

Getting started with Lambda function in Python

This document is mean to give you a quick guide to start your Lambda journey.

You should also go through AWS Lambda Developer Guide

<https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html> to avoid incurring unnecessary charges.

Create a Lambda function with the console

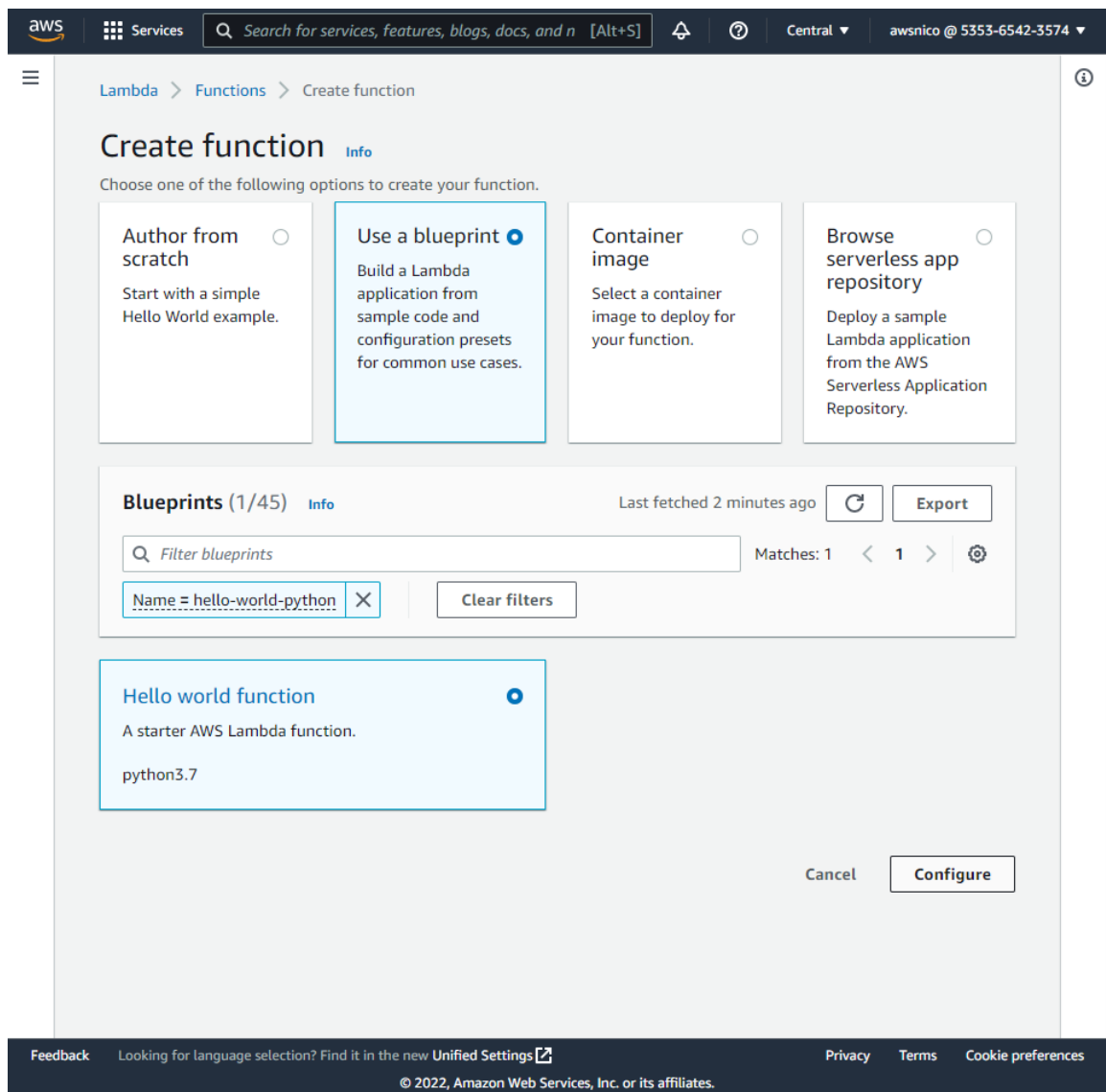
Login into your AWS account and navigate to <https://ca-central-1.console.aws.amazon.com/lambda/home?region=ca-central-1#/functions>

Here, my region is set to *Canada Centerl* as you can see at the top right corner.

