Serverless Data Analysis with Dataflow: Side Inputs (Python) | Qwiklabs

Tuesday, November 17, 2020 3:59 PM

Clipped from: https://googlecourses.qwiklabs.com/course_sessions/71782/labs/11652

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 GCP project and set of resources for a fixed time at no cost.

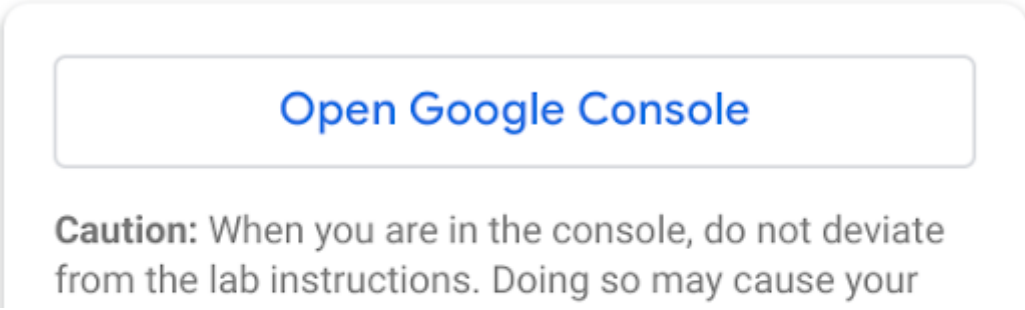
1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example,

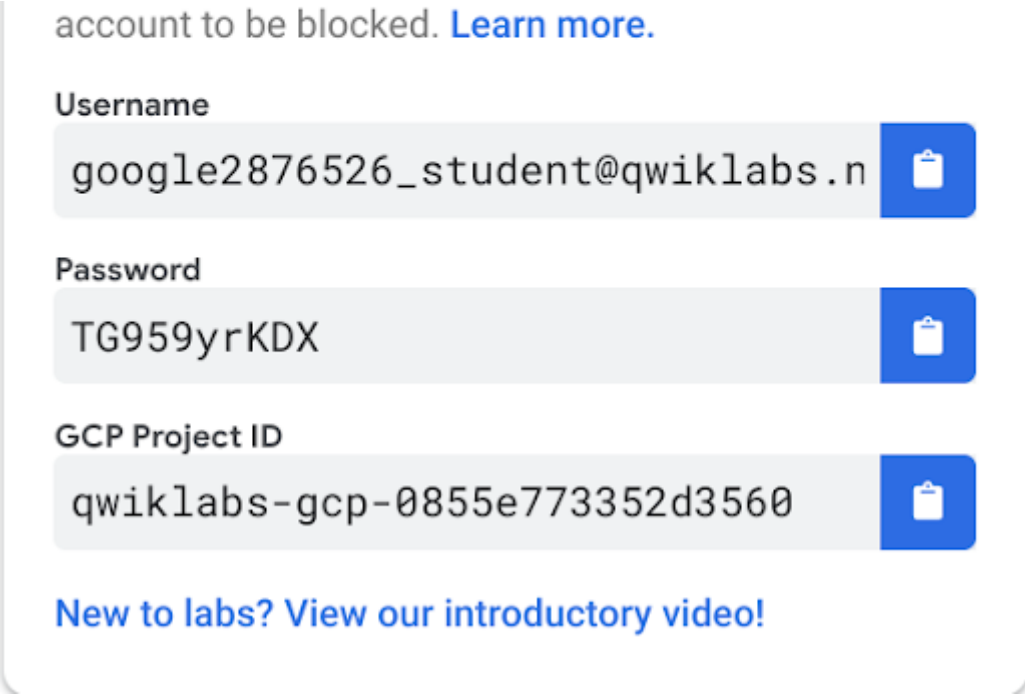
   02:00:00

   and make sure you can finish in that time block.
3. When ready, click

   START LAB

   .
4. Note your lab credentials. You will use them to sign in to Cloud Platform Console.





