Using Flex Templates

ature is covered by the Pre-GA Offerings Terms (/terms/service-terms#1) of the Google Cloud Platform Terms
e. Pre-GA features may have limited support, and changes to pre-GA features may not be compatible with othe
sions. For more information, see the launch stage descriptions (/products#product-launch-stages).

This tutorial shows you how to create and run a Dataflow Flex Template job with a custom Docker image using `gcloud` command-line tool. This tutorial walks you through a streaming pipeline example that reads JSON-encoded messages from Pub/Sub, transforms message data with Beam SQL, and writes the results to a BigQuery table.

Objectives

- Build a Docker container image.
- Create and run a Dataflow Flex Template.

Costs

This tutorial uses billable components of Google Cloud, including:

- Dataflow
- Pub/Sub
- Cloud Storage
- Cloud Scheduler
- App Engine
- Container Registry
- Cloud Build
- BigQuery

Use the Pricing Calculator (/products/calculator) to generate a cost estimate based on your projected usage.

Before you begin

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a 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 the project selector page (https://console.cloud.google.com/projectselector2/home/dashboar

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

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

   Enable the APIs (https://console.cloud.google.com/flows/enableapi?apiid=dataflow,compute_compo

5. Set up authentication:

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

      Go to the Create Service Account Key page (https://console.cloud.google.com/apis/credentia

   b. From the **Service account** list, select **New service account**.

   c. In the **Service account name** field, enter a name.

   d. From the **Role** list, select **Project** > **Owner**.

★ **Note**: The **Role** field authorizes your service account to access resources. You can view and change this field later by using the Cloud Console (https://console.cloud.google.com/). If you are developing a production app, specify more granular permissions than **Project > Owner**. For more information, see granting roles to service accounts (/iam/docs/granting-roles-to-service-accounts).

e. Click **Create**. A JSON file that contains your key downloads to your computer.

6. 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

Replace with the path of the JSON file that contains your service account key.

```
export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

For example:

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/service-account-fil
```

Example: Windows

Replace with the path of the JSON file that contains your service account key.

With PowerShell:

```
$env:GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

For example:

```
$env:GOOGLE_APPLICATION_CREDENTIALS="C:\Users\username\Downloads\my-key.json"
```

With command prompt:

```
set GOOGLE_APPLICATION_CREDENTIALS=[PATH]
```

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. See Cleaning up (/dataflow/docs/guides/templates/using-flex-templates#cleaning_up) for more detail.

Creating the example source and sink

This section explain how to create the following:

- A streaming source of data using Pub/Sub
- A dataset to load the data into BigQuery

Create a Cloud Storage bucket

Use the gsutil mb (/storage/docs/gsutil/commands/mb) command:

```
t BUCKET="my-storage-bucket"
l mb gs://$BUCKET
```

Create a Pub/Sub topic and a subscription to that topic

Use the `gcloud` command-line tool:

```
t TOPIC="messages"
t SUBSCRIPTION="ratings"

d pubsub topics create $TOPIC
d pubsub subscriptions create --topic $TOPIC $SUBSCRIPTION
```

Create a Cloud Scheduler job

In this step, we use the `gcloud` command-line tool to create and run a Cloud Scheduler job that publishes "positive ratings" and "negative ratings."

1. Create a Cloud Scheduler job for this Google Cloud project.

   ```
   gcloud scheduler jobs create pubsub positive-ratings-publisher \
     --schedule="* * * * *" \
     --topic="$TOPIC" \
     --message-body='{"url": "https://beam.apache.org/", "review": "positive"}'
   ```

   This creates and runs a publisher for "positive ratings" that publishes 1 message per minute.

2. Start the Cloud Scheduler job.

   ```
   gcloud scheduler jobs run positive-ratings-publisher
   ```

3. Create and run another similar publisher for "negative ratings" that publishes 1 message every 2 minutes.

   ```
   gcloud scheduler jobs create pubsub negative-ratings-publisher \
     --schedule="*/2 * * * *" \
     --topic="$TOPIC" \
     --message-body='{"url": "https://beam.apache.org/", "review": "negative"}'

   gcloud scheduler jobs run negative-ratings-publisher
   ```

Create a BigQuery dataset

Use the bq mk (/bigquery/docs/reference/bq-cli-reference#bq_mk) command:

```
t PROJECT="$(gcloud config get-value project)"
t DATASET="beam_samples"
t TABLE="streaming_beam_sql"
```

```
--dataset "$PROJECT:$DATASET"
```

Downloading the code sample

JavaPython (#python)

Clone the java-docs-samples repository (https://github.com/GoogleCloudPlatform/java-docs-samples) and navigate to the code sample for this tutorial.

