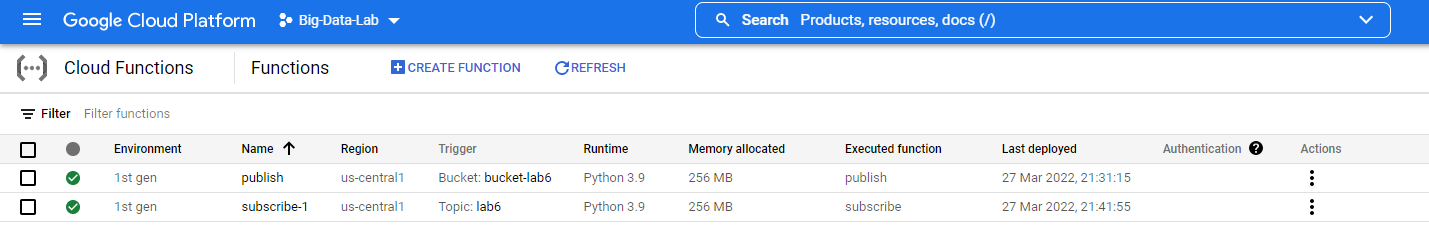
1. Two cloud functions, one each for part b and c were created.

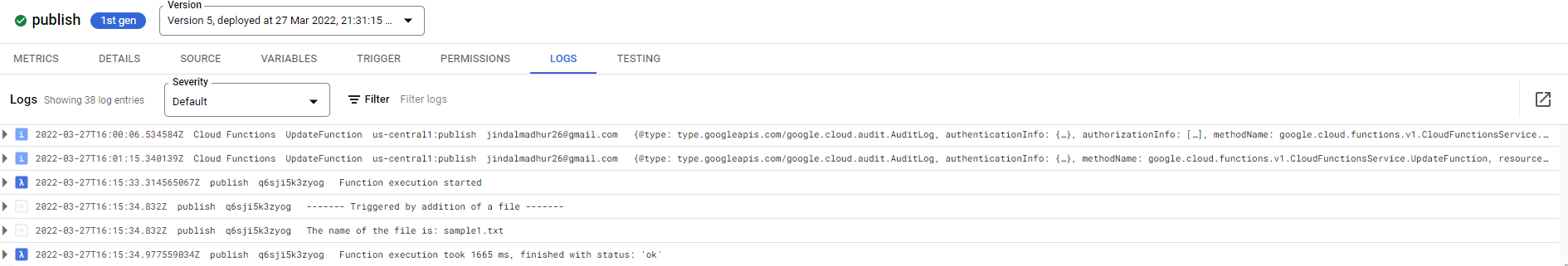


1. Publish.py

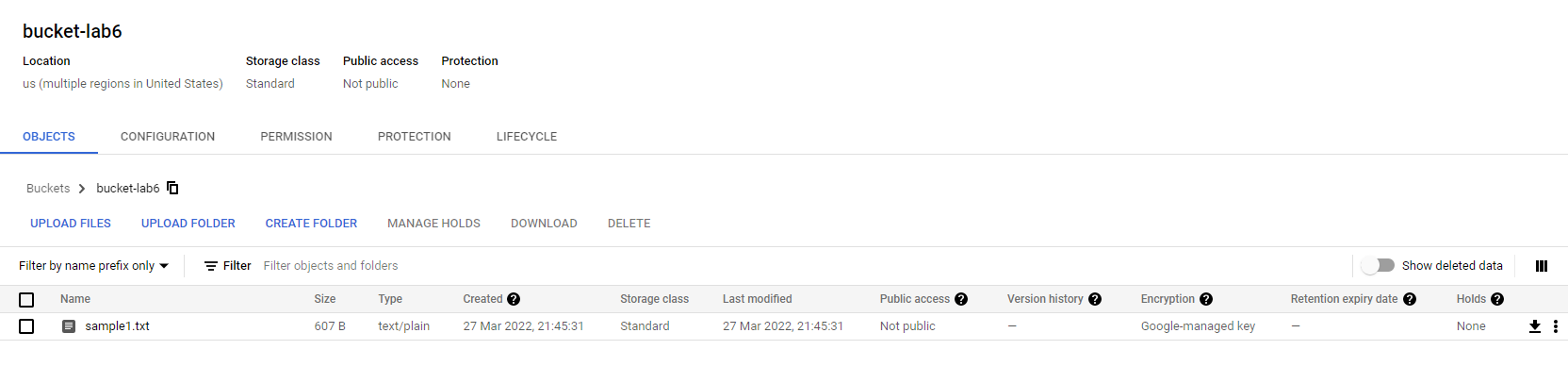


Herein, we start with importing pubsub\_v1 for publishing a message, which would be use to trigger the second function. This function is triggered on the addition of a new file to the bucket and proceeds to read the file name from the trigger function, and then uses the publisher to publish a message to the given topic. The message is sent as a bytestring, wherein the string object of file\_name is converted to a bytestring using the bytes function.

