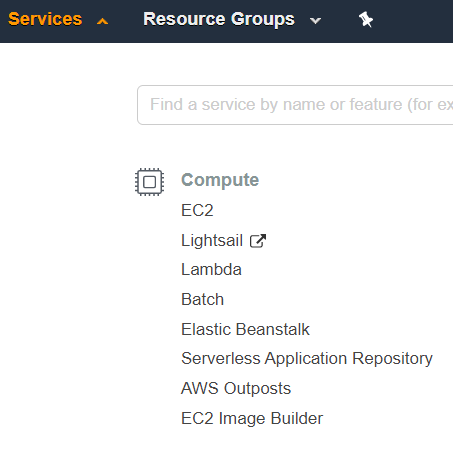
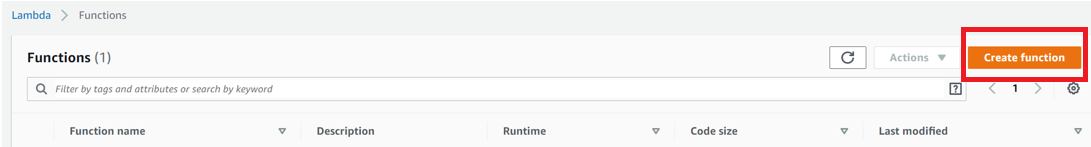
# CS 470 Module Four Assignment One Guide

## Part One: Creating and Testing a Lambda

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



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



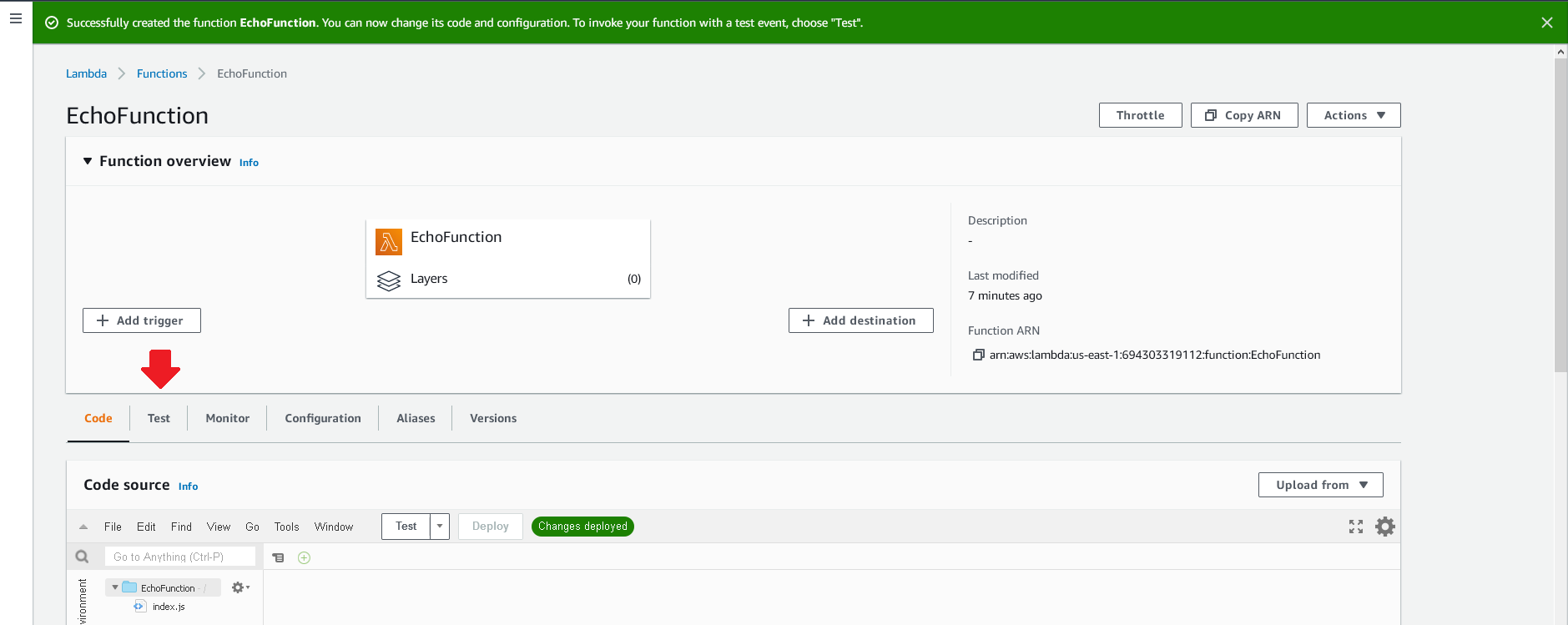
1. Make certain **Author From Scratch** is selected.
2. Enter the function name “EchoFunction”.
3. Set the runtime to **Node.js 20.x**.
4. Leave the permissions option set to **Create a new role with basic Lambda permissions.**
5. Click the orange **Create Function** button in the lower-right corner.

