

# GUIDE TO SERVERLESS TECHNOLOGIES

## **The New Stack**

### **Guide to Serverless Technologies**

Alex Williams, Founder & Editor-in-Chief

#### **Core Team:**

Bailey Math, AV Engineer

Benjamin Ball, Marketing Director

Gabriel H. Dinh, Executive Producer

Joab Jackson, Managing Editor

Judy Williams, Copy Editor

Kiran Oliver, Podcast Producer

Klint Finley, Ebook Editor

Lawrence Hecht, Research Director

Libby Clark, Editorial Director

Michelle Maher, Editorial Assistant

Sebastien Goasguen, Technical Editor

© 2018 The New Stack. All rights reserved.

20181025

# Table of Contents

Introduction ..... 4

Sponsors ..... 7

**SECTION 01 - THE SERVERLESS APPROACH TO SOFTWARE DEVELOPMENT**

01 - Serverless 101: How to Get Serverless Started in the Enterprise ..... 10

02 - Serverless Impacts on Business, Process and Culture ..... 26

KubeCon + CloudNativeCon: Creating a Standard for Serverless Events..... 34

Bibliography ..... 36

**SECTION 02 - ADOPT SERVERLESS**

03 - How Serverless Changes DevOps..... 43

04 - Serverless Security Strategies ..... 48

Stackery: Serverless for Teams ..... 56

Bibliography ..... 58

**SECTION 03 - MANAGE SERVERLESS**

05 - Migrating to Serverless in a Production Setting ..... 63

06 - Serverless Testing in Production ..... 69

07 - Serverless Pricing ..... 73

08 - Serverless Analytics..... 76

09 - Serverless Case Study of Success..... 81

Bibliography ..... 83

Closing..... 86

Appendix ..... 88

Disclosure ..... 90

# Introduction

Serverless is hard to define. No one owns the term, so everyone uses it in different ways. But what just about everyone agrees on is that although there are, of course, servers running code out there somewhere, what makes something “serverless” is that developers don’t actually have to worry about those servers.

At first this might sound “nice to have,” but not revolutionary. But think about the complexities of modern software development. Today’s programmers are burdened with performance concerns, scalability concerns, security concerns, user privacy concerns, and more. It can all be overwhelming for a beginner, or even a veteran engineer.

That’s why serverless is more than just a technology. It’s also a movement. Barriers to entry are never more apparent than when a new technology comes along and knocks them down, and serverless is doing just that. It’s providing simple ways for developers to build new software, along with an accepting community that helps them along the path towards a better understanding of their craft and, perhaps, to become professional developers. Experienced programmers, meanwhile, finally get the chance to focus on writing code, instead of managing servers or sharding databases.

It’s not just hype: Half of respondents to The New Stack’s 2018 survey said their organizations are already using serverless, and another 28 percent of respondents plan to use serverless in their organization within the next 18 months.

But as magical as these services can feel, making them work for your organization is anything but magic.

It doesn’t make sense to just take today’s monolithic enterprise applications

and foist them onto a serverless platform. First of all, it might not even work: The big name serverless platforms have some resource constraints that would probably stop most large applications from even running in their environments. But that's beside the point. Serverless platforms aren't designed to be yet another place to run the same old applications. They're a way to build software in a different, more efficient way.

More than half the respondents to our survey who are already using serverless platforms are using them for greenfield applications, which makes perfect sense. Serverless makes moving from prototype to production incredibly quick. But even for greenfield applications, there's still plenty to think about, like how new applications will talk to existing technology within an organization, like datastores and compliance tools.

Making the leap into microservices will mean more than just telling developers to start writing functions that can run on AWS Lambda or Azure Functions or some other service. It will mean rethinking the development process. To understand why, think about the way data access is handled in your organization today. Chances are, it's handled by experts on your operations (Ops) team. Now think about how data might have to flow through a serverless environment as it passes between different cloud-hosted functions. Sorting out permissions and securing data in this scenario might mean bringing your Ops team into the planning stages of application development or refactoring, instead of just tossing the finished package over the fence to them to deploy.

In short, going serverless means moving to a cloud-based, microservices-based architecture, built and maintained by a DevOps team. Yes, that's a lot of buzzwords. Let's put it another way: serverless means adopting all those cutting-edge "best practices" that you've been reading about for the past decade or more.

Yes, that could be a lot of work, especially if you’ve been putting off moving to the cloud or embracing agile development practices, or if you’re not quite sure what “microservices” even means. But the good news is that serverless makes it easier to do all of this than it’s ever been. And there are plenty of organizations that are already leading the way. For this book, we talked to companies like Fabric, which built an insurance company from the ground up on serverless, and picked the brains of the experts helping other companies make the transition into serverless to learn about common pain points, best practices, and the rich rewards they found in the process.

**Klint Finley**

Ebook Editor

**Alex Williams**

Founder and Editor-in-Chief, The New Stack

# Sponsors

We are grateful for the support of our ebook sponsors:



KubeCon + CloudNativeCon conferences gather adopters and technologists to further the education and advancement of cloud native computing. The vendor-neutral events feature domain experts and key maintainers behind popular projects like Kubernetes, Prometheus, gRPC, Envoy, OpenTracing and more.



Stackery is the complete solution for building, managing and operating serverless applications. Fast-growing companies use AWS Lambda to build architecture with limitless scalability. Stackery is the way teams build and manage serverless infrastructure quickly.

## SECTION 1

# THE SERVERLESS APPROACH TO SOFTWARE DEVELOPMENT

Serverless architecture changes the way developers build, deploy and manage applications. It also has implications for business decision-making and organizational structure. Learn the benefits and challenges of this approach.



# Contributors



**B. Cameron Gain**'s obsession with computers began when he hacked a Space Invaders console to play all day for 25 cents at the local video arcade in the early 1980s. He then started writing code for very elementary games on the family Commodore 64, and programming in BASIC on the high school PC. He has since become a long-time and steadfast Linux advocate and loves to write about IT and tech. His byline has appeared in Wired, PC World, CIO, Technology Review, Popular Science, and Automotive News.




**Mark Boyd** is a freelance writer and analyst for The New Stack and ProgrammableWeb focusing on how APIs, programmable business models, smart cities and civic tech can create a new economy where we are all co-creators in the value we receive. He works with businesses on creating compelling, actionable content and strategizing on how API-focused businesses can succeed at building new customer and partner relationships and developer communities. He also works on smart cities and civic tech projects aimed at helping grow this sector.



**Michelle Gienow** writes regularly for The New Stack, including the weekly Code Noob column. She is a frontend web developer in the making, erstwhile journalist and late-night recreational baker of peanut butter cookies.

## CHAPTER 01

# Serverless 101: How to Get Serverless Started in the Enterprise

 In the beginning, there was [bare metal](#), and it was good. Single-tenant servers were fast, reliable and secure — beholden only to their managers; verily, though also cumbersome to provision and scale. The need for agility and scalability begat virtual machines, and cloud providers brought unto us [Infrastructure as a Service](#) (IaaS), and lo, self-service in the cloud was born. Upon this fertile land arose [Amazon Web Services](#) (AWS), orchestration and [infrastructure as code](#) (IaC). Then also containerization came to pass, which begat [Platform as a Service](#) (PaaS) architecture. And lo, all was well upon the land. Well, except for developers crying forth in want of language-agnostic endpoints, horizontal scalability, and the ability to pay for the real-time consumption of services.

In response to their pleas, at last, a great gift was bestowed upon the world: serverless computing. In the serverless world, cloud providers like AWS, [Google Cloud](#) or [Microsoft Azure](#) dynamically manage the assignment and distribution of resources.

Serverless is pay-as-you-go, calculated according to actual consumption rather than pre-purchased services based on guesswork. Serverless is also accessible to people. This is infrastructure as it was meant to be: completely invisible.

## It's Not Moonbeams

Let's get one thing out of the way: Yes, the name “serverless” is confusing. Serverless computing still requires servers. There aren't, like, secret magical moonbeams powering everything.

Serverless architecture is “serverless” in terms of the user/developer never needing to take care of, or even be aware of, any individual machines — the infrastructure is fully abstracted away. Developers can simply tap into a practically limitless pool of compute, network and storage in the cloud via managed services.

“The point of calling something serverless is to say that we no longer need to run, and manage [virtual or physical] servers ourselves,” writes Peter Sbarski, Serverlessconf organizer.<sup>1</sup> “We do not even have access to them.”

The term “serverless,” which dates back to at least 2012 when [Ken Fromm](#) used the term in a ReadWrite paper entitled “[Why The Future Of Software And Apps Is Serverless](#),” arose because the server management and capacity planning decisions are completely hidden.<sup>2</sup> Serverless code can be used alongside code deployed in traditional styles, such as microservices, or applications can be written to be purely serverless and use no provisioned servers at all.

## Can You Give Me an Example?

Sure. Let's take a look at how you might use Amazon's Function as a Service (FaaS) offering Lambda, which is one of the services most associated with the serverless movement.

Let's say you're a developer responsible for maintaining a content management system (CMS) for a real estate company. Agents need to be able to add property listings with photos, and buyers need to be able to browse those listings and view the images. Your boss tells you that you need to add a new feature to the CMS that will resize the photos that agents upload to speed up page load times.

But let's say agents don't upload new photos every day, and when they upload them, they tend to upload a large number at a time.

You really don't want the whole system to bog down, resulting in a slow photo browsing experience for customers, when an agent uploads a large number of photos. Traditionally, that might mean using one big beefy server that can handle the occasional burst of image uploading. But that means committing resources to resizing photos, even though that's not something the server needs to actually do every day. So, being the serverless savvy developer you are, you decide to outsource the task to AWS Lambda.

A function is a short segment of code focused on executing a single task. Functions can be written quickly and then easily updated and replaced. Like most FaaS platforms, AWS Lambda runs functions as a stateless service, meaning the functions do not keep any state between each function invocation. They also don't store data. If they need to store or access data, they can integrate with Database as a Service offerings or serverless storage services via application programming interfaces (APIs). This is fine for your resizing function, because you want it to forget the last photo it resized before it resizes the next one anyway. The photos themselves will be accessed and saved to and from a storage system such as Amazon S3, not from Lambda itself. Events happening in the storage system trigger the functions that have been deployed. It is automated and event driven.

