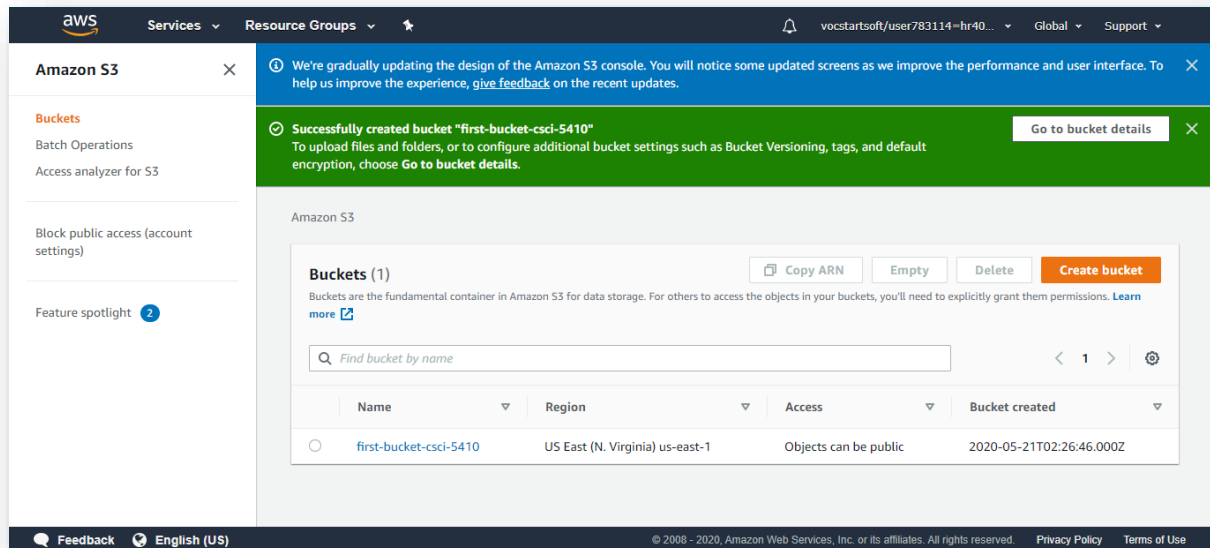


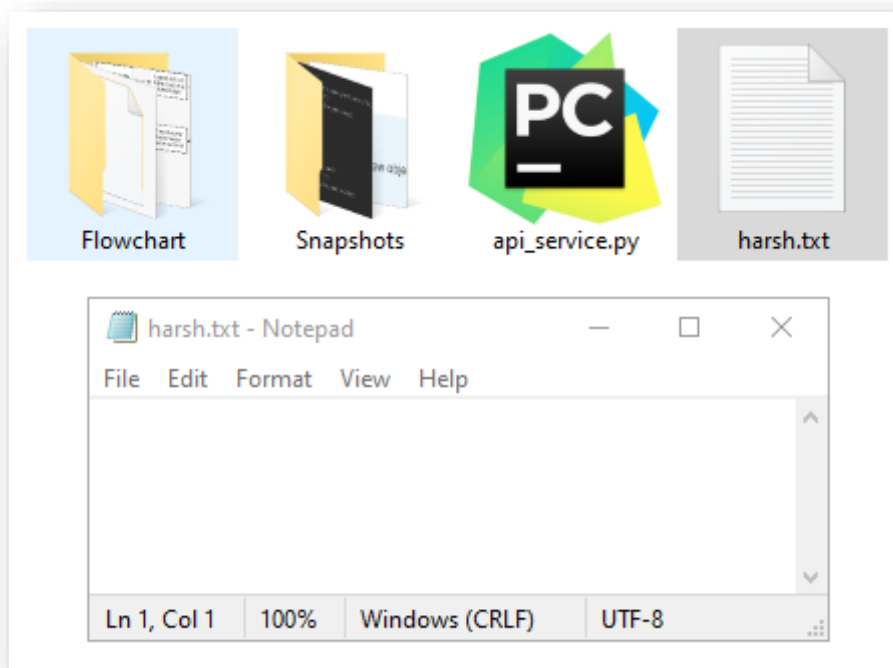
Part B: AWS S3 storage experiment

a.

