typing the following into the **training-vm** SSH terminal:

```
python3 JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $PROJECT  
--region $REGION --DataFlowRunner
```

Note: Please ignore the warning if any, such as '**BeamDeprecationWarning**', and move forward.

5. Return to the browser tab for Console. On the **Navigation menu** () , click **View All Products**, and select **Dataflow** from the Analytics section.
6. Click on your job to monitor progress.
7. Once the pipeline has finished executing, On the **Navigation menu** () click **Cloud Storage > Buckets** 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.

Assessment Completed!

Execute the pipeline

Assessment Completed!

End your lab