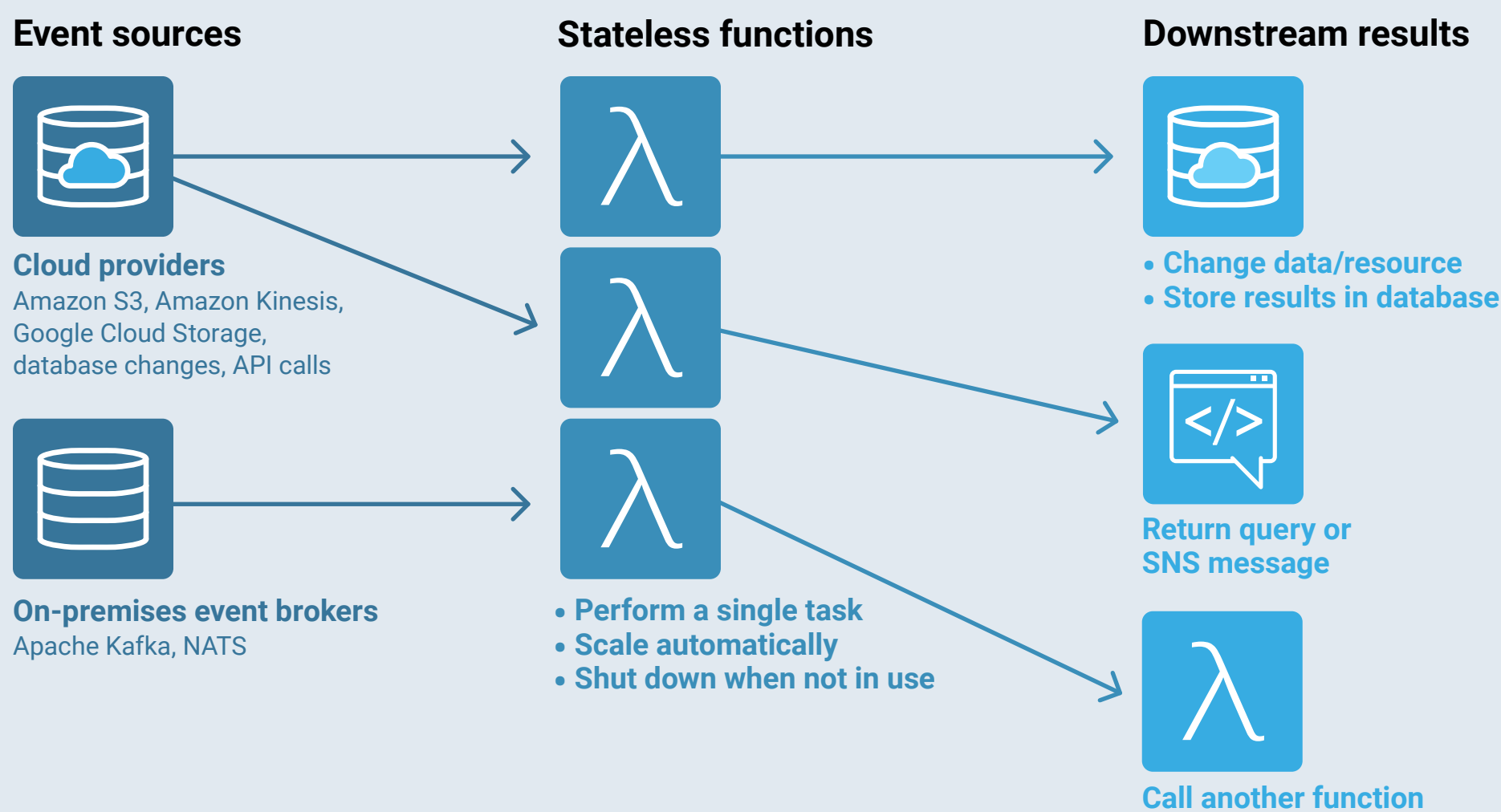
First, you write your resizing function and bundle it and its dependencies into a zip file that you upload to Lambda. Then, you'll need to make that function available to the CMS, probably through a reverse proxy service like Amazon's API Gateway. Then, the CMS can call your resizing function from Lambda as an API call when an agent uploads a photo. If your function isn't already running, Amazon will cold start an instance to resize the photo. When the function isn't in use, the service will shutdown the function, so that you're not getting charged for a server instance that's sitting around doing nothing.

Now you're able to resize images separately from the CMS serving those images to customers, so the resizing function never interferes with the performance of the CMS. You should also save money on computing resources. And, ideally, this process should be much quicker than trying to muck around in the CMS code, add a new feature, and make sure you didn't break any other code in the CMS.

Event-driven architecture (EDA) is conceptually different than the client/server architecture most commonly used today, explained [Cornelia Davis](#), senior director of technology at [Pivotal Software](#), at the Emit Conference last year.<sup>3</sup> A software component is executed when triggered by an event notification that arises from a change in state. A key feature is that the server component that sends the notification does not know which

**FIG 1.1:** All logic in an event-driven architecture is embodied in functions. Events trigger these functions and then the functions, in turn, trigger something downstream, which themselves may be functions, according to Ajay Nair, principal product manager for AWS Lambda at Amazon Web Services.<sup>4</sup>

## A Simple Event-Driven Architecture





component is on the receiving end. Events are sent and received asynchronously, which allows for greater scalability and resilience of applications built on distributed systems.

An EDA communicates through APIs and events, and the actors are ephemeral and stateless. This creates a separation of concern between the business logic implemented by the software, the data used and the state of the system. New design considerations arise for application architects and developers.

Decisions, such as the amount and type of information to include in an event, or whether to allow bi-directional discovery and where to place an event store, have broad implications for scaling and resilience. Discrepancies in how developers describe events has led the [Cloud Native Computing Foundation](#) to develop the CloudEvents specification for describing event data in a consistent way.<sup>5</sup> At the same time, serverless event-driven architecture frees developers to focus on processes. EDA decouples the processes from the stack, allowing for a more flexible architecture, said [Rob Gruhl](#), senior engineering manager of the serverless platform team at [Nordstrom](#).<sup>6</sup>

## So How is That Different from Cloud Computing?

The major difference between traditional cloud computing and serverless computing is that the developer — the customer needing said computing — doesn't pay for unused, or even underutilized, resources. Previously, we had to anticipate capacity and resource requirements and pre-provision for them, whether on an in-house data center or in the cloud. In our previous example, however, this would mean spinning up a server in AWS to stand by to execute this image resizing service at any time. In a serverless setup, however, the developer is just directing the cloud provider to spin up some code execution time when, and only when, the function is called.

The FaaS service takes the developer's functions as input, performs logic,

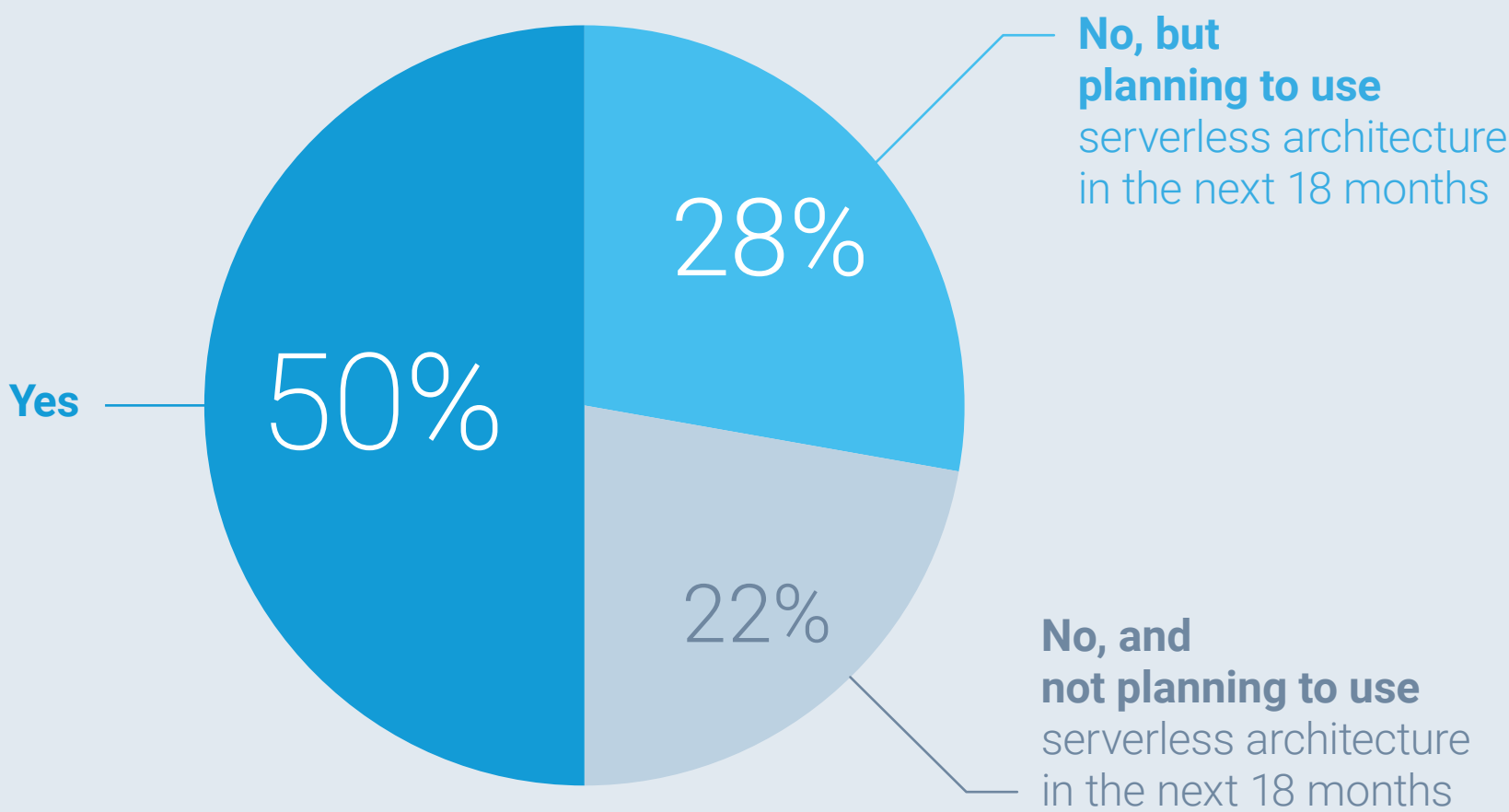
returns the output, and then shuts down. The developer is only billed for the resources used during the actual execution of those functions.

