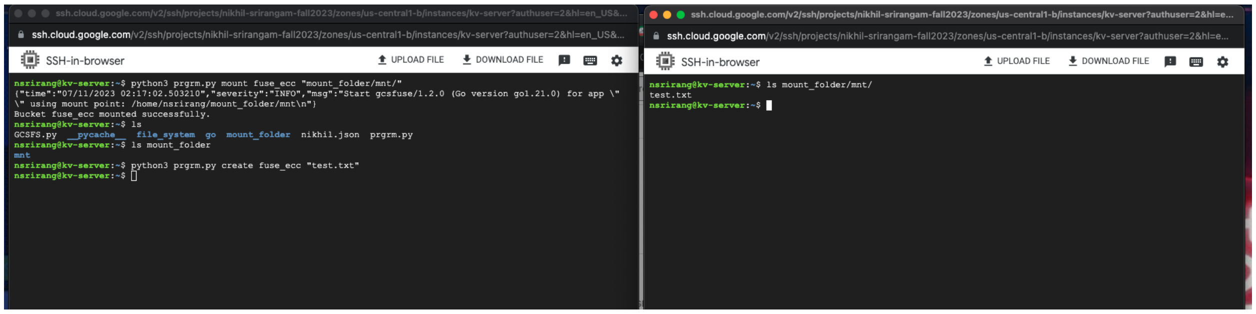
File Systems Using Cloud Storage

This program has been designed to bridge the gap between Google Cloud Storage (GCS), an object storage service, and the traditional file system interface. It achieves this by leveraging the capabilities of FUSE (Filesystem in Userspace) through the `fusepy` library, allowing the program to expose GCS buckets as if they were mounted filesystems on the host machine.
At the core is the `GCSFS` class which inherits from `fusepy`'s `Operations`. This class serves as the filesystem implementation, with methods that correspond to file operations. These methods internally call the GCS APIs to perform actions on the cloud storage. The GCS client is initialized in the constructor, and a reference to the specific GCS bucket is stored for future operations.
The `mount` method within `GCSFS` was particularly interesting to implement as it enables the GCS bucket to be mounted onto a local directory, utilizing the `gcsfuse` command line tool. This creates a seamless integration point where we can interact with cloud storage using familiar file system semantics. File operations such as `create`, `read`, `write`, `truncate`, `unlink`, and `rename` translate the file manipulation commands to their GCS counterparts. Since GCS does not support in-place modifications, the `write` method incorporates a read-modify-write sequence for updates. Similarly, directory-related methods like `mkdir`, `opendir`, `readdir`, and `rmdir` are designed to mimic directory behaviors in GCS's flat namespace by manipulating object prefixes and metadata. I also tried to include a rudimentary error handling mechanism that raises `FuseOSError` exceptions in response to issues, ensuring that errors are communicated back through FUSE to the user in a meaningful way. This integration not only facilitates the interaction with GCS through conventional file operations but also abstracts away the complexities of GCS for end-users. It serves as an intermediate layer, translating POSIX-like file operations to GCS API calls, and thus provides a more intuitive and productive way of working with cloud storage for applications expecting a filesystem interface.

Files
Program - GSCFS.py
Utility program - prgrm.py
Credentials- nikhil.json
Service-account - fuse-527@nikhil-srirangam-fall2023.iam.gserviceaccount.com

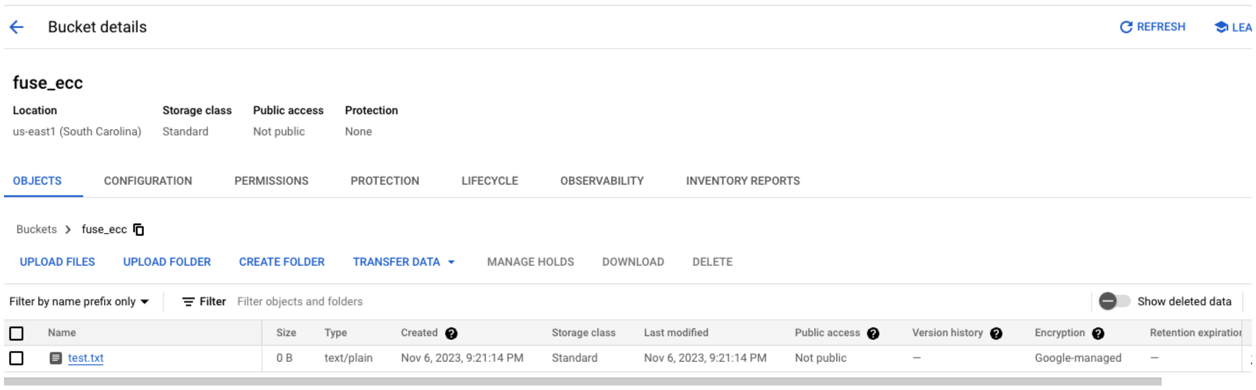
TESTING:

Starting with the testing, first I started implementing the mount command, I ran this command using python subprocess module, and for beginning to check I created a text file using the fuse


ssh.cloud.google.com/v2/ssh/projects/nikhil-srirangam-fall2023/zones/us-central1-b/instances/kv-server?authuser=2&hl=en_US&...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
nsrirang@kv-server:~$ python3 prgrm.py mount fuse_ecc "mount_folder/mnt/"
{"time":"07/11/2023 02:17:02.503210","severity":"INFO","msg":"Start gcsfuse/1.2.0 (Go version go1.21.0) for app \"
\" using mount point: /home/nsrirang/mount_folder/mnt\n"}
Bucket fuse_ecc mounted successfully.
nsrirang@kv-server:~$ ls
GCSFS.py  __pycache__  file_system  go  mount_folder  nikhil.json  prgrm.py
nsrirang@kv-server:~$ ls mount_folder
mnt
nsrirang@kv-server:~$ python3 prgrm.py create fuse_ecc "test.txt"
nsrirang@kv-server:~$
```

```
nsrirang@kv-server:~$ ls mount_folder/mnt/
test.txt
nsrirang@kv-server:~$
```



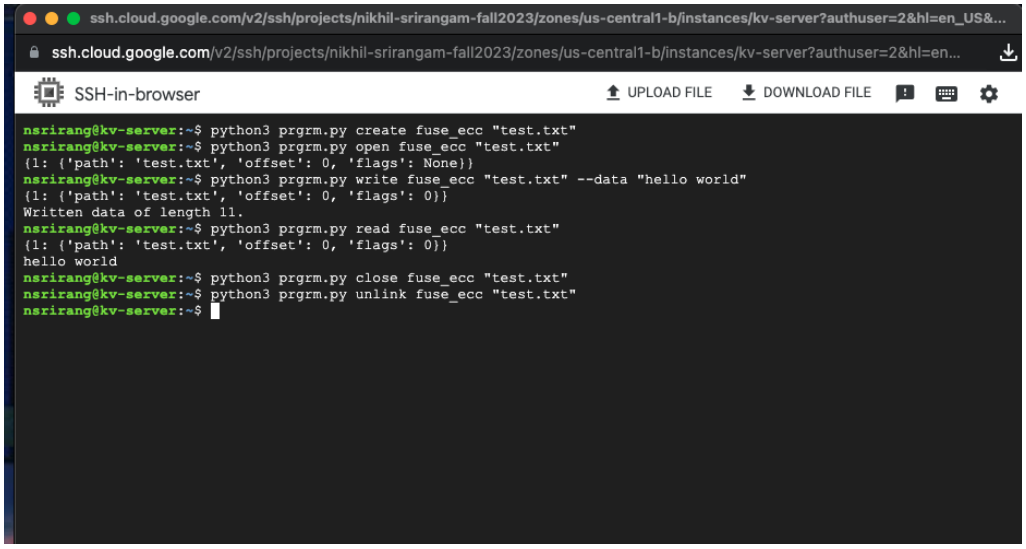
fuse_ecc

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	None

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiratio
test.txt	0 B	text/plain	Nov 6, 2023, 9:21:14 PM	Standard	Nov 6, 2023, 9:21:14 PM	Not public	—	Google-managed	—

As you can see the file is created in the gcs bucket as well as in the local file system. This marks the successful initialization of Fuse using cloud storage.

1) First test case is a simple implementation of create -> open-> write -> read -> close -> unlink. Unlinking the file is basically deleting the file from the storage.
Here is the working screenshot

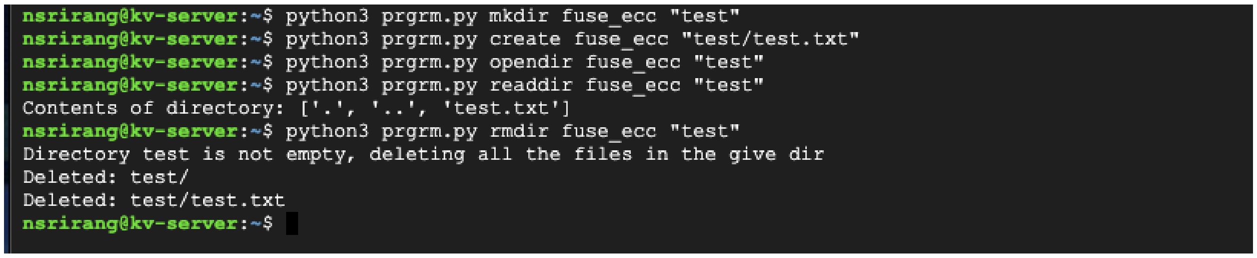

ssh.cloud.google.com/v2/ssh/projects/nikhil-srirangam-fall2023/zones/us-central1-b/instances/kv-server?authuser=2&hl=en_US&...
ssh.cloud.google.com/v2/ssh/projects/nikhil-srirangam-fall2023/zones/us-central1-b/instances/kv-server?authuser=2&hl=en...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
nsrirang@kv-server:~$ python3 prgrm.py create fuse_ecc "test.txt"
nsrirang@kv-server:~$ python3 prgrm.py open fuse_ecc "test.txt"
{1: {'path': 'test.txt', 'offset': 0, 'flags': None}}
nsrirang@kv-server:~$ python3 prgrm.py write fuse_ecc "test.txt" --data "hello world"
{1: {'path': 'test.txt', 'offset': 0, 'flags': 0}}
Written data of length 11.
nsrirang@kv-server:~$ python3 prgrm.py read fuse_ecc "test.txt"
{1: {'path': 'test.txt', 'offset': 0, 'flags': 0}}
hello world
nsrirang@kv-server:~$ python3 prgrm.py close fuse_ecc "test.txt"
nsrirang@kv-server:~$ python3 prgrm.py unlink fuse_ecc "test.txt"
nsrirang@kv-server:~$
```

