😊 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

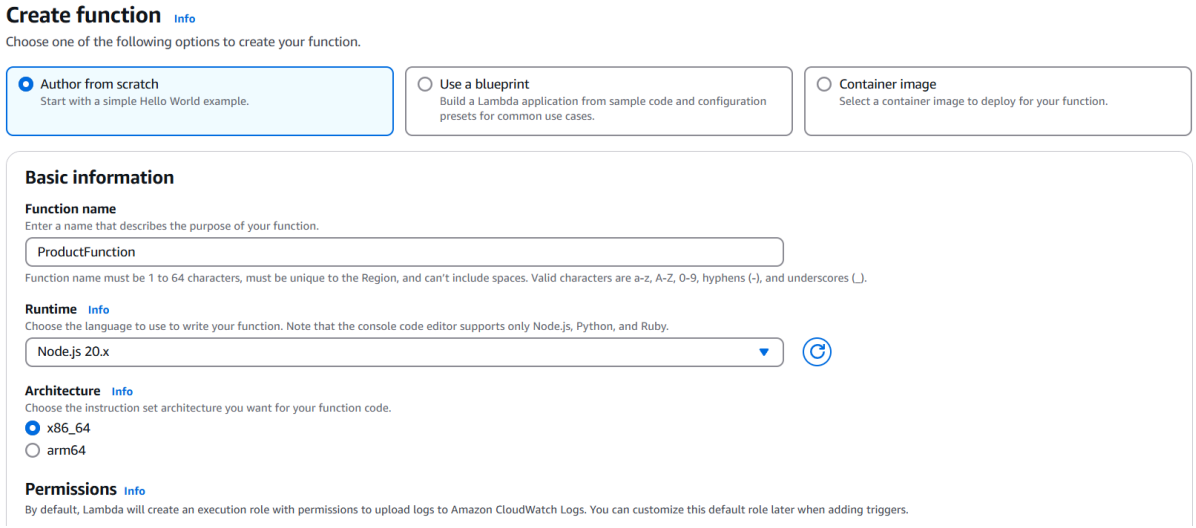
- **GET Method**: Fetch product info by ID.
- **POST Method**: Add a new product with details from the request body.

Conclusion

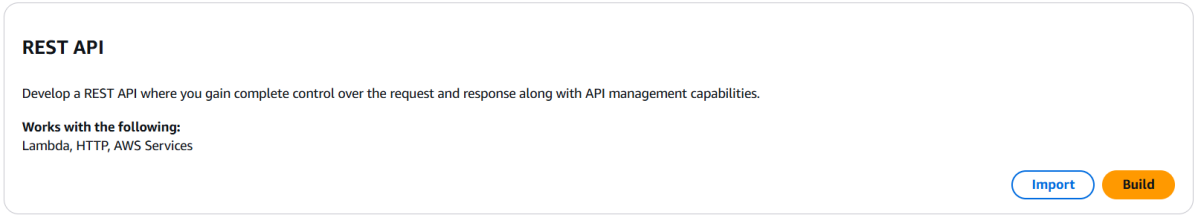
This lab demonstrates how to build scalable microservices using **AWS Lambda** and **API Gateway**, where the Lambda function processes requests based on different event objects from the client.

😄 To begin with the Lab:

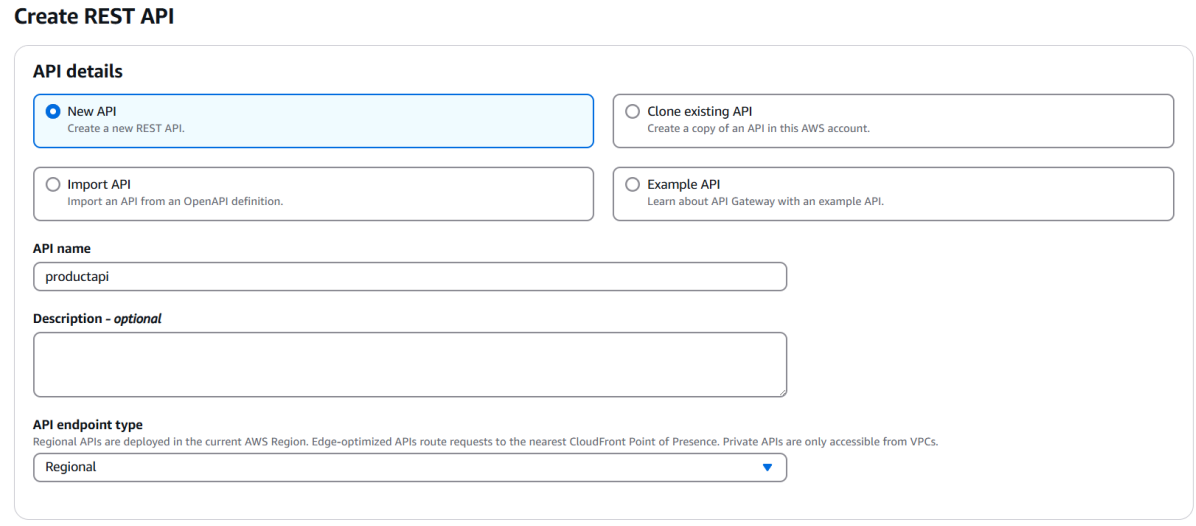
1. First, we are going to create a Lambda function for our environment. So, go to lambda and click on the create function. Here you need to choose the author from scratch option and give your function a name choose runtime as Node.js and create your function.



