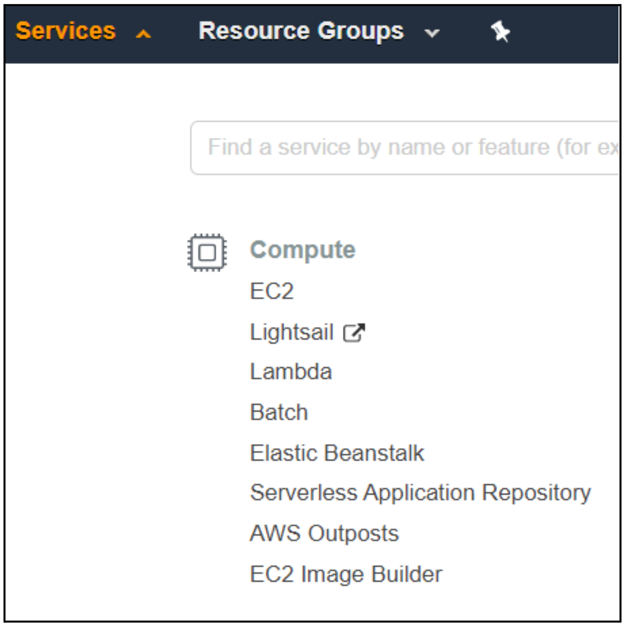
snhu

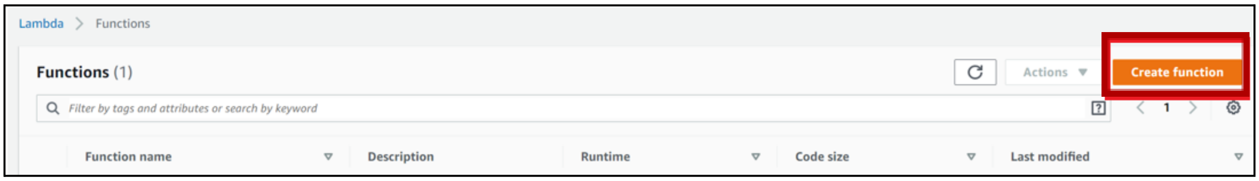
CS 470 Module Four Assignment One Guide

Part One – Creating and Testing a Lambda

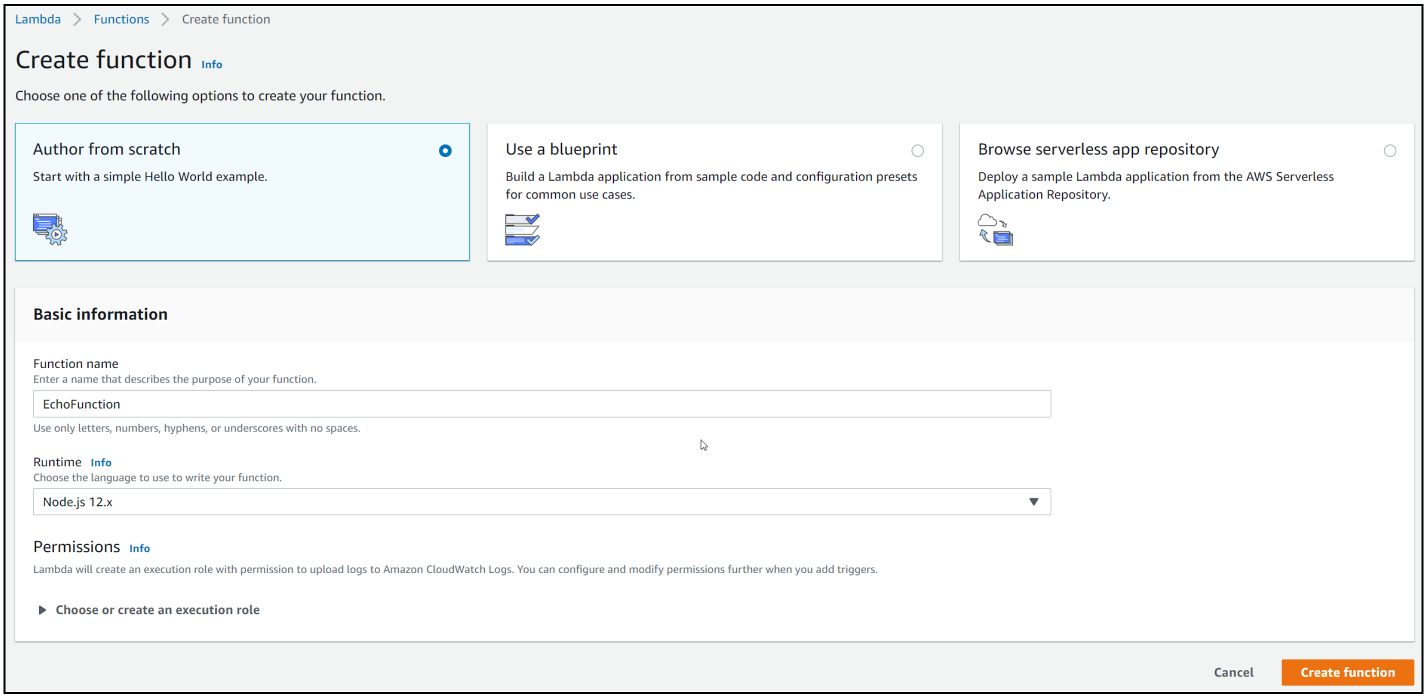
1. Navigate to **AWS Lambda** by typing “Lambda” into the console search bar or selecting **Lambda** under **Compute**.



