

The Amazon API Gateway Workshop

Introduction

▶ Getting Started

▶ Module 1 - Introduction to Amazon API Gateway

▶ Module 2 - Deploy your first API with IaC

▼ Module 3 - API Gateway REST Integrations

Module Goals

▶ Mock

▶ HTTP Integration

▶ AWS Lambda

▶ AWS Step Functions

▶ Amazon SQS

▶ Amazon SNS

▶ Amazon Kinesis

▼ Amazon DynamoDB

Setup your AWS SAM Project

Build Amazon DynamoDB Integration with AWS SAM and OpenAPI

Test the Integration

▶ Amazon EventBridge

▶ Private Integration

▶ Amazon S3

Clean up

▶ Module 4 - Observability in API Gateway

▶ Module 5 - WebSocket APIs

▶ Module 6 - Enable fine-grained access control for your APIs

Clean up

Resources

Build Amazon DynamoDB Integration with AWS SAM and OpenAPI

You can integrate an API method directly with an Amazon DynamoDB without the need for a compute layer such as Lambda in the middle. This is achieved through the **AWS service integration** type.

Use AWS SAM and OpenAPI to create an API Gateway REST API with a direct Amazon DynamoDB integration

1. Using AWS Cloud9 console, return to the root folder `module-3/dynamodb`
2. This code belongs in your [SAM](#) template file `template.yaml`

Review the code and then **copy/paste** it into the `template.yaml` file.

```
1  AWSTemplateFormatVersion: '2010-09-09'
2  Transform: 'AWS::Serverless-2016-10-31'
3  Description: API Gateway with DynamoDB integration using SAM
4
5  Globals:
6
7    # Enable Logs
8  Api:
9    MethodSettings:
10     - ResourcePath: "/"
11     HttpMethod: "*"
12     DataTraceEnabled: True
13     LoggingLevel: INFO
14     MetricsEnabled: True
15
16  Resources:
17    DynamoDBTable:
18      Type: 'AWS::DynamoDB::Table'
19      Properties:
20        TableName: MyTable
21        AttributeDefinitions:
22          - AttributeName: id
23            AttributeType: S
24        KeySchema:
25          - AttributeName: id
26            KeyType: HASH
27        ProvisionedThroughput:
28          ReadCapacityUnits: 5
29          WriteCapacityUnits: 5
30
31  # APIGW Rest API for DynamoDB Integration Example
32  RestApiForDynamoDB:
33    Type: 'AWS::Serverless::Api'
34    Properties:
35      StageName: dev
36      DefinitionBody: # an OpenApi definition
37        'Fn::Transform':
38          Name: 'AWS::Include'
39          Parameters:
40            Location: 'openapi.yaml'
41      OpenApiVersion: 3.0.3
42      EndpointConfiguration:
43        Type: REGIONAL
44
45  # IAM Role to allow APIGW to call DynamoDB
46  RestApiRole:
47    Type: 'AWS::IAM::Role'
48    Properties:
49      AssumeRolePolicyDocument:
50        Version: 2012-10-17
51        Statement:
52          - Effect: Allow
53            Principal:
54              Service:
55                - apigateway.amazonaws.com
56            Action:
57              - 'sts:AssumeRole'
58      Policies:
59        - PolicyName: AllowDynamoDBExec
60          PolicyDocument:
61            Version: 2012-10-17
62            Statement:
63              - Effect: Allow
64                Action:
65                  - "dynamodb:Query"
66                  - "dynamodb:PutItem"
67                  - "dynamodb:UpdateItem"
68                Resource:
69                  - !GetAtt DynamoDBTable.Arn
70
71  Outputs:
72    MyApiUrl:
73      Description: URL of the API
74      Value: !Sub "https://${RestApiForDynamoDB}.execute-api.${AWS::Region}.amazonaws.com/dev"
```

The `DynamoDBTable` resource defines a **DynamoDB** table that will be accessed by the API. The `RestApiRole` resource defines a *Role* that will be used by the routes of the API.

The *Policies* attribute (**lines 58-69**) specifies the actions that will be allowed to the API calls. In this case, the caller will be able to *Query*, *PutItem* and *UpdateItem* of the DynamoDB table referred (**line 69**).

3. This code belongs in your [OpenAPI](#) definition file `openapi.yaml`

Review the code and then **copy/paste** it into the `openapi.yaml` file

```
1  openapi: "3.0.1"
2  info:
3    title: "module-3-dynamodb-integration"
4    description: "API Gateway example for Amazon DynamoDB Integration"
5    version: "1.0"
6  paths:
7    /resource:
8      get:
9        responses:
10         "200":
11           description: "200 response"
12           content:
13             application/json:
14               schema:
15                 $ref: "#/components/schemas/Empty"
16         x-amazon-apigateway-integration:
17           type: "aws"
18           credentials:
19             Fn::Sub: "${RestApiRole.Arn}"
20           httpMethod: "POST"
21           uri: "arn:aws:apigateway:${AWS::Region}:dynamodb:action/Query"
22           responses:
23             default:
24               statusCode: "200"
25               responseTemplates:
26                 application/json: "#set($inputRoot = $input.path('$'))\n{\n  \"Names\":\n    : {\n      #foreach($elem in $inputRoot.Items) {\n        id\": \"$elem.id.S\", \n        \"name\": \"$elem.name.S\", \n        \n        \"lastName\": \"$elem.lastName.S\", \n        \"email\": \"$elem.email.S\" \n      } \n    }\n  }\n}"
27                 requestTemplates:
28                 application/json: "{\n  \"TableName\": \"$MyTable\", \n  \"KeyConditionExpression\":\n    : \"$id = :v1\", \n  \"ExpressionAttributeValues\": {\n    \":v1\":\n    : {\n      \"S\": \"$input.params('id')\" \n    } \n  }\n}"
29                 passthroughBehavior: "when_no_templates"
30       /resource/{id}:
31       post:
32         parameters:
33           - name: "id"
34             in: "path"
35             required: true
36             schema:
37               type: "string"
38         responses:
39         "200":
40           description: "200 response"
41           content:
42             application/json:
43               schema:
44                 $ref: "#/components/schemas/Empty"
45         x-amazon-apigateway-integration:
46           type: "aws"
47           credentials:
48             Fn::Sub: "${RestApiRole.Arn}"
49           httpMethod: "POST"
50           uri: "arn:aws:apigateway:${AWS::Region}:dynamodb:action/PutItem"
51           responses:
52             default:
53               statusCode: "200"
54               requestTemplates:
55                 application/json: "{\n  \"TableName\": \"$MyTable\", \n  \"Item\": {\n    \":v1\": {\n      \"id\": \"$input.params('id')\", \n      \"name\": \"$input.path('$name')\", \n      \"lastName\": \"$input.path('$lastName')\", \n      \"email\": \"$input.path('$email')\" \n    } \n  }\n}"
56                 passthroughBehavior: "when_no_templates"
57         components:
58           schemas:
59             Empty:
60               title: "Empty Schema"
61               type: "object"
```

Lines (**27-57**) are particular important as these are what create the direct service integration for DynamoDB. The `x-amazon-apigateway-integration` is an OpenAPI extension that allows us to integrate our APIs to AWS backends.