Click Create function button to create your first function

Select Use a blueprint option and then type hello-world-python into the filter box

Select the Hello world function python3.7 blueprint and click Configure



Input a function name like myAverage

Choose Create a new role with basic Lambda permissions

You will also see the following code preconfigured

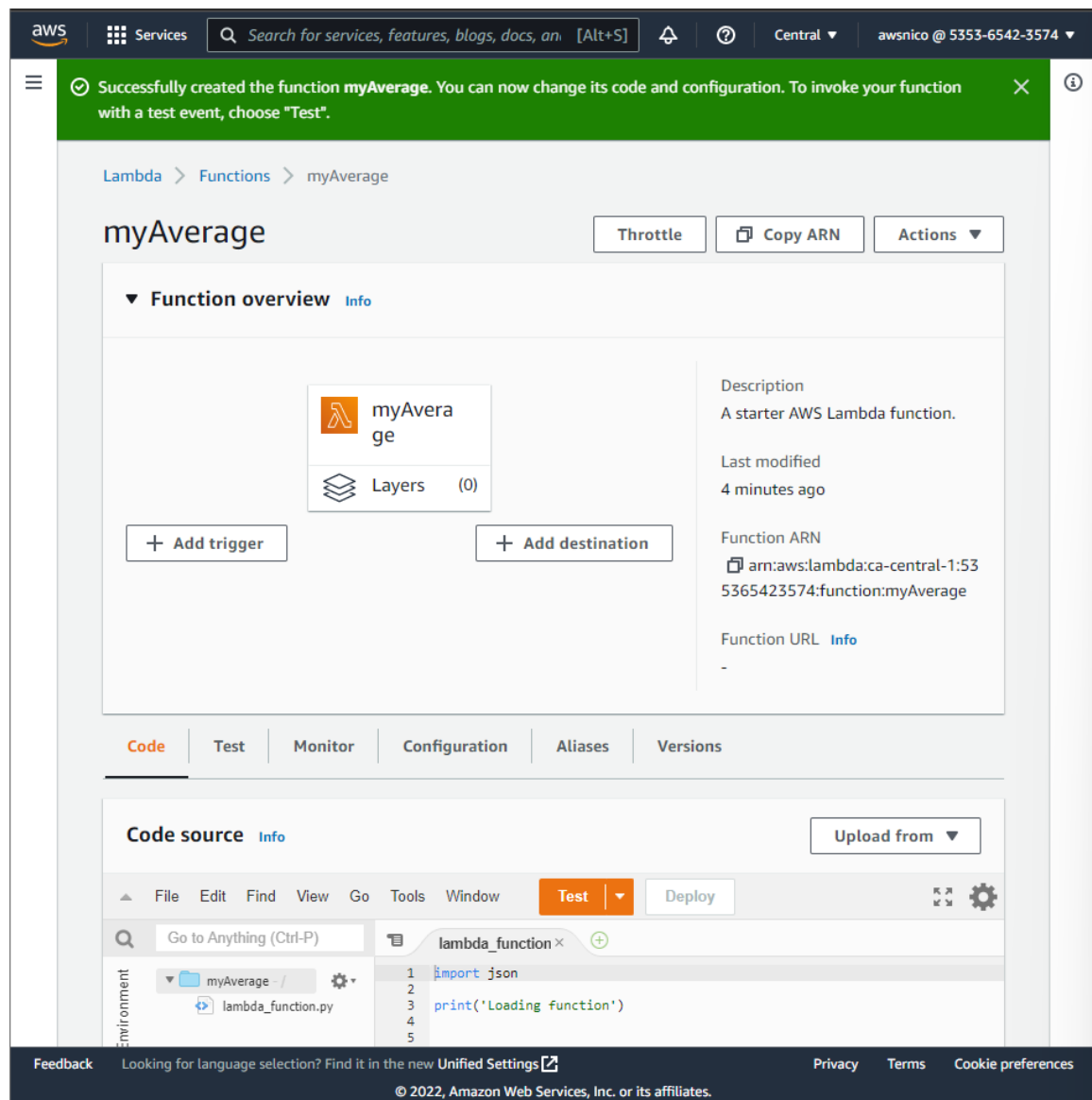
```
import json

print('Loading function')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1'] # Echo back the first key value
    #raise Exception('Something went wrong')
```

