

Server-less micro web application developed using AWS components

The Simpson's Database

powered by 

Enter the ID here:

Click me



AWS serverless web app by Jeffy Abraham

By:
Jeffy Abraham
20082962

Table of Content

1.	Introduction
2.	Overall Architecture
3.	AWS Components
	3.1 Api Gateway
	3.2 Lambda Function
	3.3 DynamoDB
	3.4 SNS
	3.5 S3 Bucket
4.	Screenshot and Supporting Documents
	4.1 Front End Screenshots
	4.2 Api Gateway Screenshots
	4.3 DynamoDB Screenshot
	4.4 SNS Screenshot
	4.5 S3 Bucket Screenshot
	4.6 Overall working
5.	References

1.Introduction

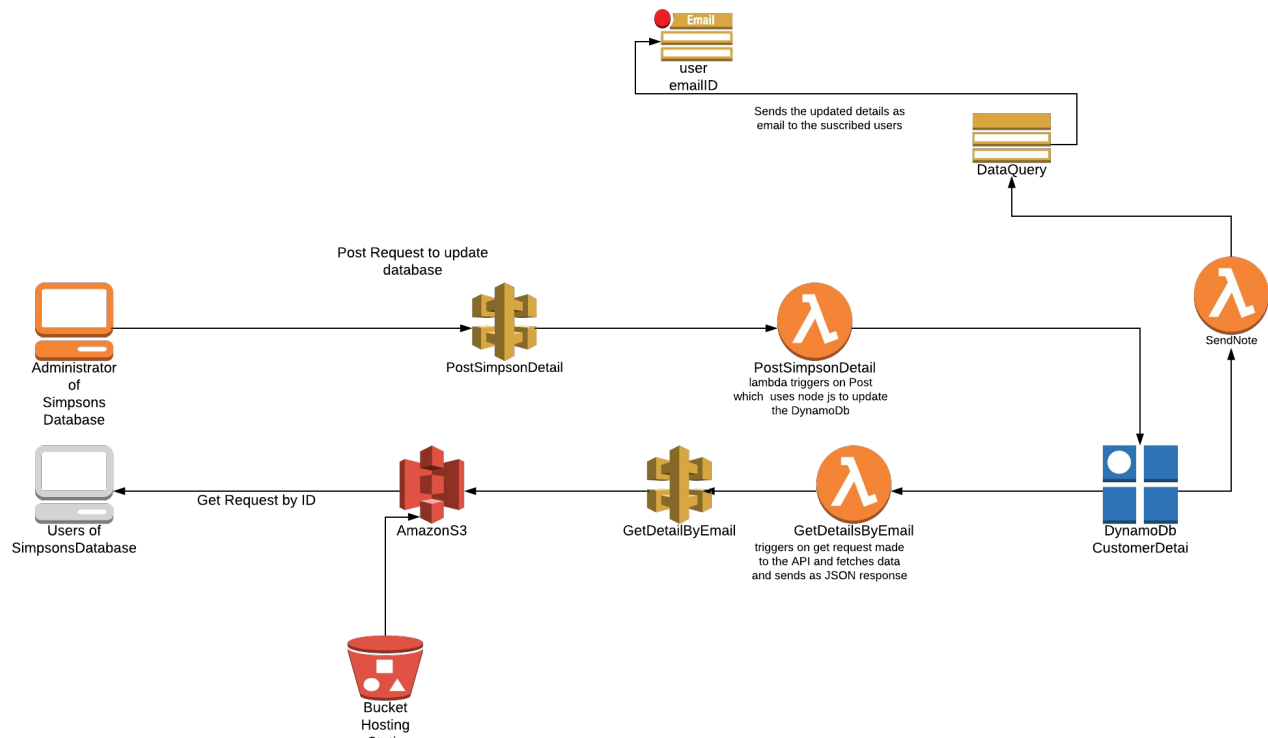
Server-less is an architecture where the application is executed without visible Operating system whereas micro-service is a collection of loosely coupled services .There are many advantages of using a server-less architecture over traditional ones. Some are

- 1.High Availability
- 2.Scale up and down easily
- 3.Lower operational cost
- 4.Can be integrated with micro services
- 5.Resources are only used when triggered(lambda)

For the assignment I have created a Web application database for the one of the world's most popular animated sitcom "The Simpson".The application has a webpage which can be used by users to search for details of their favourite Simpson character. In addition to that the administrator can also add data to the DynamoDB database of any new characters that are missing from the database. Each and every time the database is updated a SNS message is sent to the email id subscribed to the topic. The webpage is hosted in S3 bucket.

