Nolan Mullins
0939270
Cloud task 4

Task 1

Connecting:

This is the initial connection between client and server. The objects returned will be used by other procedures.

AWS

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

Azure

```
client = BlobServiceClient.from_connection_string(connect_str)
container_client =
ContainerClient.from_connection_string(connect_str,container_name=containerName)
```

Creating container:

Each of these will create and return a reference to a specific container. The main difference here is that if the container exists then Azure will throw an exception and AWS will continue without issue.

AWS

```
client.create_bucket(Bucket=name)
```

Azure

```
client.create_container(name)
```

Uploading a file:

Uploading a file is straightforward for each platform but Azure needs to create a specific blob client for the file. AWS can use its generic client.

AWS

```
client.put_object(Bucket=bucketName, Key=fileName, Body=ffile)
```

Azure

```
blob_client = client.get_blob_client(container=containerName, blob=fileName)
blob_client.upload_blob(file)
```

Get containers:

Each of these returns a list of containers and performs very similarly.

AWS

```
Buckets = s3.buckets.all()
```

Azure

```
containers = client.list_containers(include_metadata=True)
```

Get files from container:

Again getting the objects from a container is very similar between platforms.

AWS

```
objList = s3.Bucket(bucketName).objects.all()
```

Azure

```
blobList = container_client.list_blobs()
```

Task 2

Connecting to database:

Basic api calls needed to connect the the platform. These will be used by subsequent procedures.

AWS

```
dynamodb = boto3.resource('dynamodb', region_name='us-east-1')
client = boto3.client('dynamodb')
```

Azure

```
client = cosmos_client.CosmosClient(endpoint,  {'masterKey': prim_key})
```

Creating a table:

Procedure to create a table on each platform. Aws is a little simpler and only needs you to call create_table() with the structure of the table. Azure requires you to create a database and then put a container inside the database. Each require setup.

AWS

```
table = dynamodb.create_table(
        TableName='Movies',
        KeySchema=[
          {
            'AttributeName': 'year',
            'KeyType': 'HASH'  #Partition key
          },
          {
            'AttributeName': 'title',
            'KeyType': 'RANGE'  #Sort key
          }
        ],
        AttributeDefinitions=[
          {
            'AttributeName': 'year',
            'AttributeType': 'N'
          },
          {
            'AttributeName': 'title',
            'AttributeType': 'S'
          },

        ],
        ProvisionedThroughput={
          'ReadCapacityUnits': 10,
          'WriteCapacityUnits': 10
        }
    )
```

Azure

```
try:
  db = client.CreateDatabase({'id': database_name})
except errors.HTTPFailure:
  #Database already exists

container_definition = {'id': container_name,
                        'partitionKey':
                                {
```

```
                    'paths': ['/year'],
                    'kind': documents.PartitionKind.Hash
                }
            }
try:
    container = client.CreateContainer("dbs/" + database_name, container_definition, {'offerThroughput': 400})
except errors.HTTPFailure as e:
    if e.status_code == http_constants.StatusCodes.CONFLICT:
        #Container exists already
```

Retrieving a table:

How to get a reference to a table. Each are pretty straightforward but again azure requires a little more work.

AWS

```
dynamodb.tables.all()
```

or

```
table = dynamodb.Table('Movies')
```

Azure (Not 100% working for me)

```
db = client.ReadDatabase("dbs/" + database_name)
container = client.ReadContainer("dbs/" + database_name + "/colls/" + container_definition['id'])
```

Table information:

This is a simple request that returns some basic information about the table. In this example we retrieve the number of items in the database.

AWS

```
response = client.describe_table(
    TableName=table.name
)
response['Table']['ItemCount']
```

Azure

Not really sure how to do this in Azure as I haven't been able to get the database to connect

Upload item to table:

Both services will either insert the item in to the database or if the item with the same primary key exists it will update the entry.

AWS

```
table.put_item(
        Item={
            'year': year,
            'title': title,
            'info': info,
            }
        )
```

Azure (Not working)

```
client.UpsertItem("dbs/" + database_name + "/colls/" + container_name,
{
    'year': year,
    'title': title,
    'info': info,
})
```

Query data:

The response will contain a list of items based on the query parameters

AWS

```
    response = table.scan(
        FilterExpression = filter
    )
```

Azure

Haven’t got this far with azure