Big Data Lab 6 - Assignment

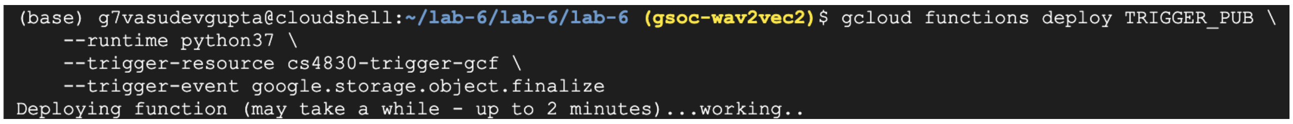
Vasudev Gupta (ME18B182)

1) Following workflow is followed for the setting up Pub/Sub:

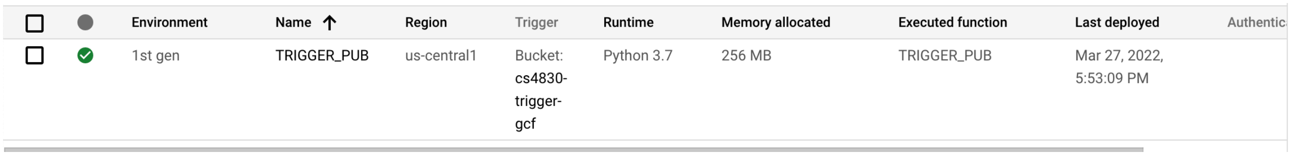
A topic is created and the subscription is setup using following commands.

```
# gcloud pubsub topics create me18b182-pub
# gcloud pubsub subscriptions create me18b182-sub --topic me18b182-pub
```

Then, a google cloud function is deployed using the following command:



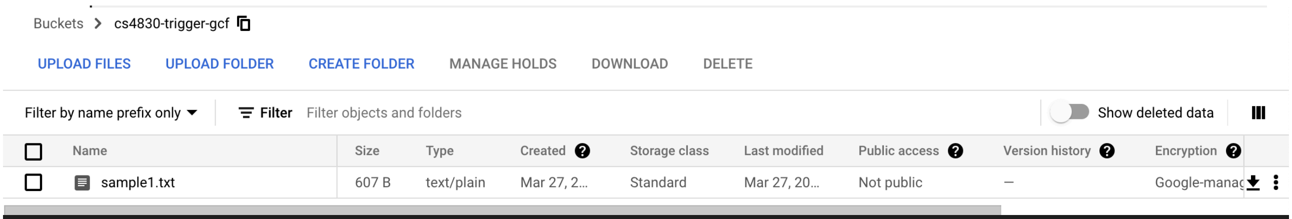
```
(base) g7vasudevgupta@cloudshell:~/lab-6/lab-6/lab-6 (gsoc-wav2vec2)$ gcloud functions deploy TRIGGER_PUB \
    --runtime python37 \
    --trigger-resource cs4830-trigger-gcf \
    --trigger-event google.storage.object.finalize
Deploying function (may take a while - up to 2 minutes)...working..
```



☐	●	Environment	Name ↑	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentic
☐	✓	1st gen	TRIGGER_PUB	us-central1	Bucket: cs4830-trigger-gcf	Python 3.7	256 MB	TRIGGER_PUB	Mar 27, 2022, 5:53:09 PM	

Then, script ***sub.py*** is run and following command is run to download the dataset and copying it to the GCS bucket. After few seconds, we got the response as shown in third image below.

```
# wget https://filesamples.com/samples/document/txt/sample1.txt
# gsutil cp sample1.txt gs://cs4830-trigger-gcf
```

