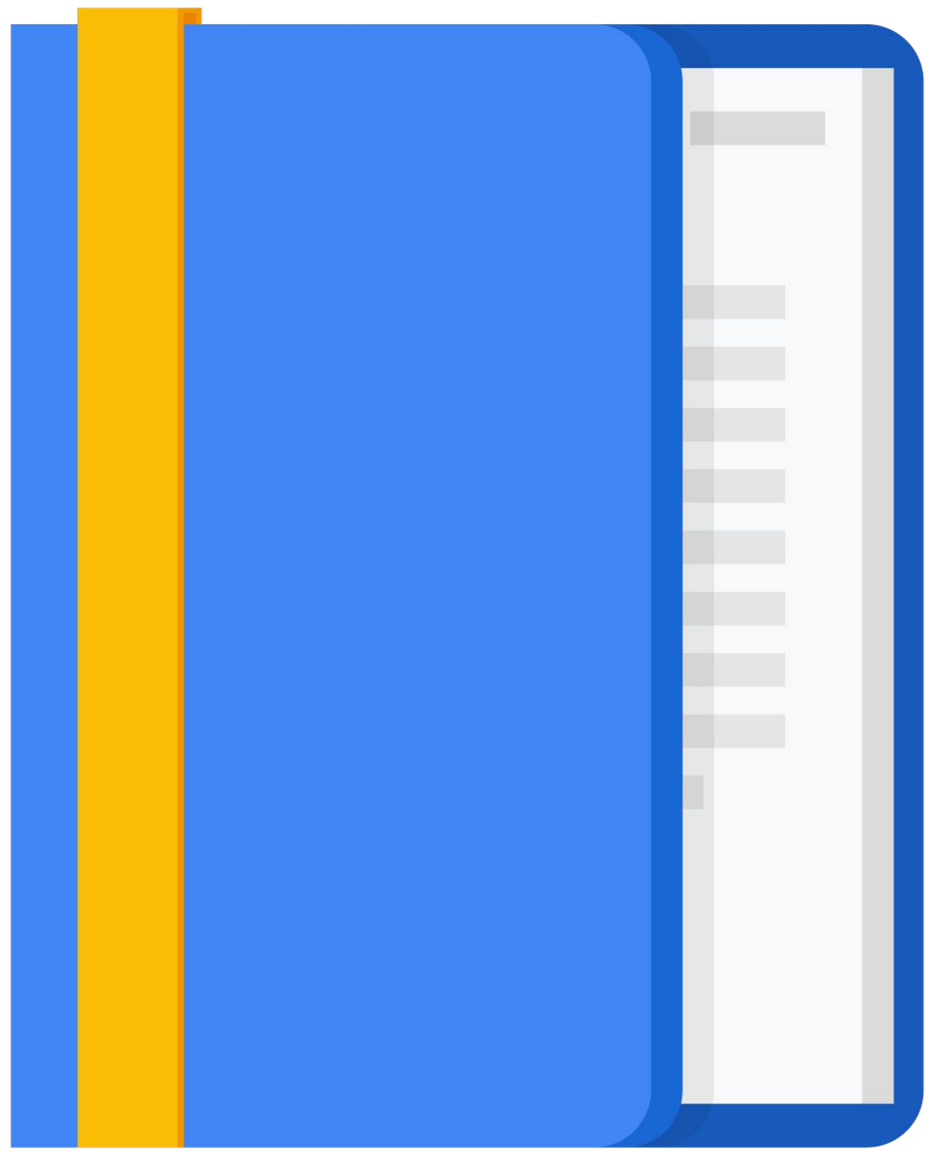
Flex Templates

Prathap Reddy

Cloud Data Engineer,
Google Cloud

Agenda



Flex Templates

Agenda

- Classic templates
- Flex templates
- Using flex templates
- Google-provided templates

