Serverless Data Analysis with Dataflow: A Simple Dataflow Pipeline (Java)

## Overview

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

## Objective

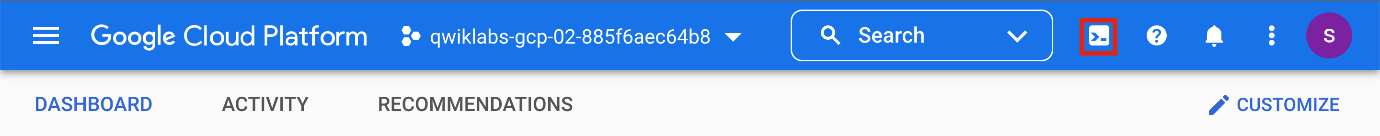
In this lab, you will learn how to write a simple Dataflow pipeline and run it both locally and on the cloud.

* Setup a Java Dataflow project using Maven
* Write a simple pipeline in Java
* Execute the query on the local machine
* Execute the query on the cloud

### **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**

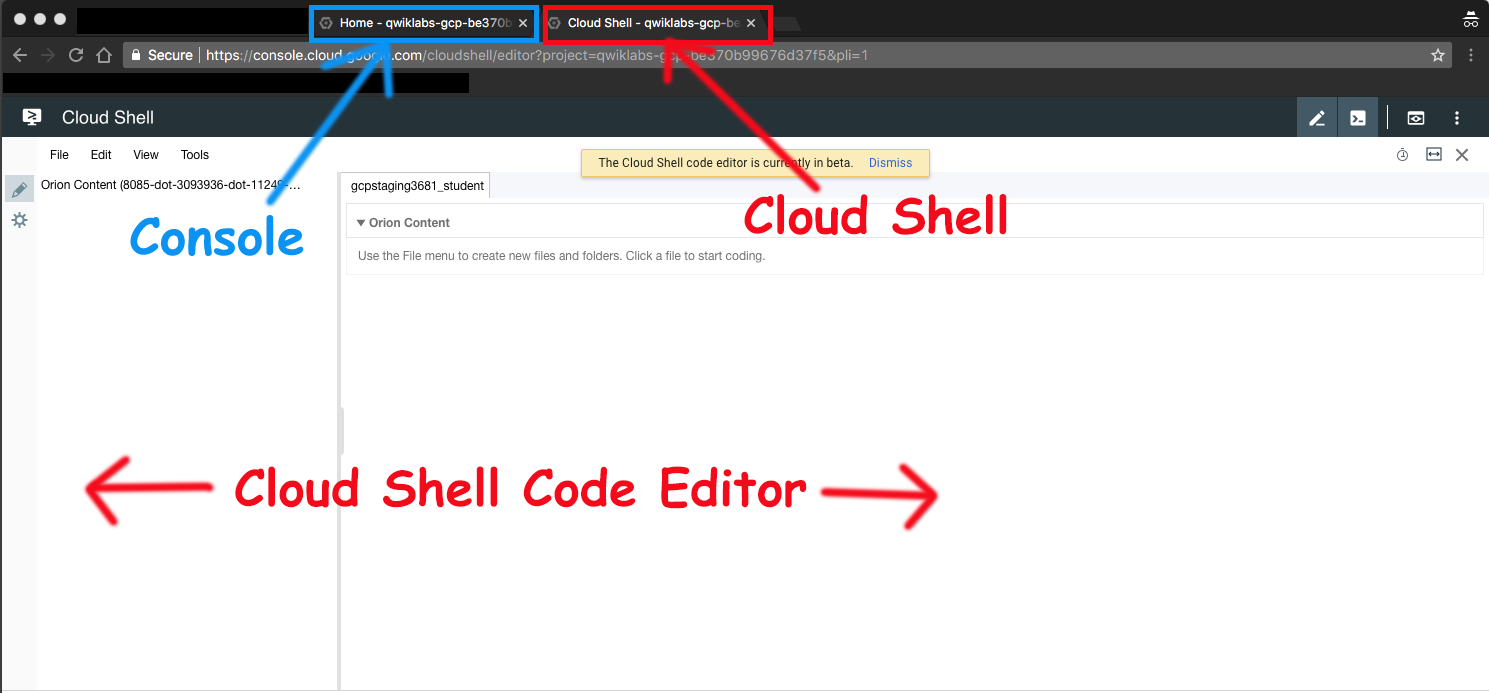
Use the Google Cloud Shell Code Editor to easily create and edit directories and files in the Cloud Shell instance.

