

CS4830 Big Data Laboratory

Jan – May 2021

Lab 6 - Assignment



ME17B158 - Omkar Nath

Q1. In this assignment, you will count the number of lines in a file uploaded to the GCS bucket in real-time by using Google Cloud Functions and Pub/Sub.

a. Download the file from here:

<https://filesamples.com/samples/document/txt/sample1.txt>

b. Write a Google cloud Function which gets triggered whenever a file is added to a bucket and publishes the file name to a topic in Pub/Sub.

c. Write a python file, which acts as a subscriber to this topic and prints out the number of lines in the file in real-time. (Documentation for pub/sub available at:

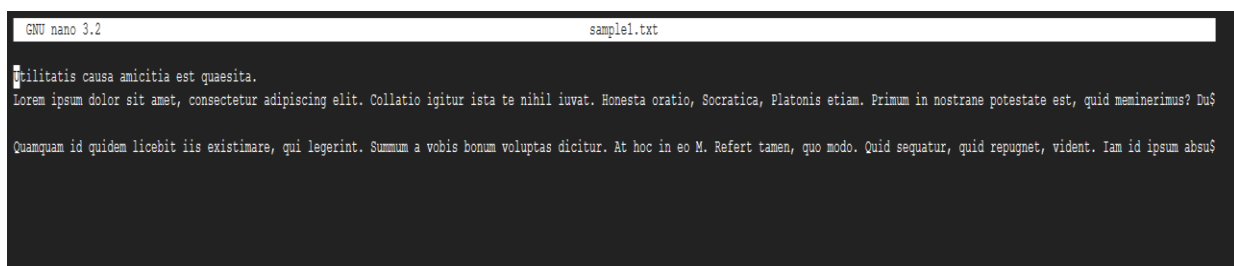
<https://pypi.org/project/google-cloud-pubsub/>)

Test the steps (b) and (c) using the file downloaded in step (a). Ensure that you add screenshots and share the code for all steps.

Part a:

The file is downloaded from the given location, and then uploaded to the Google Cloud, as seen in the below screen shot:

```
me17b158@cloudshell:~/lab6 (splendid-sector-305218)$ nano sample1.txt
```



```
GNU nano 3.2 sample1.txt
Utilitatis causa amicitia est quaesita.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Collatio igitur ista te nihil iuvat. Honestas oratio, Socratica, Platonis etiam. Primum in nostrane potestate est, quid meminerimus? Du$
Quamquam id quidem licebit iis existimare, qui legerint. Summum a vobis bonum voluptas dicitur. At hoc in eo M. Refert tamen, quo modo. Quid sequatur, quid repugnet, vident. Iam id ipsum absu$
```

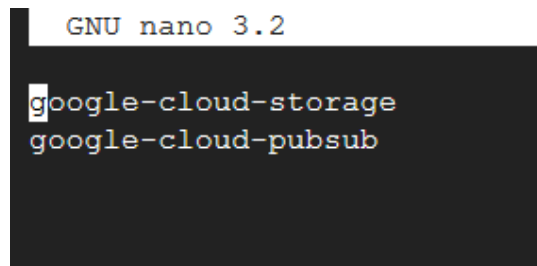
Part b:

The python code for the publisher is as given below:

```
GNU nano 3.2
from google.cloud import pubsub_v1

def publish(input, context):
    publisher = pubsub_v1.PublisherClient()
    topic = 'projects/splendid-sector-305218/topics/lab6'
    data = input['name'].encode("utf-8")
    publisher.publish(topic, data)
```

A requirements file is also uploaded as shown below.



This function is set to get triggered on adding a file as shown below.

```
me17b158@cloudshell:~/lab6 (splendid-sector-305218)$ gcloud functions deploy publish --runtime python37 --trigger-resource me17b158_cs4830 --trigger-event google.storage.object.finalize
```

```
gcloud functions deploy publish --runtime python37 --trigger-resource
me17b158_cs4830 --trigger-event google.storage.object.finalize
```

```
me17b158@cloudshell:~/lab6 (splendid-sector-205218)$ gcloud functions deploy publish --runtime python37 --trigger-resource me17b158_cs4830 --trigger-event google.storage.object.finalize
Deploying function (may take a while - up to 2 minutes)...:
For Cloud Build Stackdriver Logs, visit: https://console.cloud.google.com/logs/viewer?project=splendid-sector-305218&advancedFilter=resource.type%3Dbuild%0Aresource.labels.build_id%3De01f9170-7117-4033-baaa-d705e5232fb6%0AlogName%3Dprojects%2Fsplendid-sector-305218%2Flogs%2Fcloudbuild
Deploying function (may take a while - up to 2 minutes)...done.
availableMemoryMb: 256
buildId: e01f9170-7117-4033-baaa-d705e5232fb6
entryPoint: publish
eventTrigger:
  eventType: google.storage.object.finalize
  failurePolicy: {}
  resource: projects/_/buckets/me17b158_cs4830
  service: storage.googleapis.com
ingressSettings: ALLOW_ALL
labels:
  deployment-tool: cli-gcloud
name: projects/splendid-sector-305218/locations/us-central1/functions/publish
runtime: python37
serviceAccountEmail: splendid-sector-305218@appspot.gserviceaccount.com
sourceUploadUrl: https://storage.googleapis.com/gcf-upload-us-central1-6ad7020a-8b55-41a6-a11e-7ddd0e06ee6/c8de1c2d-db20-48c0-90b9-0037bdd9e181.zip?GoogleAccessId=service-80935782343@gcf-admi
n-robot.iam.gserviceaccount.com&Expires=1618152778&Signature=YRZWH3uuEFO6W745%2F4k%2BjwTviY2zRkdX7JlUPWQ0cdUG%2BgAajPee445IQH34KQVLA3Vd33UKIEFdFfjWYJJ4JKnoZrsgnVqvDlqIkI6VzEng%2B2dEH1ob
afmqm5OfSfmG5GuaszwLBRNWCz848S9xowGpTv2Vlpsy5ejI5fWaQoDnnC6xsZ%2BB8RP14r1NQ39Bu9apT526Ukz7fCtqU55uQANC3u%2B7HmIV9uOiSkI6Le2aKaFudn9aBz98Xzt4CKJdfyYvYKnFL90oQyDsPJH2jas8Ozx7nMgqtVpRI1LLUsG8QM7
ZCTghKJOTMRIR0bu82FnOadczAJU0%2BA33D%3D
status: ACTIVE
timeout: 60s
updateTime: '2021-04-11T14:23:44.323Z'
versionId: '2'
```