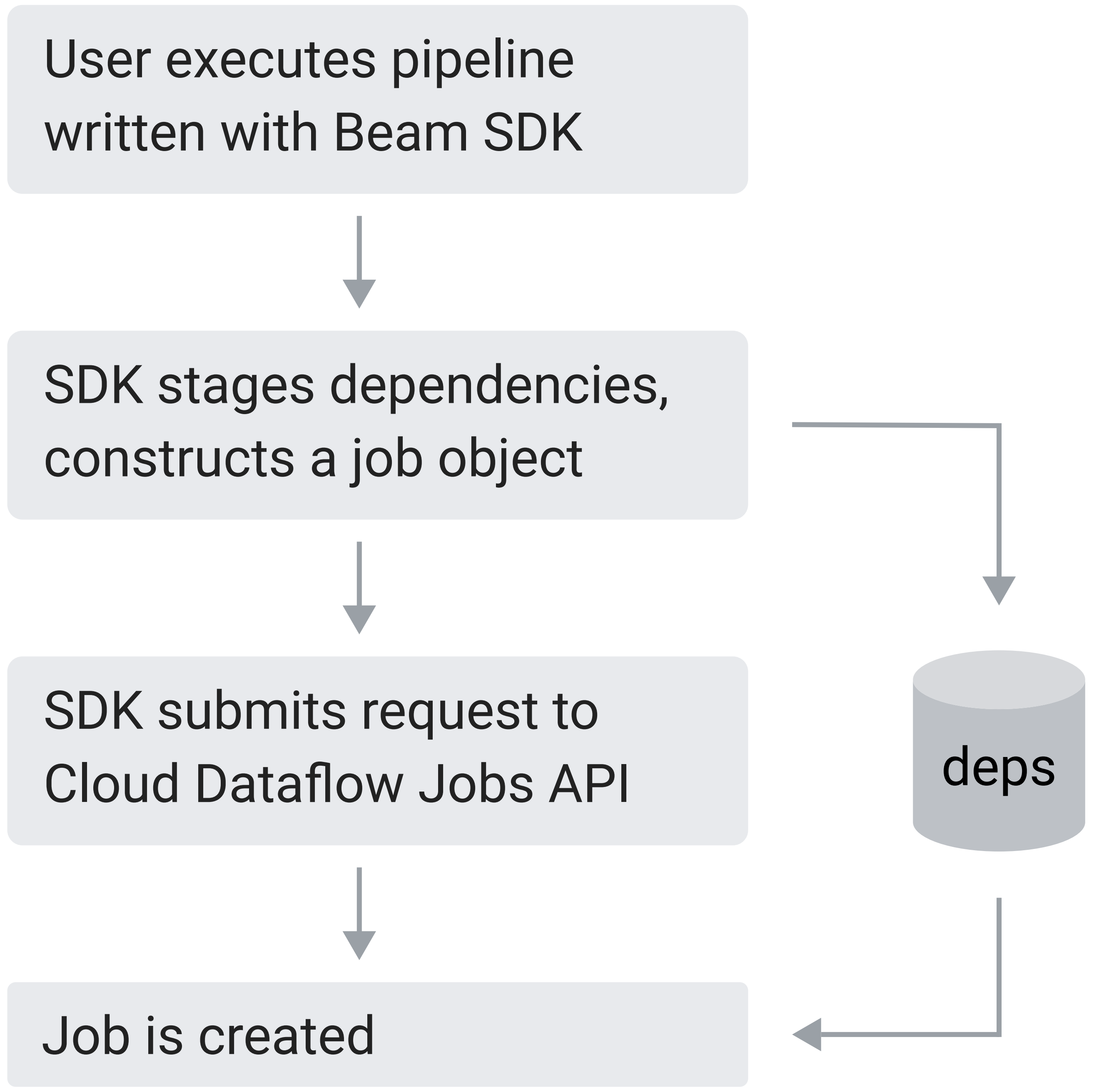
Google Cloud

Flex Templates

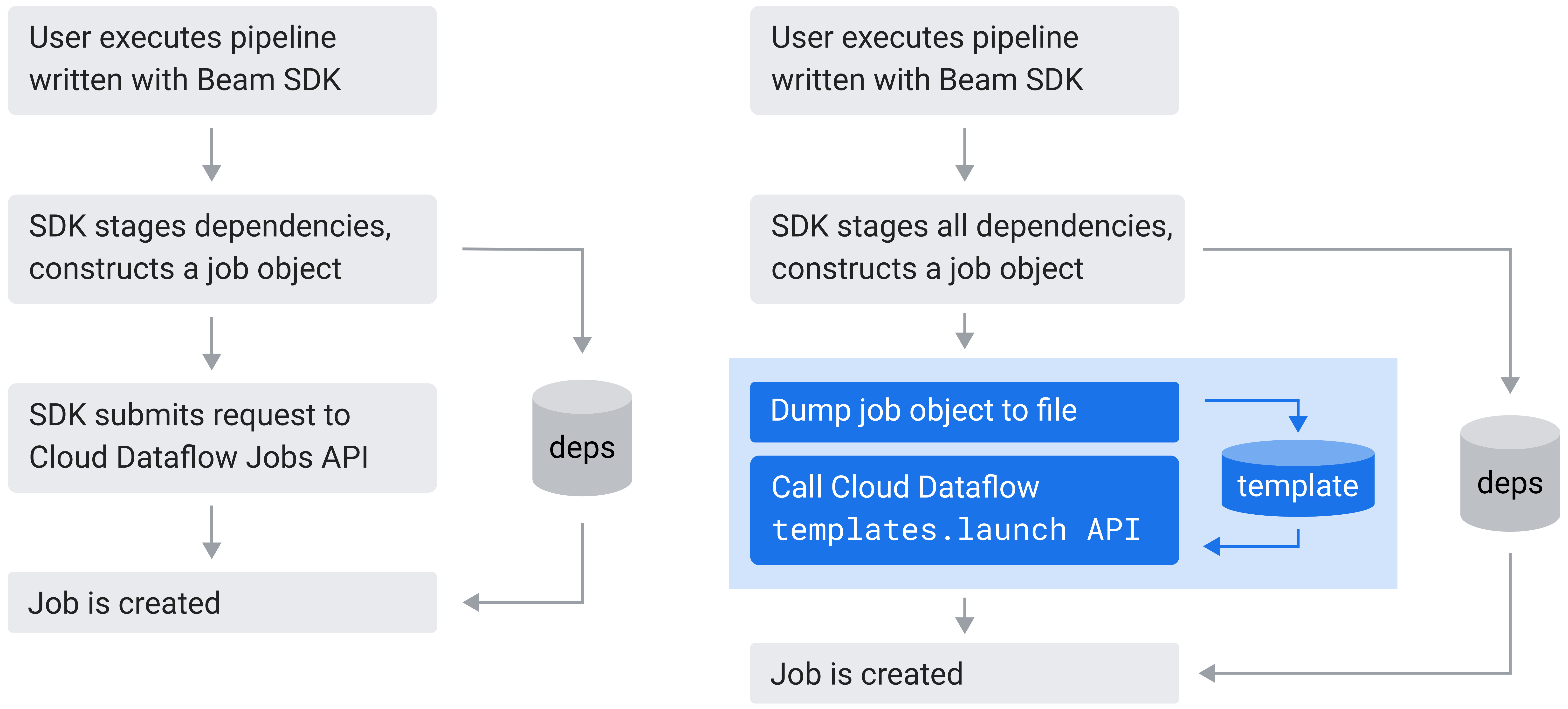
Agenda

- Classic templates
- Flex templates
- Using flex templates
- Google-provided templates

Dataflow job


User executes pipeline written with Beam SDK
SDK stages dependencies, constructs a job object
SDK submits request to Cloud Dataflow Jobs API
Job is created
deps

Classic templates


User executes pipeline written with Beam SDK
SDK stages dependencies, constructs a job object
SDK submits request to Cloud Dataflow Jobs API
Job is created
deps
User executes pipeline written with Beam SDK
SDK stages all dependencies, constructs a job object
Dump job object to file
Call Cloud Dataflow templates.launch API
template
deps
Job is created

Google Cloud

Classic templates challenges

1 ValueProvider support for Beam I/O transforms

Classic templates challenges

1 ValueProvider support for Beam I/O transforms

2 Lack of support for dynamic DAG (Direct Acyclic Graph)

ValueProvider support