The architecture use various AWS components to make this application completely serverless. Furthermore it takes the advantage of lambda function to make this application a microservice. The lambda function are trigged when specific condition are met e.g.: get request to the API gateway or updating DynamoDB.

2.Overall Architecture



In the architecture we have a bucket which hosts the website <http://simpsonsdatabase.s3-website-us-east-1.amazonaws.com/>. Whenever a text is entered into the textbox and the click me button is pressed a “get” request is made to the API “GetCustomerdetailbyemail” <https://4re0d6upu1.execute-api.us-east-1.amazonaws.com/Dev/getcustomerdetailsbyemail?EmailID=Marge Simpson>. This triggers the lambda and sends a JSON response of the data stored in the database.

The administrator of the website can enter the data into the web page by put request(“PostSimpsonDetail”). Postman is used to enter the Simpsons data which triggers the lambda and furthermore puts the data into the DynamoDB.

DynamoDB is connected to a trigger “SendNote” which sends SNS notification to the subscribed email id about the newly added character .

3 AWS Components

3.1 API Gateway:

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. API Gateway has no minimum fees or start-up costs. You pay only for the API calls you receive and the amount of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scale

Here for this Assignment I have created 2 API(refer screen shot 4.2)

1.GetCustomerDetailbyEmailId

2.PostSimpsonDetail

1.GetCustomerDetailbyEmailId:

It has a GET method attached to it and it is associated with the lambda

GetCustomerDetailbyEmailId

2.PostSimpsonDetail

It has a POST method attached to it and is associated with the lambda “PostSimpsonDetails”

Here when ever a GET request is made it will trigger the lambda “GetCustomerDetailbyEmailId”

it will fetch the data from the DynamoDB and will send the data to the user browser, Similarly for the POST request will trigger the PostSimpsonDetail and will update the data base with the new database

links to API

<https://4re0d6upu1.execute-api.us-east-1.amazonaws.com/Dev/getcustomerdetailsbyemail?EmailID=Marge Simpson>

<https://kta7hfibcd.execute-api.us-east-1.amazonaws.com/Dev/postsimpsonsdetails>

3.2 Lambda Functions:

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume there is no charge when the code is not running.

Lambdas used for the assignment(refer screen shot 4.3)

All together 3 lambdas are used (code provided as .txt)

1.PostSimpsonDetail(node js)

2.getCustomerDetailsByEmail (node js)

3.SendNote (python 2.7)

The functionalities of first 2 lambdas has already been explained.

The SendNote is attached to the DynamoDB CustomerDetails. Whenever data is updated in the DynamoDB it sends a SNS message to the subscribed email IDs.

3.3 Dynamo DB:

Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multi-region, multi-master database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second.

We have a table called "CustomerDetail" with primary key set on EmailID and has attributes like Characters "Husband", "Wife", "Children"(refer 4.4)

3.4 SNS

In SNS we have a topic call "DataQuery" .It has a subscriber with email Id jeffybrhm@gmail.com so whenever the DynamoDB is updated a notification is sent to this mail.(refer 4.5)

