Github Link: nirbhay221/CDS-DS-561-hw2 (github.com)

Project Name: CDSDS561 Project 1 **Project ID**: cdsds561-project-1

Bucket Name: hw2nirbhgsutil

Service Account Name:

nirbhhwbucketpermission@cdsds561-project-1.iam.gserviceaccount.com

Directory Name: test-dir/

In order to generate 10,000 files for the following project with each file consisting of a maximum of 250 links, we need to first run the generate_content.py. Also, we need to modify the generate_content.py file and add random.seed(0) which is used for making random number generation in code predictable and reproducible which can be beneficial for debugging, testing and ensuring consistent results when generating random files. It ensures that the sequence of random numbers generated by the program is the same every time you try to run, which is essential for reproducibility.

generate_content.py code:

```python
#!env python3
import argparse
import random
from google.cloud import storage
from google.oauth2 import service_account

random.seed(0)

def add_text(f):
  text = "Lorem ipsum dolor sit amet, \
consectetur adipiscing elit, sed do \
eiusmod tempor incididunt ut labore \
et dolore magna aliqua. Ut enim ad\n\
minim veniam, quis nostrud exercitation \
ullamco laboris nisi ut aliquip ex ea \
commodo consequat. Duis aute irure dolor \
in reprehenderit in voluptate velit esse\n\
cillum dolore eu fugiat nulla pariatur. \
Excepteur sint occaecat cupidatat non \
proident, sunt in culpa qui officia \
deserunt mollit anim id est laborum.\n<p>\n"
  f.write(text)

def add_headers(f):
  text = "<!DOCTYPE html>\n\
<html>\n\
<body>\n"
  f.write(text)


def add_footers(f):
  text = "</body>\n\
</html>\n"
  f.write(text)

def add_link(f, lnk):
  text = "<a HREF=\""
  f.write(text)
  text = str(lnk) + ".html\""
  f.write(text)
  text = "> This is a link </a>\n<p>\n"
  f.write(text)

def generate_file(idx, max_refs, num_files):
  fname = str(idx) + ".html"
  with open(fname, 'w', encoding="utf-8") as f:
    # how many references in this file
    add_headers(f)
    num_refs = random.randrange(0,max_refs)
    for i in range(0,num_refs):
      add_text(f)
      lnk = random.randrange(0,num_files)
      add_link(f, lnk)
    add_footers(f)
    f.close()

def main():
  parser = argparse.ArgumentParser()
  parser.add_argument('-n', '--num_files', help="Specify the number of files to generate", type=int, default=10000)
  parser.add_argument('-m', '--max_refs', type=int, help="Specify the maximum number of references per file", default=250)
  args = parser.parse_args()
  credentials = service_account.Credentials.from_service_account_file("C:/Users/nirbh/Downloads/cdsds561-project-1-3185d9128a0f.json")
  storage_client = storage.Client(project = "cdsds561-project-1",credentials = credentials)
  gcs_bucket_name= "hw2nirbhgsutil"
  gcs_bucket = storage_client.bucket(gcs_bucket_name)
  print(args.num_files, args.max_refs)
  for i in range(0,args.num_files):
    generate_file(i, args.max_refs, args.num_files)

if __name__ == "__main__":
  main()
```

Now In order to generate 10,000 files with each file consisting of maximum of 250 links, we can use the following command:

generate command:

```
C:\Users\nirbh\AppData\Local\Google\Cloud SDK>python generate-content-check.py -n 10000 -m 250
```

