

[Data Analytics Products](https://cloud.google.com/products/big-data/) (https://cloud.google.com/products/big-data/)

[Cloud Pub/Sub](https://cloud.google.com/pubsub/) (https://cloud.google.com/pubsub/)

[Documentation](https://cloud.google.com/pubsub/docs/) (https://cloud.google.com/pubsub/docs/) [Guides](#)

Using Pub/Sub with Dataflow

[Cloud Dataflow](https://cloud.google.com/dataflow/) (https://cloud.google.com/dataflow/) is a fully-managed service for transforming and enriching data in stream (real-time) and batch modes with equal reliability and expressiveness. It provides a simplified pipeline development environment using the Apache Beam SDK, which has a rich set of windowing and session analysis primitives as well as an ecosystem of source and sink connectors. This quickstart shows you how to use Dataflow to:

- Read messages published to a Pub/Sub topic
- Window (or buffer) the messages into batches
- Write the messages to Cloud Storage

This quickstart introduces you to using Dataflow in Java and Python. [SQL](https://cloud.google.com/dataflow/docs/guides/sql/dataflow-sql-intro) (https://cloud.google.com/dataflow/docs/guides/sql/dataflow-sql-intro) is also supported.

You can also start by using UI-based Dataflow [templates](https://cloud.google.com/dataflow/docs/guides/templates/overview) (https://cloud.google.com/dataflow/docs/guides/templates/overview) if you do not intend to do custom data processing.

Before you begin

Set up your Cloud Storage environment

1. Follow the instructions for [installing and initializing the Cloud SDK](https://cloud.google.com/sdk/docs/) (https://cloud.google.com/sdk/docs/).
2. [Enable billing](https://cloud.google.com/billing/docs/how-to/modify-project) (https://cloud.google.com/billing/docs/how-to/modify-project) for your project.
3. To complete this quickstart, you need to enable the following APIs: Compute Engine, Stackdriver, Cloud Storage, Cloud Storage JSON, Pub/Sub, Cloud Scheduler, Resource Manager, and App Engine.

[ENABLE THE APIS](https://console.cloud.google.com/flows/enableapi?apiid=DATAFLOW) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=DATAFLOW)

It might take a few moments before the APIs appear in the console.

4. Create a service account key:

CREATE A SERVICE ACCOUNT KEY ([HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/CREDENTIALS/S](https://console.cloud.google.com/apis/credentials/s))

- a. From the **Service account** list, select *New service account*.
- b. Enter a name in the **Service account name** field.
- c. From the **Role** list, select **Project > Owner**.
- d. Click **Create**.

The key is sent to your browser's default downloads folder.

5. Set the `GOOGLE_APPLICATION_CREDENTIALS` environment variable to point to the service account key.

```
export GOOGLE_APPLICATION_CREDENTIALS=path/to/my/credentials.json
```

6. Create variables for your bucket and project. Cloud Storage bucket names must be globally unique.

```
BUCKET_NAME=bucket-name
PROJECT_NAME=$(gcloud config get-value project)
```

7. Create a Cloud Storage bucket owned by this project:

```
gsutil mb gs://$BUCKET_NAME
```

8. Create a Pub/Sub topic in this project:

```
gcloud pubsub topics create cron-topic
```

9. Create a Cloud Scheduler job in this project. The job publishes a message to a Cloud Pub/Sub topic at one-minute intervals.

If an App Engine app does not exist for the project, this step will create one.

```
gcloud scheduler jobs create pubsub publisher-job --schedule="* * * * *" \
  --topic=cron-topic --message-body="Hello!"
```

Start the job.

```
gcloud scheduler jobs run publisher-job
```



10. Use the following command to clone the quickstart repository and navigate to the sample code directory:

JAVA

PYTHON

```
git clone https://github.com/GoogleCloudPlatform/java-docs-samples.git
cd java-docs-samples/pubsub/streaming-analytics
```



Stream messages from Pub/Sub to Cloud Storage

Code sample

This sample code uses Dataflow to:

- Read Pub/Sub messages.
- Window (or group) messages by timestamp in 2-minute intervals.
- Save the messages in each window as files in Cloud Storage.