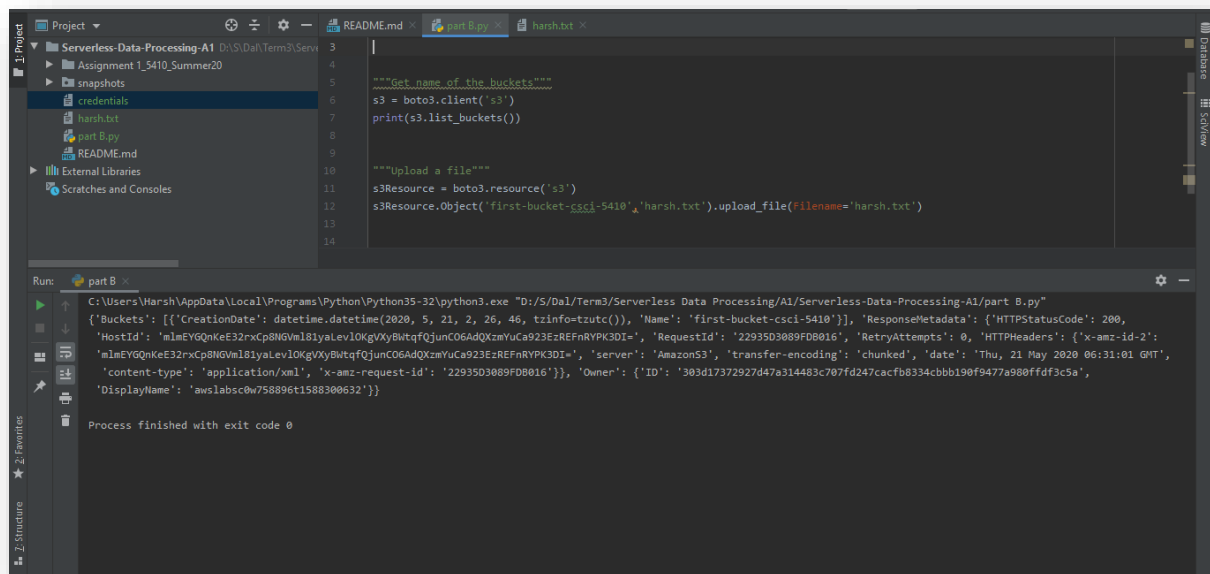
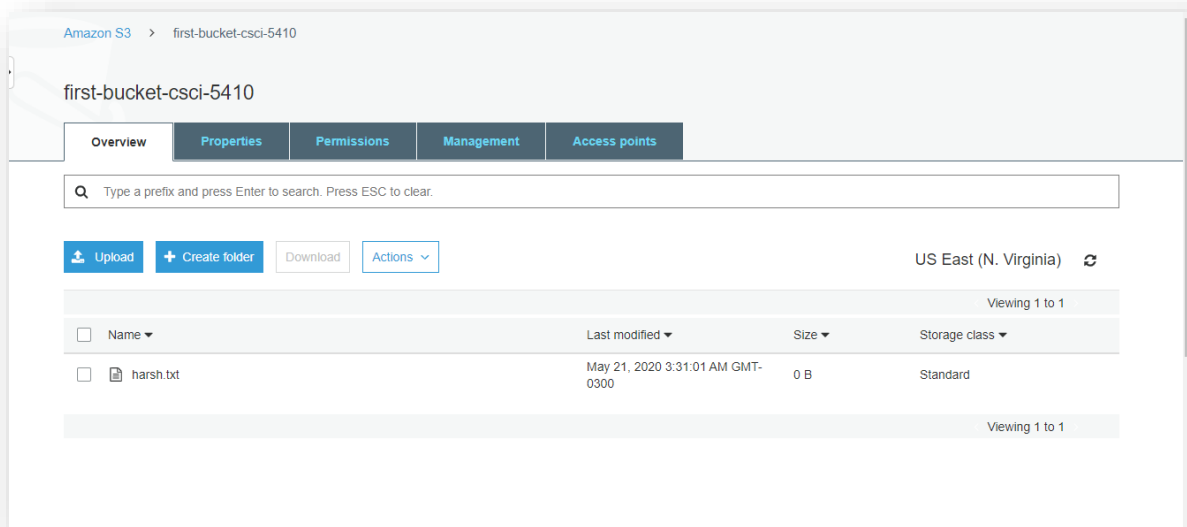
1. Create an empty bucket named “first-bucket-csci-5410”



