PubSub, FaaS and Data Ingestion

Overview

Introduction

Pub/Sub Ingestion with Cloud Functions

One way to do this is to use Cloud Functions to trigger Ingestion from a specified Messaging event occurs. For example, you can create a function that triggers a DAG when an object changes in a Cloud Storage bucket, or when a message is pushed to a Pub/Sub topic, as is done in this experiment.

Before you begin

Enable APIs for your project

- Enable the Cloud Functions APIs.

Enable the APIs

Enable the Airflow REST API

Depending on your version of Airflow you could have to enable this.

Trigger a pipeline run with Cloud Pub/Sub

The following code samples show you how to write, deploy, and trigger a pipeline using an Event-Driven Cloud Function with a Cloud Pub/Sub trigger.

Build and compile a simple Pipeline

Using Kubeflow Pipelines SDK, build a scheduled pipeline and compile it into a JSON file.

Sample hello-world-scheduled-pipeline:

```
import json
from kfp.v2 import compiler
from kfp.v2 import dsl
from kfp.v2.dsl import component

# A simple component that prints and returns a greeting string
@component
def hello_world(message: str) -> str:
    greeting_str = f'Hello, {message}'
    print(greeting_str)
    return greeting_str

# A simple pipeline that contains a single hello_world task
@dsl.pipeline(
    name='hello-world-scheduled-pipeline')
def hello_world_scheduled_pipeline(greet_name: str):
    hello_world_task = hello_world(greet_name)

# Compile the pipeline and generate a JSON file
compiler.Compiler().compile(pipeline_func=hello_world_scheduled_pipeline,
                            package_path='hello_world_scheduled_pipeline.j
son')
```

Upload compiled pipeline JSON to Cloud Storage bucket

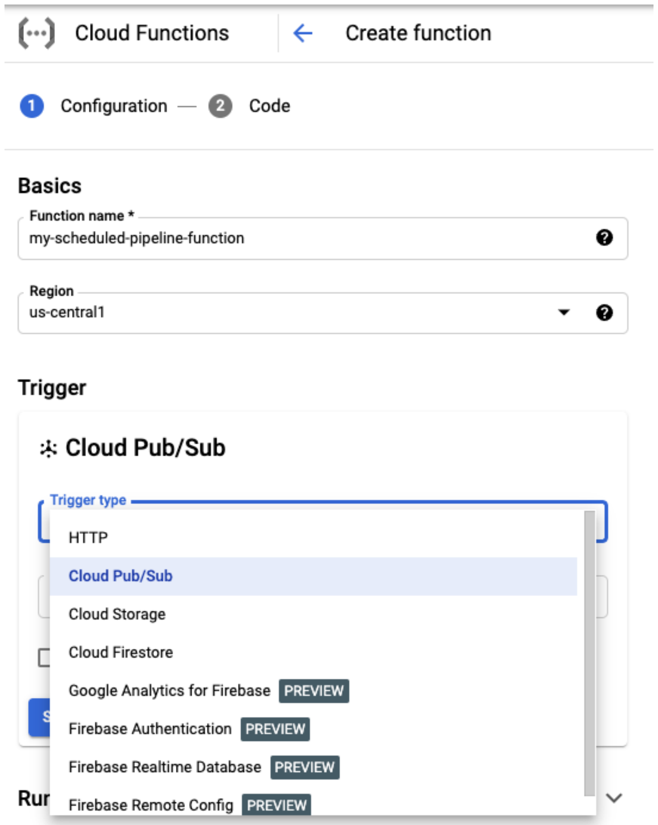
1. Open the Cloud Storage browser in the Google Cloud Console.

   Cloud Storage Browser
2. Click the Cloud Storage bucket you created when you configured your project.
3. Using either an existing folder or a new folder, upload your compiled pipeline JSON (in this example hello_world_scheduled_pipeline.json) to the selected folder.
4. Click the uploaded JSON file to access the details. Copy the **gsutil URI** for later use.

