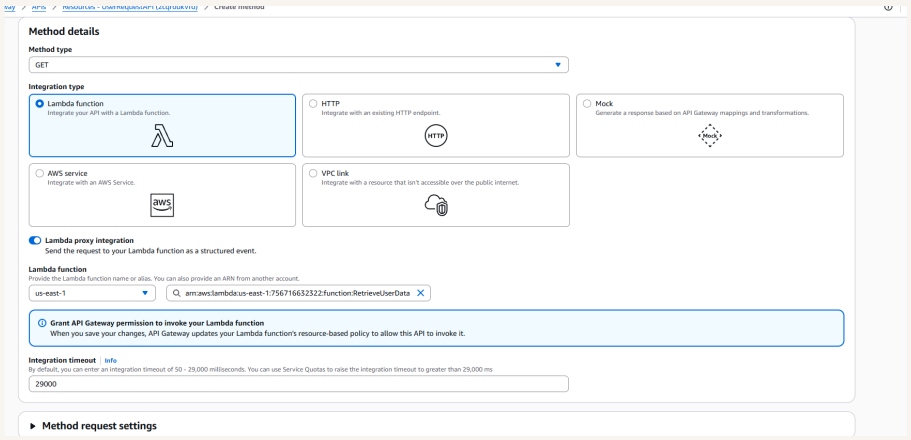
nextwork.org

APIs with Lambda + API Gateway

Louis Moyo



L
Louis Moyo
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to build a serverless API using AWS Lambda and API Gateway. I'm doing this project to learn how to create a backend without managing traditional servers, understand how serverless functions interact with an API gateway, and practice documenting APIs in JSON. This will help me build the logic tier of a three-tier architecture and strengthen my skills in cloud development.

Tools and concepts

Services I used were AWS Lambda and Amazon API Gateway. Key concepts I learnt include Lambda functions for running code without servers, API Gateway for managing requests and connecting them to Lambda, resources and methods for structuring an API, stages for deployment, and documentation for making APIs easier to use.

Project reflection

This project took me approximately 45 minutes. The most challenging part was understanding how Lambda proxy integration works, and it was most rewarding to see my API deployed and accessible through a public endpoint.

L

Louis Moyo
NextWork Student

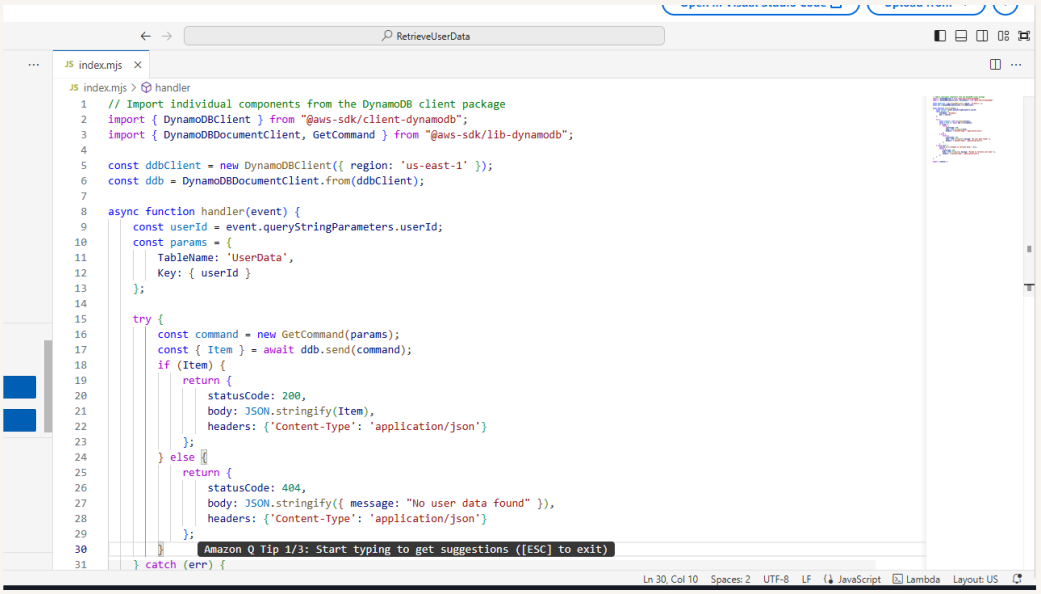
nextwork.org

I did this project today to gain hands-on experience with serverless APIs and to understand how Lambda and API Gateway work together in the logic tier of a three-tier architecture. Yes, this project met my goals by giving me practical skills that I can apply to real-world cloud development.

Lambda functions

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. It automatically handles scaling, availability, and execution so you only pay for the time your code runs. I'm using Lambda in this project to build a backend function that retrieves and returns user data as part of the logic layer in a three-tier architecture.

The code I added to my function will query a DynamoDB table using a given userId and return that user's data if it exists. If the userId is not found or another error occurs, the function will return an error message instead. This makes the Lambda act like a backend API that fetches specific user records from the database.