2. Create an empty text file ‘harsh.txt’

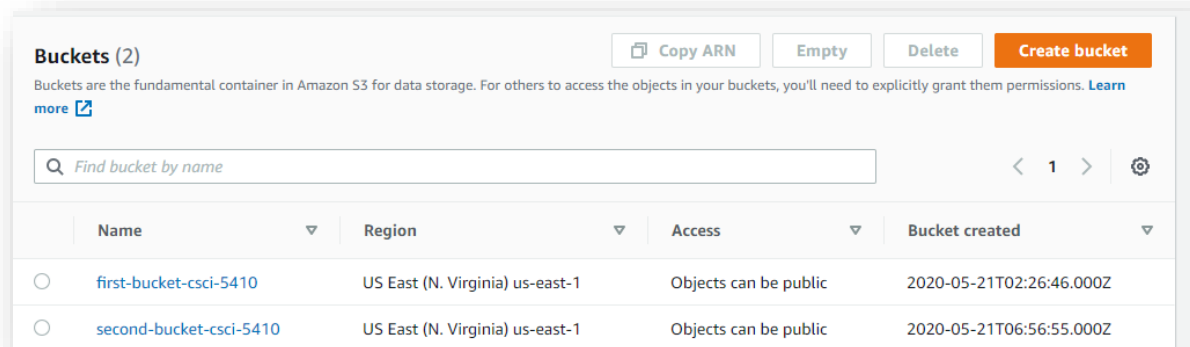


b. Upload the file on the bucket “first-bucket-csci-5410”



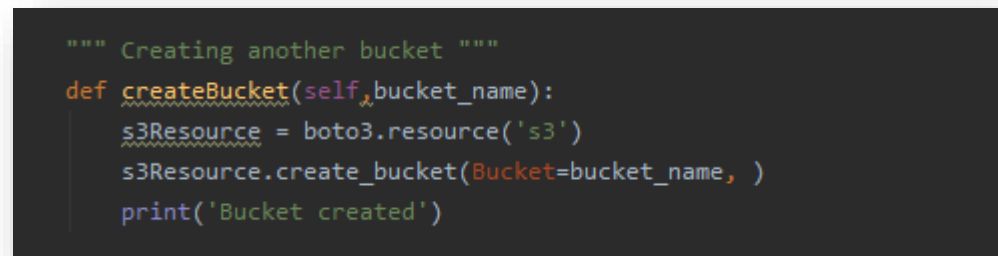
C.

1. Create a second bucket named “second-bucket-csci-5410” using python.



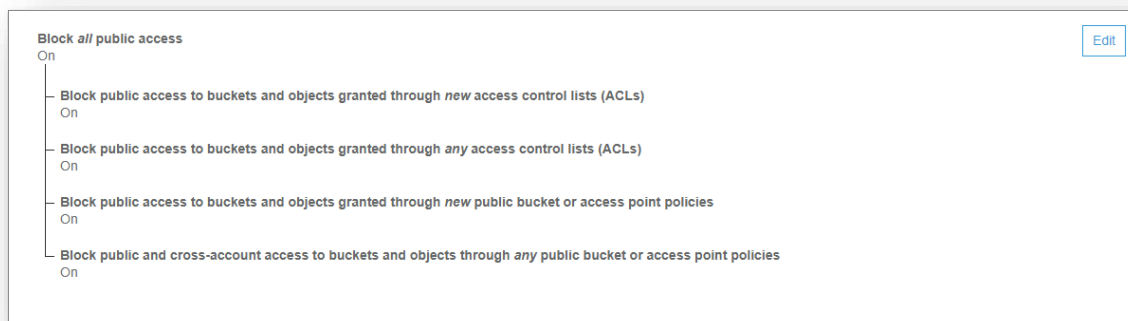
The screenshot shows the Amazon S3 Buckets console. At the top, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. Below these is a search bar labeled 'Find bucket by name'. A table lists the buckets with columns for Name, Region, Access, and Bucket created. Two buckets are listed: 'first-bucket-csci-5410' and 'second-bucket-csci-5410', both in 'US East (N. Virginia) us-east-1' region with 'Objects can be public' access.