On the `/resource` path, lines (**33-35**) defines the model schema of the *Query* request to be made using the [json schema](#) pattern. This will transform the simple pageld path parameter on the GET request to the needed DynamoDB Query API, which requires an HTTP POST. Lines (**26-31**) defines the response template from the *Query*. [DynamoDB](#) uses a specific query syntax that is been transformed for json one.

On the `/resource/{id}` path, lines (**33-35**) defines the model schema of the *PutItem* request. This mapping template creates the JSON structure required by the DynamoDB PutItem API. The entire mapping template is static. The three input variables are referenced from the request json using the `$input` variable and each comment is stamped with a unique identifier. Also, the `id` is obtained from the parameters of the request.

Deploy the project

1. To deploy the API Gateway and Kinesis resources to your AWS account, run the following commands from the application root `module-3/dynamodb`, where the `template.yaml` file for the sample application is located:

```
1 sam build && sam deploy --guided
```

The first time that you run the `sam deploy --guided` command, AWS SAM starts an AWS CloudFormation deployment. In this case, you need to say what are the configurations that you want SAM to have in order to get the guided deployment. You can configure it as below.

- **Stack Name:** `module-3-dynamodb-integration`
- **AWS Region:** Put the chosen region to run the workshop. e.g. `us-east-1`
- **Parameter KinesisStreamName:** leave blank
- **Confirm changes before deploy:** `Y`
- **Allow SAM CLI IAM role creation:** `Y`
- **Disable rollback:** `N`
- **Save arguments to configuration file:** `Y`
- **SAM configuration file and SAM configuration environment** leave blank

```
Configuring SAM deploy
=====

Looking for config file [samconfig.toml] : Found
Reading default arguments : Success

Setting default arguments for 'sam deploy'
=====
Stack Name [module-3-dynamodb-integration]: module-3-dynamodb-integration
AWS Region [us-west-2]:
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [Y/n]: Y
#SAM needs permission to be able to create roles to connect to the resources in your template
Allow SAM CLI IAM role creation [Y/n]: Y
#Preserves the state of previously provisioned resources when an operation fails
Disable rollback [y/N]: N
Save arguments to configuration file [Y/n]: Y
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:
```

2. After configuring the deployment, AWS SAM will display assets that will be created. But first, it will automatically upload the template to a temporary bucket it creates. Then, it will ask you to confirm the changes. Type `y` to confirm.

```
CloudFormation outputs from deployed stack

Outputs

Key MyApiUrl
Description URL of the API
Value https://[redacted].execute-api.us-west-2.amazonaws.com/dev

Successfully created/updated stack - module-3-dynamodb-integration in us-west-2
```

3. Once the deployment has been successful, you will see an **'Outputs'** section that contains the API Gateway invoke URL

Select your cookie preferences

We use essential cookies and similar tools that are necessary to provide our site and services. We use performance cookies to collect anonymous statistics, so we can understand how customers use our site and make improvements. Essential cookies cannot be deactivated, but you can choose "Customize" or "Decline" to decline performance cookies.

If you agree, AWS and approved third parties will also use cookies to provide useful site features, remember your preferences, and display relevant content, including relevant advertising. To accept or decline all non-essential cookies, choose "Accept" or "Decline." To make more detailed choices, choose "Customize."

Accept

Decline

Customize

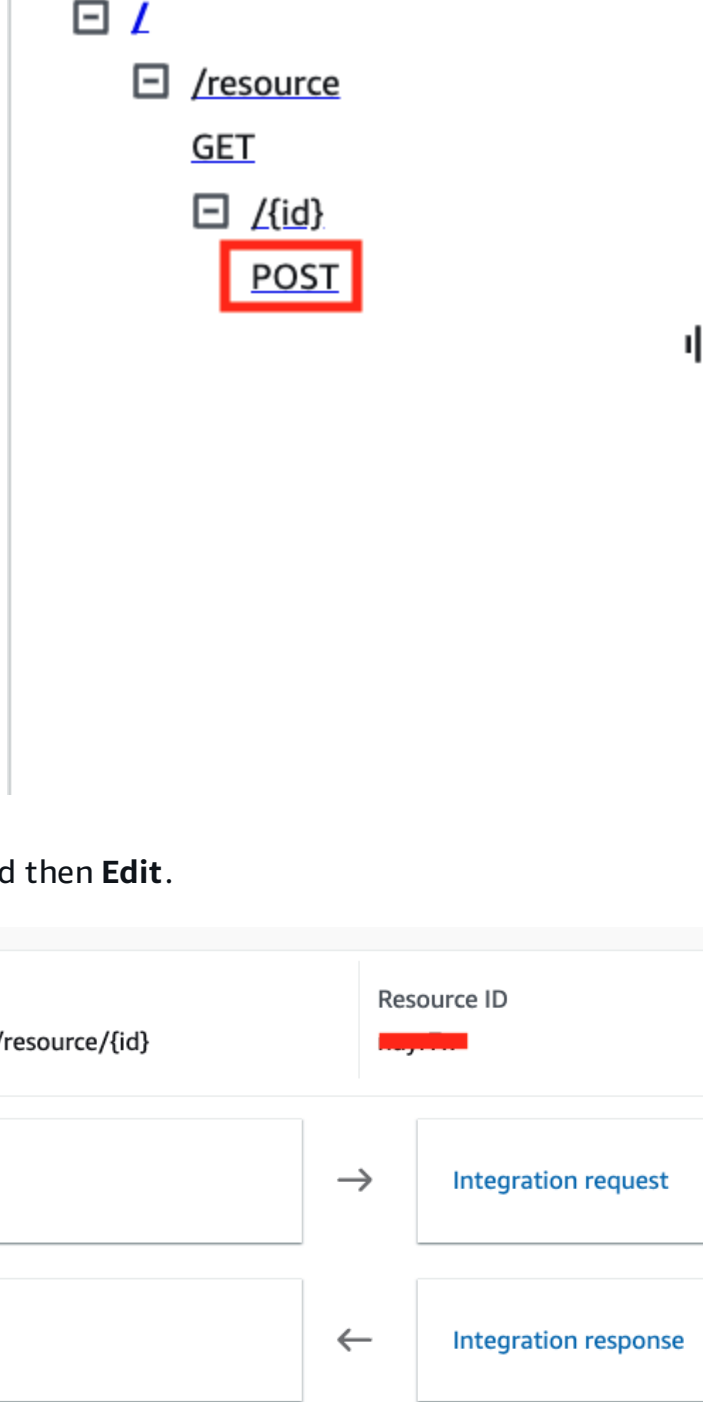
Test the Integration

Now the API has been deployed, the table is in place, its time to test the integration.