**Note:** If you get an error about not being authorized to perform jam:CreateRole while creating a Lambda Function, complete the following steps:

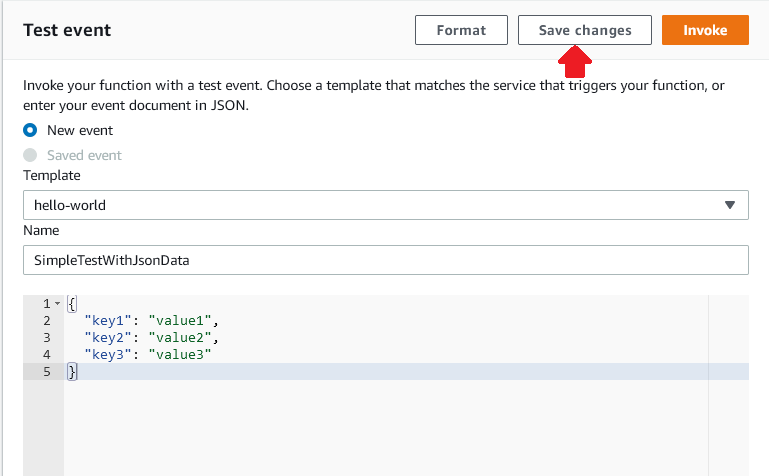
* 1. Click C**hange default execution role**.
  2. Click **Use existing role**”.
  3. Choose **LabRole** from the drop-down menu.
  4. Click **Create Function**.

Basic information section. The function name entered is EchoFunction. The runtime entered is Node.js 20.x. The architecture choices are x86_64 and arm64. X86_64 is selected. Under the Permissions section, the text reads “By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.” The Change default execution role and Advanced settings sections are collapsed. 

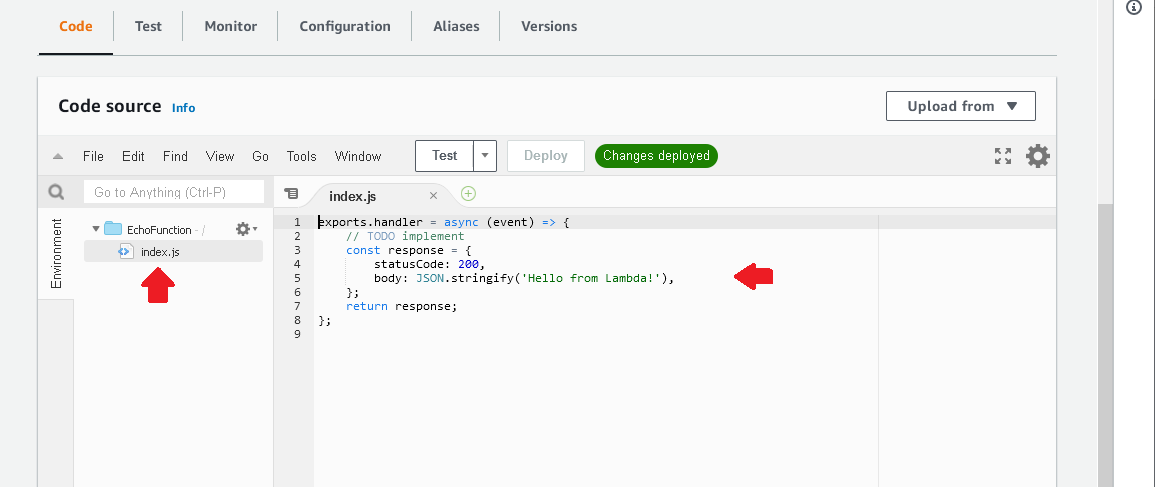

1. Congratulations! You have created your first serverless compute function: a bouncing baby Lambda! You should now see a screen like the screenshot below.



