Program 3 Architecture and Design

By: Mohammed Mohiuddin

# Program Outline

The program consists of two main files, backup.py and restore.py each of them utilize several methods to successfully backup and restore files to and from the cloud.

**backup.py**

* Responsible for taking files and directories from a given local directory and backing it to a bucket in the Amazon S3 cloud.
* Methods used:
  + **backup():** this is the main function used to backup local directories, it is called and passed the command line args and calls helper functions as needed
  + **does\_bucket\_exist()**: this is the helper function which is used to check if the bucket given exists or not. If this method returns false, it gives us the flag to create a new bucket
  + **does\_file\_existS3():** this function is another helper to check whether the file given to us already exists in the bucket or not. If it does exist, we need to execute the code that checks timestamps for modification. If this method returns false, we know that this file is a new one
  + **upload():** this function is used to simply upload the file to the given location in the bucket.
  + Lastly at the very bottom of the program, I have some code that acts as the main using sys and argv to capture the arguments and called the methods accordingly.

**restore.py**

* Responsible for taking directories/files from the bucket and downloading them to a local directory.
* Methods used:
  + **restore():** this is the main function used to restore the files to local directories, It uses all other helper functions accordingly.
  + **does\_bucket\_exist()**: this is the helper function which is used to check if the bucket given exists or not. If this method returns false, we simply issue an error since the given destination has no files to restore
  + **download():** this method is used to simply download files from the bucket to the local directory. If a directory doesn’t exist, it will also create one.
  + Similar to backup.py I also utilize some code at the bottom of the program to act as a main and retrieve command line inputs and pass parameters accordingly

# How To Run Instructions

The instructions to run the programs are fairly simple. I have ran this program on Windows locally and it works fine.

**backup.py:**

python3 backup.py backup <loacl\_directory\_name> <bucket\_name>::<directory\_name>

For example: python3 backup.py backup dirA css436bucket::dirA

**restore.py:**

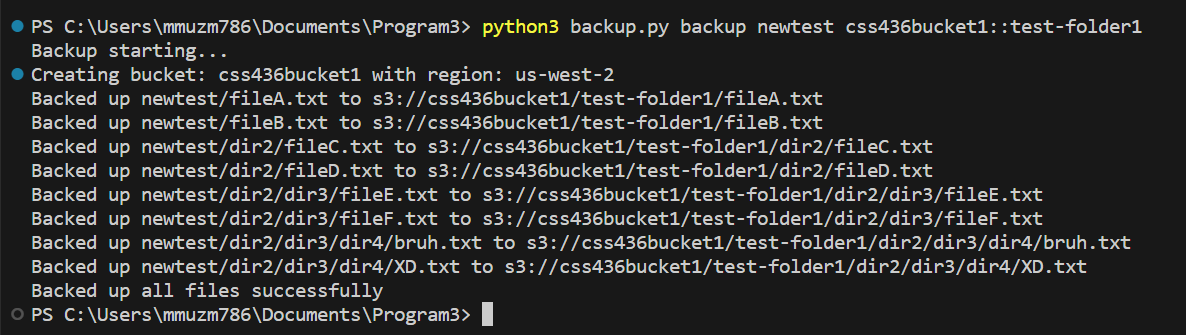
python3 restore.py restore <bucket\_name>::<directory\_name> <local\_directory\_name>

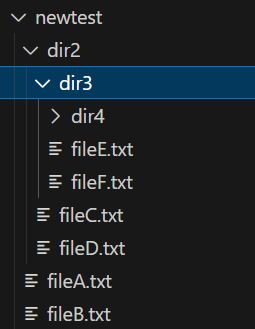
For example: python3 restore css436bucket::dirA test