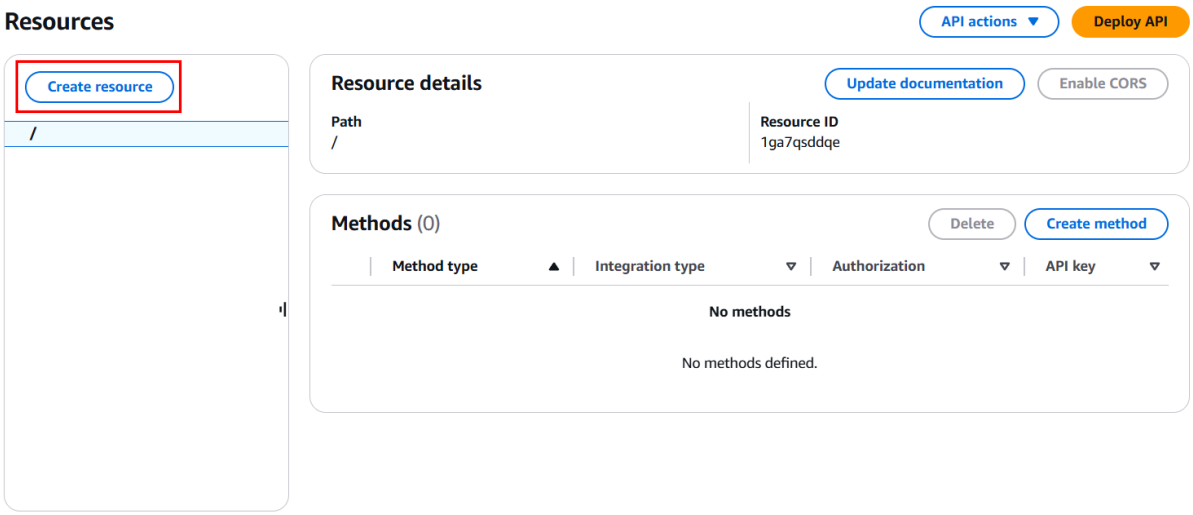
2. Once your lambda function is created you need to navigate to API gateway and choose to build a REST API.



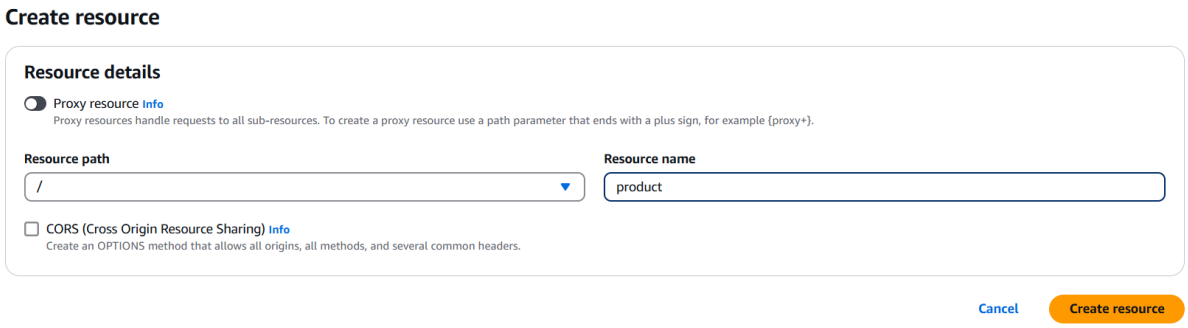
3. Choose to create a new API, give it a name and click on create.



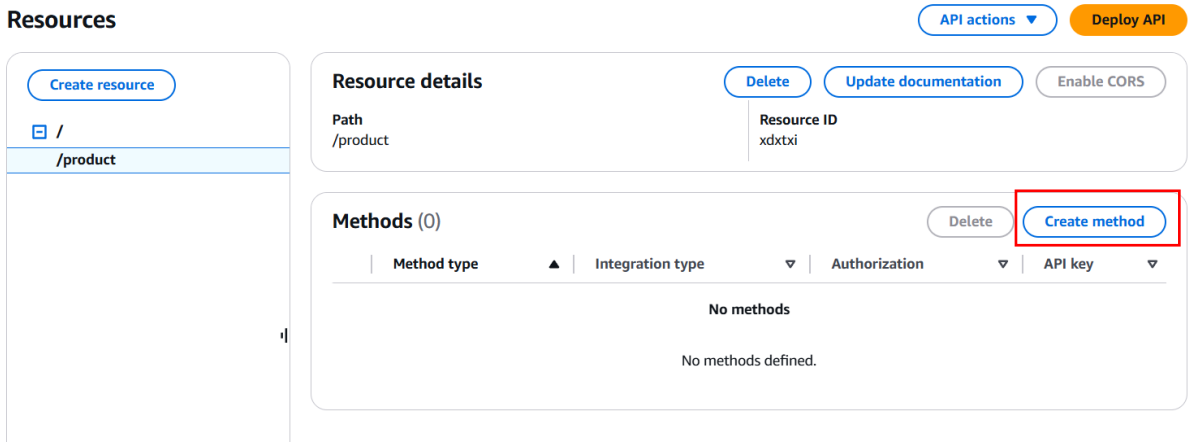
4. Now select your root resource and click on Create Resource button.



