Cloud Shape Your Idea

Lab 6: Storage as a Service

The purpose of the lab is to demonstrate concepts presented at the Storage as a Service course using AWS S3 and DynamoDB.

A. Setup an EC2 instance that will host the development environment

Log in into the AWS Console.

- Check you are in region **Ireland**
- Go into the **CloudFormation Console**
- Refresh constantly and wait for the stack **Status** to become CREATE_COMPLETE.
    - In case it becomes ROLLBACK_COMPLETE, call lab assistant.
- Select the stack and check the **Resources** tab
- SSH[1] on the **DevelopmentBox** instance
- Run **aws configure** to configure the AWS CLI
    - Use the **Access Key Id** and **Secret Access Key** provided in the Qwiklab lab
    - Enter **eu-west-1** as the default region
- Validate that you have access to S3 and DynamoDB by running the following commands:

```
$ aws s3 ls
$ aws dynamodb list-tables
```

B. Use S3 to store static content (cat pictures) accessible via the Web 1 point

For this exercise, you can use either the S3 CLI or the S3 Console (the Web UI).

- Create an S3 bucket called *cloudshape-04-07-catpics-<your_name>*
- Upload a cat picture to your S3 bucket.
    - Name your cat picture object appropriately. Make it freely accessible to the world!
    - Hint: use **wget** to first download a cat pic on the **DevelopmentBox**
- Test your result by opening this URL in the browser:
  https://s3-eu-west-1.amazonaws.com/cloudshape-04-07-catpics-<your_name>/<cat_pic_name>

C. Understand eventual consistency in S3 no points – read this **later** for fun

http://docs.amazon.com/AmazonS3/latest/dev/Introduction.html#ConsistencyModel

D. Grasp basic DynamoDB operations 1 point

(for this exercise, don't forget to save the commands you run and the responses you receive)

- **Using the AWS CLI**, let's create a DynamoDB table to store cat picture metadata:
    - Table name: *cloudshape-catpics-<your_name>*
    - Table schema:
      CatID – string / hashkey
      CatURL – string / range key
    - Set provisioned throughput to 5reads and 5writes.
- Use **put-item** to upload your first cat metadata: CatID: 1, CatURL: the URL from exercise B
- Now put the same item with a condition for that CatID to not already exist. What happens?
- **Using the AWS Console**, look at your table and find the operation that will allow you to return cats with IDs between 1 and 5.

[1] If you don't remember how to SSH on the EC2 hosts, check:

- using the **ssh** client: **steps 2. -> 7.** @ [Connecting to Your Linux Instance Using SSH > To connect to your instance using SSH]
- or using Putty: **steps 2. -> 7.** @ [Connecting to Your Linux Instance from Windows Using PuTTY > To start a PuTTY session]

E. Use the S3 triggers to update the DynamoDB table **1.5 point**

What would you say if somehow for every cat picture you uploaded to your S3 bucket, the picture's metadata (id an URL) would automatically be added to your DynamoDB metadata table? Let's create a Lambda function that does that!

E.1 Create a function that is invoked by S3 every time an object is created in your bucket