Serverless is the next evolution of a microservice architecture. Cloud providers are essentially taking what were best practices with containers and Docker, but enforcing them as part of the serverless model, explained [Erica Windisch](#), co-founder and chief technology officer (CTO) of IOpipe.<sup>7</sup> They are providing four of the factors of a traditional twelve-factor application built on a microservice architecture, and baking them into a serverless platform, she said. Those four factors — processes, concurrency, disposability and logs as event streams — are now just provided to developers as part of the serverless platform.

Pay-as-you-play, and only for resources actually consumed, is obviously a

**FIG 1.2:** Half of the survey respondents currently use serverless architecture. Their views on serverless are compared to respondents that are planning to use serverless, as well as those that do not expect to use serverless architecture.

**Survey Shows More than 75% Use or Plan to Use Serverless in Next 18 Months**



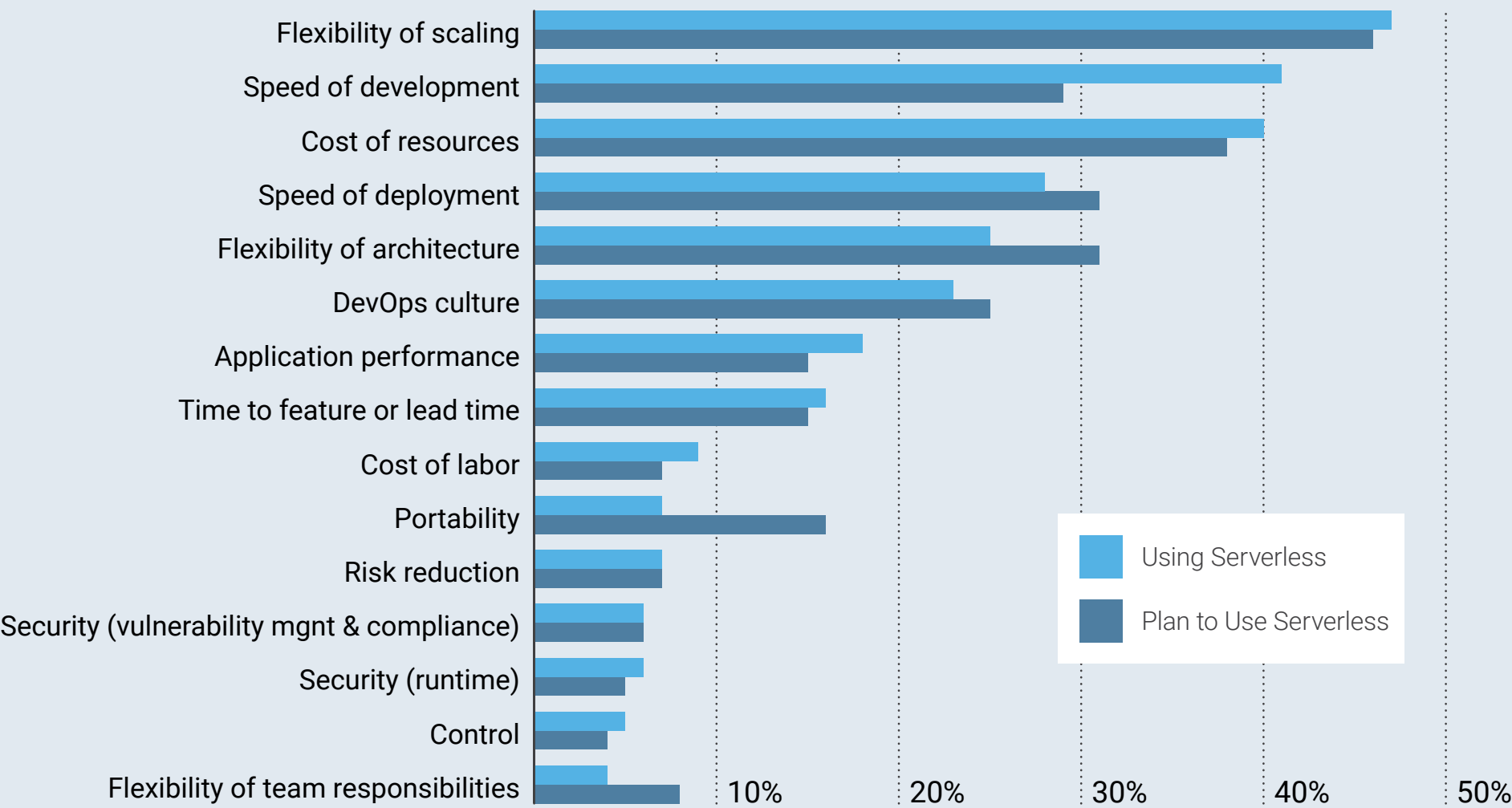
great thing. However, cloud native professionals stress that the true value of serverless is not cost efficiency, but time efficiency.

The New Stack conducted a survey in the first two weeks of August 2018, with most respondents having an interest in serverless technologies.<sup>8</sup> Four hundred seventy-nine people completed the survey, and another 129 respondents' incomplete data was included in the analysis. About half of the participants were subscribers to The New Stack newsletter. The remaining respondents were solicited via social media. Half of the survey respondents currently use a serverless architecture, and 28 percent were planning to do so in the next 18 months. Full demographic information about the respondents can be found in the appendix at the end of this book, but some highlights include:

- Twenty percent work for a company that provides cloud native or serverless services, and 18 percent work in the larger IT industry.

**FIG 1.3:** Compared to those who are only considering serverless architecture, users of serverless are more likely to say it increases the speed of development.

### Positive Impact on Software Development Life Cycle





- Thirty-eight percent work in developer or software engineering roles, and 27 percent are architects.
- Thirty-two percent work for a company with more than 1,000 employees.
- Thirty-eight percent of respondents are based in North America, and 32 percent are from Europe.

Of those serverless users, 41 percent cited speed of development as one of the top benefits of using serverless. Speed is a benefit that survey respondents who were planning serverless deployments weren't necessarily expecting, as only 29 percent indicated speed of development as a top benefit.

## Kind of Like a Time Machine?

Well, yeah, it kind of is. Serverless' time-machine power lies in shortening the time from code development to putting that code in production. It really is “here is my code, now run it” — or, as the case may be, “here is my data, now store it” — with almost no infrastructural drag in between.

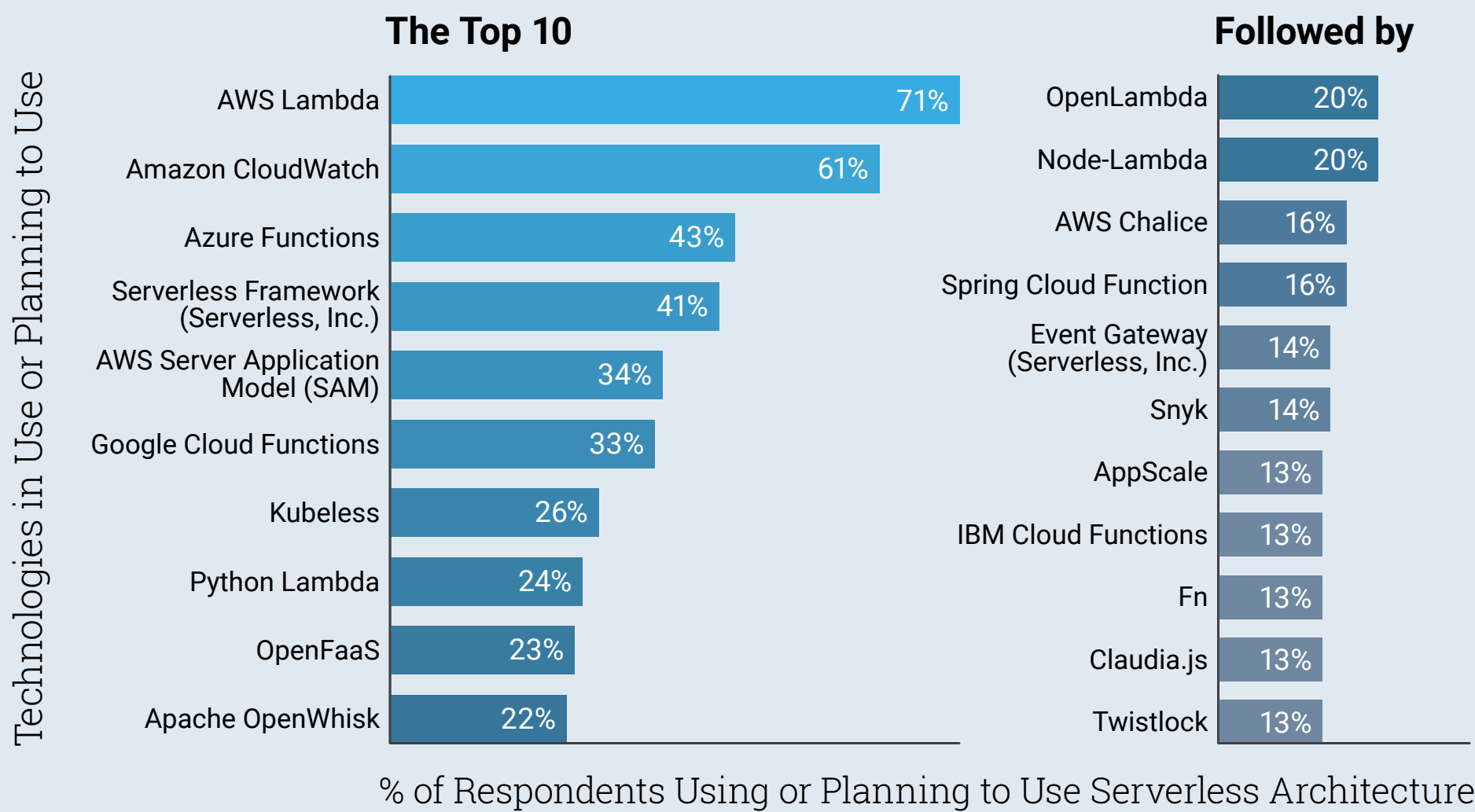
“The basic idea is a developer just writes code and pushes to a serverless service. That's it. The rest is handled by the service,” said [Chad Arimura](#), vice president of serverless at [Oracle](#). Better yet, he added, dependencies like database and storage are also services folded in seamlessly under the serverless hood.