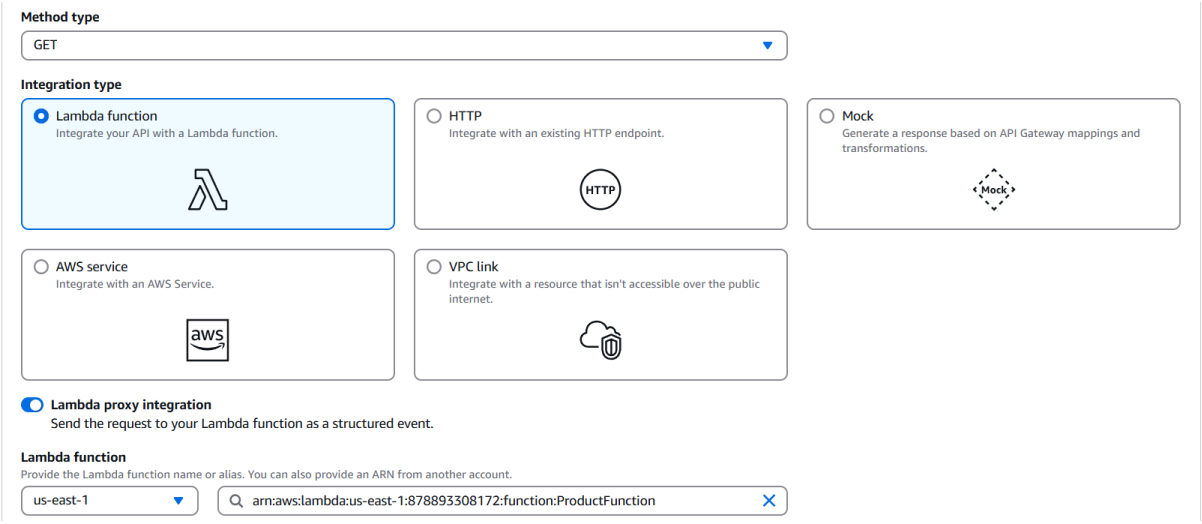
5. Here you need to give it a resource name and click on create resource.



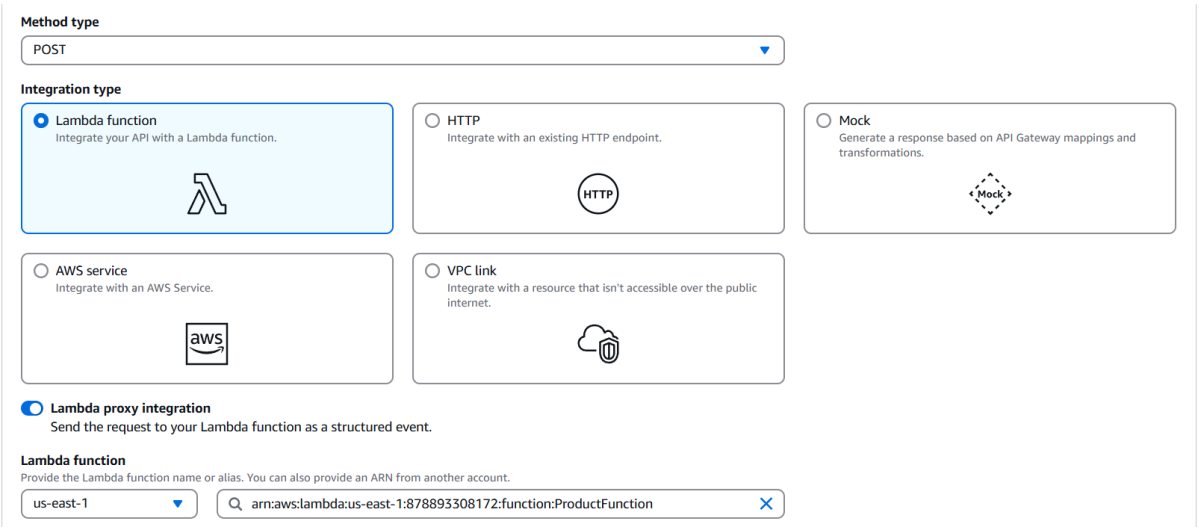
6. Now choose your product resource and click on create method.



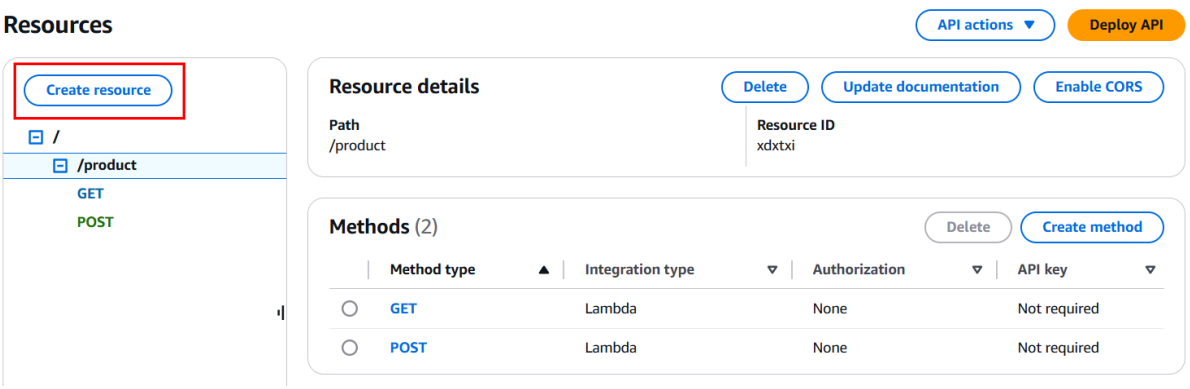
7. From the method details you have to choose GET as your method type and in the integration, type choose lambda function then enable the lambda proxy integration choose your lambda function and save your method.



