# **API Gateway HTTP APIs**

Hello! folks its awesome to have you here working towards understanding how HTTP APIs are created using API gateway. Before we get started its always important to know the difference between REST API and HTTP API and where must they be used. If you are already familiar with this, please skip directly to Section 1.

**Section 0**: **Difference between** [**REST API & HTTP API**](https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest.html)

## **Section 1: Tutorial -> Build a CRUD API with Lambda and DynamoDB**

**Objective**: Create a serverless API that creates, reads, updates, and deletes items from a DynamoDB table.

Graphical user interface, diagram

Description automatically generated with medium confidence

Inception of this project would be creation of a [DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html) table, which can be used to store data for our API, So let’s get started with creation of the table, please click [ [here](https://console.aws.amazon.com/dynamodb/) ].

Create your table by giving a desired name and Partition key which would act as the table’s primary key, Sort key is optional, and it determines the order of how data with the same partition key is stored.

**Note [** Partition and Sort keys can’t be changed after the table is provisioned.**]**

DynamoDB Table Creation


Once you press create table, you’d have a similar screen like the below picture

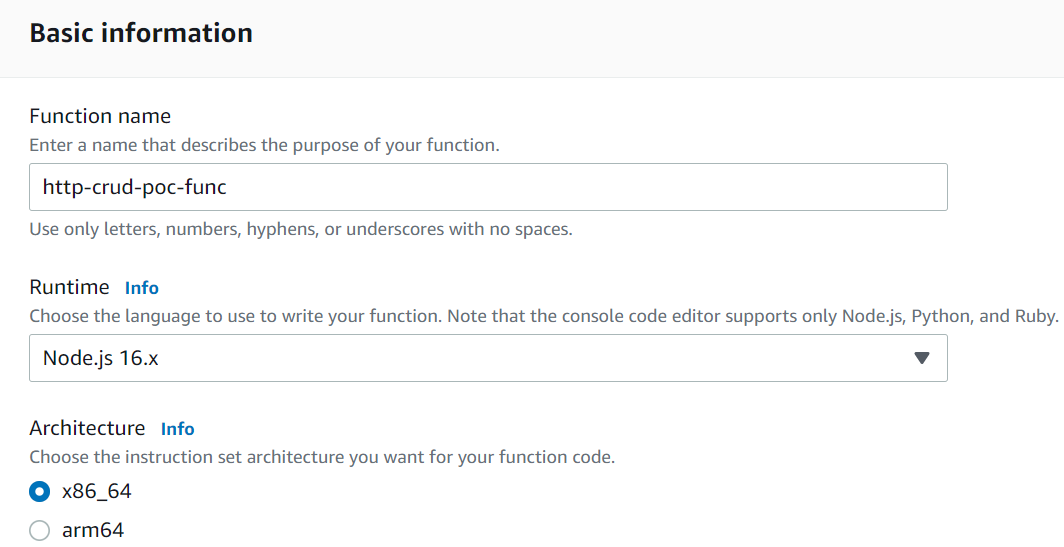
Graphical user interface, text, application

