Troubleshooting Gateway Integrations

In this lesson, you will learn how to troubleshoot gateway integrations using the AWS Web Console.

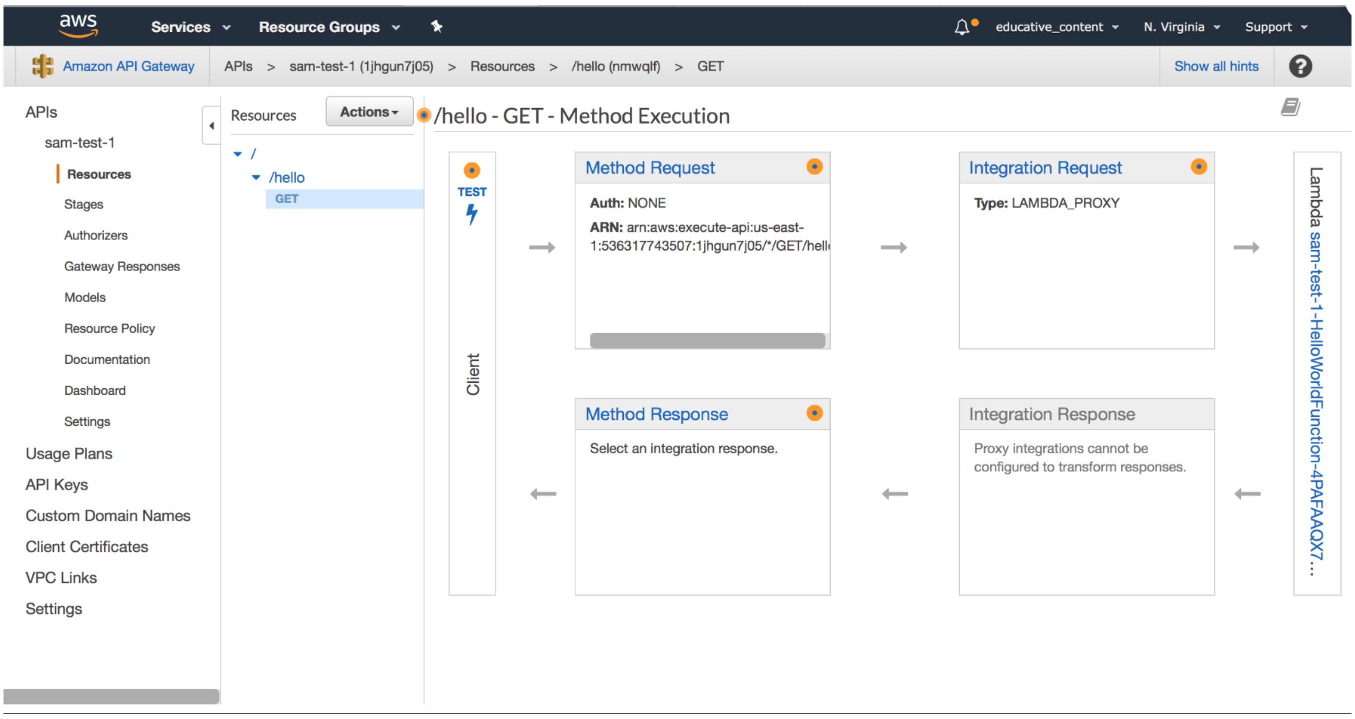
Reasons for a generic error

API Gateway masks unhandled errors (such as a missing resources), with a `403 Not Authorized` response. If anything goes wrong when talking to the back-end service, you'll only see that error instead of useful troubleshooting information. This is great from a security perspective, because it prevents sensitive data leaks, but it's a major pain for troubleshooting problems. Here are just some of the cases when you'll see this generic error:

- If the request does not even reach the Lambda function, for example, due to misconfigured authorisation or resource configuration problems.
- If the function blows up without returning a proper response.
- If the function returns a response that's not in the exact format that API Gateway expects.

Using API Gateway to troubleshoot errors

The *Resources* screen in the API Gateway Web Console is super useful for troubleshooting integration problems. When you get a 403 response and it's baffling you, navigate to the web console page for the API Gateway resource and click the *Test* link in the *Client* box (see the left side of the figure given below).