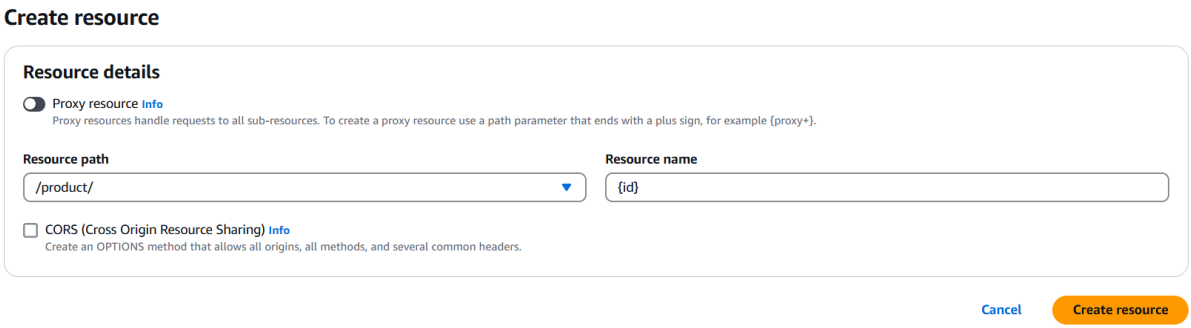
8. After that again choose your product resource and click on Create method.
9. Then choose the method type as POST and in the integration, type choose lambda function. Enable the lambda proxy integration, and choose your function. Click on save.



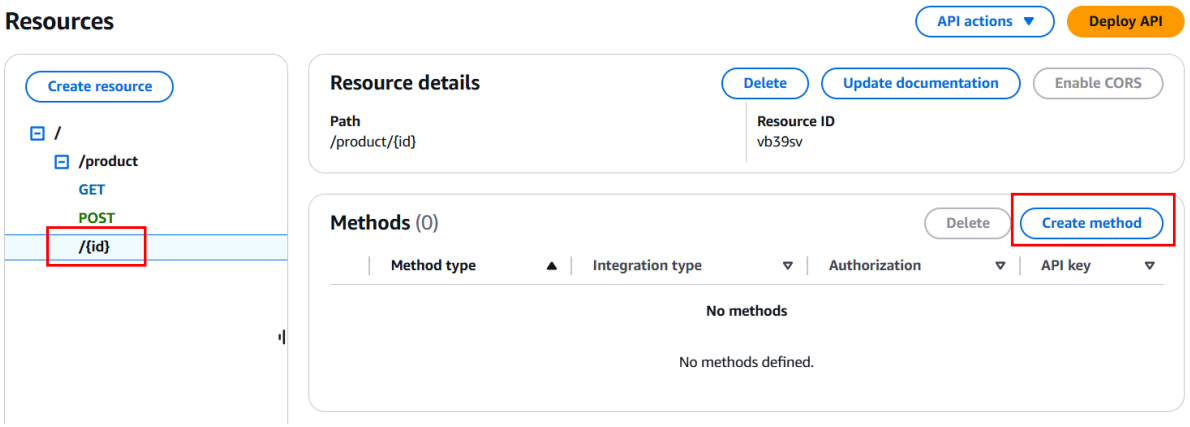
10. Here you can see that currently, we have two methods inside our product resource. Now choose the product resource and click on create resource again.



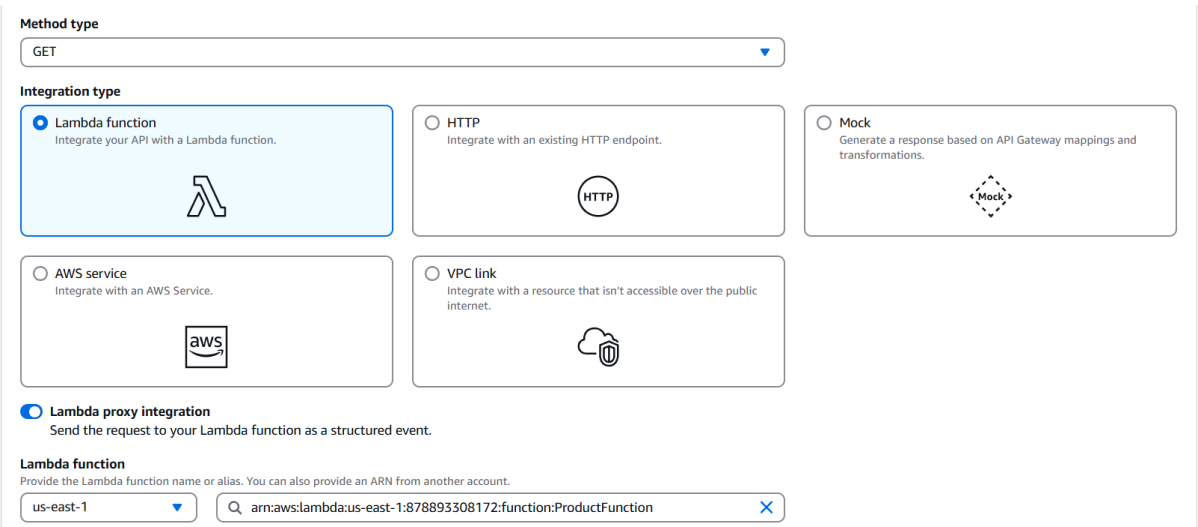
11. Here you need to create a resource with a name {id}.



