SentinelOne blog

GET A DEMO

AWS Lambda Use Cases: 11 Reasons Devs Should Use Lambdas

July 15, 2020
by SentinelOne

AWS Lambda is Amazon's serverless compute service. You can run your code on it without having to manage servers or even containers. It'll automatically scale depending on how much work you feed into it.

You can use it in data pipelines or to respond to web requests or even to compose and send emails. It's the jack-of-all-trades way to execute code in the AWS cloud.

Although intended for use in the cloud, you can absolutely run Lambdas locally during development. When you do run Lambdas in the cloud, you'll only pay for what you use.

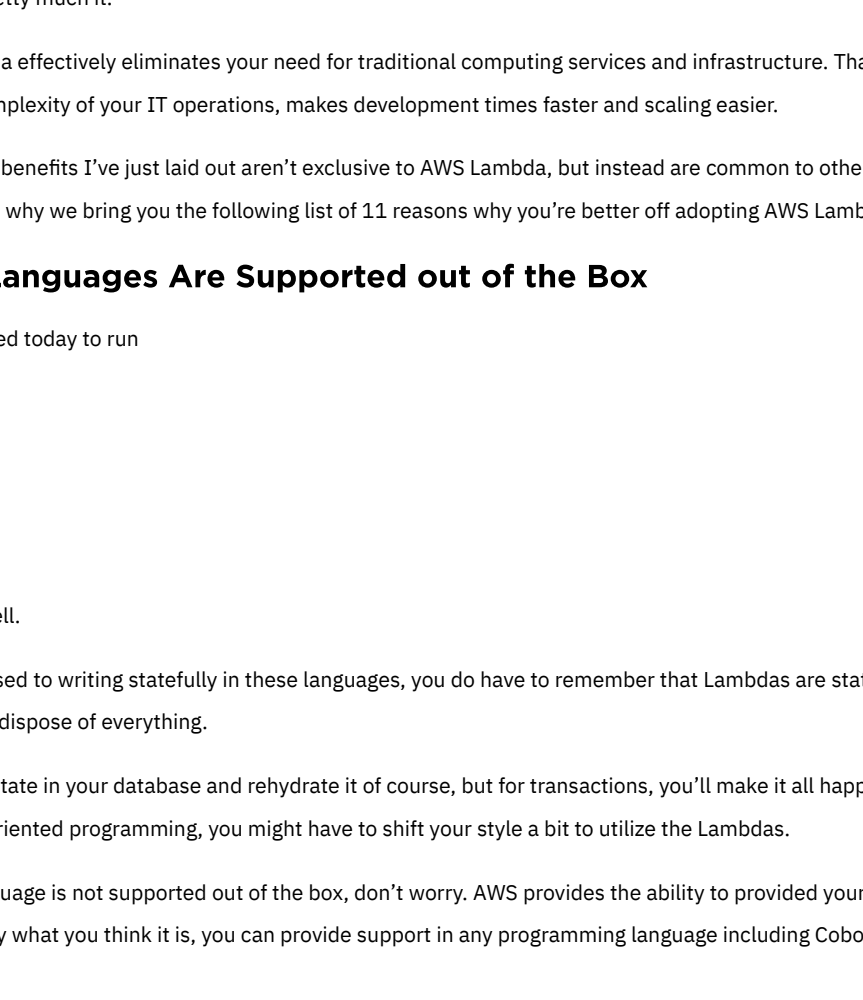
In some cases, this can save significant sums of money compared to the cost of running VMs or containers.

While I've already touched on some of the reasons you'll benefit from using AWS Lambdas, I want to focus on 11 specific reasons in more detail. That's what today's post is about.

We start with a high-level overview of AWS Lambda, diving a little deeper into what this service is, how it works, and what's so attractive about it. After that, we give you the promised list of more specific use cases.

After that, we briefly cover some of the limitations of AWS Lambda, so you're aware of scenarios where it might not fit so nicely. Finally, we wrap-up by summarizing the post and sharing a few final considerations.

Let's dig in!



What Is AWS Lambda? Why Should You Care?

