Using Amazon DynamoDB with the AWS CLI

The AWS Command Line Interface (AWS CLI) provides support for all of the AWS database services, including Amazon DynamoDB. You can use the AWS CLI for ad hoc operations, such as creating a table. You can also use it to embed DynamoDB operations within utility scripts.

For more information about using the AWS CLI you can view the documentation from the following weblink. AWS CLI Command Reference — AWS CLI 2.1.39 Command Reference (amazonaws.com). You may be able to find the documentation specific to using DynamoDB on your own from the reference guide or use the following weblink dynamodb — AWS CLI 2.1.39 Command Reference (amazonaws.com).

Prerequisites

To run the `dynamodb` commands, you need to have:

- AWS CLI installed, see resources from week 5 to revisit installing AWS CLI.
- AWS CLI configured, see resources from week 5 to revisit configuring AWS CLI.

Help

To list the AWS CLI commands for DynamoDB, use the following command.

```
$ aws dynamodb help
```

Creating and using DynamoDB tables

The command line format consists of a DynamoDB command name, followed by the parameters for that command.

The AWS CLI supports the CLI shorthand syntax for the parameter values, and full JSON.

The following example creates a table named `MusicCollection`.

```
$ aws dynamodb create-table \
    --table-name MusicCollection \
    --attribute-definitions AttributeName=Artist,AttributeType=S
    AttributeName=SongTitle,AttributeType=S \
    --key-schema AttributeName=Artist,KeyType=HASH
    AttributeName=SongTitle,KeyType=RANGE \
    --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

You may like to revisit table creation with boto3 from last week's jupyter notebook now and compare the syntax.

Once you have compared the syntax on basic table creation you may want to go ahead compare notes on the following CLI syntax from DynamoDb with boto3's python syntax for the same operation.

```
$ aws dynamodb put-item \
    --table-name MusicCollection \
    --item '{
        "Artist": {"S": "No One You Know"},
        "SongTitle": {"S": "Call Me Today"} ,
        "AlbumTitle": {"S": "Somewhat Famous"}
    }' \
```

```
        --return-consumed-capacity TOTAL
        {
                { "ConsumedCapacity":  {
                        "CapacityUnits": 1.0,
                        "TableName": "MusicCollection"
                }
        }

$ aws dynamodb put-item \
        --table-name MusicCollection \
        --item '{
                "Artist": {"S": "Acme Band"},
                "SongTitle": {"S": "Happy Day"} ,
                "AlbumTitle": {"S": "Songs About Life"}
        }' \
        --return-consumed-capacity TOTAL
{
        { "ConsumedCapacity": {
                "CapacityUnits": 1.0,
                "TableName": "MusicCollection"
        }
}
```

It can be difficult to compose valid JSON in a single-line command.

To make this easier, the AWS CLI can read JSON files.

For example, consider the following JSON snippet, which is stored in a file named `expression-attributes.json`.

```
{
        ":v1": {"S": "No One You Know"},
        ":v2": {"S": "Call Me Today"}

}
```

You can use that file to issue a `query` request using the AWS CLI.

In the following example, the content of the `expression-attributes.json` file is used as the value for the `--expression-attribute-values` parameter.

```
$ aws dynamodb query --table-name MusicCollection \
   --key-condition-expression "Artist = :v1 AND SongTitle = :v2" \
   --expression-attribute-values  file :/ / expression-attributes.json
{
   "Count": 1,
   "Items": [
      {
         "AlbumTitle": {
            "S": "Somewhat Famous"
         },
         "SongTitle": {
            "S": "Call Me Today"
         },
         "Artist": {
            "S": "No One You Know"
         }
```

```
      }
   ],
   "ScannedCount": 1,
   "ConsumedCapacity": null
}
```


(please note there aren't supposed to be any spaces in the file path)