CS4830 Big Data Lab Assignment 4

Shashank H S BE18B006

1 a) The text file is downloaded and saved as “lab6_data.txt”. It is then uploaded to google cloud shell from where it is copied to the bucket “shashank_be18b006”.

b) The topic and subscription were created using the commands as shown below:

```
CLOUD SHELL
Terminal (big-data-lab-345012)                                  Open Editor

shashankhiremath73@cloudshell:~ (big-data-lab-345012)$ gcloud pubsub topics create pubsub-lab6
Created topic [projects/big-data-lab-345012/topics/pubsub-lab6].
shashankhiremath73@cloudshell:~ (big-data-lab-345012)$
```

```
shashankhiremath73@cloudshell:~ (big-data-lab-345012)$ gcloud pubsub subscriptions create linecount-sub --topic pubsub-lab6
Created subscription [projects/big-data-lab-345012/subscriptions/linecount-sub].
shashankhiremath73@cloudshell:~ (big-data-lab-345012)$
```

Next, the google cloud function is created which gets triggered when a file is added to the bucket. The main.py and requirements.txt files are in the folder.

“data” and “context” are the inputs to the GCF. “data” is a dictionary which contains the name of the bucket, the name of the file that was added to the bucket, etc. “context” acts like metadata of the event. When triggered, the GCF publishes the name of the file that was added into the bucket into “pubsub-lab6”.

The GCF is deployed as shown below:

```
CLOUD SHELL
Terminal (big-data-lab-345012)                                  Open Editor

shashankhiremath73@cloudshell:~ (big-data-lab-345012)$ nano requirements.txt
shashankhiremath73@cloudshell:~ (big-data-lab-345012)$ nano main.py
shashankhiremath73@cloudshell:~ (big-data-lab-345012)$ gcloud functions deploy publish \
> --runtime python39 \
> --trigger-resource shashank_be18b006 \
> --trigger-event google.storage.object.finalize
Deploying function (may take a while - up to 2 minutes)...working..
For Cloud Build Logs, visit: https://console.cloud.google.com/cloud-build/builds;region=us-central1/5ca96d11-930b-48d1-880e-d458eee272d1?project=240985245922
Deploying function (may take a while - up to 2 minutes)...done.
availableMemoryMb: 256
buildId: 5ca96d11-930b-48d1-880e-d458eee272d1
buildName: projects/240985245922/locations/us-central1/builds/5ca96d11-930b-48d1-880e-d458eee272d1
dockerRegistry: CONTAINER_REGISTRY
entryPoint: publish
eventTrigger:
  eventType: google.storage.object.finalize
  failurePolicy: {}
  resource: projects/_/buckets/shashank_be18b006
  service: storage.googleapis.com
ingressSettings: ALLOW_ALL
labels:
  deployment-tool: cli-gcloud
name: projects/big-data-lab-345012/locations/us-central1/functions/publish
runtime: python39
serviceAccountEmail: big-data-lab-345012@appspot.gserviceaccount.com
sourceUploadUrl: https://storage.googleapis.com/gcf-upload-us-central1-2ba69d99-f402-42fb-b147-c25f1c83b3a7/1651ce2d-405e-4664-a975-df856a8bbfc5.zip
status: ACTIVE
timeout: 60s
updateTime: '2022-03-27T13:57:19.037Z'
versionId: '1'
```

A sample file “data.txt” was uploaded to check if the GCF is functioning as expected.

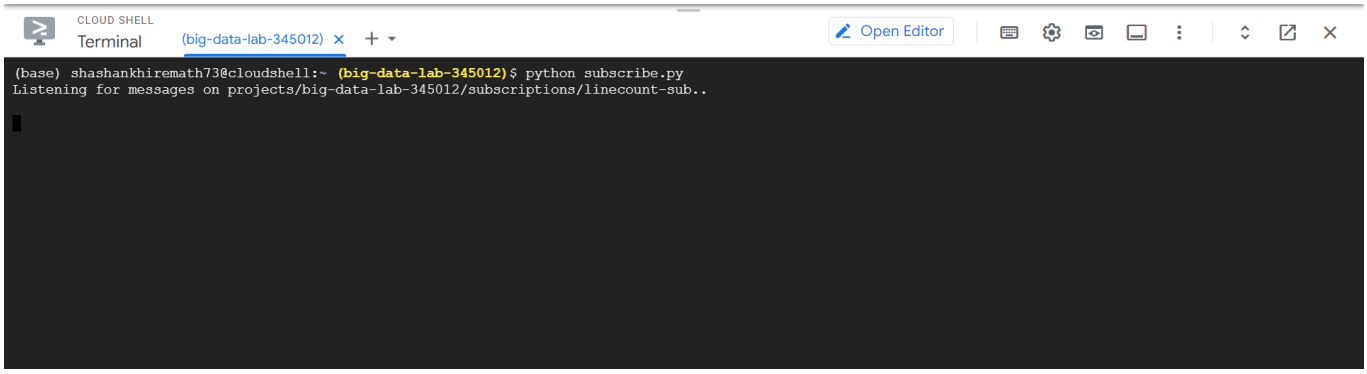
```
2022-03-27T14:11:47.921760Z  Cloud Functions  CreateFunction  us-central1:publish  shashankhiremath73@gmail.com  {@type: type.googleapis.com/google.cloud.audit.AuditLog, authenticat…
2022-03-27T14:13:02.034778380Z  publish  6urny2fts45z  Function execution started
2022-03-27T14:13:04.717Z  publish  6urny2fts45z  4272600276264025
2022-03-27T14:13:04.717Z  publish  6urny2fts45z  Published b'data.txt' to projects/big-data-lab-345012/topics/pubsub-lab6
2022-03-27T14:13:04.719601610Z  publish  6urny2fts45z  Function execution took 2687 ms, finished with status: 'ok'
No newer entries found matching current filter.
```