The file is copied to trigger the function as shown below.

```
me17b158@cloudshell:~/lab6 (splendid-sector-305218)$ gsutil cp sample1.txt gs://me17b158_cs4830
Copying file://sample1.txt [Content-Type=text/plain]...
/ [1 files][ 608.0 B/ 608.0 B]
Operation completed over 1 objects/608.0 B.
```

Part c

The code for the subscriber is as shown below:

```
GNU nano 3.2

from google.cloud import pubsub_v1

subscriber = pubsub_v1.SubscriberClient()
subscription = 'projects/splendid-sector-305218/subscriptions/lab6-sub'

def count(input):
    print(input.data)
    print('Data Received Successfully')

    with open('sample1.txt', 'r') as file:
        lines = 0
        for line in file:
            lines = lines + 1

    print('Number of lines in file: ' + str(lines))
    input.ack()

runner = subscriber.subscribe(subscription, count)

try:
    runner.result()
except KeyboardInterrupt:
    runner.cancel()
```

The code is run as shown below.

```
me17b158@cloudshell:~/lab6 (splendid-sector-305218)$ python3 subscriber
b'sample1.txt'
Data Received Successfully
Number of lines in file: 4
```

The output is correctly shown as 4 lines.

Attached Files:

The codes for the Python files for the Publisher, Subscriber and the Requirements file have all been attached.

Q2. There are two kinds of subscribers - pull and push subscribers. What are the differences between the two and when would you prefer one over the other?

About Subscribers:

To receive messages published to a topic, you must create a subscription to that topic. Only messages published to the topic after the subscription is created are available to subscriber applications. The subscription connects the topic to a subscriber application that receives and processes messages published to the topic.

Pull Subscription:

In pull delivery, your subscriber application initiates requests to the Pub/Sub server to retrieve messages. The subscribing application explicitly calls the pull method, which requests messages for delivery. The Pub/Sub server responds with the message (or an error if the queue is empty), and an ack ID. The subscriber explicitly calls the acknowledge method, using the returned ack ID to acknowledge receipt.

Push Subscription:

In push delivery, Pub/Sub initiates requests to your subscriber application to deliver messages. The Pub/Sub server sends each message as an HTTPS request to the subscriber application at a pre-configured endpoint. The endpoint acknowledges the message by returning an HTTP success status code. A non-success response indicates that the message should be resent.

Differences between Push and Pull Subscribers:

	Pull	Push
Endpoints	Any device on the internet that has authorized credentials is able to call the Pub/Sub API.	An HTTPS server with non-self-signed certificate accessible on the public web.
Load balancing	Multiple subscribers can make pull calls to the same "shared" subscription.	The push endpoint can be a load balancer.
Configuration	No configuration is necessary.	No configuration is necessary for App Engine apps in the same project as the subscriber. Endpoints must be reachable via DNS names and have SSL certificates installed.
Flow control	The subscriber client controls the rate of delivery.	The Pub/Sub server automatically implements flow control.
Efficiency and throughput	Achieves high throughput at low CPU and bandwidth by allowing batched delivery and acknowledgments as well as massively parallel consumption.	Delivers one message per request and limits maximum number of outstanding messages.

Cases when Pull Subscriber is preferred:

- Large volume of messages (many more than 1/second).
- Efficiency and throughput of message processing is critical.
- Public HTTPS endpoint, with non-self-signed SSL certificate, is not feasible to set up.

Cases when Push Subscriber is preferred:

- Multiple topics that must be processed by the same webhook.
- App Engine Standard and Cloud Functions subscribers.
- Environments where Google Cloud dependencies are not feasible to set up.

Reference:

<https://cloud.google.com/pubsub/docs/subscriber>