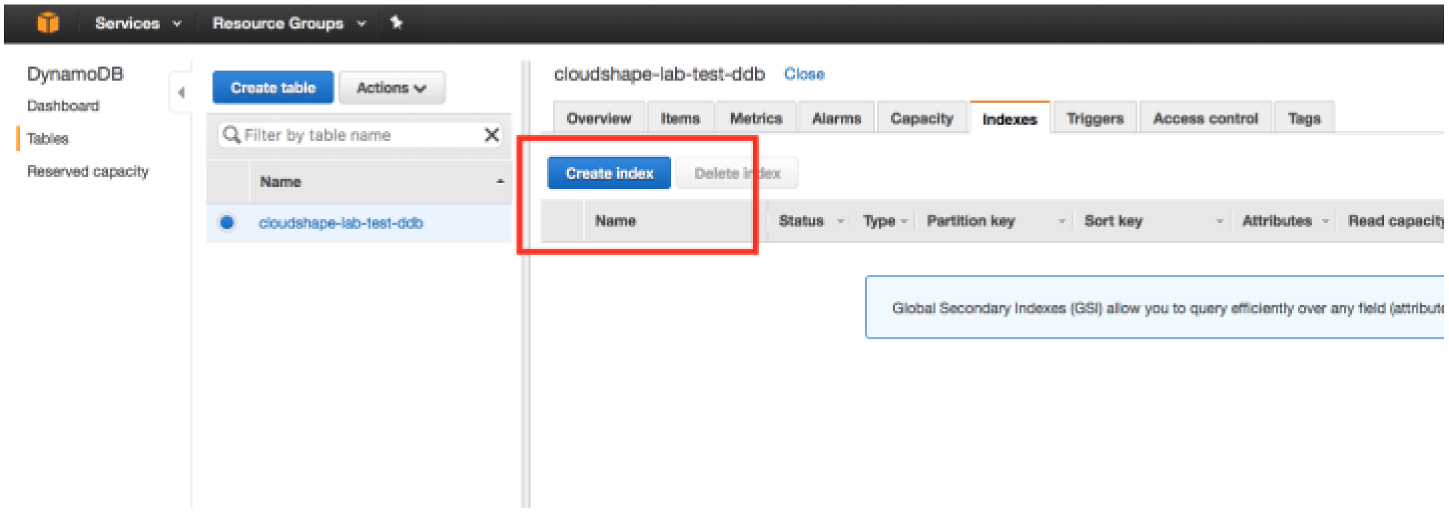
- Go to AWS Console > AWS Lambda > Get started.
- In **Select Blueprint**, search *s3-get-object* and select one of the following:
  - the ***s3-get-object-python*** blueprint for Python
  - the ***s3-get-object*** blueprint for Javascript
- In **Configure triggers**, fill in the following details:
  - Bucket name – the bucket created at B.
  - Event type – Object Created (All) – your Lambda function will be called by S3 when a new object is created in your bucket
  - Prefix – leave it empty
  - Suffix – leave it empty
  - Check the **Enable trigger** checkbox.
- In **Configure function**, set:
  - Name – *cloudshape-function-<yourName>*
  - Role name – *cloudshape-function-role*
  - Leave everything else with their default values.
- Click **Create Function**.
- Use the test event at https://goo.gl/IdJAxm to check your function works:
  - **Actions** button > Configure test event
  - Make sure to replace the strings *<your_bucket_name>* and *<your_cat_pic_object_name>* in the json with your s3 bucket's name and the name of the cat picture file you uploaded at exercise B.
  - After running the test you should see **Execution result: succeeded.**

E.2 Update the Lambda function to write a record in the DynamoDB catpic metadata table

- Use the following data from the s3 record to create the DynamoDB metadata entry:
  CatID – s3 object **eTag**
  CatURL – s3 object **key**
  EventTime – new string field, **eventTime**
- At this point, you can try testing the function using the test event, but it will fail due to access issues. When the function was created, it also created and associated with the function a IAM Role that would grant it read access to S3, as predefined by the template, but your updated function also needs write access to DynamoDB. You'll need to update the IAM Role associated with this function to allow it full access to DynamoDB.
  - Go to the IAM Console > Roles
  - Click on the *cloudshape-function-<yourName>* role
  - Click **Attach Policy**
  - Search and select **AmazonDynamoDBFullAccess**. Click **Attach Policy**.
- Retry testing using the test event. **Execution should succeed**.
- Test the updated function by inserting 5 new cat images in your S3 bucket using the AWS CLI or the AWS Console.
- Go to the **DynamoDB Console** and confirm that the DynamoDB table was updated with the new metadata entries.

F. DynamoDB table SCAN vs QUERY **1 point**

- From the AWS Console or the AWS CLI, use the DynamoDB SCAN function along with a condition on the EventTime field to get the elements introduced after a certain date (Hint: for the CLI see dynamodb scan documentation)
- Using the DynamoDB Console, create a GlobalSecondaryIndex on the EventTime field (see print screen below) and achieve the same result using the DynamoDB Query function.



Global Secondary Indexes (GSI) allow you to query efficiently over any field (attribut

G. Use AWS Athena to process S3 data **0.5 point**

Change the console region to **US East (N. Virginia)**

Using the S3 AWS Console:

- Create a new S3 bucket called *cloudshape-storage-athena-<your_name>* in **US East (N. Virginia)**
- Download the cats.zip archive with cat json files from https://goo.gl/ODUtd7 and unarchive it.
- Create a folder named ***cats*** in the bucket.
- Upload the cats_[1..10].json files to the ***cats*** folder.

Using the Athena AWS Console:

- Create a new database and table definition based on the data included in the JSON file by running the following query:

```
CREATE EXTERNAL TABLE IF NOT EXISTS sampledb.cats_table (
 `CatID` int,
 `CatURL` string,
 `CatAge` int
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
 'serialization.format' = '1'
) LOCATION 's3://cloudshape-storage-athena-<your_name>/cats/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

- Confirm that the table was created correctly by previewing the first elements.
    - In the column on the left, select the **sampledb** database
    - Click the little eye symbol for the **cats_table** table.

- Run a SQL queries to get:

1. How many cats are in the table.
2. How many cats are older than 7 years.
3. How many cats are between 3 and 6 years old.