	Name	Region	Access	Bucket created
<input type="radio"/>	first-bucket-csci-5410	US East (N. Virginia) us-east-1	Objects can be public	2020-05-21T02:26:46.000Z
<input type="radio"/>	second-bucket-csci-5410	US East (N. Virginia) us-east-1	Objects can be public	2020-05-21T06:56:55.000Z



```
""" Creating another bucket """
def createBucket(self, bucket_name):
    s3Resource = boto3.resource('s3')
    s3Resource.create_bucket(Bucket=bucket_name, )
    print('Bucket created')
```

2. Change the access permission to “disable public access”



The screenshot shows the 'Block all public access' settings in the Amazon S3 console. The 'On' toggle is checked. Below it, four options are listed, each with an 'On' toggle:

- Block public access to buckets and objects granted through new access control lists (ACLs) On
- Block public access to buckets and objects granted through any access control lists (ACLs) On
- Block public access to buckets and objects granted through new public bucket or access point policies On
- Block public and cross-account access to buckets and objects through any public bucket or access point policies On

```

"""No Public Access"""
s3Client = boto3.client('s3',region_name='us-east-1')
s3Client.put_public_access_block(
    Bucket="second-bucket-csci-5410",
    PublicAccessBlockConfiguration={
        'BlockPublicAcls': True,
        'IgnorePublicAcls': True,
        'BlockPublicPolicy': True,
        'RestrictPublicBuckets': True
    },
)

```

3. Change the ACL write option to “no” for bucket owner.

Block public access
Access Control List
Bucket Policy
CORS configuration

Access for bucket owner

Canonical ID	List objects	Write objects	Read bucket permissions	Write bucket permissions
<input type="radio"/> 303d17372927d47a314483c707fd247cacfb8334cbbb190f9477a980ffdf3c5a (Your AWS account)	-	-	Yes	-

Access for other AWS accounts

+ Add account
Delete

Canonical ID	List objects	Write objects	Read bucket permissions	Write bucket permissions
--------------	--------------	---------------	-------------------------	--------------------------

```

30
31
32 """get ACL permissions"""
33 s3Client = boto3.client('s3')
34 json_file=s3Client.get_bucket_acl(bucket='second-bucket-csci-5410')
35 json_file.pop('ResponseMetadata')
36 json_file['Grants'][0]['Permission'] = 'READ'
37 s3Client.put_bucket_acl(Bucket='second-bucket-csci-5410', AccessControlPolicy=json_file)
38
39 print(s3Client.get_bucket_acl(bucket='second-bucket-csci-5410'))
40
41
42
43
44
45

```

Run: C:\Users\Harsh\AppData\Local\Programs\Python\Python35-32\python.exe "D:/S/Dal/Term3/Serverless Data Processing/A1/Serverless-Data-Processing-A1/part B.py"

```

{'Owner': {'DisplayName': 'awslabsc0w758896t1588300632', 'ID': '303d17372927d47a314483c707fd247cacfb8334cbbb190f9477a980ffdf3c5a'}, 'ResponseMetadata': {'HostId': 'oA0ua47YxpH9z96OH9brh8UAAnkpp8eU48FZcau/Sj17JgP3+TC2hQ29nQ92fwDQ1F99Hg4RxxI=', 'RequestId': '29F286A5CDC74A50', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'oA0ua47YxpH9z96OH9brh8UAAnkpp8eU48FZcau/Sj17JgP3+TC2hQ29nQ92fwDQ1F99Hg4RxxI=', 'x-amz-request-id': '29F286A5CDC74A50', 'date': 'Fri, 22 May 2020 01:29:13 GMT', 'content-type': 'application/xml', 'server': 'AmazonS3', 'transfer-encoding': 'chunked', 'RetryAttempts': 0}, 'Grants': [{'Grantee': {'Type': 'CanonicalUser', 'DisplayName': 'awslabsc0w758896t1588300632', 'ID': '303d17372927d47a314483c707fd247cacfb8334cbbb190f9477a980ffdf3c5a'}, 'Permission': 'READ'}]

```

Process finished with exit code 0

d. Move the file from “first-bucket-csci-5410” to “second-bucket-csci-5410”.

second-bucket-csci-5410

Overview Properties Permissions Management Access points

🔍 Type a prefix and press Enter to search. Press ESC to clear.

