Practical Worksheet 3

Program

Preparation

Create a directory rootdir

Create a file in rootdir called rootfile.txt and put some content in it "1\n2\n3\n4\n5\n"

Create a second directory in rootdir called subdir and create another file subfile.txt with the same content as rootfile.txt

```
(venv) >_ Labs/lab3 $ mkdir rootdir
(venv) >_ Labs/lab3 $ echo "1\n2\n3\n4\n5\n" > rootdir/rootfile.txt
(venv) >_ Labs/lab3 $ mkdir rootdir/subdir
(venv) >_ Labs/lab3 $ cp rootdir/rootfile.txt rootdir/subdir/subfile.txt
```

Save to S3

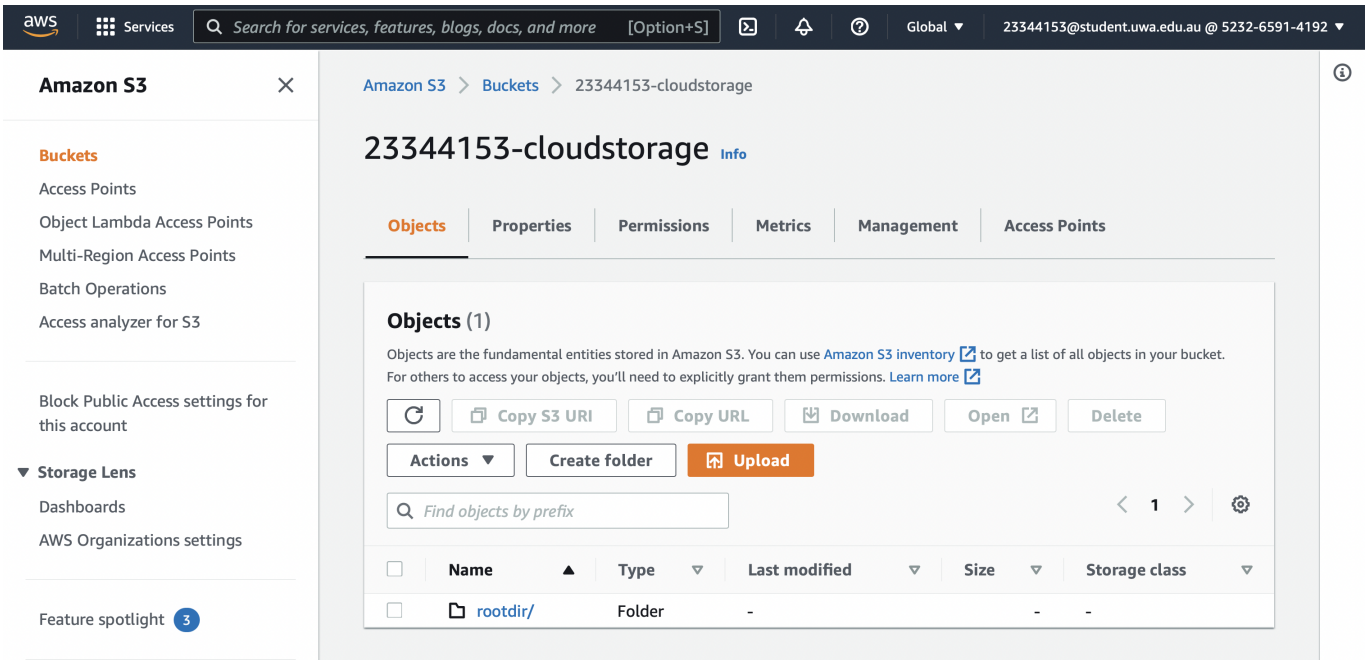
```
'''
cloudstorage.py

Skeleton application to copy local files to S3

Given a root local directory, will return files in each level and
copy to same path on S3
'''
import os
import sys
import boto3
from lab3 import ROOT_S3_DIR

s3 = boto3.resource("s3")
bucket_config = {'LocationConstraint': 'ap-southeast-2'}

def upload_file(file):
    s3.meta.client.upload_file(file, ROOT_S3_DIR, file)
    print("Uploading %s" % file)

# create bucket if not there
argv = sys.argv[1:]
if len(argv) == 1 and argv[0] in ['-i', '--initialise=True']:
    try:
        response = boto3.client("s3").create_bucket(
            Bucket=ROOT_S3_DIR,
            CreateBucketConfiguration=bucket_config,
        )
        print(response)
    except Exception as error:
        print(error)
```