Click Create function button

On Functions -> myAverage Page, you can see the function information including its ARN and you can also authoring, testing, deploy your Lambda function.



Authoring AWS Lambda Functions

At this page, you can

Click the Throttle button which will throttle all future invocations of this function. Please only use this in case of emergencies.

Click the arrow by the Actions button to choose Publish new version, Create alias, Export function, Delete function

or Add trigger, Add destination

The Code source section can help you write you code, test and deploy

By default, there will be a folder with the same name of your Lambda function.

There is also a python file named * lambda_function.py * with a function called lambda_handler in it.

This is the default entrance of your Lambda function. You can change this in the Runtime settings section down below including language, version and handler.

We don't change it today.

Let's look at the Python code in the editor windows.

This editor is good for any Python code without 3rd party libraries other than the AWS SDK.

```
In [ ]: # Here is the code we have in the editor
# This code will print keys in event object
# and return event['key1'] as the result
import json

print('Loading function')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    print("value1 = " + event['key1'])
    print("value2 = " + event['key2'])
    print("value3 = " + event['key3'])
    return event['key1'] # Echo back the first key value
    #raise Exception('Something went wrong')

# After this handler returns, the Lambda function is ready for the next call.
```

Test our AWS Lambda Function

To test our function, we need to click the Test tab.

Input an event name as it is required and click test button.

Here we use the following JSON for testing.

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

Test event

SaveTest

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Event name

func

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private

Shareable

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

Format JSON

1 {

2 "key1": "value1",

3 "key2": "value2",

4 "key3": "value3"

5 }

And the test result is,

Execution result: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

"value1"

Summary

Code SHA-256	Request ID
9RelgzHek7RvqZvziXuEer6Q1rtzSCIE6V9b1dHRTig	2896ff6b-d1a0-4ccb-990e-059ec4eba950
=	
Init duration	Duration
113.40 ms	1.77 ms
Billed duration	Resources configured
2 ms	128 MB
Max memory used	
35 MB	

Log output

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

Loading function
START RequestId: 2896ff6b-d1a0-4ccb-990e-059ec4eba950 Version: \$LATEST
value1 = value1
value2 = value2
value3 = value3
END RequestId: 2896ff6b-d1a0-4ccb-990e-059ec4eba950
REPORT RequestId: 2896ff6b-d1a0-4ccb-990e-059ec4eba950 Duration: 1.77 ms Billed Duration: 2 ms
Memory Size: 128 MB Max Memory Used: 35 MB Init Duration: 113.40 ms

As you can see from the screenshot, there is 2 ms billed duration.

Resources configured is also a factor relate directly to the charges.

More on charges and configurations <https://docs.aws.amazon.com/lambda/index.html>

Improve our AWS Lambda Function

OK, with enough backaground. let's create our own Lambda function to calculate average of an array.

One of the best practice is to seperate your business logic with your handler.

So we won't do our calcuation inside the handler function.

