opsview

Serverless Computing: A Datacenter Distraction?

Special Report: 2018

www.opsview.com

January 2018

Introduction

Unless your organization writes a lot of custom applications, serverless computing probably isn't for you in 2018.

Serverless computing in 2018 is a hot buzzword for technology analysts. In this special report, we describe what serverless computing is and what are some of its useful purposes. This paper takes a bit of a contrarian point of view to the current hype around serverless computing. While its popularity with cloud service providers is growing rapidly, it's not necessarily an everyday technology that every enterprise needs to focus on. Serverless is a nascent market, and while the technology may offer significant benefits, it's probably not something that requires mainstream focus just yet.

Most readers of this report are active in IT modernization of some sort. Any modernization strategy should include the review and understanding of serverless computing. This paper helps the reader position and prioritize serverless computing and its various derivations in their planning.

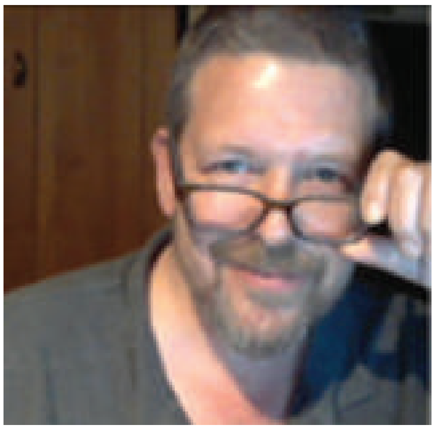
Contents

1. What you will learn

You will learn about event-driven computing, function-based computing, Functions as a Service (FaaS). The purposes and best use cases, with some examples, are discussed, to give context to the conversation. You should come away with an understanding of what are serverless technologies, where they fit in a modern IT strategy, and whether they're right for you and your organization today.

2. About the authors

John Jainschigg



I am a software developer, a DevOps advocate, and a long-time technology journalist. I recently joined Opsview from Mirantis, a provider of OpenStack and Kubernetes technologies, where I worked in Product and Partner Alliance marketing.

I have been Executive Director of the Internet and Community Lab for publisher Ziff Davis Enterprise, where I developed online products for brand and community engagement on web, mobile, video, audio, and 3D virtual reality platforms. I was Online Editor in Chief of Dr. Dobb's Journal of Software Development (United Business Media) and other market-leading b2b and b2c technology titles.

Bill Bauman

I am the Head of Innovation and Product Strategy at Opsview, an infrastructure and applications monitoring software company. I work cross-functionally with the entire organization. It's incredibly fun; sometimes daunting and humbling.

After a fun progression of technical, sales and business development roles, I realized I love telling the story of technology. I used to love technology for technology's sake, but now I'm more interested in its intersection with humanity. I like to analyze and talk about how we consume technology, and to an extent, how technology consumes us. That brought me to marketing and further on to where I am today.

I enjoy sales, marketing and engineering. I have a great time building business relationships, and working with a development team turning feedback into a roadmap for a product that people are excited to use and that truly makes their professional lives easier and better. I'm a passionate proponent of DevOps, the opportunity for people in the IT organization to expand beyond their traditional roles and contribute in multifaceted manners to everyone's success.

3. What is serverless computing?

Serverless computing is a bit of a misnomer. It seems to imply that there aren't any servers. For a workload to run, there still must be infrastructure. The term serverless, rather, refers to the abstraction of the infrastructure from the developer and the application, including the operating system itself. Where cloud computing brought about the ability for a developer or end-user to deploy their own operating system and development environments, serverless computing makes those aspects of DevOps largely unnecessary altogether.

Basic Terminology

When discussing serverless computing, two key concepts are important to understand - functions and events.

Functions are the basic building blocks of a service. Properly written, each function should be a microservice in its own right. The functions are the 'programs' that developers write to execute on serverless infrastructure.

Events are the triggers that activate a function. Events can be many different things, they can be user initiated, file uploads, form completions, file content changes, etc. Each serverless infrastructure supports different event triggers.