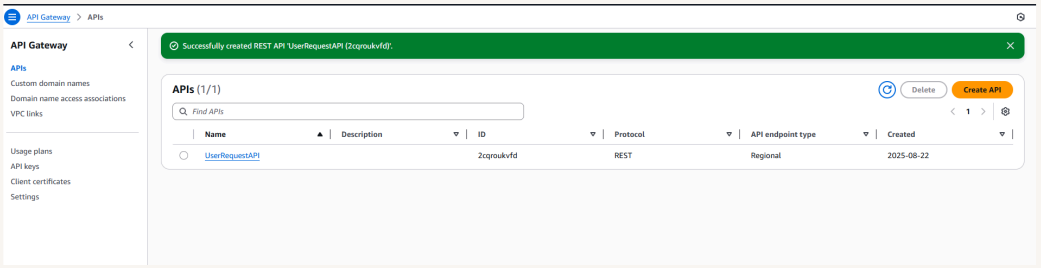

RetrieveUserData
index.mjs
index.mjs > handler
// Import individual components from the DynamoDB client package
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";
const ddbClient = new DynamoDBClient({ region: 'us-east-1' });
const ddb = DynamoDBDocumentClient.from(ddbClient);
async function handler(event) {
const userId = event.queryStringParameters.userId;
const params = {
TableName: 'UserData',
Key: { userId }
};
try {
const command = new GetCommand(params);
const { Item } = await ddb.send(command);
if (Item) {
return {
statusCode: 200,
body: JSON.stringify(Item),
headers: {'Content-Type': 'application/json'}
};
} else {
return {
statusCode: 404,
body: JSON.stringify({ message: "No user data found" }),
headers: {'Content-Type': 'application/json'}
};
}
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
} catch (err) {
Ln 30, Col 10
Spaces: 2
UTF-8
LF
JavaScript
Lambda
Layout: US

API Gateway

APIs are interfaces that let different software systems communicate with each other by sending requests and receiving responses. There are different types of APIs, like REST APIs, GraphQL APIs, and WebSocket APIs, each suited for different use cases. My API is a REST API that connects a user's request from the browser to my Lambda function so it can process the request and return data.

Amazon API Gateway is a fully managed service that allows developers to create, publish, and secure APIs at scale. I'm using API Gateway in this project to act as the entry point for user requests, manage authentication, and route those requests to my Lambda function.

When a user makes a request, API Gateway receives it and securely forwards it to the Lambda function. Lambda then processes the request, retrieves or processes the data, and sends the result back through API Gateway, which delivers the response to the user's browser or application.

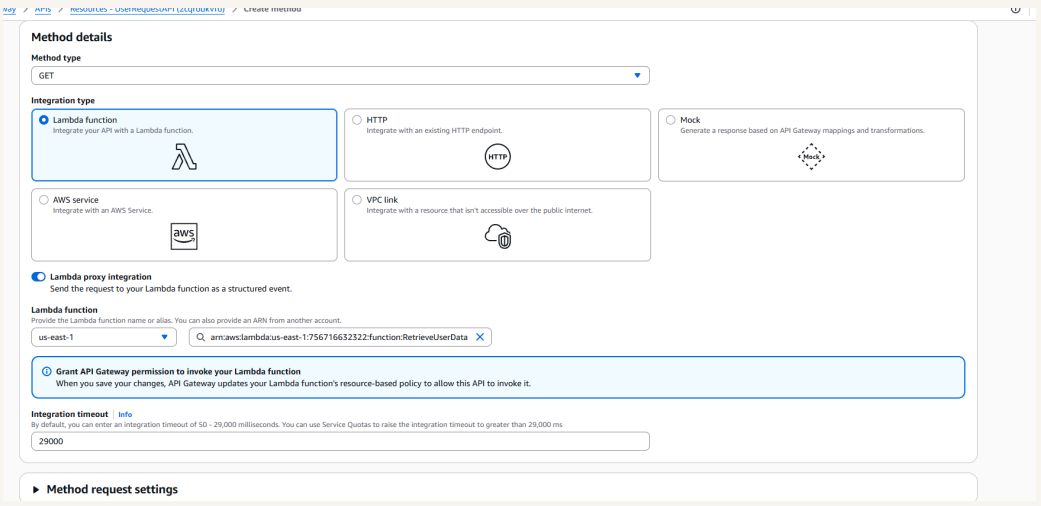

API Gateway > APIs
API Gateway
APIs
Custom domain names
Domain name access associations
VPC links
Usage plans
API keys
Client certificates
Settings
Successfully created REST API 'UserRequestAPI (2cqroukvfd)'.
APIs (1/1)
Find APIs
Delete
Create API
Name
Description
ID
Protocol
API endpoint type
Created
UserRequestAPI
2cqroukvfd
REST
Regional
2025-08-22

API Resources and Methods

An API is made up of resources, which are individual endpoints that represent specific parts of the API's functionality. Each resource has its own path (like /users or /messages) and can support different request methods such as GET, POST, PUT, or DELETE

Each resource consists of methods, which are the actions you can perform on that resource, such as GET to read data, POST to create data, PUT to update data, and DELETE to remove data.

