

# AWS Lambda Functions Tutorial/Lab

## More AWS: Lambda Functions

For this tutorial/lab you will walk through a tutorial about creating an AWS “hello world” Lambda function. After the tutorial, you will create your own AWS Lambda function to calculate the nth number in the Fibonacci sequence.

## Build a Hello World API with AWS Lambda

In the tutorial below you will create a Hello World AWS Lambda function that can be triggered through an API call. You will complete the tutorial by taking a screenshot of the API response in your browser.

The original tutorial from AWS is written in JavaScript. If you know Javascript you can complete the steps below. Otherwise, read the [Java Tutorial Google Doc](#) to do the tutorial in Java instead. After completing either tutorial continue to the “Create an AWS Lambda Fibonacci Function” section in this document.

1. Click the link below and follow the tutorial stopping before the section titled “To test the deployed API using URL:”
  - a. [TUTORIAL: Hello World API with Lambda Proxy Integration](#)
2. Change the API request to be the name of the person to be your first and last name and the city to be the city that you are from.
3. **Take a screenshot of your browser making the new API Request**

## Verify that your Lambda function is working correctly

***If you already did this in the java lambda tutorial, you don't have to do this section again.***

Follow the directions given by the following document to create a test and a log.

**\*\* Make sure the values are different\*\***

[Lambda Testing and Debugging](#)

1. Take a Screenshot of your helloWorld Lamba test JSON. The screenshot below shows what we want in the screenshot

Configure test event

Create new test event

Edit saved test events

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Event template

hello-world

Event name

Test1

1- {

2-   "city": "value1",

3-   "name": "value2"

4- }

Cancel

Format JSON

Create

- Take a Screenshot of a test log containing some output/log from your lambda test. The screenshot below shows what we want in the screenshot

Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

View as text

Actions

Create Metric Filter

Filter events

Clear 1m 30m 1h 12h Custom

Timestamp	Message
	No older events at this moment. <a href="#">Retry</a>
2021-02-03T09:53:16.779-07:00	START RequestId: dc451626-44ba-47da-ab38-469ba859ea26 Version: \$LATEST
2021-02-03T09:53:16.794-07:00	example log! Request is being handled!
2021-02-03T09:53:16.794-07:00	value1
2021-02-03T09:53:16.794-07:00	value2
2021-02-03T09:53:16.800-07:00	END RequestId: dc451626-44ba-47da-ab38-469ba859ea26
2021-02-03T09:53:16.800-07:00	REPORT RequestId: dc451626-44ba-47da-ab38-469ba859ea26 Duration: 20.00 ms Billed Duration: 20 ms Memory Size: 512 MB Max Memory Used: 90 MB Init Duratio...
	No newer events at this moment. <a href="#">Auto retry paused</a> . <a href="#">Resume</a>

## Create an AWS Lambda Fibonacci Function

Now that you have created a Hello World Lambda function, you will apply your knowledge to create an AWS Lambda Function for the Fibonacci function. Create the function that meets the

requirements below then take a screenshot of the completed api request to your function. (Note: you can use any language that AWS lambda supports)

**Note:** If you did the Java tutorial, create a new Maven project called FibonacciLambda. **You will need to copy the properties and dependencies from the HelloLambdaWorld project's pom.xml file into the new project's pom.xml file.** Create new handler, request and response classes similar to the ones from the first project but updated to support the new Fibonacci requirements. Remember to update your **Handler** field in the AWS Lambda console after you upload your JAR file.

## Fibonacci Function

Create a function that returns the nth number in the sequence, where the first two numbers are 1 and the following numbers are defined by the previous two numbers.

*Fibonacci(n) = if n > 2 then return Fibonacci(n - 1) + Fibonacci(n - 2); else return 1*

Example of Result:

The 8th fibonacci number is 21

1, 1, 2, 3, 5, 8, 13, 21

## Function Requirements

The API request should look similar to the following string "url/fibonacci?n=8"

It should return a response that looks something like "{ 'fibonacci':21, 'author': 'John Doe' }

- Argument(s)
  - "n" - an integer
- Response
  - "fibonacci" - the nth Fibonacci number based on the argument above
  - "author" - the full name of the developer (you)

## Hints

1. Hints with the API Mapping template

application/json

Generate template:

```
1 {  
2   "name": "$input.params().querystring.get('name')",  
3   "city": "$input.params().querystring.get('city')"  
4 }
```

- a. The variable name in Box #1 **MUST MATCH** the variable name you have in your project request object constructor
- b. The variable name in Box #2 defines how you will identify the variable in your url query string

**Note:** If you don't remove the AWS services you may accrue AWS service fees once your free tier eligibility expires or if you have already used up your free-tier eligibility