Replace our code with the following.

```
In [ ]: import json

print('Loading function')

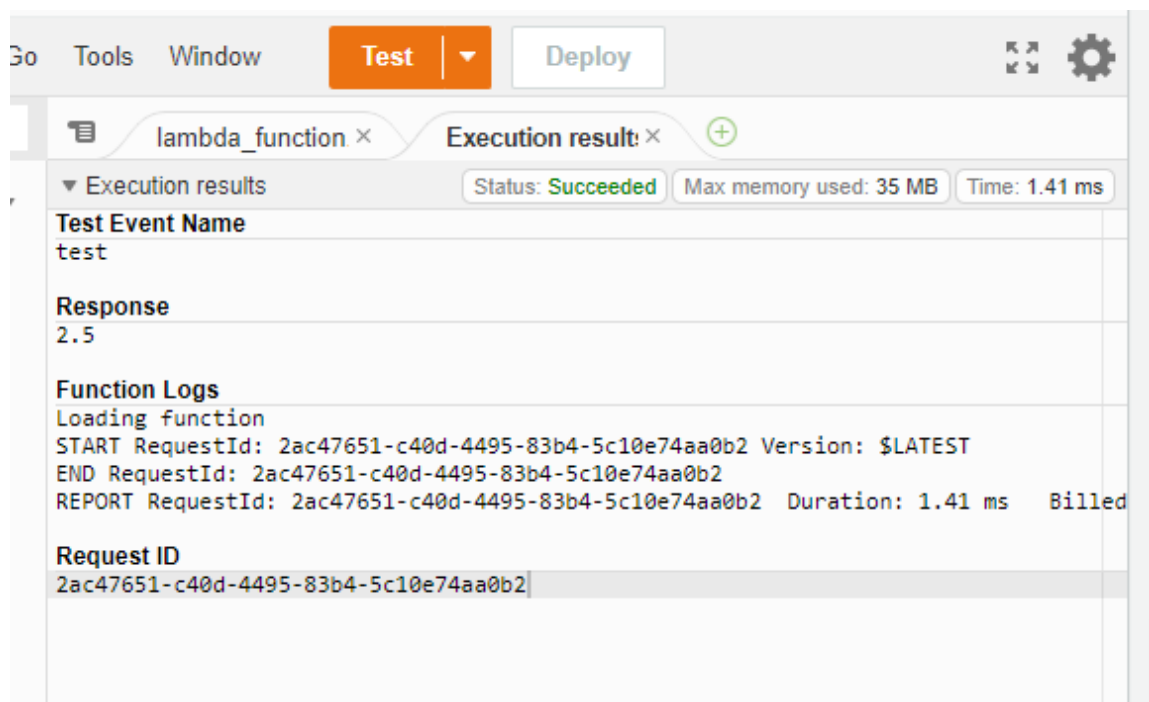
def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    #print("value1 = " + event['key1'])
    #print("value2 = " + event['key2'])
    #print("value3 = " + event['key3'])
    #return event['key1'] # Echo back the first key value
    return myAverage([1,2,3,4])
    #raise Exception('Something went wrong')

def myAverage(nums):
    ''' calculate average of list nums '''
    return sum(nums) / len(nums)
```

Each time after we modified our code, before we test it again, we need to deploy it by click the Deploy button.

You can click the down arrow button by Test and choose Configure test event and save it so that you don't need to switch to Test tab to do testing.

Here shows my testing result.



Test Event Name	Response	Function Logs	Request ID
test	2.5	Loading function START RequestId: 2ac47651-c40d-4495-83b4-5c10e74aa0b2 Version: \$LATEST END RequestId: 2ac47651-c40d-4495-83b4-5c10e74aa0b2 REPORT RequestId: 2ac47651-c40d-4495-83b4-5c10e74aa0b2 Duration: 1.41 ms Billed	2ac47651-c40d-4495-83b4-5c10e74aa0b2

Let's improve our Lambda function so that it can accept an array as its input.