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:

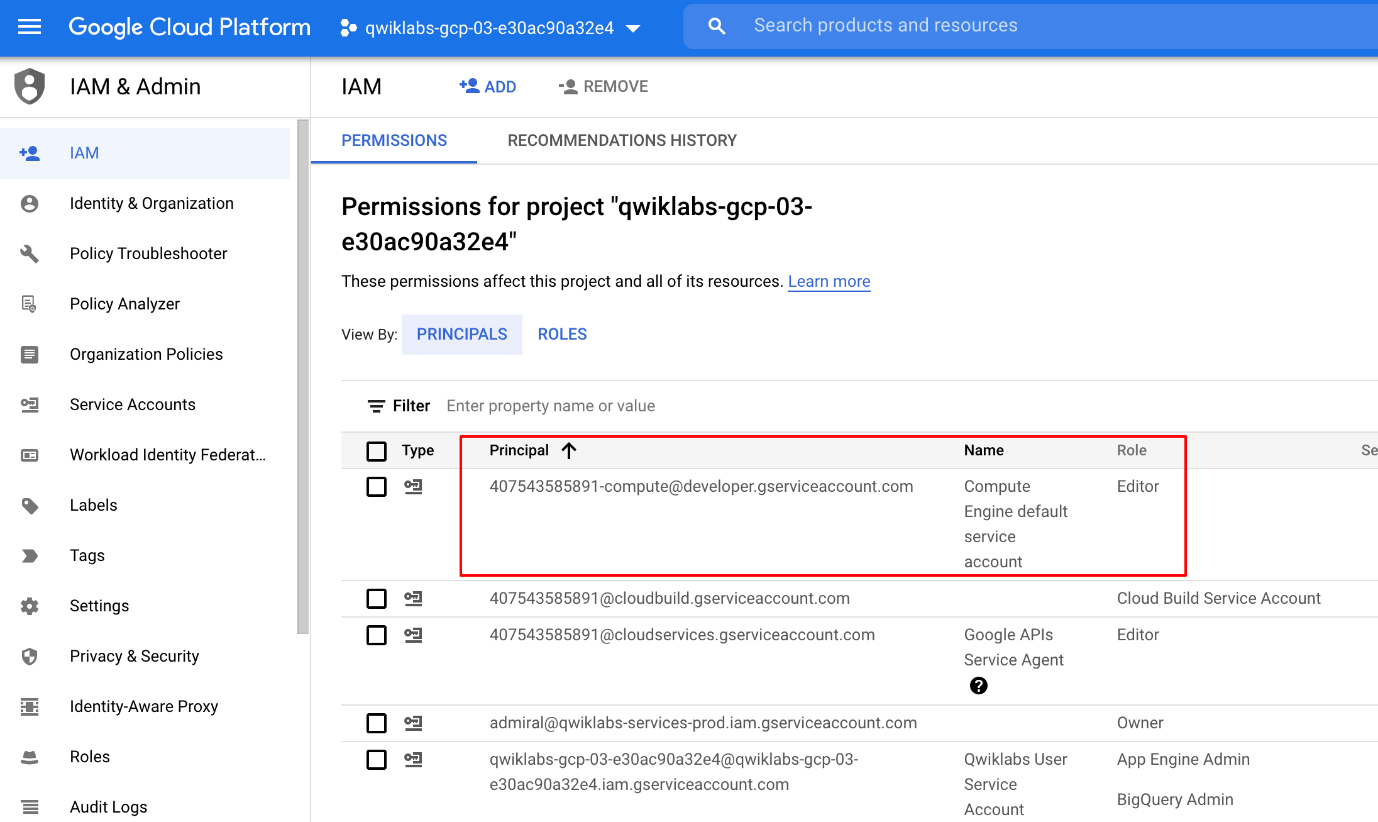
* The Cloud Shell Code Editor
* Console (By clicking on the tab). You can switch back and forth between the Console and Cloud Shell by clicking on the tab.
* The Cloud Shell Command Line (By clicking on **Open Terminal** in the Console)



### **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**.

## Task 1. Preparation

For this lab, you will need the training-data-analyst files and a Cloud Storage bucket.

