Serverless Web Apps using Amazon DynamoDB - Part 2

**SPL-133 - Version 2.0.15**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Overview**

Continue the mission! In this lab, you will continue building a mission dossier generator using DynamoDB. This is Part 2 of a three-part series of labs. You already created a DynamoDB table and set up IAM in Part 1. In Part 2, this lab, you will create and test Lambda functions that interact with DynamoDB and retrieve data in several different ways. You will then test your new functions. In Part 3, you will publish the app, utilizing all of the DynamoDB skills you learn through the three labs, and send your team of super heroes off to save the world!

**Topics covered**

By the end of this lab you will be able to:

* Create Lambda functions that retrieve data in a DynamoDB table in two different ways
* Test Lambda functions with a template
* Configure custom tests for Lambda functions

**Prerequisites**

You should familiarize yourself with key concepts by taking the **Introduction to Amazon DynamoDB** lab.

This is the second in a three-part series of labs. Each of the three labs can stand alone, but because the labs build upon what you learned previously, you should take the **Serverless Web Apps using Amazon DynamoDB - Part 1** prior to taking this lab.

OTHER AWS SERVICES

Other AWS Services than the ones needed for this lab are disabled by an IAM policy during your access time in this lab. In addition, the capabilities of the services used in this lab are limited to what’s required by the lab and in some cases are even further limited as an intentional aspect of the lab design. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

AMAZON DYNAMODB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications.

You can use Amazon DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic. Amazon DynamoDB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent and fast performance.

DYNAMODB TERMINOLOGY

**Tables** Similar to other database management systems, DynamoDB stores data in tables. A table is a collection of data. For example, in this lab you will work with a table named SuperMission, where mission information is stored. **Items** Each table contains multiple items. An item is a group of attributes that is uniquely identifiable among all of the other items. In the SuperMission table, each item represents Mission information. **Attributes** Each item is composed of one or more attributes. An attribute is a fundamental data element, something that does not need to be broken down any further. In the SuperMission table, items have attributes like SuperHero, MissionStatus, Villain1, etc. Attributes in DynamoDB are similar in many ways to fields or columns in other database management systems. **Primary Key** When you create a table, in addition to the table name, you must specify the primary key of the table. As in other databases, a primary key in DynamoDB uniquely identifies each item in the table, so that no two items can have the same key. When you add, update, or delete an item in the table, you must specify the primary key attribute values for that item. The key values are required; you cannot omit them. DynamoDB supports two different kinds of primary keys: **Partition Key** and **Partition Key and Sort Key**. **Secondary Indexes** In DynamoDB, you can read data in a table by providing primary key attribute values. If you want to read the data using non-key attributes, you can use a secondary index to do this. After you create a secondary index on a table, you can read data from the index in much the same way as you do from the table. By using secondary indexes, your applications can use many different query patterns, in addition to accessing the data by primary key values.

AWS LAMBDA

AWS Lambda is a compute service that provides resizable compute capacity in the cloud to make web-scale computing easier for developers. Upload your code to AWS Lambda and AWS Lambda takes care of provisioning and managing the servers that you use to run the code. AWS Lambda supports multiple coding languages: Node.js, Java, or Python. You can use AWS Lambda in two ways:

* As an event-driven compute service where AWS Lambda runs your code in response to events, such as uploading image files as you’ll see in this lab.
* As a compute service to run your code in response to HTTP requests using Amazon API Gateway or API calls.

AWS Lambda passes on the financial benefits of Amazon’s scale to you. Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second. Lambda makes it easy to build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Amazon Kinesis, or create your own back end that operates at AWS scale, performance, and security.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Check CloudFormation Status**

1. In the AWS Management Console, use the **AWS search bar** to search for

**CloudFormation**

 and then choose the service from the list.

1. You should see one stack with a status of **CREATE\_IN\_PROGRESS**.
2. Wait for the status to change to **CREATE\_COMPLETE** before you proceed.

You can click the **refresh** button to refresh the screen.

**Note:** If you do not see a stack being created, double check that you are in the “Oregon” region.

The CloudFormation template is preparing what you need to complete this lab. It may take a few minutes for all the resources to be ready. While you wait, you can observe the creation process for the following resources.