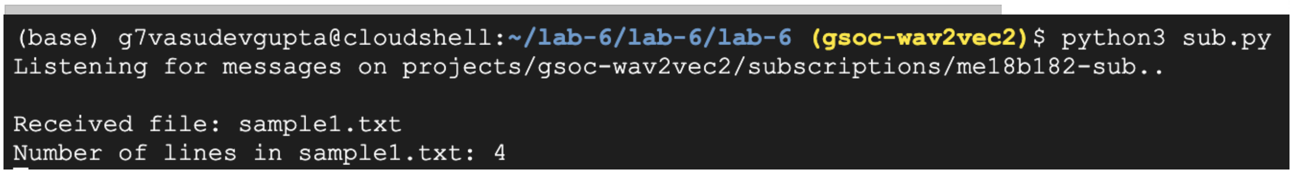


Buckets > cs4830-trigger-gcf

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

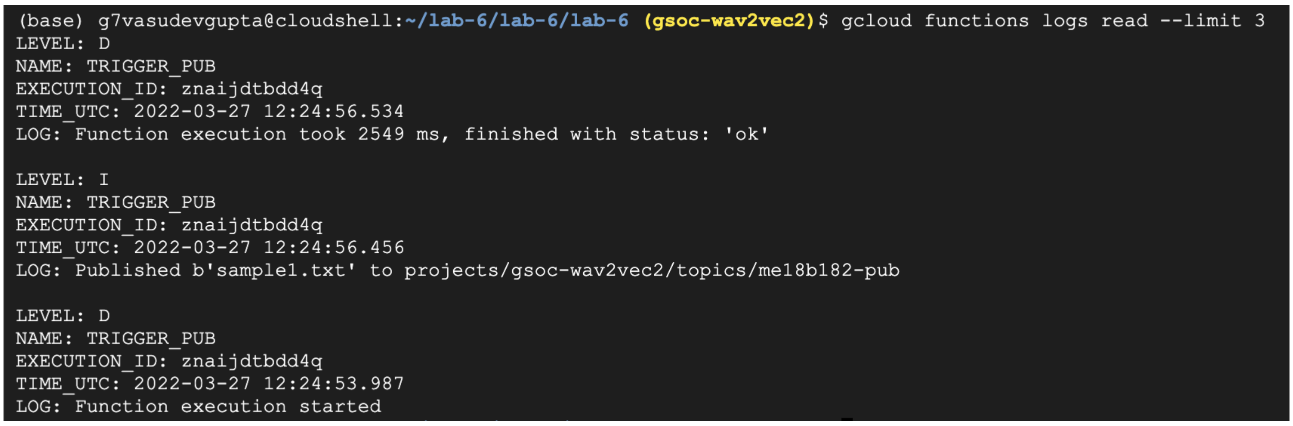
Filter by name prefix only ▾ Filter Filter objects and folders Show deleted data

☐	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
☐	sample1.txt	607 B	text/plain	Mar 27, 2…	Standard	Mar 27, 20…	Not public	–	Google-manag



```
(base) g7vasudevgupta@cloudshell:~/lab-6/lab-6/lab-6 (gsoc-wav2vec2)$ python3 sub.py
Listening for messages on projects/gsoc-wav2vec2/subscriptions/me18b182-sub..

Received file: sample1.txt
Number of lines in sample1.txt: 4
```



```
(base) g7vasudevgupta@cloudshell:~/lab-6/lab-6/lab-6 (gsoc-wav2vec2)$ gcloud functions logs read --limit 3
LEVEL: D
NAME: TRIGGER_PUB
EXECUTION_ID: znaijdtbdd4q
TIME_UTC: 2022-03-27 12:24:56.534
LOG: Function execution took 2549 ms, finished with status: 'ok'

LEVEL: I
NAME: TRIGGER_PUB
EXECUTION_ID: znaijdtbdd4q
TIME_UTC: 2022-03-27 12:24:56.456
LOG: Published b'sample1.txt' to projects/gsoc-wav2vec2/topics/me18b182-pub

LEVEL: D
NAME: TRIGGER_PUB
EXECUTION_ID: znaijdtbdd4q
TIME_UTC: 2022-03-27 12:24:53.987
LOG: Function execution started
```

Following code is used to trigger the cloud function upon adding file to bucket:

```python
def TRIGGER_PUB(data, context):
    from google.cloud import pubsub_v1

    project_id = "gsoc-wav2vec2"
    topic_id = "me18b182-pub"

    # Create a publisher client to topic
    publisher = pubsub_v1.PublisherClient()
    topic_path = publisher.topic_path(project_id, topic_id)

    # Publish path to file in the topic & Data must be a bytestring
    pub_data = data['name'].encode("utf-8")

    # Client returns future when publisher publish.
    future = publisher.publish(topic_path, pub_data)

    print(f"Published {pub_data} to {topic_path}")
```

Following is the code contained in ***sub.py*** :

```python
project_id = "gsoc-wav2vec2"
subscription_id = "me18b182-sub"

# Create a subscriber client
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(project_id, subscription_id)

def callback(message):
    # Decode the filename back to string
    filename = message.data.decode("utf-8")
    print(f"Received file: {filename}")

    # Create a google storage client to read the uploaded file from bucket.
    storage_client = storage.Client()
    bucket_name = "cs4830-trigger-gcf"
    bucket = storage_client.get_bucket(bucket_name)

    blob = bucket.blob(filename).download_as_string().decode('utf-8')

    # Splits the text based on \n
    lines = blob.split('\n')
    print(f"Number of lines in {filename}: {len(lines)}")

    message.ack()

# Default subscriber is a pull subscriber
pull_sub_future = subscriber.subscribe(subscription_path, callback=callback)
print(f"Listening for messages on {subscription_path}..\n")

try:
    # The subscriber listens indefinitely
    pull_sub_future.result()
except KeyboardInterrupt:
    pull_sub_future.cancel()
```

2) A subscription can use either the pull or push mechanism for message delivery.

Pull Subscription - In this system, the subscriber application initiates requests to the Pub/Sub server to retrieve messages.

1. The pull method is called explicitly by the subscribing application, which requests that messages be sent.
2. The message (or an error if the queue is empty) and an ack ID are returned by the Pub/Sub server.
3. The subscriber expressly invokes the acknowledge method and acknowledges receipt using the returned ack ID.

Push Subscription - In push delivery, Pub/Sub initiates requests to your subscriber application to deliver messages.

1. Each message is sent to the subscriber application as an HTTPS request to a pre-configured endpoint by the Pub/Sub server.
2. The message is acknowledged by the endpoint, which returns an HTTP success code. A message should be resent if it receives a non-success answer.

Scenarios where pull subscription is favourable -

- When there is a large volume of messages.
- The efficiency and throughput of message processing are critical.
- Public HTTPS endpoint, with a non-self-signed SSL certificate, is not feasible to setup

Scenarios where push subscription is favourable -

- Multiple topics must be processed by the same webhook.
- App Engine Standard and Cloud Functions subscribers.
- Environments where Google Cloud dependencies (such as credentials and the client library) are not feasible to set up.