



Build CRUD Microservice with REST API and Lambda

Overview

In this tutorial, we'll create a system that uses **Amazon API Gateway** and **AWS Lambda** for synchronous microservice communication. A **REST API** will trigger a Lambda function, which processes the request and returns a response.

Key Steps

1. Create Lambda Function:

- Use the AWS Lambda Console to create a new function.
- Write logic for handling requests (e.g., retrieving or processing data).

2. Create REST API in API Gateway:

- Use the API Gateway Console to create a REST API.
- Define resources and HTTP methods (GET, POST) linked to the Lambda function.

3. Invoke API from Client:

- The client sends an HTTP request to the API Gateway.
- API Gateway triggers the Lambda function, which processes the request and sends a response back.

4. Handle Different Event Objects in Lambda:

- Lambda differentiates requests based on the **event object**, which includes details like query parameters, body data, and headers.
- For example:
 - **GET request:** Fetch data (e.g., product info).
 - **POST request:** Process data (e.g., create a new product).

Example Use Case: Product API

- **GET Method:** Fetch product info by ID.
- **POST Method:** Add a new product with details from the request body.

Conclusion

This lab demonstrates how to build scalable microservices using **AWS Lambda** and **API Gateway**, where the Lambda function processes requests based on different event objects from the client.

To begin with the Lab:

1. First, we are going to create a Lambda function for our environment. So, go to lambda and click on the create function. Here you need to choose the author from scratch option and give your function a name choose runtime as Node.js and create your function.

Create function Info

Choose one of the following options to create your function.

<input checked="" type="radio"/> Author from scratch Start with a simple Hello World example.	<input type="radio"/> Use a blueprint Build a Lambda application from sample code and configuration presets for common use cases.	<input type="radio"/> Container image Select a container image to deploy for your function.
--	--	--

Basic information

Function name

Enter a name that describes the purpose of your function.

ProductFunction

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Node.js 20.x



Architecture Info

Choose the instruction set architecture you want for your function code.

x86_64

arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

2. Once your lambda function is created you need to navigate to API gateway and choose to build a REST API.

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:

Lambda, HTTP, AWS Services

[Import](#) [Build](#)

3. Choose to create a new API, give it a name and click on create.

Create REST API

API details

New API
Create a new REST API.

Clone existing API
Create a copy of an API in this AWS account.

Import API
Import an API from an OpenAPI definition.

Example API
Learn about API Gateway with an example API.

API name

productapi

Description - optional

API endpoint type

Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Regional

4. Now select your root resource and click on Create Resource button.

The screenshot shows the AWS API Gateway 'Resources' section. On the left, there's a sidebar with a 'Create resource' button highlighted by a red box. The main area shows a resource named '/' with a path of '/'. The 'Resource details' section includes a 'Resource ID' field with the value '1ga7qsddqe'. The 'Methods (0)' section has a 'Create method' button at the top right.

5. Here you need to give it a resource name and click on create resource.

The screenshot shows the 'Create resource' dialog. It has a 'Resource details' section with a 'Proxy resource info' checkbox (unchecked) and a note about proxy resources. Below are fields for 'Resource path' (set to '/') and 'Resource name' (set to 'product'). There's also a 'CORS (Cross Origin Resource Sharing) Info' section with a checkbox (unchecked). At the bottom are 'Cancel' and 'Create resource' buttons.

6. Now choose your product resource and click on create method.

The screenshot shows the AWS API Gateway 'Resources' section again. The resource '/' now has a path of '/product'. The 'Resource details' section shows a 'Resource ID' of 'xdxtxi'. The 'Methods (0)' section has a 'Create method' button highlighted by a red box.

7. From the method details you have to choose GET as your method type and in the integration, type choose lambda function then enable the lambda proxy integration choose your lambda function and save your method.

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-1:878893308172:function:ProductFunction

8. After that again choose your product resource and click on Create method.
9. Then choose the method type as POST and in the integration, type choose lambda function. Enable the lambda proxy integration, and choose your function. Click on save.

Method type

POST

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-1:878893308172:function:ProductFunction

10. Here you can see that currently, we have two methods inside our product resource. Now choose the product resource and click on create resource again.

Resources

Resource details

Path
/product

Methods (2)

Method type	Integration type	Authorization	API key
GET	Lambda	None	Not required
POST	Lambda	None	Not required

11. Here you need to create a resource with a name {id}.

Create resource

Resource details

Proxy resource Info
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path: /product/ Resource name: {id}

CORS (Cross Origin Resource Sharing) Info
Create an OPTIONS method that allows all origins, all methods, and several common headers.

12. Now by choosing your ID resource you need to click on the create method.

Resources

Path: /
 /product
 GET
 POST
 /({id})

Resource details

Path: /product/{id}
Resource ID: vb39sv

Methods (0)

Method type	Integration type	Authorization	API key
No methods			

No methods defined.

13. Similarly, you need to create a GET method and choose your lambda function by enabling lambda integration.

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.

14. Then again choose to create a new method in your ID resource and this time create a DELETE method.

Method type

DELETE ▾

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 ▾ X

15. Once we are done creating resources and methods now, we need to deploy our API.
16. Choose to create a new stage and give it a name then click Deploy.

