Connecting Google BigQuery and Wiliot Platform via Google Cloud Functions

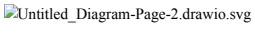
Disclaimer

Wiliot offers this tutorial leveraging third-party tools but does not offer such tools as a product. For help and support with the tools described in this article, please refer directly to the respective vendors' documentation and associated support systems (links are provided at the bottom of this article).

Important Notice

In this method, Google Cloud Function is using BigQuery API for "Streaming insert" and this feature does not have Free Tier, and a payment-related setup is always needed.

Overview



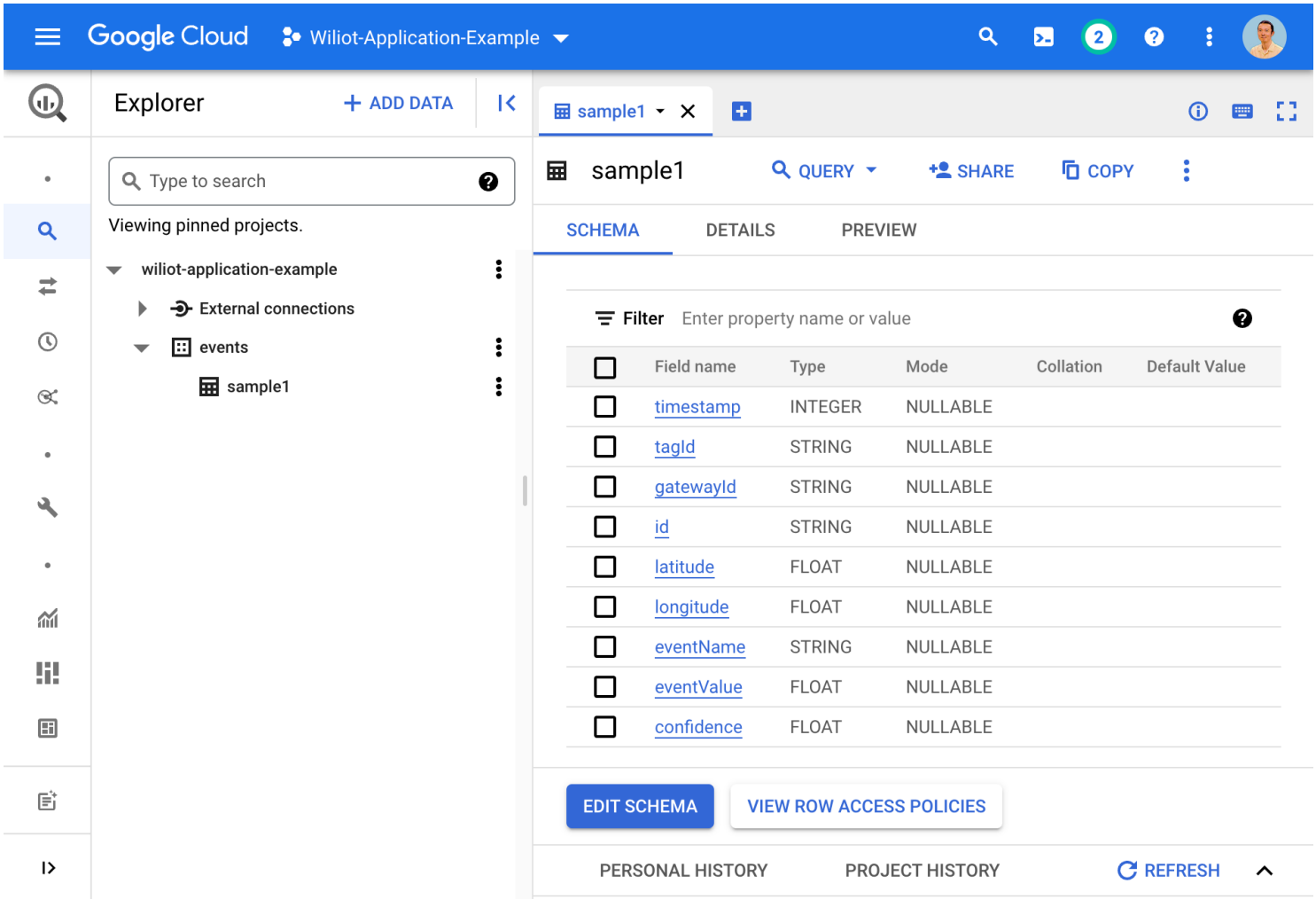
1. HTTPS trigger of Google Cloud Functions is used as an Endpoint for Wiliot C2C Application.
    1. In this example, unauthenticated access is allowed to simplify steps.
    2. For simplicity, use JSON format for the message body used by the C2C Application.
2. In Cloud Functions, convert data format if needed and write formatted data to the BigQuery , using BigQuery API Client Library . At this step, internal authorization is managed by IAM.