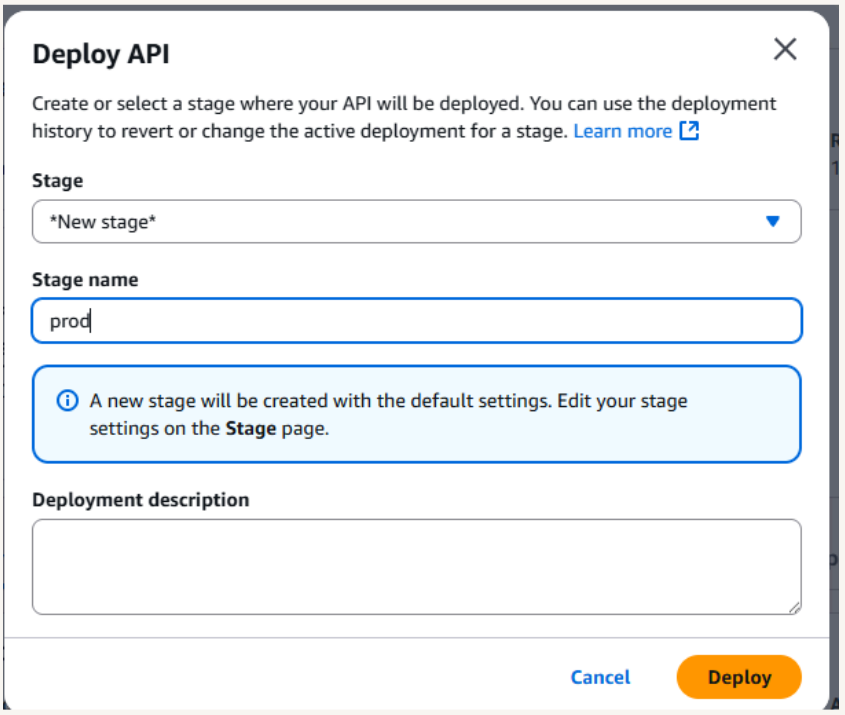
I created a GET method for the /users resource, which is integrated with my Lambda function. This means that whenever someone calls this endpoint, API Gateway will trigger the Lambda function to retrieve user data.


Method details
Method type
GET
Integration type
Lambda function
Integrate your API with a Lambda function.
HTTP
Integrate with an existing HTTP endpoint.
Mock
Generate a response based on API Gateway mappings and transformations.
AWS service
Integrate with an AWS Service.
VPC link
Integrate with a resource that isn't accessible over the public internet.
Lambda proxy integration
Send the request to your Lambda function as a structured event.
Lambda function
Provide the Lambda function name or alias. You can also provide an ARN from another account.
us-east-1
arn:aws:lambda:us-east-1:756716632322:function:RetrieveUserData
Grant API Gateway permission to invoke your Lambda function
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.
Integration timeout
Info
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms
29000
Method request settings

API Deployment

When you deploy an API, you deploy it to a specific stage. A stage is a versioned environment of your API, such as dev, test, or prod, that controls how and where the API is accessed. I deployed to a new stage so my API can be publicly available for testing.

To visit my API, I copied the Invoke URL from the prod stage page and opened it in a new browser tab. The API displayed an error because the DynamoDB table hasn't been created yet, so there's no data source for the Lambda function to retrieve.



API Documentation

For my project's extension, I am writing API documentation because it explains how the API works, what endpoints exist, and what responses users can expect. You can do this in API Gateway so developers can easily understand and integrate with the API.

Once I prepared my documentation, I published it to the prod stage in API Gateway. You have to publish your API to a specific stage because documentation needs to match the deployed version of the API, ensuring that developers see the correct endpoints and details for that stage.

My published and downloaded documentation showed me both the custom JSON description I wrote, as well as automatically generated details from API Gateway such as the API's version, title, resources like /users, and the GET method. These were included to give a full picture of the API's structure and functionality.

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "2025-08-22T03:19:26Z",
    "title" : "UserRequestAPI"
  },
  "host" : "2cqroukvfd.execute-api.us-east-1.amazonaws.com",
  "basePath" : "/prod",
  "schemes" : [ "https" ],
  "paths" : {
    "/users" : {
      "get" : {
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
east-1:756716632322:function:RetrieveUserData/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000,
          "contentHandling" : "CONVERT_TO_TEXT",
          "type" : "aws_proxy"
        }
      }
    }
  },
  "definitions" : {
    "Empty" : {
      "type" : "object",
      "title" : "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation" : {
    "version" : "1",
    "createdDate" : "2025-08-22T03:29:26Z",
    "documentationParts" : [ {
      "location" : {
        "type" : "API"
```

The place to learn & showcase your skills

Check out nextwork.org for more projects