📁 Upload + Create folder Download Actions ▾

US East (N. Virginia) 🔁

Viewing 1 to 1

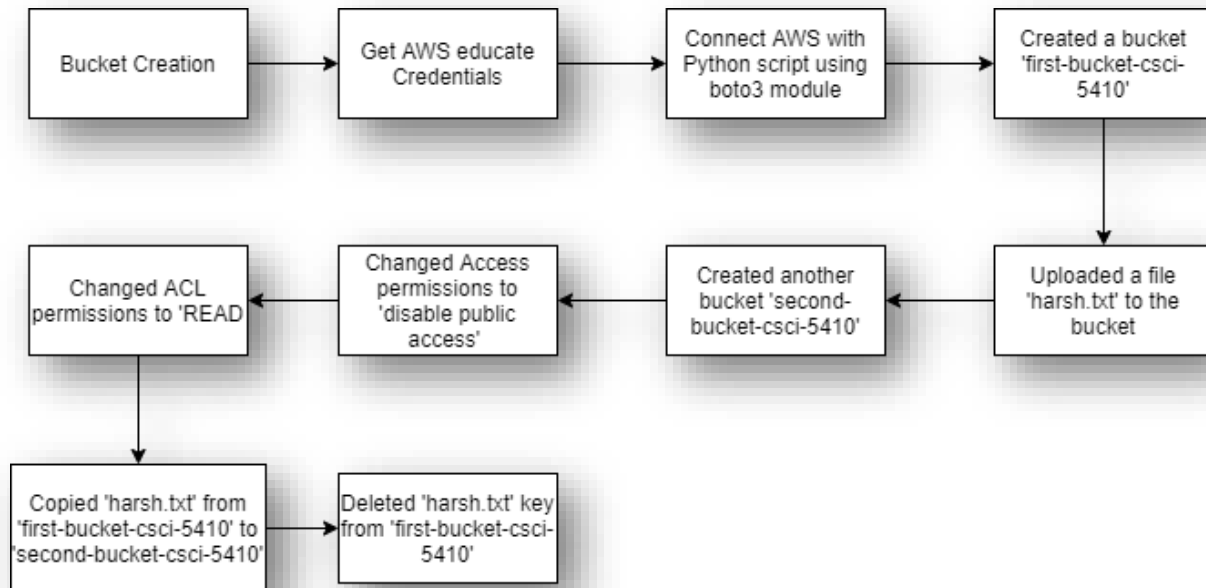
<input type="checkbox"/>	Name ▾	Last modified ▾	Size ▾	Storage class ▾
<input type="checkbox"/>	📄 harsh.txt	May 22, 2020 11:14:07 PM GMT-0300	0 B	Standard

Viewing 1 to 1

```
""" Copy keys between buckets """
def CopyKeysBetweenBuckets(self, source_bucket, dest_bucket, file_name):
    s3_resource = boto3.resource('s3')
    s3_resource.Object(dest_bucket, file_name).copy({
        'Bucket': source_bucket,
        'Key': file_name
    })
    print('key Copied')

""" Delete keys from a bucket """
def deleteKeys(self, bucketName, file_name):
    s3_resource = boto3.resource('s3')
    s3_resource.Object(bucketName, file_name).delete()
    print('Key deleted')
```

e. A flowchart showing the steps performed in the experiment



In the experiment, first and foremost I created a bucket 'first-bucket-csci-5410'. Then I started creating a user in IAM, but later I found that we cannot create IAM users in AWS educate account. Then I integrated my python script with AWS S3 using the credentials put in 'Users/<username>/.aws/credentials' file. After checking that my set up is complete, I uploaded 'harsh.txt' key to the bucket. Then I created another bucket named 'second-bucket-csci-5410' and changed its access permission using the 'put_public_access_block()' method. I also changed its ACL permissions to 'READ' using 'put_bucket_acl()' method. Then I copied the key from former bucket to later created bucket and deleted the key from former bucket which ultimately moved the key.