# Runtime & Test Cases

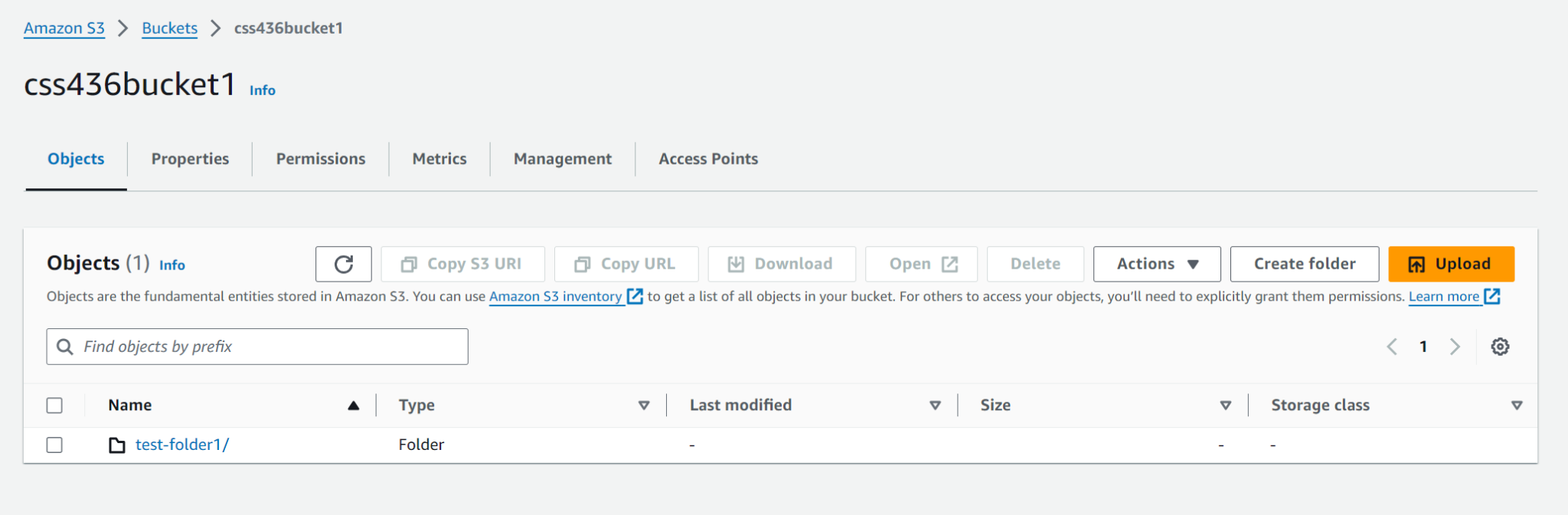
**Backup:**

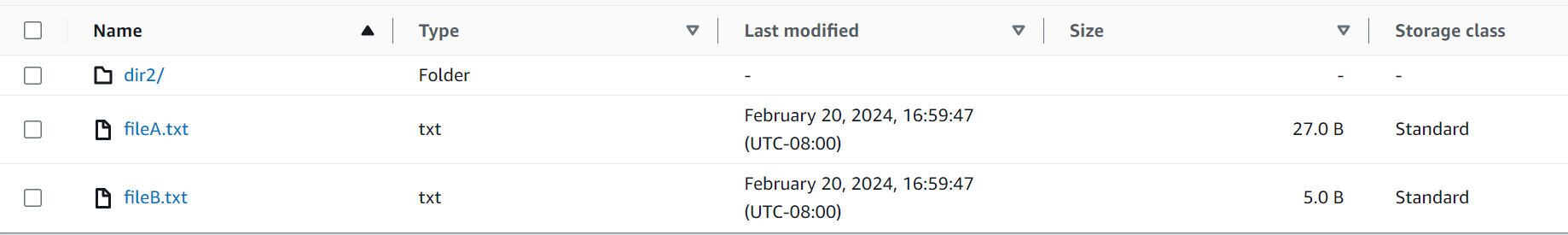
Test Case 1: Initial backup of files and creating a bucket since it doesn’t exist