Event driven

A traditional application is typically a pervasive process running on an operating system. Even when it has no active users or tasks, it is consuming some sort of resource and it is a potential attack vector for an intruder to exploit. Serverless is event-driven. This means that there is a trigger that tells the application to begin running. When the serverless application has completed its task, it goes dormant, waiting for another event trigger before executing again.

Function-based and FaaS
When an application is developed to be serverless it means that it is written to execute directly against a relatively abstract plane of programming instructions provided by a service provider or an internally implemented environment to support the approach. The serverless infrastructure is a fully integrated execution environment. It provides access to a specific programming language, or languages, on top of which a developer directly executes the code they write.

Functions as a Service, or FaaS, refers to the event-driven, serverless model. When a developer writes an application that leverages a serverless environment, it becomes a function (microservice) to be executed on-demand. Much like cloud computing provides Infrastructure as a Service, that you can access on-demand, FaaS #provides functional programs that can be executed on-demand. Multiple functions could be integrated to offer a complete business service.

Function-based applications offer near instantaneous access to execute and run an application with virtually infinite scalability, depending on the infrastructure supporting the serverless environment on which they are executed.

4. Serverless Infrastructure

Most serverless development is done in the public cloud, with major providers like Amazon, Microsoft and Google leading the way. Those respective services are discussed in the section below. Less common, though growing in popularity, is on-premises serverless infrastructure. Much like private clouds, on-premises serverless solutions provide data sovereignty and privacy assurances that, for the most part, public clouds cannot.

Much like hybrid cloud infrastructure, an organization could choose to combine public and private serverless infrastructure for a hybrid serverless infrastructure. Sensitive data could remain in-house, while the need for scalable or burstable non-sensitive workloads could execute on one of the public services.

Serverless cloud providers
A multitude of public cloud providers offer serverless solutions. Of the leading public offerings, there are Amazon's AWS Lambda, Microsoft's Azure Functions, IBM's Cloud Functions and Google's Cloud Functions. Amazon and Microsoft both offer visual orchestrators, AWS Step Functions and Azure Logic Apps, respectively, however, serverless visual orchestration technologies are beyond the scope of this paper.

Amazon's AWS Lambda is a market leader in serverless infrastructure. It supports a multitude of event trigger catalysts, like their API gateway, S3 object status changes, or from DynamoDB. There are many other ways to trigger an event on Lambda, including a developer's own creativity. AWS Lambda runs all of its functions on Linux containers. Amazon has, arguably, the richest ecosystem of supporting technologies around its serverless solution. Available languages for function development include Python, JavaScript (on

Node.js), Java, and C#.

Azure Functions, from Microsoft, is another market leader in Functions as a Service. With an equally impressive list of event-driven methodologies to that of AWS Lambda, it's a viable enterprise offering. A major differentiator of Azure Functions to most other public serverless infrastructures is that the functions are executed on Windows infrastructure, as opposed to Linux. For Windows development, this means along with C# the F# language is available for the creation of functions, as well. There are currently two version of Azure Functions infrastructure on which to create functions, for a more complete breakdown of the capabilities of each, refer to: https://-docs.microsoft.com/en-us/azure/azure-functions/sup-ported-languages.

IBM Cloud Functions is based on OpenWhisk, an open source project available from the Apache Foundation. Prebuilt triggers are not as numerous as on Lambda or Azure Functions, but any external API-driven event can be used as a trigger. While JavaScript, Java and Python are supported, it also supports Swift for mobile development. There is also native Docker container integration. Having an on-premises option also offers unique hybrid infrastructure possibilities.

Google's Cloud Functions is the least mature offering of the major providers. Its event triggers are limited to its own internal event bus, or for mobile events via Firebase (a recent acquisition). Currently, JavaScript is the only supported programming language. There is no visual orchestrator available for it. While existing Google Cloud Platform customers may find use in the Cloud Functions offering, it's lack of features make it potentially less attractive than offerings from Amazon, Microsoft and IBM.

Serverless on-premises solutions

There's an evolving landscape of on-premises, and largely open source, serverless infrastructure offerings. IBM's OpenWhisk is available open source as an Apache Foundation project. In the container orchestration space, both Fission and Kubeless are active projects that support serverless methodologies on Kubernetes implementations. Another recent to market solution is OpenFaaS.

IBM's OpenWhisk is available hosted on their Bluemix cloud, as well as on-premises. There is support for JavaScript, Java, Python, and Swift, just like the hosted offering. It also has native support for Docker containers, which allows the execution of custom logic inside a container. The capabilities of OpenWhisk on-premises largely mirror those of the hosted IBM Cloud Functions, with the exception that there isn't the readily available integration with IBM machine learning and processing capabilities provided by the Bluemix cloud (without external connection).

Both Fission, by Platform9, and Kubeless, by Bitnami, aim to provide serverless infrastructure frameworks for Kubernetes. Both are under active, open source development and available on GitHub. Kubeless claims language support for Python, JavaScript, Ruby, .NET Core, and custom runtimes, while Fission claims programming extensibility to any language, but current support for Python, JavaScript, Go, C#, and PHP. Both support user-defined event triggers. With the growing popularity of Kubernetes, both offerings would seem to provide a relatively low friction onramp to serverless infrastructure in your datacenter.

OpenFaaS is a relative newcomer to the serverless infrastructure space. Their compelling claim is simple installation, claiming 1 minute to be up and running. There are numerous event triggers available, including API gateway and a function watchdog. The list of supported languages is significant, including Golang, .NET Core, Javascript, Java, Python, Ruby, and additional languages if you can build your own template. There is integrated container orchestration support with both Kubernetes and Docker Swarm.

Serverless infrastructure decisions

There are many viable public cloud and in-house infrastructure solutions for serverless workloads. Key decision making criteria include whether you're currently using any of the associated ecosystem components to any of them, as well as specifics around programming language support and features. The list above is not exhaustive, but includes the most significant offerings at the time of this paper. If you feel your organization is ready to pursue a serverless computing agenda, it's best to evaluate at least a few of the major offerings to get a feel both for their respective capabilities, as well as serverless computing in general.

5. Benefits of serverless computing

Serverless computing purports to simplify development and operations. The premise is to let developers focus on building and testing functions (small applications) that create value as opposed to managing infrastructure and operations. That means integrated into the serverless infrastructure are the programming languages, supporting applications and frameworks, scalability and security. It can also eliminate the need to develop and maintain complex deployment tooling for continuous integration and continuous delivery (CI/CD), as the conceptual model for that approach is generally inherent in serverless computing.

Operational efficiency

Operational overhead is reduced by the prescriptive, fit for purpose infrastructure. The platform is built exclusively to support event-driven functions, including event brokering, function execution and resource consumption awareness. Such a platform can be far more straightforward to maintain than traditional application infrastructure or integrated development platform hosting environments (like PaaS or container orchestration engines).

Scalability and resource consumption

Functions can be written even smaller and simpler than microservices. With the proper approach this gives functions near linear scalability and incrementally decreases resource utilization. Designed to be only running when necessary, they can, in principle, scale more cleanly and achieve higher utilization percentages than conventional application workloads running in virtual machines or containers, even microservices.

Performance and latency

The focused use case of well-written functions makes them inherently low overhead and quick to execute. There is a critique that serverless systems can be slow to dispatch a function when it is initially executed in response to an event. Depending on the platform in use, containerized functions may be on 'hot standby' until events arrive, and some platforms keep a function in memory unless there's been no activity for several minutes or longer. Depending on the configuration of the serverless environment a function can have the same or potentially lower latency than some applications.

The case for security

Functions are executed by a read only platform. They communicate via secured APIs that can't be modified. The functions are also isolated processes and have no ability to see neighboring processes in the environment around themselves. Functions can also be made very short-lived, making them resistant to attacks that require time to execute. Securing the functions themselves is mostly a matter of sanitizing and validating the inputs that provide the data and instructions they execute.

