A Simple Dataflow Pipeline (Python) 2.5

## Overview

In this lab, you will open a Dataflow project, use pipeline filtering, and execute the pipeline locally and on the cloud.

* Open Dataflow project
* Pipeline filtering
* Execute the pipeline locally and on the cloud

## Objective

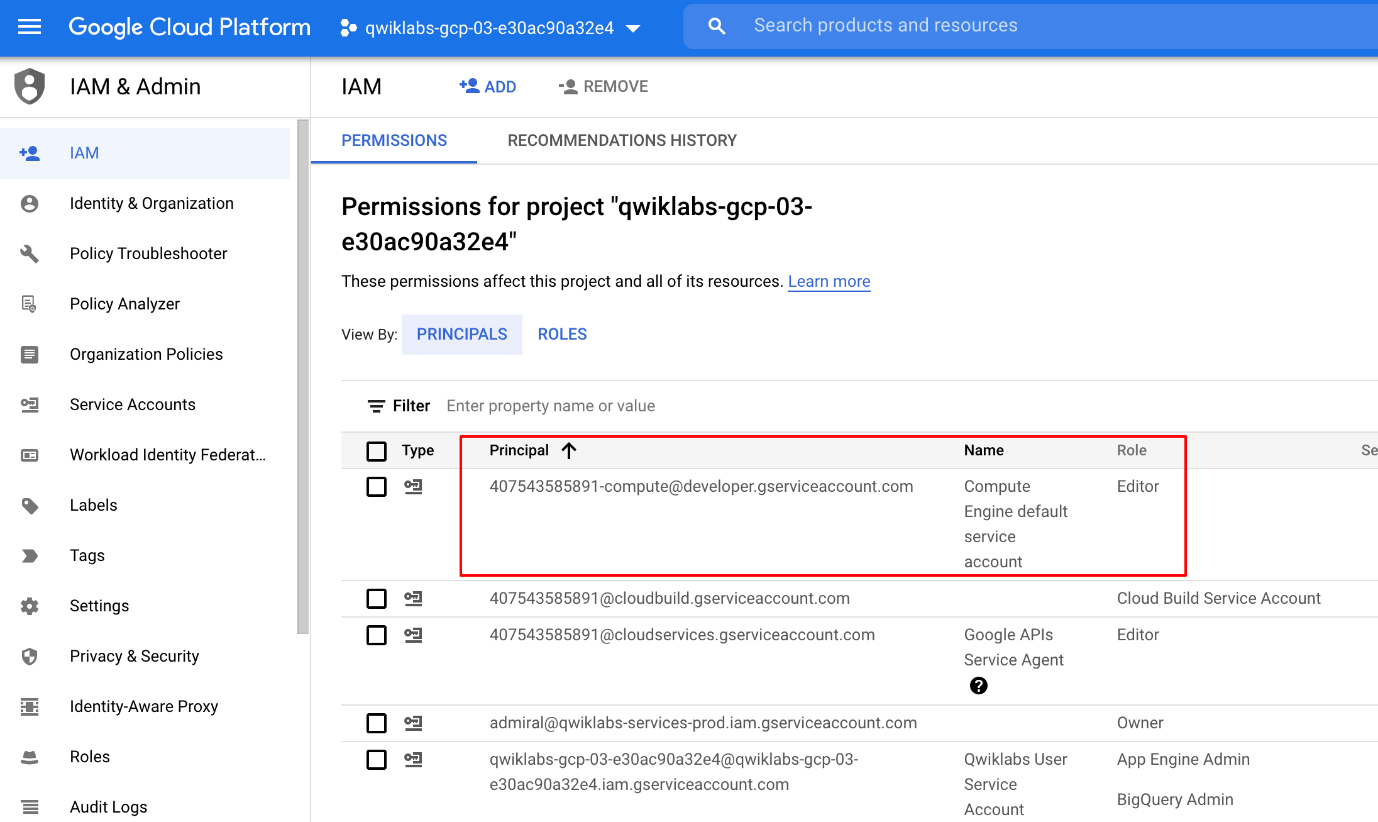
In this lab, you learn how to write a simple Dataflow pipeline and run it both locally and on the cloud.

* Setup a Python Dataflow project using Apache Beam
* Write a simple pipeline in Python
* Execute the query on the local machine
* Execute the query on the cloud

### **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. 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.
2. Click on the result for **Dataflow API**.
3. Click **Manage**.
4. Click **Disable API**.
5. If asked to confirm, click **Disable**.
6. Click **Enable**.

## Task 2. Preparation

### **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 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 terminal 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.

## Task 3. Pipeline filtering

The goal of this lab is to become familiar with the structure of a Dataflow project and learn how to execute a Dataflow pipeline.

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 grep.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 grep.py

Copied!

content\_copy

Can you answer these questions about the file grep.py?

* What files are being read?
* What is the search term?
* Where does the output go?

There are three transforms in the pipeline:

* What does the transform do?
* What does the second transform do?
* Where does its input come from?
* What does it do with this input?
* What does it write to its output?
* Where does the output go to?
* What does the third transform do?

## Task 4 Execute the pipeline locally

1. In the **training-vm** SSH terminal, locally execute grep.py.

python3 grep.py

Copied!

content\_copy

The output file will be output.txt. If the output is large enough, it will be sharded into separate parts with names like: output-00000-of-00001.

