

Report

Assignment 5

Cloud Functions and Pub/Sub Messaging Service

For this assignment, I have implemented a text to pdf converter and used **Google Cloud Functions** to host the different services, **Google Pub/Sub** to establish communications among them and **Google Cloud Storage** to cache intermediate results . The application is hosted on homes.sice.indiana.edu and the languages used for (*Link provided in the next section*)

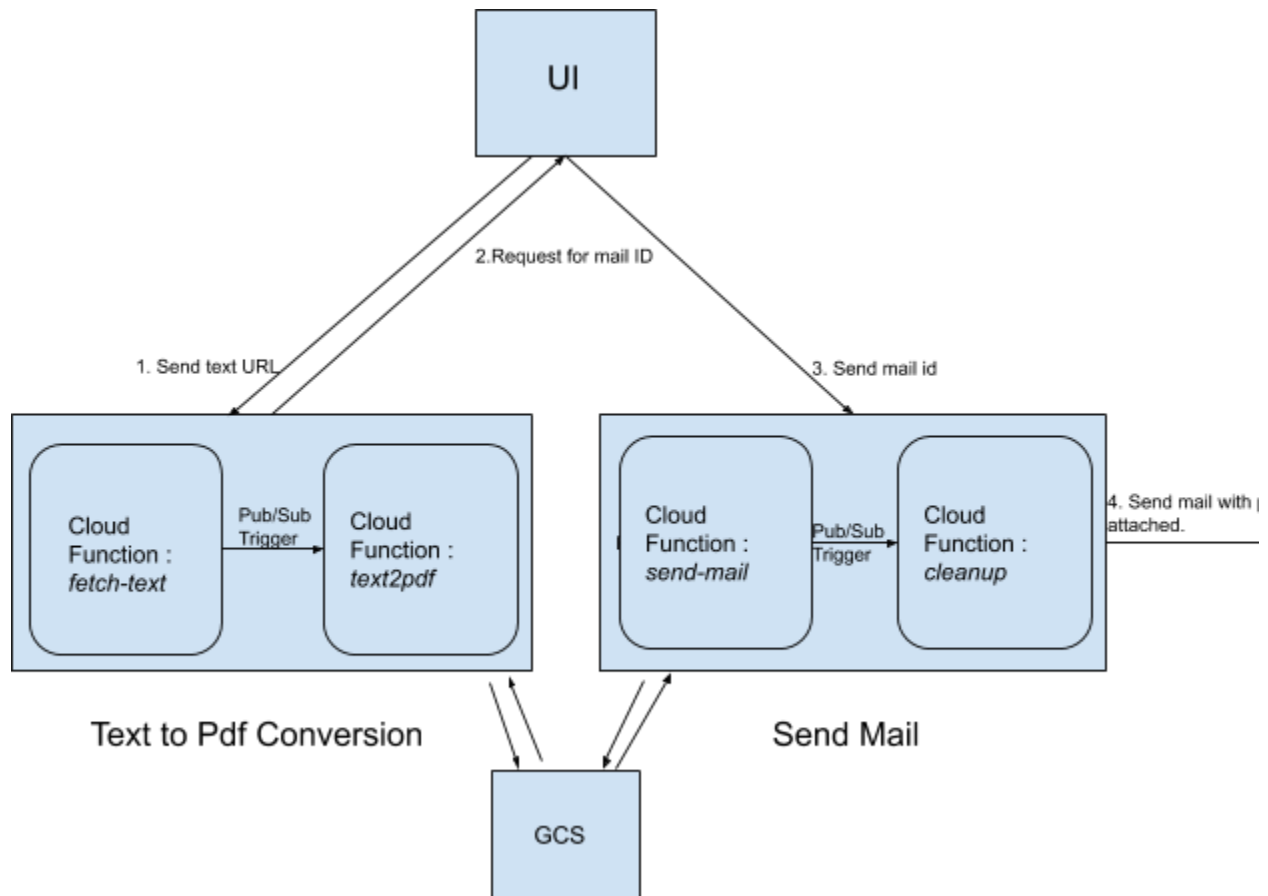
Architecture and Pseudocode

In brief, the following steps are performed when a user visits the application (Pub/Sub topics are highlighted in green and cloud functions are highlighted in blue):