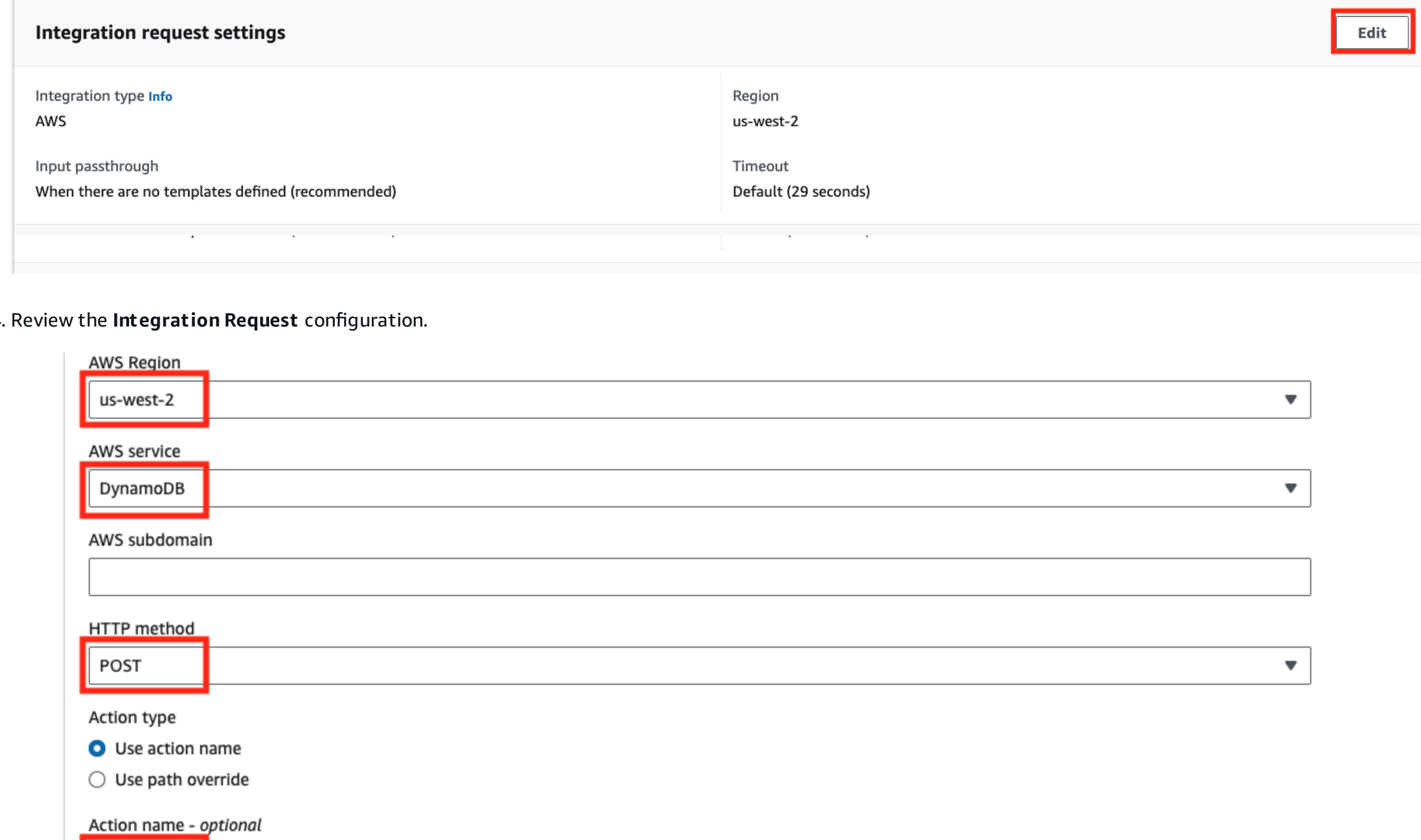
Test DynamoDB Integration using API Gateway console to WRITE an item

First, let's test the write integration (*PutItem* api call).

1. Navigate to the API Gateway console and select the API that has been created. It will be named **module-3-dynamodb-integration**.
2. Select the POST method from the resources section to the left of the screen to bring up the API request flow window.



