Practical Lesson: Stream Processing

Hint for Naming Resources

When naming resources, use the format `de-2023-[RESOURCE NAME]-[SHORTNAME]` where `[SHORTNAME]` is your unique identifier and should be filled by you. Adjust the `[RESOURCE NAME]` accordingly based on the specific resource you're creating.

Exercise: Streaming Data and BigQuery Integration

In this exercise, you will work with various GCP services including Google Storage, Pub/Sub, BigQuery, and Dataflow. You will create a data pipeline that ingests streaming data, processes it, and stores it in BigQuery.

Part 1: Setting Up Your Environment

1. **Create a Google Cloud Storage Bucket**

```
BUCKET_NAME=de-2023-[YOUR BUCKET]-[SHORTNAME]
gsutil mb gs://$BUCKET_NAME
```

Description: This command creates a new Google Cloud Storage bucket. The bucket will be used to store temporary files and the JavaScript UDF (User-Defined Function) file for Dataflow. **Variables**: We store the bucket name in the variable BUCKET_NAME to reuse it later in the script.

2. **Create a Pub/Sub Topic**

```
TOPIC_NAME=de-2023-topic01-[SHORTNAME]
gcloud pubsub topics create $TOPIC_NAME
```

Description: This command creates a new Pub/Sub topic. The topic will be used to ingest streaming data into our data pipeline.

Part 2: Scheduling Data Ingestion

1. **Create a Cloud Scheduler Job for Positive Reviews** `bash LOCATION=europe-west6 gcloud scheduler jobs create pubsub positive-ratings-publisher \ --schedule="* * * * *" \ --location=$LOCATION \ --topic=$TOPIC_NAME \ --message-body='{"url": "https://beam.apache.org/", "review": "positive"}'`

Description: This command creates a Cloud Scheduler job that publishes a message to the Pub/Sub topic every minute. The message represents a positive review of a website. **Variables**: The location is stored in the `LOCATION` variable for reuse.

2. **Verify Cloud Scheduler Jobs in Google Cloud Console**

After creating the Cloud Scheduler jobs, navigate to the Cloud Scheduler section of the Google Cloud Console to verify that the jobs have been created successfully.

Instructions: Ensure that you see two jobs listed: positive-ratings-publisher and negative-ratings-publisher. Check that their schedules and configurations are correct.

3. **Run the Cloud Scheduler Job Manually**

```
gcloud scheduler jobs run --location=$LOCATION positive-ratings-publisher
```

Description: This command manually triggers the Cloud Scheduler job created in the previous step.

1. **Create a Cloud Scheduler Job for Negative Reviews** `bash gcloud scheduler jobs create pubsub negative-ratings-publisher \ --schedule="*/2 * * * *" \ --location=$LOCATION \ --topic=$TOPIC_NAME \ --message-body='{"url": "https://beam.apache.org/", "review": "negative"}'`

    - **Description**: This command creates another Cloud Scheduler job that publishes a negative review message to the Pub/Sub topic every two minutes.

Part 3: Setting Up BigQuery

1. **Create a BigQuery Dataset**

```
bq --location=$LOCATION mk \
[PROJECT_ID]:tutorial_dataset
```

Description: This command creates a new BigQuery dataset named `tutorial_dataset` in your GCP project. **Note**: Replace `[PROJECT_ID]` with your actual Google Cloud Project ID.

2. **Create a BigQuery Table**

```
bq mk \
    --table \
    [PROJECT_ID]:tutorial_dataset.tutorial \
    url:STRING,review:STRING
```

Description: This command creates a new BigQuery table named `tutorial` within the previously created dataset. The table has two STRING fields: `url` and `review`.

3. Verify BigQuery Dataset and Table

After creating the BigQuery dataset and table, navigate to the BigQuery section of the Google Cloud Console to verify their existence.

Instructions: In the Explorer panel, expand your project and dataset to see the tutorial table. Click on it and review its schema to ensure that it has the url and review fields.

Part 4: Setting Up Dataflow

1. **Create and Edit a JavaScript UDF File**
    - First, create the file:
    -

```
touch dataflow_udf_transform.js
```

- Then, edit the file using nano or your preferred text editor:

```
nano dataflow_udf_transform.js
```

Description: This JavaScript file contains a User-Defined Function (UDF) that will be used by Dataflow to process the streaming data.

2. **Add the JavaScript UDF Code**

After creating and opening the `dataflow_udf_transform.js` file, copy and paste the following JavaScript code into the file:

```
/**
 * User-defined function (UDF) to transform events
 * as part of a Dataflow template job.
 *
 * @param {string} inJson input Pub/Sub JSON message (stringified)
 */
function process(inJson) {
    const obj = JSON.parse(inJson);
    const includePubsubMessage = obj.data && obj.attributes;
    const data = includePubsubMessage ? obj.data : obj;

    if (!data.hasOwnProperty('url')) {
      throw new Error("No url found");
```

```
    } else if (data.url !== "https://beam.apache.org/") {
      throw new Error("Unrecognized url");
    }

    return JSON.stringify(obj);
}
```

Description: This JavaScript function is a User-Defined Function (UDF) that will be used by Dataflow to process incoming Pub/Sub messages. The function checks if the incoming message has a specific URL and throws an error if it does not meet the criteria. This is a basic example, and you can customize the function according to your needs.

