


# AWS Lambda Tutorial: A Guide to Creating Your First Function

July 2, 2021  
By SentinelOne

AWS Lambda is one of the leading serverless architectures in the cloud today. It was first-to-market among the major cloud vendors, offers the most

In this tutorial, you'll set up your first AWS Lambda function. You'll create a service, add a few lines of code, and test it from inside the AWS console

 A Lambda sign in the AWS boxes signifying AWS Lambda tutorial


## What Is AWS Lambda?

AWS Lambda is a way to run code without creating, managing, or paying for servers. You supply AWS with the code required to run your function, and

Your code can access any other AWS service, or it can run on its own. While some rules about how long a function has to respond to a request, there's

The real power, though, comes from the scalability that Lambda offers you. AWS will scale your code for you, depending on the number of requests

If you would like to explore some more use cases for AWS Lambda, [try reading this article](#).

 The real power, though, comes from the scalability that Lambda offers you.

That all sounds great, but let's balance out some of the excitement with a quick look at some trade-offs you might have to make if you use AWS Lambda.

## Should I Use It?

While AWS Lambda and serverless architecture have some really cool benefits, there are caveats. State management in a serverless architecture requires

Because Lambdas are not persistent, you cannot use connection pooling from the Lambda. If you try to use traditional database connections and Amazon

Amazon wants to help you solve this data access problem by providing additional services such as [DynamoDB](#) or [Aurora](#). This is great to help you scale

Lambdas have a hard limit on execution time. After 15 minutes, your function will be stopped whether it's finished or not. If this is a problem, then you

This limit comes with a few others: AWS doesn't allow more than 512MB of disk space for your functions and the invocation payload (request and response)

Another potential issue is the cold start.

Lambdas can be really cheap because you only pay for what you use, but that means that when the lambda is finally triggered, there will be some overhead

## AWS Lambda vs. AWS EC2 vs. Elastic Beanstalk

AWS Lambda is not the only computing service that AWS provides. So how is it different from other services like EC2 and Beanstalk?

Lambda is a Platform as a Service (PaaS) whereas EC2 is an Infrastructure as a Service (IaaS). EC2 is more flexible and customizable when compared to

towards serverless architecture.

Beanstalk is a Platform as a Service (PaaS) to deploy and manage applications on the cloud. Beanstalk provides default provisioning, load balancing and

application on the cloud, but when using Lambda, only the functional part of the code goes on the cloud.

The bottom line is that all these computing services have their own pros and cons, and are specialized for different purposes. Which suits you best?

So it looks like there are some things to consider regarding data and state management, vendor lock-in, and performance. That being said, you should

## AWS Lambda Tutorial

### AWS Account


The first thing you'll need to create a Lambda function is an AWS account if you don't already have one. The good news is that Amazon makes it very

easy to get started. Go to the [AWS Management Console](#) and create an account.

Once you have an account, log in to AWS.

### Create a Lambda

Once you're at the console, you can start setting up your function. Click on the services menu near the upper right-hand side of the page. Then, you

 AWS Lambda tutorial

Click the Lambda entry, and AWS will take you to your Lambda console. Here's mine.

If you have a new account, your console will be empty. Either way, click the **Create Function** button.

Before you proceed, let's cover some more basics.

### Lambda Functions

Before you create a Lambda function, you need to identify its inputs and triggers, choose a runtime environment, and decide what permissions and

resources your function needs. Lambda functions accept JSON-formatted input and usually respond using the same format. Your function's input and output contents are closely tied

to the event source. An obvious event source is a web request. You could set up your lambdas as an HTTP service. But they are capable of responding to events from

many other sources. You also need to select a runtime for your function. AWS Lambda supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby. There's also a Run

Time layer. Finally, your function will need an AWS role.


This role defines the entitlements the function has within the AWS platform. [AWS security](#) is a deep enough topic that it deserves a series of articles.