3. In the *Method Execution*, click in **Integration Request** and then **Edit**.



4. Review the **Integration Request** configuration.

AWS Region us-west-2

AWS service DynamoDB

AWS subdomain

HTTP method POST

Action type
☒ Use action name
☐ Use path override

Action name - optional PutItem

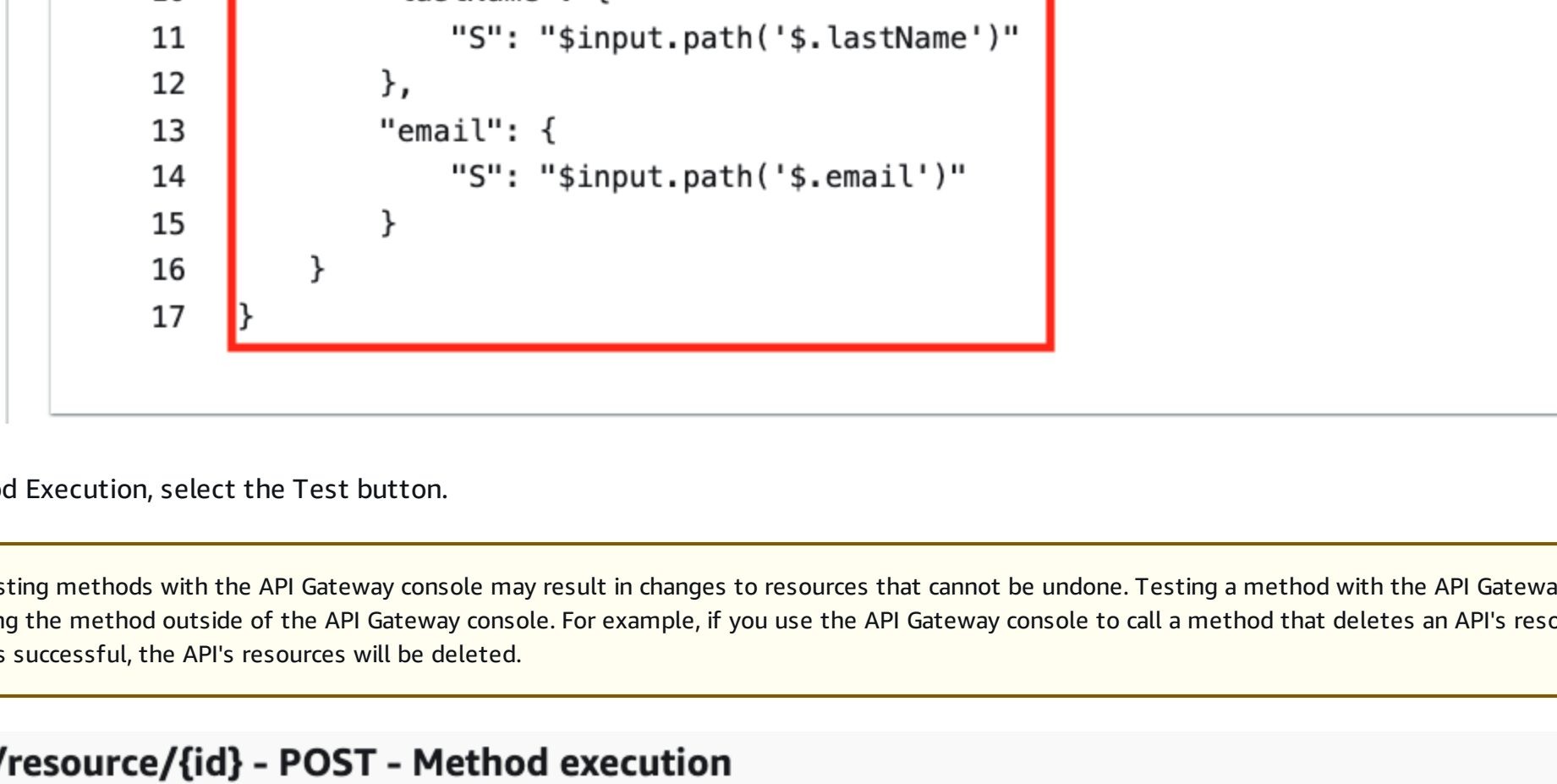
Execution role arn:aws:iam::123456789012:role/module-3-dynamodb-integration-RestApiRole-VFJH6dfoWbB8

Credential cache
Do not add caller credentials to cache key

Content handling Learn more

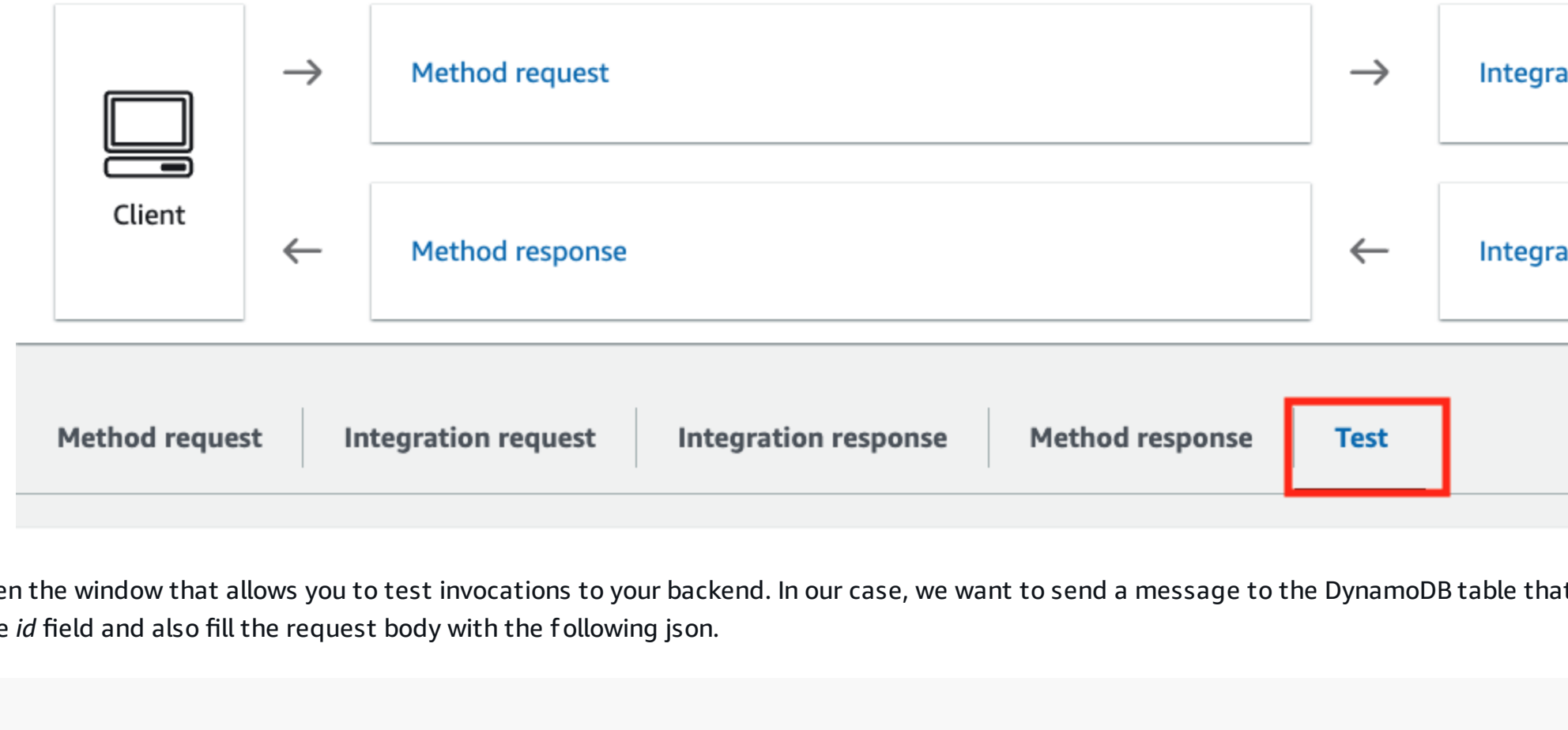
Passthrough

5. Click on **Save**. In **Integration Request** tab go down to the **Mapping Template** to see the input transformation request template:



6. Go back to Method Execution, select the Test button.

Warning! Testing methods with the API Gateway console may result in changes to resources that cannot be undone. Testing a method with the API Gateway console is the same as calling the method outside of the API Gateway console. For example, if you use the API Gateway console to call a method that deletes an API's resources, if the method call is successful, the API's resources will be deleted.



