Quickstart using Pythonbookmark_border

cloud.google.com/dataflow/docs/quickstarts/quickstart-python

In this quickstart, you learn how to use the Apache Beam SDK for Python to build a program that defines a pipeline. Then, you run the pipeline by using a direct local runner or a cloud-based runner such as Dataflow.

Before you begin

1. If you're new to Google Cloud, create an account to evaluate how our products perform in real-world scenarios. New customers also get $300 in free credits to run, test, and deploy workloads.
2. In the Google Cloud Console, on the project selector page, select or create a Google Cloud project.

   Note: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.
   Go to project selector

3. Make sure that billing is enabled for your Cloud project. Learn how to confirm that billing is enabled for your project.

4. Enable the Dataflow, Compute Engine, Cloud Logging, Cloud Storage, Google Cloud Storage JSON, BigQuery, Cloud Pub/Sub, Cloud Datastore, and Cloud Resource Manager APIs.
   Enable the APIs

5. Create a service account:

    1. In the Cloud Console, go to the **Create service account** page.

       Go to Create service account
    2. Select a project.
    3. In the **Service account name** field, enter a name. The Cloud Console fills in the **Service account ID** field based on this name.

       In the **Service account description** field, enter a description. For example, `Service account for quickstart`.
    4. Click **Create**.
    5. Click the **Select a role** field.

       Under **Quick access**, click **Basic**, then click **Owner**.

       Note: The **Role** field affects which resources your service account can access in your project. You can revoke these roles or grant additional roles later. In production environments, do not grant the Owner, Editor, or Viewer roles. For more information, see Granting, changing, and revoking access to resources.
    6. Click **Continue**.
    7. Click **Done** to finish creating the service account.

       Do not close your browser window. You will use it in the next step.

6. Create a service account key:

    1. In the Cloud Console, click the email address for the service account that you created.
    2. Click **Keys**.
    3. Click **Add key**, then click **Create new key**.
    4. Click **Create**. A JSON key file is downloaded to your computer.
    5. Click **Close**.

7. Set the environment variable `GOOGLE_APPLICATION_CREDENTIALS` to the path of the JSON file that contains your service account key. This variable only applies to your current shell session, so if you open a new session, set the variable again.

   Example: Linux or macOS

   export GOOGLE_APP_
   CREDENTIALS = '\-.X

   Example: Windows

8. Create a Cloud Storage bucket:
    1. In the Cloud Console, go to the Cloud Storage **Browser** page.
       Go to Browser
    2. Click **Create bucket**.
    3. On the **Create a bucket** page, enter your bucket information. To go to the next step, click **Continue**.
        - For **Name your bucket**, enter a unique bucket name. Don't include sensitive information in the bucket name, because the bucket namespace is global and publicly visible.
        - For **Choose where to store your data**, do the following:
            - Select a **Location type** option.
            - Select a **Location** option.
        - For **Choose a default storage class for your data**, select the following: **Standard**.
        - For **Choose how to control access to objects**, select an **Access control** option.
        - For **Advanced settings (optional)**, specify an encryption method, a retention policy, or bucket labels.
    4. Click **Create**.
9. Copy the Google Cloud project ID and the Cloud Storage bucket name. You need these values later in this document.

Set up your environment

Dataflow no longer supports pipelines using Python 2. For more information, see Python 2 support on Google Cloud page.
In this section, use the command prompt to set up an isolated Python virtual environment to run your pipeline project by using venv. This process lets you isolate the dependencies of one project from the dependencies of other projects.

If you don't have a command prompt readily available, you can use Cloud Shell. Cloud Shell already has the package manager for Python 3 installed, so you can skip to creating a virtual environment.

To install Python and then create a virtual environment, follow these steps:

1. Check that you have Python 3 and `pip` running in your system:

   ```
   python --version
   python -m pip --version
   ```

2. If required, install Python 3 and then set up a Python virtual environment: follow the instructions provided in the *Installing Python* and *Setting up venv* sections of the Setting up a Python development environment page.

Note: Cython is not required, but if it is installed, the version must be 0.28.1 or later. To check your Cython version, run `pip show cython`.

After you complete the quickstart, you can deactivate the virtual environment by running `deactivate` .

Get the Apache Beam SDK

The Apache Beam SDK is an open source programming model for data pipelines. You define a pipeline with an Apache Beam program and then choose a runner, such as Dataflow, to run your pipeline.

To download and install the Apache Beam SDK, follow these steps:

1. Verify that you are in the Python virtual environment that you created in the preceding section. Ensure that the prompt starts with `<env_name>` , where `env_name` is the name of the virtual environment.
2. Install the Python wheel packaging standard:

   ```
   pip install wheel
   ```

3. Install the latest version of the Apache Beam SDK for Python:

   ```
   pip install 'apache-beam[gcp]'
   ```

   Depending on the connection, your installation might take a while.

