Exercise: Creating an Amazon DynamoDB Table using the AWS Software Development Kit (AWS SDK)

Overview

In this exercise, you will learn how to *develop* with Amazon DynamoDB by using the AWS Software Development Kit (AWS SDK). Following the scenario provided, you will create a DynamoDB table. This exercise gives you hands-on experience with both Amazon DynamoDB and AWS Cloud9.

Objectives

After completing this exercise, you will be able to use the AWS SDKs to do the following:

- Create a DynamoDB table.

Story

One of your best friends (Mary) who you've known for years has come to you for help.

Mary has always been obsessed with fantasy dragon card games, and she has recently set up a gaming company and her latest project is a new dragon card game being developed internally by a local group of enthusiasts (who are now her employees).

They don't have a website for their game yet. They have been doing everything manually, even drawing the cards themselves.

The problem is none of them are very tech savvy, which is why Mary reached out to you.

There is currently no way to look up card details online.

Mary has asked you to help her create a simple webpage that can show all the dragon cards when the page loads, and to be able to show any card and all of its information by doing a simple search on a dragon name.

You don't even ask what budget she has, as you already know the answer. Zero.

Besides, you want the practice, as you are sitting your developing on AWS exam shortly.

The plan

You figure that a basic HTML page with a search box, and some CSS and a bit of AJAX to a database would probably work as a prototype. You decide to set that up on your own account inside your AWS free tier.

You recently took a serverless course, and feel obliged to make this whole serverless just out of principle.

As you are planning on sitting your AWS developer associate exam shortly, you decide to force yourself to do most of these activities using the AWS-SDK and get comfortable with code.

You think the best place to start is to create a DynamoDB table with an open schema. This way you can decide on the schema once you know what the data looks like, and how you plan on querying the data.

You imagine it will look a bit like this, as dragon names are unique.

Primary Key (dragon_name)	
sparky	...
tallie	...

Prepare the exercise

Before you can start this exercise, you need to import some files and install some modules in the AWS Cloud9 environment that you will create.

1. From the AWS Management Console, go to the **Services** menu and choose **Cloud9**.
2. Choose **Create environment**.
3. Under **Name**, input `dynamolab`.
4. Choose **Next step**, **Next step**, **Create environment**.
5. Choose **Open IDE** to open the AWS Cloud9 environment.
6. To seed your Cloud9 filesystem, go to the Cloud9 **bash terminal** (at the bottom of the page) and run the following `wget` command: *You should be in /home/ec2-user/environment*

```
wget \
https://s3.amazonaws.com/awsu-hosting/edx_dynamo/c9/dynamo-create/lab1.zip \
-P /home/ec2-user/environment
```

You should also see that a root folder called **dynamolab** with a `lab1.zip` file has been downloaded and added to your AWS Cloud9 filesystem (on the top left). There is also a `README.md` file in there by default, which you will remove in a bit.

7. To unzip the `lab1.zip` file, run the following command:

```
unzip lab1.zip
```

8. To keep things clean, run the following commands to remove the zip and README files. Install any dependencies in the root and node folder. Then navigate to the lab1 folder path (*/home/ec2-user/environment/lab1*) in the terminal by using these commands.

    ```
    rm lab1.zip
    rm README.md
    cd lab1
    npm install aws-sdk
    echo "done"
    ```

9. Once you see "done", select the black arrow next to the `lab1` folder (far left top) to expand it. Notice that there is a solution folder. **Try not to peek at the solution unless you really get stuck. Always TRY to code first.**
10. You should see that some packages and modules have been installed. Ignore any warnings in the terminal.

As this course is self-paced, often people will start the lab then come back to it later. In the interim period, we may have made adjustments to the code.

Ensure that inside your lab1 folder that the name of the version markdown file is matching the version number at the top of this document.

If they are out of sync, you will run into problems. To get them synced, simply remove the old folder and run through the wget steps above one more time.

You are now ready to do the exercise tasks.

Step 1: Create a DynamoDB table using the AWS SDK

You want to use the AWS SDK to create a DynamoDB table. You figure the method name is probably something a bit like **create table**, but you check the AWS SDK documentation anyway.

1. From the table below, open the link to the method for creating DynamoDB tables in the AWS SDK documentation. Confirm the method name and establish what parameters you need to pass in.

Language	AWS SDK Documentation deep link
Node.js (8.16.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#createTable-property

Time to write some code that creates a new DynamoDB table.

In the AWS Cloud9 environment, do the following:

2. Open (double-click) the `create_table.js` file inside the `lab1` folder (not the one inside the solution folder).
3. Using the SDK documentation to help you, simply replace the <FMI> sections of the code in that file so that the code will create a table called **dragons** in **us-east-1**.

*Tip: You think initially you will only need to search for a dragon by its name, so you figure that the best choice for the primary key will be `dragon_name` with no sort key. You are not sure how often this table will be used so you think it best to use **dynamic capacity** rather than guessing the necessary provisioned capacity.*

4. Save the `create_table.js` file.
5. **LAB TIP:** Before you run this script, you should double check the solution code first. This way when you run the file it will just work. It is very easy to end up with some parts of a script working and some not, due to a typo or a small mistake, and end up with partial or inaccurate resources being created. It is better to check your code meticulously against the solution code before actually executing your code, rather than trying to undo partial changes after the script has run.
6. In the Cloud9 terminal, type this run command, *rather* then just pressing the run button.

```
node create_table.js
```

Confirm that your code worked.