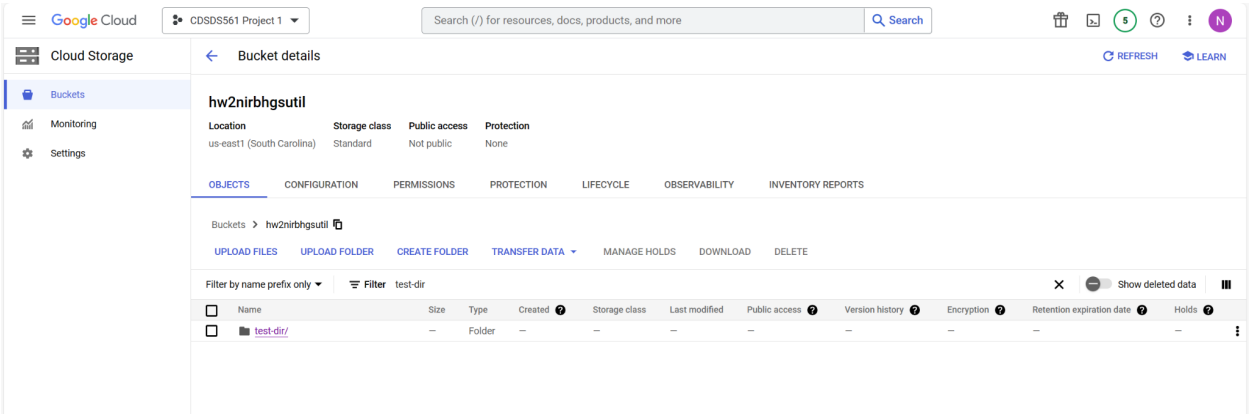
Once all of the 10,000 files are generated, we need to copy those files created on our local system to the google cloud storage bucket's directory using the following command :

Copy command:

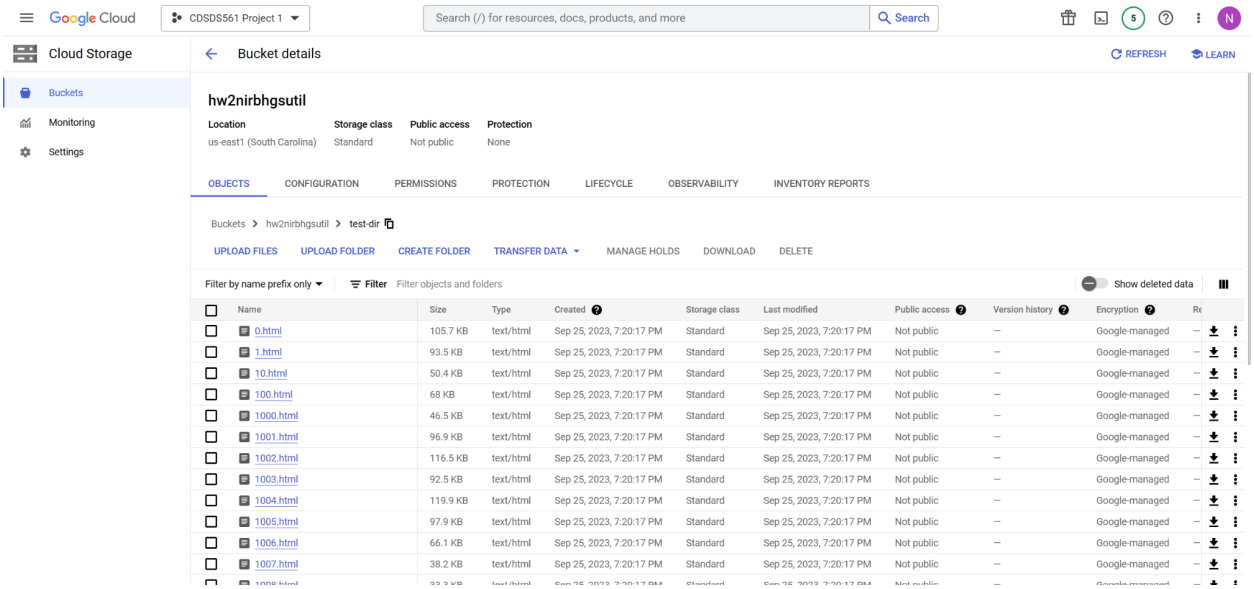
```
C:\Users\nirbh\AppData\Local\Google\Cloud SDK>gsutil -m cp *.html gs://hw2nirbhgsutil/test-dir
```

Since the directory doesn't exist as of now, it will be created. Other than that we use '-m' for enabling parallel transfers which can speed up the copying process when dealing with multiple files. Using the following command we can copy the following 10, 000 files from your local file system to the bucket's directory on the google cloud storage.