2. Click the orange **Create function** button in the upper-right corner.



3. Make sure **Author From Scratch** is selected.
4. Enter the function name “EchoFunction”.
5. Set the runtime to “Node.js 12.x”.
6. Leave the permissions option set to **Create a new role with basic Lambda permissions.**
7. Click the orange **Create Function** button in the lower-right corner.



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

● Author from scratch
Start with a simple Hello World example.

○ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

○ Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

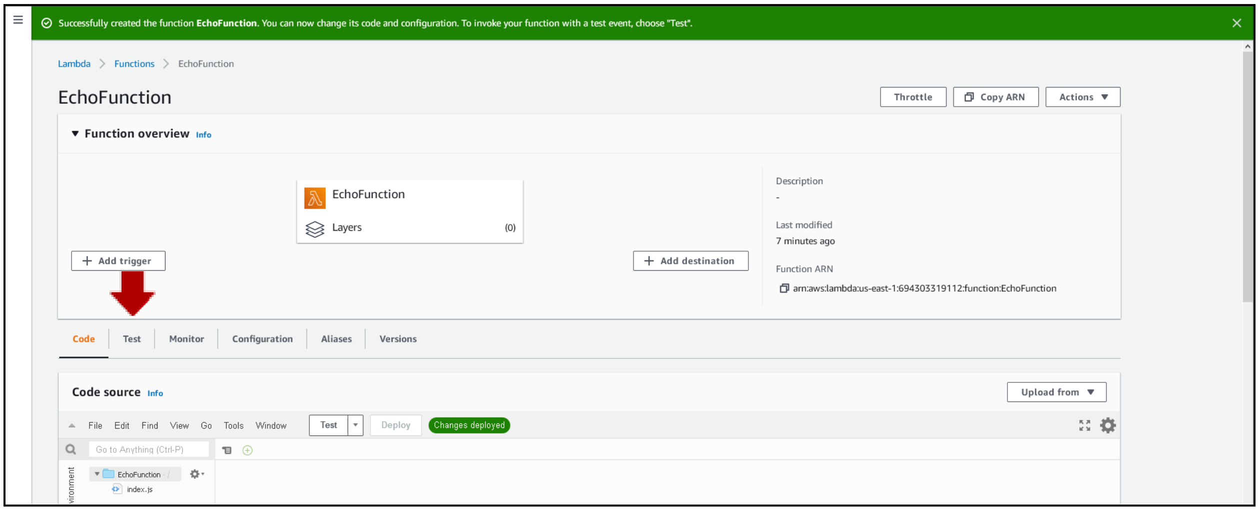
Function name
Enter a name that describes the purpose of your function.
EchoFunction
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Node.js 12.x

Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▶ Choose or create an execution role

8. Congratulations! You have created your first serverless compute function – a bouncing baby Lambda! You should now see a screen like this:



Successfully created the function **EchoFunction**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > EchoFunction

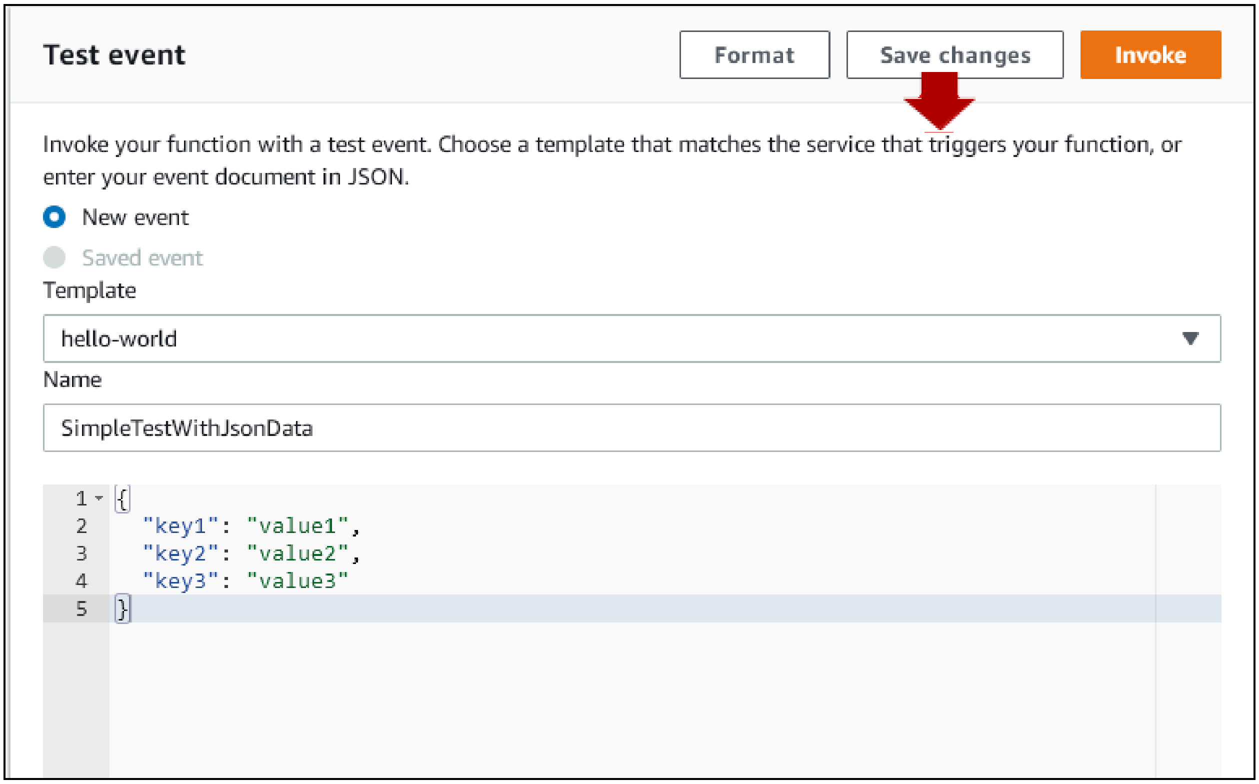
EchoFunction

▼ Function overview Info

EchoFunction
Layers (0)

Description
-
Last modified
7 minutes ago
Function ARN
arn:aws:lambda:us-east-1:694303319112:function:EchoFunction

9. Click the **Test** tab in the left-hand side of the screen (see the red arrow).
10. In the **Test event** box, enter an event name of "SimpleTestWithJsonData".
11. Leave everything else the same and click the **Save changes** Button in the upper-right part of the screen.



Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON.

● New event

○ Saved event

Template

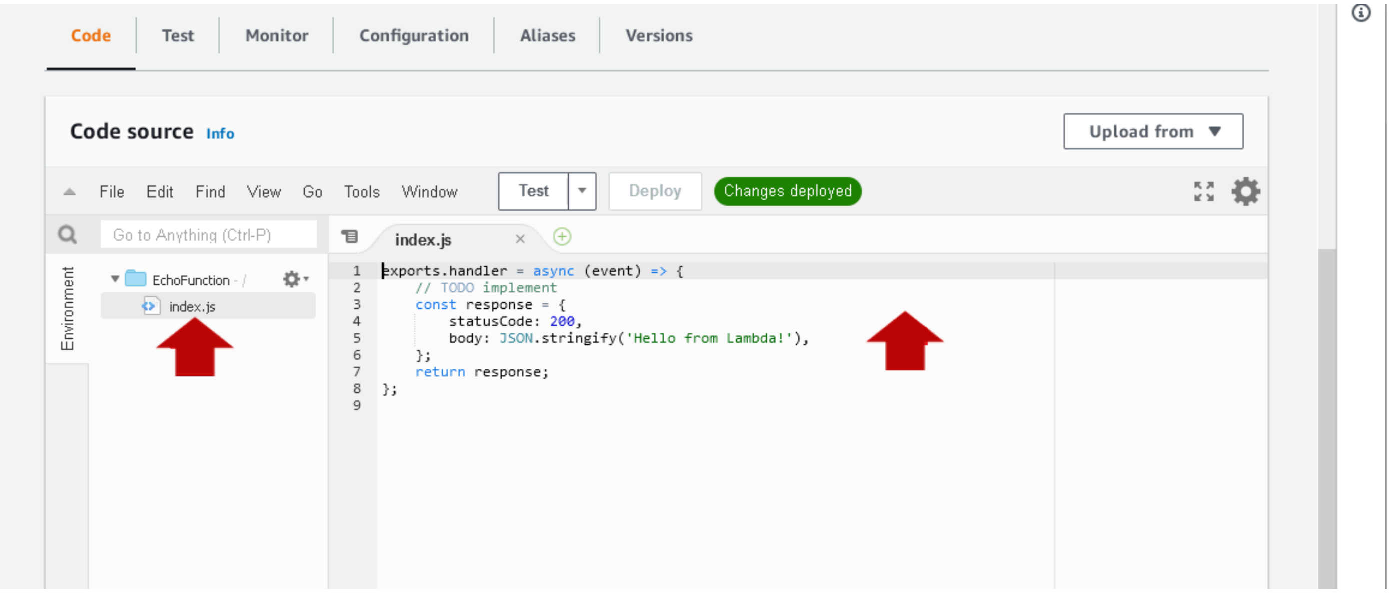
hello-world

Name

SimpleTestWithJsonData

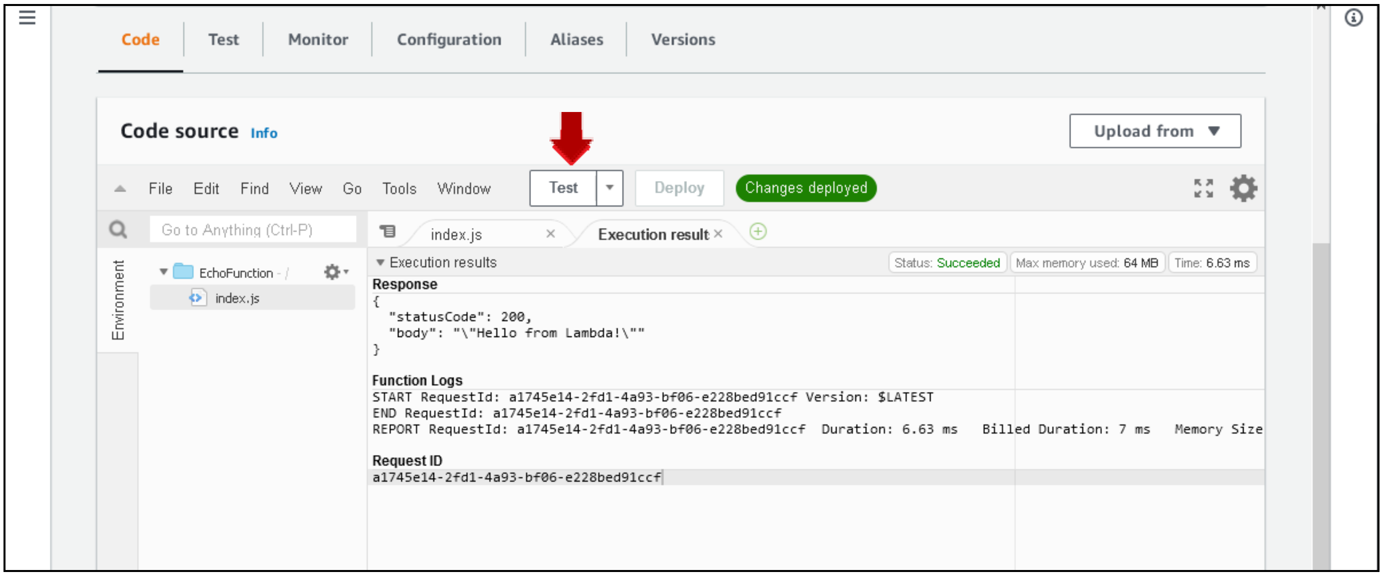
```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

12. You will still be looking at the Test tab in the Lambda console.
13. Now click on the Code tab and you will see the inline Lambda code editor in the box titled **Code source**. Double-left-click on the index.js symbol on the left side of the screen, and the code for the function will appear.



```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