**Task 2: Create Your Lambda Functions**

In this task, you will create two Lambda functions. The first will retrieve a list of super heroes that are stored in the **SuperMission** DynamoDB table. The second will retrieve the mission details from the same DynamoDB table.

1. In the AWS Management Console, use the **AWS search bar** to search for

**Lambda**

 and then choose the service from the list.

1. Click **Create function**
2. Select **Author from scratch**, then configure:

* **Function name:**

getheroeslist

* **Runtime:** *Node.js 18.x*
* Expand  **Change default execution role**
* **Execution role**: Use an existing role
* **Existing role:** *SuperDynamoDBScanRole*
* Click **Create function**

1. Click **Code** tab and then in the **Code source** section, **delete** existing code in *index.mjs*
2. Within the **index.mjs** tab, paste the code shown below:

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

import { DynamoDBDocumentClient, ScanCommand } from "@aws-sdk/lib-dynamodb";

const dynamodb = new DynamoDBClient({});

const docClient = DynamoDBDocumentClient.from(dynamodb);

export const handler = async (event, context) => {

const command = new ScanCommand({

TableName:'SuperMission'

});

const response = await docClient.send(command);

return response.Items;

};

1. Examine the code and review what the function does.

This function retrieves a list of super heroes stored in the **SuperMission** DynamoDB table.

1. Click **Deploy**
2. Click **Test** tab next to the **Code** tab and then configure:

* **Event name:**

myTest

1. In the editor, remove all of the placeholder code except

{}

Removing **{}** will result in error *There is an error in your JSON event. Correct it before saving.*

1. Click **Save**
2. Click **Test**
3. Scroll to the top of the page.

You will see a message telling you that **Execution Result: Succeeded**.

1. Expand  **Details**.

You will see a text box displaying the contents of the SuperMission DynamoDB table.

 Congratulations! You created and tested a Lambda function that retrieve list of Super heroes that are stored in the **SuperMission** DynamoDB table.

Next, you will create a second Lambda function to retrieve the mission details from the SuperMission DynamoDB table.

1. At the top of the page, click **Functions**
2. Click **Create function** then configure:

* **Function name:**

getmissiondetails

* **Runtime:** *Node.js 18.x*
* Expand  **Change default execution role**
* **Execution role:** Use an existing role
* **Existing role:** *SuperDynamoDBQueryRole*
* Click **Create function**

1. Click the **Code** tab and then then in the **Code source** section, **delete** existing code in *index.mjs*
2. Within the **index.mjs** tab, paste the code shown below:

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

import { QueryCommand, DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const docClient = DynamoDBDocumentClient.from(client);

export const handler = async (event, context) => {

try {

const command = new QueryCommand({

TableName:'SuperMission',

ProjectionExpression:"SuperHero, MissionStatus, Villain1, Villain2, Villain3",

KeyConditionExpression: "SuperHero = :superHero",

ExpressionAttributeValues: {

":superHero": event.superhero

}

});

const response = await docClient.send(command);

return response.Items;

}

catch (err) {

console.log(err)

}

};

1. Examine the code and review what the function does.

This function retrieves the mission details from the DynamoDB table.

1. Click **Deploy**
2. Click **Test** tab next to the **Code** tab and then configure:

* **Event name:**

myTest2

1. In the editor, remove all of the placeholder code.
2. Copy the code below and paste it into the editor.

This code looks for details of Batman’s mission in the DynamoDB table.

{

"superhero": "Batman"

}

1. Click **Save**
2. Click **Test**

Near the top of the page, you will see a message telling you that **Execution Result: Succeeded**. Below that, there will be a text box displaying the mission details for “Batman”.

1. Expand  **Details**.
2. Review the details.

 Congratulations! You created and tested a Lambda function that retrieves mission details from the DynamoDB table.

**If you have time**

Go back to the last test you ran. Replace “Batman” with another super hero (Superman, Iron Man) and practice retrieving details for other missions.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Conclusion**

 Congratulations! You have completed this lab. You have successfully:

* Created Lambda functions that retrieve data in a DynamoDB table
* Tested Lambda functions using a template
* Configured custom tests for Lambda functions

**Additional Resources**

* [Amazon DynamoDB](https://aws.amazon.com/documentation/dynamodb)
* [AWS Lambda](https://aws.amazon.com/documentation/lambda)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).