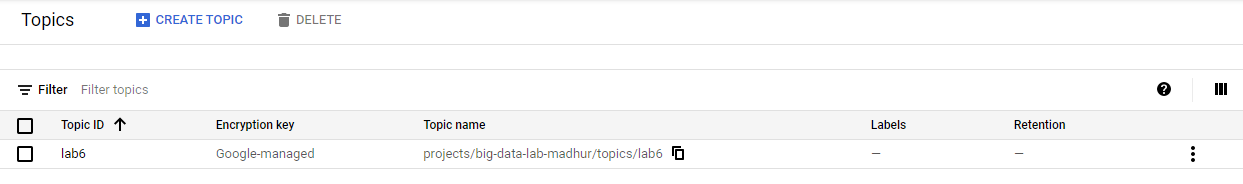
The output logs are given below:



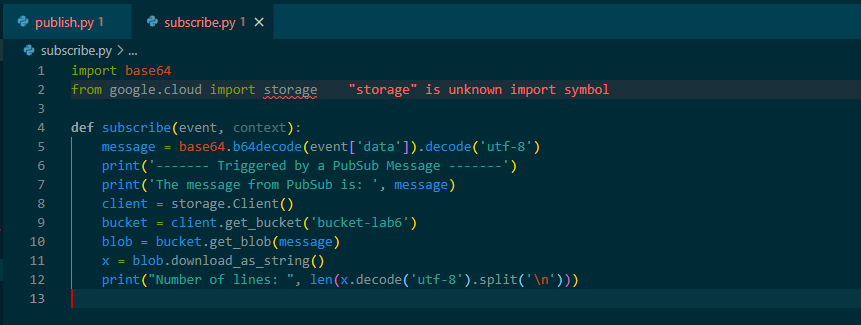
We see that the file name is the same name of the file as we uploaded in the bucket. Hence our function is working properly. The screenshot of the bucket is provided below.



The screenshot for the topics is provided below:



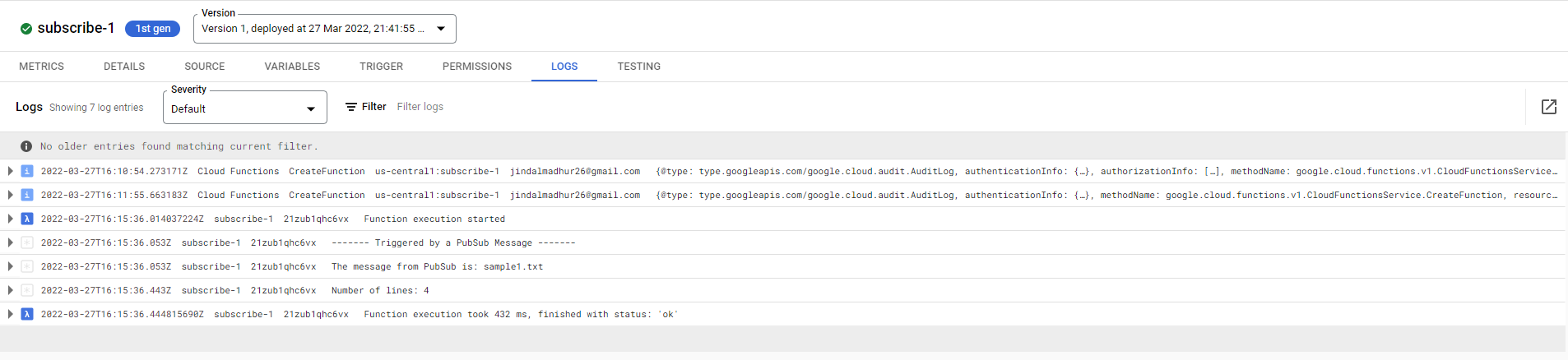
1. Subscribe.py



This function is triggered by any PubSub message in the given topic. Thus, as we upload the file in the bucket, the Publish function publishes a message and hence triggers the subscribe function.

