

# Running classic templates

[Send feedback](#)

After you create and stage your Dataflow template, run the template with the Google Cloud Console, REST API, or `gcloud` command-line tool. You can deploy Dataflow template jobs from many environments, including App Engine standard environment, Cloud Functions, and other constrained environments.

★ **Note:** In addition to the template file, running templated pipeline relies on files that we staged and referenced at the time of template creation. If you move or remove the staged files, your pipeline job will fail.

## Using the Cloud Console

You can use the Cloud Console to run Google-provided and custom Dataflow templates.

### Google-provided templates

To run a Google-provided template:

1. Go to the Dataflow page in the Cloud Console.

[Go to the Dataflow page](#)

2. Click **+ CREATE JOB FROM TEMPLATE**.



3. Select the Google-provided template that you want to run from the Dataflow template drop-down menu.



← Create job from template

Job name \*  
job-custom  
Must be unique among running jobs

Regional endpoint \*  
us-central1  
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. [Learn more](#)

Dataflow template \*  
Custom Template  
Execute a custom template that you've uploaded to Cloud Storage

Template path \*  
gs://[redacted]/custom BROWSE  
Path to your template file stored in Cloud Storage

An example pipeline that uses custom template.

Required parameters

Temporary location \*  
gs://my-bucket/temp  
Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp

Encryption

☒ Google-managed key  
No configuration required

☐ Customer-managed key  
Manage via Google Cloud Key Management Service

▼ SHOW OPTIONAL PARAMETERS

Additional parameters ⓘ

Name *	Value *
num_to_find	42

+ ADD PARAMETER

RUN JOB

4. Enter a job name in the **Job Name** field.
5. Enter the Cloud Storage path to your template file in the template Cloud Storage path field.
6. If your template needs parameters, click on **+ ADD PARAMETER** in the **Additional parameters** section. Enter in the **Name** and **Value** of the parameter. Repeat this step for each needed parameter.
7. Click **Run Job**.

## Using the REST API

To run a template with a [REST API request](#), send an HTTP POST request with your project ID. This request requires [authorization](#).

See the REST API reference for [projects.templates.launch](#) to learn more about the available parameters.

### Example 1: Creating a custom template batch job

This example [projects.templates.launch](#) request creates a batch job from a template that reads a text file and writes an output text file. If the request is successful, the response body contains an instance of [LaunchTemplateResponse](#).

You must modify the following values:

- Replace `YOUR_PROJECT_ID` with your project ID.
- Replace `LOCATION` with the Dataflow regional endpoint of your choice.
- Replace `JOB_NAME` with a job name of your choice.
- Replace `YOUR_BUCKET_NAME` with the name of your Cloud Storage bucket.
- Set `gcsPath` to the Cloud Storage location of the template file.
- Set `parameters` to your list of key/value pairs.
- Set `tempLocation` to a location where you have write permission. This value is required to run Google-provided templates.

```
POST https://dataflow.googleapis.com/v1b3/projects/YOUR_PROJECT_ID/locations/LOCATION/jobs
{
  "jobName": "JOB_NAME",
  "parameters": {
    "inputFile": "gs://YOUR_BUCKET_NAME/input/my_input.txt",
    "outputFile": "gs://YOUR_BUCKET_NAME/output/my_output"
  },
  "environment": {
    "tempLocation": "gs://YOUR_BUCKET_NAME/temp",
    "zone": "us-central1-f"
  }
}
```

## Example 2: Creating a custom template streaming job

This example `projects.templates.launch` request creates a streaming job from a template that reads from a Pub/Sub topic and writes to a BigQuery table. The BigQuery table must already exist with the appropriate schema. If successful, the response body contains an instance of `LaunchTemplateResponse`.

You must modify the following values:

- Replace `YOUR_PROJECT_ID` with your project ID.
- Replace `LOCATION` with the Dataflow regional endpoint of your choice.
- Replace `JOB_NAME` with a job name of your choice.
- Replace `YOUR_BUCKET_NAME` with the name of your Cloud Storage bucket.
- Replace `YOUR_TOPIC_NAME` with your Pub/Sub topic name.
- Replace `YOUR_DATASET` with your BigQuery dataset, and replace `YOUR_TABLE_NAME` with your BigQuery table name.
- Set `gcsPath` to the Cloud Storage location of the template file.
- Set `parameters` to your list of key/value pairs.
- Set `tempLocation` to a location where you have write permission. This value is required to run Google-provided templates.

```
POST https://dataflow.googleapis.com/v1b3/projects/YOUR_PROJECT_ID/locations/LOCATION/jobs
{
  "jobName": "JOB_NAME",
  "parameters": {
    "topic": "projects/YOUR_PROJECT_ID/topics/YOUR_TOPIC_NAME",
    "table": "YOUR_PROJECT_ID:YOUR_DATASET.YOUR_TABLE_NAME"
  },
  "environment": {
    "tempLocation": "gs://YOUR_BUCKET_NAME/temp",
    "zone": "us-central1-f"
  }
}
```

### Example 3: Updating a custom template streaming job

This example `projects.templates.launch` request shows you how to update a template streaming job.

