JAVAWORLD

ANALYSIS

What serverless computing really means

For developers, worrying about infrastructure is a chore they can do without. Serverless computing relieves that burden

By Eric Knorr
Editor in Chief, InfoWorld

JUL 11, 2016 3:00 AM PDT

It's always unfortunate to start the definition of a phrase by calling it a misnomer, but that's where you have to begin with serverless computing: Of course there will always be servers. Serverless computing merely adds another layer of abstraction atop cloud infrastructure, so developers no longer need to worry about servers, including virtual ones in the cloud.

To explore this idea, I spoke with one of serverless computing's most vocal proponents: Chad Arimura, CEO of the startup Iron.io, which develops software for microservices workload management. Arimura says serverless computing is all about the modern developer's evolving frame of reference:

> *What we've seen is that the atomic unit of scale has been changing from the virtual machine to the container, and if you take this one step further, we're starting to see something called a function ... a single-purpose block of code. It's something you can conceptualize very easily: Process an image. Transform a piece of data. Encode a piece of video.*

[A developer's guide: Serverless computing: AWS vs. Google Cloud vs. Microsoft Azure. | Then learn how to use Microsoft's Azure Functions and how to use AWS Lambda for serverless computing.]

To me this sounded a lot like microservices architecture, where instead of a building a monolithic application, you assemble an application from single-purpose services. What then is the difference between a microservice and a function?

> *A service has a common API that people can access. You don't know what's going on under the hood. That service may be powered by functions. So functions are the building block code aspect of it; the service itself is like the interface developers can talk to.*

Just as developers use microservices to assemble applications and call services from functions, they can grab functions from a library to build the services themselves -- without having to consider server infrastructure as they create an application.

AWS Lambda is the best-known example of serverless computing. As an Amazon instructional video explains, "once you upload your code to Lambda, the service handles all the capacity, scaling, patching, and administration of the infrastructure to run your code." Both AWS Lambda and Iron.io offer function libraries to further accelerate development. Provisioning and autoscaling are on demand.

Keep in mind all of this is above the level of service orchestration -- of the type offered by Mesos, Kubernetes, or Docker Swarm. Although Iron.io offers its own orchestration layer, which predated those solutions being generally available, it also plugs into them, "but we really play at the developer/API later," says Arimura.

In fact, it's fair to view the core of Iron.io's functionality roughly equivalent to that of AWS Lambda, only deployable on all major public and private cloud platforms. Arimura sees the ability to deploy on premises as a particular Iron.io advantage because the hybrid cloud is becoming more and more essential to the enterprise approach to cloud computing. Think of the consistency and application portability enabled by the same serverless computing environment across public and private clouds.

[Learn Java from beginning concepts to advanced design patterns in this comprehensive 12-part course!]

Arimura even goes as far as to use the controversial "no-ops," coined by former Netflix cloud architect Adrain Cockcroft. Again, just as there will always be servers, there will always be ops to run them. Again, no-ops and serverless computing take the developer's point of view: Someone else has to worry about that stuff, but not me while I create software.

Serverless computing, then, represents yet another leap in developer efficiency, where even virtual infrastructure concerns melt away and libraries of services and functions reduce once again the amount of code developers need to write from scratch.

Enterprise dev shops have been slow to adopt agile, CICD, devops, and the like. But as we move up the stack to serverless computing levels of abstraction, the palpable benefits of modern development practices become more and more enticing.

This story, "What serverless computing really means" was originally published by InfoWorld.

Eric Knorr is the Editor in Chief of IDG Enterprise.

Follow