3.5 S3 Bucket

The S3 bucket is used to host the static website

<http://simpsonsdatabase.s3-website-us-east-1.amazonaws.com/>

4.1 Front end Screenshot

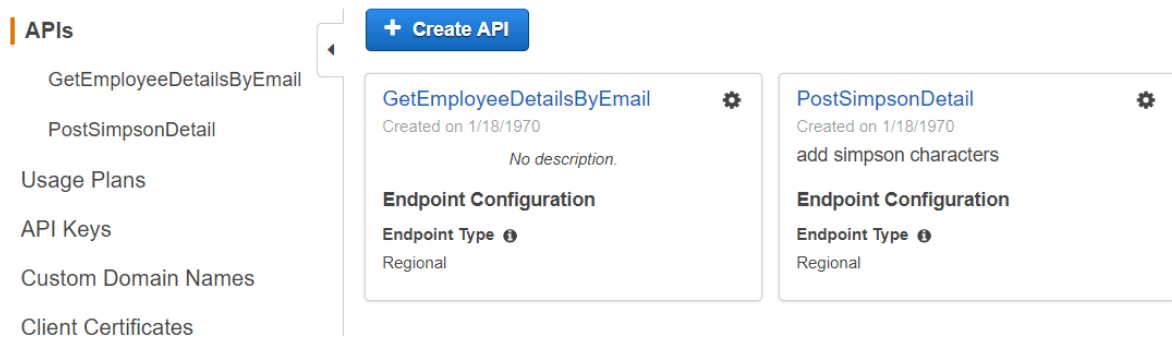


After press of button

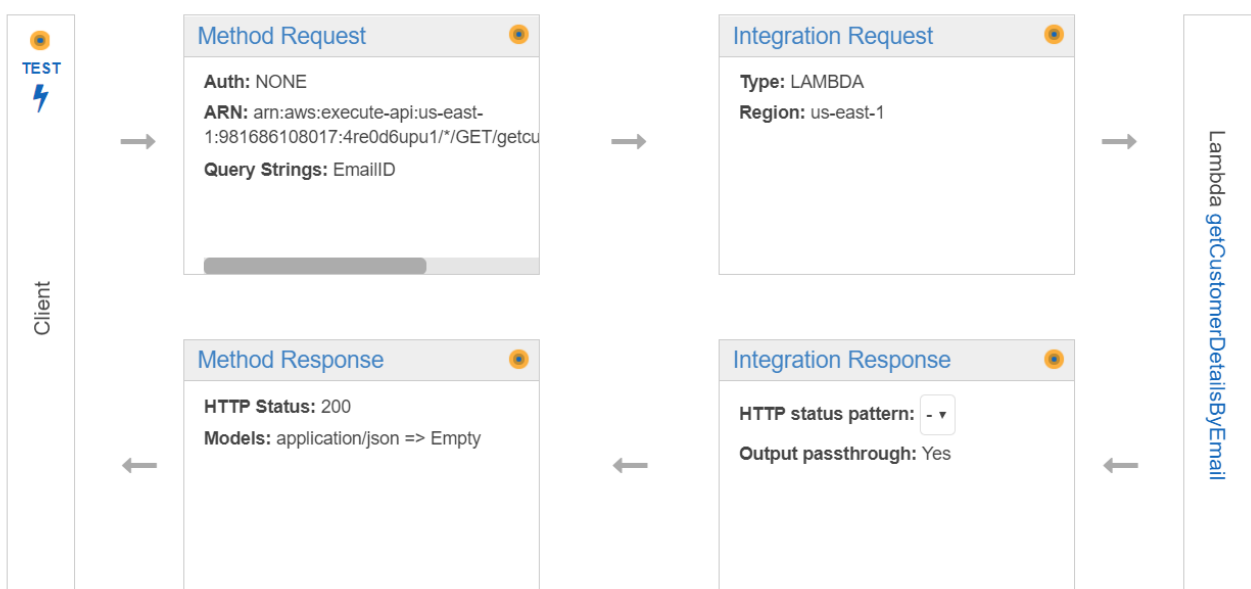
```
{"Item":{"Husband":"Homer","EmailID":"Marge Simpson","Children":["Lisa Simpson","Bart Simpson","Maggie Simpson"]}}
```