1. Run [Example 2: Creating a custom template streaming job](#) to start a streaming templ: job.
2. Send the following `HTTP POST` request, with the following modified values:
  - Replace `YOUR_PROJECT_ID` with your project ID.
  - Replace `LOCATION` with the Dataflow [regional endpoint](#) of your choice.
  - Replace `JOB_NAME` with a job name of your choice.
  - Replace `YOUR_BUCKET_NAME` with the name of your Cloud Storage bucket.
  - Replace `YOUR_TOPIC_NAME` with your Pub/Sub topic name.
  - Replace `YOUR_DATASET` with your BigQuery dataset, and replace `YOUR_TABLE_NAME` with your BigQuery table name.
  - Set `gsPath` to the Cloud Storage location of the template file.
  - Set `parameters` to your list of key/value pairs.
  - Set `tempLocation` to a location where you have write permission. This value is required to run Google-provided templates.

```
POST https://dataflow.googleapis.com/v1b3/projects/YOUR_PROJECT_ID
{
  "jobName": "JOB_NAME",
  "parameters": {
    "topic": "projects/YOUR_PROJECT_ID/topics/YOUR_TOPIC_NAME",
    "table": "YOUR_PROJECT_ID:YOUR_DATASET.YOUR_TABLE_NAME"
  },
  "environment": {
    "tempLocation": "gs://YOUR_BUCKET_NAME/temp",
    "zone": "us-central1-f"
  }
  "update": true
}
```

3. Access the [Dataflow monitoring interface](#) and verify that a new job with the same name was created. This job has the status **Updated**.

### Using the Google API Client Libraries

Consider using the [Google API Client Libraries](#) to easily make calls to the [Dataflow REST API](#). This sample script uses the [Google API Client Library for Python](#).

In this example, you must set the following variables:

- `project`: Set to your project ID.
- `job`: Set to a unique job name of your choice.
- `template`: Set to the Cloud Storage location of the template file.
- `parameters`: Set to a dictionary with the template parameters.

`dataflow/run_template/main.py`

[View on GitHub](#)

[Feedback](#)

```
from googleapiclient.discovery import build
```

```
# project = 'your-gcp-project'
# job = 'unique-job-name'
# template = 'gs://dataflow-templates/latest/Word_Count'
# parameters = {
#     'inputFile': 'gs://dataflow-samples/shakespeare/kinglear.txt',
#     'output': 'gs://<your-gcs-bucket>/wordcount/outputs',
# }

dataflow = build('dataflow', 'v1b3')
request = dataflow.projects().templates().launch(
    projectId=project,
    gcsPath=template,
    body={
        'jobName': job,
        'parameters': parameters,
    }
)

response = request.execute()
```

For more information on the available options, see the `projects.templates.launch` method in the [Cloud Dataflow REST API reference](#).

## Using gcloud

★ **Note:** To use the `gcloud` command-line tool to run templates, you must have [Cloud SDK](#) version 138.0.0 or higher.

The `gcloud` command-line tool can run either a custom or a [Google-provided](#) template using the `gcloud dataflow jobs run` command. Examples of running Google-provided templates are documented in the [Google-provided templates page](#).

For the following custom template examples, set the following values:

- Replace `JOB_NAME` with a job name of your choice.
- Replace `YOUR_BUCKET_NAME` with the name of your Cloud Storage bucket.
- You must include the `--gcs-location` flag. Set `--gcs-location` to the Cloud Storage location of the template file.
- Set `--parameters` to the comma-separated list of parameters to pass to the job. Spaces between commas and values are not allowed.

### Example 1: Custom template, batch job

This example creates a batch job from a template that reads a text file and writes an output text file.

```
gcloud dataflow jobs run JOB_NAME \
  --gcs-location gs://YOUR_BUCKET_NAME/templates/MyTemplate \
  --parameters inputFile=gs://YOUR_BUCKET_NAME/input/my_input.txt,ou
```

The request returns a response with the following format.

```
id: 2016-10-11_17_10_59-1234530157620696789
projectId: YOUR_PROJECT_ID
type: JOB_TYPE_BATCH
```

### Example 2: Custom template, streaming job

This example creates a streaming job from a template that reads from a Pub/Sub topic and writes to a BigQuery table. The BigQuery table must already exist with the appropriate schema.

```
gcloud dataflow jobs run JOB_NAME \
  --gcs-location gs://YOUR_BUCKET_NAME/templates/MyTemplate \
  --parameters topic=projects/project-identifier/topics/resource-name
```

The request returns a response with the following format.

```
id: 2016-10-11_17_10_59-1234530157620696789
projectId: YOUR_PROJECT_ID
type: JOB_TYPE_STREAMING
```

For a complete list of flags for the `gcloud dataflow jobs run` command, see the [gcloud tool reference](#).

## Monitoring and Troubleshooting

The [Dataflow Monitoring Interface](#) allows you to monitor your Dataflow jobs. If a job fails, you can find troubleshooting tips, debugging strategies, and a catalog of common errors in the [Troubleshooting Your Pipeline](#) guide.

Rate and review



Send feedback