Serverless Data Analysis with Dataflow: Side Inputs (Java)

## 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

### **Activate Google Cloud Shell**

Google 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. Google Cloud Shell provides command-line access to your GCP resources.

1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.



1. Click **Continue**. 

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your PROJECT\_ID. For example:



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

You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

Output:

Credentialed accounts:

- <myaccount>@<mydomain>.com (active)

Example output:

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

Output:

[core]

project = <project\_ID>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](https://cloud.google.com/sdk/gcloud).

### **Launch Google Cloud Shell Code Editor**

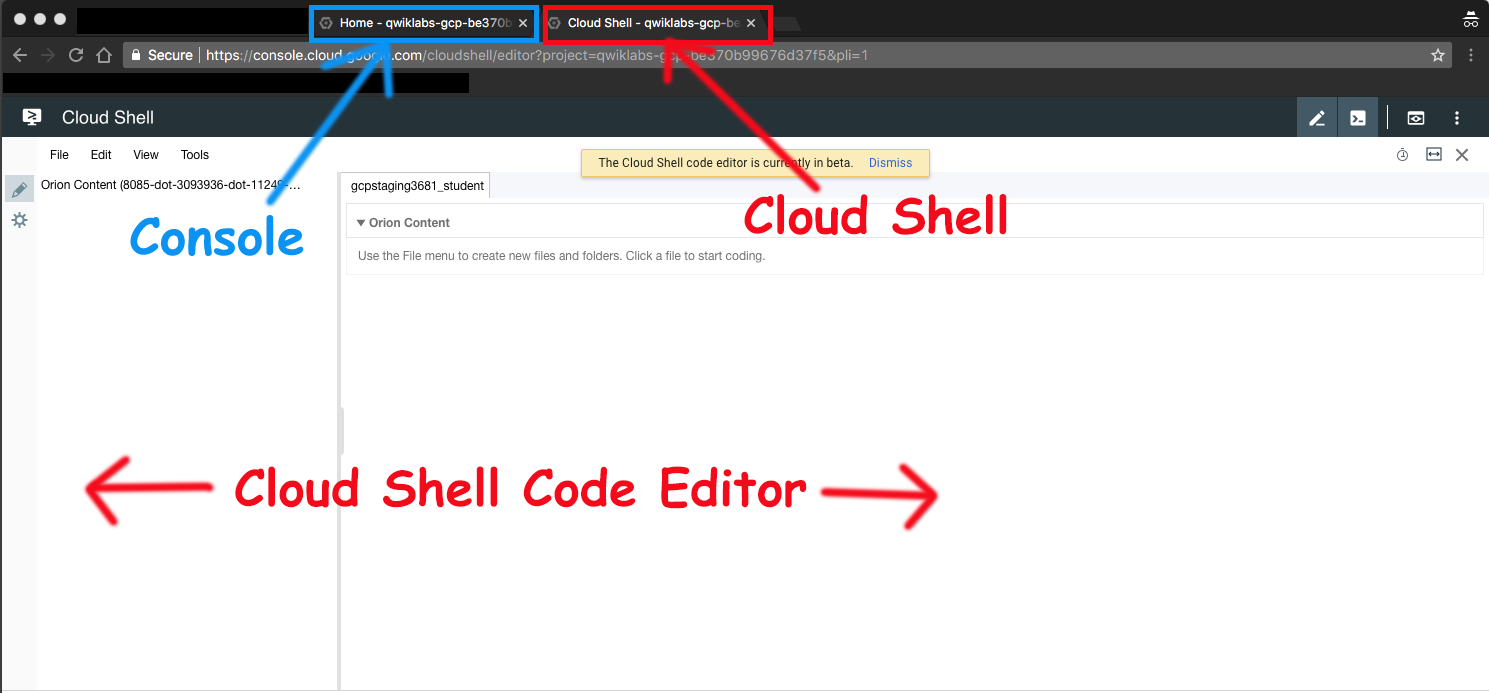
Use the Google Cloud Shell Code Editor to easily create and edit directories and files in the Cloud Shell instance.