Description automatically generated

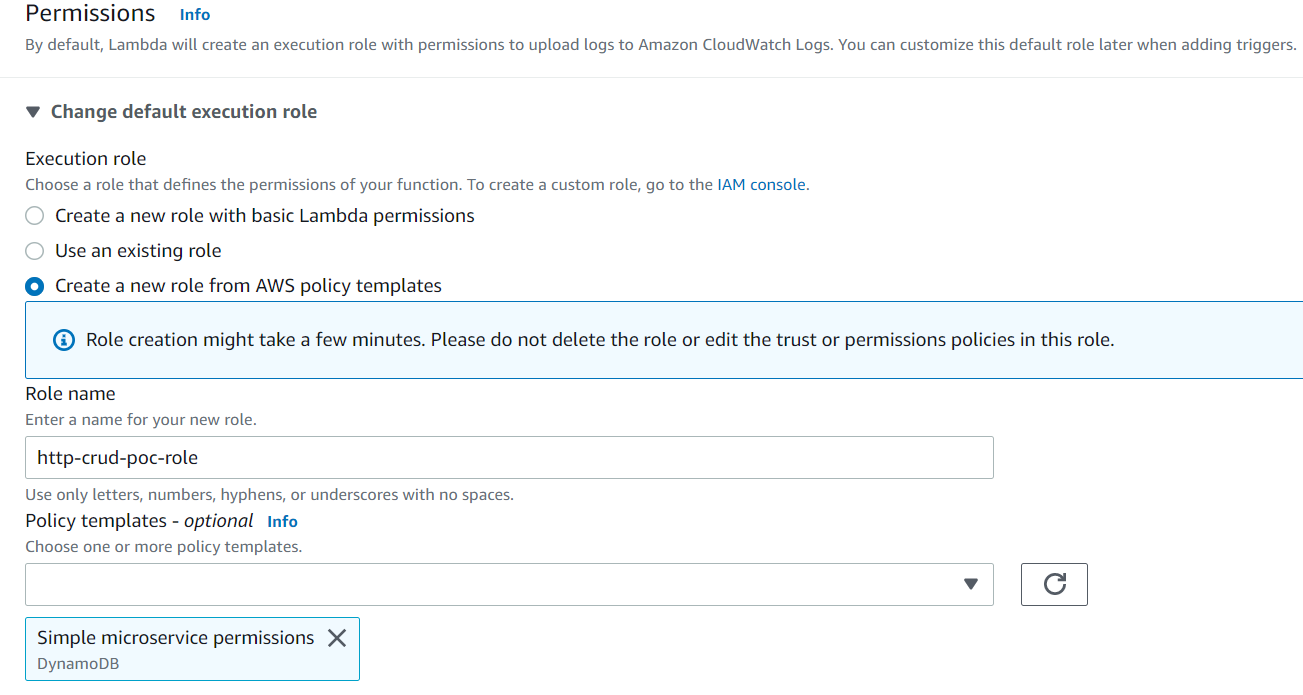
This marks the end of DynamoDB Table creation, The next step would be to create a **Lambda Function**.

Creation of Lambda function is for a backend for the API. All the CRUD operations on the table can be performed via this function. Please click [ [here](https://console.aws.amazon.com/lambda) ] to create a Lambda function.