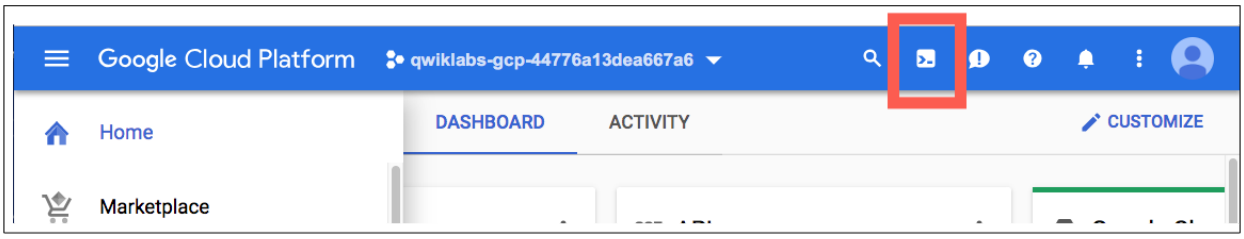
5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

1. Accept the terms and skip the recovery resource page.

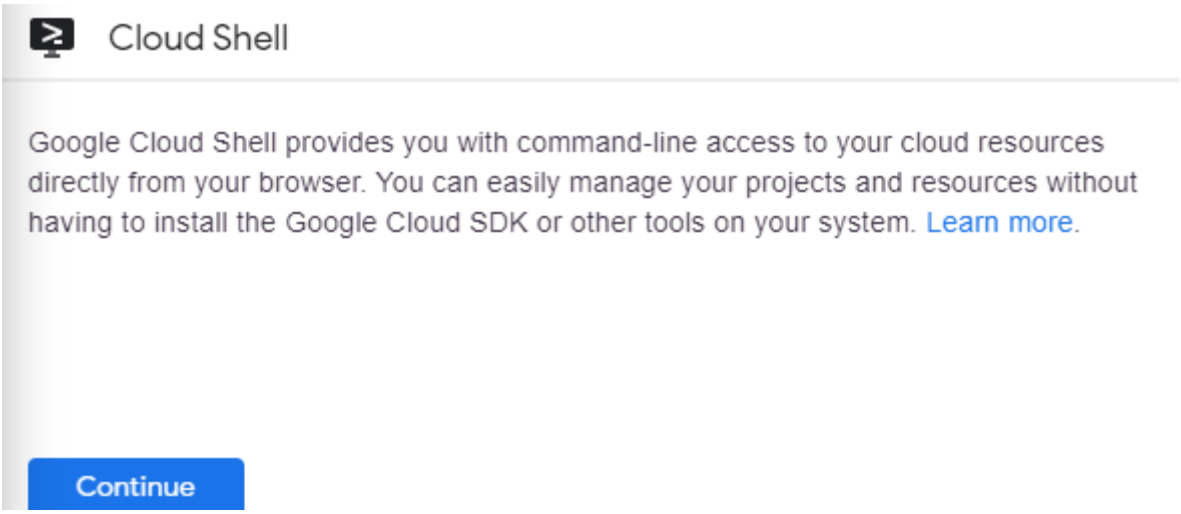
Activate Google Cloud Shell

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.

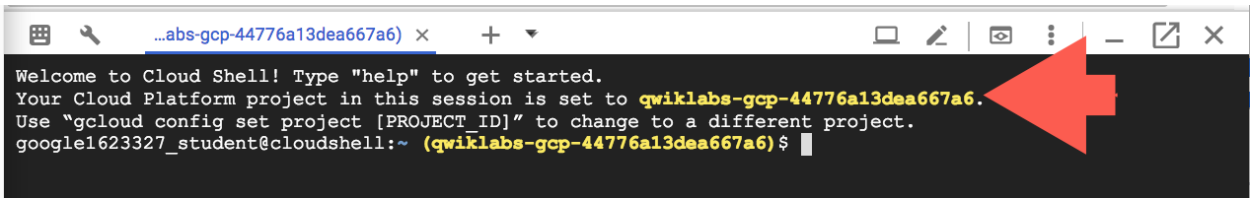
1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.



2. 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
```

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
```

Output:

```
[core]
project = <project_ID>
```

Example output:

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

Full documentation of **gcloud** is available on Google Cloud gcloud Overview .

