In this homework you should recreate the work from HW3 but use virtual machines instead of cloud functions

1. For your first app write a web server in python that can accept HTTP GET requests from web clients. The app should be able to respond to requests for the files in your bucket that you created in homework 2 and return the contents of the requested file along with a 200-OK status. You should run this webserver on a VM instance for which you have configured a static IP address. You should start this web server automatically upon VM creation.
2. Requests for non-existent files should return a 404-not found status. Such erroneous requests should be logged to cloud logging.
3. Requests for other HTTP methods (PUT, POST, DELETE, HEAD, CONNECT, OPTIONS, TRACE, PATCH) should return a 501-not implemented status. Such erroneous requests should be logged to cloud logging.
4. Demonstrate the functionality of your app by using the provided http client to request a few hundred of your cloud storage files. Run your http client on another VM, different from the one using the server.
5. Use the curl command line utility to demonstrate the functionality of your app with respect to the 404 and 501 use cases
6. Use a browser to demonstrate one request for each of the aforementioned response status cases.
7. Use your previous created second app to track requests from banned countries. The US defines a list of countries (North Korea, Iran, Cuba, Myanmar, Iraq, Libya, Sudan, Zimbabwe and Syria) to which export of sensitive cryptographic material is prohibited. We will pretend that files stored in your storage bucket contain such materials. The first app should communicate such “forbidden” requests to the second app which should print an appropriate error message to its standard output.
8. Your second app must run on a 3rd VM instance.
9. Start your python web server on the smallest VM possible and use more than one instance of your client to stress test it. After how many concurrent clients does it start returning errors due to being unable to serve the requests (ignore the 400 (permission denied) error as those do not indicate inability to serve requests). Be careful when you set up this stress test to avoid spending too much money! Report how much money you spent running it.

What to turn in:

- The python code for your first and second apps as a github link
- A pdf file describing all the necessary steps to configure and run your apps, including screenshots of your browser work from step #6, and the contents of cloud logging for the erroneous requests. Also a description of your findings when stress testing your server.