In this tutorial, you do the following:

Create a Lambda function with a simple Hello World functionality. We will then look at Cloudwatch and add a feature to send emails from Lambda to yourself via SNS.

We will add more things as we go on in this tutorial too.

[What are Lambda functions?](#_plxgvfdg6qo0)

[Create a Lambda function](#_b88apdpip744)

[Test the Lambda function](#_hbf8gh1wy698)

[Let’s ramp it up](#_jrqsa4ecepw7)

[Your chance to change the code anyway you like](#_9zl5k2944pqy)

[Let’s see Cloudwatch logs](#_rltlm38zzeuc)

[What is SNS?](#_2o634d6xe2nb)

[Let’s connect to SNS (Successful output)](#_5cjjbn1c4ec3)

[Let’s connect to SNS (Failed output)](#_sqpitiegouv)

### What are Lambda functions?

AWS Lambda is a compute service that lets you **run code without provisioning or managing servers**.

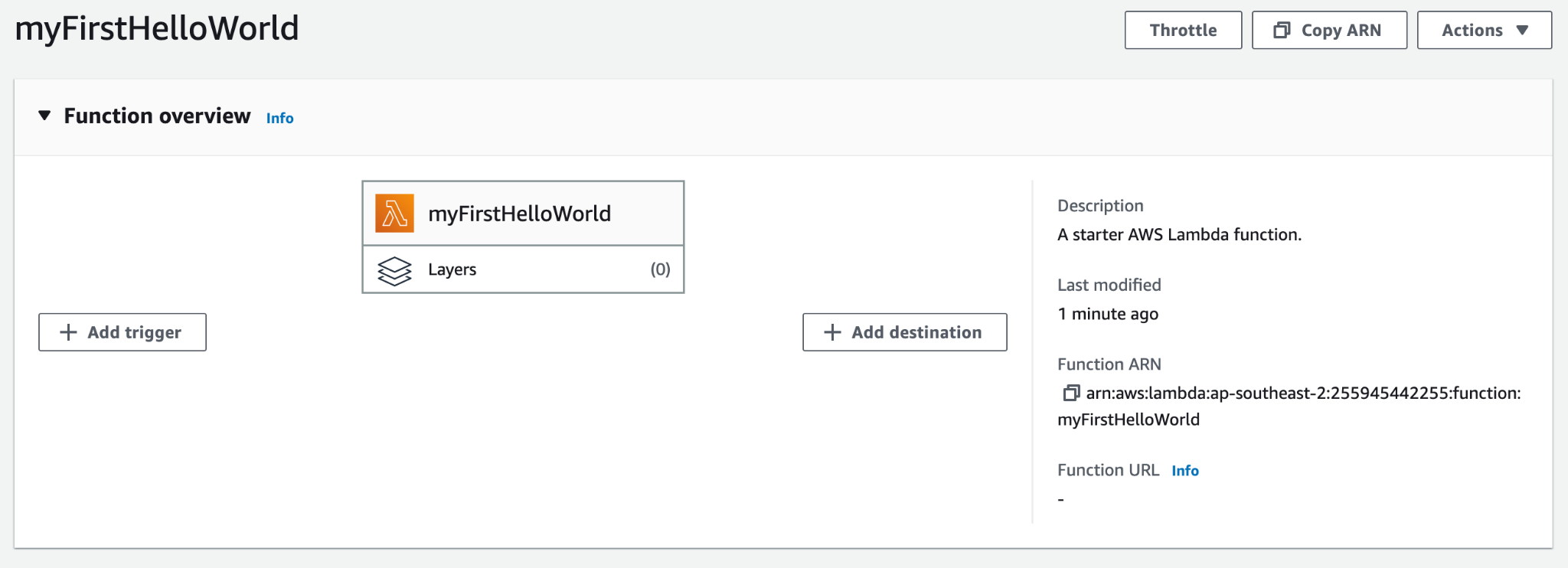
Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging.

With Lambda, you can **run code for virtually any type of application or backend service**. All you need to do is supply your code in one of the languages that Lambda supports.

You organize your code into Lambda functions. Lambda runs your function only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

### Create a Lambda function

* Sign in to the Amazon Web Services Management Console and open the Amazon Lambda console at https://console.amazonaws.cn/lambda/.
* Choose Create function.
* Choose Use a blueprint.
* Open the Select blueprint dropdown list and choose the Hello world function (python 3.7) option.
* Configure the following settings:
  + Function name – <name>-lambda-hello-world e.g. danny-lambda-hello-world
  + Execution role – Create a new role with basic Lambda permissions.
  + Under Lambda function code, take a look at the code and try to understand what it is doing. Discuss with your peers what you think this code does.
* Click Create function.