4.2

First API



/getcustomerdetailsbyemail - GET - Method Execution




Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

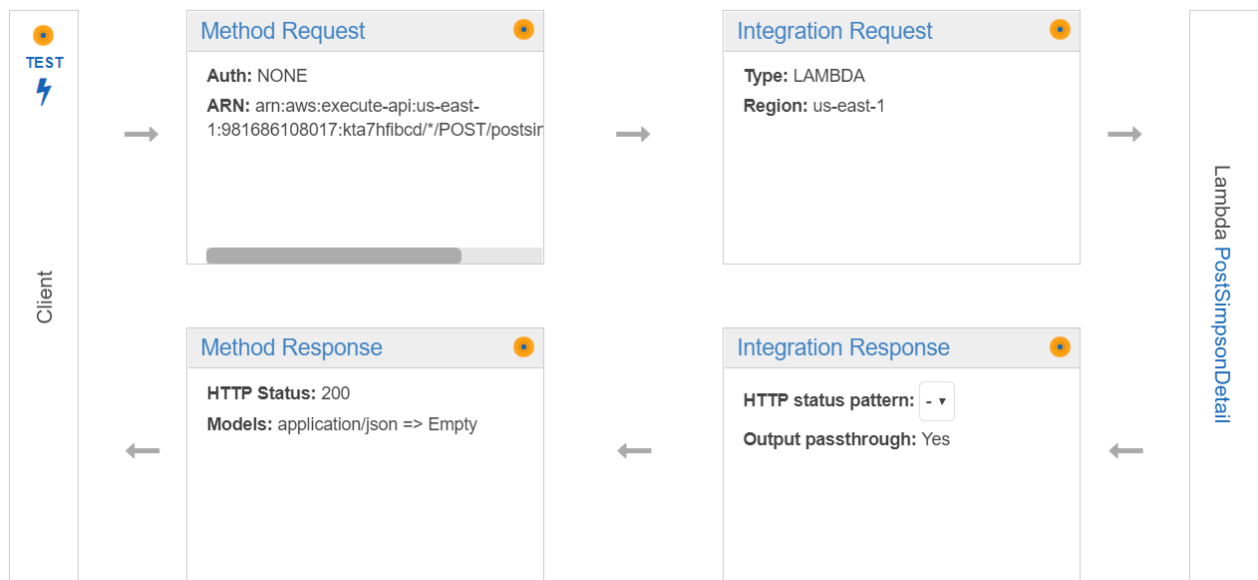
Lambda Region us-east-1 

Lambda Function getCustomerDetailsByEmail 

Execution role 

Second API

/postsimpsonsdetails - POST - Method Execution 




← **Method Execution** /postsimpsonsdetails - POST - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
☐ VPC Link ⓘ