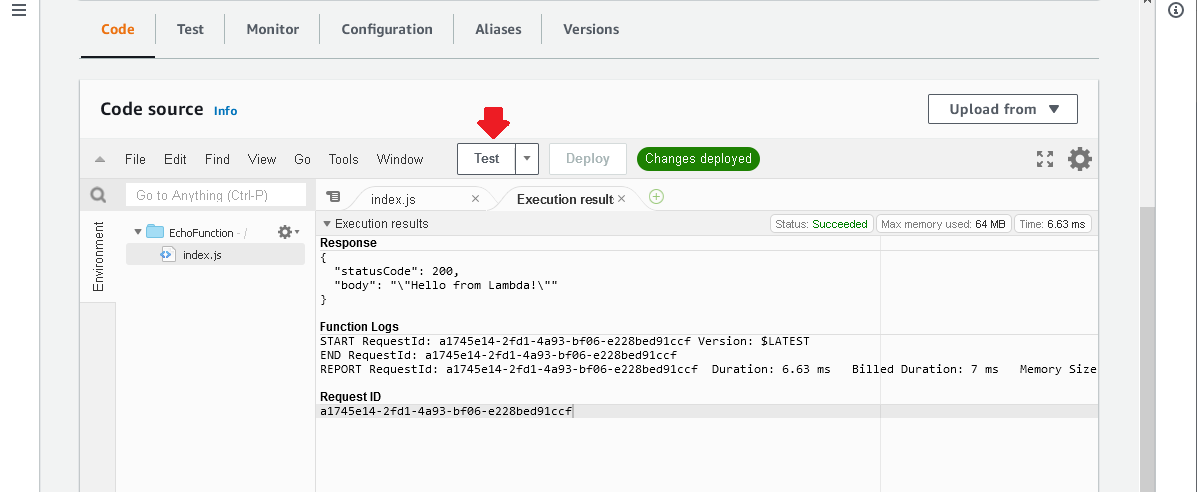
1. Click the **Test** tab on the left side of the screen where the red arrow is indicating in the image above.
2. In the **Test event** window, enter “SimpleTestWithJsonData” in the Name box.
3. Leave everything else the same and click the **Save changes** button in the upper-right part of the screen.



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



1. You will review the code in a moment. For now, click the **Test** button in the editor.
2. Congratulations! You have run your first Lambda. Your screen should look like this:



## Part Two: Creating an Echo Function

1. You will modify the code provided by AWS to perform a simple echo function.
2. Add the following code to index.js after the response object is created and before the return:

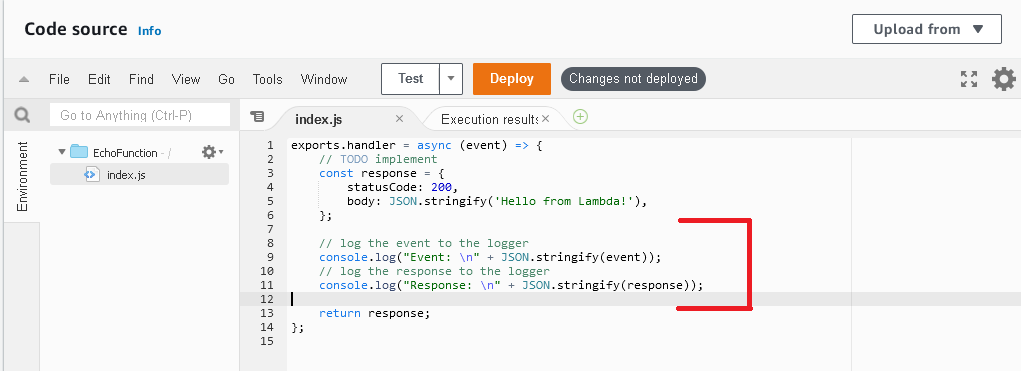
    // log the event to the logger

    console.log("Event: \n" + JSON.stringify(event));