14. We will review the code in a moment, but for now, click the **Test** button on the editor.
15. Congratulations! You have now run your first Lambda! Your screen should look like this:



Response

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

```
START RequestId: a1745e14-2fd1-4a93-bf06-e228bed91ccf Version: $LATEST
END RequestId: a1745e14-2fd1-4a93-bf06-e228bed91ccf
REPORT RequestId: a1745e14-2fd1-4a93-bf06-e228bed91ccf  Duration: 6.63 ms   Billed Duration: 7 ms   Memory Size
```

Request ID

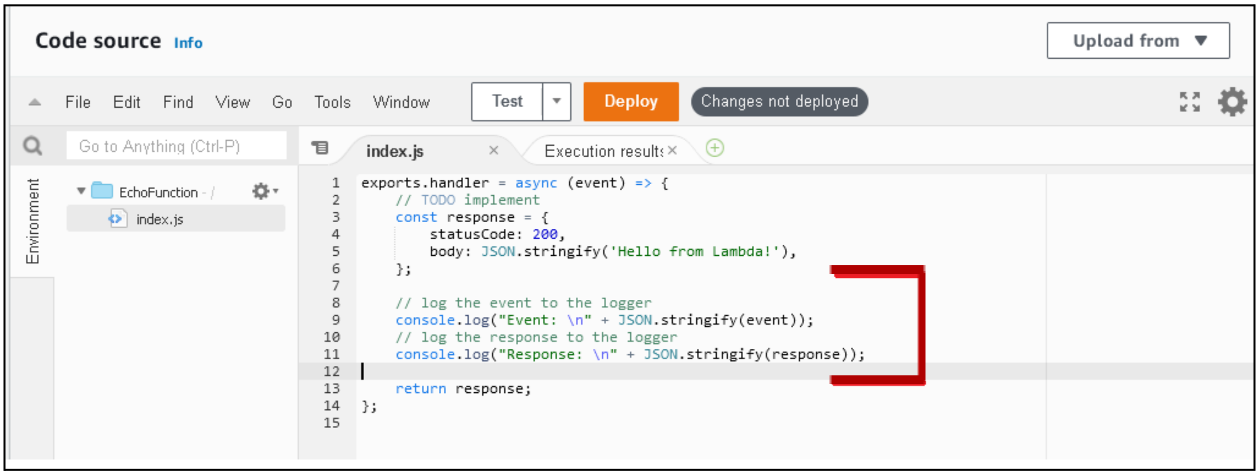
```
a1745e14-2fd1-4a93-bf06-e228bed91ccf
```