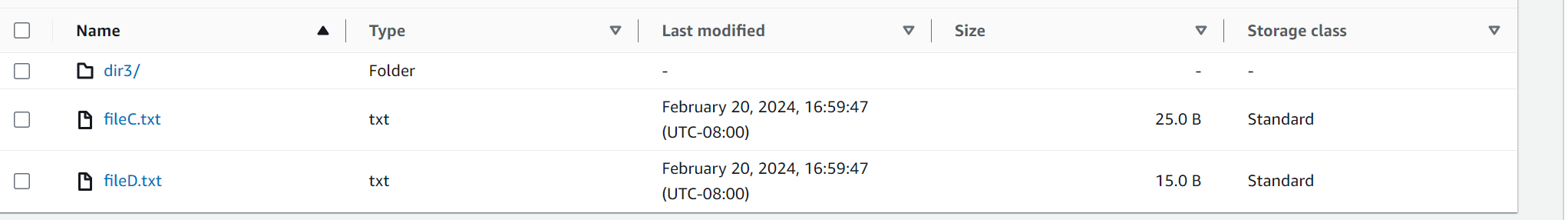


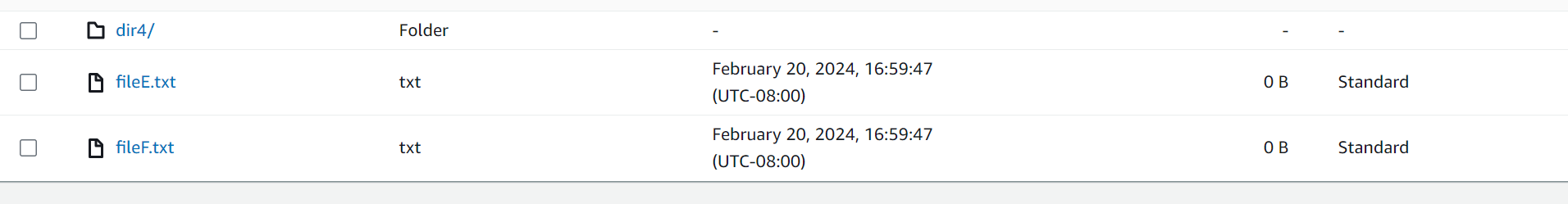


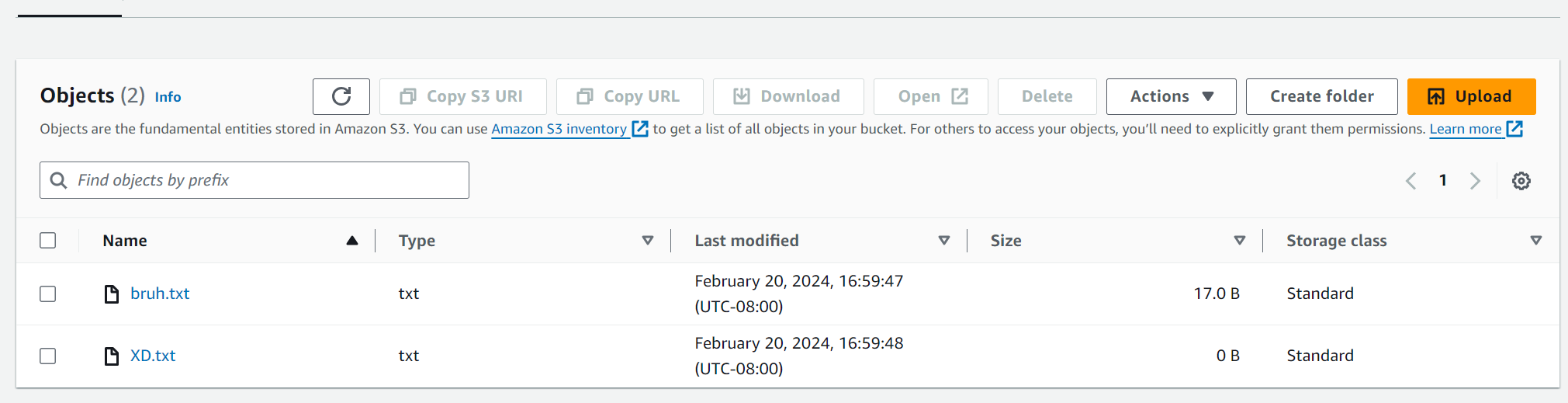
Here are the contents in the cloud



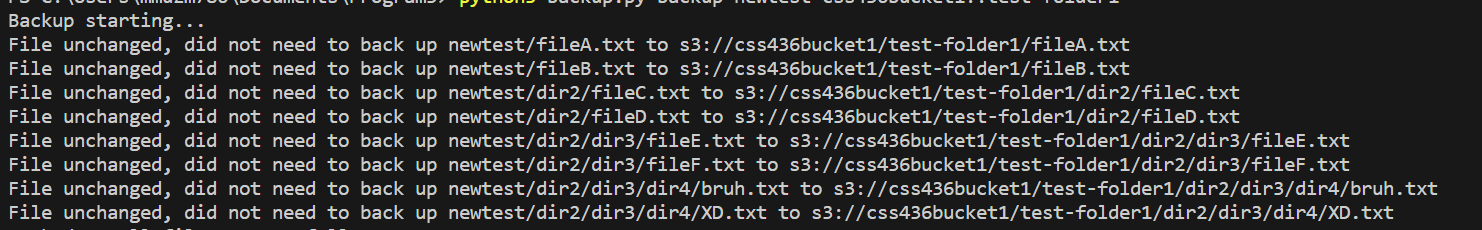




