

Assignments 7 and 8 – Lambda with CRUD operations

Serverless is awesome and costs \$0. You don't have to delete them. Do assignments in your personal account.

Upload screenshots of:

- Source code for all CRUD operations. Otherwise, you will get 1 out of 2 points.
- Records in the database table
- Responses after testing for each CRUD operations.

Check out the “[Be a better dev](#)” channel. It is a great channel for serverless services.

Task 0. Practice version, alias, and weighted (canary) deployment

Task 1. Create a DynamoDB table, Lambda, and implement the Save functionality

- Create an IAM policy that allows CRUD operations on the table.
- Create a DynamoDB table, “CourseTable”.
 - a. courseCode -> Partition key
 - b. teacherName -> Sort key
- Implement the PutItem (Create) operation in the Lambda. Add some columns such as (just add in the Lambda code, DynamoDB is flexible and schemaless)
 - a. courseName
 - b. month – Which month of the year the course was taught
 - c. year
 - d. students – String Set

Task 2 – Implement the rest of the CRUD operations

1. Update the Course Lambda to do the rest of the **CRUD** operations. Scan items with filters.
 - a. **GetItem** and **DeleteItem** – Relatively simple. You just need to provide the composite key (courseCode and teacherName)
 - b. **Scan** - Get items with some filter for example, month, and year. You can get these values as a query parameter or path parameter.
 - c. **Query** – get an item based on the partition key. It can also efficiently query on the index if you have one.
 - d. **UpdateItem** - update (month, and year) of the item. You need to provide the composite key (courseCode and teacherName) to update a specific item.

You will implement the rest of the CRUD operations in the coming days. You can do your own research on the DynamoDB CRUD APIs by referring

- [Official documentation](#)
- [SDK](#)
- [Internet blogs](#)

Sample code for creating an item.

```
import {
  DynamoDBClient,
  PutItemCommand,
} from "@aws-sdk/client-dynamodb";

const dynamodb = new DynamoDBClient({
  apiVersion: "2012-08-10"
});

export const handler = async(event) => {
  const saveParameters = {
    TableName: 'prep',
    Item: {
      "courseCode": {
        S: 'CS516'
      },
      "courseName": {
        S: 'CC'
      },
      "teacherName": {
        S: 'Uno'
      }
    }
  };

  const command = new PutItemCommand(saveParameters);
  await dynamodb.send(command);

  const response = {
    statusCode: 200,
    body: 'success'
  };

  return response;
};
```