7. This will open the window that allows you to test invocations to your backend. In our case, we want to send a message to the DynamoDB table that we created early. Fill the *id* field and also fill the request body with the following json.

```
1 {
2   "name": "John",
3   "lastName": "Doe",
4   "email": "jd@example.com"
5 }
```

Path
id
1

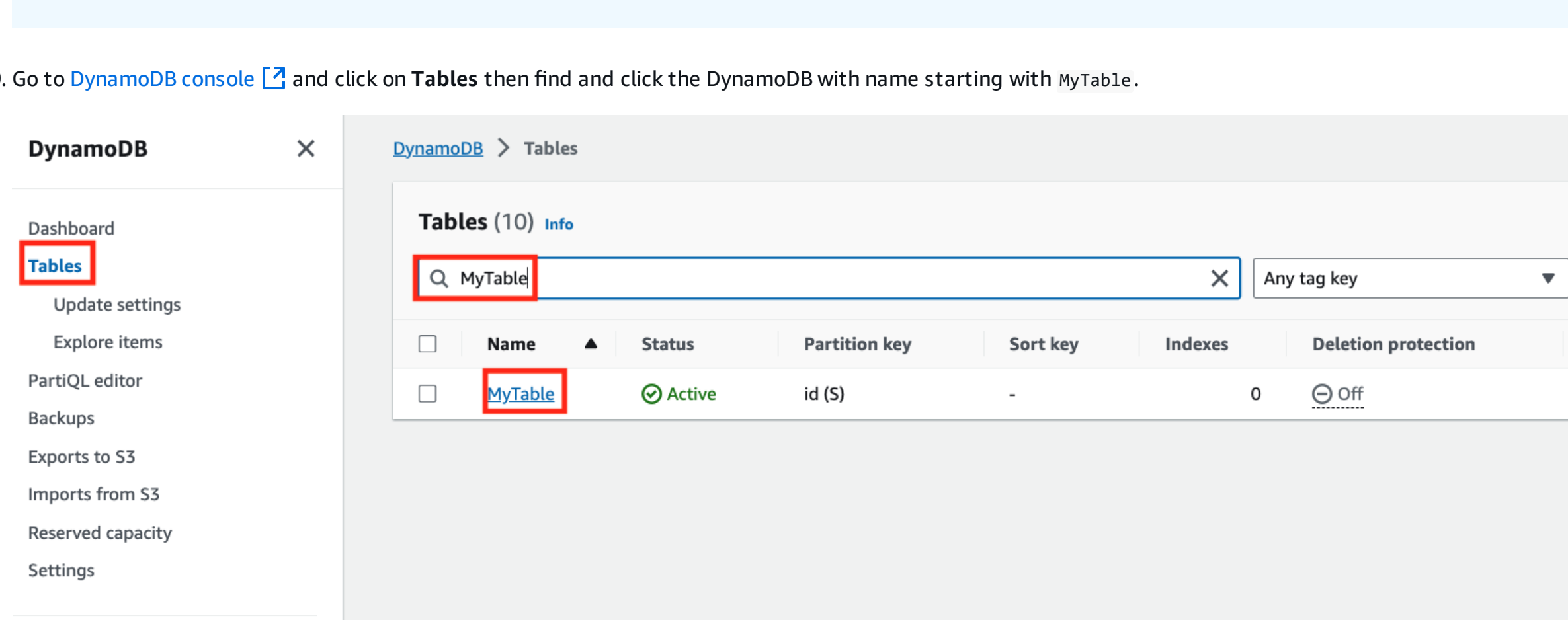
Query strings
param1=value1¶m2=value2

Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2

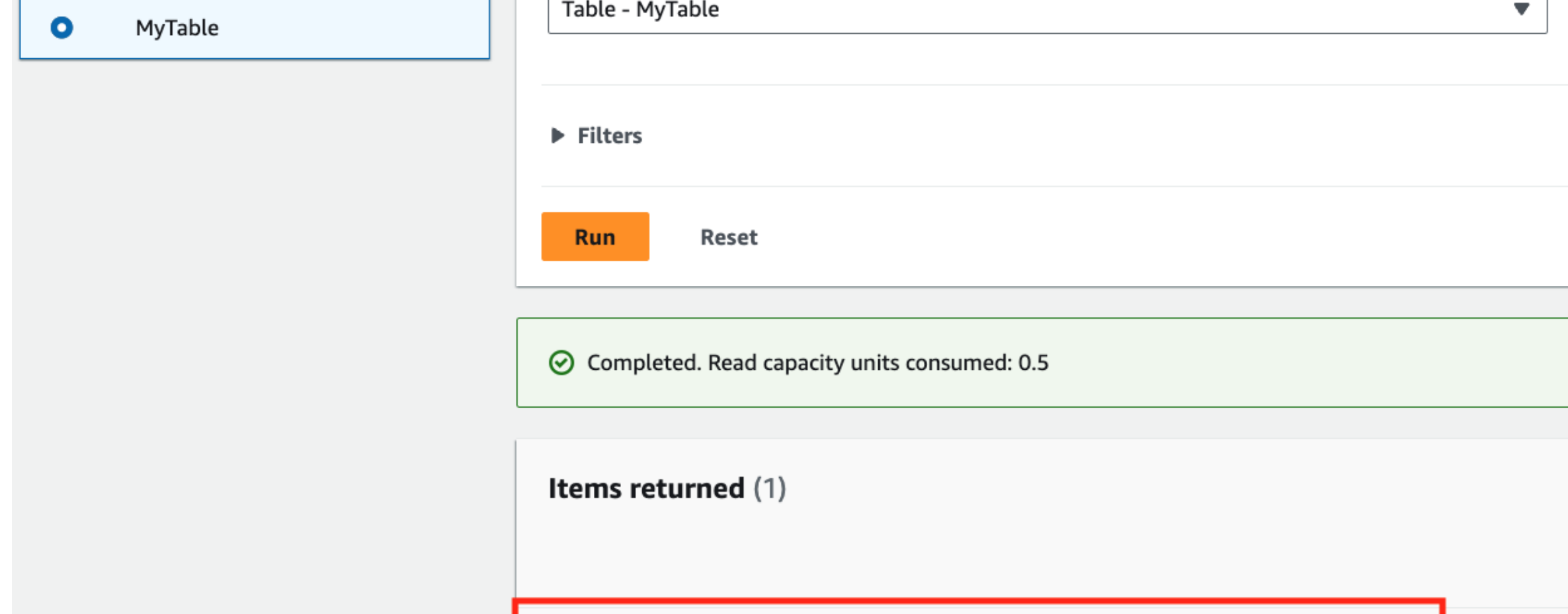
Client certificate
No client certificates have been generated.