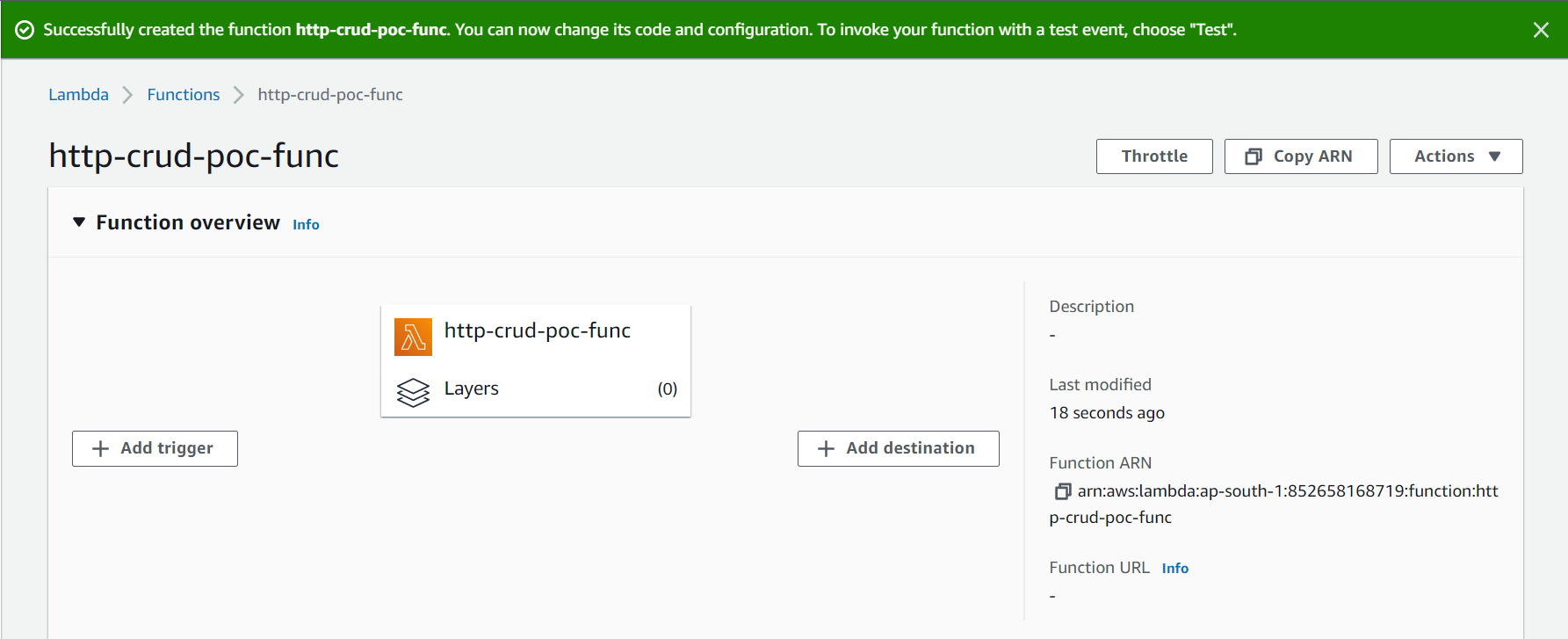
You’ll have a similar screen



The next configuration is Permissions where you **choose default execution role** and under it select **Create a new role from AWS policy templates.** In Simple terms a Lambda function’s **execution role** (an AWS Identity and Access Management (IAM) role that grants the function permission to access AWS services and resources)



From the above screen you understand that we have selected one of the managed permissions i.e Simple microservice permissions in AWS api gateway to know more about **execution role** click [ [here](https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html#permissions-executionrole-console)]. After all the configurations are done you can **create function,** which then would be followed up by a similar screen such as



Go to **index.js** in Code Source and paste this code

**const AWS = require("aws-sdk");**

**const dynamo = new AWS.DynamoDB.DocumentClient();**

**exports.handler = async (event, context) => {**

**let body;**

**let statusCode = 200;**

**const headers = {**

**"Content-Type": "application/json"**

**};**

**try {**

**switch (event.routeKey) {**

**case "DELETE /items/{id}":**

**await dynamo**

**.delete({**

**TableName: "http-api-poc",**

**Key: {**

**id: event.pathParameters.id**

**}**

**})**

**.promise();**

**body = `Deleted item ${event.pathParameters.id}`;**

**break;**

**case "GET /items/{id}":**

**body = await dynamo**

**.get({**

**TableName: "http-api-poc",**

**Key: {**

**id: event.pathParameters.id**

**}**

**})**

**.promise();**

**break;**

**case "GET /items":**

**body = await dynamo.scan({ TableName: "http-api-poc" }).promise();**

**break;**

**case "PUT /items":**

**let requestJSON = JSON.parse(event.body);**

**await dynamo**

**.put({**

**TableName: "http-api-poc",**

**Item: {**

**id: requestJSON.id,**

**price: requestJSON.price,**

**name: requestJSON.name**

**}**

**})**

**.promise();**

**body = `Put item ${requestJSON.id}`;**

**break;**

**default:**

**throw new Error(`Unsupported route: "${event.routeKey}"`);**

**}**

**} catch (err) {**

**statusCode = 400;**

**body = err.message;**

**} finally {**

**body = JSON.stringify(body);**

**}**

**return {**

**statusCode,**

**body,**

**headers**

**};**

**};**

Then **Deploy** to update your function.

Now the next step involves creation of **HTTP API,** to create an HTTP API click [ [here](https://console.aws.amazon.com/apigateway) ], choose **Create API** and then choose HTTP API **Build,** For now don’t create any routes or integrations, lets create an empty Http API.

You’d have a similar screen

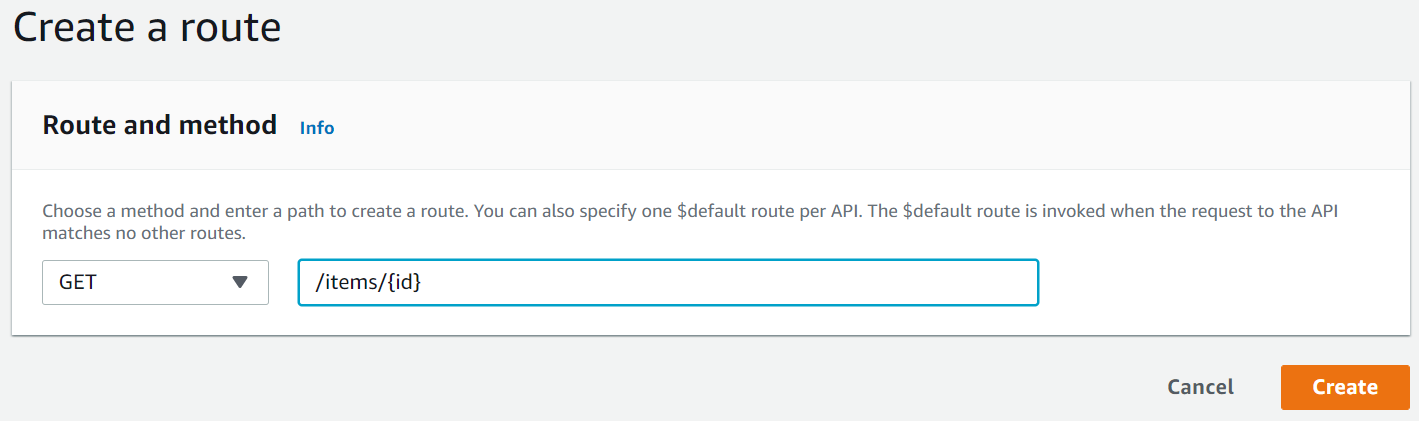
Graphical user interface, text, application