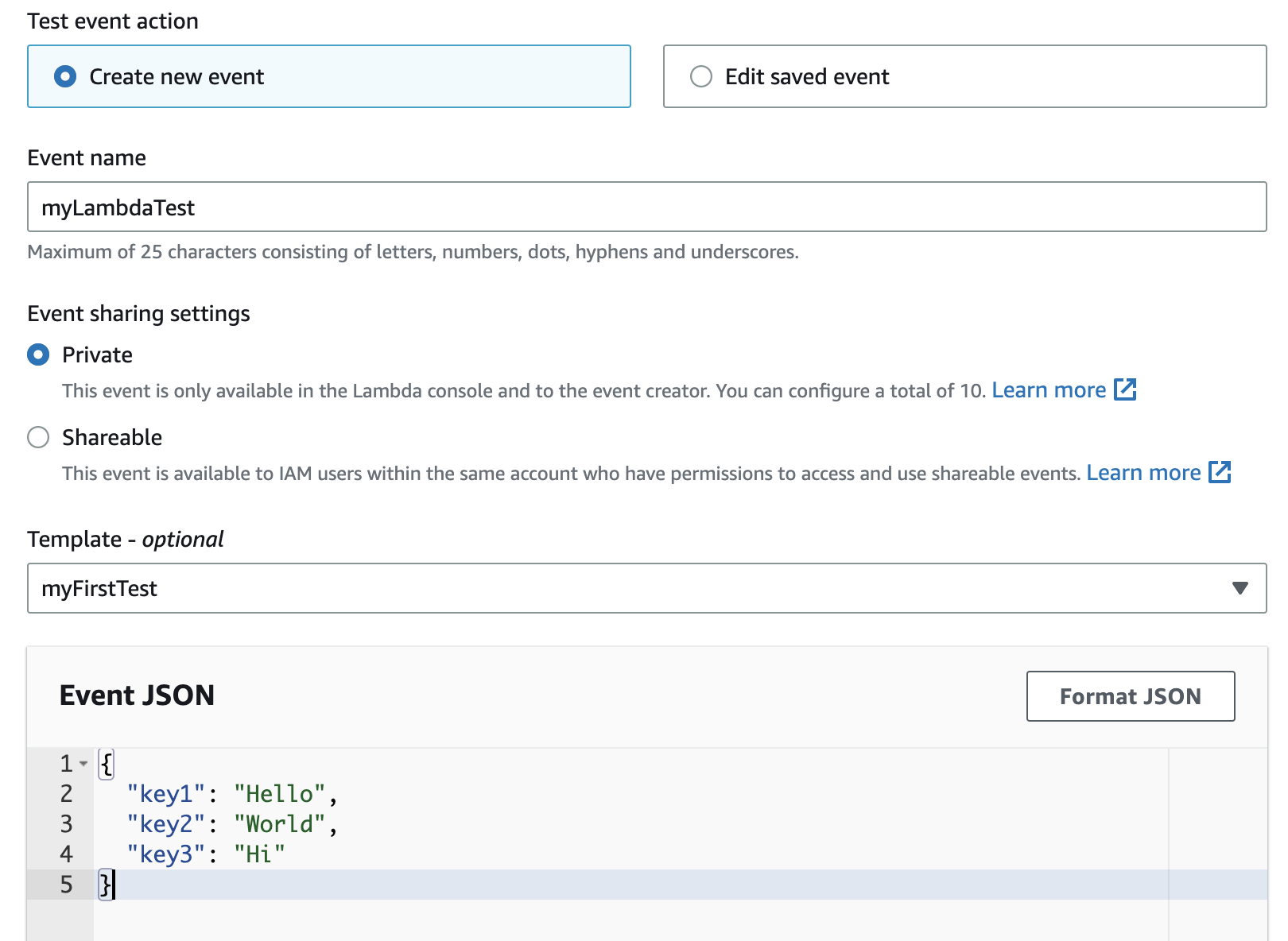


### Test the Lambda function

What are test functions?

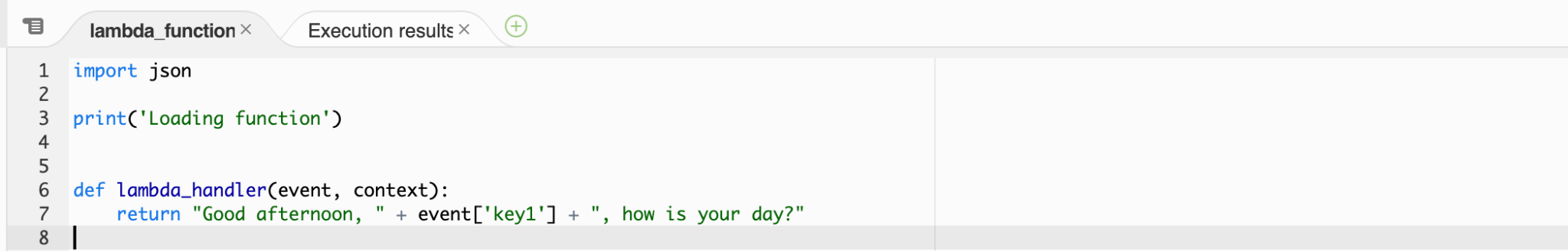
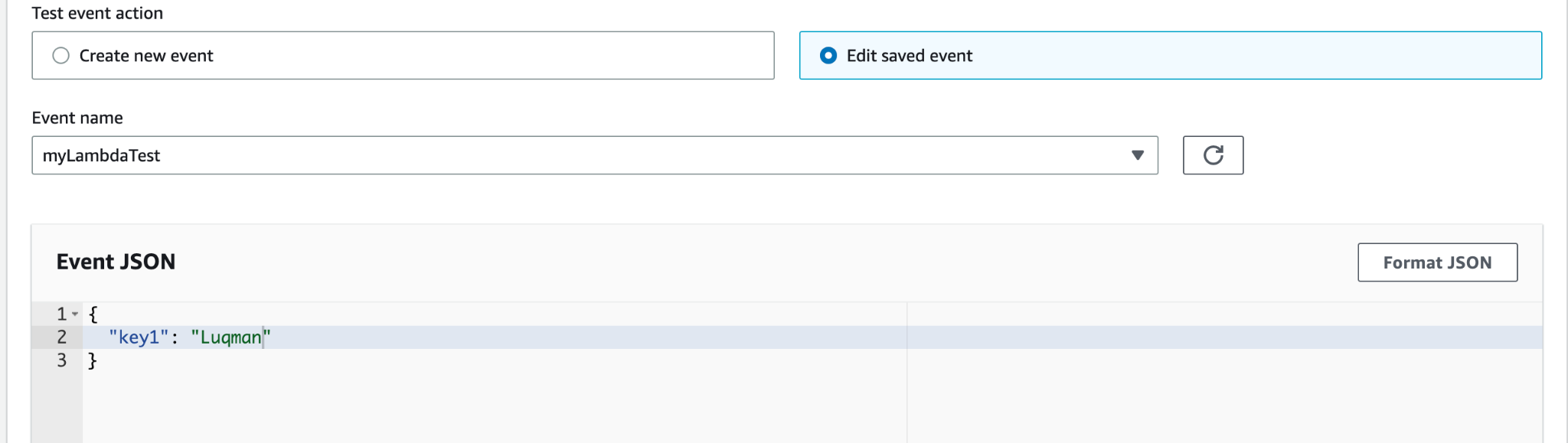
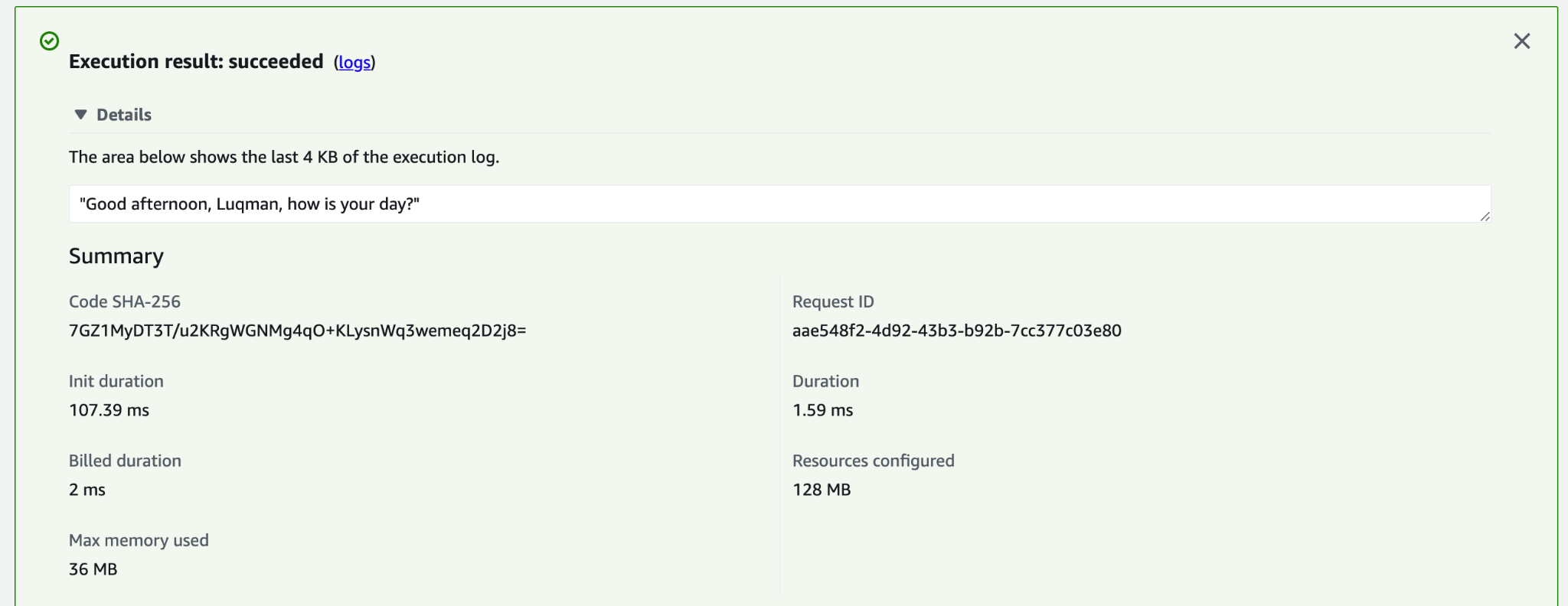
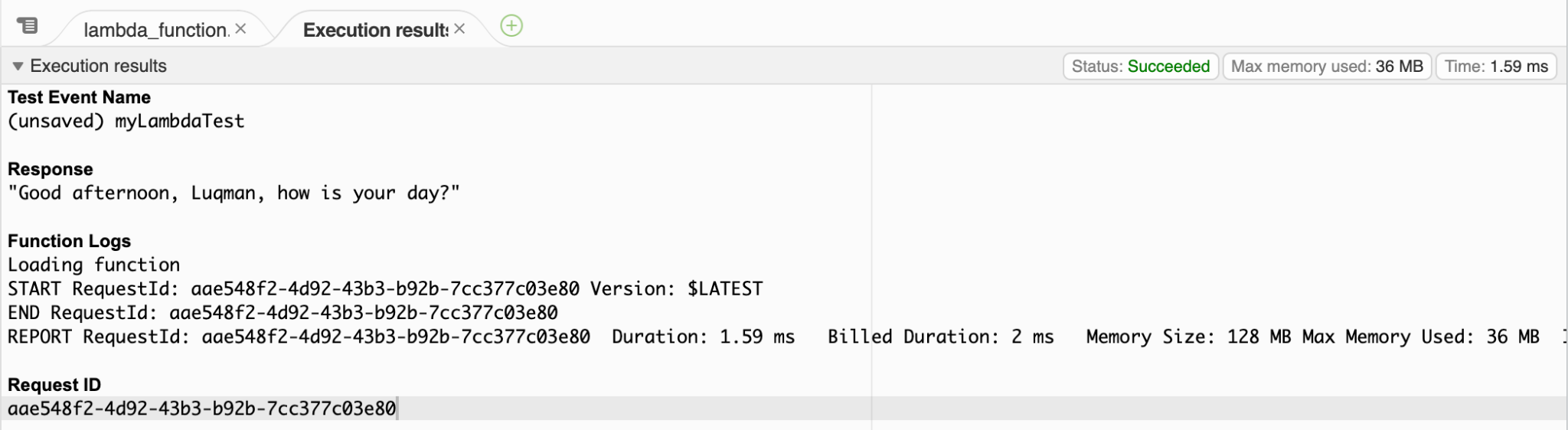
You can test your Lambda function in the console by invoking your function with a test event. A test event is a JSON input to your function. If your function doesn't require input, the event can be an empty document ({}).