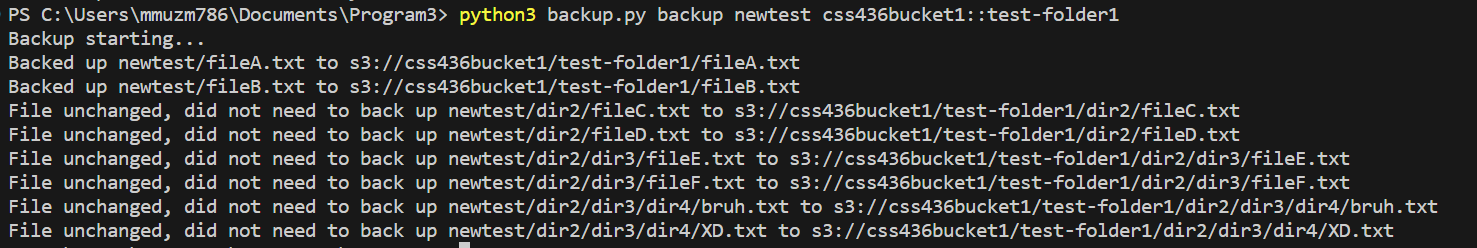




Test Case 2: Try performing a backup again



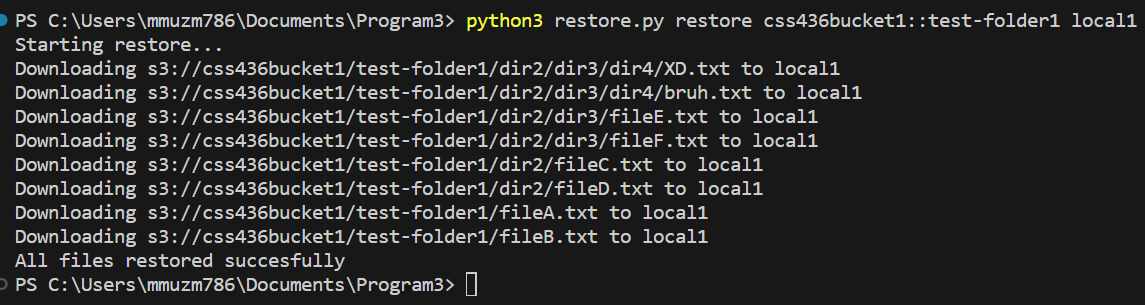
Now let’s try making one change to fileA and fileB and see what happens