Once you activate the Google Cloud Shell, click the **Open editor** button to open the Cloud Shell Code Editor.



You now have three interfaces available:

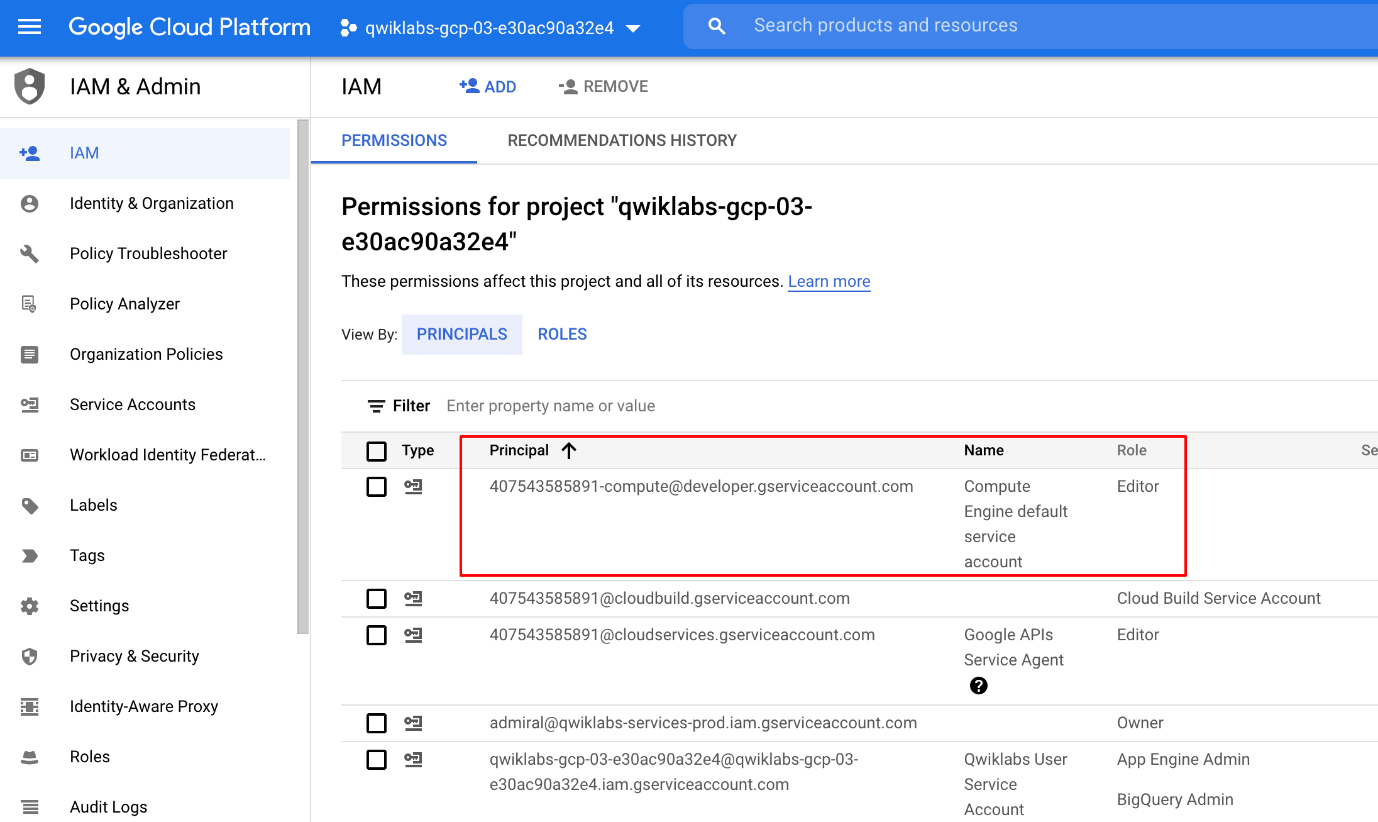
* The Cloud Shell Code Editor
* Console (By clicking on the tab). You can switch back and forth between the Console and Cloud Shell by clicking on the tab.
* The Cloud Shell Command Line (By clicking on **Open Terminal** in the Console)