Run the pipeline locally

To see how a pipeline runs locally, use a ready-made Python module for the `wordcount` example that is included with the `apache_beam` package.

The `wordcount` pipeline example does the following:

1. Takes a text file as input.

   This text file is located in a Cloud Storage bucket with the resource name `gs://dataflow-samples/shakespeare/kinglear.txt` .

2. Parses each line into words.
3. Performs a frequency count on the tokenized words.

To stage the `wordcount` pipeline locally, follow these steps:

1. From your local terminal, run the `wordcount` example:

   ```
   python -m apache_beam.examples.wordcount \
     --output outputs
   ```

2. View the output of the pipeline:

   ```
   more outputs*
   ```

3. To exit, press `q`.

Running the pipeline locally lets you test and debug your Apache Beam program. You can view the `wordcount.py` source code on Apache Beam GitHub.

Run the pipeline on the Dataflow service

In this section, run the `wordcount` example pipeline from the `apache_beam` package on the Dataflow service. This example specifies `DataflowRunner` as the parameter for `--runner`.

Run the pipeline:

```
python -m apache_beam.examples.wordcount \
    --region DATAFLOW_REGION \
    --input gs://dataflow-samples/shakespeare/kinglear.txt \
    --output gs://STORAGE_BUCKET/results/outputs \
    --runner DataflowRunner \
    --project PROJECT_ID \
    --temp_location gs://STORAGE_BUCKET/tmp/
```

Replace the following:

- `DATAFLOW_REGION`: the region where your project is located—for example, `europe-west1`
  The `--region` flag overrides the default region that is set in the metadata server, your local client, or environment variables.
- `STORAGE_BUCKET`: the Cloud Storage name that you copied earlier
- `PROJECT_ID`: the Google Cloud project ID that you copied earlier

View your results

When you run a pipeline using Dataflow, your results are stored in a Cloud Storage bucket. In this section, verify that the pipeline is running by using either the Cloud Console or the local terminal.

To view your results in Cloud Console, follow these steps:

1. In the Cloud Console, go to the Dataflow **Jobs** page.
   Go to Jobs

   The **Jobs** page displays details of your `wordcount` job, including a status of **Running** at first, and then **Succeeded**.

2. Go to the Cloud Storage **Browser** page.
   Go to Browser

3. From the list of buckets in your project, click the storage bucket that you created earlier.

   In the `wordcount` directory, the output files that your job created are displayed.

Modify the pipeline code

The `wordcount` pipeline in the previous examples distinguishes between uppercase and lowercase words. The following steps show how to modify the pipeline so that the `wordcount` pipeline is not case-sensitive.

1. On your local machine, download the latest copy of the wordcount code from the Apache Beam GitHub repository.
2. From the local terminal, run the pipeline:

   ```
   python wordcount.py --output outputs
   ```

3. View the results:

   ```
   more outputs*
   ```

4. To exit, press `q`.
5. In an editor of your choice, open the `wordcount.py` file.
6. Inside the `run` function, examine the pipeline steps:

   ```
   counts = (
           lines
           | 'Split' >>
   (beam.ParDo(WordExtractingDoFn()).with_output_types(str))
           | 'PairWIthOne' >> beam.Map(lambda x: (x, 1))
           | 'GroupAndSum' >> beam.CombinePerKey(sum))
   ```

   After `split`, the lines are split into words as strings.

7. To lowercase the strings, modify the line after `split`:

   ```
   counts = (
           lines
           | 'Split' >>
   (beam.ParDo(WordExtractingDoFn()).with_output_types(str))
           | 'lowercase' >> beam.Map(str.lower)
           | 'PairWIthOne' >> beam.Map(lambda x: (x, 1))
           | 'GroupAndSum' >> beam.CombinePerKey(sum))
   ```

   This modification maps the `str.lower` function onto every word. This line is equivalent to `beam.Map(lambda word: str.lower(word))`.
8. Save the file and run the modified `wordcount` job:

   ```
   python wordcount.py --output outputs
   ```

9. View the results of the modified pipeline:

   ```
   more outputs*
   ```

10. To exit, press `q`.

Clean up

To avoid incurring charges to your Google Cloud account for the resources used in this quickstart, follow these steps.

1. In the Cloud Console, go to the Cloud Storage **Browser** page.
   Go to Browser
2. Click the checkbox for the bucket that you want to delete.
3. To delete the bucket, click delete **Delete**, and then follow the instructions.

Rate and review