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

client.UpsertItem("dbs/" + database\_name + "/colls/" + container\_name,

{

'year': year,

'title': title,

'info': info,

})

Check number of entries in a table:

I found an easy way to do this in AWS but could not find an equivalent method in azure

AWS

response = client.describe\_table(

TableName=table.name

)

Count = response['Table']['ItemCount']

Azure

items = client.QueryItems("dbs/" + database\_name + "/colls/" + container\_name,

'SELECT \* FROM ' + container\_name + ' r',

{'enableCrossPartitionQuery': True})

count = 0

For 1 in items:

count = count + 1

Query data:

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

AWS

response = table.scan(

FilterExpression = filter

)

Azure

items = client.QueryItems("dbs/" + database\_name + "/colls/" + container\_name,

'SELECT \* FROM ' + container\_name + ' r '+filter,

{'enableCrossPartitionQuery': True}