First, go to configure test event page again and change the Event JSON to

```
{
  "nums": [
```

```

1,
2,
3,
4,
5,
6,
7,
8,
9,
10
]
}

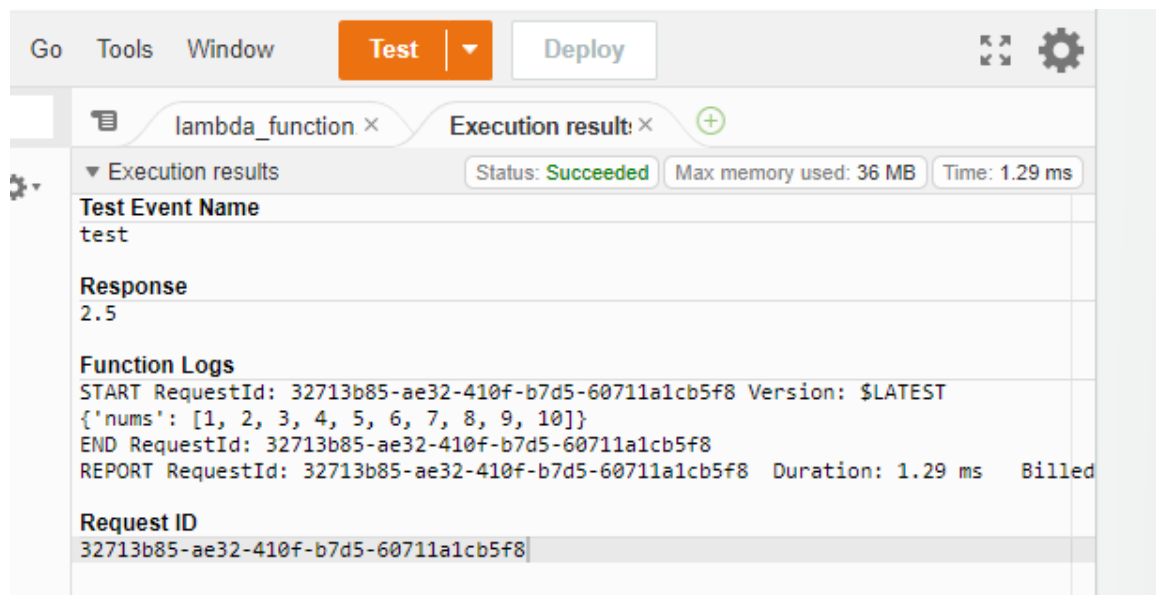
```

Also, add `print(event)` before the return expression in handler function.

In this way, we can understand more about the event object.

Test our function again.

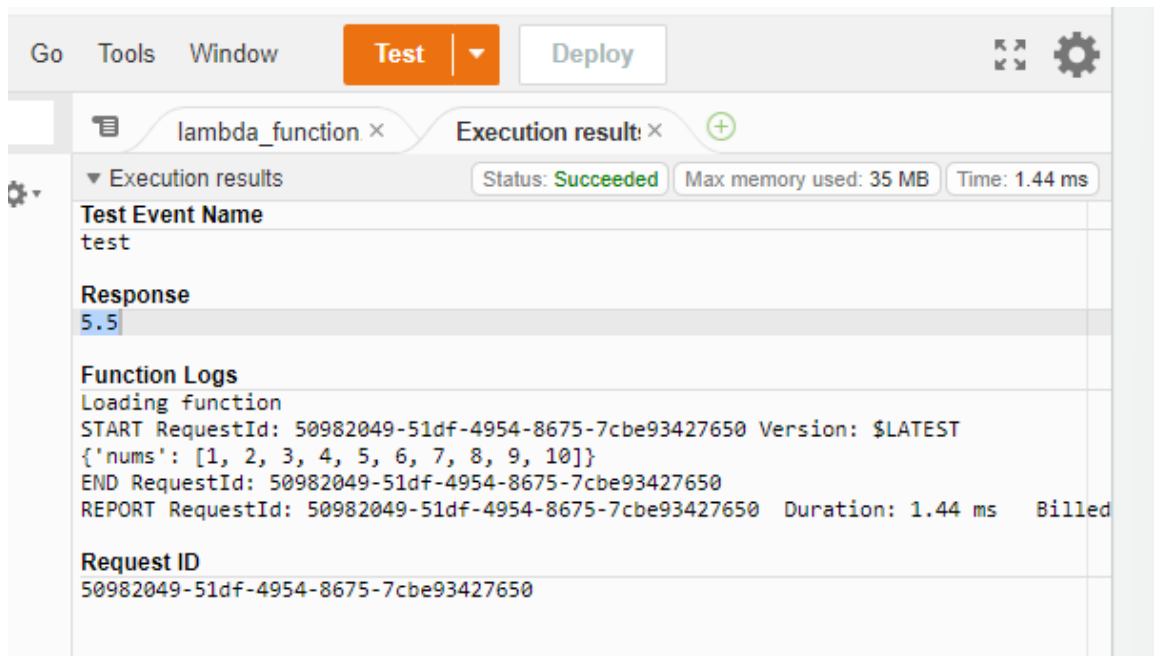
Here is the new result.