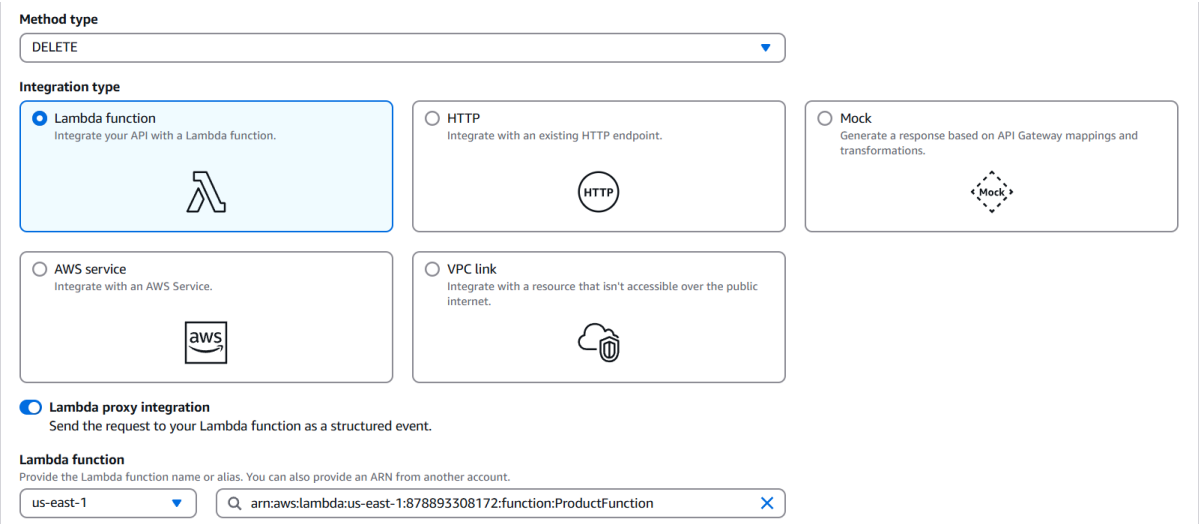
12. Now by choosing your ID resource you need to click on the create method.



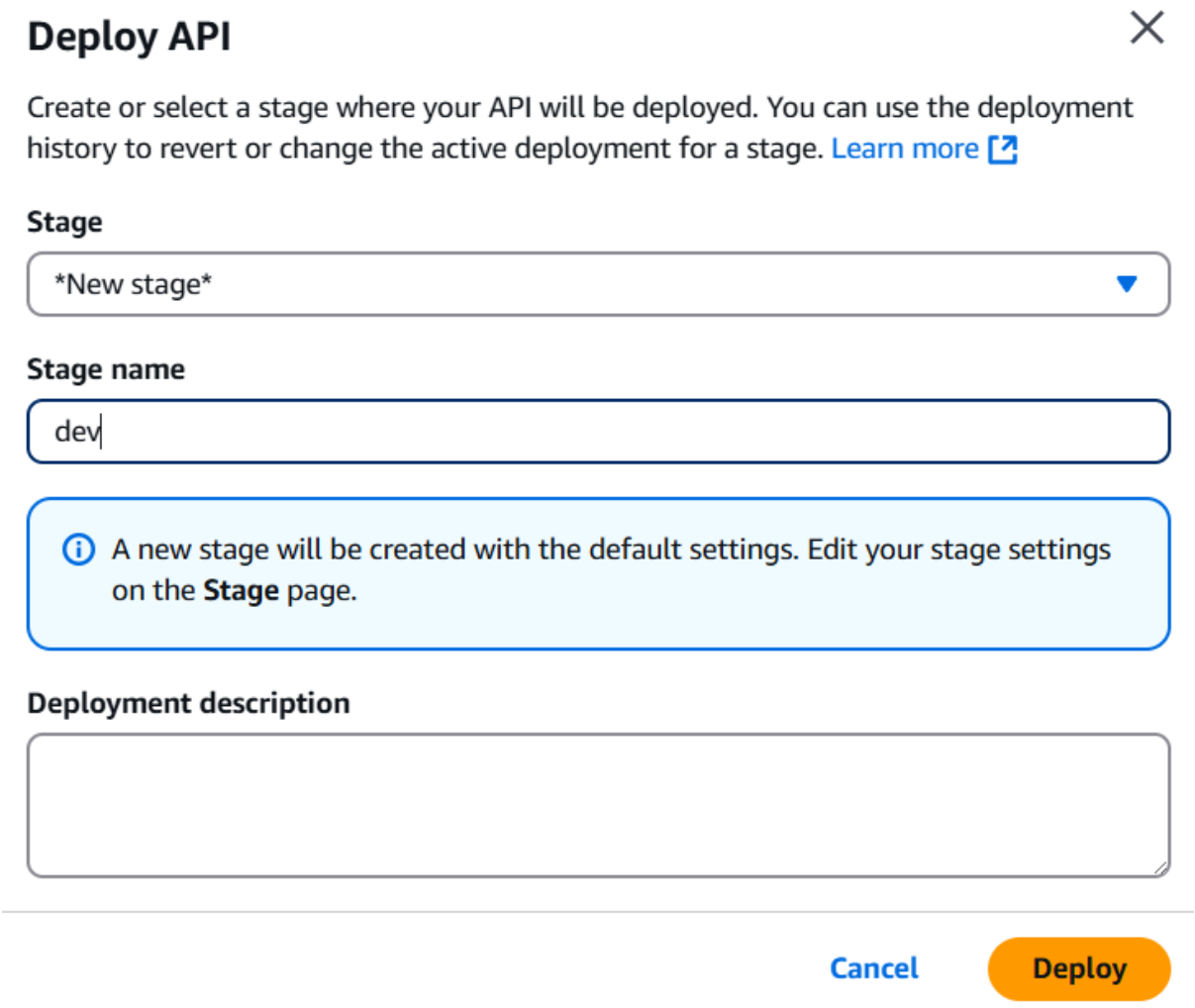
13. Similarly, you need to create a GET method and choose your lambda function by enabling lambda integration.