1. User will enter a URL to text file (ends with .txt)
2. A JavaScript function will send HTTP trigger the cloud function, **fetch-text**.
 - a. **fetch-text** simply fetches the text file from the URL and dumps it into a storage bucket.
 - b. Then a message containing the name of this dumped text is sent to Pub/Sub topic **text_dumped**. Once the message is published, a trigger is sent to the cloud function, **text2pdf** which has subscribed to this topic.
 - i. **text2pdf** fetches the text from the bucket and performs the conversion.
 - ii. After conversion, the pdf is dumped into the bucket.
 - c. After publishing the message to **text_dumped**, the cloud function finishes execution and notifies the JS script that the file is converted.
3. The user is notified of this and is requested an email id to which the pdf will be sent.
4. A JavaScript function will send an HTTP trigger with the email id to the cloud function, **send-mail**.
 - a. **send-mail** fetches the pdf from the bucket and attaches it to the mail and sends it to the user.
 - b. Now the pdf and text files can be deleted from the bucket. So a message is published with the names of the files to be deleted to the Pub/Sub topic **cleanup**. This will trigger the final cloud function **cleanup**.
 - i. **cleanup** will delete the pdf and text files from the storage bucket
 - c. After **cleanup** has been notified, **send-mail** finishes execution and notifies the JS script that email has been sent.
5. User is alerted that the pdf has been mailed to them.

To try out the application, use the link: <https://homes.luddy.indiana.edu/msureshk/text2pdf/>
'**sample_text_url.txt**' contains simple text file links you can use for the application

The diagram given helps to further visualize this architecture.



Mail

Python's email.mime modules were used to create mails and smtpplib was used to send the mail. Below is a screenshot of how the mail will be received in your inbox

[External] Here's your PDF

← REPLY ← REPLY ALL → FO



text2pdfconverter@gmail.com

Sat 12/12/2020 06:47

To: ☐ Suresh Kumar, Manisha;

📎 1 attachment



This message was sent from a non-IU address. Please exercise caution when clicking links or opening attachments from external sources.

PFA your PDF

The 'sample.pdf' attachment as seen in the mail shown in the screenshot is provided in the submission with the same name. The pdf contains text found in <http://www.sci.utah.edu/~macleod/docs/txt2html/sample.txt>

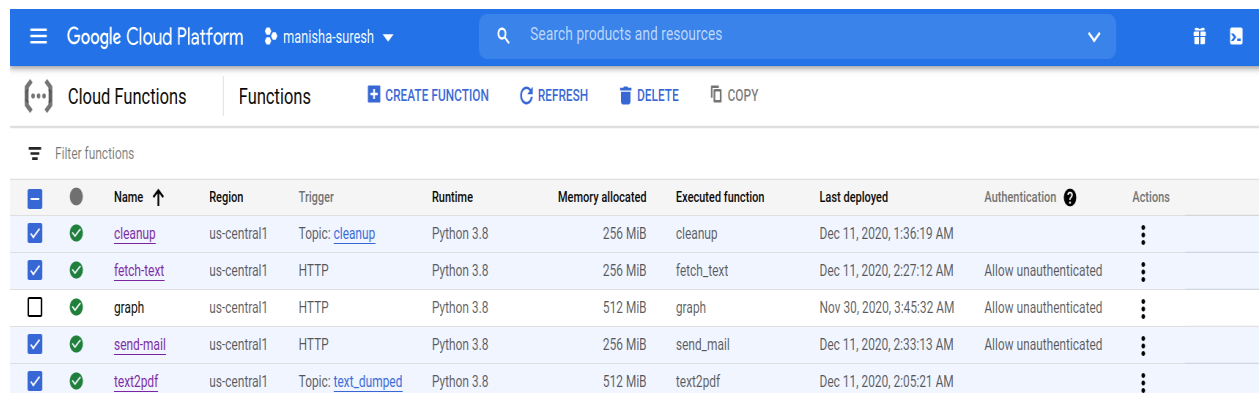
Attempts Made with Signed URL

At this point, I feel its best to mention an attempt I made with generating signed URLs to objects in cloud storage. A signed URL provides access to a particular object for a specified amount of time. My plan was to use this URL and have the user directly upload and download the text and pdf files respectively directly from storage.

Although I was successful in being able to generate a signed URL using Google Cloud Storage API's `generate_signed_url()`, I received CORS errors while sending requests to it through my JS script. After researching this, I realized that a bucket sub-subdomain style URL (`bucket.storage.googleapis.com`) is expected and that will send the right CORS headers. The path variant (`storage.googleapis.com/bucket`) generated by `generate_signed_url()` does not serve the right headers. [This GitHub](#) discussion talks in more technical detail about this issue.