Test the function with a sample event provided by the Lambda console.

* Open the Functions page of the Lambda console.
* You have 2 options to create a test function for this lambda - through the “Code” tab > Blue “Test” button or through the “Test” tab. Let’s try from the “Code” tab.
* Click the dropdown next to the “Test” button and choose “Configure test event”
* Enter event name as “<name>LambdaTest” e.g. dannyLambdaTest
* Under Event JSON, you will see 3 values for key1, key2 and key3 called value1, value2 and value3 respectively. Feel free to edit the values randomly. Example:  
  
* Scroll down and click “Save”
* Now click the blue “Test” button with your previously created Lambda test. You should see:
* 
* This is because the code only returns event['key1'] (which is “Hello”) but prints the Function Logs:
  + value1 = Hello
  + value2 = World
  + value3 = Hi
* Depending on your 3 values, these values will be shown differently.

### Let’s ramp it up

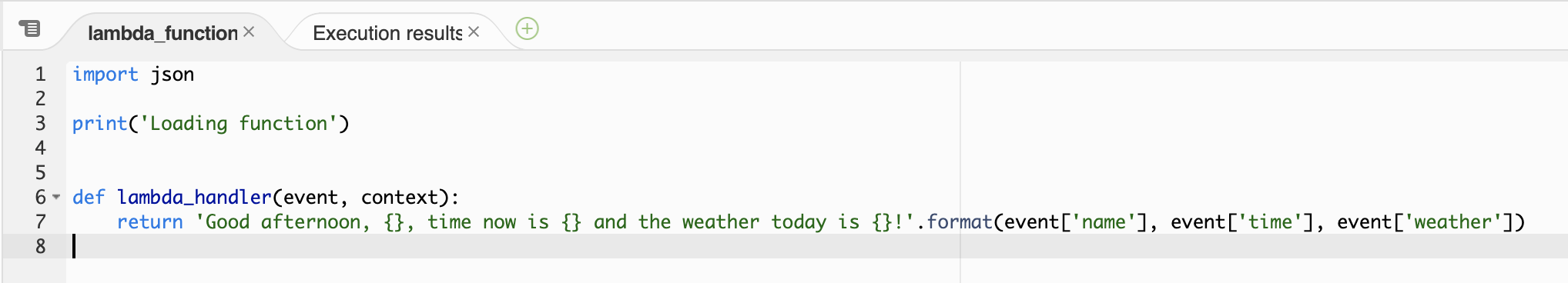
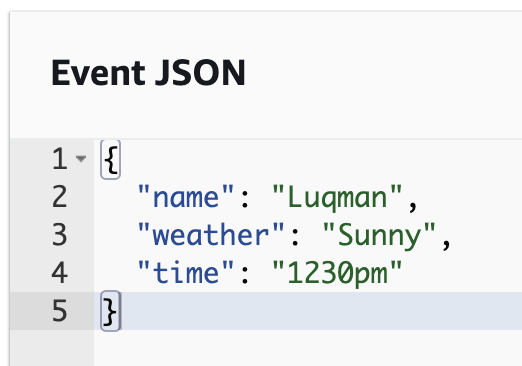
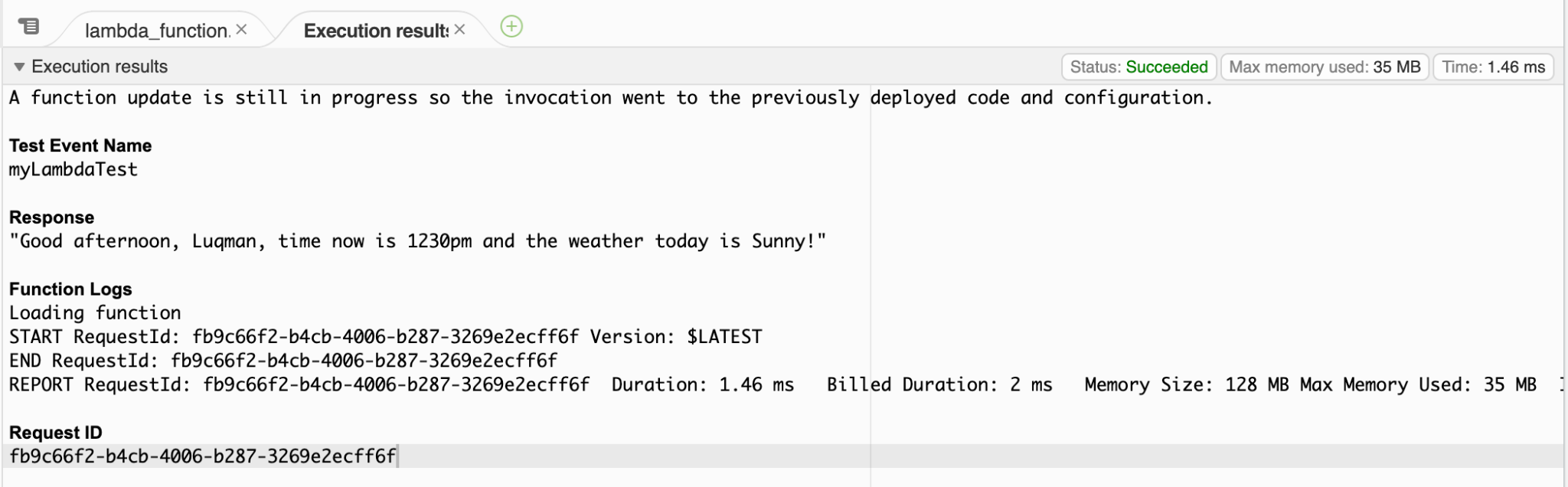
Now, let’s edit the code a little to request for only one input variable - your name.