Description automatically generated

Now let’s move on to **creating** **routes** which are a way to send incoming requests to backend source, it comprises of two parts **method/ resource path,** For our tutorial we’ll have

* GET /items/{id}
* GET /items
* PUT /items
* DELETE /items/{id}

To start creating routes click [ [here](https://console.aws.amazon.com/apigateway) ], Select your API, then select routes and create, you’d have a similar screen



After creating repeat, the process for your other routes, after completion you’d have a similar screen

Graphical user interface, application

Description automatically generated

Next step Involves creating an **Integration,** which bridges a route to a backend resource. To create one click [ [here](https://console.aws.amazon.com/apigateway) ]. Select your API and create integration follow it up by choosing **Manage integrations** and create. Skip the step to attach the integration to a route for now.

Choose Lambda function as your Integration target, and in details select the AWS Region that you have created your Lambda function and select your Lambda function.

Graphical user interface, text, application, email

Description automatically generated

Choose **Create**, you’d have a similar screen

Graphical user interface, application

Description automatically generated

Now next step would include **attaching the integration to the routes**. Go under Attach integrations to routes and select a route, now follow it up by choosing an existing integration in by doing so your integration would get populated, Last is to **Attach Integration**. Once its done you’d have a similar screen

Graphical user interface, text, application

Description automatically generated

Repeat the process for all the routes. At completion you’d have a similar screen

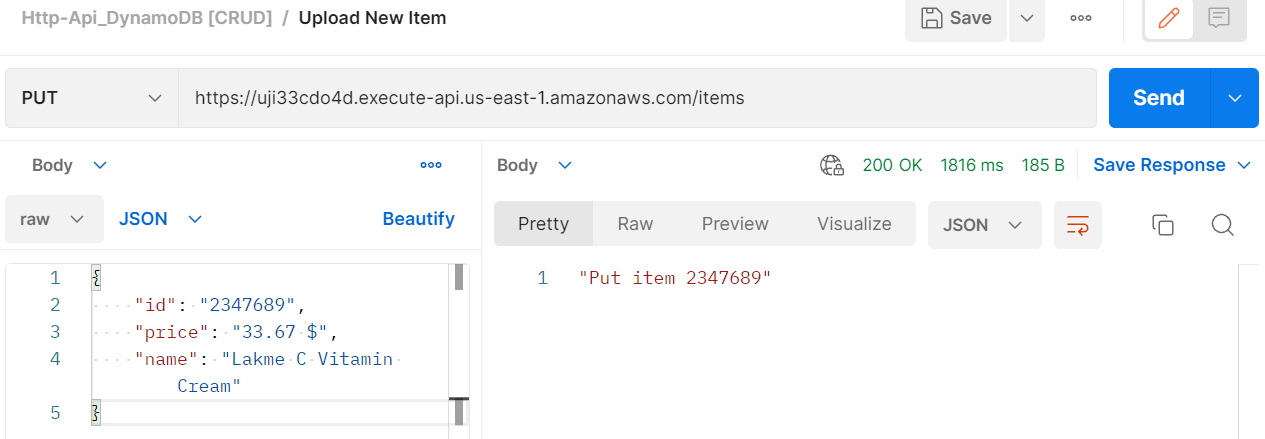
Graphical user interface, application, email

Description automatically generated

Now your API is ready to be tested, to do so click [ [here](https://console.aws.amazon.com/apigateway) ]. Choose your API and then take the invoke URL and use **Postman**, I’d be sharing the postman collections to you. You ‘ll have a similar screen once you invoke the following requests.

