Configuring Flex Templatesbookmark_border

cloud.google.com/dataflow/docs/guides/templates/configuring-flex-templates

This page provides information about permissions, required Dockerfile environment variables, and supported pipeline options for Dataflow Flex Templates.

To configure a sample Flex Template, see the Flex Template tutorial.

Understanding Flex Template permissions

When you're working with Flex Templates, there are three sets of permissions to be aware of:

- Permissions to create resources
- Permissions to build a Flex Template
- Permissions to run a Flex Template

Permissions to create resources

To develop and run a Flex Template pipeline, you need to create various resources (for example, a staging bucket). For one-time resource creation tasks, you can use the basic Owner role.

Permissions to build a Flex Template

As the developer of a Flex Template, you need to build the template to make it available to users. Building involves uploading a template spec to a Cloud Storage bucket and provisioning a Docker image with the code and dependencies needed to run the pipeline. To build a Flex Template, you need read and write access to Cloud Storage and read and write access to Container Registry. You can grant these permissions by assigning the following role:

Storage Admin (`roles/storage.admin`)

Permissions to run a Flex Template

When you run a Flex Template, Dataflow creates a job for you. To create the job, the Dataflow service account needs the following permission:

dataflow.serviceAgent

When you first use Dataflow, the service assigns this role for you, so there's no action required to grant the above permission.

By default, the Compute Engine service account is used for launcher VMs and worker VMs. The service account needs the following roles and abilities:

- **Storage Object Admin (** `roles/storage.objectAdmin` **)**
- **Viewer (** `roles/viewer` **)**
- **Dataflow Worker (** `roles/dataflow.worker` **)**
- Read and write access to the staging bucket
- Read access to the Flex Template image

To grant read and write access to the staging bucket, you can use the role **Storage Object Admin (** `roles/storage.objectAdmin` **)**. For more information, see IAM roles for Cloud Storage.

To grant read access to the Flex Template image, you can use the role **Storage Object Viewer (** `roles/storage.objectViewer` **)**. For more information, see Configuring access control.

Setting required Dockerfile environment variables

If you want to create your own Docker file for a Flex Template job, you must specify the following environment variables:

Java Python

You must specify `FLEX_TEMPLATE_JAVA_MAIN_CLASS` and `FLEX_TEMPLATE_JAVA_CLASSPATH` in your Dockerfile.

Changing the base image

You can use a Google-provided base image to package your containers using Docker. Choose the most recent version name from the Flex Templates base images reference. Do not select `latest` .

Specify the base image in the following format:

```
gcr.io/dataflow-templates-base/IMAGE_NAME:VERSION_NAME
```

Replace the following:

- *`IMAGE_NAME`* : a Google-provided base image
- *`VERSION_NAME`* : a version name for the base image, found in the Flex Templates base images reference

Using custom containers

You can also build your own custom container to package a flex template. Choose the most recent version name from the Flex Templates launcher images reference and copy the launcher binaries into your custom containers. The binaries are built for Debian GNU/Linux operating Systems. Do not select `latest` .

Create a new Dockerfile, specifying the flex template launcher image as parent and add any customizations. For more information on writing Dockerfiles, see Best practices for writing Dockerfiles.

```
FROM gcr.io/dataflow-templates-base/IMAGE_NAME:VERSION_NAME as template_launcher
FROM USER_CUSTOM_IMAGECOPY --from=template_launcher
/opt/google/dataflow/python_template_launcher /opt/google/dataflow/

python_template_launcher

ARG WORKDIR=/dataflow/template
RUN mkdir -p ${WORKDIR}
WORKDIR ${WORKDIR}COPY streaming_beam.py .ENV
FLEX_TEMPLATE_PYTHON_PY_FILE="${WORKDIR}/streaming_beam.py"ENTRYPOINT
["/opt/google/dataflow/python_template_launcher"]
```

Replace the following:

- `IMAGE_NAME` : a Google-provided launcher image
- `VERSION_NAME` : a version name for the launcher image, found in the Flex Templates launcher images reference
- `USER_CUSTOM_IMAGE` : your custom container image

Specifying pipeline options

For information on pipeline options that are directly supported by Flex Templates, read Pipeline options.

You can also use any Apache Beam pipeline options indirectly. If you're using a `metadata.json` file for your Flex Template job, include these pipeline options in the file. This metadata file must follow the format in `TemplateMetadata`. For an example of a `metadata.json` file, view the streaming SQL Flex Template sample.

Otherwise, when you launch the Flex Template job, pass these pipeline options using the parameters field.

APIgcloud
Include pipeline options by using the `parameters` field.

What's next

- To know more about Classic and Flex Templates and their use-case scenarios, see Dataflow templates.
- For Flex Templates troubleshooting information, see Common error guidance.
- Explore reference architectures, diagrams, tutorials, and best practices about Google Cloud. Take a look at our Cloud Architecture Center.

Rate and review