3. **Save and Exit the Text Editor**

If you are using `nano`, you can save and exit by pressing `Ctrl+X`, then press `Y` to confirm saving, and finally press `Enter` to exit.

Now, you can continue with the rest of the script as previously provided. This UDF will be used by the Dataflow job to process the incoming Pub/Sub messages. Ensure that your students understand the purpose of the UDF and how it integrates into the data pipeline.

4. **Copy the JavaScript UDF File to Your Bucket**

```
gcloud storage cp dataflow_udf_transform.js gs://$BUCKET_NAME
```

Description: This command uploads the JavaScript UDF file to your Google Cloud Storage bucket.

5. **Enable and Disable Dataflow API (if necessary)**

    - If you need to enable or disable the Dataflow API, you can use the following commands:

```
gcloud services disable dataflow.googleapis.com
gcloud services enable dataflow.googleapis.com
```

Wait for 1 - 2 minutesn: It can take a few minutes for the API to be enabled or disabled. **Description**: These commands toggle the Dataflow API on and off. Sometimes, toggling the API can resolve issues with deploying Dataflow jobs.

6. **Deploy a Dataflow Job**

```
gcloud dataflow jobs run de-2023-dflow01-[SHORTNAME] \
    --gcs-location gs://dataflow-templates/latest/PubSub_to_BigQuery \
    --region $LOCATION \
    --staging-location gs://$BUCKET_NAME/temp \
    --parameters \
inputTopic=projects/[PROJECT_ID]/topics/$TOPIC_NAME,\
```

```
outputTableSpec=[PROJECT_ID]:tutorial_dataset.tutorial,\
javascriptTextTransformGcsPath=gs://$BUCKET_NAME/dataflow_udf_transform.js,\
javascriptTextTransformFunctionName=process

- **Description**: This command deploys a Dataflow job that reads messages from the P
```

7. Verify Dataflow Job in Google Cloud Console

After deploying the Dataflow job, navigate to the Dataflow section of the Google Cloud Console to check the status of your job.

Instructions: Look for the job named de-2023-dflow01-[SHORTNAME] and ensure its status is running or succeeded. Click on the job name to see more details and understand how the data is flowing through different stages of your pipeline.

Part 5: Testing and Cleanup

1. **Test Dead-Letter Queue**

```
gcloud pubsub topics publish $TOPIC_NAME \
    --message='{"url": "https://beam.apache.org/documentation/sdks/java/",
     ↪  "review": "positive"}'
```

Description: This command publishes a message to the Pub/Sub topic that is expected to be rejected by the Dataflow job, testing the dead-letter queue functionality.

1. Validate Data in BigQuery

Now that your streaming data pipeline is up and running, and positive and negative reviews are being published every couple of minutes, you should see this data showing up in your BigQuery table.

Instructions: Navigate to the BigQuery section of the Google Cloud Console, find your tutorial table, and run a query to view the latest data. Sample Query:

```
SELECT * FROM `[PROJECT_ID].tutorial_dataset.tutorial`
LIMIT 10;
```

Replace [PROJECT_ID] with your actual Google Cloud Project ID. Expected Outcome: You should see rows of data representing both positive and negative reviews, with new rows appearing every couple of minutes.

3. Understand Why Messages Failed (Optional)

If you want to gain a deeper understanding of why certain messages ended up in the dead-letter queue, you can analyze the error messages stored in the tutorial_error_records table.

Instructions: Run the following query to extract and count the distinct error messages.

```
SELECT errorMessage, COUNT(*) as error_count
FROM `[PROJECT_ID].tutorial_dataset.tutorial_error_records`
GROUP BY errorMessage;
```

Replace [PROJECT_ID] with your actual Google Cloud Project ID. Expected Outcome: You should see the different error messages explaining why certain messages couldn't be processed.

Clean Up Resources

First, cancel the running Dataflow job:

```
DATAFLOW_JOB_ID=$(gcloud dataflow jobs list \
--filter 'NAME=de-2023-dflow01-[SHORTNAME] AND STATE=Running' \
--format 'value(JOB_ID)' \
--region $LOCATION)
gcloud dataflow jobs cancel --region $LOCATION $DATAFLOW_JOB_ID
```

Then, delete the Cloud Scheduler jobs:

```
gcloud scheduler jobs delete negative-ratings-publisher --location=$LOCATION
gcloud scheduler jobs delete positive-ratings-publisher --location=$LOCATION
```

And finally, delete the Pub/Sub topic and any subscriptions:

```
gcloud pubsub subscriptions delete de-2023-sub01-[SHORTNAME]
gcloud pubsub topics delete $TOPIC_NAME
```