**CRUD API**

**Overview**

In this demo, we will create a simple CRUD(Create, read, update and delete) API.Initially, you will need to create a DynamoDB table using the console. Next, you will have to configure the HTTP API using the API Gateway console and lastly, we can use AWS Cloud9 IDE for testing the APIs.



When you invoke your HTTP API, API Gateway routes the request to your Lambda function. The Lambda function interacts with DynamoDB and returns response to the API Gateway. The API Gateway will return the respective response.

**Pre-requisites**

1. DynamoDB
2. Lambda Function
3. HTTP API
4. Routes
5. Integration
6. Cloud9

**Procedure**

1. **Create a DynamoDB table**

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB will allow you to create database tables that can store and retrieve any amount of data and serve any level of request traffic.

To create DynamoDB table:

1. Open the DynamoDB console at <https://console.aws.amazon.com/dynamodb/>
2. Choose create table
3. For table name, enter http-crud-tutorial-items
4. For primary key, enter id
5. Choose Create
6. After few seconds the DynamoDB table becomes available.
7. **Create a lambda function**

Compute service lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

1. Sign in to the Lambda console at <https://console.aws.amazon.com/lambda>
2. Choose Create function
3. Select Author from scratch
4. For function name, enter http-crud-tutorial-function
5. Under Runtime info select Node.js 14.x
6. Under Permissions choose Change default execution role
7. Select Create a new role from AWS policy templates
8. For Role name, enter http-crud-tutorial-role
9. For Policy templates, choose Simple microservice permissions. This policy grants the Lambda function permission to interact with DynamoDB
10. Choose Create function
11. Scroll down to the console’s code source editor
12. Open index.js and replace its contents with the following 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-crud-tutorial-items",

Key: {

id: event.pathParameters.id

}

})

.promise();

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

break;

case "GET /items/{id}":

body = await dynamo

.get({

TableName: "http-crud-tutorial-items",

Key: {

id: event.pathParameters.id

}

})

.promise();

break;

case "GET /items":

body = await dynamo.scan({ TableName: "http-crud-tutorial-items" }).promise();

break;

case "PUT /items":

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

await dynamo

.put({

TableName: "http-crud-tutorial-items",

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

};

};

1. Choose Deploy to update your function
2. **Create an HTTP API**

With API Gateway, we can create RESTful APIs using either HTTP APIs or REST APIs. Together with AWS Lambda, API Gateway forms the app-facing part of the AWS serverless infra.

1. Sign in to the API Gateway console at

<https://console.aws.amazon.com/apigateway>

1. Choose Create API
2. For HTTP API choose Build
3. For API name, enter http-crud-tutorial-api
4. Choose Next
5. For configure routes, choose Next to skip route creation, You create routes later
6. Review the stage that API Gatway creates for you ($default), and then choose Next
7. Choose Create
8. **Create routes**

Routes are a way to send incoming API requests to backend resources. Routes consist of two parts: an HTTP method and a resource path, for example, GET /items.For this example API, we create four routes:

GET/itens/{id}

GET/items

PUT/items

DELETE/items/{id}

1. Sign in to the API Gateway console at

<https://console.aws.amazon.com/apigateway>

1. Choose your API (http-crud-tutorial-api)
2. On the left panel choose Routes
3. Choose Create
4. For Method, choose GET
5. For the path, enter /items/{id}. The {id} at the end of the path is a path parameter that API Gateway retrieves from the request path when a client makes a request
6. Choose Create
7. Repeat steps 4-7 for GET /items, DELETE /items/{id} and PUT/items
8. Confirm all routes are created
9. **Create an integration**

You create an integration to connect a route to backend resources. At a later step you will attach these integrations to a route. For this example API, you create one Lambda integration that you use for all routes.

1. Sign in to the API Gateway console at

<https://console.aws.amazon.com/apigateway>

1. Choose your API (http-crud-tutorial-api)
2. Choose Integrations
3. Choose Manage integrations and then choose Create
4. Skip Attach this integration to a route. You complete that in a later step
5. For integration type, choose Lambda function
6. For Lambda function, enter http-crud-tutorial-function
7. Choose Create
8. **Attaching integration to the routes**

After you attach the integration to all of the API’s routes, your Lambda function is invoked when a client calls any of your routes.

1. Sign in to the API Gateway console at

<https://console.aws.amazon.com/apigateway>

1. Choose your API (http-crud-tutorial-api)
2. Choose Integrations
3. Choose a route
4. Under Choose an existing integration, choose http-crud-tutorial-function
5. Choose Attach integration
6. Repeat steps 4-6 for all routes. All routes show that an AWS Lambda integration is attached. Now that you have an HTTP API with routes and integrations, you can test your API.
7. **Accessing the Cloud9**

Cloud9 is an integrated development environment or IDE. The AWS Cloud9 IDE offers a rich code-editing experience with support for several programming languages and runtime debuggers and a built-in terminal. It contains a collection of tools that you use to code, build, run, test and debug software and mainly helps you release software to the cloud.