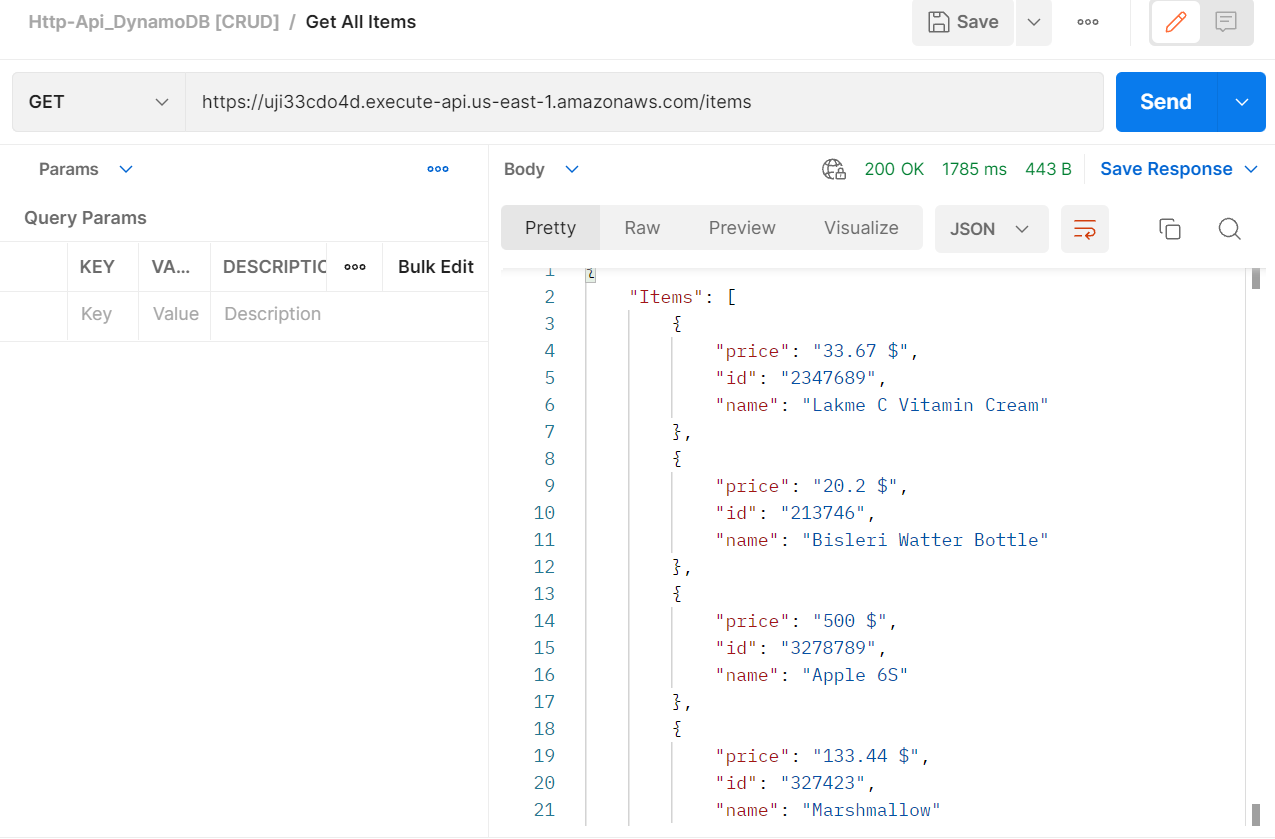
Initially as there are no items try updating the list of items by inserting few records. Make sure you use your invoke URL

1. **Put Items**



-----------------------------------------------------------------------------------------------------------------

1. **Get All Items**



-------------------------------------------------------------------------------------------------------------