```
git clone https://github.com/GoogleCloudPlatform/java-docs-samples.git
cd java-docs-samples/dataflow/flex-templates/streaming_beam_sql
```

Setting up your development environment

JavaPython (#python)

1. Download and install the Java Development Kit (JDK) (http://www.oracle.com/technetwork/java/javase/downloads/index.html) version 11. Verify that the JAVA_HOME (https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/envvars001.html) environment variable is set and points to your JDK installation.
2. Download and install Apache Maven (http://maven.apache.org/download.cgi) by following Maven's installation guide (http://maven.apache.org/install.html) for your specific operating system.
3. (Optional) Run the Apache Beam pipeline locally for development.

```
mvn compile exec:java \
  -Dexec.mainClass=org.apache.beam.samples.StreamingBeamSQL \
  -Dexec.args="\
    --project=$PROJECT \
```

```
    --inputSubscription=$SUBSCRIPTION \
    --outputTable=$PROJECT:$DATASET.$TABLE \
    --tempLocation=gs://$BUCKET/samples/dataflow/temp"
```

4. Build the Java project into an Uber JAR file.

```
mvn clean package
```

5. (Optional) Note the size of the Uber JAR file compared to the original file.

```
ls -lh target/*.jar
```

This Uber JAR file has all the dependencies embedded in it. You can run this file as a standalone application with no external dependencies on other libraries.

Creating and building a container image

1. (Optional) Enable Kaniko cache use by default.

```
gcloud config set builds/use_kaniko True
```

Kaniko (/cloud-build/docs/kaniko-cache) caches container build artifacts, so using this option speeds up subsequent builds.

2. (Optional) Create the Dockerfile. You can customize the Dockerfile from this tutorial. The starter file looks like the following:

JavaPython (#python)

```
FROM gcr.io/dataflow-templates-base/java11-template-launcher-base:latest

# Define the Java command options required by Dataflow Flex Templates.
ENV FLEX_TEMPLATE_JAVA_MAIN_CLASS="org.apache.beam.samples.StreamingBeamSQL"
```

```
ENV FLEX_TEMPLATE_JAVA_CLASSPATH="/template/pipeline.jar"

# Make sure to package as an uber-jar including all dependencies.
COPY target/streaming-beam-sql-1.0.jar ${FLEX_TEMPLATE_JAVA_CLASSPATH}
```

This Dockerfile contains the `FROM`, `ENV`, and `COPY` commands, which you can read about in the Dockerfile reference (https://docs.docker.com/engine/reference/builder/).

Images starting with `gcr.io/PROJECT/` are saved into your project's Container Registry, where the image is accessible to other Google Cloud products.

3. Build the Docker (https://docs.docker.com/) image using a `Dockerfile` with Cloud Build (/cloud-build/docs/quickstart-build#build_using_dockerfile).

★ **Note:** If you created your own Dockerfile from the previous step, you must update the `TEMPLATE_IMAGE` in the following code sample.

```
export TEMPLATE_IMAGE="gcr.io/$PROJECT/samples/dataflow/streaming-beam-sql:late
gcloud builds submit --tag $TEMPLATE_IMAGE .
```

Creating a Flex Template

To run a template, you need to create a template spec file in a Cloud Storage containing all of the necessary information to run the job, such as the SDK information and metadata.

The metadata.json (https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/dataflow/flex-templates/streaming_beam_sql/metadata.json) file in this example contains additional information for the template such as the `name`, `description`, and input `parameters` fields.

1. Create a template spec file containing all of the information necessary to run the job, such as the SDK information and metadata.