Use Lambda Proxy integration ☐ ⓘ

Lambda Region us-east-1 

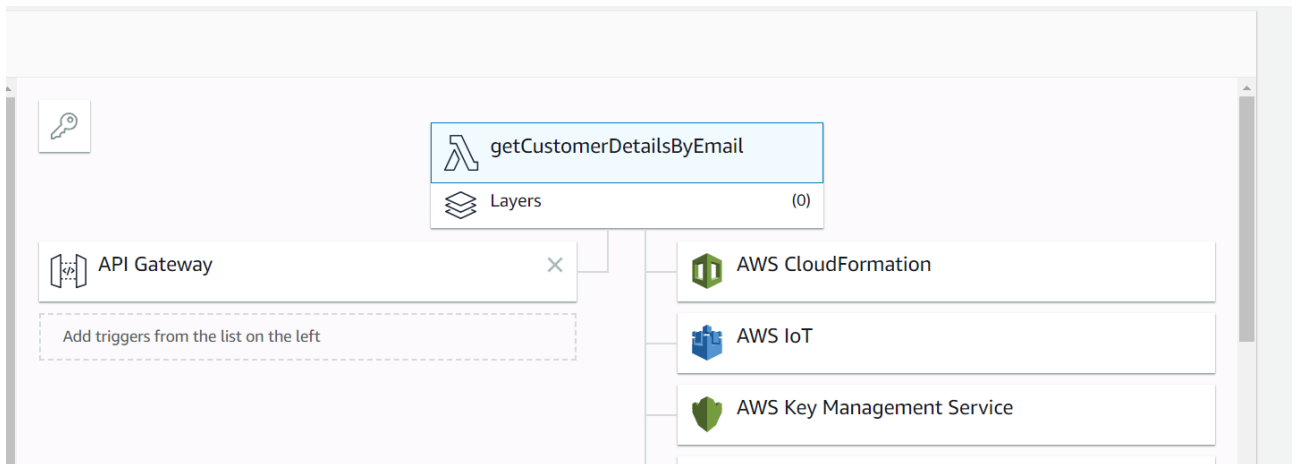
Lambda Function PostSimpsonDetail 

Execution role 

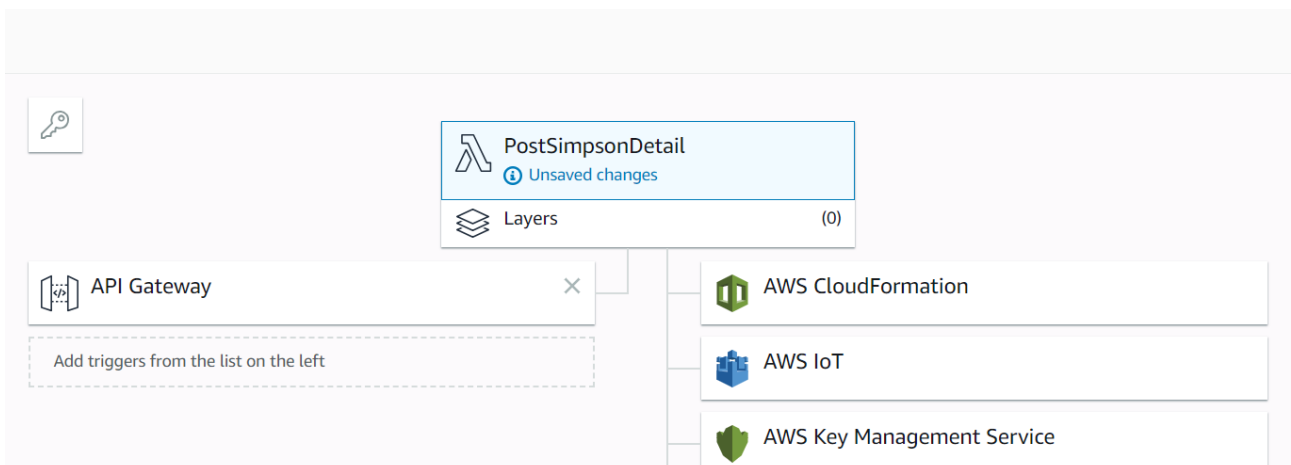
4.3

Lambda Functions

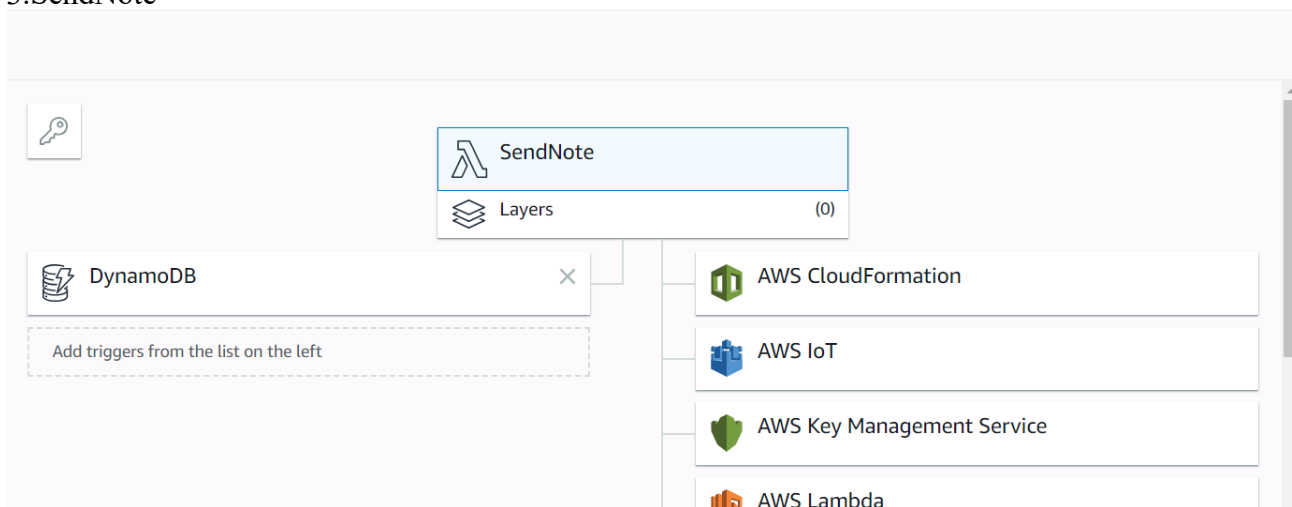
1.getCustomerDetailsByEmail



2.PostSimpsonDetail



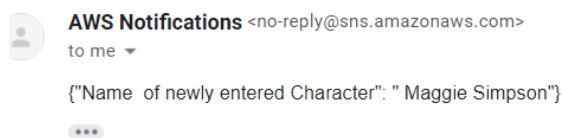
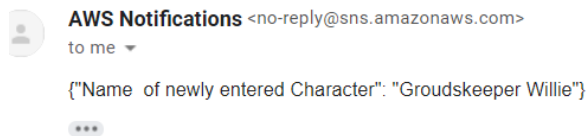
3.SendNote



3.4 SNS

<input type="checkbox"/>	DataQuery	arn:aws:sns:us-east-1:981686108017:DataQuery
<input type="checkbox"/>	dynamodb	arn:aws:sns:us-east-1:981686108017:dynamodb

SNS Message send to the subscribed email



3.5 S3 Bucket

Hosted Webpage files

Amazon S3 > simpsonsdatabase

Overview Properties Permissions **Public** Management

Q Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions

US East (N. Virginia)