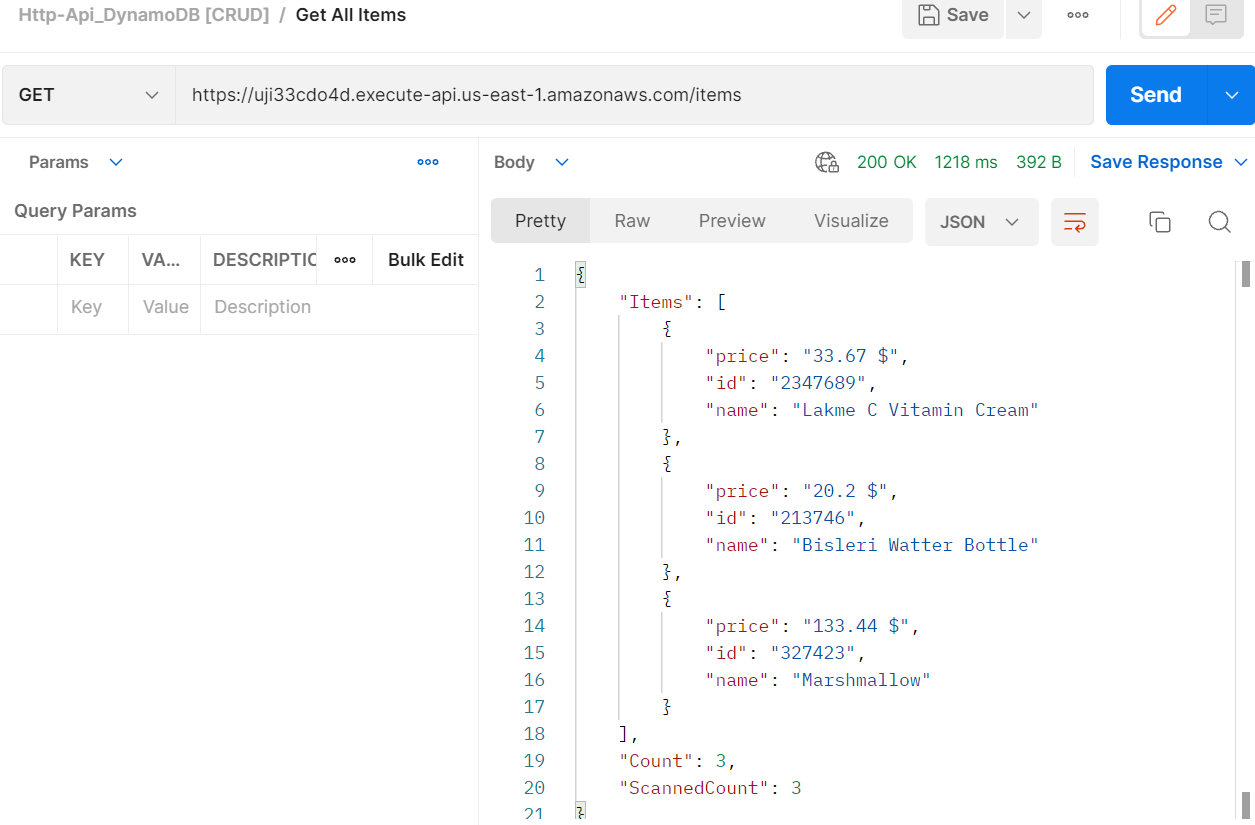
1. **Delete An Item**

Graphical user interface, text, application

Description automatically generated

-----------------------------------------------------------------------------------------------------------------

* 1. Now **Get** Items List, to see if the item you deleted has been removed from the list



------------------------------------------------------------------------------------------------------------

As you can see the Count has changed from 4 to 3 and the designated item with id **3278789** has been removed.

As you have completed a major part of the tutorial, congratulations for coming so far patiently! Just one more thing to wrap up this tutorial, which Is very important too, so to prevent unnecessary costs, its best that you delete the resources that you have developed

1. To delete **DynamoDB** **Table** click [ [here](https://console.aws.amazon.com/dynamodb/) ] -> Choose your table and **DELETE** it.
2. To delete **HTTP API**, click [ [here](https://console.aws.amazon.com/apigateway) ] -> Select API -> Choose Actions -> **DELETE**
3. To delete **Lambda function,** click [ [here](https://console.aws.amazon.com/lambda) ] -> Functions Page -> Select Function -> Choose Actions -> Delete.
4. To delete Lambda Function’s Log Group -> Search CloudWatch -> Select Log group -> Actions -> **DELETE**.

Graphical user interface, application

Description automatically generated

-------------------------------------------------------------------------------------------------------

1. To delete Lambda Function’s Execution Role -> Search IAM in search bar -> Roles Page-> Select the function’s role for our example [ [http-crud-poc-role](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/roles/details/http-crud-poc-role) ] -> **DELETE .**

Graphical user interface, application

Description automatically generated

----------------------------------------------------------------------------------------------------------------

I hope this tutorial has helped you to understand the basics of implementing HTTP API, Lambda function, DynamoDB. Keep up the great work and **happy learning**!!