Dataflow templatesbookmark_border

cloud.google.com/dataflow/docs/concepts/dataflow-templates

Dataflow templates allow you to stage your pipelines on Google Cloud and run them using the Google Cloud Console, the `gcloud` command-line tool, or REST API calls. Classic templates are staged as execution graphs on Cloud Storage while Flex Templates package the pipeline as a Docker image and stage these images on your project's Container Registry. You can use one of the Google-provided templates or create your own.

Templates provide you with additional benefits compared to non-templated Dataflow deployment, such as:

- You can run your pipelines without the development environment and associated dependencies that are common with non-templated deployment. This is useful for scheduling recurring batch jobs.
- Templates separate the pipeline construction (performed by developers) from the running of the pipeline. Hence, there's no need to recompile the code every time the pipeline is run.
- Runtime parameters allow you to customize the running of the pipeline.
- Non-technical users can run templates with the Google Cloud Console, `gcloud` command-line tool, or the REST API.

Comparing non-templated and templated jobs

Dataflow templates introduce a new development and execution workflow that differs from non-templated job execution workflow. The template workflow separates the development step from the execution step.

Dataflow jobs without templates

Apache Beam pipeline development and job execution typically all happen within a development environment.

Typical workflow for non-templated Dataflow jobs:

1. Developers create a development environment and develop their pipeline. The environment includes the Apache Beam SDK and other dependencies.
2. Users run the pipeline from the development environment. The Apache Beam SDK stages files in Cloud Storage, creates a job request file, and submits the file to the Dataflow service.

Templated Dataflow jobs

If you use classic templates or Flex Templates, staging and execution are separate steps. This separation gives you additional flexibility to decide who can run jobs and where the jobs are run from.

Typical workflow for templated Dataflow jobs:

1. Developers create a development environment and develop their pipeline. The environment includes the Apache Beam SDK and other dependencies.
2. This step depends on which type of template you use.
    - **Classic templates.** Developers run the pipeline and create a template. The Apache Beam SDK stages files in Cloud Storage, creates a template file (similar to job request), and saves the template file in Cloud Storage.
    - **Flex Templates.** Developers package the pipeline into a Docker image and then use the `gcloud` command-line tool to build and save the Flex Template spec file in Cloud Storage.
3. Other users can easily run jobs with the Cloud Console, `gcloud` command-line tool, or the REST API to submit template file execution requests to the Dataflow service.

In addition, you can clone classic template jobs through the Dataflow monitoring user interface. The submission form for cloned jobs has parameter values pre-populated; however, they can be modified before you run the job.

To clone a job, perform the following steps:

1. Navigate to the **job details** page for the job you want to clone.
2. Click content_copy**Clone**.
3. If needed, update the job parameters.
4. Click **Run Job** to submit the new job.

Note: Use template metadata to keep track of custom parameters.
To understand the different stages of pipeline construction and running, with or without using a classic template or Flex template, see Turn any Dataflow pipeline into a reusable template.

Evaluating which template type to use

Flex Templates bring more flexibility over classic templates by allowing minor variations of Dataflow jobs to be launched from a single template and allowing the use of any source or sink I/O. For classic templates, the execution graph is built during the template creation process. The execution graph for Flex Templates is dynamically built based on runtime parameters provided by the user when the template is executed. This means that when you use Flex Templates, you can make minor variations to accomplish different tasks with the same underlying template, such as changing the source or sink file formats.

Comparing templated jobs

The following table summarizes the similarities and differences between classic and Flex templates:

Feature	Classic templates	Flex Templates
Separate staging and execution steps	Yes	Yes
Run the template using the Google Cloud console, `gcloud` tool, or REST API calls	Yes	Yes
Run pipeline without recompiling code	Yes	Yes
Run pipeline without development environment and associated dependencies	Yes	Yes
Customize pipeline execution with runtime parameters	Yes	Yes
Run validations upon job graph construction to reduce runtime errors	No	Yes
Can change job execution graph after the template is created	No	Yes
Can update streaming jobs[1]	Yes	Yes
Supports FlexRS[1]	Yes	Yes
Supports SQL parameters	No	Yes
Supports I/O interfaces beyond `ValueProvider`	No	Yes

Notes:

1. To use the streaming update or FlexRS feature, users must rebuild their FlexRS templates with base image version 20210120_RC00.
For both classic templates and Flex Template jobs, staging and execution are performed in different steps. However, the two types of templated jobs result in different staging artifacts. When you use classic templates, the staged artifact is the template with the JSON serialized execution graph. With Flex templates, the staged artifact is a Docker image containing the JAR file or Python code.

SDK version requirements

To create your own templates, make sure your Apache Beam SDK version supports template creation.

To create templates with the Apache Beam SDK 2.x for Java, you must have version 2.0.0-beta3 or higher.

To run templates with `gcloud` command-line tool, you must have Cloud SDK version 138.0.0 or higher.

Rate and review