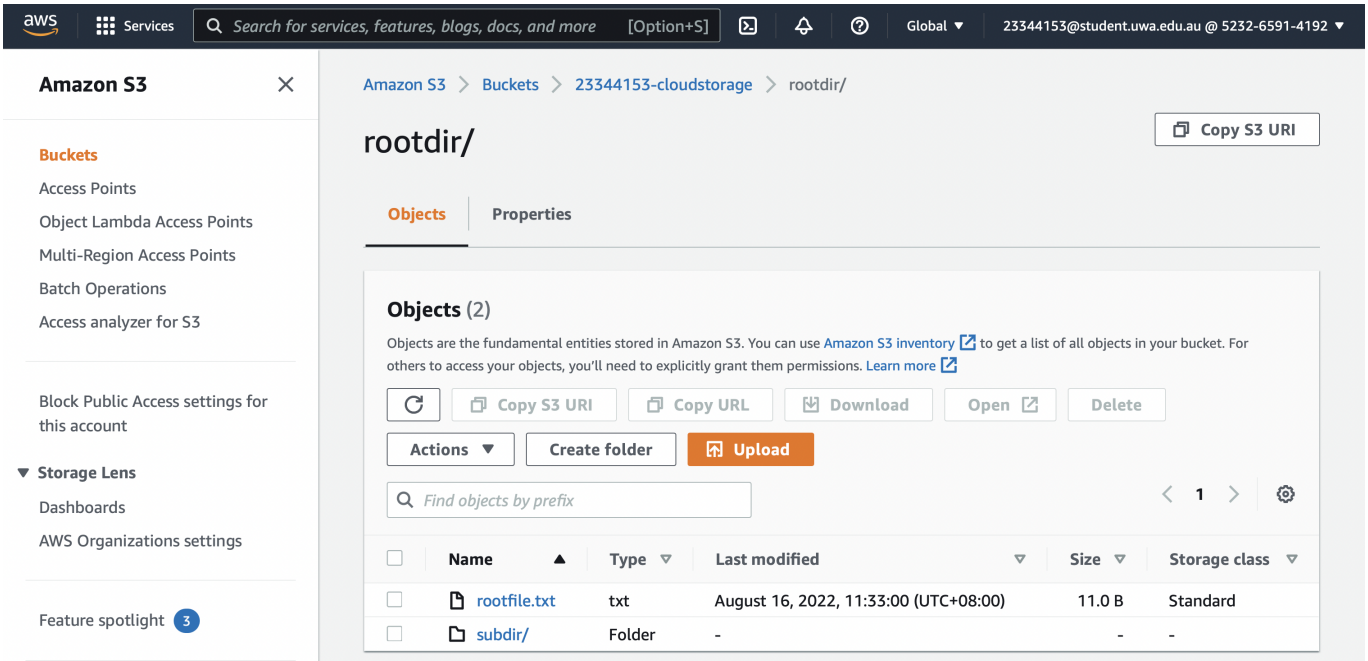
```
# parse directory and upload files
for dir_name, subdir_list, file_list in os.walk('rootdir', topdown=True):
    for fname in file_list:
        upload_file(f'{dir_name}/{fname}')

print("done")
```