In the introduction, we've briefly defined AWS Lambda and briefly mentioned some of its benefits. Now, we'll get a little bit deeper into that. Feel free to skip this section if you're already familiar with the service.

AWS Lambda is a service that allows developers and software companies to run code without having to set up and manage servers, which is often known as "serverless architecture." How does it work?

In a nutshell, you send your function's code to AWS, and they run it. You pay for the time the function actually gets executed, in a "pay as you go model." That's pretty much it.

So, adopting AWS Lambda effectively eliminates your need for traditional computing services and infrastructure. That, in its turn, greatly reduces the cost and complexity of your IT operations, makes development times faster and scaling easier.

You might argue that the benefits I've just laid out aren't exclusive to AWS Lambda, but instead are common to other serverless solutions. You'd be right. And that's why we bring you the following list of 11 reasons why you're better off adopting AWS Lambda.

1. Most Major Languages Are Supported out of the Box

AWS Lambdas can be used today to run

- Node.js,
- Python,
- Java,
- Go,
- C#,
- and even PowerShell.

Although you might be used to writing statefully in these languages, you do have to remember that Lambdas are stateless. They just run your function once, then dispose of everything.

You'll be able to persist state in your database and rehydrate it of course, but for transactions, you'll make it all happen in one go. If you're used to stateful object-oriented programming, you might have to shift your style a bit to utilize the Lambdas.

And if your favourite language is not supported out of the box, don't worry. AWS provides the ability to provided your own custom runtime. A custom runtime is exactly what you think it is, you can provide support in any programming language including Cobol, if that's your preferred language.

2. You Only Pay for What You Use

As mentioned in the introduction, you'll only pay for what you use. Billing for functions is based on 100 ms intervals.

Interestingly, this creates an incentive to use more efficient code if you're considering the costs. Also, you can tune the maximum resource usage of each Lambda, which affects the base price for your function executions.

To give you a quick idea of what it might cost, consider a function that simply logs an Amazon Cognito authorization. With a minimum memory setting of 128 MB, this Node.js function simply does the following:

```
console.log(JSON.stringify(event));
```

This logs to an AWS S3 bucket. It runs in under 100 ms, so the execution time charged for each run is 100 ms. The estimated charges for 10 million executions of this Lambda function per month is around $2.

And that's only part of the cost savings. Lack of infrastructure maintenance is the bigger story here.

AWS Lambdas are simply another tool in your toolset.

3. There's No Infrastructure to Manage

Managers like this benefit, but there are plenty of reasons for developers to like it too. How many times have you turned a server upsidedown to resolve an issue with a server patch?

As a developer, would you want to spend a good portion of your time managing servers? Or would you rather be coding up some cool stuff that matters?

Sure, it's a fun engineering project to fine-tune a cluster of containers or a VM using shared resources, but you can still have that if you'd like.

AWS Lambdas are simply another tool in your toolset. You can use them in many applications, but not every situation is right for a Lambda.

The benefit of having no infrastructure to manage is that you can spin up a new Lambda without having to add a new resource to your infrastructure.

It's a good way to extend your existing infrastructure or deploy new services in a microservices architecture. And for some services on AWS, Lambdas are the only real way to perform certain tasks such as logging Cognito login requests. And because there is no infrastructure to maintain you also don't need to take care about scalability. AWS Lambda automatically scales to support the rate of incoming requests without requiring you to configure anything. There is no limit to the number of requests your code can handle, the performance remains the same when the number of requests increases. Just remember that your code needs to be stateless in order to scale.

Your code needs to be stateless in order to scale.

4. You Can Edit Directly Online

For some of the supported languages (JavaScript and Python), there's an online editor in the Lambda interface on the AWS web console. You can literally code and edit the function using the web browser.

This means you can create or edit a Lambda using your mobile device. It saves each version too, and you can switch between new and old versions. And you can map versions to "environments" using aliases.

Don't get me wrong. You can still create and edit functions locally and deploy them using various deployment methods.

It's just that you might come across a situation where it would be handy to create a function when you aren't quite able to get to your favorite IDE.