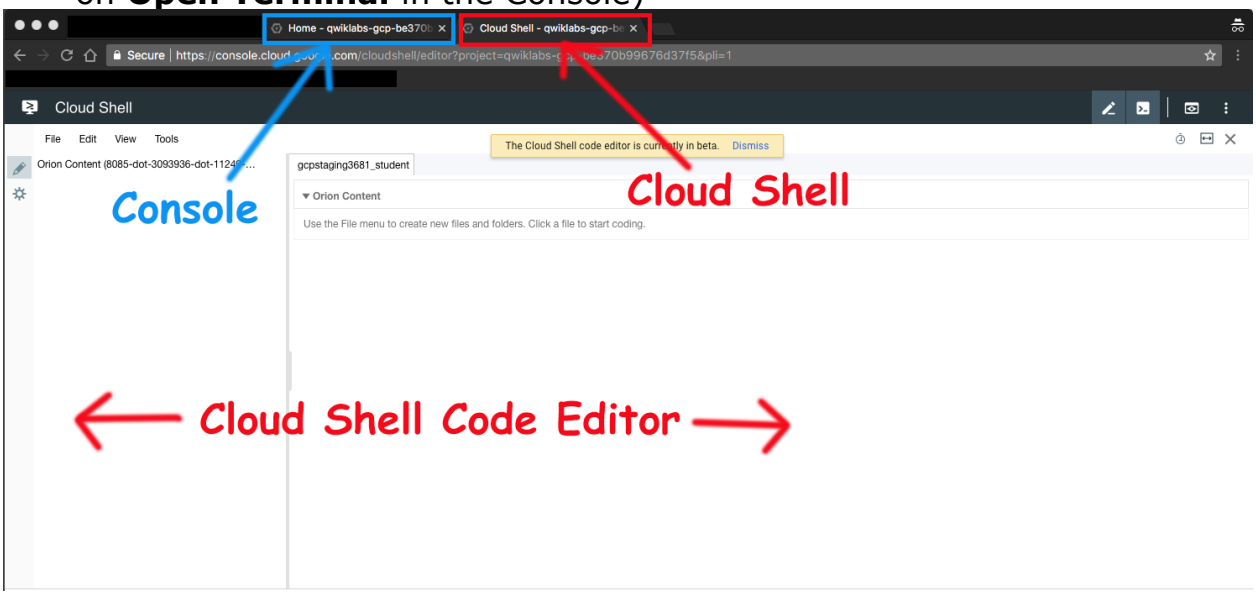
Launch Google Cloud Shell Code Editor

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:

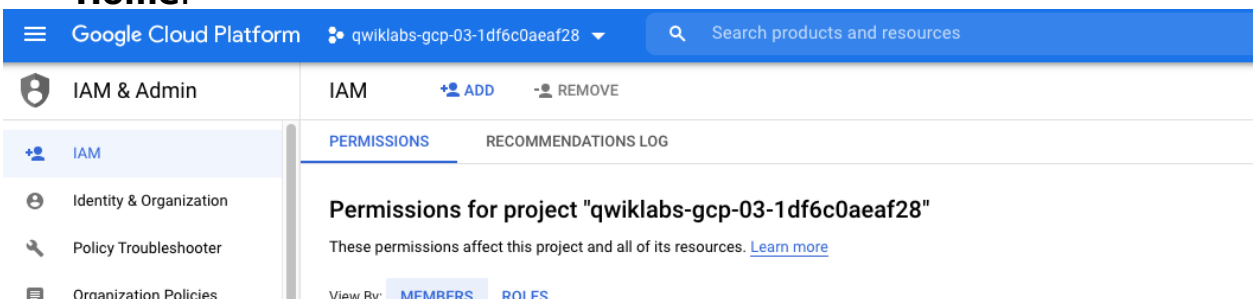
- The Cloud Shell Code Editor
- Console (By clicking on the tab). You can switch back and forth between the Console and Cloud Shell by clicking on the tab.
- The Cloud Shell Command Line (By clicking on **Open Terminal** in the Console)