* Under the “Code” tab, look for lambda\_function.py file. Let’s edit the lambda\_handler method.
* You will remove all the other lines under lambda\_handler and only include this one line:
  + return "Good afternoon, " + event['key1'] + ", how is your day?"
* Click “Deploy” to save your latest code.
* 
* Click the “Test” tab next to the “Code” tab.
* We will Edit saved event - dannyLambdaTest
* Remove key2 and key3, leaving only key1 behind. Change the value of value1 to your name, example:
* 
* Scroll up a little and click on the orange “Test” button, and you should see this success message:
* 
* Great! Now click save.
* Back in the “Code” tab, click the blue “Test” button again, and you should get a successful output with your variables set.
* 

### Your chance to change the code anyway you like

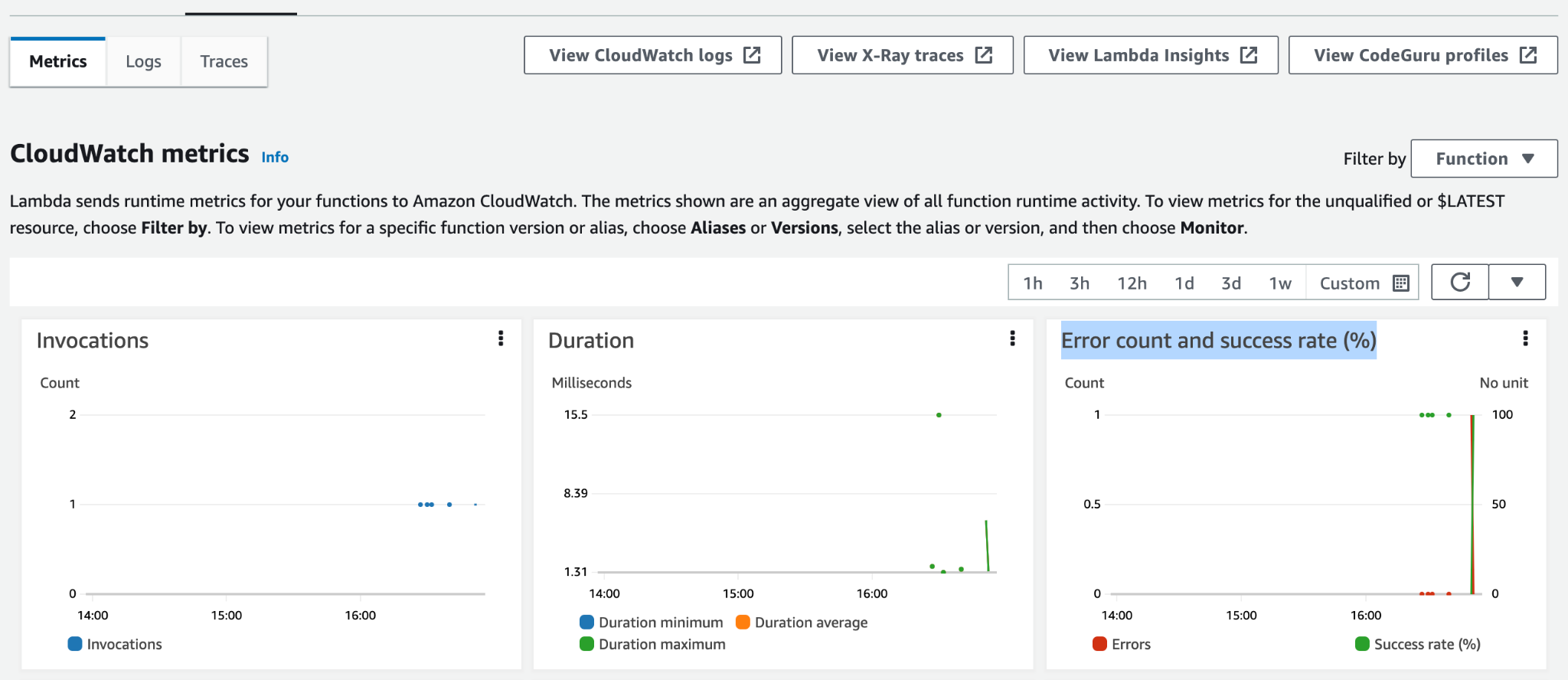
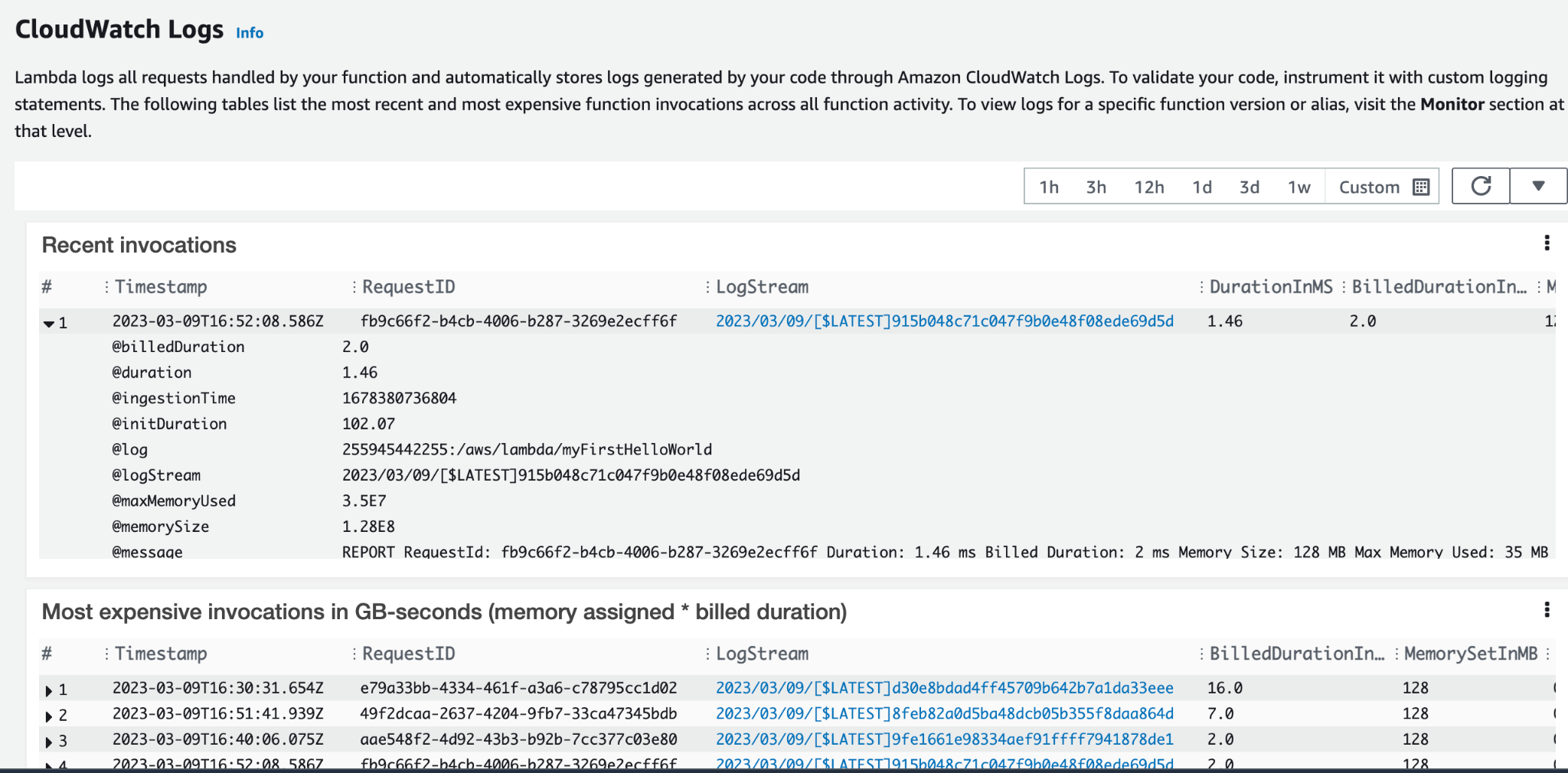
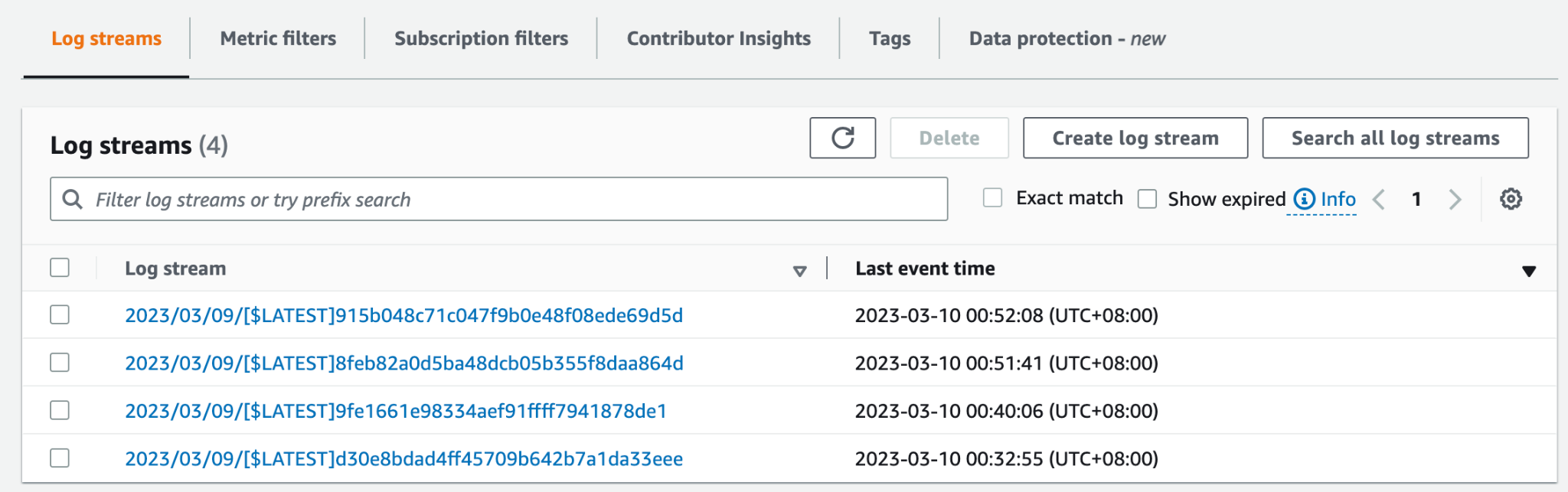
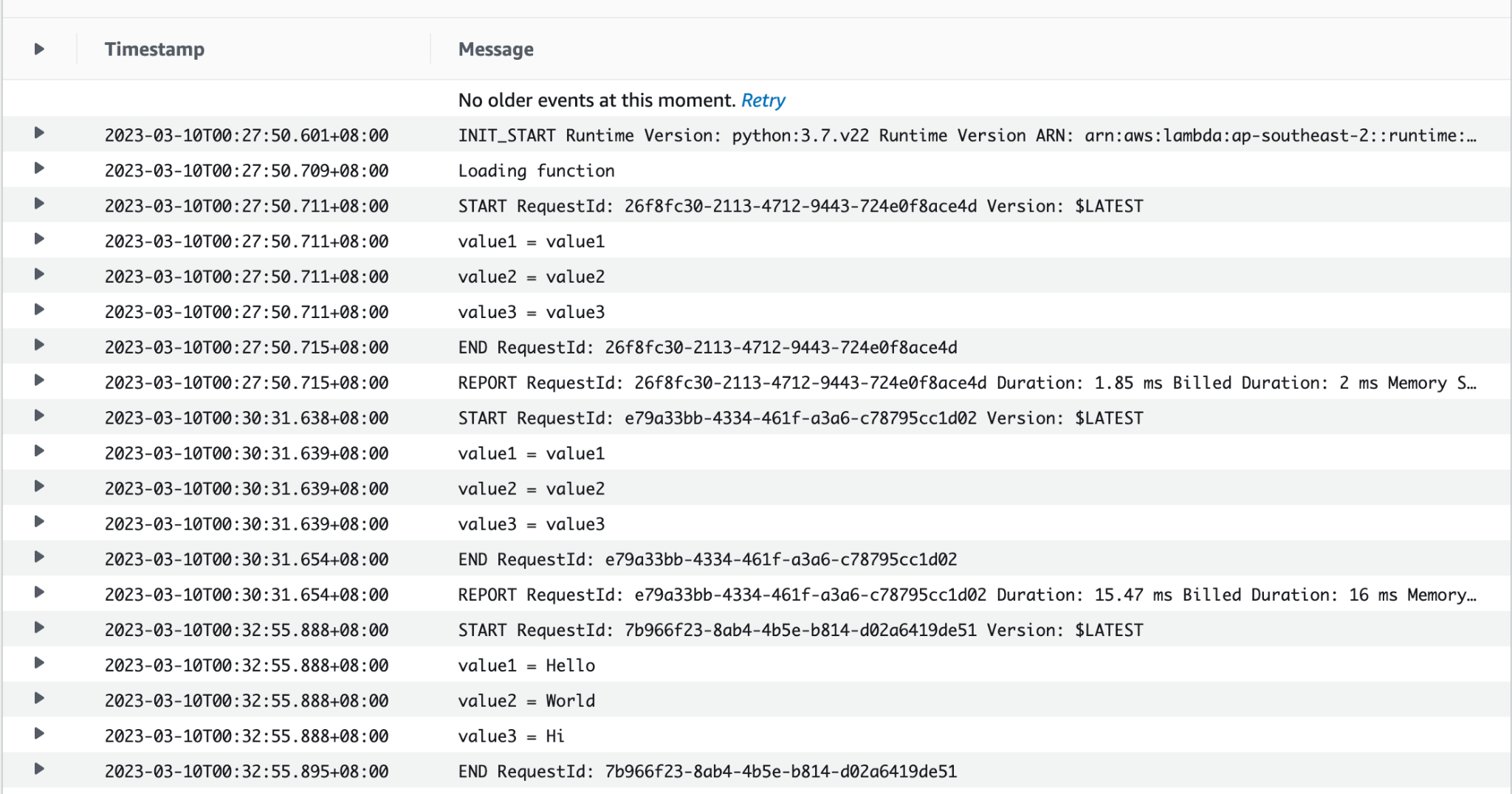
Now’s your chance to be creative and edit the code any way you’d like to. You can ask for 2 variables e.g. Name and Age, Name and Job, Country of residence and weather forecast etc. or more variables!