“Behind the scenes, specialized teams combined with a lot of automation are operating these systems at scale, so that the developer doesn't have to think about this stuff,” said Arimura. “It does kind of look and feel like magic and moonbeams, which is why the hype cycle is strong with serverless. Because it's such a better experience.”

## Is Serverless More Than Just FaaS?

FaaS gets most of the attention these days, largely because it's the most

# Top Technologies on Serverless Roadmaps for Next 18 Months



Source: The New Stack Serverless Survey 2018. Q. Please indicate which of the following your organization is using or planning to use within the next 18 months. n=382. Chart shows all respondents that answered “using” or “planning to use in the next 18 months”. © 2018 THE NEW STACK

**FIG 1.4:** AWS Lambda, Azure Functions, Google Functions and IBM Cloud Functions are the top FaaS solutions. Frameworks and installable platforms also populate user shortlists.

transformative aspect of serverless. It fundamentally changes the way developers build applications. And of the FaaS products in use, AWS Lambda was the number one product on serverless roadmaps, according to our survey, with 71 percent of respondents indicating they were using or planning to use Lambda in the next 18 months.

But FaaS is only one part of a complete serverless stack. There are also serverless storage services, including block storage services, like AWS S3, and Database as a Service (DBaaS) offerings, like Amazon DynamoDB, Firebase by Google and IBM Cloudant. These are serverless in the sense that you don’t have to configure a redundant array of independent disks (RAID) or decide how to load balance your database cluster. The configuration, management and scaling of the infrastructure is all handled for you. Then, there are specialized

services delivered over APIs, like the authentication provider Auth0, and the reverse proxies you need to make your FaaS services publicly accessible.

You can think of FaaS and many of the single services as the compute component of your serverless stack, DBaaS and cloud object stores as the storage component, and API gateways as the network component.

## Feel the FaaS Platform Power

A frontend developer may find serverless fascinating as it opens new dimensions for building applications. There's no need to worry about the intricacies of building microservices on container infrastructure. Most people using Lambda and serverless technologies rely on the service to manage the complexities for them. They are not configuring containers.

“Dockerfile, infrastructure details, Kubernetes manifests — all of these are still too complicated for a developer-minded audience,” said [Sebastien Goasguen](#), co-founder at [TriggerMesh](#).

Coding functions for serverless architectures is challenging enough for developers. “Questions about serverless approaches and Kubernetes don't really come up with developers,” said [Timirah James](#), a developer advocate with [Clouduary](#), on The New Stack Analysts podcast.<sup>9</sup> Learning Kubernetes is an additional investment that James has made, but serverless platforms like [AWS Lambda](#), [Azure Functions](#) or [Google Cloud Functions](#) relieve developers of these burdens and handle resource management, load balancing and code deployment, while the developers get to just focus on their code, and enterprise organizations on their mission. These cloud services have largely become synonymous with serverless computing, as users don't need to care about the underlying infrastructure. Though technically enterprises could provide a developer-facing, on-premises serverless experience using a FaaS solution.

“Companies that are really big — like your Walmarts, your Verizons, your AT&Ts — want to own this infrastructure for security reasons, for compliance issues, for whatever,” IOpipe’s Windisch said. “Those can be valid reasons. But, for the majority of companies, if your product is not systems engineering, why are you doing it? Do as little of that as you need to.”<sup>10</sup>

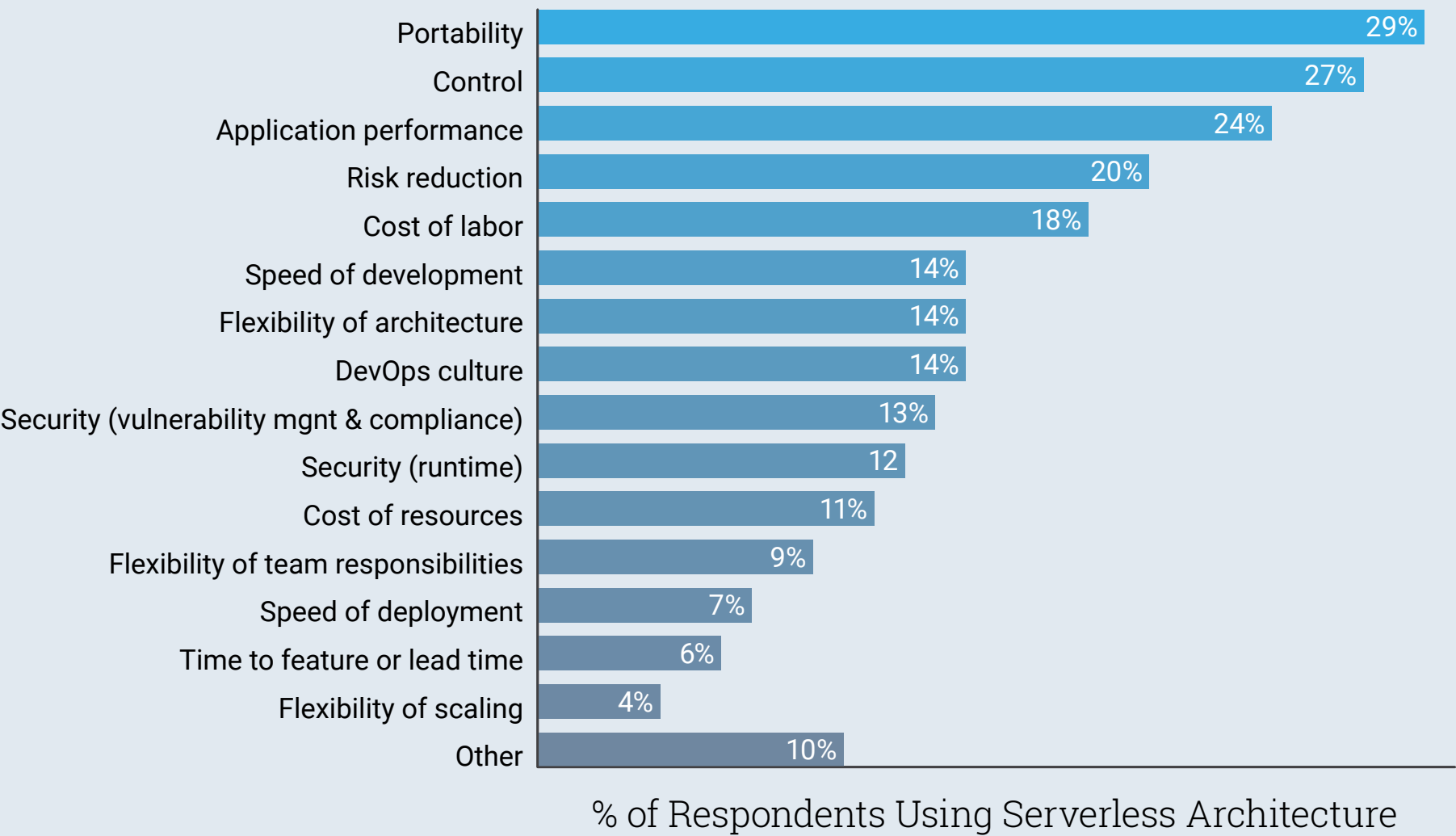
## Wait, What’s the Catch?

For all the benefits, there are some potential downsides to serverless as well. Portability, control, application performance and risk reduction were the most common shortfalls noted by our survey’s respondents who already use serverless in their organizations.

FaaS providers will typically spin down runtime environments that aren’t seeing much use. This means they also limit the total amount of resources

**FIG 1.5:** *Problems with portability and control are ongoing concerns for users of serverless architecture.*

### Areas Where Serverless Has Fallen Short



available to you, which introduces latency and potentially reduces performance. Monitoring, debugging and security can also be tricky with these cloud providers — as they would be with any cloud computing workflow — due to the fact that it all runs in a public cloud that you don't have access to or control of.

IT teams need to consider how their serverless applications will communicate with other parts of the stack. Adron Hall, developer advocate at DataStax and founder of [Thrashing Code](#), points out that while it's easy to upload functions to the cloud, it's not always so easy to move data stores, and that can create headaches. "How do we deal with our data?" he asks.<sup>11</sup> "We have 14 petabytes of data and we can't just whimsically move it somewhere."

When deciding what can be ported to a serverless infrastructure, less powerful computing performance is one drawback associated with serverless deployments. Dr. Donna Malayeri, product manager at Pulumi, points out that public cloud serverless environments, particularly FaaS platforms, have severe resource limits. "Most of them don't let you run for more than five or ten minutes," she says. "The challenge is, what if you can't run in that uniform environment?"<sup>12</sup>

Memory can also be limited in typical FaaS environments. "Anything requiring over 3GB of memory will have to be re-architected or it won't run," [Albert Qian](#), product marketing manager for the Cloud Academy, said.

But not all limitations apply to all platforms. Slow cold start, slow performance, short-lived functions and a closed set of triggers are limitations that are often assumed to universally apply to all serverless platforms, when in fact those may be attributed to implementation choices. There are newer serverless platforms, such as [Nuclio](#), which evolved to serve a broader set of use cases. These platforms have fewer restrictions, provide high performance, and can run in multiple clouds or even on premises.

Obviously, given that serverless technology is emerging and evolving day by day, not all aspects can be comfortably known entities. While not necessarily a downside, this factor definitely makes entire boards of directors squirm in their Aeron chairs.

“The geeks love it, but enterprises are still testing the water — they haven’t even gotten used to Docker [and Kubernetes] and now there is serverless,” said [Yaron Haviv](#), founder and CTO of [iguazio](#), the company behind Nuclio. “Enterprises simply have to become more agile and take risks, since the digital transformation is taking casualties in the form of incumbents who are disrupted by innovators. No one wants to be Blockbuster in the land of Netflix.”