Test Directory in Bucket:

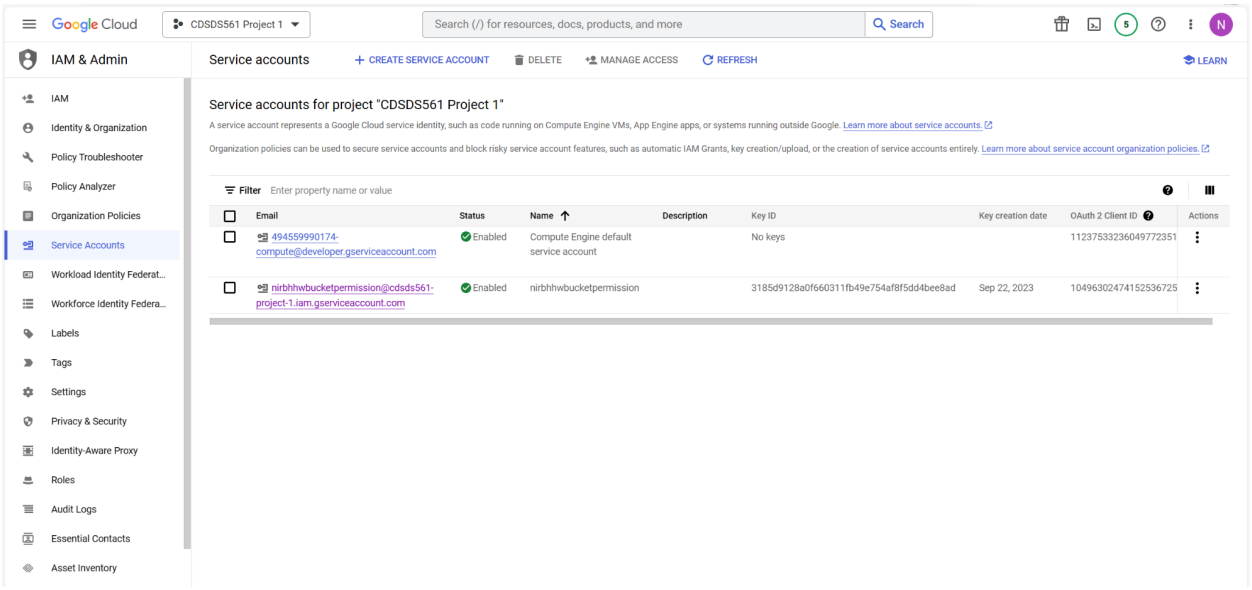


Test Directory Files Created:

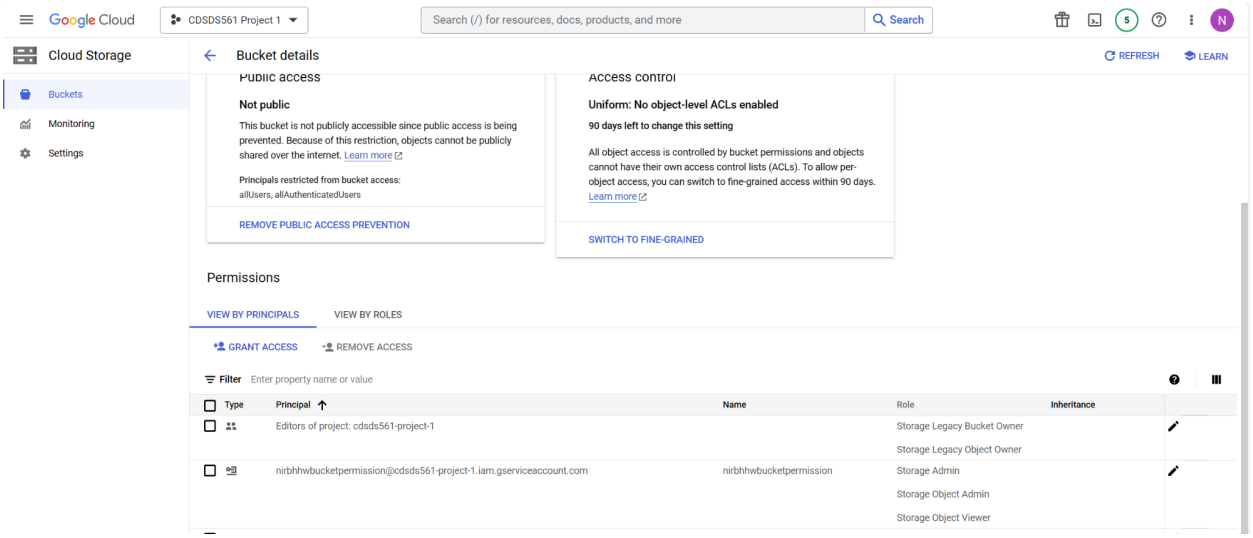


Now, after the following we need to add a service account and we need to modify the bucket's permission so that service account can have access to the bucket. We need to assign a role to the service account as a storage admin, storage object admin and the storage object viewer.