Request body
1 {
2 "name": "John",
3 "lastName": "Doe",
4 "email": "jd@example.com"
5 }

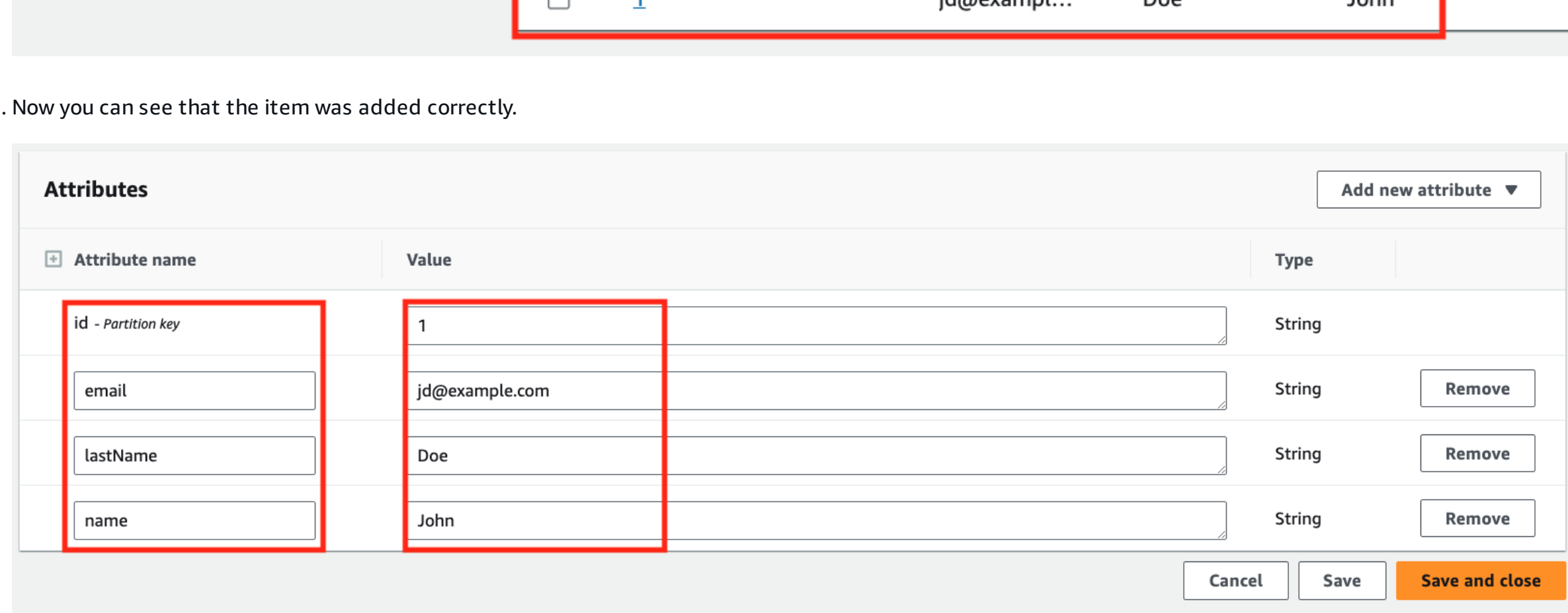
8. Scroll to the bottom of the page and hit the **Test** button. From this, you should receive a Status code of 200 meaning the message was successfully delivered to the DynamoDB table.



9. Go to **DynamoDB console** and click on **Tables** then find and click the DynamoDB with name starting with *mytable*.



10. In the right of the console, select **Explore Table Items**. Then, in *Items Returned* click in the one with the *id=1*.



11. Now you can see that the item was added correctly.

Attributes

Attribute name	Value	Type	
id - Partition key	1	String	
email	jd@example.com	String	Remove
lastName	Doe	String	Remove
name	John	String	Remove

Cancel Save Save and close

Test the deployed API using cURL to WRITE an item

1. Open a new terminal window in your AWS Cloud9 environment.
2. Copy the below cURL command and paste it into the terminal window, replacing *<api-id>* with your API ID and *<region>* with the region where your API is deployed. You can also get the full URL from the Outputs section from your SAM deployment. You can also find the full invoke URL in the API Gateway console by navigating to **Stages** > **dev**.

```
1 curl -X POST \
2   https://<api-id>.execute-api.<region>.amazonaws.com/dev/resource/1 \
3   -H 'Content-Type: application/json' \
4   -d '{"name": "John", "lastName": "Doe", "email": "jd@example.com"}'
```

3. The **Response Body** output should be:

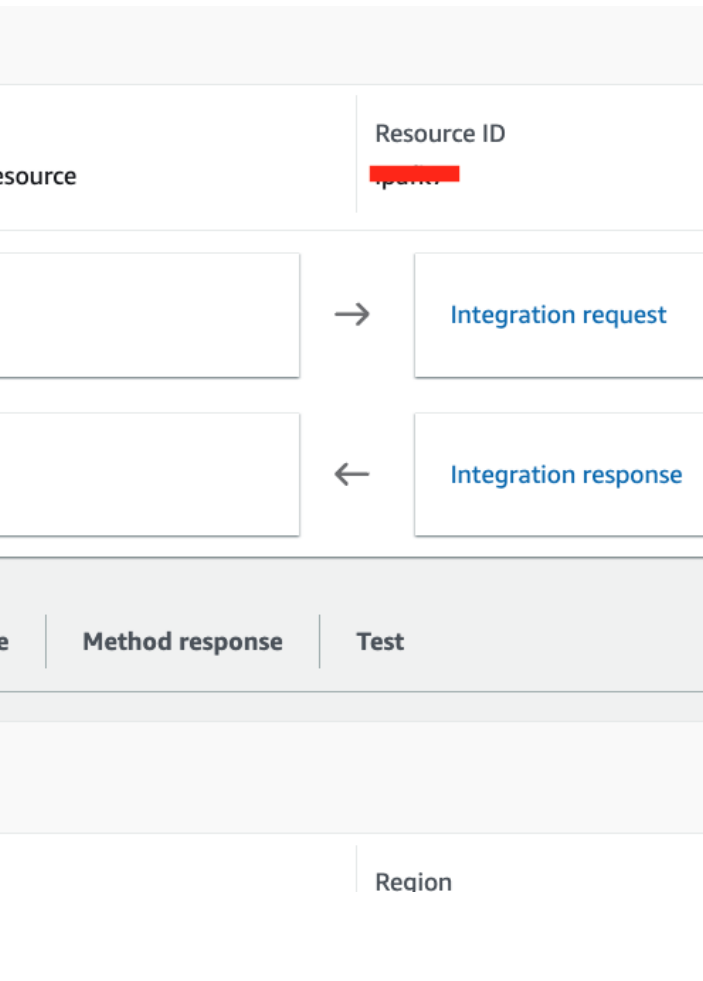
```
1 {
2 }
```

Challenge
Create new **Response Body** in the API Gateway to send a success message when add a record into the table.