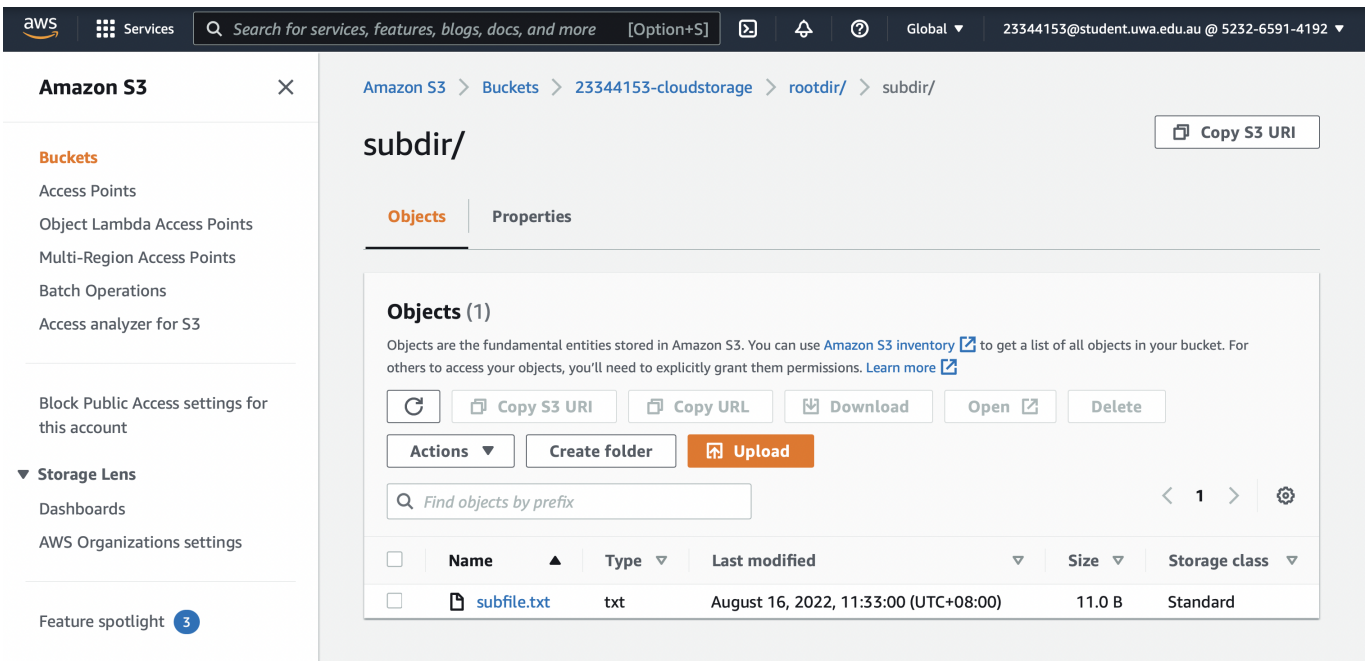
Running `cloudstorage.py`.

```
(venv) >_ Labs/lab3 $ python cloudstorage.py -i
{'ResponseMetadata': {'RequestId': 'BZ8BMSNME0GJPGER', 'HostId': 'K8ImmEnO3HgCDeXVNb3Qw/lNCboR1ol5cI
sjxOBk5Jo8XsELCAoTZOSRv4CQHWjBtKDvsn1N0S0=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'K
8ImmEnO3HgCDeXVNb3Qw/lNCboR1ol5cIsjxOBk5Jo8XsELCAoTZOSRv4CQHWjBtKDvsn1N0S0=', 'x-amz-request-id': 'B
Z8BMSNME0GJPGER', 'date': 'Tue, 16 Aug 2022 03:32:47 GMT', 'location': 'http://23344153-cloudstorage
.s3.amazonaws.com/', 'server': 'AmazonS3', 'content-length': '0'}, 'RetryAttempts': 0}, 'Location':
'http://23344153-cloudstorage.s3.amazonaws.com/'}
Uploading rootdir/rootfile.txt
Uploading rootdir/subdir/subfile.txt
done
(venv) >_ Labs/lab3 $ python cloudstorage.py --initialise=True
An error occurred (BucketAlreadyOwnedByYou) when calling the CreateBucket operation: Your previous r
equest to create the named bucket succeeded and you already own it.
Uploading rootdir/rootfile.txt
Uploading rootdir/subdir/subfile.txt
done
(venv) >_ Labs/lab3 $ python cloudstorage.py
Uploading rootdir/rootfile.txt
Uploading rootdir/subdir/subfile.txt
done
```

If `-i` or `--initialise=True` is passed in as commandline arguments it will try to create a `S2` bucket, if the bucket already exists it will print an error. Otherwise it will just upload the files without trying to create a bucket first.