Create a Cloud Function with Pub/Sub Trigger

1. Visit the Cloud Functions page in the console.

   [Go to the Cloud Functions page]
2. Click the **Create function** button.
3. In the **Basics** section, give your function a name (for example my-scheduled-pipeline-function).
4. In the **Trigger** section, select **Cloud Pub/Sub** as the Trigger type.


Cloud Functions ← Create function

1 Configuration — 2 Code

Basics

Function name *: my-scheduled-pipeline-function

Region: us-central1

Trigger

Cloud Pub/Sub

Trigger type
HTTP
Cloud Pub/Sub
Cloud Storage
Cloud Firestore
Google Analytics for Firebase PREVIEW
Firebase Authentication PREVIEW
Firebase Realtime Database PREVIEW
Firebase Remote Config PREVIEW

5. In the **Select a Cloud Pub/Sub topic** dropdown, click **Create a topic**.
6. In the **Create a topic** box, give your new topic a name (for example my-scheduled-pipeline-topic), and select **Create topic**.
7. Leave all other fields as default and click **Save** to save the Trigger section configuration.
8. Leave all other fields as default and click **Next** to proceed to the Code section.
9. Under **Runtime**, select **Python 3.7**.
10. In **Entry** point, input "subscribe" (the example code entry point function name).
11. Under **Source code**, select **Inline Editor** if it's not already selected.
12. In the main.py file, add in the following code:

```
import base64
import json
from google.cloud import aiplatform
```

```
  PROJECT_ID = 'your-project-id'                     # <---CHANGE
THIS
  REGION = 'your-region'                             # <---CHANGE
THIS
  PIPELINE_ROOT = 'your-cloud-storage-pipeline-root' # <---CHANGE
THIS

  def subscribe(event, context):
    """Triggered from a message on a Cloud Pub/Sub topic.
    Args:
          event (dict): Event payload.
          context (google.cloud.functions.Context): Metadata for the
event.
    """
    # decode the event payload string
    payload_message = base64.b64decode(event['data']).decode('utf-
8')
    # parse payload string into JSON object
    payload_json = json.loads(payload_message)
    # trigger pipeline run with payload
    trigger_pipeline_run(payload_json)

  def trigger_pipeline_run(payload_json):
    """Triggers a pipeline run
    Args:
          payload_json: expected in the following format:
            {
              "pipeline_spec_uri": "<path-to-your-compiled-
pipeline>",
              "parameter_values": {
                "greet_name": "<any-greet-string>"
              }
            }
    """
    pipeline_spec_uri = payload_json['pipeline_spec_uri']
    parameter_values = payload_json['parameter_values']

    # Create a PipelineJob using the compiled pipeline from
pipeline_spec_uri
    aiplatform.init(
        project=PROJECT_ID,
        location=REGION,
    )
    job = aiplatform.PipelineJob(
        display_name='hello-world-pipeline-cloud-function-
invocation',
        template_path=pipeline_spec_uri,
        pipeline_root=PIPELINE_ROOT,
        enable_caching=False,
        parameter_values=parameter_values
    )

    # Submit the PipelineJob
    job.submit()
```

Replace the following:

- *PROJECT_ID*: The Google Cloud project that this pipeline runs in.
- *REGION*: The region that this pipeline runs in.
- *PIPELINE_ROOT*: Specify a Cloud Storage URI that your pipelines service account can access. The artifacts of your pipeline runs are stored in the pipeline root.

13. In the requirements.txt file, replace the contents with the following package requirements:

```
google-api-python-client>=1.7.8,<2
google-cloud-aiplatform
```

14. Click **DEPLOY** to deploy the Function.
15. Use the Pub/Sub console to enter a sample message into the Topic that was created.
16. View the processed pipeline.

Cleanup

To avoid incurring charges to your GCP account for the resources used in this experiment:

1. (Optional) To save your data, download the data from the Cloud Storage bucket for the Cloud Composer environment and the storage bucket you created for experiment.
2. Delete the Cloud Function.

You've now learned to do some cool stuff with Cloud Composer, Cloud Functions and Pub/Sub.

Congratulations!

In this experiment, you created and ran a serverless data ingestion pipeline triggering Pub/Sub with FaaS and Apache Airflow.