The subscribe function takes decodes the base64 encoded message and hence gets the file name. This is used to read the file and finally compute the number of lines in the file.

The logs for Subscribe function are provided below:



We see that the message is same as the file name that was uploaded in the bucket. Hence our function is working properly and gives us the line count as 4.

1. The two types of subscribers are:

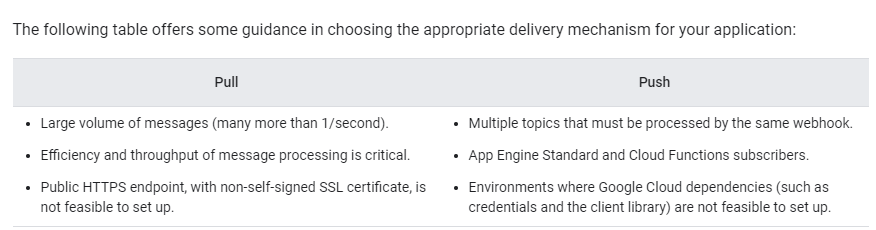
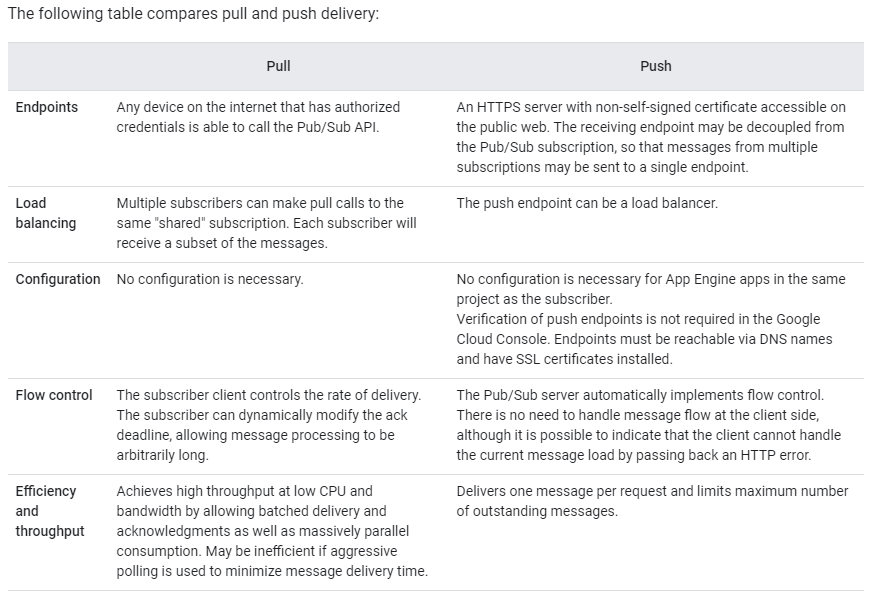
**Pull Subscriber:** In pull delivery, your subscriber application initiates requests to the Pub/Sub server to retrieve messages.

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

**Push Subscriber:** In push delivery, Pub/Sub initiates requests to your subscriber application to deliver messages.

1. The Pub/Sub server sends each message as an HTTPS request to the subscriber application at a pre-configured endpoint.
2. The endpoint acknowledges the message by returning an HTTP success status code. A non-success response indicates that the message should be resent.

Pub/Sub dynamically adjusts the rate of push requests based on the rate at which it receives success responses.

An example for pull subscriber will be using a search engine wherein people would want to pull different information and hence only need the info when the request is put through, and thus pull all the info needed to show the desired results.

An example of a push subsriber is a Notification manager which is pushed the required message and thus acts on it according to the info of the message and serves multiple topics.