Services
Search for services, features, blogs, docs, and more [Option+S]
Global
23344153@student.uwa.edu.au @ 5232-6591-4192
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
Amazon S3 > Buckets > 23344153-cloudstorage
23344153-cloudstorage Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
rootdir/
Folder
-
-
-


Services
Search for services, features, blogs, docs, and more [Option+S]
Global
23344153@student.uwa.edu.au @ 5232-6591-4192
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
Amazon S3 > Buckets > 23344153-cloudstorage > rootdir/
rootdir/
Copy S3 URI
Objects
Properties
Objects (2)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
rootfile.txt
txt
August 16, 2022, 11:33:00 (UTC+08:00)
11.0 B
Standard
subdir/
Folder
-
-
-


Services
Search for services, features, blogs, docs, and more [Option+S]
Global
23344153@student.uwa.edu.au @ 5232-6591-4192
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
Amazon S3 > Buckets > 23344153-cloudstorage > rootdir/ > subdir/
subdir/
Copy S3 URI
Objects
Properties
Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
subfile.txt
txt
August 16, 2022, 11:33:00 (UTC+08:00)
11.0 B
Standard

Restore from S3

```
'''restorefromcloud.py'''
import os
import boto3
from lab3 import ROOT_S3_DIR

client = boto3.client("s3")
s3 = boto3.resource('s3')

for content in client.list_objects(Bucket=ROOT_S3_DIR)['Contents']:
    paths = content['Key'].split('/')
    # create the directories listed in the path
    for dir in paths[:-1]:
        if not os.path.isdir(dir):
            os.mkdir(dir)
        os.chdir(dir)
    # download the file from S3 to the deepest directory
    s3.meta.client.download_file(ROOT_S3_DIR, content['Key'], paths[-1])
    # go back to the root directory
    for i in range(len(paths) - 1):
        os.chdir('..')
```

Running `restorefromcloud.py`.

```
(venv) >_ Labs/lab3 $ rm -r rootdir
(venv) >_ Labs/lab3 $ python restorefromcloud.py
(venv) >_ Labs/lab3 $ ls rootdir/*
rootdir/rootfile.txt

rootdir/subdir:
subfile.txt
```

First delete the `rootdir` directory so the file structure can be restored from the cloud exactly the way it was before deleting.

Write information about files to DynamoDB

Create a directory for DynamoDB.

```
(venv) >_ Labs/lab3 $ mkdir dynamodb
(venv) >_ Labs/lab3 $ cd dynamodb
```

Install DynamoDB from `aws` and unzip it.

```
(venv) >_ lab3/dynamodb $ unzip dynamodb_local_latest.zip
Archive:  dynamodb_local_latest.zip
   creating: DynamoDBLocal_lib/
  inflating: DynamoDBLocal.jar
  inflating: DynamoDBLocal_lib/Apache-HttpComponents-HttpClient-4.5.x.jar
  inflating: DynamoDBLocal_lib/Apache-HttpComponents-HttpCore-4.4.x.jar
  inflating: DynamoDBLocal_lib/AwsFlowJava-1.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-Annotations-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-Auth-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-AwsCore-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-AwsJsonProtocol-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-JsonUtils-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-MetricsSpi-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-Profiles-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-ProtocolCore-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-Regions-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-Core-Utils-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-DynamoDb-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-DynamoDb-Enhanced-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-HttpClient-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-HttpClient-ApacheClient-2.0.jar
  inflating: DynamoDBLocal_lib/AwsJavaSdk-HttpClient-NettyNioClient-2.0.jar
  inflating: DynamoDBLocal_lib/DynamoDBPartiQLShared-1.0.jar
  inflating: DynamoDBLocal_lib/GoogleGuava-20.x.jar
  inflating: DynamoDBLocal_lib/IonJava.jar
  inflating: DynamoDBLocal_lib/JMESPathJava-1.12.x.jar
  inflating: DynamoDBLocal_lib/JakartaCommons-logging-1.2.jar
  inflating: DynamoDBLocal_lib/Log4j-api-2.x.jar
  inflating: DynamoDBLocal_lib/Log4j-core-2.x.jar
  inflating: DynamoDBLocal_lib/Log4j-slf4j-2.x.jar
```

Run DynamoDB using Java.

```
(venv) >_ lab3/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:       true
shouldDelayTransientStatuses:   false
CorsParams:     *
```

Create a table on your local DynamoDB with the key `userId`.