Part Two – Creating an Echo Function

1. You are going to modify the code provided by AWS to perform a simple echo function.
2. Add the following code to index.js after the creation of the response object and before the return:

```
// log the event to the logger
console.log("Event: \n" + JSON.stringify(event));
// log the response to the logger
console.log("Response: \n" + JSON.stringify(response));
```

3. Your code should look like this:

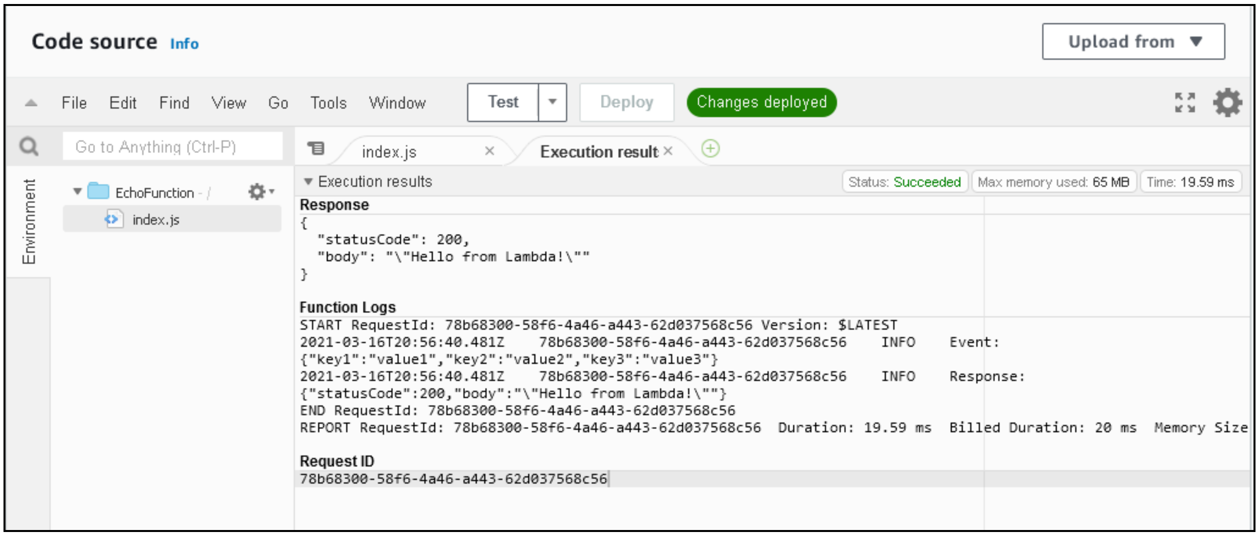


```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };

    // log the event to the logger
    console.log("Event: \n" + JSON.stringify(event));
    // log the response to the logger
    console.log("Response: \n" + JSON.stringify(response));

    return response;
};
```

4. Click **Deploy**.
5. Click **Test** again.
6. Your execution results will now look like this:



Response

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

```
START RequestId: 78b68300-58f6-4a46-a443-62d037568c56 Version: $LATEST
2021-03-16T20:56:40.481Z    78b68300-58f6-4a46-a443-62d037568c56    INFO    Event:
{"key1":"value1","key2":"value2","key3":"value3"}
2021-03-16T20:56:40.481Z    78b68300-58f6-4a46-a443-62d037568c56    INFO    Response:
{"statusCode":200,"body":"\"Hello from Lambda!\""}
END RequestId: 78b68300-58f6-4a46-a443-62d037568c56
REPORT RequestId: 78b68300-58f6-4a46-a443-62d037568c56  Duration: 19.59 ms  Billed Duration: 20 ms  Memory Size
```