1. Locate the correct file by examining the file's time.

ls -al /tmp

Copied!

content\_copy

1. Examine the output file(s).
2. You can replace "-\*" below with the appropriate suffix.

cat /tmp/output-\*

Copied!

content\_copy

Does the output seem logical?

## Task 5. Execute the pipeline on the cloud

1. Copy some Java files to the cloud. In the **training-vm** SSH terminal, enter the following commmand:

gsutil cp ../javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/\*.java gs://$BUCKET/javahelp

Copied!

content\_copy

1. Using Nano, edit the Dataflow pipeline in grepc.py.

nano grepc.py

Copied!

content\_copy

1. Replace PROJECT and BUCKET with your Project ID and Bucket name.

Example strings before:

PROJECT='cloud-training-demos'

BUCKET='cloud-training-demos'

Copied!

content\_copy

Example strings after edit (use your values):

PROJECT='qwiklabs-gcp-your-value'

BUCKET='qwiklabs-gcp-your-value'

Copied!

content\_copy

Save the file and close Nano by pressing the **CTRL+X** key, then press **Y**, and **Enter**.

1. Submit the Dataflow job to the cloud:

python3 grepc.py

Copied!

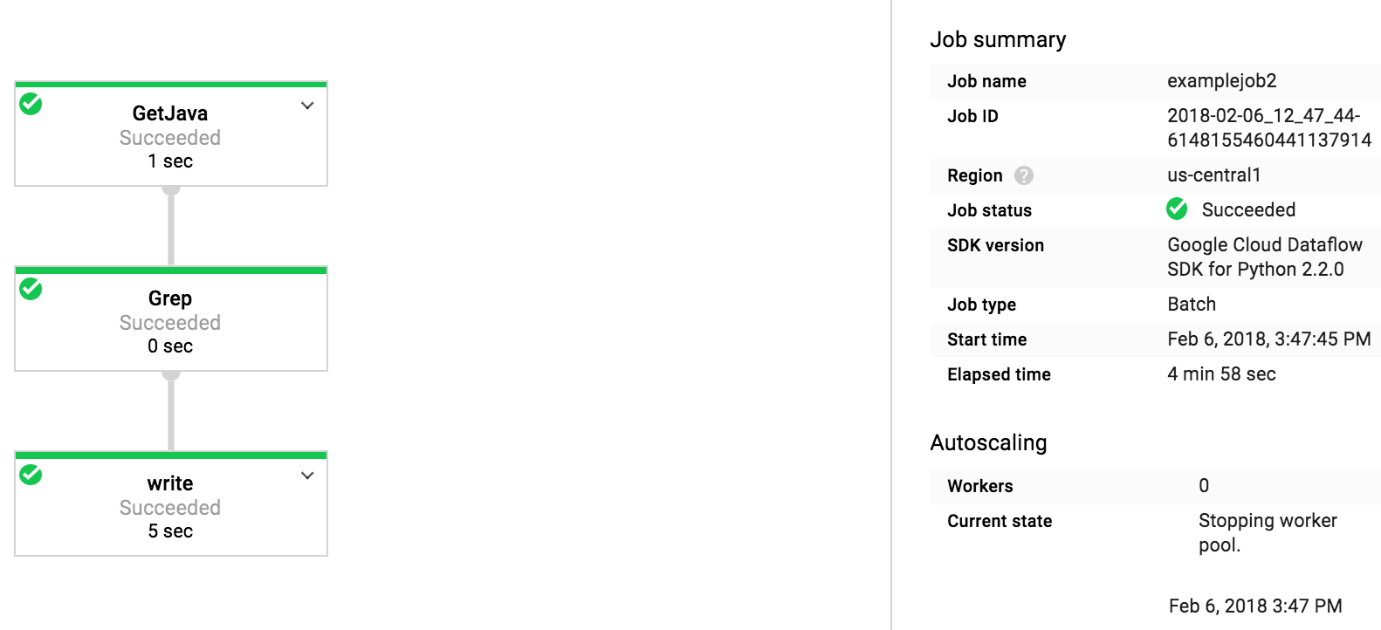
content\_copy

**Note:** You may ignore the message: **WARNING:root:Make sure that locally built Python SDK docker image has Python 3.7 interpreter.** Your Dataflow job will start successfully.

Because this is such a small job, running on the cloud will take significantly longer than running it locally (on the order of 7-10 minutes).

1. Return to the browser tab for Console.
2. On the **Navigation menu**, click **Dataflow** and click on your job to monitor progress.

Example:



1. Wait for the job status to turn to **Succeeded**.
2. Examine the output in the Cloud Storage bucket.
3. On the **Navigation menu**, click **Cloud Storage > Browser** and click on your bucket.
4. Click the **javahelp** directory.

This job will generate the file output.txt. If the file is large enough it will be sharded into multiple parts with names like: output-0000x-of-000y. You can identify the most recent file by name or by the **Last modified** field.

1. Click on the file to view it.

Alternatively, you can download the file via the **training-vm** SSH terminal and view it:

gsutil cp gs://$BUCKET/javahelp/output\* .

cat output\*

Copied!

content\_copy

## End your lab