### **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**.

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

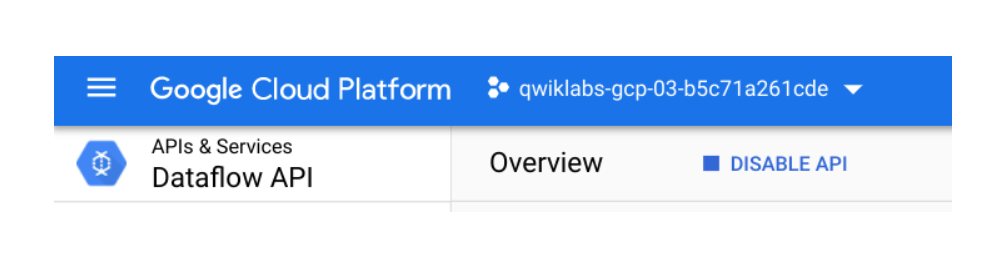
To ensure access to the necessary API, restart the connection to the Dataflow API.

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.



## 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

Copied!

content\_copy

1. Click on the **Refresh** icon 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** (), 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** (), click **Cloud 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-Region |
| **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

Copied!

content\_copy

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

Copied!

content\_copy

## Task 2. Try out 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** (), click **BigQuery** and then click **Done**.
2. 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, click **Compose New Query** and type the following query:

SELECT

COUNT(\*)

FROM

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

Copied!

content\_copy

Click on **Run**.

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

Copied!

content\_copy

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

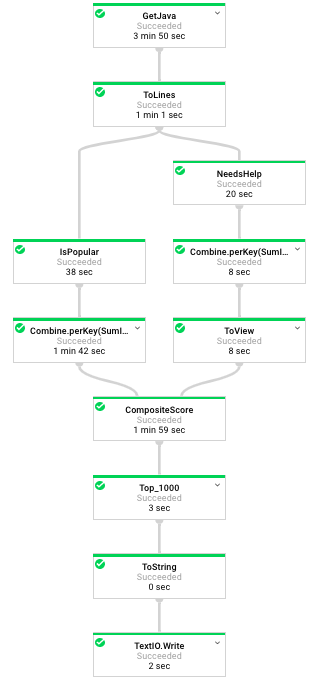
cd ~/training-data-analyst/courses/data\_analysis/lab2/javahelp

nano src/main/java/com/google/cloud/training/dataanalyst/javahelp/JavaProjectsThatNeedHelp.java

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 GetJava get Java content from?
* What does ToLines do? (Hint: look at the content field of the BigQuery result)
* Why is the result of ToLines stored in a named PCollection instead of being directly passed to another apply()?
* What are the two actions carried out on javaContent?
* 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?
* Why is the numHelpNeeded variable not enough? Why do we need to do Sum.integersPerKey()? (Hint: there are multiple files in a package)
* Why is this converted to a View?
* Which operation uses the View as a side input?
* Instead of simply ParDo.of(), this operation uses
* Besides c.element() and c.output(), this operation also makes use of what method in ProcessContext?

## Task 4. Execute the pipeline

1. Execute the pipeline by typing the following into Cloud Shell.

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

./run\_oncloud3.sh $DEVSHELL\_PROJECT\_ID $BUCKET JavaProjectsThatNeedHelp

Copied!

content\_copy

Wait until the command is fully executed. It will take around 5 to 7 minutes.

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, download and view the output:

gsutil cp gs://$BUCKET/javahelp/output.csv .

head output.csv

Copied!

content\_copy

Click Check my progress to verify the objective.

Execute the pipeline

Check my progress

## End your lab