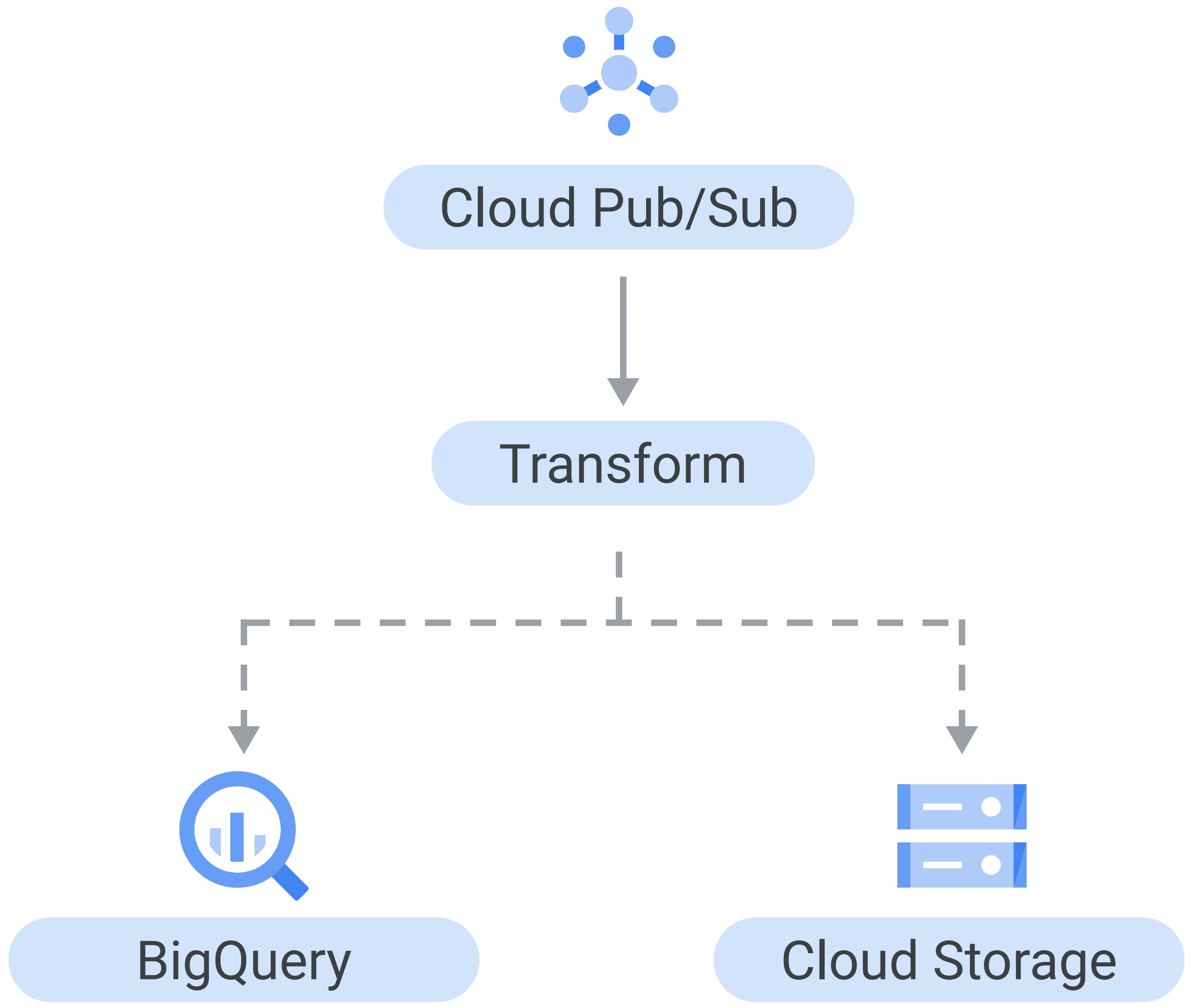
Use a ValueProvider type for all your runtime options

```
public interface WordCountOptions extends
PipelineOptions {
@Description("Path of the file to read from")
@Default.String("gs://dataflow-samples/kinglear.txt")
   String getInputFile();
   void setInputFile(String value);
}

public static void main(String[] args) {
   WordCountOptions options = //Create options
   Pipeline pipeline = Pipeline.create(options);
   pipeline.apply("ReadLines",
TextIO.read().from(options.getInputFile()));
}
```

```
public interface WordCountOptions extends
PipelineOptions {
@Description("Path of the file to read from")
@Default.String("gs://dataflow-samples/kinglear.txt")
   ValueProvider<String> getInputFile();
   void setInputFile(ValueProvider<String> value);
}

public static void main(String[] args) {
   WordCountOptions options = //Create options
   Pipeline pipeline = Pipeline.create(options);
   pipeline.apply("ReadLines",
TextIO.read().from(options.getInputFile()));
}
```