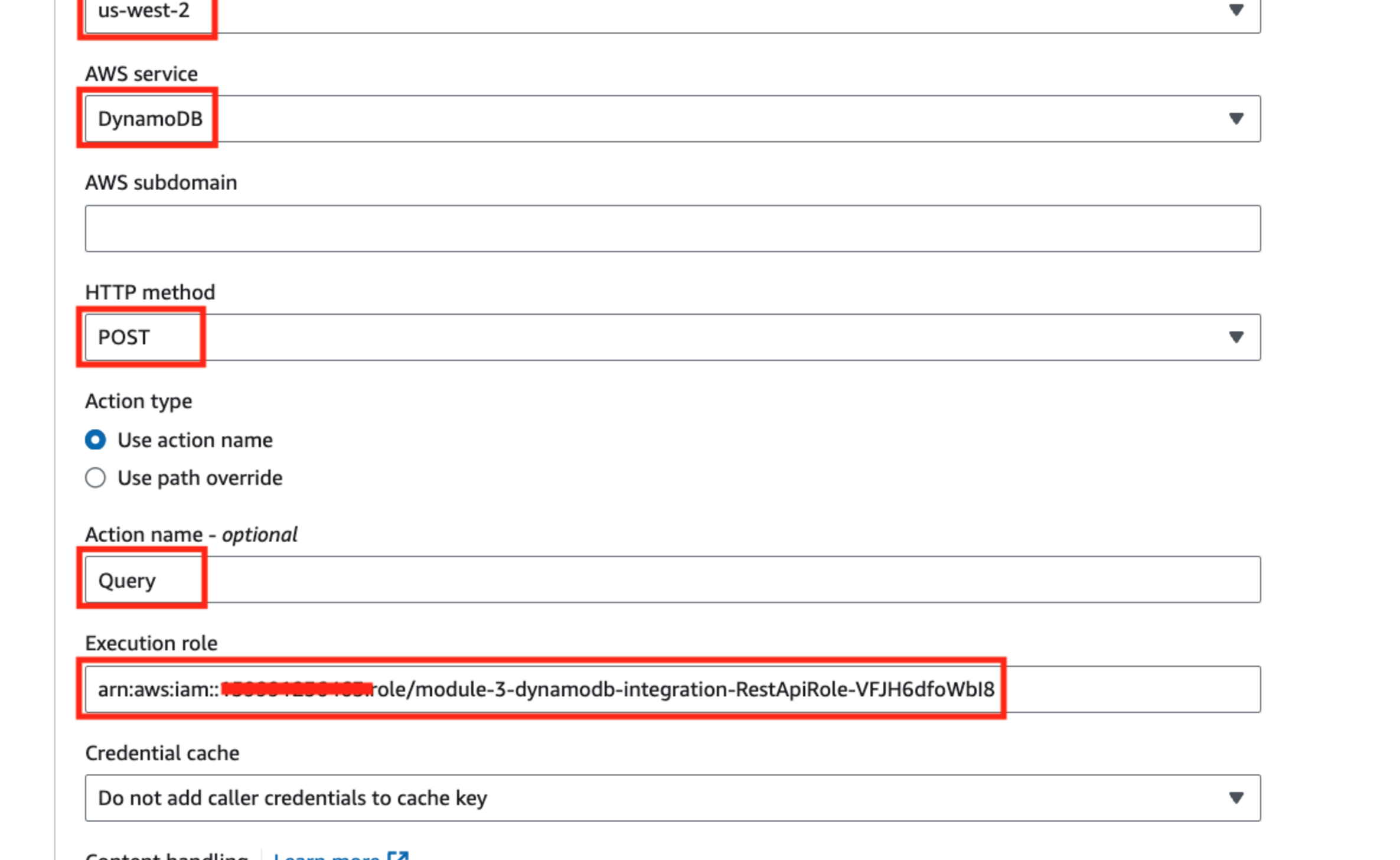
Test DynamoDB Integration using API Gateway console to READ an item

Now, let's test the read integration (*Query* api call).

1. Navigate to the API Gateway console and select the API that has been created. It will be named **module-3-dynamodb-integration**.
2. Select the GET method from the resources section to the left of the screen to bring up the API request flow window.



3. In the *Method Execution*, click in **Integration Request** and then **Edit**.



4. Review the **Integration Request** configuration.

AWS Region us-west-2

AWS service DynamoDB

AWS subdomain

HTTP method POST

Action type
☒ Use action name
☐ Use path override

Action name - optional Query

Execution role arn:aws:iam::123456789012:role/module-3-dynamodb-integration-RestApiRole-VFJH6dfoWbB8