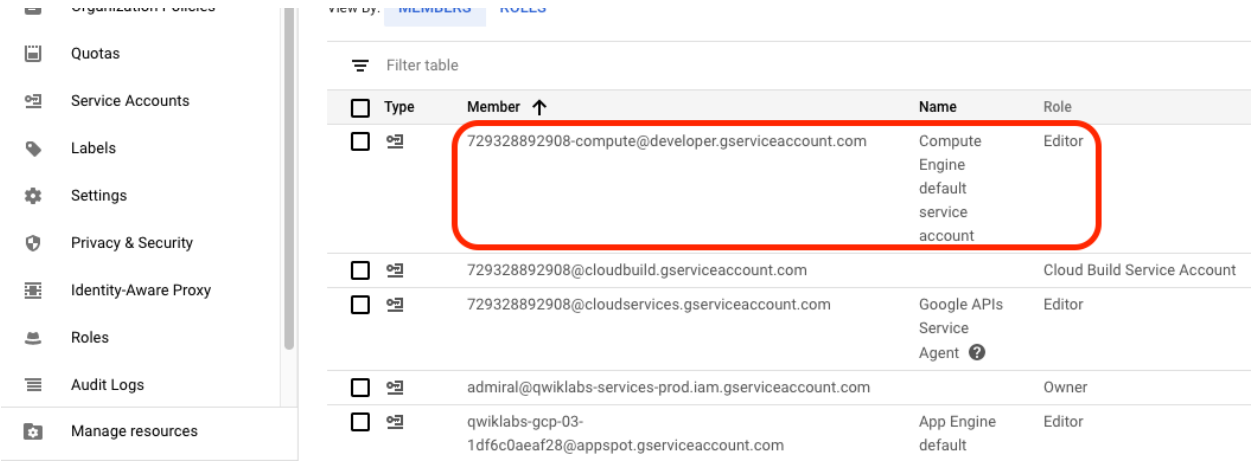


Check project permissions

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** ( ), 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**.





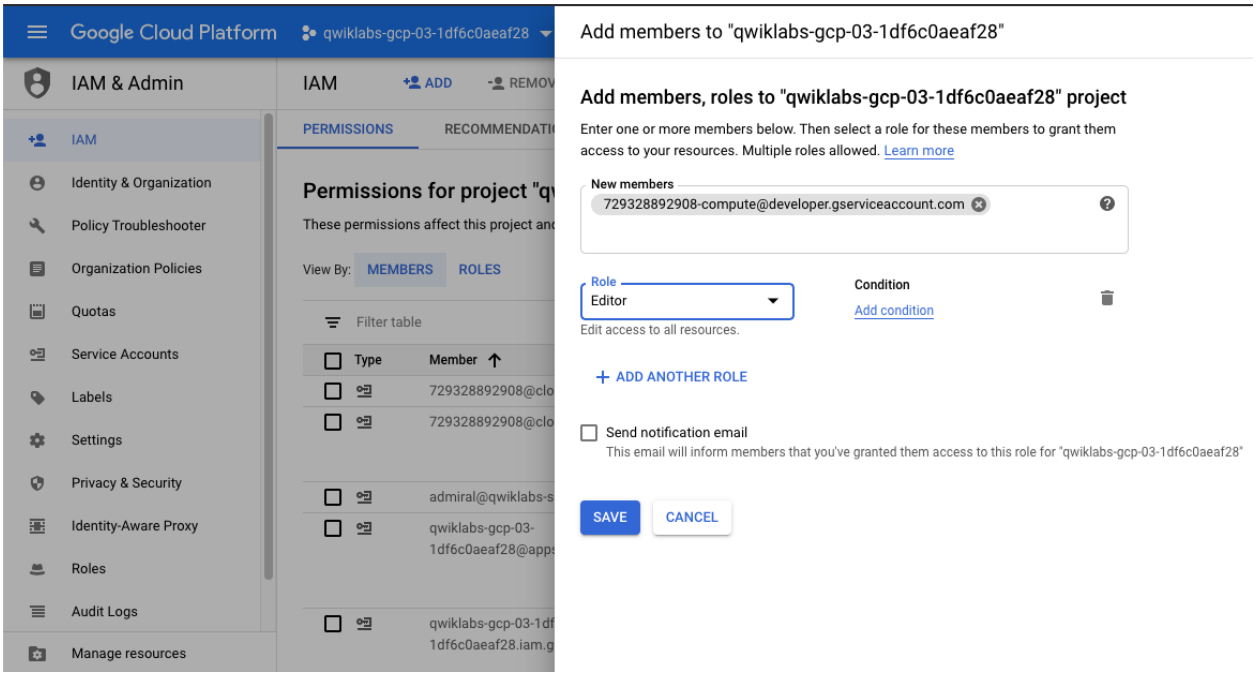
If the account is not present in IAM or does not have the `editor` role, follow the steps below to assign the required role.