Functional cost

Serverless computing can reduce system overhead, whereby reducing system cost. In a public consumption model, serverless functions are executed on-demand, reducing the need for always-on virtual machines or containers that consume runtime but may not be providing business value 100% of the time. Serverless is a natural evolution of the on-demand model, edging ever closer to paying only for what you really need.

6. Serverless: potential caveats

As profound as the potential of serverless computing appears, the paradigm isn't a universal solution to cloud complexity.

The ongoing burden of DevOps

Solutions like Apache OpenWhisk, Fission or Kubeless, are complex platforms, on the order of a large-scale PaaS. They require modern IaaS or container orchestration engines to run on, considerable skills to integrate and stabilize, and expertise to maintain, secure, update, and lifecycle-manage.

Serverless design

Not all applications are easily converted to serverless design patterns. While it can be an interesting exercise to refactor arbitrary applications in terms of the stateless, function-oriented design patterns of serverless platforms, actual upsides can be limited.

Some applications, for example, may suffer performance hits because of the lag time required to launch a container to execute a function. Others may end up bound by the strict limits that serverless computing places on function resource utilization and execution time. Similarly, applications that appear "serverless ready," like load-balanced, horizontally scaled or containerized microservices, may not materially benefit from further refactoring as serverless functions.

In general, applications with the greatest potential for building on serverless platforms are those that are written with the intent to be event driven and relatively asynchronous.

Security concerns

While serverless computing can reduce the attack surface of a system, it doesn’t eliminate all security concerns. Your functions can still expose your databases to corruption and exposure, your attached payment and other web services to abuse. The responsibility of developers to sanitize inputs, for example, doesn’t change with the paradigm.

7. What modern IT really needs

Many of the benefits that modern IT organizations are seeking from a serverless computing agenda can be achieved with other current technologies. Typically, containers, microservices and autoscaling are three key driving factors.

Containers provide lighter weight portability and isolation than traditional virtualization, reducing system cost and allowing for greater workload density. Microservices increase application scalability and flexibility while decreasing time to deliver new features. Autoscaling can be achieved either via a public cloud provider's integrated offering, or by using a container orchestration engine (COE) or datacenter automation technology like Mesosphere DC/OS.

If you're unfamiliar with containers and microservices, download and read our **Containers as a Business Service (CaaBS) paper.** CaaBS explains modern containers and microservices from a business decision maker's perspective.

8. Monitoring the modern, hybrid datacenter

The modern datacenter is increasingly a unification of systems and processes. The physical infrastructure, the virtual infrastructure, and the processes executed on them are being combined, abstracted and sometimes obfuscated.

Every aspect of traditional and modern infrastructure should be monitored for status or problem awareness, as well as inspection and availability. A unified, extensible monitoring platform like Opsview Monitor is a necessity for a modern IT organization, whether pursuing public, on-premises, or hybrid infrastructure solutions. **Download Opsview Monitor today.**

9. Conclusion

While technologies like containers, Docker, Kubernetes, or hybrid cloud can all benefit organizations of almost any size and demeanor, the focused benefits of serverless technologies are for development -heavy organizations with significantly variable performance requirements driven by events affected by interest and seasonality. It's unlikely most organizations will be ready for a major adoption or push into serverless technologies in 2018.

There are tangible benefits to be realized by serverless technologies, but the paradigm shift is still a significant factor that must be weighed by any modern IT staff. It's entirely possible that if your organization isn't ready for the shift to serverless computing pursuing a serverless agenda right now would merely be a datacenter distraction.

Whether you use Opsview Monitor or not, if you'd like to discuss any of the topics in this paper, serverless computing, container orchestrators, or any other technology topic related to your IT modernization efforts, we're here to help. Opsview uses these modern approaches in our own development efforts and has extensive experience supporting a myriad of customers using them, as well. We're happy to listen to where you are and where you might like to be and share insight as appropriate.