JAVAWORLD

FEATURE

Serverless computing: Freedom for devs at last

Strip away your infrastructure headaches with our clear-eyed guide to serverless and the public cloud and on-premises options fueling its possibilities

By Todd Hoff

InfoWorld |

MAR 6, 2017 3:00 AM PST

On-premises serverless

Hosting serverless in your own datacenter is not dead simple yet, though it's getting easier and there are a number of options. Still, you have to be willing to set up and maintain the infrastructure.

This is a fast-moving space. Vendors are jockeying for position, trying to carve out a market in the shadow of the giant public cloud providers. Tread carefully. Every application portability promise should be viewed with skepticism.

Azure Stack

If your goal is to run on-premises like the public cloud, then Azure Stack, available mid-2017, looks to be the best bet. How well it will work is TBD, but it holds great promise and is a key differentiation between cloud vendors.

IBM OpenWhisk

OpenWhisk is IBM's FaaS that can be either hosted or on-premises. It has an API gateway, native Swift and Node.js support, plus Docker image support. It is likely to see more adoption this year as it is one of the better-supported products.

Iron.io

Iron.io was the first serverless provider, but now has to make its way in a world of big cloud providers. It offers two options: IronFunctions and Project Kratos.

Project Kratos is an embrace strategy: It enables enterprises to run AWS Lambda functionality in any cloud provider, as well as on-premises, eliminating vendor lock-in.

IronFunctions is the extend strategy: It is an open source serverless/FaaS platform based on Docker that you can run anywhere: on public, private, and hybrid clouds, even on your own laptop.

Fission.io

Fission.io is an interesting new approach to serverless that uses Kubernetes for its cloud infrastructure. Kubernetes takes care of cluster management, scheduling, and networking. There's a line of thought that Kubernetes, because it has a vibrant and productive developer community, will emerge the winner as the open source cloud infrastructure. Building a serverless product on Kubernetes makes for an interesting alternate on-premises stack.

OpenStack

OpenStack currently doesn't have native support for FaaS, but there's something in the works called Project Picasso. It has two main components: Picasso API and IronFunctions. Picasso uses the back-end container engine provided by IronFunctions.

Roll your own

Serverless is not rocket science. You can build your own serverless platform on top of your current infrastructure and get all the benefits without having to migrate to a new stack. This is a good option if you don't want to treat your datacenter like it's a cloud.

Serverless case study: Coca-Cola

How does serverless work in the real world? Here's a video, Coca-Cola: Running Serverless Applications with Enterprise Requirements, in which Michael Connor, director of technical strategy at Coca-Cola North America, explains how Coca-Cola uses AWS Lambda to process transactions from its vending machines.

Some findings:

- Serverless is nearly three times cheaper than IaaS. At about 80 million transactions per month, IaaS becomes more attractive.
- Coca-Cola now mandates serverless solutions or requires a justification for why you won't use serverless.
- Serverless doesn't solve all your problems, but it does solve problems that aren't contributing to the bottom line.
- Serverless does not lay waste to head count. It removes muck work. Patching servers doesn't help sell refreshments.

Coca-Cola measured where it spent time when using IaaS vs. serverless:

Task	IaaS	Serverless
Business projects	39%	68%
Unplanned work	24%	6%
Changes	20%	17%

Task	IaaS	Serverless
IT project work	19%	9%

Coca-Cola case study: IT time allotment when using IaaS vs. serverless

Only 39 percent of time was spent delivering business projects when using IaaS. After moving to serverless, Coca-Cola spent 68 percent of its time on business projects, with room for improvement.

After moving to Amazon there was an initial rush of spinning up servers in minutes when it used to take weeks. That rush wears off as you realize the cost of maintaining servers over their lifetime. Serverless reduces this pain, according to Coca-Cola's Connor.

Coca-Cola also wants developers developing, not performing devops duties. Moving to serverless drastically reduced the amount of ops needed, and the type of ops still in play is more in line with a developer's skill set.

The vending machine example in question involves the following:

- When you buy something at a vending machine you swipe your card.
- The transaction goes from the vending machine to a Payment Gateway.
- The Payment Gateway submits a REST API call into Amazon API Gateway.
- The API Gateway calls a Lambda function.
- The function handles all the business logic: transactions, credits, debits, and so on.
- Notifications are sent back to users via an Apple/Android Push Notification service. They in turn submit that back to Apple Pay/Android Pay, so you can see a debit in the wallet on your device.
- Then there are "buy 10 get one free" and other promotions on top of all this.

With Lambda, all this happens in under a second, and Amazon charges 1/1,000 of a penny. Plus, Coca-Cola is only getting charged if a customer executes a transaction. Otherwise, Coca-Cola pays nothing. For Lambda, getting about 30 million calls a month, the cost was $4,490 per year.

Using an AWS IaaS approach, the cost for a T2 Medium Server is about $56 per month. The fully burdened cost is nearly five times that: $250 = $56 (server) + $150 (management: patching, on call, accounting) + $30 (security: antivirus) + $14 (automation: Puppet, Chef,

Ansible). With six T2 Medium Servers the cost was $12,864 per year, making the Lambda solution 65 percent cheaper than the IaaS solution.

Of course, there is a break-even point. If you are running a huge number of transactions it becomes cheaper to run your own machines. For this example, at about 80 million hits per month, moving to IaaS started to look attractive to Coca-Cola.

One other detail to note from this case study is how serverless handles long-tail lifecycles. Let's say you push out your first 10 vending machines. With IaaS you pay $13,000 for that cluster. What about the end-of-life? When you have those last 10 machines that haven't quite sunsetted yet you are still paying $13,000 per year for IaaS. Serverless, however, ramps up and down elegantly. When you get to less than a million hits per month there's a 99 percent cost savings.

The cost of going serverless is compelling unless you're running huge volumes. How many services are you running at a lower volume rate?

Related articles

- Review: AWS Lambda redefines 'on demand'
- Microsoft Azure Functions locks in on serverless computing
- Build 'em now! 5 uses for serverless frameworks

This story, "Serverless computing: Freedom for devs at last" was originally published by InfoWorld.

Todd Hoff is a long time distributed systems programmer and creator of highscalability.com. I don't just write on this stuff; I use it. To try serverless software I actually wrote take a look at Probot.

Follow

Your cloud, your way: Why Cloud Verified matters