Google Cloud

Lack of support for dynamic DAG



Flex Templates

Agenda

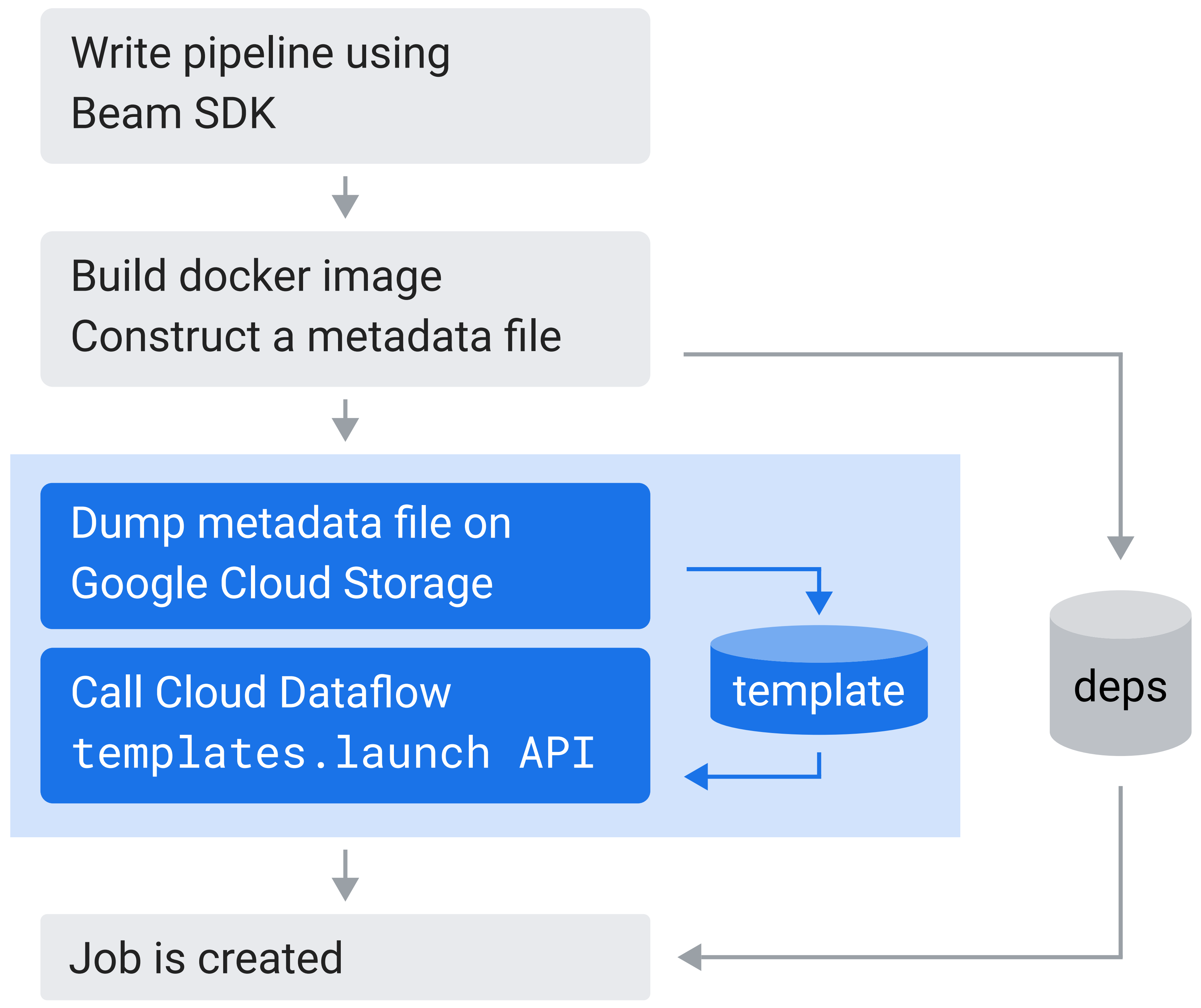
Classic templates

Flex templates

Using flex templates

Google-provided templates

Flex templates overview


Write pipeline using Beam SDK
Build docker image
Construct a metadata file
Dump metadata file on Google Cloud Storage
Call Cloud Dataflow templates.launch API
template
deps
Job is created