To get the ball rolling, you can try following something similar like below:

* Code under lambda\_handler:
  + return 'Good afternoon, {}, time now is {} and the weather today is {}!'.format(event['name'], event['time'], event['weather'])
* 
* Test - create 3 variables (name, time and weather):
* 
* Output:
* 

### Let’s see Cloudwatch logs

Running all these Lambda tests triggers a lot of logs that will be pushed and stored on Cloudwatch. To see some of them, let’s do the following:

* Click the “Monitor” tab
* You will see 3 main sub-tabs - Metrics, Logs and Traces. For today, we will explore Metrics and Logs.
* Under metrics, you will see a lot of charts showing different reports e.g. Invocation count, Duration of Lambda function ran, Error count and success rate and more. You can even toggle the time, whether you’d like to see the past 1h, 3h, 12h or custom duration.
* To get the latest metrics, click the refresh tab on the right of “Custom”
* 
* Next, under the Logs sub-tab, you can view the exact log files associated with each invocation that you run.
* Expand any of the log messages to view more information.
* 
* You can even click “View Cloudwatch Logs” to be directed to the Cloudwatch page, where you can see even more in-depth logs, example:
* 
* Click any of the log streams and you can see the details of your lambda invocations that you had done previously.
* 

### What is SNS?

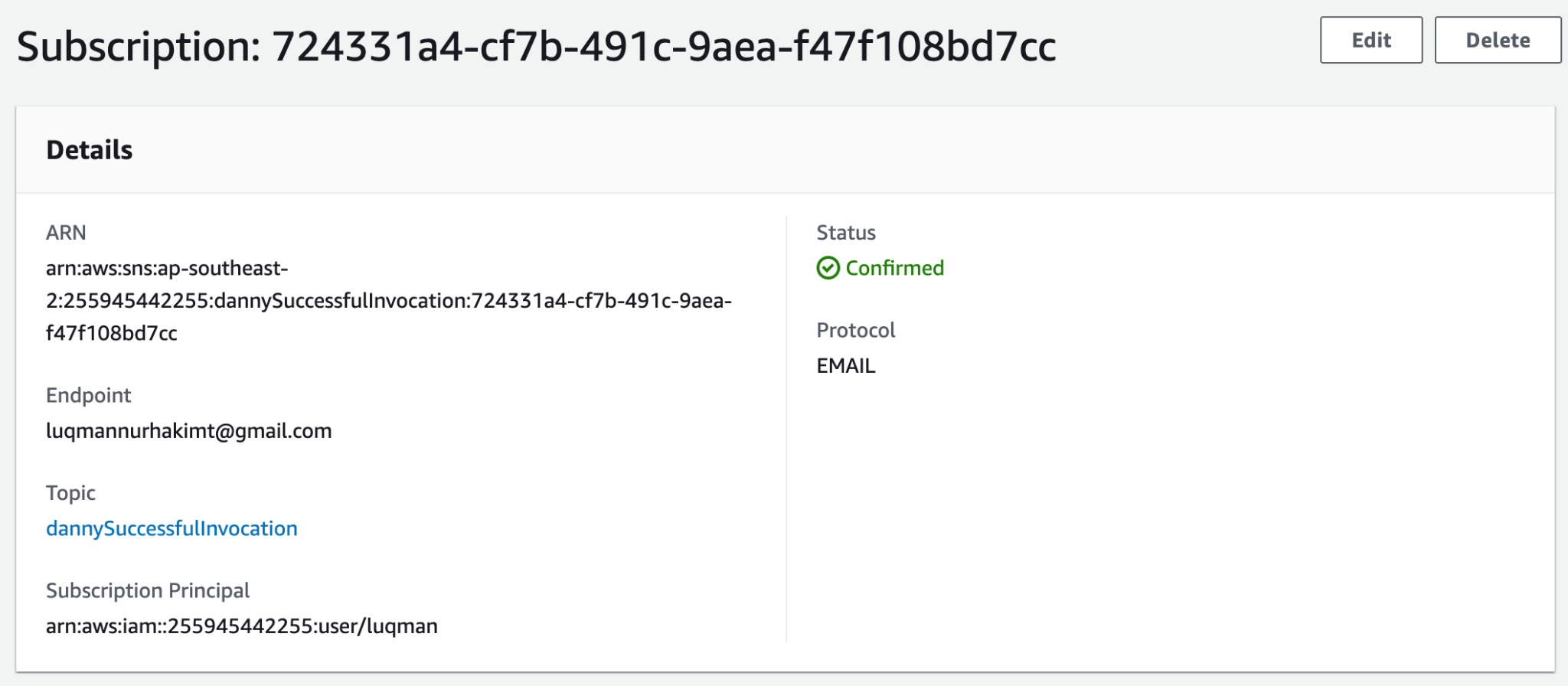
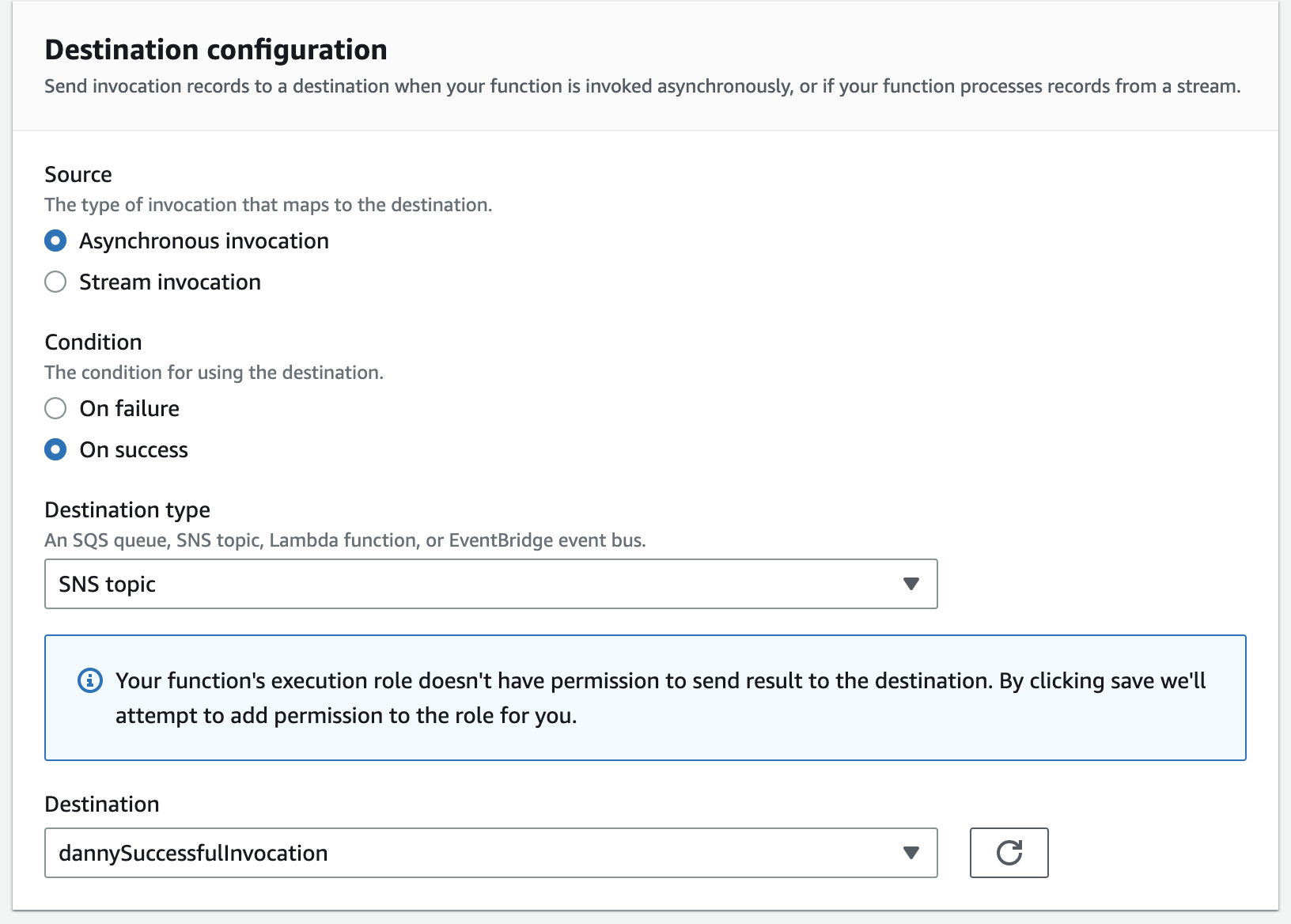
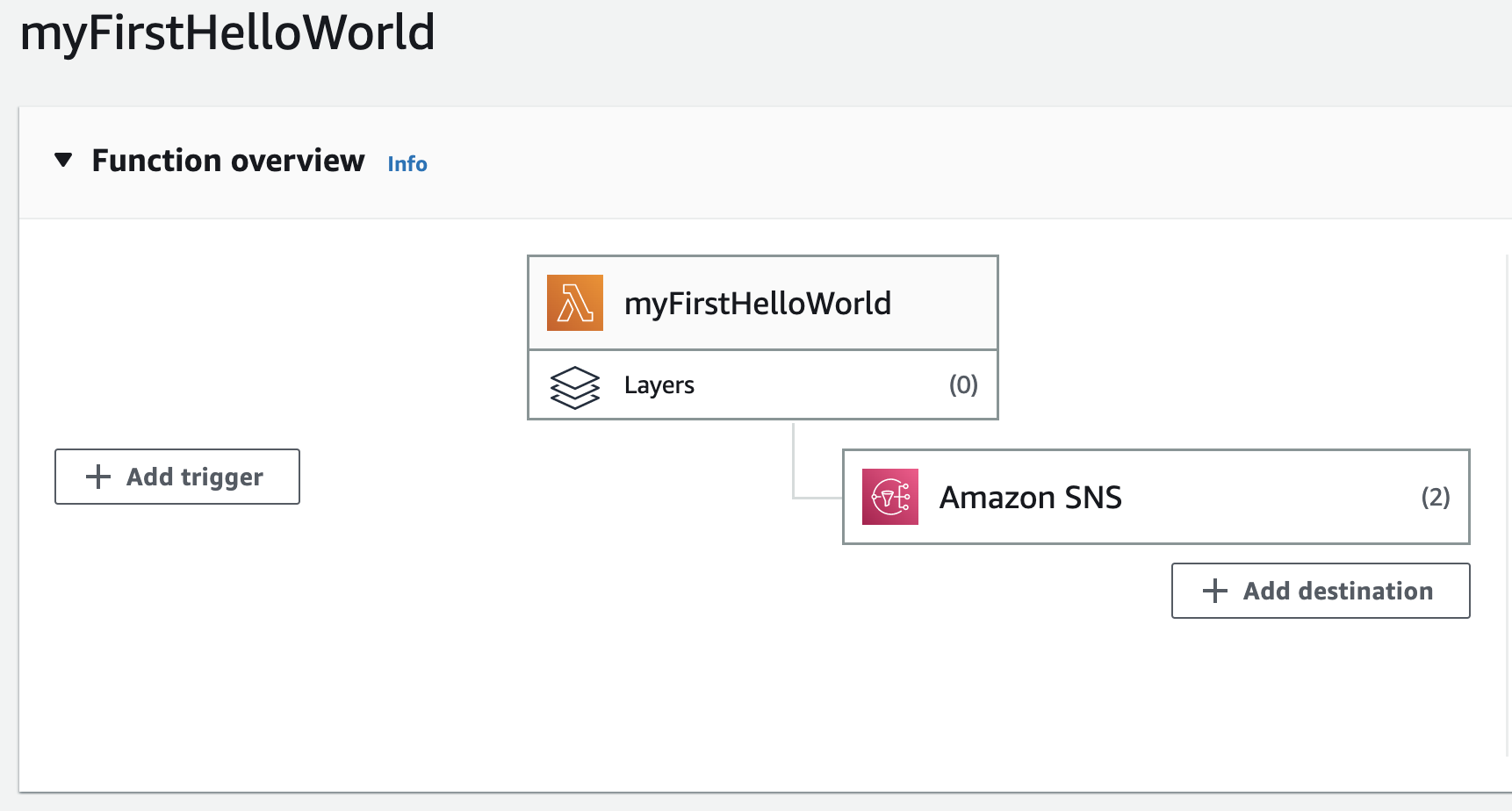
Amazon Simple Notification Service (Amazon SNS) is a **managed service that provides message delivery from publishers to subscribers** (also known as producers and consumers).

Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

Clients can subscribe to the SNS topic and receive published messages using a supported endpoint type, such as Amazon Kinesis Data Firehose, Amazon SQS, AWS Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

### Let’s connect to SNS (Successful output)

We are going to connect 2 different SNS topics, the first being for successful lambda invocations. Let’s try this out.

* Head over to SNS in AWS console.
* We will need to create a topic. Let’s call this <name>SuccessfulInvocation e.g. dannySuccessfulInvocation
* Go into the topic, and click “Create subscription”. This is where you specify your email for sending emails from Lambda.
* Choose the Protocol - Email
* Choose the Endpoint - <Your email address>
* Once this is confirmed, you will receive an email in your inbox from AWS, and you need to click “Confirm subscription”.
* Once done, you will see this reflected in the Subscription:
* 
* And you’re all set from SNS!
* Next, let’s go back to your Lambda function earlier. In the diagram at the top, click on “+ Add Destination” button next to your lambda image
* Choose source - Asynchronous invocation
* Choose condition - On success
* Choose destination type - SNS topic
* Choose destination - <your previously created SNS topic>
* 
* Click Save and you should see something similar
* 
* Here’s where it gets a little tricky, so pay close attention!
* But before that, what is boto3?

You use the **AWS SDK** for Python (Boto3) to **create, configure, and manage AWS services**, such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3) and AWS SNS. The SDK provides an object-oriented API as well as low-level access to AWS services.