For every file that is stored in S3, get the information to put in the DynamoDB item and write it to the table. You will have to find functions in Python to get details like time lastUpdated, owner and permissions. All of this information can be stored as strings.

```
'''lab3.py'''
import boto3

ROOT_S3_DIR = '23344153-cloudstorage'
TABLE_NAME='CloudFiles'

dynamodb = boto3.client('dynamodb', endpoint_url='http://localhost:8000')
s3 = boto3.client("s3")

if __name__ == '__main__':
    # create table if not exists
    try:
        dynamodb.create_table(
            TableName=TABLE_NAME,
```

```
            AttributeDefinitions=[
                {
                    'AttributeName': 'userId',
                    'AttributeType': 'S'
                },
            ],
            KeySchema=[
                {
                    'AttributeName': 'userId',
                    'KeyType': 'HASH',
                },
            ],
            ProvisionedThroughput={
                'ReadCapacityUnits': 1,
                'WriteCapacityUnits': 1
            },
        )
    except Exception as error:
        print(error)

    # scan table to find the number of items
    items = dynamodb.scan(TableName=TABLE_NAME)['Items']
    # set the userId to be 0 if there is no items otherwise the last
item's userId
    userId = int(items[-1]['userId']['S']) if len(items) > 0 else 0

    for content in s3.list_objects(Bucket=ROOT_S3_DIR)['Contents']:
        # the userId key will be the last item's userId + 1
        userId += 1
        path = content['Key']
        filename = path.split('/')[-1]
        # get all the attributes from S3
        response = s3.get_object(
            Bucket=ROOT_S3_DIR,
            Key=path,
        )
        lastUpdated = response['LastModified']
        response = s3.get_object_acl(
            Bucket=ROOT_S3_DIR,
            Key=path,
        )
        owner = response['Owner']['DisplayName']
        permissions = [{ 'S': grant['Permission'] } for grant in
response['Grants']]

        # put the file item in the table
        dynamodb.put_item(
            Item={
                'userId': {
                    'S': str(userId),
                },
                'fileName': {
                    'S': filename,
                },
```

```
                    'path': {
                        'S': path,
                    },
                    'lastUpdated': {
                        'S': str(lastUpdated),
                    },
                    'owner': {
                        'S': owner,
                    },
                    'permissions': {
                        'L': permissions,
                    },
                },
                ReturnConsumedCapacity='TOTAL',
                TableName=TABLE_NAME,
            )
```

Running `lab3.py`.

```
(venv) >_ Labs/lab3 $ aws dynamodb scan --table-name CloudFiles --endpoint-url=http://localhost:8000


An error occurred (ResourceNotFoundException) when calling the Scan operation: Cannot do operations
on a non-existent table
(venv) >_ Labs/lab3 $ aws dynamodb list-tables --endpoint-url http://localhost:8000
{
    "TableNames": []
}
(venv) >_ Labs/lab3 $ python lab3.py
(venv) >_ Labs/lab3 $ aws dynamodb list-tables --endpoint-url http://localhost:8000
{
    "TableNames": [
        "CloudFiles"
    ]
}
```

At first there is no table named `CloudFiles` on DynamoDB, after running `lab3.py` it then exists.

```
(venv) >_ Labs/lab3 $ aws dynamodb scan --table-name CloudFiles --endpoint-url=http://localhost:8000

{
    "Items": [
        {
            "owner": {
                "S": "mdanwarulkaium.patwary"
            },
            "path": {
                "S": "rootdir/rootfile.txt"
            },
            "lastUpdated": {
                "S": "2022-08-17 04:39:13+00:00"
            },
            "fileName": {
                "S": "rootfile.txt"
            },
            "userId": {
                "S": "1"
            },
            "permissions": {
                "L": [
                    {
                        "S": "FULL_CONTROL"
                    }
                ]
            }
            }
        },
        {
            "owner": {
                "S": "mdanwarulkaium.patwary"
            },
            "path": {
                "S": "rootdir/subdir/subfile.txt"
            },
            "lastUpdated": {
                "S": "2022-08-17 04:39:13+00:00"
            },
            "fileName": {
                "S": "subfile.txt"
            },
            "userId": {
                "S": "2"
            },
            "permissions": {
                "L": [
                    {
                        "S": "FULL_CONTROL"
                    }
                ]
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

Scan the `CloudFiles` table using `awscli` and there should be 2 items as there are 2 files uploaded to S3. One is `rootfile.txt` with the full path of `rootdir/rootfile.txt` and the other is `subfile.txt` with the full path of `rootdir/subdir/subfile.txt`.