In the terminal you should see something a bit like this:

```
null { TableDescription:
   { AttributeDefinitions: [ [Object] ],
     TableName: 'dragons',
     KeySchema: [ [Object] ],
     TableStatus: 'CREATING',
     CreationDateTime: 2019-06-03T13:38:16.594Z,
     ProvisionedThroughput:
      { NumberOfDecreasesToday: 0,
        ReadCapacityUnits: 0,
        WriteCapacityUnits: 0 },
     TableSizeBytes: 0,
     ItemCount: 0,
     TableArn: 'arn:aws:dynamodb:us-east-1:000000000000:table/dragons',
     TableId: '4b78ab3a-288b-46a4-b3df-74395804dfbc',
     BillingModeSummary: { BillingMode: 'PAY_PER_REQUEST' } } }
```

We can also verify that the table was created in the DynamoDB console.

1. Choose **AWS Cloud9** in the upper left.

2. Then choose **Go To Your Dashboard**.
3. Choose **Services** and search for **dynamo** then choose **DynamoDB** to pivot to the console.
4. Choose **Tables**.

If your code worked, you will see that your table has been (or is being) created in the US East (N. Virginia) Region.

Select the name of your table. In the **Overview** tab, you should see something similar to this:

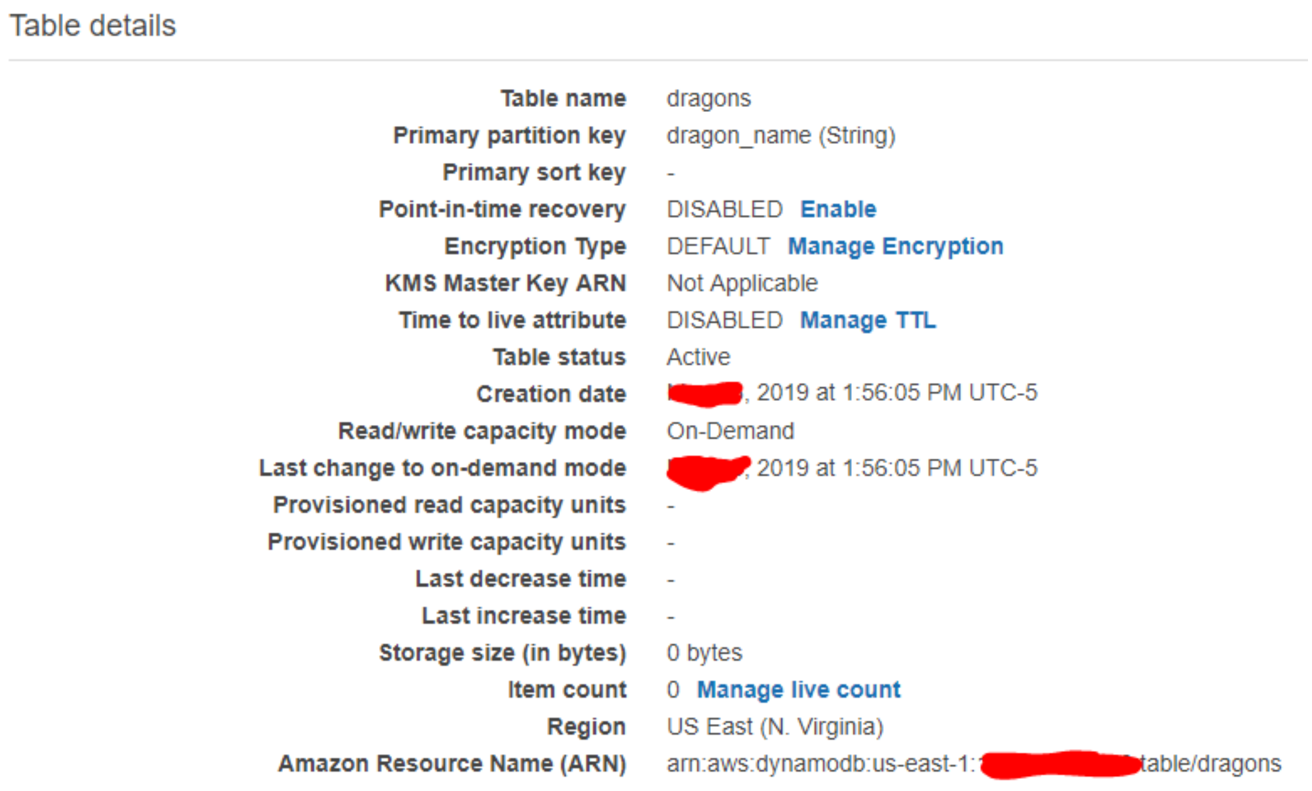

Table details

Table name	dragons
Primary partition key	dragon_name (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	[redacted], 2019 at 1:56:05 PM UTC-5
Read/write capacity mode	On-Demand
Last change to on-demand mode	[redacted], 2019 at 1:56:05 PM UTC-5
Provisioned read capacity units	-
Provisioned write capacity units	-
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0 Manage live count
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:[redacted]table/dragons

IF YOU GET STUCK, OR YOUR CODE DOES NOT WORK, LOOK AT THE SOLUTION CODE.

Congratulations! You have completed exercise 1. You have created a DynamoDB table in the N.Virginia region.

Now you are ready to add Dragons!