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	466825.jpg	Dec 5, 2018 12:14:36 AM GMT+0000	60.8 KB	Standard
<input type="checkbox"/>	AMAZON_LOGO.png	Dec 5, 2018 12:14:37 AM GMT+0000	3.1 KB	Standard
<input type="checkbox"/>	Stylesheet.css	Dec 5, 2018 12:14:38 AM GMT+0000	1.7 KB	Standard
<input type="checkbox"/>	index.html	Dec 5, 2018 12:14:37 AM GMT+0000	1.0 KB	Standard
<input type="checkbox"/>	newjavascript.js	Dec 5, 2018 12:14:37 AM GMT+0000	188.0 B	Standard

Static website hosting

Endpoint : <http://simpsonsdatabase.s3-website-us-east-1.amazonaws.com>

☒ Use this bucket to host a website [Learn more](#)

Index document [i](#)

Error document [i](#)

Redirection rules (optional) [i](#)

☐ Redirect requests [Learn more](#)

☐ Disable website hosting

Cancel Save

4.6 Overall Working

1. Enters the character name



The Simpson's D

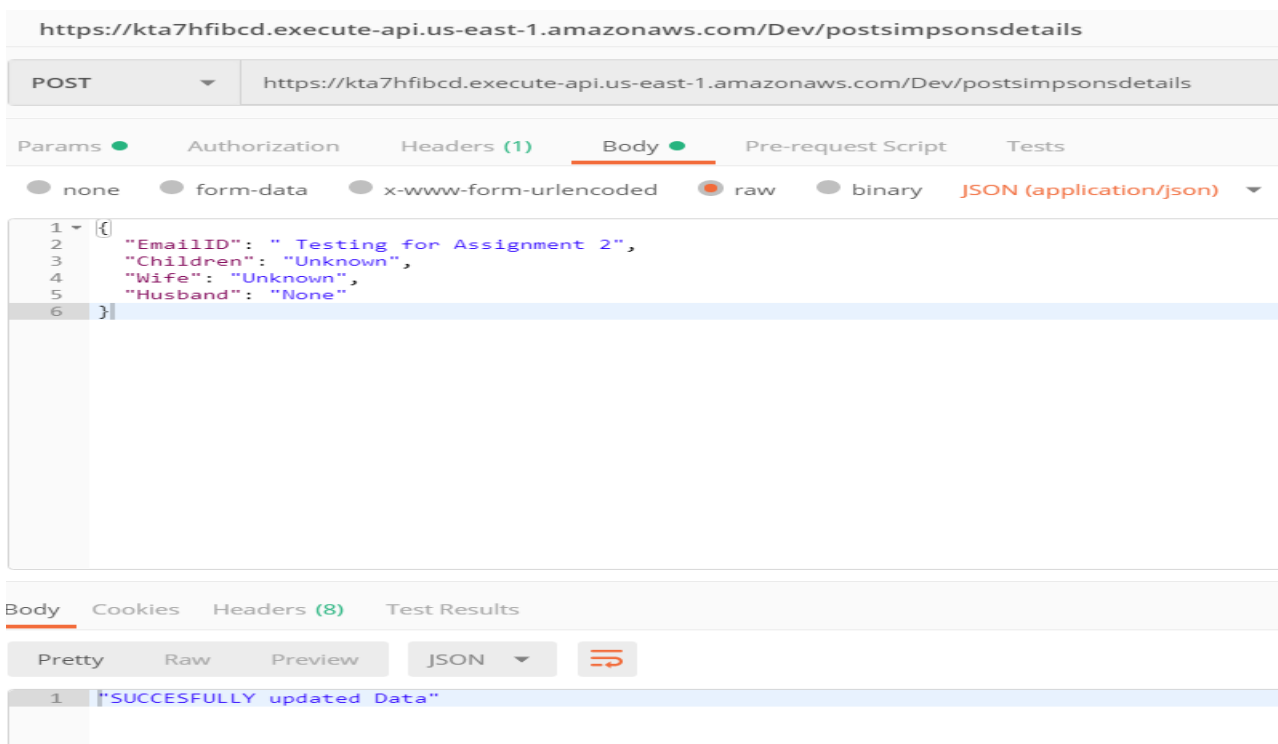
Enter the ID here:

Marge Simpson

2. This is the JSON response for Marge Simpsons

```
{"Item":{"Husband":"Homer","EmailID":"Marge Simpson","Children":"Lisa Simpson,Bart Simpson,Maggie Simpson"}}
```

3. Now administrator wants to add new character to the database



https://kta7hfibcd.execute-api.us-east-1.amazonaws.com/Dev/postsimpsonsdetails

POST https://kta7hfibcd.execute-api.us-east-1.amazonaws.com/Dev/postsimpsonsdetails

Params Authorization Headers (1) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

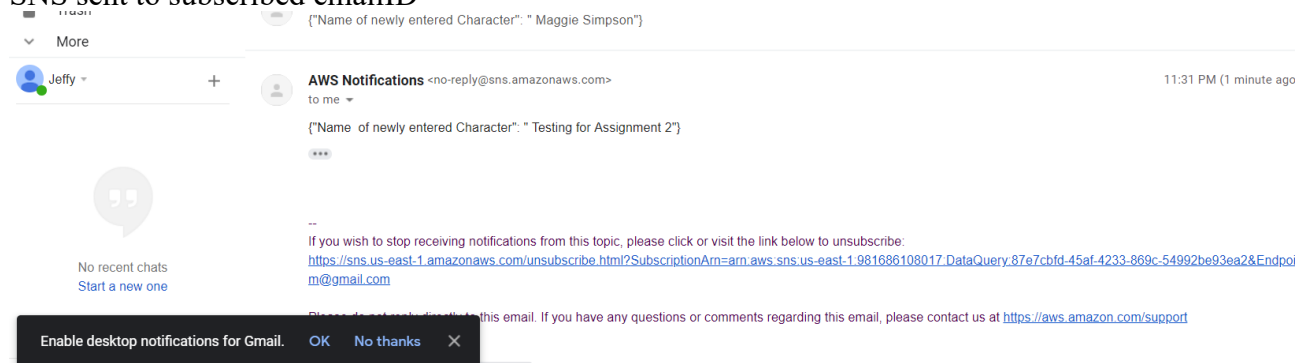
```
1 {  
2   "EmailID": " Testing for Assignment 2",  
3   "Children": "Unknown",  
4   "Wife": "Unknown",  
5   "Husband": "None"  
6 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview JSON

```
1 "SUCCESFULLY updated Data"
```

SNS sent to subscribed emailID



Try it out with the link below

<http://simpsonsdatabase.s3-website-us-east-1.amazonaws.com/>

References

1. <https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/>
2. <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/dynamodb-example-table-read-write.html>
3. Quicklabs
4. https://www.youtube.com/watch?v=G_-aEXmluq8&t=218s
5. <https://linuxacademy.com/>