Flex Templates

Agenda

Classic templates

Flex templates

Using flex templates

Google provided templates

Google Cloud

Creating a flex template

✓ **1** Create a metadata file

Creating a flex template

✓ 1 Create a metadata file

✓ 2 Run the flex-template build gcloud command

Google Cloud

Create a metadata file

```
{
  "name": "PubSub To Bigquery",
  "description": "An Apache Beam streaming pipeline that reads JSON
      encoded messages from Pub/Sub,and writes the results to a BigQuery",
  "parameters": [
    {
      "name": "inputSubscription",
      "label": "Pub/Sub input subscription.",
      "helpText": "Pub/Sub subscription to read from.",
      "regexes": ["[a-zA-Z][-_.~+%a-zA-Z0-9]{2,}"]
    },
    {
      "name": "outputTable",
      "label": "BigQuery output table",
      "helpText": "BigQuery table spec to write to, in the form
                  'project:dataset.table'.",
      "regexes": [ "[^:]+:[^.]+[.].+"]
    }
  ]
}
```

Build the flex template

```
gcloud dataflow flex-template build "$TEMPLATE_SPEC_PATH" \
      --image-gcr-path "$TEMPLATE_IMAGE" \
      --sdk-language "JAVA" \
      --flex-template-base-image JAVA8 \
      --metadata-file "metadata.json" \
      --jar "target/pubsub-bigquery-1.0.jar" \
      --env
FLEX_TEMPLATE_JAVA_MAIN_CLASS="com.google.cloud.PubSubBigquery"
```