Commands used to run it

```
python3 prgrm.py create fuse_ecc "test.txt"
python3 prgrm.py open fuse_ecc "test.txt"
python3 prgrm.py write fuse_ecc "test.txt" --data "hello world"
python3 prgrm.py read fuse_ecc "test.txt"
python3 prgrm.py close fuse_ecc "test.txt"
python3 prgrm.py unlink fuse_ecc "test.txt"
```

2. Second test case is using the directory commands for successful implementation
Mkdir -> opendir -> readdir -> rmdir
Also included a create in the middle, to see the contents or else it just prints [‘.’,’..’]

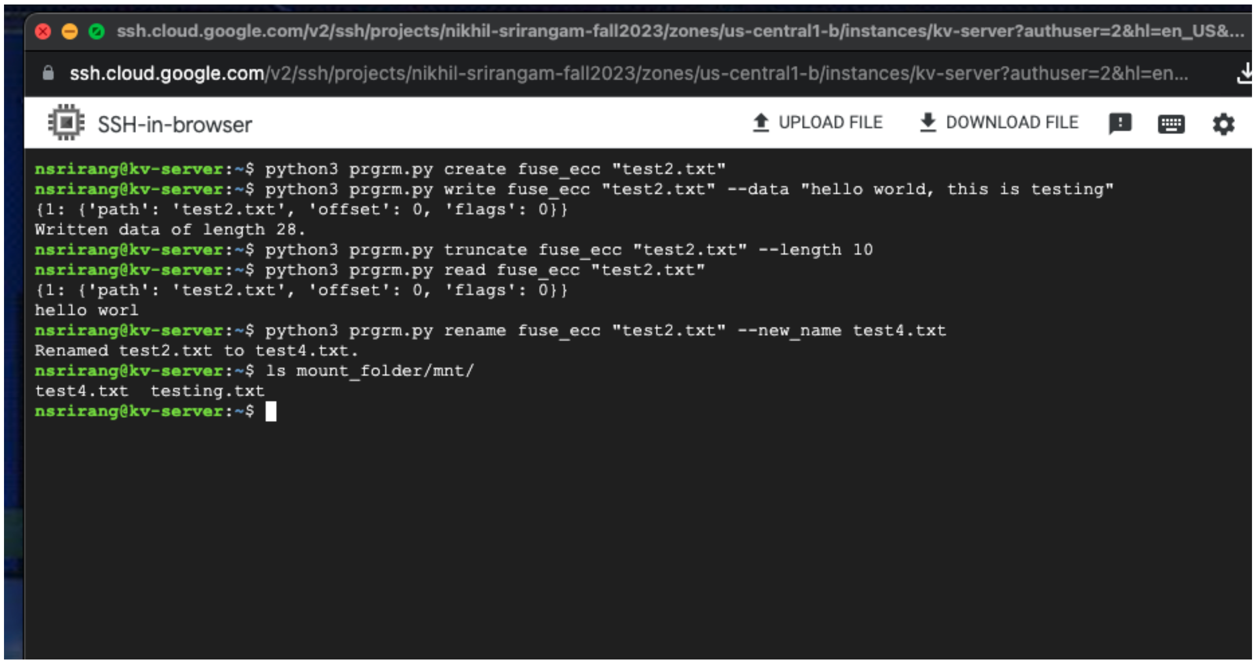


```
nsrirang@kv-server:~$ python3 prgrm.py mkdir fuse_ecc "test"
nsrirang@kv-server:~$ python3 prgrm.py create fuse_ecc "test/test.txt"
nsrirang@kv-server:~$ python3 prgrm.py opendir fuse_ecc "test"
nsrirang@kv-server:~$ python3 prgrm.py readdir fuse_ecc "test"
Contents of directory: ['.', '..', 'test.txt']
nsrirang@kv-server:~$ python3 prgrm.py rmdir fuse_ecc "test"
Directory test is not empty, deleting all the files in the give dir
Deleted: test/
Deleted: test/test.txt
nsrirang@kv-server:~$
```

Commands used to run it

```
Python3 prgrm.py mkdir fuse_ecc “test”
Python3 prgrm.py create fuse_ecc “test/test.txt”
Python3 prgrm.py opendir fuse_ecc “test”
Python3 prgrm.py readdir fuse_ecc “test”
Python3 prgrm.py rmdir fuse_ecc “test”
```

3. Third test case is to showcase the additional commands ‘rename’ and ‘truncate’
Here first we created a file with name “test2.txt” and named it as test2.txt, with 28 bits of test , then we truncated it to the first 10 bits and renamed to test2.txt