Services and Code Snippets

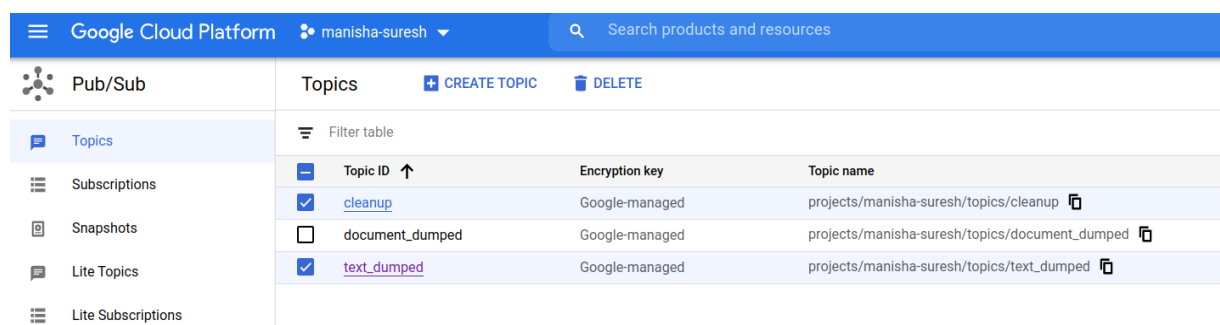
1. In all, there are 4 cloud functions. Two are HTTP triggered and two are triggered by Pu/Sub topics. The highlighted functions in the screenshot below are the functions used for this assignment.



The screenshot shows the Google Cloud Platform interface for the 'manisha-suresh' project. The 'Functions' tab is selected, displaying a table of cloud functions. The functions are listed with their names, regions, triggers, runtimes, memory allocations, and last deployment times. The functions 'cleanup', 'fetch-text', 'send-mail', and 'text2pdf' are highlighted with a blue background.

Name	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentication	Actions
cleanup	us-central1	Topic: cleanup	Python 3.8	256 MiB	cleanup	Dec 11, 2020, 1:36:19 AM		
fetch-text	us-central1	HTTP	Python 3.8	256 MiB	fetch_text	Dec 11, 2020, 2:27:12 AM	Allow unauthenticated	
graph	us-central1	HTTP	Python 3.8	512 MiB	graph	Nov 30, 2020, 3:45:32 AM	Allow unauthenticated	
send-mail	us-central1	HTTP	Python 3.8	256 MiB	send_mail	Dec 11, 2020, 2:33:13 AM	Allow unauthenticated	
text2pdf	us-central1	Topic: text_dumped	Python 3.8	512 MiB	text2pdf	Dec 11, 2020, 2:05:21 AM		

There are two Pub/Sub topics both of which are highlighted below



The screenshot shows the Google Cloud Platform interface for the 'manisha-suresh' project. The 'Pub/Sub' tab is selected, displaying a table of topics. The topics 'cleanup' and 'text_dumped' are highlighted with a blue background.

Topic ID	Encryption key	Topic name
cleanup	Google-managed	projects/manisha-suresh/topics/cleanup
document_dumped	Google-managed	projects/manisha-suresh/topics/document_dumped
text_dumped	Google-managed	projects/manisha-suresh/topics/text_dumped

- Below is an example of how `fetch-text` publishes a message, specifically, the name of the text file that needs to be converted to `text_dumped`.

```
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(project_id, topic_id)
messageID = publisher.publish(topic_path,
                              destination_blob_name.encode("utf-8"))
```

'destination_blob_name' is the name of the text file.

- Similarly, `send-mail` publishes the name of the pdf to `cleanup`.

```
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(project_id, topic_id)
messageID = publisher.publish(topic_path,
                              pdf.encode("utf-8"))
```

Using Google Pub/Sub

Google defines Pub/Sub as *"Pub/Sub is an asynchronous messaging service that decouples services that produce events from services that process events."*

When using multiple cloud functions that have a flow among them, each function has to call the next function and has to wait till the second function executes for it to finish executing itself. Using Pub/Sub, the first function, in our case, `fetch-text` will just send a message to the corresponding Pub/Sub topic, `text_dumped` and finish execution. `text2pdf` which has subscribed to this topic will be notified immediately and it starts its execution separately and unrelated to `fetch-text`.

This is helpful for this application since the user experiences no delay and can move on with providing their mail id while their pdf is being generated.

The `send-mail` → `cleanup` → `cleanup` system also follows the same mechanism. As soon as the mail is sent, the user is notified and doesn't have to wait till the files are removed from the cloud storage. The Pub/Sub topic takes care of it.

Logs

Logs for two executions of the applications can be found in the below files numbered in the order in which they were generated

1. 1_fetch_text.csv
2. 2_text2pdf.csv
3. 3_send_mail.csv
4. 4_cleanup.csv