Launching the flex template

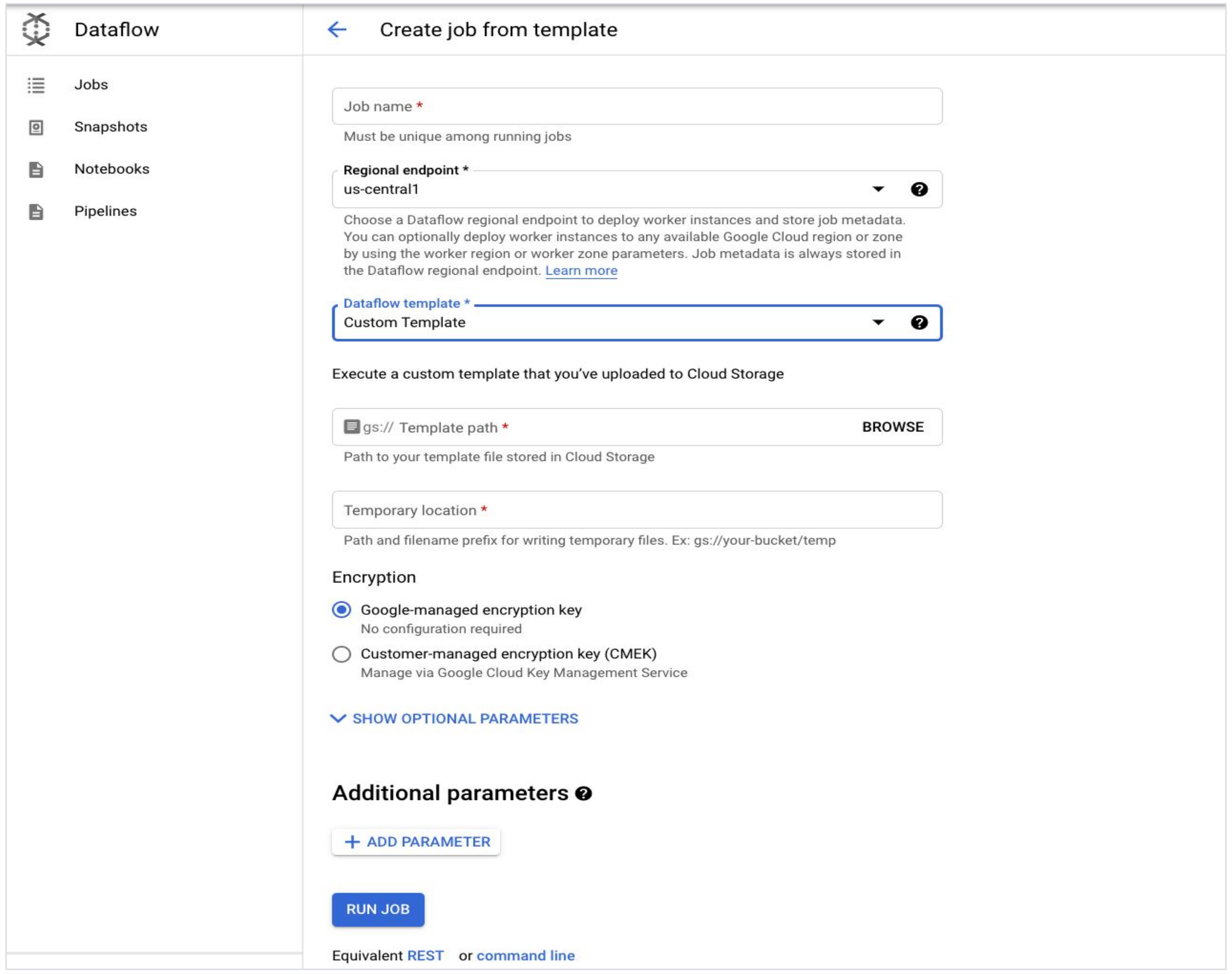
Console

gcloud

REST API

Cloud Scheduler

Launch a flex template using Console



Create job from template

Job name *

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

gs:// Template path * BROWSE

Path to your template file stored in Cloud Storage

Temporary location *

Path and filename prefix for writing temporary files. Ex: gs://your-bucket/temp

Encryption

- [✓ Google-managed encryption key] No configuration required
- Customer-managed encryption key (CMEK) Manage via Google Cloud Key Management Service

SHOW OPTIONAL PARAMETERS

Additional parameters

+ ADD PARAMETER

RUN JOB

Equivalent REST or command line

Google Cloud

Launch a flex template using gcloud

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

Launch a flex template using REST API

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

Launch a flex template using Cloud Scheduler

```
gcloud scheduler jobs create http scheduler-job --schedule="*/30 * * * *"
--uri="https://dataflow.googleapis.com/v1b3/projects/$PROJECT/locations/${REGI
ON}/flexTemplates:launch" --http-method=POST \
 --headers Content-Type=application/json \
 --oauth-service-account-email=email@project.iam.gserviceaccount.com \
 --message-body='{
     "launch_parameter": {
       "jobName": "job-name"
       "parameters": {
         "inputSubscription": "'$SUBSCRIPTION'",
         "outputTable": "'$PROJECT:$DATASET.$TABLE'"
       },
       "containerSpecGcsPath": "'$TEMPLATE_PATH'"
     }
   }'
```

Google Cloud

Classic vs flex templates

Features	Classic	Flex
Any authorized user can invoke the template via Google Cloud Console, gcloud cmd line tool, or REST API	✓	✓
Running pipeline does not require recompiling code	✓	✓
Run pipeline without development environment and associated dependencies	✓	✓
Runtime parameters to customize execution of the pipeline	✓	✓
Separation of staging and execution steps	✓	✓
Job execution graph can be changed after the template is created	✗	✓
Support IOs beyond ValueProvider	✗	✓
Support using SQL as a parameter	✗	✓
Reduce runtime errors by running validations upon job graph construction	✗	✓

Flex Templates

Agenda

- Classic templates
- Flex templates
- Using flex templates
- Google-provided templates

Google-provided templates

✓ Extensive collection of predefined templates

Google Cloud

Google-provided templates

✓ Extensive collection of predefined templates

✓ Intended for point-to-point transfers

Google Cloud

Google-provided templates

- ✓ Extensive collection of predefined templates
- ✓ Intended for point to point transfers
- ✓ Simple transformation using Javascript UDF

Google Cloud

Google-provided templates

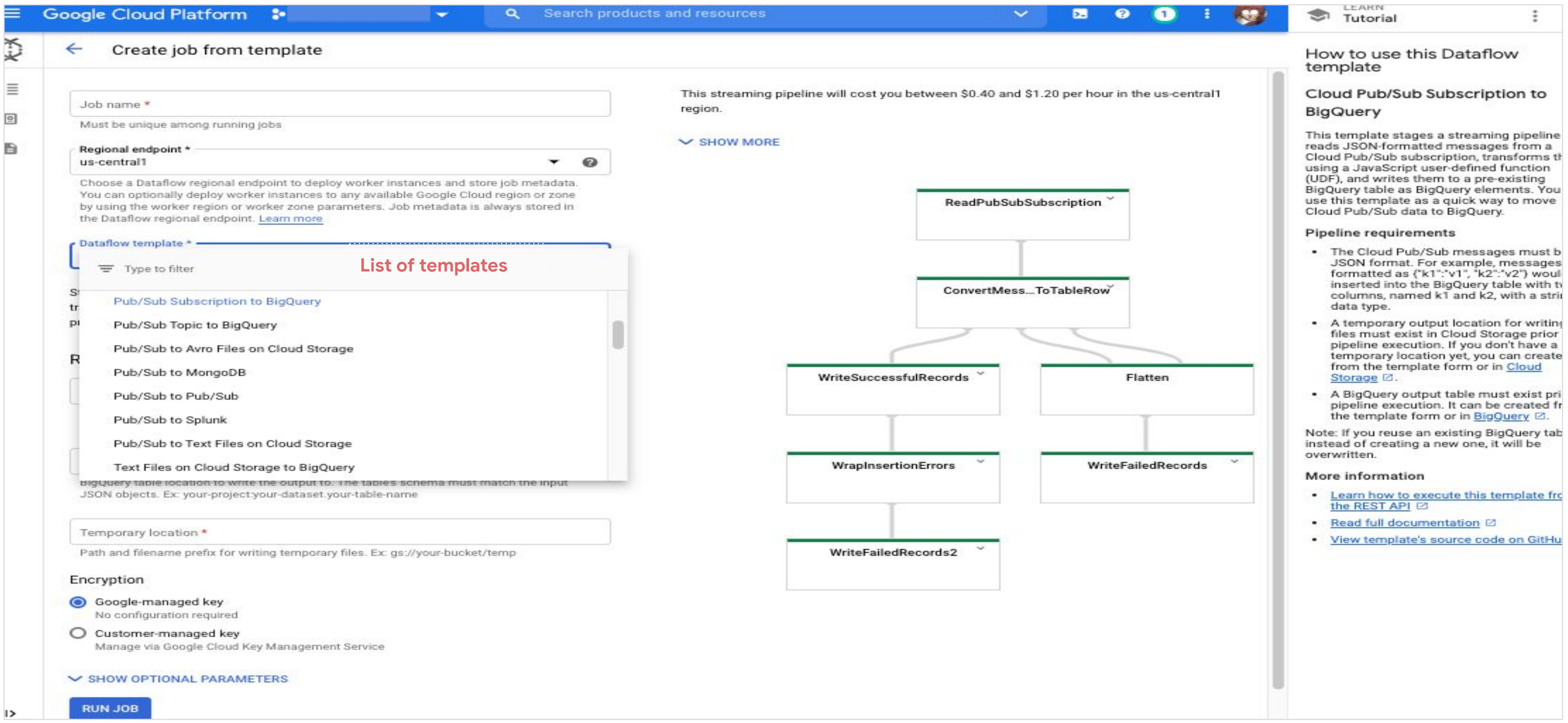
- ✓ Extensive collection of predefined templates
- ✓ Intended for point-to-point transfers
- ✓ Simple transformation using Javascript UDF
- ✓ Code available on GitHub