ssh.cloud.google.com/v2/ssh/projects/nikhil-srirangam-fall2023/zones/us-central1-b/instances/kv-server?authuser=2&hl=en_US&...
ssh.cloud.google.com/v2/ssh/projects/nikhil-srirangam-fall2023/zones/us-central1-b/instances/kv-server?authuser=2&hl=en...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
nsrirang@kv-server:~$ python3 prgrm.py create fuse_ecc "test2.txt"
nsrirang@kv-server:~$ python3 prgrm.py write fuse_ecc "test2.txt" --data "hello world, this is testing"
{1: {'path': 'test2.txt', 'offset': 0, 'flags': 0}}
Written data of length 28.
nsrirang@kv-server:~$ python3 prgrm.py truncate fuse_ecc "test2.txt" --length 10
nsrirang@kv-server:~$ python3 prgrm.py read fuse_ecc "test2.txt"
{1: {'path': 'test2.txt', 'offset': 0, 'flags': 0}}
hello worl
nsrirang@kv-server:~$ python3 prgrm.py rename fuse_ecc "test2.txt" --new_name test4.txt
Renamed test2.txt to test4.txt.
nsrirang@kv-server:~$ ls mount_folder/mnt/
test4.txt  testing.txt
nsrirang@kv-server:~$
```

Commands used to implement this are

```
python3 prgrm.py create fuse_ecc "test2.txt"
python3 prgrm.py write fuse_ecc "test2.txt" --data "hello world, this is testing"
python3 prgrm.py truncate fuse_ecc "test2.txt" --length 10
python3 prgrm.py read fuse_ecc "test2.txt"
python3 prgrm.py rename fuse_ecc "test2.txt" --new_name test4.txt
```

References:
https://thepythoncorner.com/posts/2017-02-27-writing-a-fuse-filesystem-in-python/#google_vignette
https://cloud.google.com/storage/docs/gcsfuse-quickstart-mount-bucket