Perks of Using Cloud9 IDE

1. Store your project’s files locally on the instance or server
2. Clone a remote code repository such as a repo in AWS CodeCommit- into your environment
3. Work with a combination of local and cloned files in the env

Steps:

1. Connect to Cloud9 by browsing to

<https://console.aws.amazon.com/cloud9/>

1. In console, make sure you are in right region and search for the button “Create environment”
2. Name instance: api-workshop
3. Configure the options of your Cloud9 instance, Environment type should be Create a new EC2 instance for environment(direct access). Instance type, t2.micro (1 GB RAM + 1 vCPU), Platform: Amazon Linux 2 (recommended), Cost-saving setting should be After four horse, IAM role should be AWSServiceRoleForAWSCloud9
4. Review the options and choose Create! After a few minutes your cloud 9 environment is created.
5. Close the Welcome Screen
6. Prepare a terminal in full screen
7. Your terminal in Cloud9 is ready for your API testing!
8. You can proceed to test your API
9. **Testing API**
10. To get the URL to invoke your API

1. Sign in to the API Gateway console at

<https://console.aws.amazon.com/apigateway>

2. Choose your API

3. Note your API’s invoke URL. It appears under invoke URL on the Details page

4. Copy your API’s invoke URL.

1. To create or update an item
2. Connect to Cloud9 by browsing to

https://console.aws.amazon.com/cloud9/

1. Open the Cloud9 IDE
2. In the following command replace "https://abcdef123.execute-api.eu-west-1.amazonaws.com" with your Invoke URL from the previous step to set a variable with the Invoke URL

#Replace URL with the Invoke URL above

export INVOKE\_URL="[https://\*\*abcdef123\*\*.execute-api.eu-west-1.amazonaws.com](about:blank)"

1. Create or update an item. The command includes a request body with the item’s ID, price and name.

curl -X "PUT" -H "Content-Type: application/json" -d "{

\"id\": \"abcdef234\",

\"price\": 12345,

\"name\": \"myitem\"

}" $INVOKE\_URL/items

1. Use the foll command to list all items

curl -s $INVOKE\_URL/items | js-beautify

1. Use the foll command to get an item by its ID

curl -s $INVOKE\_URL/items/abcdef234 | js-beautify

1. Use the foll command to delete an item

curl -X "DELETE" $INVOKE\_URL/items/abcdef234

1. Get all items to verify that the item was deleted

curl -s $INVOKE\_URL/items | js-beautify

1. Delete

## **[To delete an DynamoDB table:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-an-dynamodb-table:)**

1. Open the DynamoDB console at <https://console.aws.amazon.com/dynamodb/>
2. Select your table.
3. Choose Delete table.
4. Confirm your choice, and choose Delete.

## **[To delete an HTTP API:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-an-http-api:)**

1. Sign in to the API Gateway console at <https://console.aws.amazon.com/apigateway>
2. On the APIs page, select an API. Choose Actions, and then choose Delete.
3. Choose Delete.

## **[To delete a Lambda function:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-a-lambda-function:)**

1. Sign in to the Lambda console at <https://console.aws.amazon.com/lambda/>
2. On the Functions page, select a function. Choose Actions, and then choose Delete.
3. Choose Delete.

## **[To delete a Lambda function's log group:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-a-lambda-function's-log-group:)**

1. In the Amazon CloudWatch console, open the [Log groups page](https://console.aws.amazon.com/cloudwatch/home#logs:)
2. On the Log groups page, select the function's log group (/aws/lambda/http-crud-tutorial-function). Choose Actions, and then choose Delete log group.
3. Choose Delete.

## **[To delete a Lambda function's execution role:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-a-lambda-function's-execution-role:)**

1. In the AWS Identity and Access Management console, open the [Roles page](https://console.aws.amazon.com/iam/home?#/roles)
2. Select the function's role, for example, http-crud-tutorial-role.
3. Choose Delete role.
4. Choose Yes, delete.

## **[To delete the Cloud9 IDE:](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up#to-delete-the-cloud9-ide:)**

1. Sign in to the Cloud9 console at <https://console.aws.amazon.com/cloud9/>
2. In the Your environments page, an IDE. Choose Delete.
3. To confirm your choice type Delete.
4. Choose Delete.

**Output:**

**Demo URL:**

RUN the app in Cloud9

**Links Demo**

[**https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up**](https://catalog.us-east-1.prod.workshops.aws/workshops/2c8321cb-812c-45a9-927d-206eea3a500f/en-US/090-clean-up)

**Commands to Run**

**Invoke URL from API Gateway -**

[**https://s4s23hs8n7.execute-api.us-east-2.amazonaws.com/**](https://s4s23hs8n7.execute-api.us-east-2.amazonaws.com/)