Google Cloud

Google-provided templates

- ✓ Extensive collection of predefined templates
- ✓ Intended for point-to-point transfers
- ✓ Simple transformation using Javascript UDF
- ✓ Code available on GitHub
- ✓ Active community support

Launch a Google-provided template using Console



Create job from template

Job name *
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 *
Type to filter

- Pub/Sub Subscription to BigQuery
- Pub/Sub Topic to BigQuery
- Pub/Sub to Avro Files on Cloud Storage
- Pub/Sub to MongoDB
- Pub/Sub to Pub/Sub
- Pub/Sub to Splunk
- Pub/Sub to Text Files on Cloud Storage
- Text Files on Cloud Storage to BigQuery

BigQuery table location to write the output to. The table's schema must match the input JSON objects. Ex: your-project:your-dataset.your-table-name

Temporary location *
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

This streaming pipeline will cost you between $0.40 and $1.20 per hour in the us-central1 region.

How to use this Dataflow template

Cloud Pub/Sub Subscription to BigQuery

This template stages a streaming pipeline reads JSON-formatted messages from a Cloud Pub/Sub subscription, transforms th using a JavaScript user-defined function (UDF), and writes them to a pre-existing BigQuery table as BigQuery elements. You use this template as a quick way to move Cloud Pub/Sub data to BigQuery.

Pipeline requirements

- The Cloud Pub/Sub messages must b JSON format. For example, messages formatted as {"k1":"v1", "k2":"v2"} woul inserted into the BigQuery table with tv columns, named k1 and k2, with a strin data type.
- A temporary output location for writin files must exist in Cloud Storage prior pipeline execution. If you don't have a temporary location yet, you can create from the template form or in Cloud Storage.
- A BigQuery output table must exist pri pipeline execution. It can be created fr the template form or in BigQuery.

Note: If you reuse an existing BigQuery tab instead of creating a new one, it will be overwritten.

More information

- Learn how to execute this template fro the REST API
- Read full documentation
- View template's source code on GitHu

Google Cloud

Template portfolio

Streaming templates

- Data Masking/Tokenization from Cloud Storage to BigQuery (using Cloud DLP)
- Kafka to BigQuery
- Pub/Sub Aro to BigQuery
- Pub/Sub Subscription to BigQuery
- Pub/Sub Topic to BigQuery
- Pub/Sub to Avro Files on Cloud Storage
- Pub/Sub to MongoDB
- Pub/Sub to Pub/Sub
- Pub/Sub to Splunk
- Pub/Sub to Text Files on Cloud Storage
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Pub/Sub

Batch templates

- Avro Files on Cloud Storage to Cloud Bigtable
- Avro Files on Cloud Storage to Cloud Spanner
- BigQuery export to Parquet(via Storage API)
- BigQuery to TFRecords
- Cloud BigTable to SequenceFile Files on Cloud Storage
- Cloud Bigtable to Avro Files on Cloud Storage
- Cloud Bigtable to Parquet Files on Cloud Storage
- Cloud Spanner to Avro Files on Cloud Storage
- Cloud Spanner to Text Files on Cloud Storage
- Parquet Files on Cloud Storage to Cloud Bigtable
- SequenceFile Files on Cloud Storage to Cloud BigTable
- Synchronizing CDC data to BigQuery
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Cloud Spanner
- Text Files on Cloud Storage to Datastore
- Text Files on Cloud Storage to Pub/Sub
- Cassandra to Cloud Bigtable