Still, while serverless could be relatively simple to adopt, at least compared to other cutting edge technologies like Kubernetes or microservices, adopting a full-on serverless architecture in production is going to require rethinking many processes. Fortunately, as we’ll see, you can ease your way into it. You can call functions from a FaaS, or use a cloud-based data syncing tool from an existing application, without going “all in” on serverless.

## How to Know if Serverless Is Right for Your Company

It’s not about a company being a good or bad fit with serverless, say those leading the way. “Literally every company and organization that writes any software is a good fit for serverless,” said Oracle’s Arimura. “That said, current culture and distance on the journey to ‘cloud native’ can make it tougher to adopt serverless.” In other words, if a company has zero public cloud or microservice usage, then serverless is not where they should start.

Arimura says an old monolithic application might be broken into 10 microservices, which might be broken down further into 100 different serverless functions. That requires some new thinking, and opens the door to



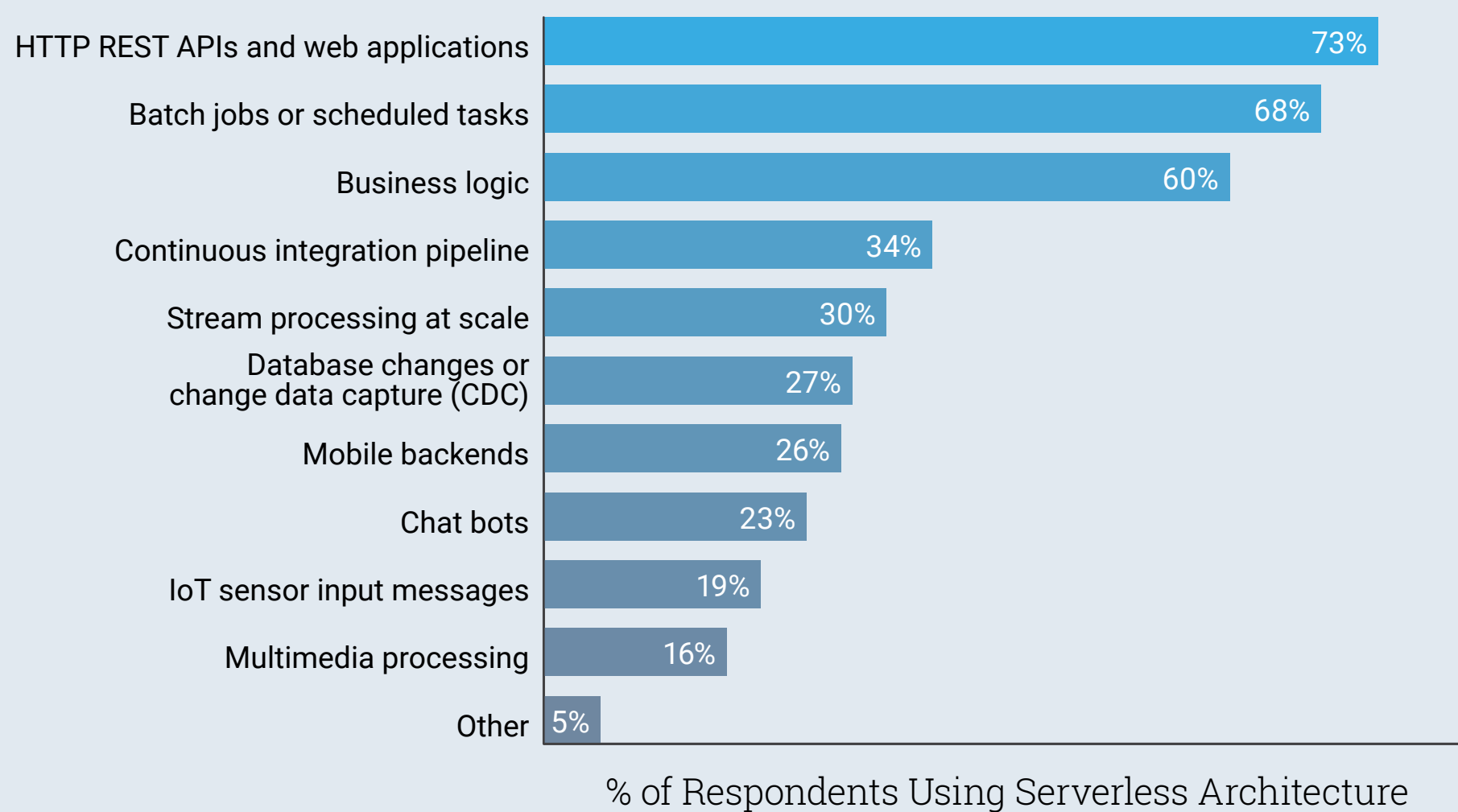
new problems. “Which is why DevOps doesn’t just become ‘NoOps’ — that is entirely the wrong way to think about it,” said Arimura. “In fact, serverless makes DevOps even more important than ever.”

As we’ll discuss in the next chapter, operations teams will often need to be brought into serverless architecture discussions at the earliest stages of a project, at least in projects that are operating at scale. But while the scale required to tap into the true economic benefits of serverless will require deep DevOps practices, teams can get started at a smaller scale.

“There is no need to take a monolith and completely convert it to microservices and/or functions,” advised Arimura. “Nor is there a need to take a company’s most important project and use that to learn a new architecture, especially if [the company’s] culture is still adapting to DevOps.”

**FIG 1.6:** Chat bots, processing of IoT sensor data and multimedia processing are each used in less than a quarter of serverless architectures.

Technical Use Cases for Serverless



In fact, noted TriggerMesh's Goasguen, most of the companies he has observed adopting serverless — AWS Lambda especially — are developer-focused organizations that were already on AWS and used it to link services together. “So chances are that if you are not on AWS today, you may not get the full benefit of serverless until other platforms provide seamless integrations with other services via a set of event triggers,” said Goasguen. “However, you should still keep it on your radar, start evaluating, identify event sources in your enterprise and see how they could be used to build complete application pipelines.”

Start small, he advised. Perhaps a few automation tasks or event-driven use cases. The most common uses among those already using serverless, according to our survey, include REST APIs, batch jobs and scheduled tasks, business logic and mobile backends. Workloads that will be executed with FaaS in the future will most likely be web services and those that support application development activities.

“ I like to think of [serverless] as a mini-PaaS for glue-like software.”

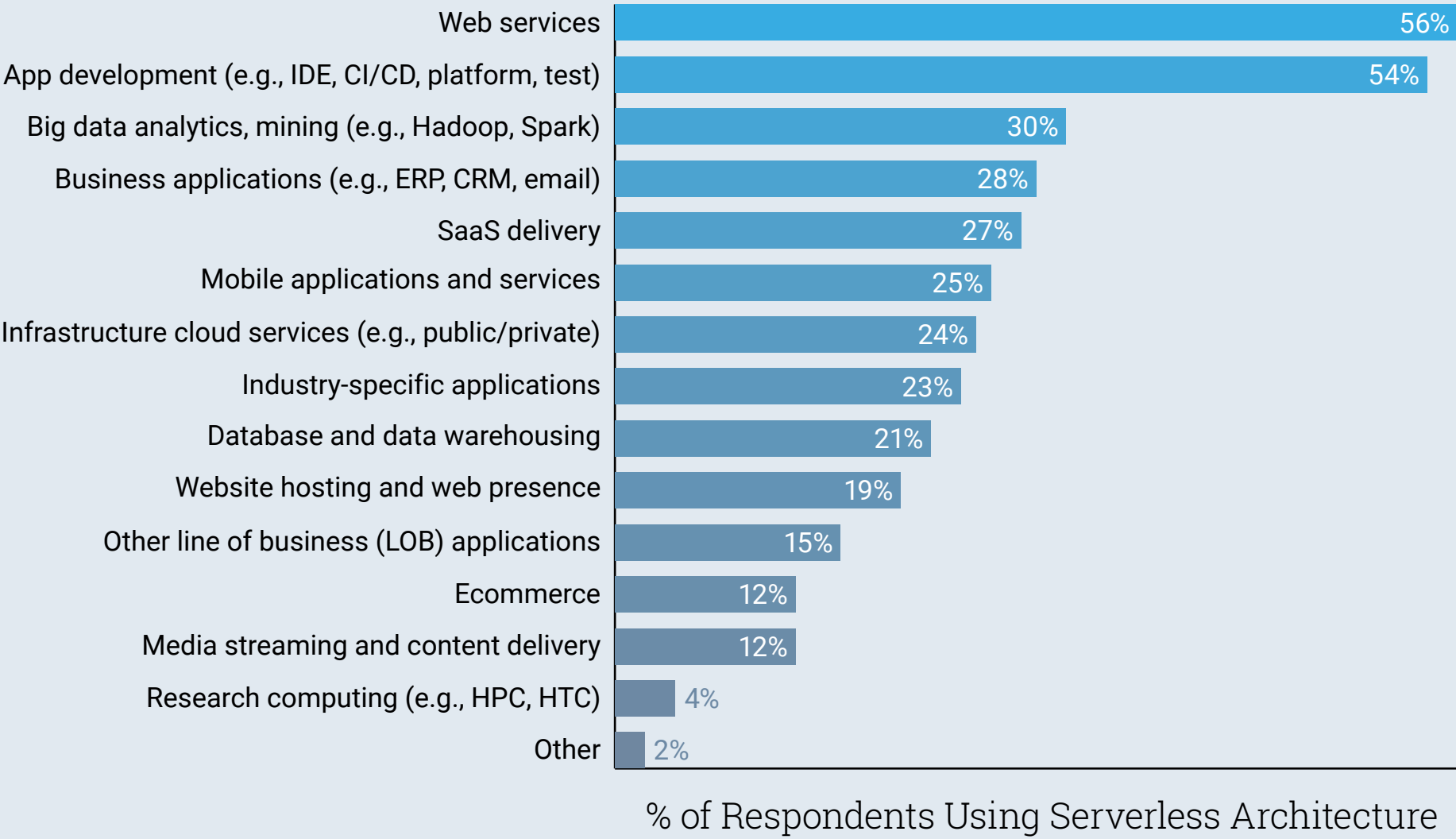
— [Sebastien Goasguen](#), co-founder of [TriggerMesh](#).