### **Verify that the repository files are in Cloud Shell Editor**

1. Clone the repository from the Cloud Shell command line:

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

Copied!

content\_copy

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 **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-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="<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 that Dataflow API is enabled for this project**

1. Return to the browser tab for Console. In the top search bar, enter **Dataflow API**. This will take you to the page, **Navigation menu > APIs & Services > Library > Dataflow API**.

It will either show a status information or it will give you the option to **Enable** the API.

1. If necessary, **Enable** the API.

## Task 2. Create a new Dataflow project

The goal of this lab is to become familiar with the structure of a Dataflow project and learn how to execute a Dataflow pipeline. You will use the powerful build tool [Maven](https://maven.apache.org/) to create a new Dataflow project.

1. Return to the browser tab containing Cloud Shell. In Cloud Shell navigate to the directory for this lab:

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

Copied!

content\_copy

1. Copy and paste the following Maven command:

mvn archetype:generate \

-DarchetypeArtifactId=google-cloud-dataflow-java-archetypes-starter \

-DarchetypeGroupId=com.google.cloud.dataflow \

-DgroupId=com.example.pipelinesrus.newidea \

-DartifactId=newidea \

-Dversion="[1.0.0,2.0.0]" \

-DinteractiveMode=false

Copied!

content\_copy

* What directory has been created?
* What package has been created inside the src directory?

1. Examine the Maven command that was used to create the lab code:

cat ~/training-data-analyst/courses/data\_analysis/lab2/create\_mvn.sh

Copied!

content\_copy

* What directory will get created?
* What package will get created inside the src directory?

## Task 3. Pipeline filtering

1. In the Cloud Shell code editor navigate to the directory /training-data-analyst/courses/data\_analysis/lab2.

Then select the path javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/ and view the file Grep.java.

Alternatively, you could view the file with nano editor. **Do not make any changes to the code**.

cd ~/training-data-analyst/courses/data\_analysis/lab2/javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp/

nano Grep.java

Copied!

content\_copy

Can you answer these questions about the file Grep.java?

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

There are three apply statements in the pipeline:

* What does the first apply() do?
* What does the second apply() 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 apply() do?

## Task 4. Execute the pipeline locally

1. In Cloud Shell, paste the following Maven command:

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

export PATH=/usr/lib/jvm/java-8-openjdk-amd64/bin/:$PATH

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

mvn compile -e exec:java \

-Dexec.mainClass=com.google.cloud.training.dataanalyst.javahelp.Grep

Copied!

content\_copy

1. 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. If necessary, you can locate the correct file by examining the file's time.

ls -al /tmp

Copied!

content\_copy

1. Examine the output file. 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.

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

Copied!

content\_copy

Click Check my progress to verify the objective.

Copy Java files to the Cloud

Check my progress

1. Edit the Dataflow pipeline in Grep.java. In the Cloud Shell code editor navigate to the directory /training-data-analyst/courses/data\_analysis/lab2/javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp and edit the file Grep.java.

cd ~/training-data-analyst/courses/data\_analysis/lab2/javahelp/src/main/java/com/google/cloud/training/dataanalyst/javahelp

Copied!

content\_copy

1. Replace Input and Output variables with your Bucket name. These must be the actual value, not the environment variable. Recall your Bucket name:

echo $BUCKET

Copied!

content\_copy

Replace the variables.

String input = "gs://<YOUR-BUCKET-NAME>/javahelp/\*.java";

String outputPrefix = "gs://<YOUR-BUCKET-NAME>/javahelp/output";

Copied!

content\_copy

Make sure that you changed the **input** and **outputPrefix** strings that are already present in the source code (do not copy-and-paste the entire line above because you will then end up with two variables named input).

Example lines before:

String input = "src/main/java/com/google/cloud/training/dataanalyst/javahelp/\*.java";

String outputPrefix = "/tmp/output";

Copied!

content\_copy

Example lines after edit (use your values):

String input = "gs://qwiklabs-gcp-your-value/javahelp/\*.java";

String outputPrefix = "gs://qwiklabs-gcp-your-value/javahelp/output";

Copied!

content\_copy

1. Examine the script to submit the Dataflow to the cloud:

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

cat run\_oncloud1.sh

Copied!

content\_copy

What is the difference between this Maven command and the one to run locally?