* Back in the “Code” tab, edit your code to follow the below code snippet without the lines:

—------------------------------------------------

import json

import boto3

def lambda\_handler(event, context):

notification = 'Good afternoon, {}, time now is {} and the weather today is {}!'.format(event['name'], event['time'], event['weather'])

print(notification)

client = boto3.client('sns')

print("client initialized")

response = client.publish (

TopicArn = "arn:aws:sns:ap-southeast-2:255945442255:dannySuccessfulInvocation",

Message = json.dumps({'default': notification}),

MessageStructure = 'json'

)

print(response)

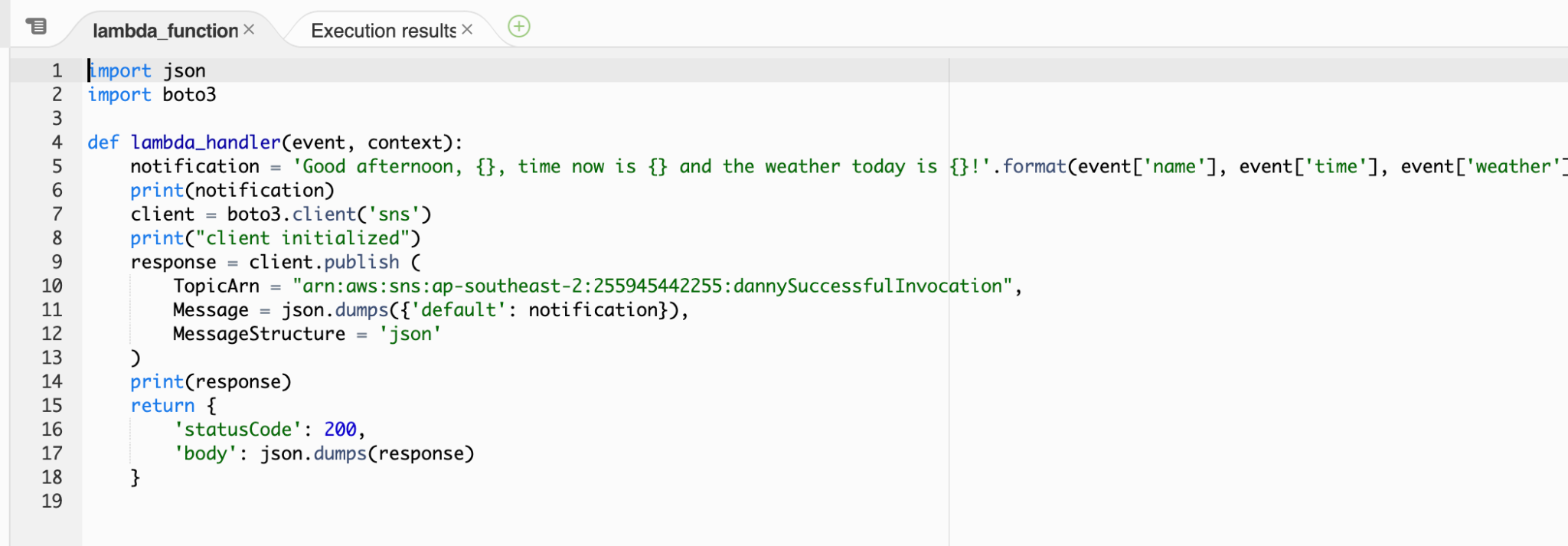
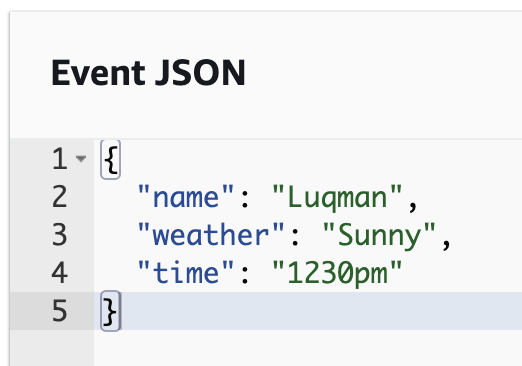
return {

'statusCode': 200,

'body': json.dumps(response)

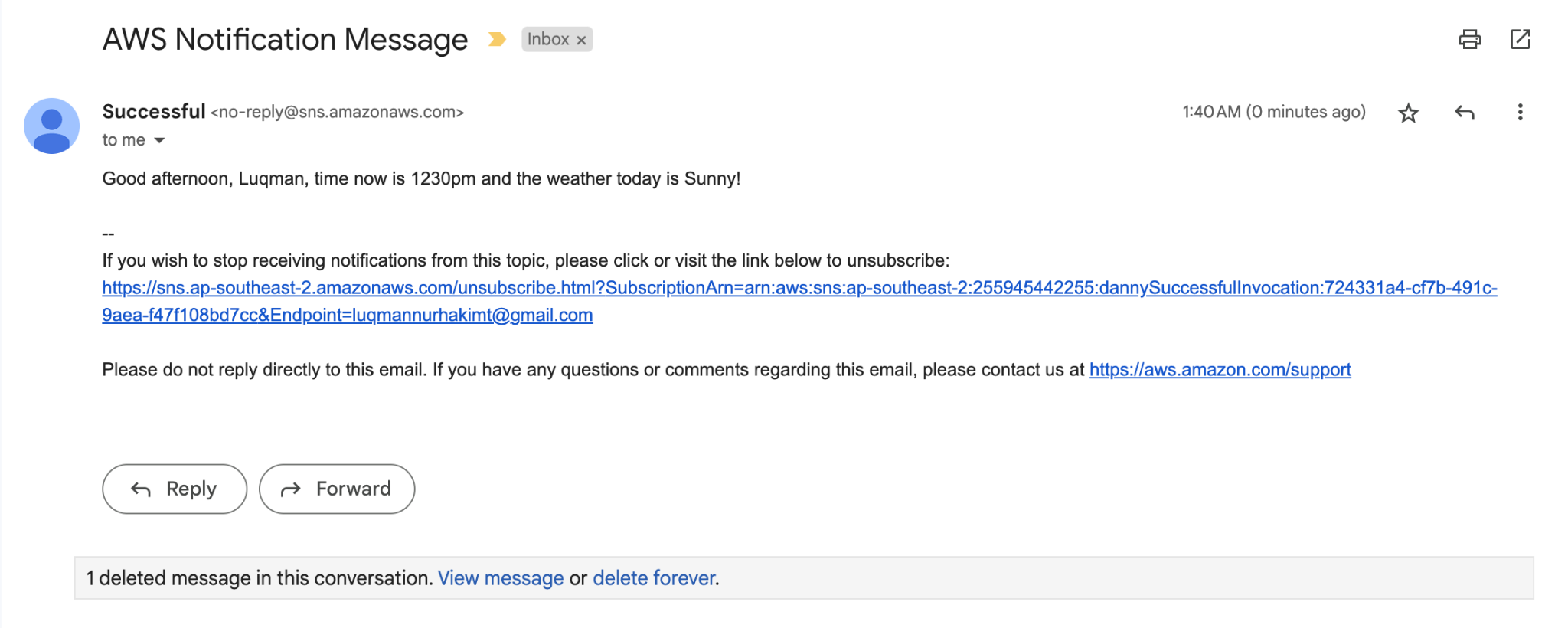
}

—------------------------------------------------

* Example:
* 
* Click Deploy once you have copied the above snippet.
* Edit your “Test” function earlier to include 3 variables:
  + name
  + weather
  + time
* 
* Save your changes.

### Testing the lambda function to SNS topic

Now you’re ready to run the lambda function!

* Click the blue “Test” button again.
* If you have successfully followed the above steps to configure your Lambda to SNS topic, you should get this in your email inbox.
* 

### 