[Aaron Miller](#), a senior solutions engineer at open source API Gateway [Kong](#), suggests looking for processing tasks where the amount of resources required is unpredictable, like our image resizing example. “Anywhere where you are unsure of the capacity of the compute in the equation, [ask yourself]: Am I going to need one server or 100? If there is no way of predicting that, it is a good candidate for a serverless approach,” said Miller. “The other one is where you have high, spiky demand, and you have serious concerns about budget and have to justify costs closely.”

“If you can define your application as being event driven, then it should be serverless,” Windisch said.<sup>13</sup> “And most applications can be defined that way



# Planned Function as a Service Workloads



Source: The New Stack Serverless Survey 2018.  
Q. What types of applications or workloads does your enterprise or organization plan to use with FaaS? (check all that apply) n=328. © 2018 THE NEW STACK

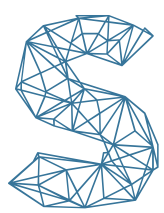
**FIG 1.7:** *Web services and application development are the workloads most likely to be used with FaaS.*

one way or another. Sometimes the technical ability isn’t there yet, like you can’t build databases on Lambda necessarily, unless you backup by S3 or something, but I think that’s kind of a good way of looking at it.”

The New Stack’s serverless ebook is here to get your company started.

## CHAPTER 02

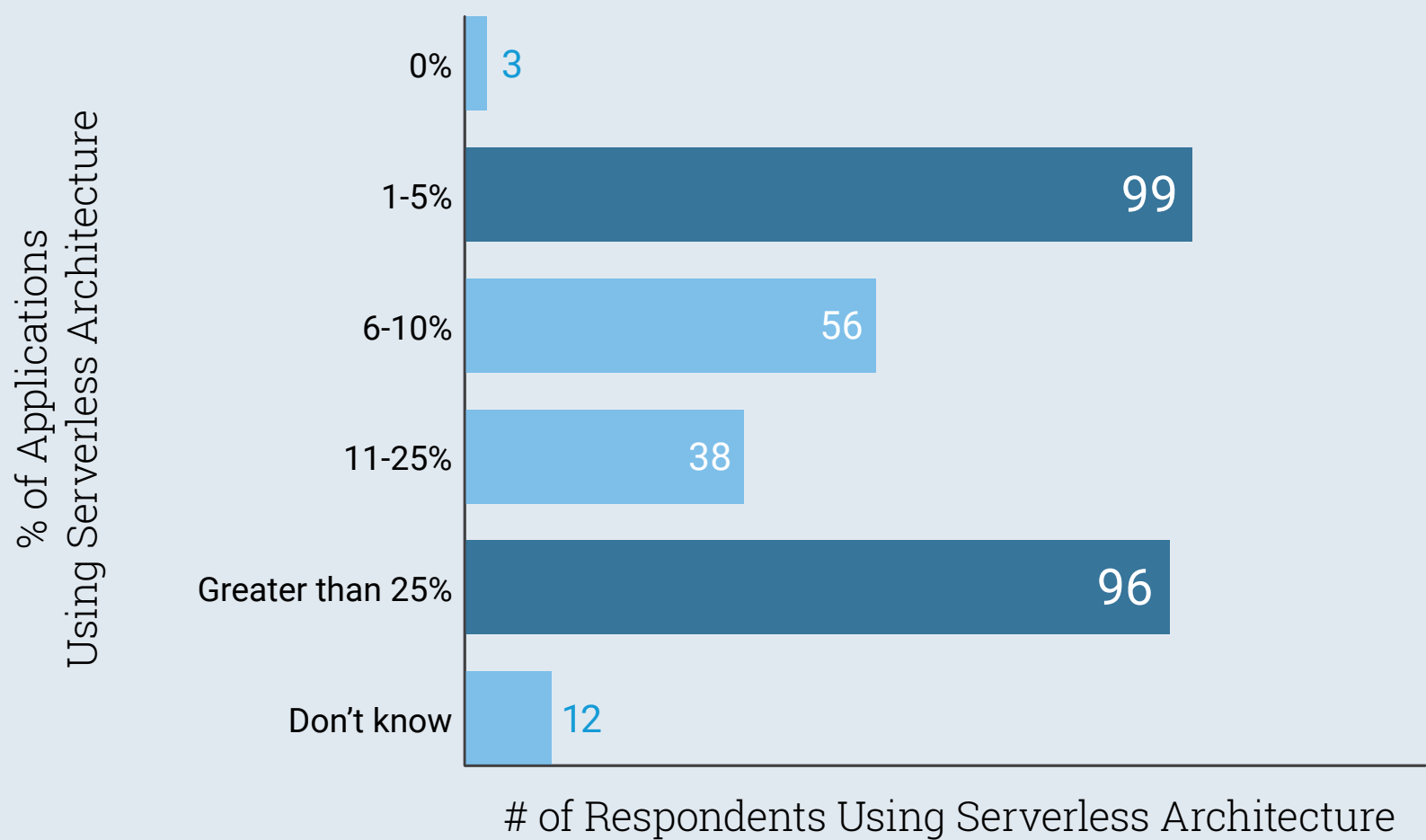
# Serverless Impacts on Business, Process and Culture



erverless technologies are gaining traction in both enterprise and startup environments, with profound effects on production cycles, speed and agility. In fact, serverless is being evaluated and adopted much more quickly than anyone anticipated — on par with the velocity of container evaluation and adoption in 2016, according to a 2018 [Cloud Foundry Foundation](#) report.<sup>14</sup> Just how fast is serverless catching on? Let's do the numbers:

- Twenty percent of backend developers are using serverless platforms, up from 16 percent a year ago. Those platforms are now reaching parity with virtual machine usage, at least according to the latest [survey from Developer Economics](#).<sup>15</sup>
- Fifty percent of respondents to The New Stack's 2018 serverless survey said their organization is already using serverless architecture, and another 28 percent said they planned to start in the next 18 months.
- Of those in our survey already using serverless, 32 percent said more than a quarter of their organization's application workloads use serverless architectures.

# Just As Many Serverless Users Run >25% of Workloads On Serverless, As Those Who Run <5%



Source: The New Stack Serverless Survey 2018.  
Q. What percentage of the total number of applications or workloads in your organization use serverless architecture? n=304. © 2018 THE NEW STACK

**FIG 2.1:** *There is room for growth among those using existing serverless architectures. At half of these organizations less than 11 percent of applications incorporate a serverless architecture.*

- Spending on FaaS saw a quarter-over-quarter growth rate of 357 percent in Q4 2017 according to Cloudability’s [State of the Cloud 2018 report](#), which analyzed the IT spend of 1,500 of its customers.

Some startups might go all-in on serverless. Half of The New Stack’s survey respondents with serverless workloads say that three-quarters of these applications execute at least one function, and some organizations are moving to integrate serverless functions into all of their applications. But enterprises are mostly still testing serverless platforms for specific use cases. Those tests are starting a domino effect that can change organizations in surprising ways. Sure that sounds scary, but remember that change can be a good thing. Done properly, serverless will mean positive changes, like lower employee turnover and happier customers.

## How Serverless Changes the Bottom Line

Let's start by talking about money. That's what [James Beswick](#) does. Beswick is the co-founder of the firm [Indevelo](#), which creates web and mobile applications, completely on serverless architectures. When starting with new customers, he usually uses the cost savings as his major focal point.

Beswick says that because serverless pricing is difficult to estimate he tends to give potential customers a range for estimates. "Web-scale apps are incredibly expensive, however you run them," said Beswick. "I tend to show, if an application is doing 10,000 transactions, what does that look like on a monthly account? If an event was costing \$500 a month, what could cause it to suddenly cost \$5,000 a month?"

Starting with cost savings also inspires businesses to think about the huge impact serverless may have on how they operate in the future. For example, in the U.S., IT projects may be budgeted based on staffing resources and capital expenditures (CapEx), rather than on pure cost allocations and operating expenditure (OpEx). A lot of enterprises see their IT as CapEx, which garners specific tax discounts. But with cloud environments, and even more so with serverless, where pricing is based on actual usage, it is all OpEx. That worries some chief financial officers, who fear losing tax write-offs. Others are not really set up to discount payroll costs of existing staff when budgeting projects.

One area that Beswick points to is that his design shop is usually hired by lines of business outside of IT: marketing and sales teams in enterprises, for example. Like with [SaaS tools that may be taken up by one department](#), but then have a data integration and mobile device implications that stretch across lines of business, application development has to find workarounds to limit the participation of the traditional IT department. "When you work with IT departments, there are a whole set of factors in play: their kingdom is disappearing, they are uncomfortable with what happens in case of a

catastrophe, there is really a disbelief that it is being run in serverless environments,” Beswick warned. “IT groups are so behind the curve on this.”

“IT has become paralyzed: They fall into the trap of not changing anything, but most businesses are seeing change happening in their industries, so they turn away from their IT departments and look externally to get things done,” Beswick says.

Beswick believes most businesses are in a middle ground where the organizational structure and role of traditional IT departments will disappear, and serverless is speeding up that move towards new organizational structures. Chris Munns, principal developer advocate for serverless at AWS, has gone so far as to predict that by 2025 most companies that are not cloud providers themselves will not need operations roles.<sup>16</sup> But the “NoOps” future is still far away.

## How Serverless Speeds Up Processes

The joy of serverless is about more than the smiles that lower IT infrastructure costs will bring to CFOs’ faces. [Aaron Miller](#), a senior solutions engineer at open source API Gateway [Kong](#), argued that while cost reduction is definitely a benefit of serverless, velocity of development is more important, and The New Stack’s survey respondents agree. Speed to deployment actually tied with cost of resources as one of the two biggest benefits of serverless amongst respondents who are already using serverless, with 41 percent citing deployment speeds as a benefit. Only flexibility of scaling, at 47 percent, was more widely cited.

Some of that speed comes from focus. Serverless platforms free teams to spend less time worrying about what version of Linux their code will run on, and more time on writing the actual code. Beswick says serverless makes prototypes or proof-of-concept projects more scalable, and therefore can actually become the foundation of real products and services. “Most of that

work would get discarded if the idea took off and got to the production stage,” he says. That means less work for developers, who are saved the work of re-architecting an application, and less effort to make sure something scales.

**“ We use serverless to decrease the time from our idea to getting the end product in our customer’s hand.”**

— [Steven Surgnier](#), co-founder and chief technology officer at insurance technology startup [Fabric](#).