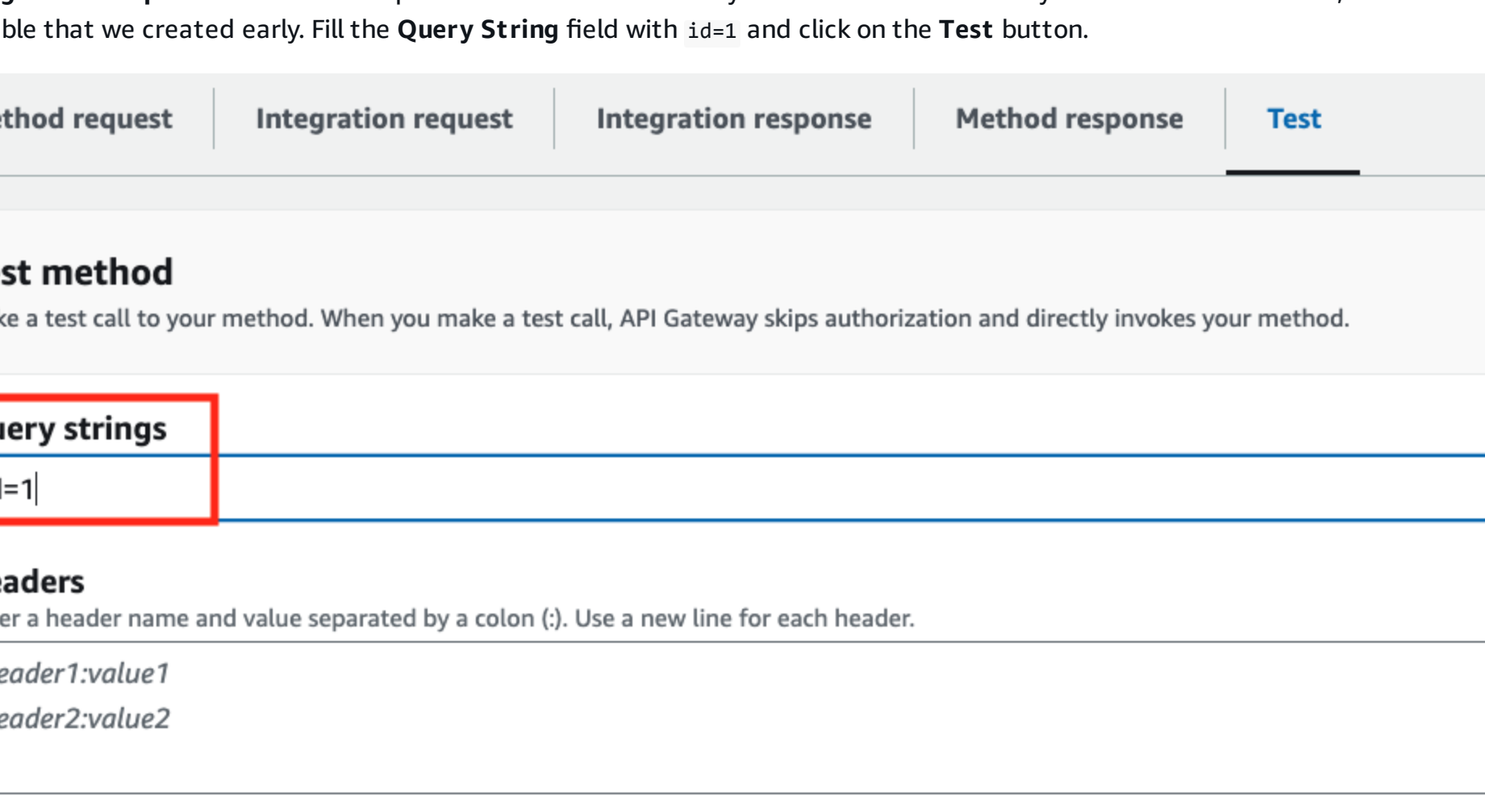
Credential cache
Do not add caller credentials to cache key

Content handling Learn more

5. Click on **Save**. In **Integration Request** tab and go down to the **Mapping Template** to see the input transformation request template:



6. Now, click on **Integration Response** tab and go down to the **Mapping Template** to see the output transformation request template:



7. Now, click on **Integration Response** tab. This will open the window that allows you to test invocations to your backend. In our case, we want to get a record from the DynamoDB table that we created early. Fill the **Query String** field with *id=1* and click on the **Test** button.

Method request **Integration request** **Integration response** **Method response** **Test**

Test method
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

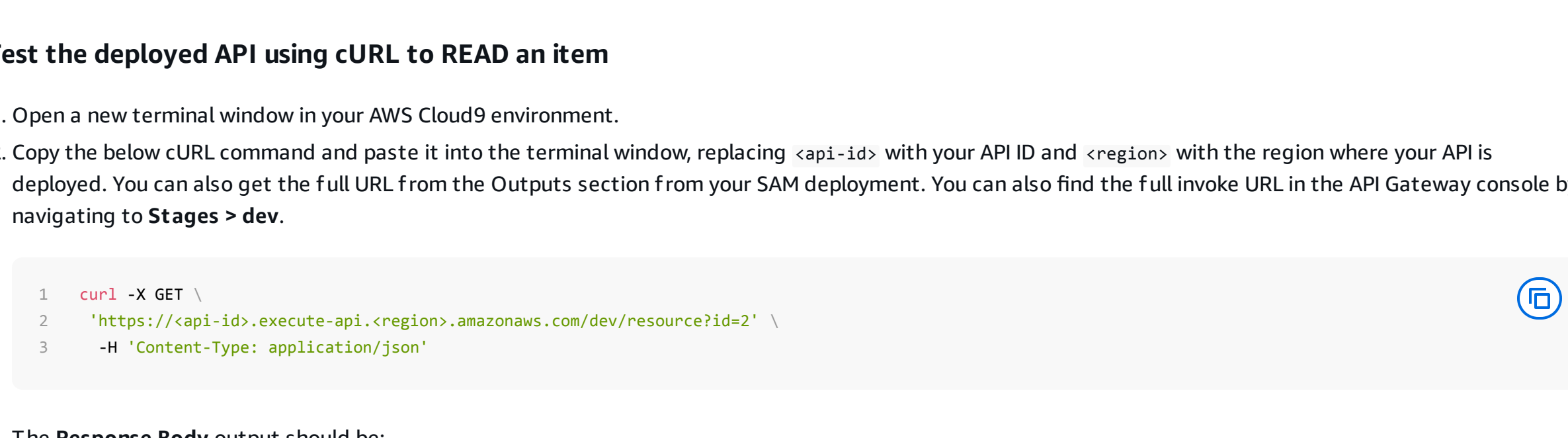
Query strings
id=1

Headers
Enter a header name and value separated by a colon (:). Use a new line for each header.
header1:value1
header2:value2

Client certificate
No client certificates have been generated.

Test

8. Scroll to the bottom of the page and hit the **Test** button. From this, you should receive a Status code of 200 meaning the record was successfully received to the DynamoDB table. Also, a json object that match with the mapping template of the response that we saw earlier as a **body** of the response



1. Open a new terminal window in your AWS Cloud9 environment.
2. Copy the below cURL command and paste it into the terminal window, replacing *<api-id>* with your API ID and *<region>* with the region where your API is deployed. You can also get the full URL from the Outputs section from your SAM deployment. You can also find the full invoke URL in the API Gateway console by navigating to **Stages** > **dev**.

```
1 curl -X GET \
2   https://<api-id>.execute-api.<region>.amazonaws.com/dev/resource?id=1 \
3   -H 'Content-Type: application/json'
```

3. The **Response Body** output should be:

```
1 {
2   "Names": [
3     {
4       "id": "1",
5       "name": "John",
6       "lastName": "Doe",
7       "email": "jd@example.com"
8     }
9   ]
10 }
```

Challenge
Create new **resource** and **method** in the API Gateway to read multiple items from the DynamoDB table in the same request.

Select your cookie preferences

We use essential cookies and similar tools that are necessary to provide our site and services. We use performance cookies to collect anonymous statistics, so we can understand how customers use our site and make improvements. Essential cookies cannot be deactivated, but you can choose "Customize" or "Decline" to decline performance cookies.

If you agree, AWS and approved third parties will also use cookies to provide useful site features, remember your preferences, and display relevant content, including relevant advertising. To accept or decline all non-essential cookies, choose "Accept" or "Decline". To make more detailed choices, choose "Customize."

Accept

Decline

Customize