Source Code

```
import boto3

class s3Api:
    """ Get name of the buckets """

    def listBuckets(self):
        s3 = boto3.client('s3')
        return s3.list_buckets()

    """ Upload a file """

    def fileUpload(self, bucket_name, source_file_name, file_name):
        s3Resource = boto3.resource('s3')
        s3Resource.Object(bucket_name, source_file_name).upload_file(Filename=file_name)
        print('file uploaded')

    """ Creating another bucket """

    def createBucket(self, bucket_name):
        s3Resource = boto3.resource('s3')
        s3Resource.create_bucket(Bucket=bucket_name, )
        print('Bucket created')

    """ Change bucket access (private, public etc) """

    def manageBucketAccess(self, bucket_name, block_public_acls=True, ignore_public_acls=True,
block_public_policy=True, restrict_public_buckets=True, region='us-east-1'):
        s3_client = boto3.client('s3', region_name=region)
        s3_client.put_public_access_block(
            Bucket=bucket_name,
            PublicAccessBlockConfiguration={
                'BlockPublicAcls': block_public_acls,
                'IgnorePublicAcls': ignore_public_acls,
                'BlockPublicPolicy': block_public_policy,
                'RestrictPublicBuckets': restrict_public_buckets
            },
        )
        print('Access Changed')

    """ Get ACL permissions """

    def getAclPermissions(self, bucket_name):
        s3_client = boto3.client('s3')
        return s3_client.get_bucket_acl(Bucket=bucket_name)
```

```
""" Set ACL permissions """
```

```
def changeAclPermissions(self, bucket_name, permission):  
    s3_client = boto3.client('s3')  
    json_file = s3_client.get_bucket_acl(Bucket=bucket_name)  
    json_file.pop('ResponseMetadata')  
    json_file['Grants'][0]['Permission'] = permission  
    s3_client.put_bucket_acl(Bucket=bucket_name, AccessControlPolicy=json_file)  
    print('ACL permission changed')
```

```
""" Copy keys between buckets """
```

```
def copyKeysBetweenBuckets(self, source_bucket, dest_bucket, file_name):  
    s3_resource = boto3.resource('s3')  
    s3_resource.Object(dest_bucket, file_name).copy({  
        'Bucket': source_bucket,  
        'Key': file_name  
    })  
    print('key Copied')
```

```
""" Delete keys from a bucket """
```

```
def deleteKeys(self, bucketName, file_name):  
    s3_resource = boto3.resource('s3')  
    s3_resource.Object(bucketName, file_name).delete()  
    print('Key deleted')
```

```
""" Downloading a file from the bucket """
```

```
def downloadFile(self, bucket_name, obj_name, dest_file_name):  
    s3Client = boto3.client('s3')  
    s3Client.download_file(bucket_name, obj_name, dest_file_name)  
    print('file downloaded')
```

```
s3 = s3Api()  
s3.listBuckets()  
s3.fileUpload("first-bucket-csci-5410", "harsh.txt", "harsh.txt")  
s3.createBucket("second-bucket-csci-5410")  
s3.manageBucketAccess("second-bucket-csci-5410")  
s3.getAclPermissions("second-bucket-csci-5410")  
s3.changeAclPermissions("second-bucket-csci-5410", "READ")  
s3.copyKeysBetweenBuckets("first-bucket-csci-5410", "second-bucket-csci-5410", "harsh.txt")  
s3.deleteKeys("first-bucket-csci-5410", "harsh.txt")  
s3.downloadFile("first-bucket-csci-5410", "Lookup5410.txt", "Lookup.txt")
```


References

- [1]"Python S3 Examples — Ceph Documentation", *Docs.ceph.com*, 2020. [Online]. Available: <https://docs.ceph.com/docs/master/radosgw/s3/python/>. [Accessed: 25- May- 2020].
- [2]"Amazon S3 examples — Boto3 Docs 1.13.16 documentation", *Boto3.amazonaws.com*, 2020. [Online]. Available: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html>. [Accessed: 25- May- 2020].
- [3]R. Python, "Python, Boto3, and AWS S3: Demystified – Real Python", *Realpython.com*, 2020. [Online]. Available: <https://realpython.com/python-boto3-aws-s3/>. [Accessed: 25- May- 2020].