


# AWS Lambda Makes Serverless Applications A Reality

Ron Miller

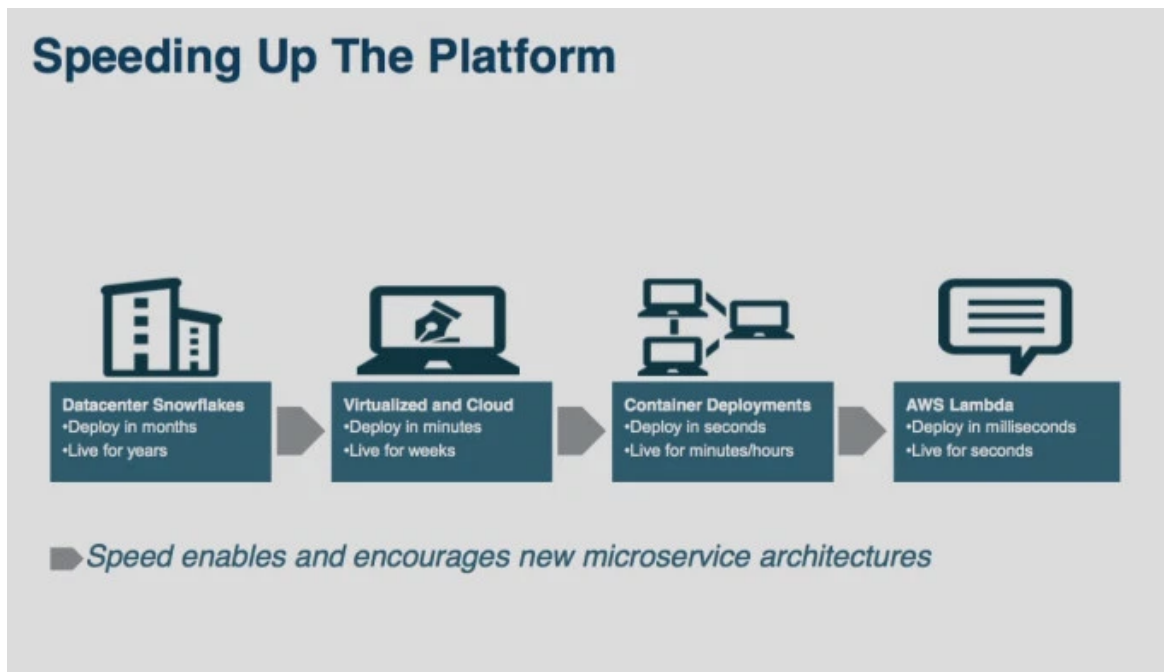
4 years



Most companies today develop applications and deploy them on servers — whether on-premises or in the cloud. That means figuring out how much server, storage and database power they need ahead of time, and deploying all of the hardware and software it takes to run the application. Suppose you didn't want to deal with all of that and were looking for a new model that handled all of the underlying infrastructure deployment for you?

**Amazon Web Services'**  [Lambda Service](#) offers a way to do just that today. With Lambda, instead of deploying these massively large applications, you deploy an application with some single-action triggers and you only pay for the compute power you use, priced in 100 millisecond increments of usage. You can have as many triggers as you like running in tandem or separately. When the conditions are met, it triggers the programmed actions.

Welcome to the world of the serverless app.



*Evolution of development (Credit: Adrian Cockcroft, Battery Ventures. Used by permission)*

Over the years, the rate of deployment speed and how long these deployments live has gone down dramatically — and Lambda reduces that to milliseconds.

We are in the midst of an evolutionary shift where Lambda encapsulates shifting developer priorities and requirements. As I wrote last year [when Lambda launched at AWS re:invent](#):

As AWS's CTO Werner Vogels pointed out, this will enable programmers to reduce their overall development effort. You simply write the code and define the event triggers, and it will run for you automatically when the conditions are met.

Triggers could be actions like a user uploading a file from a smart phone or clicking the Buy button on a website, or they could be machine-to-machine actions without humans involved. The idea is that they are flexible so just about anything can be a trigger. What's more, developers can use familiar programming tools to create the triggers, and Amazon provides a list of prewritten common ones.

Those conditions could be met every fraction of a second as with an Internet of Things scenario with sensors constantly feeding an application a stream of data or it could be weekly. For instance, suppose you have a drone feeding pictures of power lines once a week. Each time the drone uploads the photos, it triggers certain actions. When the photos are processed and stored, the work is done. You don't need a server sitting there to do that and Amazon is giving customers an

option to pay as you go. It's all about providing a simpler and easier way to deploy applications without worrying about the underlying infrastructure needed to run it.

## Brave New World

Technically no application can be serverless of course. There has to be some sort of hardware underpinning the application, but what Amazon has done with Lambda is enabled developers to automate programming to the point AWS takes care of all of the complexity related to the server deployments, storage and the database, Matt Wood general manager of product strategy at AWS explained.

To make it more understandable for mere mortals, he likened it to a file storage service like Amazon S3 or Dropbox. When you move a file to your favorite online storage service, you simply save it and forget it. You don't think, 'Oh I need to deploy a server and some storage space.' The service takes care of that for you. Similarly AWS is taking care of all the key technology for programmers without explicitly having to provision a server, database or storage to do it.

It sounds simple, but it's hard for even technical people to understand, says [Adrian Cockcroft](#), Technology Fellow at venture capitalist Battery Ventures. That's probably because it breaks the model people are used to, he says.

"Most people were baffled by Lambda, but lots of people [have been] thinking about serverless architecture. You're not scaling machines up and down. It lets the machines be invisible and is a very cost-effective architecture," he said.

He says this approach could have a number of advantages including making these programs much more secure because even if you could hack into one, you are getting at one piece of the puzzle instead of the whole kit and kaboodle — and one that lives for just fractions of a second.

## When Lambda Comes Into Play

Lambda works best in two types of scenarios, AWS's Wood says. On one end of the spectrum, you might have a situation where actions happen rather infrequently and it makes little sense to pay for servers you're not using most of the time such as the weekly drone photos scenario.

On the other end, you might be building something big and complex that needs to scale quickly and trying to deploy the infrastructure would be challenging. Suppose you have a network of weather sensors feeding you information and once that

information is collected a number of things have to happen. You could trigger an event each time the sensor sends data, and program the series of required actions, keeping mind that this is likely is quite often, measured in fractions of seconds.

One client using this technology is Major League Baseball. The trigger is the action of the pitch being thrown, the ball being hit, the runner taking off and so forth. They can then track this data in real time and Lambda deals with all of the infrastructure for them, providing as much firepower as needed at the time to capture the information and run the data. And for the six months of the year where there isn't any baseball, MLB isn't paying for infrastructure it doesn't need.

While this approach to programming isn't a magic bullet by any means, it's a new tool for developers who might not need a more traditional server set up, and it gives them options when they are designing the program and deciding how to deploy it.

"Lambda lets [developers] focus on developing applications without worrying about the heavy lifting of all the behind the scenes stuff of building the application," Wood said. And it's ushered in a world of serverless app deployment.