The logs in the Cloud Function GUI show that the GCF is working and the file name is published to the topic “pubsub-lab6”.

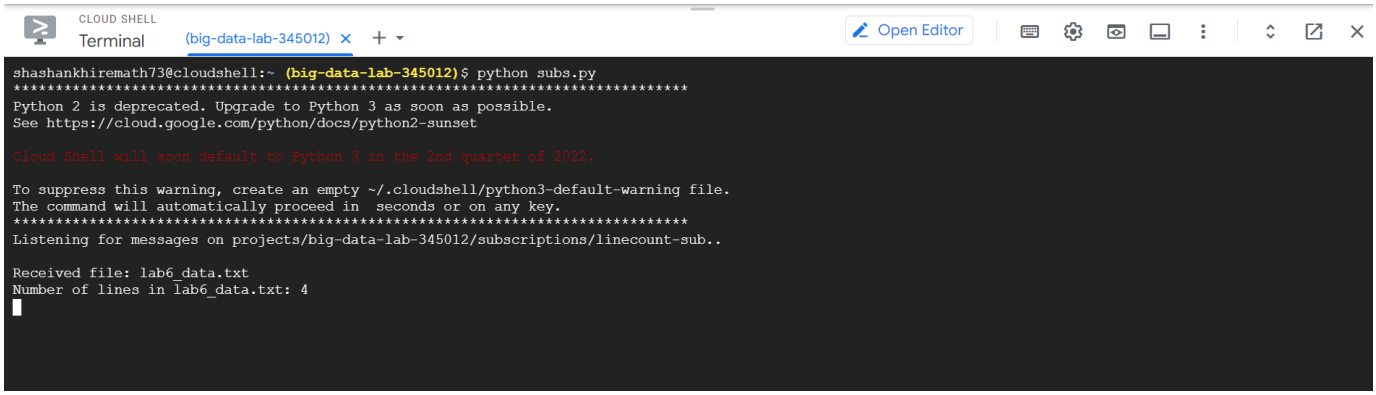
c) The subscription “linecount-sub” that was created in 1 b) is a pull subscription by default. Next, we create the subscription python file, “subs.py”, which subscribes to “pubsub-lab6” and prints the number of lines present in the file added to the bucket.

The callback function is the important part of “subs.py”. The input to this function is the message the subscriber is listening to. This is the function that processes the message. Google storage client is used to read the file from the bucket using the file name that is published.

To enable real time printing of number of lines, the subscriber is made to listen to messages in “pubsub-lab6” indefinitely.



After triggering the GCF, the file name is published to “pubsub-lab6”. The subscriber listens to this message in real-time and prints the number of lines.



To stop the subscriber to listen, we should interrupt it using keyboard.

2.

Pull subscription	Push subscription
In pull subscription, the subscriber requests the Pub/Sub server for message delivery. Hence, the endpoint can be any device on the internet with authorized credentials to call Pub/Sub API.	In push subscription, the Pub/Sub server sends an HTTPS request to the subscriber. Since the server initiates the request, the endpoint must be reachable via a DNS name and must have an SSL certificate installed.

After this request, the Pub/Sub server returns a message and an acknowledgement ID. Once the subscriber receives the acknowledgement ID, it acknowledges receipt by using the acknowledgement ID to call the acknowledgement method. If there are no messages, the Pub/Sub server would send an error.	On receiving the message, the subscriber acknowledges it by returning a HTTP success code to the server. If the success code is not returned, the Pub/Sub server will resend the message.
Multiple subscribers can access the same shared pull subscription where each subscriber will receive a subset of the messages.	The push endpoint balances the load of the server to accommodate more topics.
The acknowledgement deadline can be altered by the subscriber to allow message processing time to be arbitrarily long.	The Pub/Sub server controls the flow of messages by itself.

Pull subscription is preferred over push subscription when large volume of messages should be processed with high efficiency and throughput or when it is not possible to pre-configure the endpoint to be reachable via DNS name with non-self-signed SSL certificate.

Push subscription is preferred over pull subscription when multiple topics must be processed by the same endpoint or when client libraries and credentials are not feasible to set up at the endpoint (example: if GCF was required for subscribers).