Next, let replace the magic array in the return expression to `event['nums']`.

Check the test result again.

Now the returned response is 5.5



Continue improve our function :)

Let continue improve our function by return a more meaningful JSON object instead of an integer.

The JSON we plan to have will look like this,

```
{
  'statusCode': 200,    // error code shows success function call
  'nums': [1, 2, 3, 4], // input array
  'result': 5.5         // result
}
```

There are other documents and blog posts talking about how to design a better API.... We are happy at this design right now.

So we change our code to the following,

```
In [ ]: import json

print('Loading function')

def lambda_handler(event, context):
    nums = event['nums']
    result = myAverage(nums)
    return {
        'statusCode': 200,
        'nums': nums,
        'result': result
    }
    #raise Exception('Something went wrong')
```

```
def myAverage(nums):
    ''' calculate average of list nums '''
    return sum(nums)/len(nums)
```

Invoke our AWS Lambda Function using Python

At this moment, we can say our minumumu function is ready to work.... :)

Let's try it using our provide python script.

```
In [ ]: """
If you are using AWS Academy Learner Lab, you can't create users.
You can use Cloud9 to run this code without access key.

This module will run the specified AWS Lambda function and return the results.
Need to create IAM user with programmatic access to Lambda.
Need to store IAM credentials in ~/.aws/credentials
Need to store region in ~/.aws/config

credentials file:
[default]
aws_access_key_id = blahblah
aws_secret_access_key = blahblah

config file:
[default]
region = us-east-1 (or whatever region you created function in)
"""

import json
import boto3
import botocore.response as br

# connect to AWS Lambda
lambda_client = boto3.client("lambda")

# dictionary containing function input data
p = {"nums": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}

# invoke the function with input and fetch result
response = lambda_client.invoke(FunctionName="StdDev", Payload=json.dumps(p))

# convert result (StreamingBody) into a regular Python dictionary
payload = json.loads(response["Payload"].read())

# print original list of numbers and result, rounded to 3 decimal places
print(f"StdDev of nums {payload['nums']} is: {float(payload['result']):.3f}")

# If you are following this document but encounter a error running this code .... I

StdDev of nums [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] is: 5.500
```

This code uses boto3 to consume our Lambda function.

Documentation of Boto3 can be found here:

<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

TLDR,

Install Boto3 package by

```
pip install boto3
```

Save your credential and config into your home folder.

To me, `ca-central-1` is my region and I got my credential from * Security credentials * and click * Create access key * button.

Please remember, if you lose or forget your secret key, you cannot retrieve it. You have to create a new access key.

Since our function named different than `stddev`, we need change the code to match our function.

```
In [ ]: # invoke the function with input and fetch result
        response = lambda_client.invoke(FunctionName="myAverage", Payload=json.dumps(p))

        # convert result (StreamingBody) into a regular Python dictionary
        payload = json.loads(response["Payload"].read())

        # print original list of numbers and result, rounded to 3 decimal places
        print(f"Average of nums {payload['nums']} is: {float(payload['result']):.3f}")
```

Congratulations, you now have your first Lambda Function confirmed!!

At the end

As we said in the beginning, this document is meant to get you start writing your AWS Lambda function as quick as possible. Please refer to the full Developer Guide form AWS before put your AWS Lambda function into production.

PDF: <https://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf>

AWS Lambda Documentation: <https://docs.aws.amazon.com/lambda/index.html>

Delete your function when you don't need it

After your assignment graded, you can choose Actions - Delete function to delete your function on the Lambda function page. This will save your from incurring any charges.

With Amazon free tier and your credential kept screate to yourself, you should have no billing on Lambda functions.