Request ID

```
78b68300-58f6-4a46-a443-62d037568c56
```

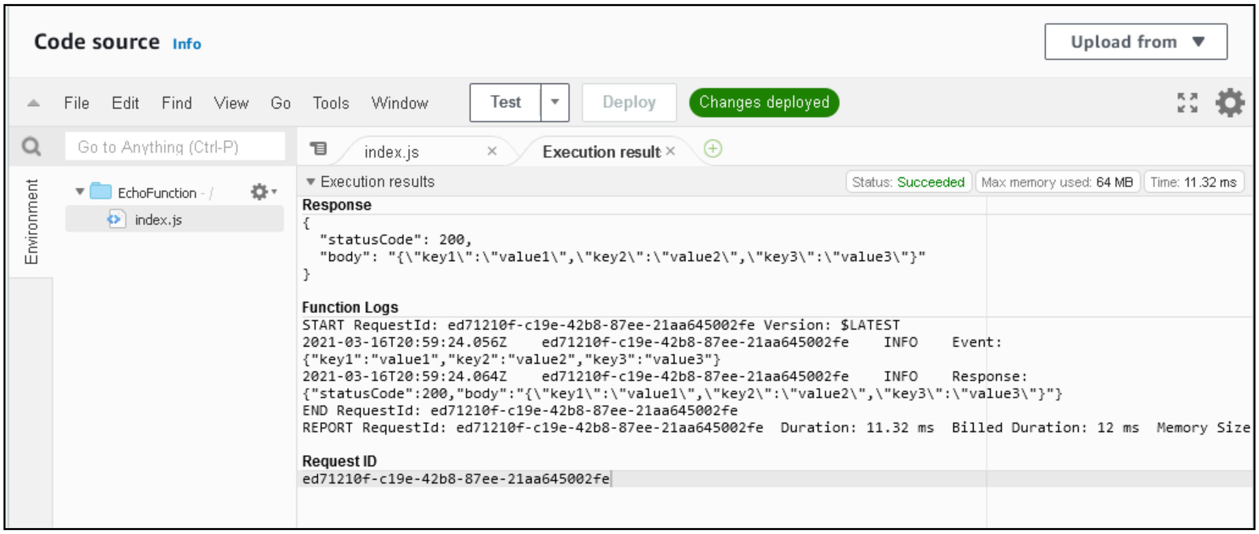
7. The values you passed in through the test event “SimpleTestWithJsonData” are now logged as the event object, and your response is the response object being created.
8. We want the response to echo what was passed in, so modify the code to do so. This will change the following code from:

```
    // create some text to send back to the client
    body: JSON.stringify('Hello from Lambda!'),
```

To:

```
  // create some text to send back to the client
  body: JSON.stringify(event),
```

9. Make sure to click **Save**.
10. Now click **Test** again and your execution results should be the following:



Response

```
{
  "statusCode": 200,
  "body": "{\"key1\":\"value1\",\"key2\":\"value2\",\"key3\":\"value3\"}"
}
```

Function Logs

```
START RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe Version: $LATEST
2021-03-16T20:59:24.056Z    ed71210f-c19e-42b8-87ee-21aa645002fe    INFO    Event:
{"key1":"value1","key2":"value2","key3":"value3"}
2021-03-16T20:59:24.064Z    ed71210f-c19e-42b8-87ee-21aa645002fe    INFO    Response:
{"statusCode":200,"body":"{\"key1\":\"value1\",\"key2\":\"value2\",\"key3\":\"value3\"}"}
END RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe
REPORT RequestId: ed71210f-c19e-42b8-87ee-21aa645002fe  Duration: 11.32 ms  Billed Duration: 12 ms  Memory Size
```

Request ID

```
ed71210f-c19e-42b8-87ee-21aa645002fe
```

Part Three – Enhancing the Echo Function

1. You can now echo your results, at least for that JSON data that was passed in. Let’s do something different.
2. First, create a new test class by clicking the down arrow next to **Test** and selecting **Configure test event.**
3. Select **Create New Test Event** and name it “EchoWithQuery”, with the following JSON body:

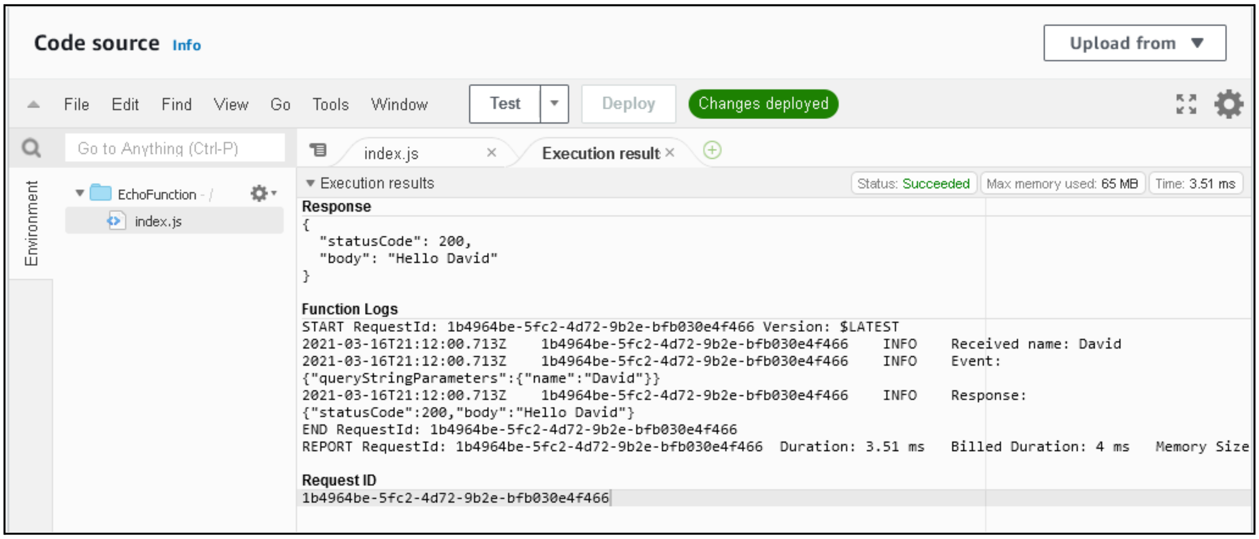
```
{
 "queryStringParameters": {
  "name": "David"
 }
}
```

4. Click **Create**.
5. Now replace the code from the TODO to the log statements with the following:

```
var name = 'unknown';
if (event.queryStringParameters && event.queryStringParameters.name) {
   console.log("Received name: " + event.queryStringParameters.name);
   name = event.queryStringParameters.name;
}

const response = {
   statusCode: 200,
   body: "Hello " + name,
};
```

6. Click **Deploy**.
7. Run your EchoWithQuery test and you should see the following execution results:



Status: Succeeded | Max memory used: 65 MB | Time: 3.51 ms

Response

```
{
  "statusCode": 200,
  "body": "Hello David"
}
```

Function Logs

```
START RequestId: 1b4964be-5fc2-4d72-9b2e-bfb030e4f466 Version: $LATEST
2021-03-16T21:12:00.713Z	1b4964be-5fc2-4d72-9b2e-bfb030e4f466	INFO	Received name: David
2021-03-16T21:12:00.713Z	1b4964be-5fc2-4d72-9b2e-bfb030e4f466	INFO	Event:
{"queryStringParameters":{"name":"David"}}
2021-03-16T21:12:00.713Z	1b4964be-5fc2-4d72-9b2e-bfb030e4f466	INFO	Response:
{"statusCode":200,"body":"Hello David"}
END RequestId: 1b4964be-5fc2-4d72-9b2e-bfb030e4f466
REPORT RequestId: 1b4964be-5fc2-4d72-9b2e-bfb030e4f466	Duration: 3.51 ms	Billed Duration: 4 ms	Memory Size
```

Request ID

```
1b4964be-5fc2-4d72-9b2e-bfb030e4f466
```