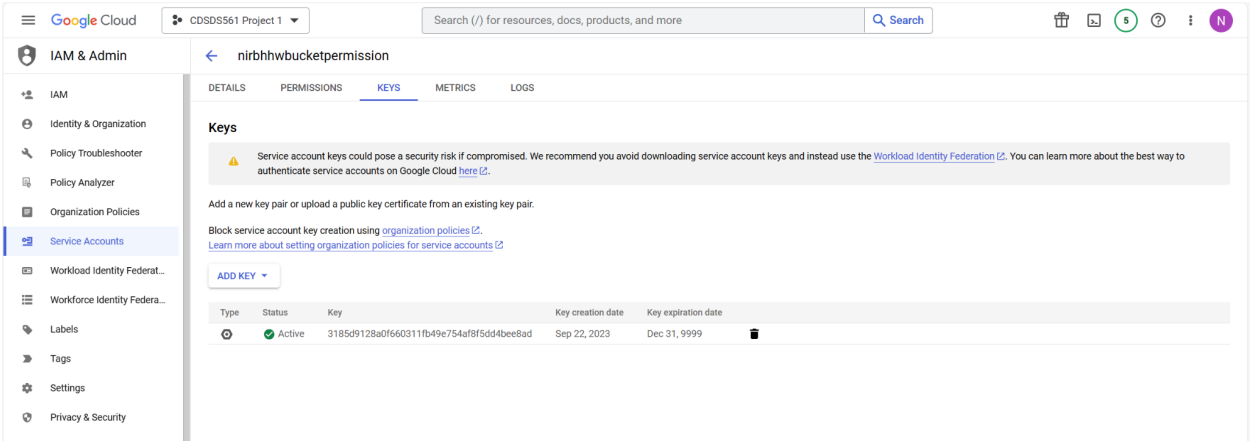
Service Account Creation :



Modifying Bucket's Permissions :



Other than this, we need to create the key for the service account to use the following key as credential in the code for parsing the 10,000 files which reside in the test-dir directory. And we need to store the following key on our local file system.



PROGRAM SUMMARY:

In the following code, first we parse the command line arguments for accessing the GCS buckets and directory for analyzing the files inside the directory. After the following, google cloud storage client is set up using the provided service account credentials (service account's key) and then the program initializes store_dictionary, incoming_link, outgoing_link dictionary and pagerank dictionary. The program iterates through all the HTML files in the GCS bucket's 'test-dir' directory and for each file it extracts the incoming links using the analyzing incoming links function and it also counts the incoming links count and these values are stored in 'store_dictionary' and all the incoming links are aggregated in the all_incoming_links dictionary so that we can find which files have what incoming file links.

Similar to the above incoming link analysis, the program iterates through the HTML files in the GCS bucket's "test-dir" directory and for each file it extracts the outgoing links and counts them. The outgoing links count is stored in the "store_dictionary" and for each file their outgoing links and relationship with the file are stored in the "outgoing_links_dictionary".

The program also selects a random file from the store_dictionary to check for all the files and their incoming and outgoing link counts. The program then calculates the statistics for the incoming and outgoing links like the average, median, minimum, maximum and quintiles across all files using the numpy library and without the numpy library.

The program also creates a scatter plot for visualizing the incoming and outgoing link counts for all files where the plot is displayed using Matplotlib (works best for 100 files). Program also creates a dataframe 'link_matrix' for storing incoming and outgoing link counts for each file and incoming and outgoing matrix is saved to the csv file. It creates two other matrices, one for outgoing links and one for incoming links.

outgoing_links_matrix: This matrix represents the outgoing links from each web page. The rows correspond to the source web pages, and the columns correspond to the target web pages. Each cell contains a 1 if there is an outgoing link from the source to the target and 0 otherwise.

incoming_links_matrix: This matrix represents the incoming links to each web page. The rows correspond to the target web pages, and the columns correspond to the source web pages. Each cell contains a 1 if there is an incoming link from the source to the target and 0 otherwise.

