## **Google Cloud Auto-Import Cloud Function**

### **Purpose**

This document provides step-by-step instructions for creating the Cloud Function that automatically exports files imported to a bucket into a specific FHIR Store.

### **Requirements**

A Google Cloud project

Logging and Cloud Function permissions

### **Procedure**

#### **Step 1: Ensure Access to Cloud Functions and Logging**

Before we can create a Cloud Function to complete the necessary tasks, we need to have the permissions to be able to both create and access Cloud Functions, and to view their output logs for debugging purposes.

1. Navigate to the Cloud Functions tab in Google Cloud, and make sure you don’t have trouble viewing any current functions
2. In this case, I was granted the Cloud Function Admin role as well as the Logging Access Admin Role
3. If you’re not sure, come back to this section once you’re sure your function has been triggered, and navigate to the Logs tab in your function. You should be able to see information about the status of your function.

#### **Step 2: Create a Bucket (Requires Storage Admin Role)**

This’ll serve as the entry point for files that you want to store in your FHIR Store.

Navigate to Buckets from the Cloud Storage page in the Google Cloud console. Make sure to make note of the location and name of your bucket, as we’ll need these in the future.

#### **Step 3: Create a Cloud Function**

Next, we’ll create the function itself.

From the Cloud Functions landing page, click Create Function at the top. We’ll use the 1st gen Cloud Function. It’ll also ask for a name. Here, I used **auto-export-to-store**. Select the necessary region, then determine when and how you want your function to actually run. Select Cloud Storage, then choose the bucket you created earlier. In this tutorial, we want the Cloud Function to trigger **On (finalizing/creating) file in the selected bucket**.

Save these options, then click Next at the bottom.

Now, we’ll get into actually coding up the response we want when our function triggers.

#### **Step 4: Create Source for Your Function**

Select the runtime environment you wish to code in. In this tutorial, we’ll use Python 3.7. Leave the **Entry point** as **hello\_gcs**.

On the left-hand sidebar, you’ll see two files: **main.py** and **requirements.txt**. These represent your source code and any dependencies you want, respectively.

For this tutorial, we won’t add anything to the **requirements.txt** file.

In your **main.py**, give the source code for your Cloud Function.

import os

from googleapiclient import discovery

def hello\_gcs(event, context):

print(f"Processing file: {event['name']}.")

# Create client to interact with GC

client = discovery.build("healthcare", "v1")

# Get names of bucket and FHIR store

fhir\_store\_id = 'FHIR\_STORE\_ID'

gcs\_uri = 'BUCKET\_URI'

# Format of FHIR file

content\_format = "BUNDLE\_PRETTY"

body = {

"contentStructure": content\_format,

"gcsSource": {"uri": gcs\_uri},

}

request = (

client.projects()

.locations()

.datasets()

.fhirStores()

.import\_(name=fhir\_store\_id, body=body)

)

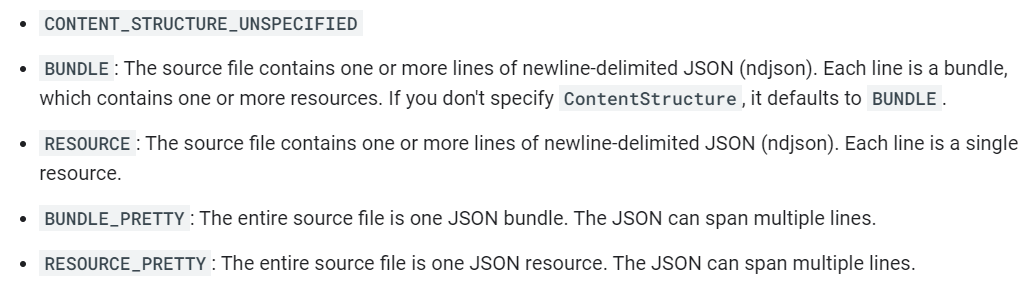
response = request.execute()

print("Imported FHIR resources: {}".format(gcs\_uri))

return response

Some things to note:

* **FHIR\_STORE\_ID** and **BUCKET\_ID** should be your FHIR Store and bucket locations, respectively
* **print( )** will output in the Logs tab.
* **content\_format** can be chosen from the following values:



In this tutorial, I used Synthea to create data in the **BUNDLE\_PRETTY** format, but you should update this value based on the format of the data you’re working with.

* This function will attempt to add the entire contents of the bucket to the FHIR Store *every time* the function is triggered. It does not delete old files after they’ve been added into the store, so adding subsequent files might become slower and slower as you will be attempting to add all the previous files in addition to the latest one.   
    
  This could be updated to do something like delete the contents of the bucket after the files have been successfully added, but again, depends on the use case.

## **Conclusion**

Provisioning a Cloud Function to automatically import files added to a bucket into a FHIR Store is the foundation of a pipeline to store medical records. It greatly simplifies having to import resources, but is highly configurable as well. I adapt this function to be used in parallel with Terraform to create the entire pipeline programmatically in **Provisioning Auto-Import Google Cloud Resources Using Terraform**. Following this tutorial, you should now be able to create a Cloud Function that enables part of this pipeline.