

Introduction to Cloud Computing – CS360

Virtual Group Hackathon Exercise : Team - 2

In order for your API to invoke your Calc Lambda function, you'll need to have an API Gateway assumable IAM role. The role you create will need to have Lambda InvokeFunction permission. Otherwise, the API caller will receive a 500 Internal Server Error response.

To give the role this permission, we have used the following code

Code :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "*"
    }
  ]
}
```

Next we created a Lambda function using the Lambda console.

While creating a Lambda function we chose Node.js as supported runtime code.

Code :

```
console.log('Loading the Calc function');

exports.handler = function(event, context, callback) {
  console.log('Received event:', JSON.stringify(event, null, 2));
  if (event.a === undefined || event.b === undefined || event.op === undefined) {
    callback("400 Invalid Input");
  }

  var res = { };
```

```
res.a = Number(event.a);
res.b = Number(event.b);
res.op = event.op;

if (isNaN(event.a) || isNaN(event.b)) {
    callback("400 Invalid Operand");
}
switch(event.op)
{
    case "+":
    case "add":
        res.c = res.a + res.b;
        break;
    case "-":
    case "sub":
        res.c = res.a - res.b;
        break;
    case "*":
    case "mul":
        res.c = res.a * res.b;
        break;
    case "/":
    case "div":

        res.c = res.b===0 ? NaN : Number(event.a) / Number(event.b);
        break;
    default:
        callback("400 Invalid Operator");
        break;
}
callback(null, res);
};
```

Then we tested it by creating an test event. Which worked finely.

To create an API for the Calc Lambda function you just created. In subsequent sections, we add resources and methods to it.

Integration 1: Create a GET method with query parameters to call the Lambda function

OUTPUTS :

The image displays two screenshots of the AWS API Gateway console, illustrating the configuration and testing of a GET method for a Lambda function.

Top Screenshot: Shows the configuration for a GET method. The path is `/calc`, and the query string is `operand1=2&operand2=2&operator=+`. The response body is a JSON object: `{ "a": 2, "b": 2, "op": "+", "c": 4 }`. The status is 200, and the latency is 69 ms.

Bottom Screenshot: Shows the configuration for a GET method. The path is `/calc`, and the query string is `operand1=2&operand2=2&operator=-`. The response body is a JSON object: `{ "a": 2, "b": 2, "op": "-", "c": 0 }`. The status is 200, and the latency is 59 ms.

Integration 2: Create a POST method with a JSON payload to call the Lambda function

OUTPUTS :

The screenshot displays the AWS API Gateway console for a resource named `/calc`. The left sidebar shows the navigation menu with options like Custom Domain Names, VPC Links, and Resources. The main panel is divided into three sections: configuration, test results, and logs.

Configuration:

- Path:** `/calc`. Note: No path parameters exist for this resource.
- Query Strings:** `{calc} param1=value1¶m2=value2`
- Headers:** `{calc}`. Note: 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.
- Request Body:** A JSON payload is shown in a text area:

```
1 {
2   "a": 1,
3   "b": 2,
4   "op": "+"
5 }
6
7
```

Test Results:

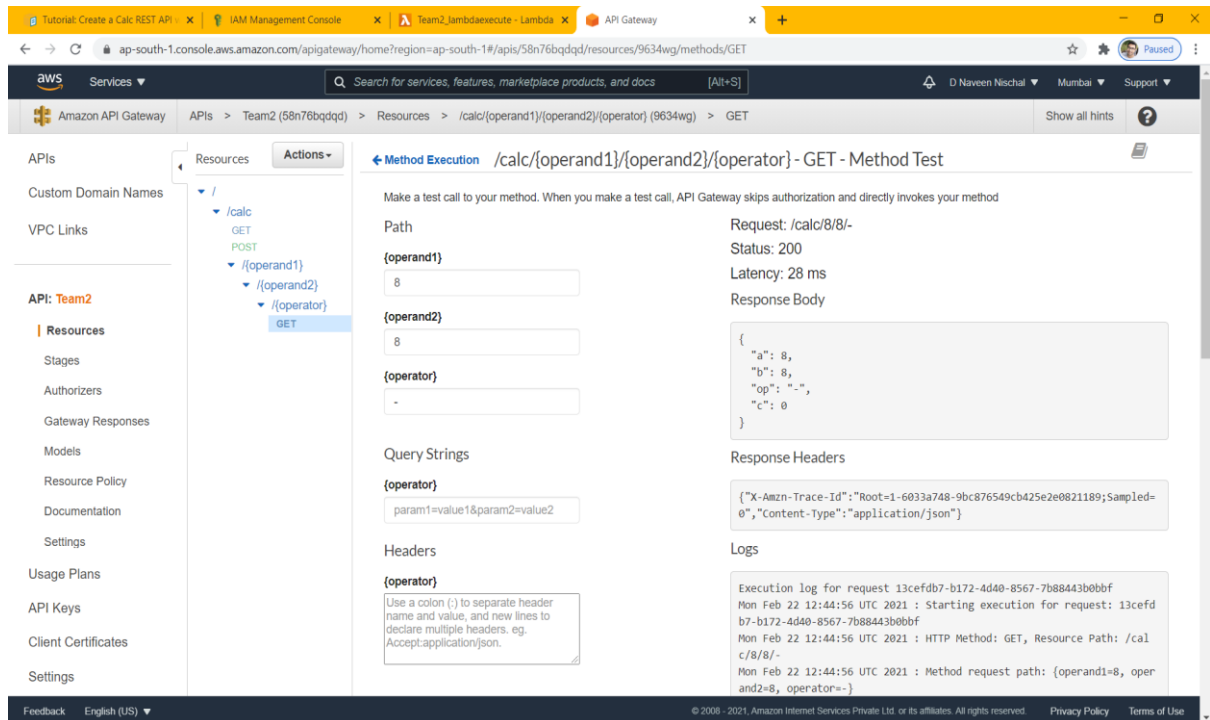
- Request:** `/calc`
- Status:** 200
- Latency:** 20 ms
- Response Body:** A JSON object:

```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```
- Response Headers:** `{"X-Amzn-Trace-Id": "Root=1-6033a7fe-65899e34685ca195c32ba9a6;Sampled=0", "Content-Type": "application/json"}`
- Logs:** An execution log for request `374667d7-66bd-41ea-9395-52fca5307001` is shown, detailing the request path, query string, headers, and body.

The bottom screenshot shows a similar view but with a different request body and response body, indicating a successful test run.

Integration 3: Create a GET method with path parameters to call the Lambda function

OUTPUTS :



Observation and Conclusion :

Based on above outputs, we learned how to create an serverless computing through API and Lambda integration. And also we have tried this by two types of integration : 1. Lambda Integration

2. AWS Service Integration

And also we tried with both Non – Proxy integration and Proxy integration as we faced “Internal Server Error” problem.

Thank You

BY TEAM - 2

Y SANTHI SWARUP(18BEC051)

N NIKHIL KUMAR(18BEC031)

D NAVEEN (18BEC011)

V HARSHA(18BCS0118)

P BHANU (18BEC035)