Deploy API X

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#) 

Stage

New stage

Stage name

dev

i A new stage will be created with the default settings. Edit your stage settings on the **Stage** page.

Deployment description

[Cancel](#)

Deploy

17. Once your API is deployed you will be on the stages page. You need to expand the dev stage and come to the GET method. Then choose the invoke URL and paste it in new tab.

The screenshot shows the AWS API Gateway Stages interface. A stage named 'dev' is selected. Under the '/product' endpoint, there are three methods: 'GET', 'POST', and 'DELETE'. The 'GET' method is highlighted. On the right, under 'Method overrides', it says 'This method inherits its settings from the 'dev' stage.' and provides an 'Invoke URL'.

18. You will see that you get a hello from lambda which means that your lambda function is invoked. Use the code given below.



19. Now you need to go back to your lambda function change the code of it to the code given below and deploy it.

```

export const handler = async (event) => {
  console.log('Received event:', JSON.stringify(event, null, 2));
  let body;

  try {
    switch (event.httpMethod) {
      case "GET":
        if (event.queryStringParameters != null) {
          body = `Processing Get Product Id with "${event.pathParameters.id}"` +
            " and Category with "${event.queryStringParameters.category}"` // GET
            product/1234?category=Phone
        }
        else if (event.pathParameters != null) {
          body = `Processing Get Product Id with "${event.pathParameters.id}"`;
          // GET product/1234
        } else {
          body = `Processing Get All Products` // GET product
        }
        break;
      case "POST":
        let payload = JSON.parse(event.body);
        body = `Processing Post Product with "${payload}"` // POST /product
        break;
      case "DELETE":
    }
  }
}

```

```

        if (event.pathParameters != null) {
            body = `Processing Delete Product Id with
"${event.pathParameters.id}"`;// DELETE product/1234
        }
        break;
    default:
        throw new Error(`Unsupported route: "${event.httpMethod}"`);
    }

    console.log(body);
    return {
        statusCode: 200,
        body: JSON.stringify({
            message: `Successfully finished operation: "${event.httpMethod}"`,
            body: body
        })
    };
}

} catch (e) {
    console.error(e);
    return {
        statusCode: 400,
        body: JSON.stringify({
            message: "Failed to perform operation.",
            errorMsg: e.message
        })
    };
}
};

```

20. Once it is done then open Postman and use the GET method. Then copy the invoke URL from the API gateway for your GET method and paste it into the Postman GET method.

The screenshot shows the Postman interface with a successful API call. The URL is `https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product`. The response is a 200 OK status with a duration of 313 ms and a size of 427 B. The response body is:

```

1 {
2   "message": "Successfully finished operation: \"GET\"",
3   "body": "Processing Get All Products"
4 }

```

21. Now you need to choose the invoke URL from the GET Method of your ID resource.

The screenshot shows the AWS Lambda function configuration under the 'Method overrides' section for the `/product/GET` method. It indicates that the method inherits its settings from the 'dev' stage. The invoke URL is listed as `https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product/{id}`.

22. Copy it and come to Postman. Here you need to append the URL with some id as you can see below and click on Send.

GET https://19u7t9hoxf.exec • | GET https://19u7t9hoxf.exec • + | No environment | ↴

HTTP <https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product/2501> Save Share ↴

GET https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product/2501 Send ↴

Params Authorization Headers (5) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results | ⏪ 200 OK • 1347 ms • 440 B • 🌐 | 🎯 | ⚙️ | ⚡

Pretty Raw Preview Visualize JSON ⚡

```

1 {
2   "message": "Successfully finished operation: \"GET\"",
3   "body": "Processing Get Product Id with \"2501\""
4 }
```

23. Similarly, copy the URL for the POST method copy the POST invoke URL from API gateway.

GET https://19u7t9hoxf.exec • | GET https://19u7t9hoxf.exec • | POST https://19u7t9hoxf.exec • + | No environment | ↴

HTTP <https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product> Save Share ↴

POST https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product Send ↴

Params Authorization Headers (7) **Body** ● Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** ⚡ Beautify

1 {	"name": "iPhone",
2	"price": "1500"
3 }	

Body Cookies Headers (7) Test Results | ⏪ 200 OK • 344 ms • 450 B • 🌐 | 🎯 | ⚙️ | ⚡

Pretty Raw Preview Visualize JSON ⚡

```

1 {
2   "message": "Successfully finished operation: \"POST\"",
3   "body": "Processing Post Product with \"[object Object]\""
4 }
```

24. Finally, choose the DELETE method and paste the invoke URL for the delete method from the ID resource in your API Gateway.

The screenshot shows the Postman interface with the following details:

- URL:** https://19u7t9hoxf.execute-api.us-east-1.amazonaws.com/dev/product/2501
- Method:** DELETE
- Response Status:** 200 OK
- Response Time:** 322 ms
- Response Size:** 446 B
- Response Body (Pretty JSON):**

```
1 {
2   "message": "Successfully finished operation: \"DELETE\"",
3   "body": "Processing Delete Product Id with \"2501\""
4 }
```

25. As you can see, we have worked through the CRUD application. Now you can delete all the resources start with API gateway and your lambda function.