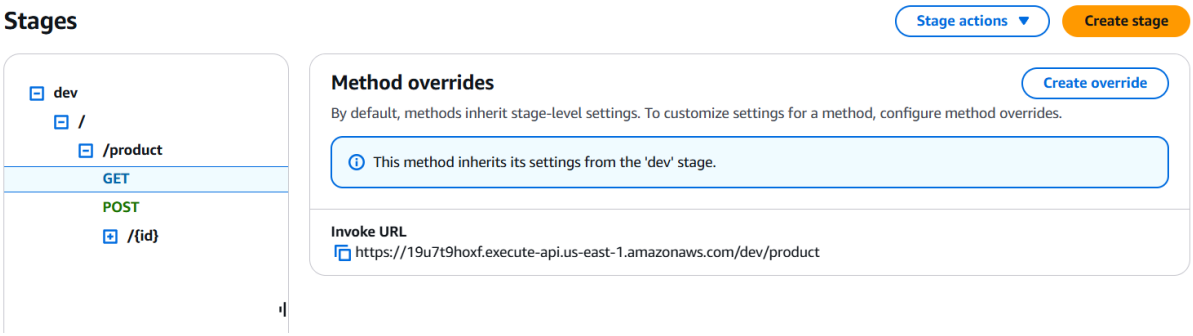
14. Then again choose to create a new method in your ID resource and this time create a DELETE method.



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.



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.



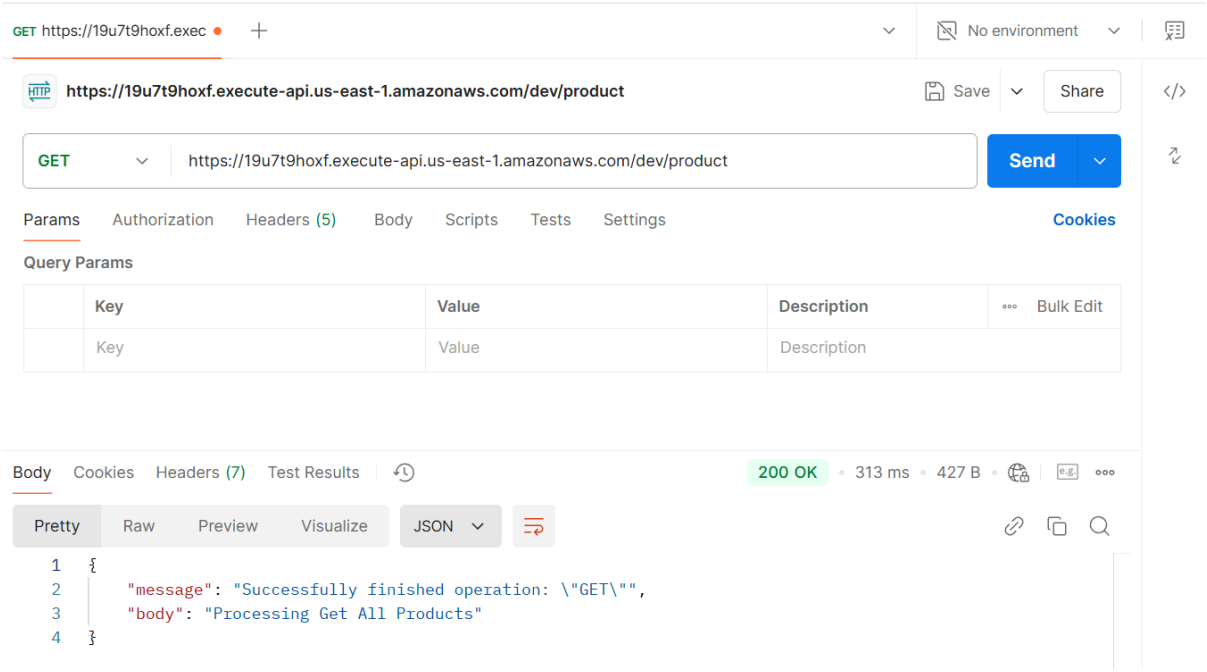
18. You will see that you get a hello from lambda which means that your lambda function is invoked. Use the code given below.