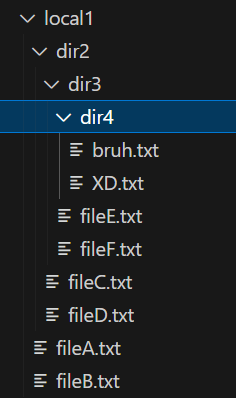


As you can see the files that were modified got backed up

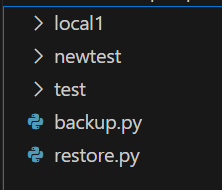
**Restore:**

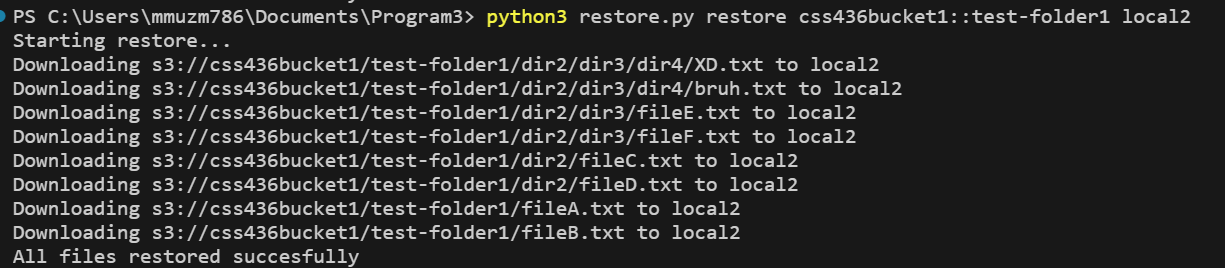
Now let’s try to restore test-folder1 into a local directory. We shall create a directory “local1” and it will be empty

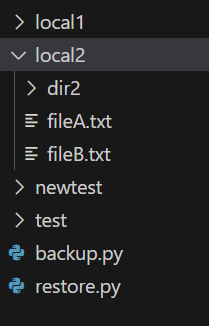


 These are the contents of local1 after running.

The program will also create a directory if it doesn’t exist. Let’s say we have these initial directories

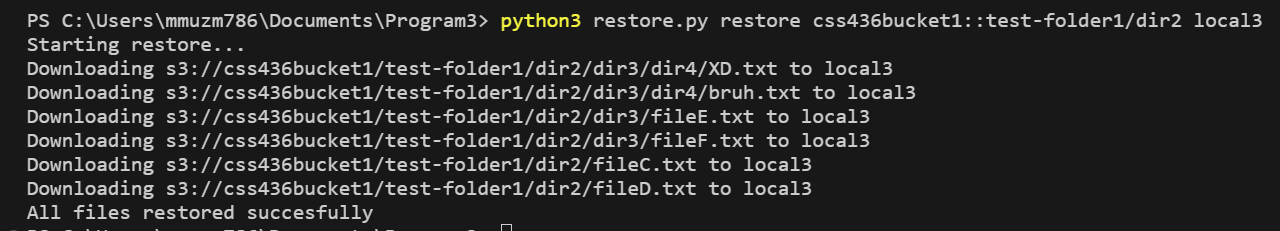


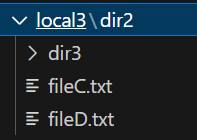




This is after the execution and as you can see it creates a new directory with all the files

Finally let’s have a case where a subdirectory is passed in. For example, maybe something like test-folder1/dir2 (If you are specifying a directory within a directory please use forward slashes instead of backslashes)





Local3 will have all of the files that are only within dir2, so fileA and fileB won’t be there

# Assumptions/Design Choices

My program does a few changes to the way files are formatted and displayed such as removing slashes from the front of a file or adding slashes to the end of a file for consistent naming conventions. I also utilize forward slashes and convert all \\ to /

Another design choice I have is I don’t require there to be a bucket-directory for the backup to happen. For example, a command like “python3 backup.py backup local3 css436bucket1::” without the bucket-directory will still be backed up to the cloud. Other than that, my program gracefully handles all other errors such as missing parameters, invalid names, etc.