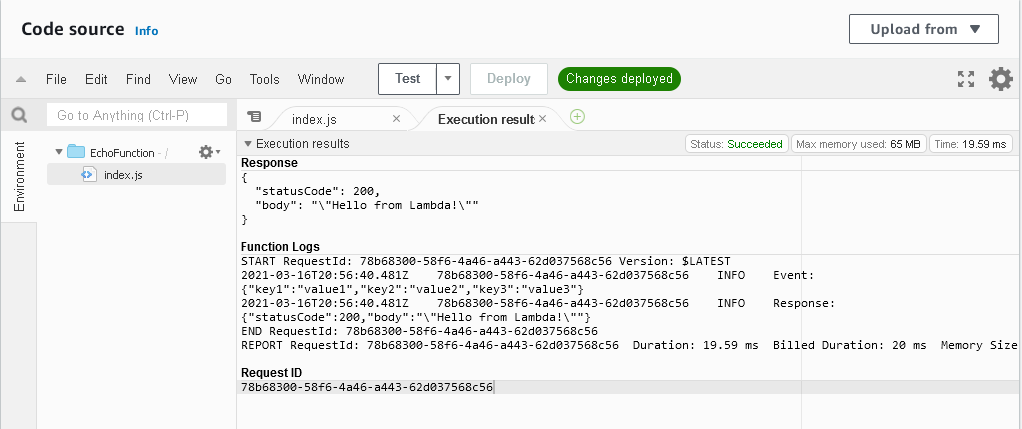
    // log the response to the logger

    console.log("Response: \n" + JSON.stringify(response));

1. Your code should look like this:



1. Click **Deploy**.
2. Click **Test** again.
3. Your execution results will now look like this:



1. The values you passed in through the **SimpleTestWithJsonData** test event are now logged as the event object. Your response is the response object being created.
2. Modify the code so the response echoes what was passed in. This modification will change the following code from this:

        // create some text to send back to the client

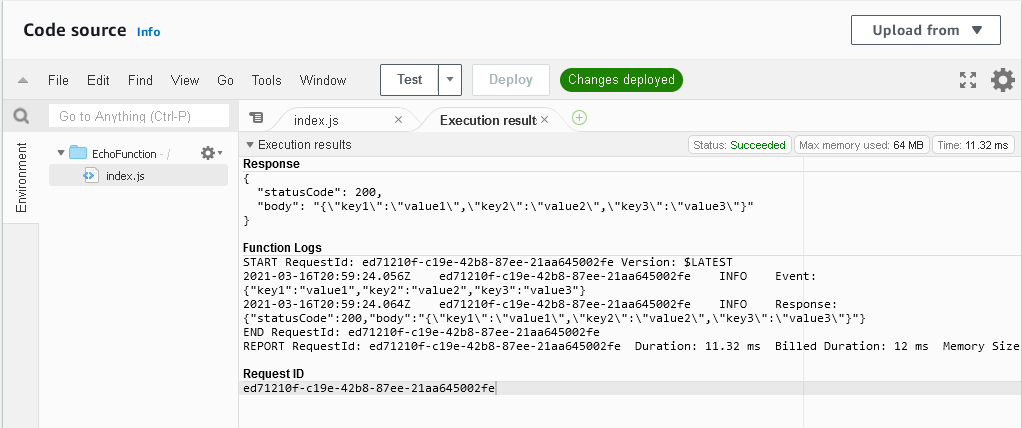
        body: JSON.stringify('Hello from Lambda!'),

to this:

      // create some text to send back to the client

      body: JSON.stringify(event),

1. Make certain to click **Deploy** when done.
2. Click **Test** again. Your execution results should look like this:



## Part Three: Enhancing the Echo Function

1. You can now echo your results, at least for the JSON data that was passed in. Now, you’ll 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"

  }

}

1. Click **Create**.
2. Now, replace the code from the TODO to the log statements with the following code:

  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

  };

1. Click **Deploy**.
2. Run your EchoWithQuery test. You should see the following execution results:

