Lab 3: Configure S3 Buckets

1. Preparation: a. Downloading cloudstorage.py

```
jookai@jookai:~/Desktop/cits5503/lab3$ curl https://raw.githubusercontent.com/zhangzhics/CITS5503_Sem2_2023
/master/Labs/src/cloudstorage.py
```

b. Creating rootdir

```
jookai@jookai:~/Desktop/cits5503/lab3$ mkdir rootdir
jookai@jookai:~/Desktop/cits5503/lab3$ echo ""1\n2\n3\n4\n5\n"" > rootfile.txt
```

c. Creating subdir

```
jookai@jookai:~/Desktop/cits5503/lab3/rootdir$ mkdir subdir
jookai@jookai:~/Desktop/cits5503/lab3/rootdir$ cd subdir/
jookai@jookai:~/Desktop/cits5503/lab3/rootdir/subdir$ echo ""1\n2\n3\n4\n5\n"" > subfile.txt
```

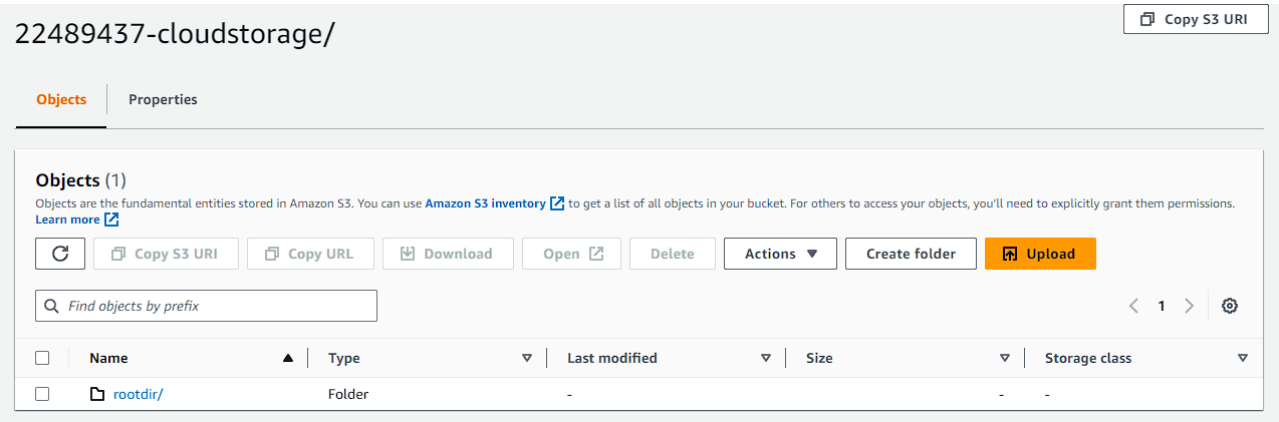
2. Editing cloudstorage.py to take the initialise argument. The following code segment uses the argparse library to allow the argparse library to take 2 arguments `-i` or `-initialise`. Using either of these 2 arguments will set the `bucketFlag` variable to True. If `bucketFlag` is True, we will attempt to create a bucket in S3. We also catch any exceptions such as `BucketAlreadyExists` and `BucketAlreadyOwnedByYou`.

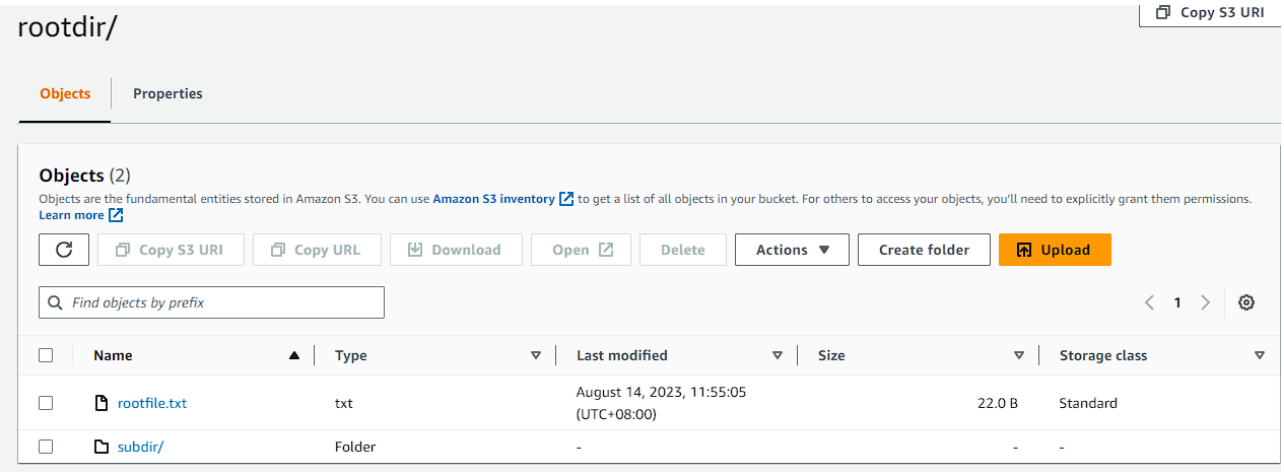
```
        argParser = argparse.ArgumentParser()
    argParser.add_argument("-i", "--initialise", action='store_true',
help="create bucket on s3")
    args = argParser.parse_args()
    bucketFlag = args.initialise
    # Insert code to create bucket if not there
    if(bucketFlag):
        try:
            response = s3.create_bucket(Bucket=ROOT_S3_DIR,
CreateBucketConfiguration=bucket_config)
            print("Bucket created successfully:", ROOT_S3_DIR)
            print(response)
        except s3.exceptions.BucketAlreadyExists:
            print("Bucket already exists:", ROOT_S3_DIR)
        except s3.exceptions.BucketAlreadyOwnedByYou:
            print("Bucket already owned by you:", ROOT_S3_DIR)
        except Exception as error:
            print("An error occurred:", error)
```

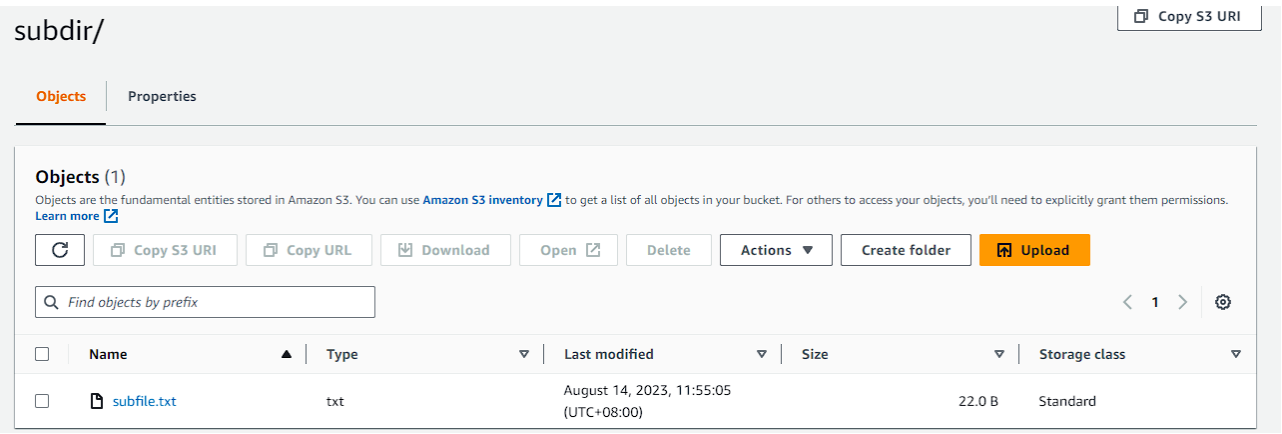
3. The `upload_file()` method found in `cloudstoraage.py` was modified to add the code required to upload files to S3. The path of the file in S3 is constructed by adding the `ROOT_S3_DIR` which was created in step 2 to the folder path and file name. The file is then uploaded to S3 using the `s3.upload_file()` command.

```
    def upload_file(folder_name, file, file_name):
    # Construct the path for files to be uploaded
    s3Path = f'{ROOT_S3_DIR}/{folder_name}/{file_name}'

    print("Uploading %s" % file)

    # Upload the file to S3
    try:
        s3.upload_file(file, ROOT_S3_DIR, s3Path)
        print(f"Uploaded {file_name} to S3")
    except Exception as e:
        print(f"Error uploading {file_name}: {e}")
```

```
jookai@jookai:~/Desktop/cits5503/lab3$ python3 cloudstorage.py -i
Bucket created successfully: 22489437-cloudstorage
{'ResponseMetadata': {'RequestId': '2H2FG2EYVQVDF0TD', 'HostId': 'ZTn69/PHD+ICRvx5AN0hPB5/jRtJrsZBKTRoCig8
8EEgthbhcB2MFYSVuWvii1YwUDf/MGk6kIE=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'ZTn69/PHD+ICR
vx5AN0hPB5/jRtJrsZBKTRoCig88EEgthbhcB2MFYSVuWvii1YwUDf/MGk6kIE=', 'x-amz-request-id': '2H2FG2EYVQVDF0TD',
'date': 'Mon, 14 Aug 2023 03:55:04 GMT', 'location': 'http://22489437-cloudstorage.s3.amazonaws.com/', 'se
rver': 'AmazonS3', 'content-length': '0'}, 'RetryAttempts': 0}, 'Location': 'http://22489437-cloudstorage.
s3.amazonaws.com/'}
Uploading ./rootdir/rootfile.txt
Uploaded rootfile.txt to S3
Uploading ./rootdir/subdir/subfile.txt
Uploaded subfile.txt to S3
done
```







4. The following code which is saved in restorefromcloud.py reads the content of the s3 bucket using the command `s3.list_objects` and writes them to the local directory. If the folder structure does not exist in the local directory, the code uses `os.makedirs` to recreate the folder structure from s3 in the local directory.

```
import os
import boto3
ROOT_DIR = '.'
ROOT_S3_DIR = '22489437-cloudstorage'
REGION = 'ap-southeast-2'

s3 = boto3.client("s3", region_name=REGION)
bucket_config = {'LocationConstraint': 'ap-southeast-2'}

for key in s3.list_objects(Bucket = ROOT_S3_DIR)['Contents']:
    if not os.path.exists(os.path.dirname(key['Key'])):
        os.makedirs(os.path.dirname(key['Key']))

    print("Downloading %s" % key['Key'])
    s3.download_file(ROOT_S3_DIR, key['Key'],  key['Key'])
```

```
jookai@jookai:~/Desktop/cits5503/lab3$ python3 restorefromcloud.py
Downloading 22489437-cloudstorage/rootdir/rootfile.txt
Downloading 22489437-cloudstorage/rootdir/subdir/subfile.txt
```

5. Setting up DynamoDB locally

```
jookai@jookai:~$ mkdir dynamodb
jookai@jookai:~$ cd dynamodb/
```

```
jookai@jookai:~/dynamodb$ sudo apt-get install default-jre
```

```
jookai@jookai:~/dynamodb$ wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local
_latest.tar.gz
--2023-08-14 13:20:26--  https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_lates
t.tar.gz
Resolving s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)... 52.219.68.104, 52.219.162.2
20, 52.219.8.76, ...
Connecting to s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)|52.219.68.104|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 44278951 (42M) [application/x-tar]
Saving to: ‘dynamodb_local_latest.tar.gz’

dynamodb_local_latest.tar. 100%[=====================================>]  42.23M  4.75MB/s    in 9.9s

2023-08-14 13:20:37 (4.27 MB/s) - ‘dynamodb_local_latest.tar.gz’ saved [44278951/44278951]
```

```
jookai@jookai:~/dynamodb$ java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
Initializing DynamoDB Local with the following configuration:
Port:   8000
InMemory:       false
DbPath: null
SharedDb:       false
shouldDelayTransientStatuses:   false
CorsParams:     null
```