Auth: NONE

ARN: arn:aws:execute-api:us-east-1:536317743507:1jhgun7j05/*/GET/hell

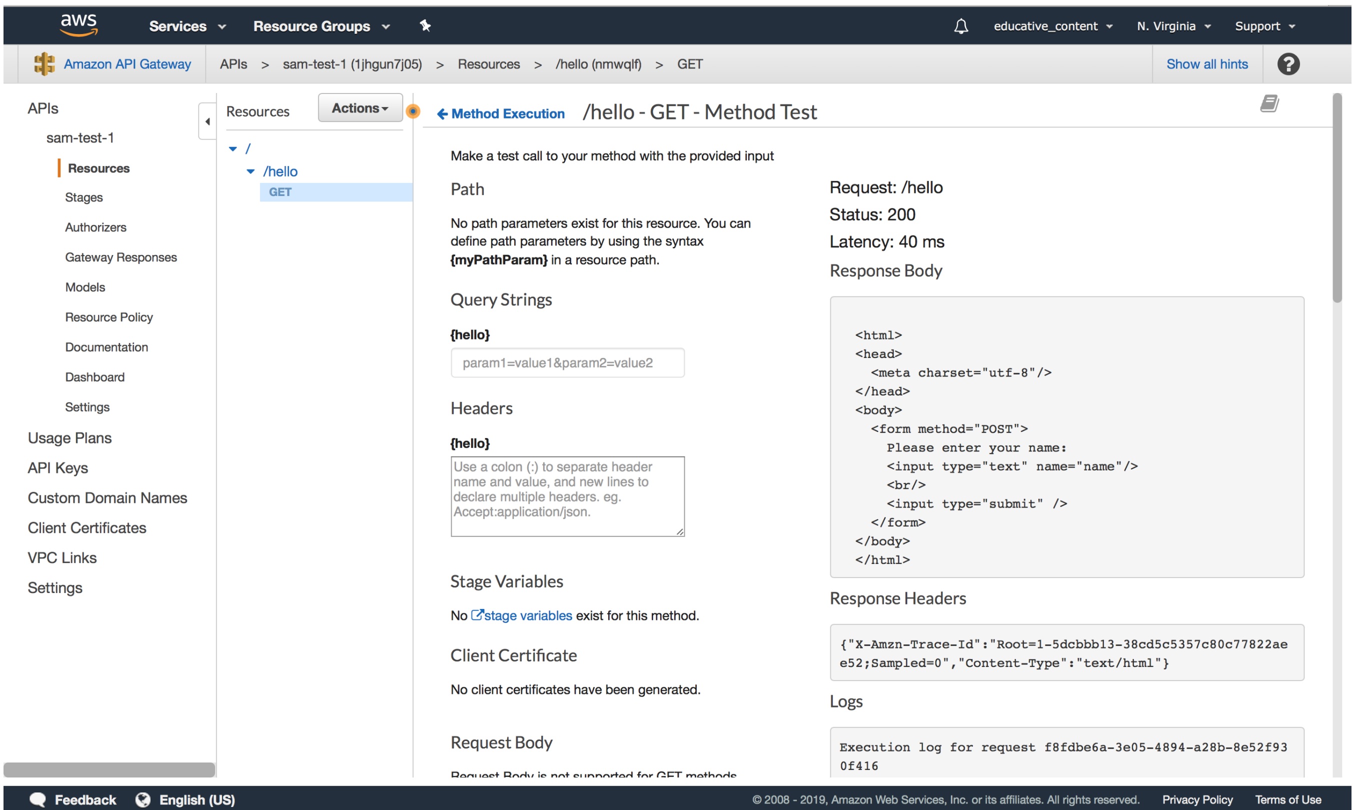
Type: LAMBDA_PROXY

Select an integration response.

Proxy integrations cannot be configured to transform responses.

API Gateway Web Console shows the details of a resource, including the authentication setup and the back end it talks to. It also enables us to send test requests easily.

A diagnostic test screen will open. You can enter request information, such as query strings and headers, then click the blue *Test* button towards the bottom of the screen. API Gateway will call the Lambda function simulating the HTTP request and then print out a bunch of diagnostic information, including all data sent to the back-end and received back (refer to the figure below). This will help you discover the real problem and find out how to fix it.



Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{hello}

param1=value1&param2=value2

Headers

{hello}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods

Request: /hello
Status: 200
Latency: 40 ms

Response Body

```
<html>
<head>
  <meta charset="utf-8"/>
</head>
<body>
  <form method="POST">
    Please enter your name:
    <input type="text" name="name"/>
    <br/>
    <input type="submit" />
  </form>
</body>
</html>
```

Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-5dcbbb13-38cd5c5357c80c77822ae
e52;Sampled=0","Content-Type":"text/html"}
```

Logs

```
Execution log for request f8fdbe6a-3e05-4894-a28b-8e52f93
0f416
```

API Gateway testing prints out detailed diagnostic information.

You can now move on to the next lesson where you will learn how to process request parameters.