### Pick a Blueprint

Now it's time to finish creating your function. You'll use Python for this function because you can enter the code right into the console.

First, select the **Use a Blueprint** box in the center of the Create Function page.


Then, type **Hello** in the search box.

 AWS Lambda tutorials

Press enter and AWS will search for blueprints with Hello in the name. One of them will be hello-world-python. Select this and click **Configure**.

### Configure and Create Your Function

This will take you to a form where you will name your function, select a role, and edit the Python code.

 AWS Lambda tutorials

Enter a name, and leave the default role. The default role allows your lambda to send system out logs to CloudWatch.

Let's take a quick look at the Python code included in the blueprint.

 AWS Lambda tutorials

```
import json
print('Loading function')
def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    value1 = value1
    value2 = value2
    value3 = value3
    END RequestId: d17a8ec3-4231-4bff-8dea-5c3f1a12342e
    REPORT RequestId: d17a8ec3-4231-4bff-8dea-5c3f1a12342e Duration: 1.54 ms Billed Duration: 100 ms Memory Size: 128 MB
```

AWS will call the `lambda_handler` function each time an event is triggered. This function prints the values associated with three JSON fields: "key1", "key2", and "key3".

Click the **Create Function** button at the bottom of the form.

You've created a Lambda function! Now let's make an edit using the web editor. Let's make a simple edit and uncomment the JSON dump on line 7.

The **Save** button at the top right of the page should go from being grayed-out to orange. Once you hit the **Save** button, you should see a banner at the top of the page.

### Test Your Lambda Function

Now, let's follow the instructions at the top of the page. Click the **Test** button that is next to the **Save** button. AWS will display a form that looks similar

to the one you see in the screenshot above. This test will pass a simple JSON document to your function with the three keys it expects set to "value1," "value2," and "value3." That's good enough

to get started. AWS saves your test, and you can run it from the function page with the **Test** button. This makes it easy to set up different test cases and run them later.

Click the test button. AWS will run your test and display a result box.

The test succeeded, and you can see your log result if you click the details disclosure icon.

Here they are:

```
START RequestId: d17a8ec3-4231-4bff-8dea-5c3f1a12342e Version: SLATEST
Received event: {
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
value1 = value1
value2 = value2
value3 = value3
END RequestId: d17a8ec3-4231-4bff-8dea-5c3f1a12342e
REPORT RequestId: d17a8ec3-4231-4bff-8dea-5c3f1a12342e Duration: 1.54 ms Billed Duration: 100 ms Memory Size: 128 MB
```

At the top of the log is the "Received event" print statement you just uncommented from the code. Then there's the rest of the messages in the log.

Let's modify the code and run a different test.

First, uncomment the last line in the code and comment the line before it, so it looks like this:

```
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')
```

Now you will see an exception when you call the function. Click the **Save** button on the top right-hand side of the page.

Next, rerun the test.

AWS caught the exception, logged it for you, and registered the test as a failure.

Congratulations! You wrote and tested your first AWS Lambda function, but now what?

### Next Steps

Now that you can log in to AWS and create functions using Amazon's blueprints, it's time to take it to the next level. It is really easy to get a quick lambda

function up and running. You might want to look at how to build and deploy functions from your local machine or a continuous integration server. To do that, you could consider

 AWS Lambda tutorials

[Amazon's Lambda documentation](#) is also well-written and is a valuable resource to learn how you can take advantage of serverless architecture, as well as

how to integrate Lambda with other AWS services. Start creating Lambdas with your code, and use them to tie your services together. Lambdas are a powerful mechanism for building scalable applications.

*This post was written by Eric Goebelbecker. Eric has worked in the financial markets in New York City for 25 years, developing infrastructure for market*

**Like this article? Follow us on [LinkedIn](#), [Twitter](#), [YouTube](#) or [Facebook](#) to see the content we post.**

### Read more about Cyber Security

- [Machine Data: What Is It & How to Get the Most Out of It](#)