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

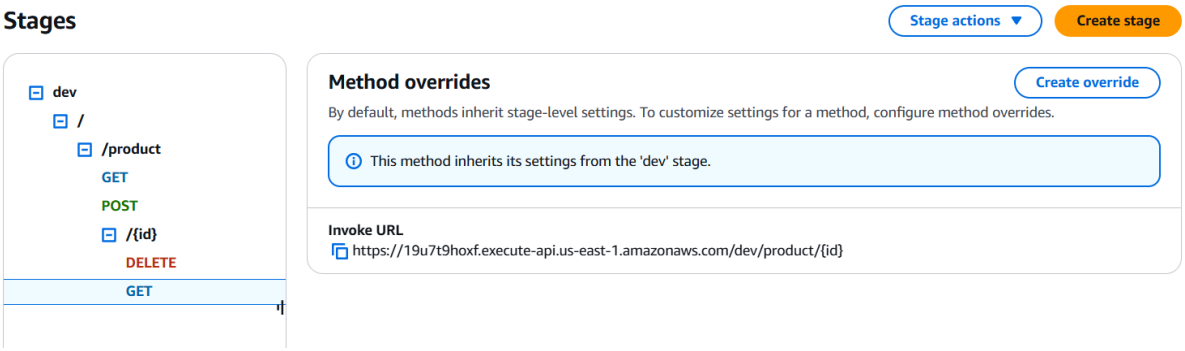
```
export const handler = async (event) => {
  console.log('Received event:', JSON.stringify(event, null, 2));
  let body;

  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
        })
    };
 }
};
```

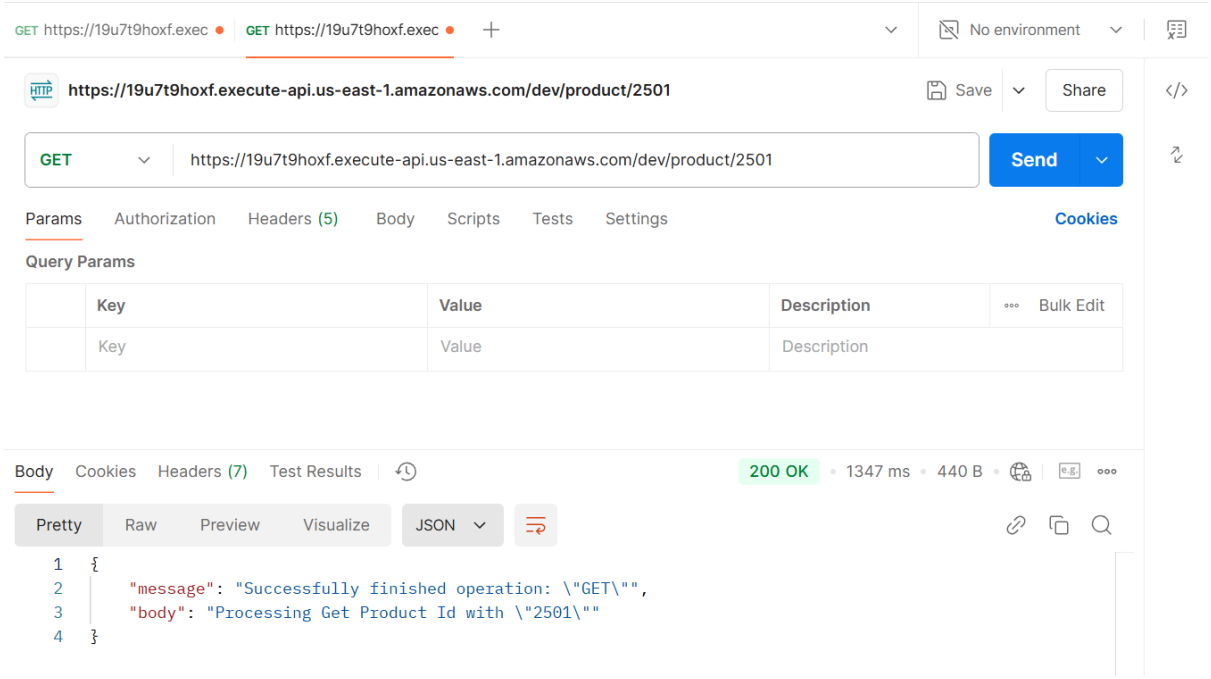
20. Once it is done then open Postman and use the GET method. Then copy the invoke URL from the API gateway for your GET method and paste it into the Postman GET method.