6. Creating a table with the following attributes: `CloudFiles = { 'userId', 'fileName', 'path', 'lastUpdated', 'owner', 'permissions' } )`

```
jookai@jookai:~/Desktop/cits5503/lab3$ aws dynamodb create-table \
> --table-name CloudFiles \
> --attribute-definitions \
> AttributeName=userId,AttributeType=S \
> AttributeName=fileName,AttributeType=S \
> --key-schema \
> AttributeName=userId,KeyType=HASH \
> AttributeName=fileName,KeyType=RANGE \
> --provisioned-throughput \
> ReadCapacityUnits=5,WriteCapacityUnits=5 \
> --endpoint-url=http://localhost:8000
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "userId",
                "AttributeType": "S"
            },
            {
                "AttributeName": "fileName",
                "AttributeType": "S"
            }
        ],
        "TableName": "CloudFiles",
        "KeySchema": [
            {
                "AttributeName": "userId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "fileName",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1691995105.354,
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": 0.0,
            "LastDecreaseDateTime": 0.0,
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:ddblocal:000000000000:table/CloudFiles"
    }
}
```

7. The following code which is saved in storeinfo.py is used to get the file information from s3 and put it into the DynamoDB table created in step 6.

```
import boto3

ROOT_S3_DIR = '22489437-cloudstorage'
REGION = 'ap-southeast-2'
TABLE = 'CloudFiles'

# Connect to S3 and dynamodb
```

```
            dynamodb = boto3.resource('dynamodb', region_name=REGION,
endpoint_url='http://localhost:8000')
            table = dynamodb.Table(TABLE)
            s3 = boto3.client('s3', region_name=REGION)

            # List objects in the specified S3 bucket
            response = s3.list_objects(Bucket=ROOT_S3_DIR)

            # Extract relevant information from the S3 response
            userId = str(response['Contents'][0]['Owner']['ID'])
            owner = response['Contents'][0]['Owner']['DisplayName']
            permission = s3.get_bucket_acl(Bucket=ROOT_S3_DIR)['Grants'][0]
['Permission']


            # Iterate through each object in the S3 bucket
            for content in response['Contents']:
                # Get attributes requested by lab sheet
                item = {
                    'userId': userId,
                    'fileName': content['Key'].split('/')[-1],
                    'path': content['Key'],
                    'lastUpdated': str(content['LastModified']),
                    'owner': owner,
                    'permissions': permission
                }

                print('Putting the following item into DynamoDB table:\n', item,
'\n')

                # Add the item to the DynamoDB table
                table.put_item(Item=item)
```

8. The code is run which stores the details of the two files into the `CloudFiles` table:

```
jookai@jookai:~/Desktop/cits5503/lab3$ python3 storefileinfo.py
Putting the following item into CloudFiles table:
 {'userId': '2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e', 'fileName': 'rootfile.txt'
, 'path': '22489437-cloudstorage/rootdir/rootfile.txt', 'lastUpdated': '2023-08-14 03:55:05+00:00', 'owner
': 'zhi.zhang', 'permissions': 'FULL_CONTROL'}

Putting the following item into CloudFiles table:
 {'userId': '2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e', 'fileName': 'subfile.txt',
 'path': '22489437-cloudstorage/rootdir/subdir/subfile.txt', 'lastUpdated': '2023-08-14 03:55:05+00:00', '
owner': 'zhi.zhang', 'permissions': 'FULL_CONTROL'}
```

9. The `CloudFiles` table is scanned using the command `aws dynamodb scan --table-name CloudFiles --endpoint-url http://localhost:8000`. This reveals the details of the two files which were added in step 8.

```
jookai@jookai:~/Desktop/cits5503/lab3$ aws dynamodb scan --table-name CloudFiles --endpoint-url http://loc
alhost:8000
{
    "Items": [
        {
            "owner": {
                "S": "zhi.zhang"
            },
            "path": {
                "S": "22489437-cloudstorage/rootdir/rootfile.txt"
            },
            "lastUpdated": {
                "S": "2023-08-14 03:55:05+00:00"
            },
            "fileName": {
                "S": "rootfile.txt"
            },
            "userId": {
                "S": "2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e"
            },
            "permissions": {
                "S": "FULL_CONTROL"
            }
        },
        {
            "owner": {
                "S": "zhi.zhang"
            },
            "path": {
                "S": "22489437-cloudstorage/rootdir/subdir/subfile.txt"
            },
            "lastUpdated": {
                "S": "2023-08-14 03:55:05+00:00"
            },
            "fileName": {
                "S": "subfile.txt"
            },
            "userId": {
                "S": "2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e"
            },
            "permissions": {
                "S": "FULL_CONTROL"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```