You can still download the functions you create in the web console and add them to source control if you choose. It's just nice to have options.

5. It Connects to API Gateway and Other Connection Points

In a manner of speaking, the API Gateway is the portal to your microcosm within AWS. It'll connect to many of the cloud services directly, but when you need flexibility, Lambda is your go-to.

See, it's where you put custom code to process requests that come in through the API Gateway. Use a Lambda when you need to access several services or do custom processing.

As data flows through services, you use Lambdas to run custom code on that data stream. This is useful in a Kinesis Pipeline that's receiving data from things like IoT devices.

And besides, Lambdas can connect to so many of the other AWS services. It basically operates as an interlink between those services.

It's useful when you want to send text messages via Amazon SMS based on a trigger. Or you can use Lambdas to create tables in DynamoDB. They can connect to Code Commit or even process your Alexa Skills and drive your home automation system via Alexa Smart Home.

At this point, you're probably wondering how you might end up developing Lambdas locally. Let's take a look at how you might do this.

When you need flexibility, Lambda is your go-to.

6. You Can Run AWS Lambdas Locally

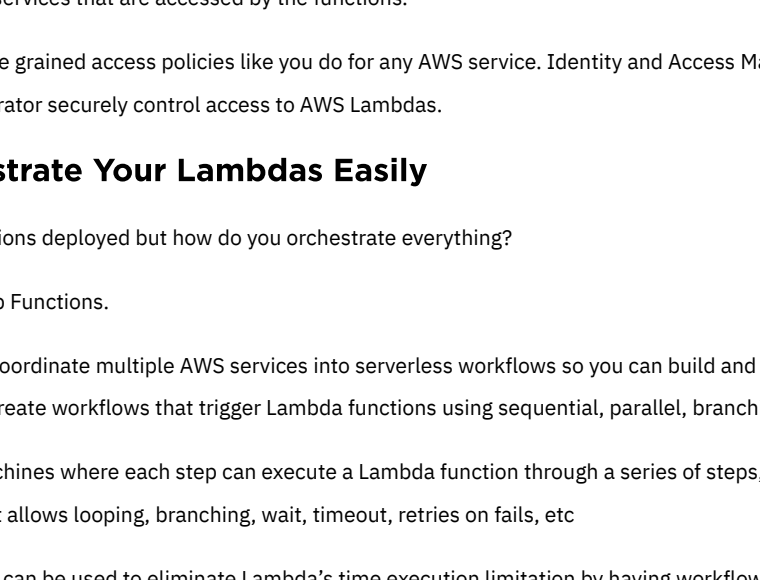
Recently, AWS introduced the SAM (serverless architecture model) CLI. This is the tool for running serverless architectures, the core of which are Lambdas, locally.

With SAM, you create a CloudFormation template to define the application. The template is written in YAML, which makes it about as clean as you can get.

Then you can invoke the Lambda locally using the SAM CLI. You'll pass to the Lambda either through the output of a SAM CLI "generate-event" command or by passing a JSON file with static data. SAM requires Python, Docker, and AWS CLI.

Another option to run Lambdas locally is the serverless framework. The serverless framework emulates AWS using Node.js. It has some limitations when it comes to Lambdas: it can only run Node.js, Python, and Java functions.

Another thing to keep in mind is that the emulation isn't perfect. There are some contextual things that won't quite translate, especially when invoking nested Lambdas. But for the most part, and for cross-cloud interoperability, it has its benefits compared to AWS SAM.



7. AWS Lambdas Are Event Driven

AWS Lambda functions gets invoked only when a particular message or event is generated. Events can be something such as an object getting uploaded to an S3 bucket or a row getting deleted from a DynamoDB table, an HTTP call to your function and so on.

Consider the following scenarios: when a user uploads a file in a bucket an event is emitted that can trigger your Lambda function. Or when a new order is placed in your online store a Lambda function can be invoked to send an email.

And this is the true power of event driven architectures: the events are triggered asynchronously so that your application will not block until the execution of the Lambda is finished.

This is the true power of event driven architectures.