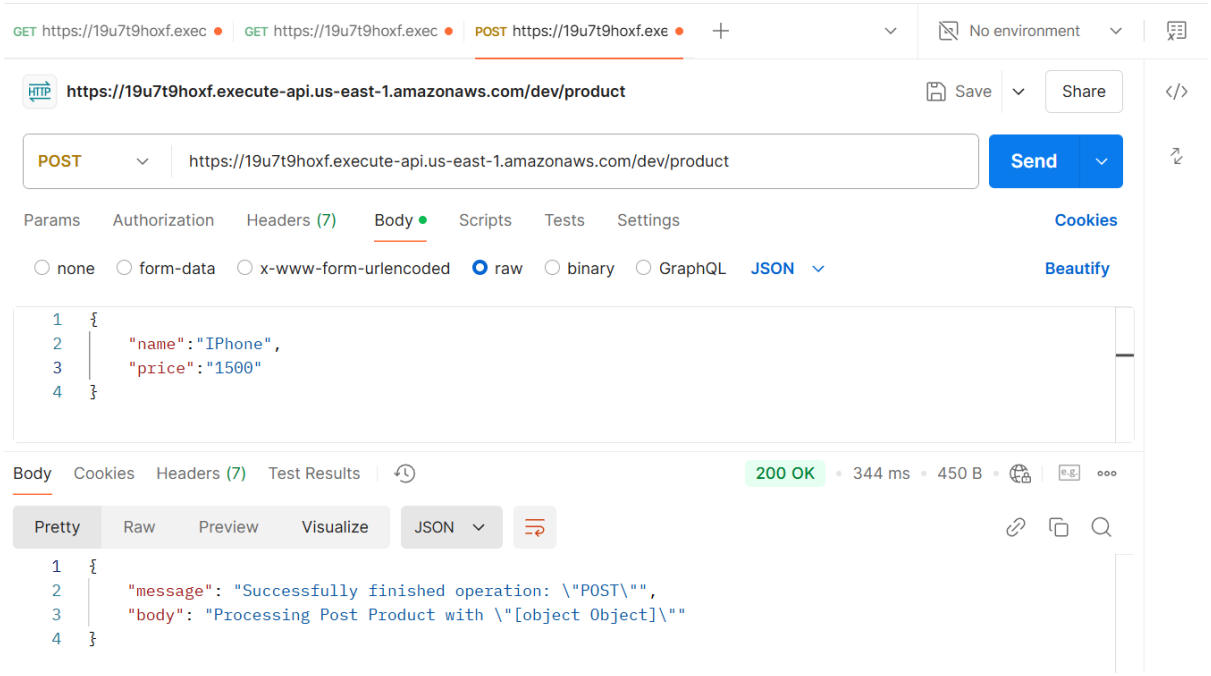
21. Now you need to choose the invoke URL from the GET Method of your ID resource.



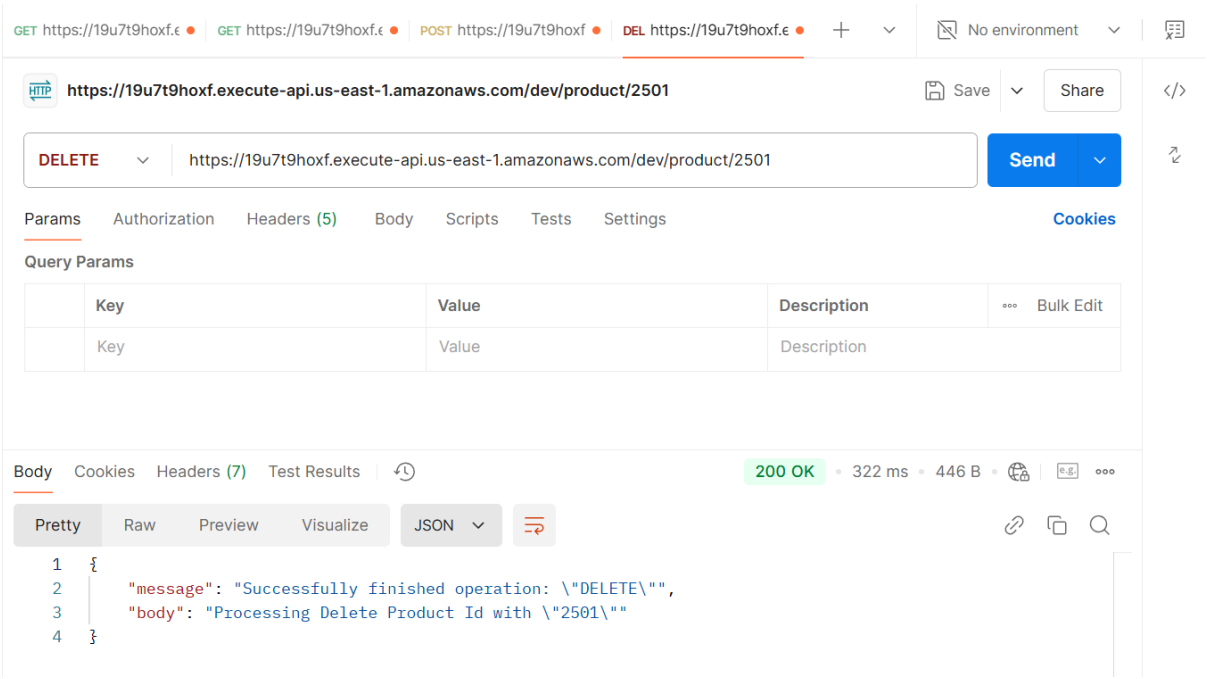
22. Copy it and come to Postman. Here you need to append the URL with some id as you can see below and click on Send.



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



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



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.