The program finally calculates Pagerank values for each file using the iterative pagerank algorithm and the algorithm continues until the sum of pagerank values converges (changes by less than 0.005). And the pagerank for all the files gets stored in the 'pagerank_dictionary' and the program prints the pagerank values of the top 5 pages with the highest pagerank scores.

Running the following code:

For running the following code, you can use the following arguments :

python filename.py bucket-name directory-name

For my case :

python check-cloud-testing.py hw2nirbhgsutil test-dir

```
C:\Users\nirbh\AppData\Local\Google\Cloud SDK>python check-cloud-testing.py hw2nirbhgsutil test-dir
```

The result for the following program are as follows:

```
C:\Users\nirbh\AppData\Local\Google\Cloud SDK>python check-cloud-testing.py hw2nirbhgsutil test-dir
Analyzing incoming links within HTML files in bucket: hw2nirbhgsutil
Analyzing outgoing links within HTML files in bucket: hw2nirbhgsutil
Statistics for Incoming Links:
Average: 123.6451
Median: 124.0
Maximum: 188
Minimum: 82
Quintiles: [114. 121. 126. 133.]

Statistics for Outgoing Links:
Average: 123.6451
Median: 123.0
Maximum: 249
Minimum: 0
Quintiles: [ 49.  98. 149. 198.]
Stats with numpy :
Stats with numpy :
Statistics for Incoming Links without Numpy:
Average: 123.6451
Median: 124.0
Maximum: 188
Minimum: 82
Quintiles: [114, 121, 126, 133]

Statistics for Outgoing Links without Numpy:
Average: 123.6451
Median: 123.0
Maximum: 249
Minimum: 0
Quintiles: [49, 98, 149, 198]
Number of files in store_dictionary: 10000
Link Matrix:
          File Incoming Links Outgoing Links
0     6311.html            126            244
1     6890.html            126            235
2      663.html            109            139
3     4242.html            114            190
4     8376.html            132            127
...         ...            ...            ...
9995  1174.html            109             62
9996  3746.html            113             80
9997  1806.html            107            160
9998  6267.html            132             28
9999  4833.html            117             60

[10000 rows x 3 columns]
Analyzing Page Rank ....
Iteration 22:
Iteration 22 completed.

Top 5 Pages by PageRank:
1. Page: 2526.html, PageRank: 2.251737
2. Page: 6846.html, PageRank: 2.035238
3. Page: 5971.html, PageRank: 1.982706
4. Page: 5778.html, PageRank: 1.971122
5. Page: 1058.html, PageRank: 1.955193
Outgoing Links Matrix:
          6311.html  6890.html  663.html  4242.html  8376.html  7961.html  6634.html  ...  6401.html  3832.html  1174.html  3746.html  1806.html  6267.html  4833.html
```

```
Outgoing Links Matrix:
          6311.html  6890.html  663.html  4242.html  8376.html  7961.html  6634.html  ...  6401.html  3832.html  1174.html  3746.html  1806.html  6267.html  4833.html
6311.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
6890.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
663.html           0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
4242.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
8376.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          1          0
...              ...        ...       ...        ...        ...        ...        ...  ...        ...        ...        ...        ...        ...        ...        ...
1174.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
3746.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
1806.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
6267.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
4833.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0

[10000 rows x 10000 columns]
Incoming Links Matrix:
          6311.html  6890.html  663.html  4242.html  8376.html  7961.html  6634.html  ...  6401.html  3832.html  1174.html  3746.html  1806.html  6267.html  4833.html
6311.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
6890.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
663.html           0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
4242.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
8376.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
...              ...        ...       ...        ...        ...        ...        ...  ...        ...        ...        ...        ...        ...        ...        ...
1174.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
3746.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
1806.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0
6267.html          0          0         0          0          1          0          0  ...          0          0          0          0          0          0          0
4833.html          0          0         0          0          0          0          0  ...          0          0          0          0          0          0          0

[10000 rows x 10000 columns]
```