1. Submit the Dataflow job to the cloud.

bash run\_oncloud1.sh $DEVSHELL\_PROJECT\_ID $BUCKET Grep

Copied!

content\_copy

Because this is such a small job, running on the cloud will take significantly longer than running it locally (on the order of 2-3 minutes).

Example completion of command line (do not copy):

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 01:50 min

[INFO] Finished at: 2018-02-06T15:11:23-05:00

[INFO] Final Memory: 39M/206M

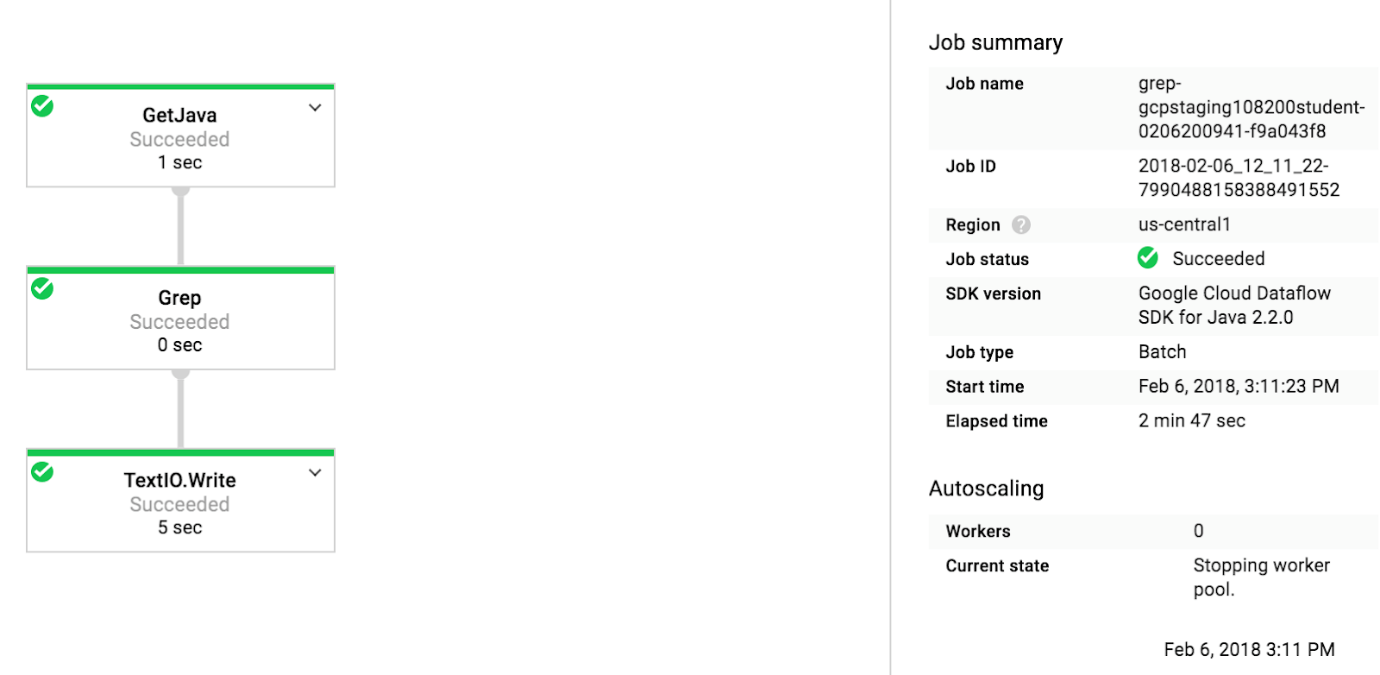
[INFO] ------------------------------------------------------------------------

Copied!

content\_copy

1. Return to the browser tab for Console. 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**. At this point, your Cloud Shell will display a command-line prompt.If Dataflow job fails the first time, then re-run the previous command to submit a fresh Dataflow job to the cloud

Click Check my progress to verify the objective.

Submit the Dataflow job to the Cloud

Check my progress

1. Examine the output in the Cloud Storage bucket. On the **Navigation menu** (), click **Cloud Storage > Browser** and click on your bucket.
2. 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. Click on the file to view it.

Alternatively, you could download the file in **Cloud Shell** and view it:

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

cat output\*

Copied!

content\_copy

## End your lab