But the speed of serverless also has a lot to do with its close relationship to the concept of [microservices](#). The idea behind a microservice architecture is simple. Instead of coding up one big application, a “monolith,” as developers call them, you build a collection of small services. As explained in The New Stack’s Guide to Cloud Native Microservices,<sup>17</sup> one of the many benefits of this approach to building software is that different programmers or teams of programmers can work on different features at the same time, thus reducing bottlenecks in the development process. If you’re responsible for writing an image resizing function, for example, you don’t have to wait for the programmer responsible for writing the image uploading function to finish work and move on to the next thing.

[Mike Anderson](#) of machine learning company [Algorithmia](#) showed how a typical tech enterprise’s standard release cycle could move from 65 days to 16 because of microservices.<sup>18</sup>

“Serverless is more than Function as a Service; It’s a new architecture,” said [Will Plusnick](#), developer advocate for IBM’s serverless platform based on [Apache OpenWhisk](#).<sup>19</sup>

Customers who start designing serverless architectures can come from either a grassroots up or top-down approach, Kong’s Miller said. Grassroots adopters have already transformed their legacy architecture into microservices. The



top-down adopters are those where the chief technology officers of their enterprise have been charged with a “digital transformation” agenda and are hoping that they can not only play catch up with microservices, but also get ahead of the curve with serverless.

It's easier to shift a microservice code base to a serverless platform than it is to do the same with a monolith, because the pieces that work well as serverless are easier to identify when they're already broken up into small parts. But serverless can also be a way to make the shift to microservices easier for enterprises.

[Rich Sharples](#), a senior director of product management at [Red Hat](#), says some early adopters of microservices struggle with the complexity of managing and monitoring distributed systems, a problem that dates back to the dawn of scale-out computing. Serverless is appealing because it simplifies things, while preserving many of the benefits of a microservice architecture.

## How Serverless Fosters a Culture of Autonomy — Or Vice Versa

Faster development cycles means delivering new features and bug fixes faster, which means happier customers and better business or organizational outcomes. But it should also mean happier developers.

“Engineers want to build,” Fabric’s Surgnier says. “So by providing them with a serverless infrastructure and an opportunity to focus on product, you are allowing the engineer to build more things in their career.”

But serverless doesn't just invoke happiness in programmers by making them more productive. According to research by University of Rochester psychologists Edward L. Deci and Richard M. Ryan, autonomy is a universal innate human need.<sup>20</sup> And it just so happens that one side effect of splitting monolithic applications into smaller chunks is it requires developers to have

more autonomy than they might have had in a more traditional development process.

In a monolithic environment, developers must take special care not to accidentally break their fellow developers' code. And, in turn, they must be ever vigilant lest some other developer breaks theirs. That's less of a problem in microservices or a FaaS environment, where different services and functions are completely separate and present little chance to interfere with each other. So not only are developers freed from having to worry about server configurations, they're also empowered to make changes that might otherwise take weeks or months of deliberation and testing.

"It removes this mental barrier of deploying to production," said [Steve Faulkner](#), who helped create the serverless infrastructure at media company [Bustle](#). "If you know that [it's] going to be some sort of process, it naturally introduces inertia, like, should I do it now, should I wait for someone else? But if anyone can deploy to any production endpoint at any time, it gives people autonomy and power."

Faulkner said that at Bustle, this was already part of the culture, whereas larger companies may have an ingrained mindset where developers are encouraged to wait for sign-off. "But I imagine if you are using serverless at a big enterprise, if the culture is already bad, serverless isn't going to help you."

That's where Conway's Law comes in. According to computer scientist Melvin Conway: "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure." <sup>21</sup>

"This insight tells us that we need to structure our organization to match the software that we want to produce," writes [Yan Cui](#), instructor of the course Production-Ready Serverless. <sup>22</sup> In other words, organizations that want to gain the development speed provided by serverless may need to restructure



themselves into these more autonomous units before making the shift to serverless and microservices. This structure also breeds a change of culture in which engineers own their code in production.

## Conclusion

Yes, it all sounds scary. Developers pushing code to production with full responsibility to fix it when it breaks. Traditional applications broken into hundreds or thousands of tiny services. Marketing managers charging IT services to their credit cards. But, remember that it's all in the pursuit of meeting your organization's goals by increasing productivity, saving money, and, most importantly, making better software.

# Creating a Standard for Serverless Events



The Cloud Native Computing Foundation (CNCF) last year established a serverless working group to help define serverless technologies and how they fit into a cloud native architecture. One of the first, most common issues the group addressed was the lack of an industry-wide definition for events and the associated metadata.

“Everyone defines an event a little bit differently and they don’t really take into account the context of the event,” said Ken Owens, vice president of digital architecture at MasterCard and a member of the CNCF’s serverless working group. “Being able to fire off something and forget about it and hope that it works was a common pattern we used to have. We now have more business logic that’s needed around those events to understand more about the context and the success or the failure of those events.”

The lack of a standard around events makes it difficult to troubleshoot performance issues or find business-differentiating patterns in production. It also impedes developers who need to quickly and easily receive events from other services, as well as deliver event data and integrate services across environments.

The [CloudEvents specification](#) from the CNCF’s serverless working group defines a standard event format, which includes the metadata, some extension data and the actual event payload itself.

“The goal was to find some interoperability between event systems, such that you can build consumers and producers independently of

each other but they can speak a common language,” said Chad Arimura, vice president of serverless at Oracle and a member of the CNCF’s serverless working group.

Ultimately, the goal of the serverless working group is to help identify how serverless functions and applications can and should run in cloud native environments, in order to create alignment between end users, vendors and cloud providers and promote interoperability and portability.

“The industry is working through how all of these pieces fit together for the community to be able to deploy [serverless] applications that sit on top of Kubernetes,” Arimura said. “These interfaces into Kubernetes will be things like Knative, Fn Project and OpenWhisk.”

In this podcast, learn more about the CloudEvents specification, the characteristics of event-driven architecture and some of the considerations for deploying serverless applications on cloud native, Kubernetes-based infrastructure. [Listen on SoundCloud.](#)



**[Chad Arimura](#)** is a 3x entrepreneur, most recently as co-founder and CEO of Iron.io, where he helped pioneer serverless computing. Oracle bought Iron.io in 2017 and Chad is now vice president of serverless at Oracle. He leads the Fn Project, an open source serverless FaaS platform.



**[Ken Owens](#)** is vice president of cloud native engineering at Mastercard, responsible for leading the platform services and development for cloud native platform design, engineering and development. Previously, he was CTO of cloud platforms and services at Cisco Systems and a Technical Committee representative to the CNCF.

# Bibliography

## 1. [Debunking Serverless Myths](#)

by [Dr. Peter Sbarski](#), [Serverlessconf](#) organizer and vice president of engineering and content at [A Cloud Guru](#), June 7, 2016.



*After holding the first Serverlessconf in New York, Sbarski wrote this blog post on Medium. The myths are still cited and discussed today in serverless communities.*

## 2. [Why The Future Of Software And Apps Is Serverless](#)

by [Ken Fromm](#), vice president of [Iron.io](#), ReadWrite, October 15, 2012.



*This article helped introduce the term “serverless” to describe a shift in cloud computing in which developers consume resources as services, without regard for physical capacity or limits.*

## 3. [Event-Driven Architecture Is the Wave of the Future](#)

by [TC Currie](#), The New Stack, September 11, 2017.



[Cornelia Davis](#), senior director of technology at [Pivotal Software](#), gave a talk at the Emit Conference on event-driven architecture in 2017 which The New Stack covered.

## 4. [Three Key Design Considerations for an Event-Driven Architecture](#)

by [TC Currie](#), The New Stack, September 19, 2017.



[Ajay Nair](#), principal product manager (AWS Lambda) at [Amazon Web Services](#) (AWS), and [Rob Gruhl](#), senior engineering manager of the serverless platform team at [Nordstrom](#), spoke at the Emit Conference in 2017 which The New Stack covered.



## 5. [CloudEvents](#)

by the [Cloud Native Computing Foundation](#).

*CloudEvents is a specification for describing event data in a common way in*

*order to ease event declaration and delivery.*

6. Same as #3, above.

7. **Observability for Better Applications**

by [Erica Windisch](#), CTO and co-founder of [IOpipe](#), at [ServerlessDays Portland](#), September 17, 2018.



*Windisch explained in this talk how serverless platforms now provide four factors of a twelve-factor application.*

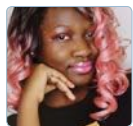
8. **2018 Serverless Survey Results**

by The New Stack, October 23, 2018.

*This is a Google Doc containing the results of The New Stack's survey conducted in the first two weeks of August 2018.*

9. **The Widening Gap Between Serverless and Kubernetes**

by [Alex Williams](#), The New Stack Analysts podcast, September 6, 2018.



*In this podcast episode, [Timirah James](#), developer advocate at [Clouduinary](#), joined The New Stack founder and Editor-in-Chief*



*Alex Williams and analyst [Klint Finley](#), a contributor at [Wired](#) and editor of this ebook, to discuss how organizations and developers both can help bridge the gap.*

SPONSOR RESOURCE

• **CNCF Serverless White Paper**

by the [Cloud Native Computing Foundation](#) (CNCF) Serverless Working Group

*This paper describes a new model of cloud native computing enabled by emerging “serverless” architectures and their supporting platforms. It defines what serverless computing is, highlights use cases and successful examples of serverless computing, and shows how serverless computing differs from, and interrelates with, other cloud application development models.*



## 10. [Serverless, Containers and Application Operations](#)

by [Alex Williams](#), The New Stack Analysts podcast, June 7, 2018.



[Erica Windisch](#), CTO and co-founder of [IOpipe](#) talks with guest host [Adron Hall](#), developer advocate at DataStax and founder of the [Thrashing Code](#) newsletter, and The New Stack founder and Editor-in-Chief Alex Williams about the architecture decisions any company is trying to make now.

## 11. [Tackling Operational Serverless and Cross-Cloud Compatibility](#)

by [Alex Williams](#), The New Stack Analysts podcast, June 21, 2018.



In this podcast episode, The New Stack founder and Editor-in-Chief Alex Williams talks with [Adron Hall](#), developer advocate at DataStax and founder of the [Thrashing Code](#) newsletter, and [Dr. Donna Malayeri](#), product manager at [Pulumi](#), about what it takes to shift to running serverless cloud functions.