Utilities

- Streaming Data Generator
- Bulk Compress Files on Cloud Storage
- Bulk Decompress Files on Cloud Storage
- Bulk Delete Entities in Datastore

Template portfolio

Streaming templates

- Data Masking/Tokenization from Cloud Storage to BigQuery (using Cloud DLP)
- Kafka to BigQuery
- Pub/Sub Aro to BigQuery
- Pub/Sub Subscription to BigQuery
- Pub/Sub Topic to BigQuery
- Pub/Sub to Avro Files on Cloud Storage
- Pub/Sub to MongoDB
- Pub/Sub to Pub/Sub
- Pub/Sub to Splunk
- Pub/Sub to Text Files on Cloud Storage
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Pub/Sub

Batch templates

- Avro Files on Cloud Storage to Cloud Bigtable
- Avro Files on Cloud Storage to Cloud Spanner
- BigQuery export to Parquet(via Storage API)
- BigQuery to TFRecords
- Cloud BigTable to SequenceFile Files on Cloud Storage
- Cloud Bigtable to Avro Files on Cloud Storage
- Cloud Bigtable to Parquet Files on Cloud Storage
- Cloud Spanner to Avro Files on Cloud Storage
- Cloud Spanner to Text Files on Cloud Storage
- Parquet Files on Cloud Storage to Cloud Bigtable
- SequenceFile Files on Cloud Storage to Cloud BigTable
- Synchronizing CDC data to BigQuery
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Cloud Spanner
- Text Files on Cloud Storage to Datastore
- Text Files on Cloud Storage to Pub/Sub
- Cassandra to Cloud Bigtable

Utilities

- Streaming Data Generator
- Bulk Compress Files on Cloud Storage
- Bulk Decompress Files on Cloud Storage
- Bulk Delete Entities in Datastore

Template portfolio

Streaming templates

- Data Masking/Tokenization from Cloud Storage to BigQuery (using Cloud DLP)
- Kafka to BigQuery
- Pub/Sub Aro to BigQuery
- Pub/Sub Subscription to BigQuery
- Pub/Sub Topic to BigQuery
- Pub/Sub to Avro Files on Cloud Storage
- Pub/Sub to MongoDB
- Pub/Sub to Pub/Sub
- Pub/Sub to Splunk
- Pub/Sub to Text Files on Cloud Storage
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Pub/Sub

Batch templates

- Avro Files on Cloud Storage to Cloud Bigtable
- Avro Files on Cloud Storage to Cloud Spanner
- BigQuery export to Parquet(via Storage API)
- BigQuery to TFRecords
- Cloud BigTable to SequenceFile Files on Cloud Storage
- Cloud Bigtable to Avro Files on Cloud Storage
- Cloud Bigtable to Parquet Files on Cloud Storage
- Cloud Spanner to Avro Files on Cloud Storage
- Cloud Spanner to Text Files on Cloud Storage
- Parquet Files on Cloud Storage to Cloud Bigtable
- SequenceFile Files on Cloud Storage to Cloud BigTable
- Synchronizing CDC data to BigQuery
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Cloud Spanner
- Text Files on Cloud Storage to Datastore
- Text Files on Cloud Storage to Pub/Sub
- Cassandra to Cloud Bigtable

Utilities

- Streaming Data Generator
- Bulk Compress Files on Cloud Storage
- Bulk Decompress Files on Cloud Storage
- Bulk Delete Entities in Datastore

Template portfolio

Streaming templates

- Data Masking/Tokenization from Cloud Storage to BigQuery (using Cloud DLP)
- Kafka to BigQuery
- Pub/Sub Aro to BigQuery
- Pub/Sub Subscription to BigQuery
- Pub/Sub Topic to BigQuery
- Pub/Sub to Avro Files on Cloud Storage
- Pub/Sub to MongoDB
- Pub/Sub to Pub/Sub
- Pub/Sub to Splunk
- Pub/Sub to Text Files on Cloud Storage
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Pub/Sub

Batch templates

- Avro Files on Cloud Storage to Cloud Bigtable
- Avro Files on Cloud Storage to Cloud Spanner
- BigQuery export to Parquet(via Storage API)
- BigQuery to TFRecords
- Cloud BigTable to SequenceFile Files on Cloud Storage
- Cloud Bigtable to Avro Files on Cloud Storage
- Cloud Bigtable to Parquet Files on Cloud Storage
- Cloud Spanner to Avro Files on Cloud Storage
- Cloud Spanner to Text Files on Cloud Storage
- Parquet Files on Cloud Storage to Cloud Bigtable
- SequenceFile Files on Cloud Storage to Cloud BigTable
- Synchronizing CDC data to BigQuery
- Text Files on Cloud Storage to BigQuery
- Text Files on Cloud Storage to Cloud Spanner
- Text Files on Cloud Storage to Datastore
- Text Files on Cloud Storage to Pub/Sub
- Cassandra to Cloud Bigtable

Utilities

- Streaming Data Generator
- Bulk Compress Files on Cloud Storage
- Bulk Decompress Files on Cloud Storage
- Bulk Delete Entities in Datastore

Google Cloud