JAVA

PYTHON

```
TREAMING-ANALYTICS/SRC/MAIN/JAVA/COM/EXAMPLES/PUBSUB/STREAMING/PUBSUBTOGCS.JAVA)
```

FEEDBACK (#)

```
import org.apache.beam.examples.common.WriteOneFilePerWindow;
import org.apache.beam.sdk.io.gcp.pubsub.PubsubIO;
import org.apache.beam.sdk.options.Default;
import org.apache.beam.sdk.options.Description;
import org.apache.beam.sdk.options.PipelineOptions;
import org.apache.beam.sdk.options.PipelineOptionsFactory;
import org.apache.beam.sdk.options.StreamingOptions;
import org.apache.beam.sdk.options.Validation.Required;
import org.apache.beam.sdk.Pipeline;
import org.apache.beam.sdk.transforms.windowing.FixedWindows;
import org.apache.beam.sdk.transforms.windowing.Window;
import org.joda.time.Duration;
```



```
import java.io.IOException;

public class PubSubToGCS {
    /*
    * Define your own configuration options. Add your own arguments to be processed
    * by the command-line parser, and specify default values for them.
    */
    public interface PubSubToGCSOptions extends PipelineOptions, StreamingOptions {
        @Description("The Cloud Pub/Sub topic to read from.")
        @Required
        String getInputTopic();
        void setInputTopic(String value);

        @Description("Output file's window size in number of minutes.")
        @Default.Integer(1)
        Integer getWindowSize();
        void setWindowSize(Integer value);

        @Description("Path of the output file including its filename prefix.")
        @Required
        String getOutput();
        void setOutput(String value);
    }

    public static void main(String[] args) throws IOException {
        // The maximum number of shards when writing output.
        int numShards = 1;

        PubSubToGCSOptions options = PipelineOptionsFactory
            .fromArgs(args)
            .withValidation()
            .as(PubSubToGCSOptions.class);

        options.setStreaming(true);

        Pipeline pipeline = Pipeline.create(options);

        pipeline
            // 1) Read string messages from a Pub/Sub topic.
            .apply("Read PubSub Messages", PubsubIO.readStrings().fromTopic(options.getI
            // 2) Group the messages into fixed-sized minute intervals.
            .apply(Window.into(FixedWindows.of(Duration.standardMinutes(options.getWindo
            // 3) Write one file to GCS for every window of messages.
```

```
.apply("Write Files to GCS", new WriteOneFilePerWindow(options.getOutput(),  
  
// Execute the pipeline and wait until it finishes running.  
pipeline.run().waitUntilFinish());  
}  
}
```

Start the pipeline

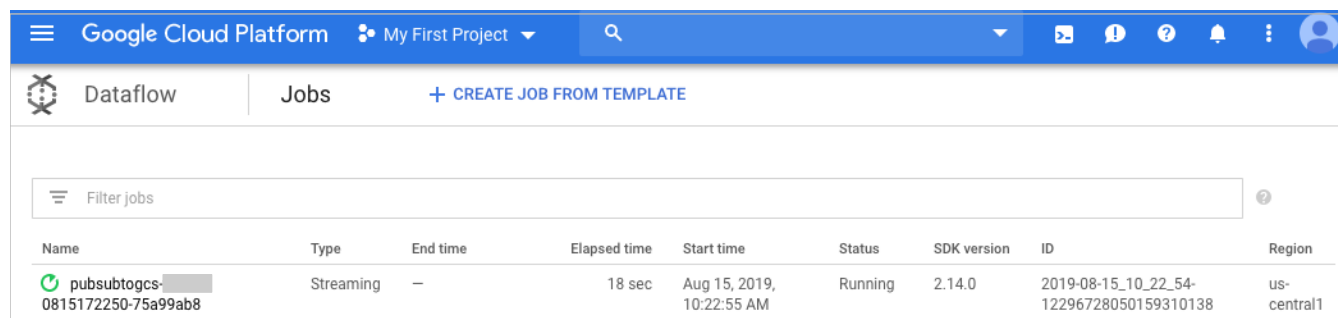
To start the pipeline, use the following command:

```
JAVA PYTHON  
  
mvn compile exec:java \  
-Dexec.mainClass=com.examples.pubsub.streaming.PubSubToGCS \  
-Dexec.cleanupDaemonThreads=false \  
-Dexec.args=" \  
  --project=$PROJECT_NAME \  
  --inputTopic=projects/$PROJECT_NAME/topics/cron-topic \  
  --output=gs://$BUCKET_NAME/samples/output \  
  --runner=DataflowRunner \  
  --windowSize=2"
```


Observe job and pipeline progress

You can observe the job's progress in the Dataflow console.

[GO TO THE DATAFLOW CONSOLE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/DATAFLOW?PROJECT=\)](https://console.cloud.google.com/dataflow?PROJECT=)



The screenshot shows the Google Cloud Platform Dataflow console. The top navigation bar includes the Google Cloud Platform logo, the project name "My First Project", and a search bar. The main content area is titled "Dataflow" and "Jobs", with a button to "CREATE JOB FROM TEMPLATE". Below this is a "Filter jobs" input field. A table lists the jobs, with one job highlighted in green, indicating it is running.