```
export TEMPLATE_PATH="gs://$BUCKET/samples/dataflow/templates/streaming-beam-sq
```

2. Build the Flex Template.

JavaPython (#python)

```
gcloud beta dataflow flex-template build $TEMPLATE_PATH \
  --image "$TEMPLATE_IMAGE" \
  --sdk-language "JAVA" \
  --metadata-file "metadata.json"
```

The template is now available through the template file in the Cloud Storage location that you specified.

Running a Flex Template pipeline

You can now run the Apache Beam pipeline in Dataflow by referring to the template file and passing the template parameters (/dataflow/docs/guides/specifying-exec-params#setting-other-cloud-dataflow-pipeline-options) required by the pipeline.

All Flex Template Python jobs must use the **disable_flex_template_entrypoint_overrride** experiment ore information, see Turning off Docker container entrypoint overrides aflow/docs/guides/templates/troubleshooting-flex-templates#override-python).

1. Run the template.

JavaPython (#python)

```
export REGION="us-central1"

gcloud beta dataflow flex-template run "streaming-beam-sql-`date +%Y%m%d-%H%M
  --template-file-gcs-location "$TEMPLATE_PATH" \
  --parameters inputSubscription="$SUBSCRIPTION" \
  --parameters outputTable="$PROJECT:$DATASET.$TABLE" \
  --region "$REGION"
```

Alternatively, run the template with a REST API request.

```
curl -X POST \
  "https://dataflow.googleapis.com/v1b3/projects/$PROJECT/locations/us-central1
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -d '{
    "launch_parameter": {
      "jobName": "streaming-beam-sql-'$(date +%Y%m%d-%H%M%S)'",
      "parameters": {
        "inputSubscription": "'$SUBSCRIPTION'",
        "outputTable": "'$PROJECT:$DATASET.$TABLE'"
      },
      "containerSpecGcsPath": "'$TEMPLATE_PATH'"
    }
  }'
```

After you execute the command to run the Flex Template, the Dataflow returns a Job ID with the job status **Queued**. It might take several minutes before the job status reaches **Running** and you can access the job graph.

2. Check the results in BigQuery by running the following query:

```
bq query --use_legacy_sql=false 'SELECT * FROM `'"$PROJECT.$DATASET.$TABLE"'`'
```

While this pipeline is running, you can see new rows appended into the BigQuery table every minute.

Cleaning up

After you've finished this tutorial, you can clean up the resources you created on Google Cloud so you won't be billed for them in the future. The following sections describe how to delete or turn off these resources.

Clean up the Flex Template resources

1. Stop the Dataflow pipeline.

```
gcloud dataflow jobs list \
  --filter 'NAME:streaming-beam-sql AND STATE=Running' \
  --format 'value(JOB_ID)' \
  --region "$REGION" \
  | xargs gcloud dataflow jobs cancel --region "$REGION"
```

2. Delete the template spec file from Cloud Storage.

```
gsutil rm $TEMPLATE_PATH
```

3. Delete the Flex Template container image from Container Registry.

```
gcloud container images delete $TEMPLATE_IMAGE --force-delete-tags
```

Clean up Google Cloud project resources

1. Delete the Cloud Scheduler jobs.

```
gcloud scheduler jobs delete negative-ratings-publisher
gcloud scheduler jobs delete positive-ratings-publisher
```

2. Delete the Pub/Sub subscription and topic.

```
gcloud pubsub subscriptions delete $SUBSCRIPTION
gcloud pubsub topics delete $TOPIC
```

3. Delete the BigQuery table.

```
bq rm -f -t $PROJECT:$DATASET.$TABLE
```

4. Delete the BigQuery dataset, this alone does not incur any charges.

The following command also deletes all tables in the dataset. The tables and data cannot be recovered.

```
bq rm -r -f -d $PROJECT:$DATASET
```

5. Delete the Cloud Storage bucket, this alone does not incur any charges.

The following command also deletes all objects in the bucket. These objects cannot be recovered.

```
gsutil rm -r gs://$BUCKET
```

Limitations

The following limitations apply to Flex Templates jobs:

- You must use a Google-provided base image to package your containers using Docker.
- Updating streaming jobs is not supported.
- Using FlexRS is not supported.

What's next

- For more information, read Troubleshooting Flex Templates (/dataflow/docs/guides/templates/troubleshooting-flex-templates).
- Try out other Dataflow features for yourself. Have a look at our tutorials (/docs/tutorials).

Last updated 2020-07-30 UTC.