- 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 members**, type:

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

Replace `{project-number}` with your project number.

- For **Role**, select **Project** > **Editor**. Click **Save**.



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. 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 **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=$(gcloud config get-value project)
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. In the console, on the **Navigation menu** ( ), click **BigQuery**.
2. Click **Compose new 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`
```

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:

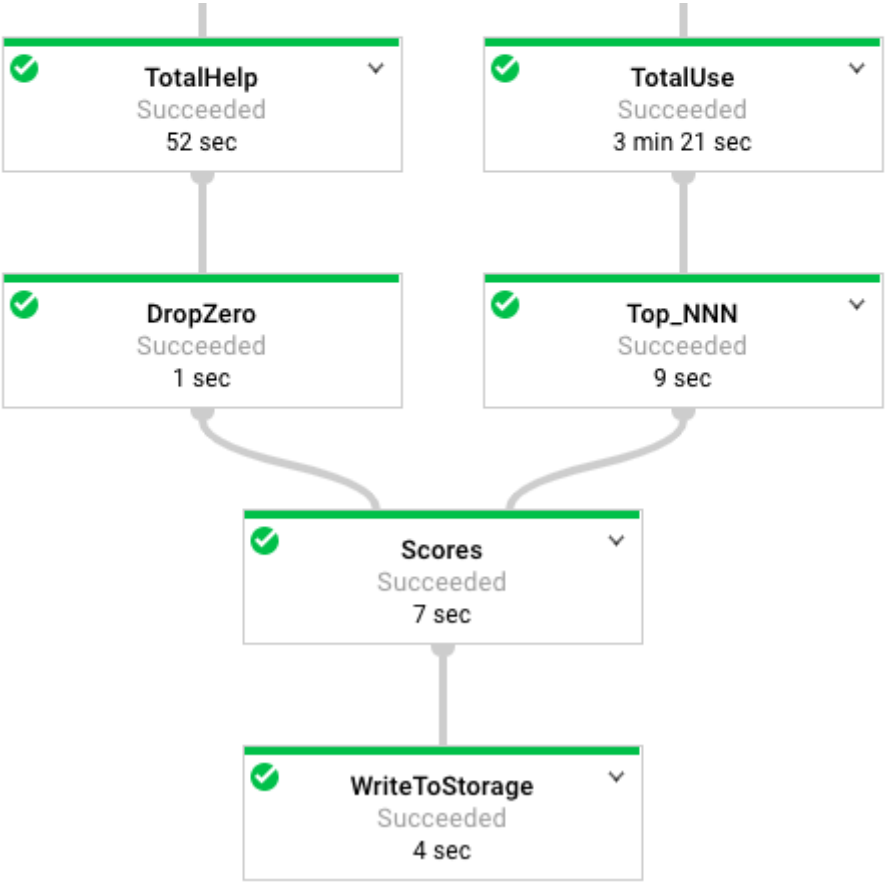
```
cd ~/training-data-analyst/courses/data_analysis/lab2/python
```

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?

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.

```
python3 JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $DEVSHELL_PROJECT_ID --DirectRunner
```

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

1. Once the pipeline has finished executing, On the **Navigation menu** ( ), 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.

```
python3 JavaProjectsThatNeedHelp.py --bucket $BUCKET --project $DEVSHELL_PROJECT_ID --DataFlowRunner
```

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 **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.

Click *Check my progress* to verify the objective. Execute the pipeline

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.