Name	Type	End time	Elapsed time	Start time	Status	SDK version	ID	Region
 pubsubtogcs- 0815172250-75a99ab8	Streaming	–	18 sec	Aug 15, 2019, 10:22:55 AM	Running	2.14.0	2019-08-15_10_22_54- 12296728050159310138	us- central1

Open the job details view to see:

- Job structure
- Job logs
- Stage metrics

The screenshot displays the 'Job details' page in Google Cloud Platform. On the left, a pipeline diagram shows three stages in a vertical sequence, all with a 'Running' status: 'Read PubSub Messages', 'Window.Into()', and 'Write Files to GCS'. On the right, the 'Job summary' section provides key performance indicators: 'System latency (seconds)' and 'Data freshness (seconds)', both showing 'No data for this time interval' for the period of 'Aug 15, 2019 10:22 AM'. Below this, a metadata table lists job details:

Job name	pubsubtogcs-0815172250-75a99ab8
Job ID	2019-08-15_10_22_54-12296728050159310138
Region	us-central1
Job status	Running Stop job
SDK version	Apache Beam SDK for Java 2.14.0
Job type	Streaming
Start time	Aug 15, 2019, 10:22:55 AM
Elapsed time	1 min 25 sec
Encryption type	Google-managed key

The 'Autoscaling' section shows 1 worker and a 'Current state' of 'Worker pool started'.

You may have to wait a few minutes to see the output files in Cloud Storage.

The screenshot shows the 'Bucket details' page for a storage bucket. The 'Objects' tab is active, displaying a list of files. At the top, there are buttons for 'Upload files', 'Upload folder', 'Create folder', 'Manage holds', and 'Delete'. A search bar is present with the placeholder 'Filter by prefix...'. Below the search bar, the breadcrumb path is 'Buckets / [bucket name] / samples'. The object list is as follows:

<input type="checkbox"/>	Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Hold
<input checked="" type="checkbox"/>	output-17:24-17:26-0-of-1	12 B	text/plain	Multi-Regional	8/15/19, 10:26:08 AM UTC-7	Not public	Google-managed key	-	None
<input checked="" type="checkbox"/>	output-17:26-17:28-0-of-1	12 B	text/plain	Multi-Regional	8/15/19, 10:28:07 AM UTC-7	Not public	Google-managed key	-	None
<input checked="" type="checkbox"/>	output-17:28-17:30-0-of-1	12 B	text/plain	Multi-Regional	8/15/19, 10:30:07 AM UTC-7	Not public	Google-managed key	-	None

Alternatively, use the command line below to check which files have been written out.

```
gsutil ls gs://{BUCKET_NAME}/samples/
```



The output should look like the following:

JAVA

PYTHON

```
gs://{BUCKET_NAME}/samples/output-22:30-22:32-0-of-1  
gs://{BUCKET_NAME}/samples/output-22:32-22:34-0-of-1  
gs://{BUCKET_NAME}/samples/output-22:34-22:36-0-of-1  
gs://{BUCKET_NAME}/samples/output-22:36-22:38-0-of-1
```



Cleanup

1. Delete the Cloud Scheduler job.

```
gcloud scheduler jobs delete publisher-job
```



2. Use **Ctrl+C** to stop the program in your terminal.
3. In the Dataflow console, stop the job. Cancel the pipeline without draining it.
4. Delete the topic.

```
gcloud pubsub topics delete cron-topic
```



5. Delete the files created by the pipeline.

```
gsutil -m rm -rf "gs://{BUCKET_NAME}/samples/output*"
```



6. Remove the Cloud Storage bucket.

```
gsutil rb gs://{BUCKET_NAME}
```



What's next

Note: Pub/Sub aggregates the messages in this Quickstart using the publish timestamps created by the Pub/Sub server.

- If you would like to output Pub/Sub messages to files using a different timestamp, you can publish those as custom attributes in the message body, then configure your Dataflow pipeline to use the alternate timestamp. For example, this Java code demonstrates how to do this using [Apache Beam's WithTimestamp](https://github.com/apache/beam/blob/master/sdks/java/core/src/main/java/org/apache/beam/sdk/transforms/WithTimestamps.java) (<https://github.com/apache/beam/blob/master/sdks/java/core/src/main/java/org/apache/beam/sdk/transforms/WithTimestamps.java>)
.
- Take a look at Google's [open-source Dataflow templates designed for streaming](https://cloud.google.com/dataflow/docs/guides/templates/provided-streaming) (<https://cloud.google.com/dataflow/docs/guides/templates/provided-streaming>).
- For more about windowing, see the [Apache Beam Mobile Gaming Pipeline](https://beam.apache.org/get-started/mobile-gaming-example/#hourlyteamscore-advanced-processing-in-batch-with-windowing) (<https://beam.apache.org/get-started/mobile-gaming-example/#hourlyteamscore-advanced-processing-in-batch-with-windowing>)
example.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 25, 2019.