8. AWS Lambda Has Strong Security Support

Security in the cloud is important for any service and AWS Lambdas are no exceptions.

AWS is responsible for protecting the infrastructure that runs lambdas but the developer is responsible for the access of the services that will trigger execution and the services that are accessed by the functions.

You can apply IAM security fine grained access policies like you do for any AWS service. Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS Lambdas.

9. You Can Orchestrate Your Lambdas Easily

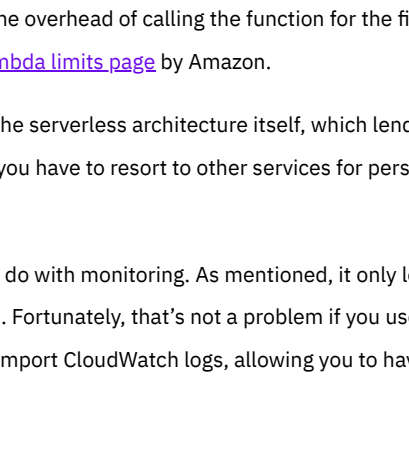
So you've got a bunch of functions deployed but how do you orchestrate everything?

You can make use of AWS Step Functions.

AWS Step Functions lets you coordinate multiple AWS services into serverless workflows so you can build and update apps quickly.
With Step Functions you can create workflows that trigger Lambda functions using sequential, parallel, branching, and error-handling steps.

Step Functions have State Machines where each step can execute a Lambda function through a series of steps, with the output of one step acting as input into the next. It allows looping, branching, wait, timeout, retries on fails, etc

But the best use case is that it can be used to eliminate Lambda's time execution limitation by having workflows of sequential function calls.



10. You Can Monitor AWS Lambda With Scalyr

You can write the lambda logs to the AWS CloudWatch as usual and then use Scalyr CloudWatch logs importer to stream your logs to Scalyr in real time.

All you need is a Scalyr Write Logs API key to get started. Try Scalyr with your AWS CloudWatch logs now.

11. You Can Leverage Pre-Built Serverless Applications

Amazon offers the AWS Serverless Application Repository, which is a managed repository that allows developers and teams to share and store reusable serverless applications.

By using the repository, your team or organization can avoid reinventing the wheel by leveraging existing components an applications that can be directly deployed to your serverless architectures, without the need to clone a repository, build an application and publish it somewhere.

AWS Lambda Limitations

There's no piece of technology without some limitations, and AWS Lambda is certainly no exception to the rule. Let's quickly cover some of its important limitations.

For starters, the Lambda function deployment package size is limited at 50MB, already zipped. Additionally, the maximum value allowed for function timeout is 900 seconds (15 minutes), being that the default is 3 seconds.

Another important limitation of AWS Lambda is the overhead of calling the function for the first time, which is often called "cold start." You can read more about these limits on the AWS Lambda limits page by Amazon.

In general, the other limitations are more about the serverless architecture itself, which lends itself well to more short-lived applications. That's why Lambda functions are short-lived, so you have to resort to other services for persistence, such as the already-mentioned DynamoDB.

A final potential limitation of AWS Lambda has to do with monitoring. As mentioned, it only logs to CloudWatch, which might create generate some friction for your monitoring needs. Fortunately, that's not a problem if you use Scalyr's log management solution. Among many valuable features, Scalyr has the ability to import CloudWatch logs, allowing you to have a centralized view of your serverless infrastructure.

Key Takeaways and Benefits

Whatever way you choose to run Lambdas, they might just be the way to go when it comes to developing serverless applications.

You can write them in different languages, you can connect several AWS components together, and you only pay for what you use—all without managing any infrastructure.

All you have to do is code the function, configure them, and deploy them. These 11 reasons should be more than enough to inspire you to consider using AWS Lambdas in your near future.

You could even start right now by pulling down the AWS SAM CLI and running it locally. Give it a try!

Like this article? Follow us on LinkedIn, Twitter, YouTube or Facebook to see the content we post.

Read more about Cyber Security

- Elasticsearch isn't the Definitive Scalable Search Engine