Detailed Steps

1. [GCP] Prepare BigQuery Table to store Events

1. Create a Project. In this example, the Project is named `wiliot-application-example` .
2. Create a Dataset to store event data, in this example, the Dataset is named `events` .
3. Create a Table inside the Dataset. In this example, the Table is named `sample2` .
   Since BigQuery is a relational database, schema shall be provided to the Table. This example is using the following schema (for simplicity, this document is not using event-specific key-sets):

```
start:INTEGER,assetId:STRING,categoryId:STRING,eventName:STRING,
value:STRING,confidence:FLOAT
```



2. [GCP] Create Cloud Function to Post Event Data

1. Start to Create a new Function on Function List, by clicking the "CREATE FUNCTION" button.
2. Set Configuration of the Function:
    1. For simplicity, select "Allow unauthorized invocations".
    2. In this example, the following setting is used:
        1. Environment: 2nd gen.
        2. Function name: any name is OK. (example: `bigquery-writer`)
    3. This example is using python as a scripting language and is verified with `python 3.10` runtime.
    4. If the script depends on environment variables, please add the following runtime environment variable from "Runtime, build, connections and security settings > Runtime environment variables". In this example, the following variables are used to specify BigQuery Dataset:
       `GCP_PROJECT` : `wiliot-application-example`
       `BIGQUERY_DATASET` : `events`
    5. For the simplest setup, it is recommended to use the Default compute service account as the service account to run the function. Otherwise, please make sure the account has permission to write data to the BigQuery table.

3. Write Code — this example is using Python as a scripting language:
    1. Create Main Python Script (`main.py`), and set the entry point to the function (in this example, `post_data`).

```
import os
import json
from google.cloud import bigquery
import functions_framework

@functions_framework.http
def post_data(request):
  project_id = os.environ.get('GCP_PROJECT')
  bigquery_dataset = os.environ.get('BIGQUERY_DATASET')
  if (not project_id or not bigquery_dataset):
        return 'Error reading Function environment variables'
  d = request.get_json(silent=True)
  bigquery_table = d['measurement']
  client = bigquery.Client(project=project_id)
  table_ref = client.dataset(bigquery_dataset).table(bigquery_table)
  table = client.get_table(table_ref)
  e = {}
  e['start'] = int(d['start'])
  e['assetId'] = d['assetId']
  e['categoryId'] = d['categoryId']
  e['eventName'] = d['eventName']
  e['value'] = d['value']
  e['confidence'] = float(d['confidence'])
  rows_to_insert = [e]
  errors = client.insert_rows_json(table, rows_to_insert, ignore_unknown_values=True)
  if len(errors) > 0:
        return str(errors)
  else:
        return 'OK'
```

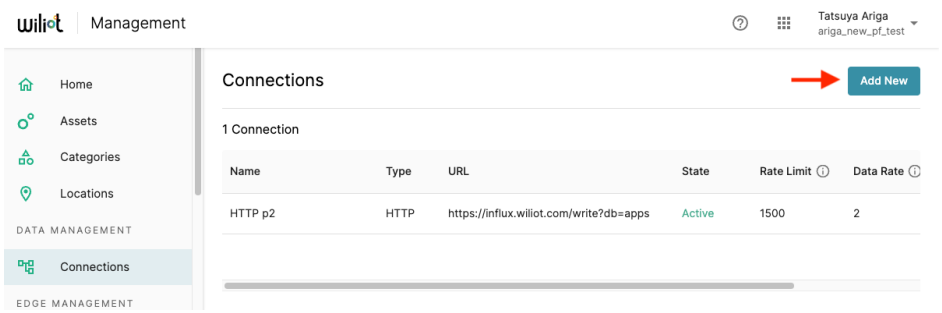
    It is notable that:
    1. this function is converting JSON data format to follow the schema of the BigQuery table defined above.
    2. the API `insert_rows_json()` is streaming insert API of BigQuery and payment is required to use this feature.
2. And `requirements.txt` :

```
functions-framework==3.*
google-cloud-bigquery
```

4. Deploy — Once it's deployed, the Cloud Function console will show the URL to access created function.

3. [Wiliot Platform] Create a Connection

Please use the URL retrieved from the last step and create a new Connection in the Wiliot Platform.



Type	HTTP
URL	The URL that is generated in Step 2.
Method	POST
Header	No header is needed for this sample
Body	`{ "measurement":"sample2", "assetId":"{{assetId}}", "categoryId":"{{categoryId}}", "eventName":"{{eventName}}", "start":"{{start}}", "ownerId":"{{ownerId}}", "confidence":"{{confidence}}", "value":"{{value}}", "createdOn":"{{createdOn}}" {{#keySet}},"{{key}}":"{{{value}}}"{{/keySet}} }`

Here, please be noted the field "measurement" is used to specify the BigQuery table.

Result

Once setup is finished, the Wiliot Platform starts to send messages to the endpoint created in Step 2. And events can be seen on the BigQuery console.

References

- Google Cloud documentation