12. Same as #11, above.

13. Same as #7, above.

## 14. [Why Serverless Is Being Adopted at a Faster Rate than Expected](#)

by [Chip Childers](#), co-founder of [Cloud Foundry Foundation](#), The New Stack, August 20, 2018.



In this article, Chip Childers analyzes the Cloud Foundry Foundation's June 2018 report, "Where PaaS, Containers and Serverless Stand in a Multi-Platform World."

## 15. [Developer Economics: State of the Developer Nation 15th Edition](#)

by [SlashData](#), 2018.

The 15th edition of this report surveyed more than 20,500 developers in 167 countries and includes data on serverless adoption.

## 16. [Serverless and DevOps](#)

by [Chris Munns](#), principal developer advocate for serverless at [Amazon Web Services](#), at [ServerlessDays Portland](#), September 17, 2018.



*In this talk, Munns questioned the need for DevOps, as serverless adoption rises and developers take on the responsibility for configuration.*

## 17. [Guide to Cloud Native Microservices](#)

by The New Stack, August 28, 2018.

*This ebook provides a high-level overview of what organizations should consider as they create, deploy and manage microservices for cloud native applications and lays the foundation for understanding serverless development and operations.*

## 18. [Introduction to Serverless Microservices](#)

by [Mike Anderson](#), director of growth, [Algorithmia](#), Algorithmia blog, January 3, 2018.



*Anderson explains how SoundCloud reduced a standard release cycle from 65 days to 16.*

## 19. [Will Plusnick](#), developer advocate, [IBM](#)

in “Serverless: The Missing Manual” at [ServerlessDays Portland](#) 2018.



*In this talk, Plusnick walked through how to create a serverless function and debug it on Apache OpenWhisk.*

### SPONSOR RESOURCE

- [Rapidly Build Serverless Architectures w/ Stackery](#)

by [Stackery](#)

*Add speed and scalability to your serverless infrastructure with the Stackery Operations Console. This product brief covers everything you need to accelerate your serverless development time and shorten your release cycles. Learn how to get complete visibility into your serverless applications through Stackery’s GUI or command-line interface.*

## 20. [The “What” and “Why” of Goal Pursuits: Human Needs and the Self-Determination of Behavior](#)

by [Edward L. Deci](#) and [Richard M. Ryan](#), psychology professors, University of Rochester, Psychological Inquiry, 2000.



*A review of scientific research on intrinsic motivation that states autonomy is a universal innate human need.*

## 21. [Conway’s Law](#)

by [Melvin Conway](#), computer scientist, Datamation magazine, April, 1968



*The original paper from which Conway’s Law is derived: “Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.”*

## 22. [How to Migrate Existing Monoliths to Serverless](#)

by [Yan Cui](#), principal engineer at [DAZN](#), writer and technology instructor, [Binaris blog](#), May 13, 2018.



*Cui details the experience of migrating a social network, Yubl, to serverless.*



## SECTION 2

# ADOPT SERVERLESS

When preparing to develop and deploy functions or serverless applications, DevOps and security best practices apply, but with some modifications.

# Contributors



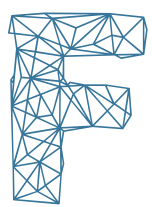
**B. Cameron Gain**'s obsession with computers began when he hacked a Space Invaders console to play all day for 25 cents at the local video arcade in the early 1980s. He then started writing code for very elementary games on the family Commodore 64, and programming in BASIC on the high school PC. He has since become a long-time and steadfast Linux advocate and loves to write about IT and tech. His byline has appeared in Wired, PC World, CIO, Technology Review, Popular Science, and Automotive News.



**Mark Boyd** is a freelance writer and analyst for The New Stack and ProgrammableWeb focusing on how APIs, programmable business models, smart cities and civic tech can create a new economy where we are all co-creators in the value we receive. He works with businesses on creating compelling, actionable content and strategizing on how API-focused businesses can succeed at building new customer and partner relationships and developer communities. He also works on smart cities and civic tech projects aimed at helping grow this sector.

## CHAPTER 03

# How Serverless Changes DevOps



For most organizations, the real question isn't whether serverless fits an organization's needs, but whether the organization can adopt serverless successfully. Organizations must be able to encourage independent, autonomous decision making amongst developer teams; must have the capacity to learn from experiments; must be willing to build in the dark where the ecosystem maturity doesn't already provide the necessary tooling; and must document architectural components in a way that other team members can pick up at their point in the workflow.

Some of these characteristics must already be in place for an organization to benefit from serverless in the first place. But adopting serverless architectures influences the way organizations work. Team dynamics change. Business processes change. Budget processes change.

Serverless is changing some of the fundamentals of how applications are designed and deployed within enterprises, said [Nate Taggart](#), CEO and co-founder of serverless DevOps tool, [Stackery](#). Traditionally, he explains, development teams designed an application, passed it through an architecture review, and then handed off to the operations (Ops) team for deployment. With the advent of microservice and serverless architectures, developers own their

code, from design through production. Development teams are also more focused on reuse — finding services that are already built to pull into the application rather than building services from scratch every time. [Erica Windisch](#), chief technology officer (CTO) and co-founder of [IOpipe](#), breaks down serverless culture this way:

- Don't build when you can buy.
- Build as little as necessary.
- Build with minimal operational complexity.<sup>1</sup>

Cloud-based video software company [Ziggeo](#) is in the thick of this transformation. The company's new [video analytics](#) feature is based on a serverless architecture, and CTO and co-founder [Oliver Friedmann](#) plans to move the entire Ziggeo stack to serverless over the next year. Moving to a completely serverless stack will help them lower operational overhead, and scale faster and more economically, he said. "Instead of operating EC2 machines with overhead to handle traffic spikes and then using auto-scaling to react to those spikes, Lambda and Fargate automatically and instantaneously scale, which provides us with much better economics and performance." CEO and co-founder [Susan Danziger](#) and Friedmann are seeing the changes to developer workflows play out firsthand and have identified at least four ways that serverless impacts business culture.

## 1. Define and Map the Data Flow

When Ziggeo's engineering teams were introducing the new video analytics feature, the first task was to define the data flow.

"First we had to define properly what each service does, and we had to map the data flow, and write the service specs in such a way that if a service crashes, the critical data doesn't disappear," said Friedmann. "All failures had to be handled gracefully."

To do this, the Ziggeo team looked at all the data and metadata present in the application and examined how it flowed through different services. “One service in that process might be an uploader, accepting a video asset from the client-side and storing it in S3,” Friedmann said. “Then, this video file might need to be processed by a transcoding service. After that, this video can be streamed by a streaming service.” Each of those states were tracked with metadata stored in databases.

The upshot is that once you know which services really need to “remember” critical data, it’s much easier to figure out which services are or can be rearchitected to what developers call “stateless,” or what the general population might call “forgetful.” These services are great candidates for FaaS platforms, which generally only run stateless functions. The benefit is that you can stop wasting money “remembering” data that isn’t necessary, plus you don’t have to worry about data from one task accidentally bleeding over to another task and creating errors.

Mapping out these data flows in advance is becoming the norm among serverless early adopters. But it’s creating some challenges. In the old application development model, Ops teams would be responsible for creating static identity and access management (IAM) policies that govern which users and applications can access what data after the application had already been designed. That’s not feasible when developers need to map data flows as they create a whole constellation of microservices. The trouble is, Friedmann says, developers don’t usually have the same level of expertise that operations teams have in setting IAM policies. So Ops now has to be a part of defining IAM policies upfront, alongside developers, as developers map the flow of data through different services.

## 2. Closer Frontend and Backend...

Two teams handled development of the Ziggeo video analytics feature once the

architecture map was defined, said Friedmann. One worked on the client, which generated analytics data, and the other worked on the AWS Lambda-based backend for receiving that data. But the two teams worked together more closely than might be expected of traditional frontend and backend teams. Taggart says this blurring of front and back skills is common in serverless environments.

### 3. ... But More Developer Independence

Projects may start with developers having more of a full stack mentality. But once the architecture is mapped and the components of a serverless stack start to get built out, each developer ends up working independently on their own service component. Here, the culture change relates back to the need for greater autonomy amongst developers.

### 4. DevOps Changes for Launch

Finally, when Ziggeo did launch the new feature, they had less work for the operations team to review. “We don’t have to worry about the servers going down. We don’t worry about making sure security patches have been updated. We don’t have to worry about traffic spikes,” Friedmann said. “[It] boils down to making sure the databases are strong enough to handle the load, so there are much fewer bottlenecks.”

In general, serverless teams work according to standard DevOps practices. But the change in workflow and pace is enough that some serverless teams have begun to break down engineers’ job roles into one of four different specialties: FaaS, services, events and queues, or infrastructure-as-code, said [Paul Johnston](#), co-founder of ServerlessDays and former CTO at tech startup Movivo, at ServerlessConf in New York last year.<sup>2</sup>

The FaaS team works quickly on tasks and tiny code units and needs to be extremely disciplined, with everyone on top of the business priorities. Services



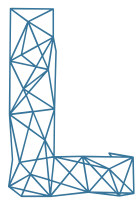
team members are all about understanding how to use services, rather than encouraging the team to “roll their own.” They need to be integrators and cloud-first in their focus, recognizing what to use and when, ensuring best practices and documenting why they used a particular service. For example, the StdLib services library lists over 500 examples in their catalog. Internal teams may have already created a number of services, while there are also directories in other serverless platforms, like AWS. The events and queues team needs to take a principled architecture approach, understanding “how to use what type of queue and when. For example, each Lambda should only do one type of data transformation,” Johnston said. The infrastructure-as-code team members are automators at their core, blurring the distinction between development and operations.

## Conclusion

At times, the changes caused by serverless can be surprising. On one hand, Dev and Ops teams must work more closely together during the design stage to ensure safe routing of data through an archipelago of services. But on the other, the two roles might need to communicate less to do the actual deployment. Instead of killing the Ops in DevOps, serverless is changing it, and making DevOps closer to what it was always meant to be: a tight collaboration between employees with overlapping but different skill sets. Still, as new specialties emerge thanks to new technologies, the relationship between Dev and Ops teams will continue to evolve.

## CHAPTER 04

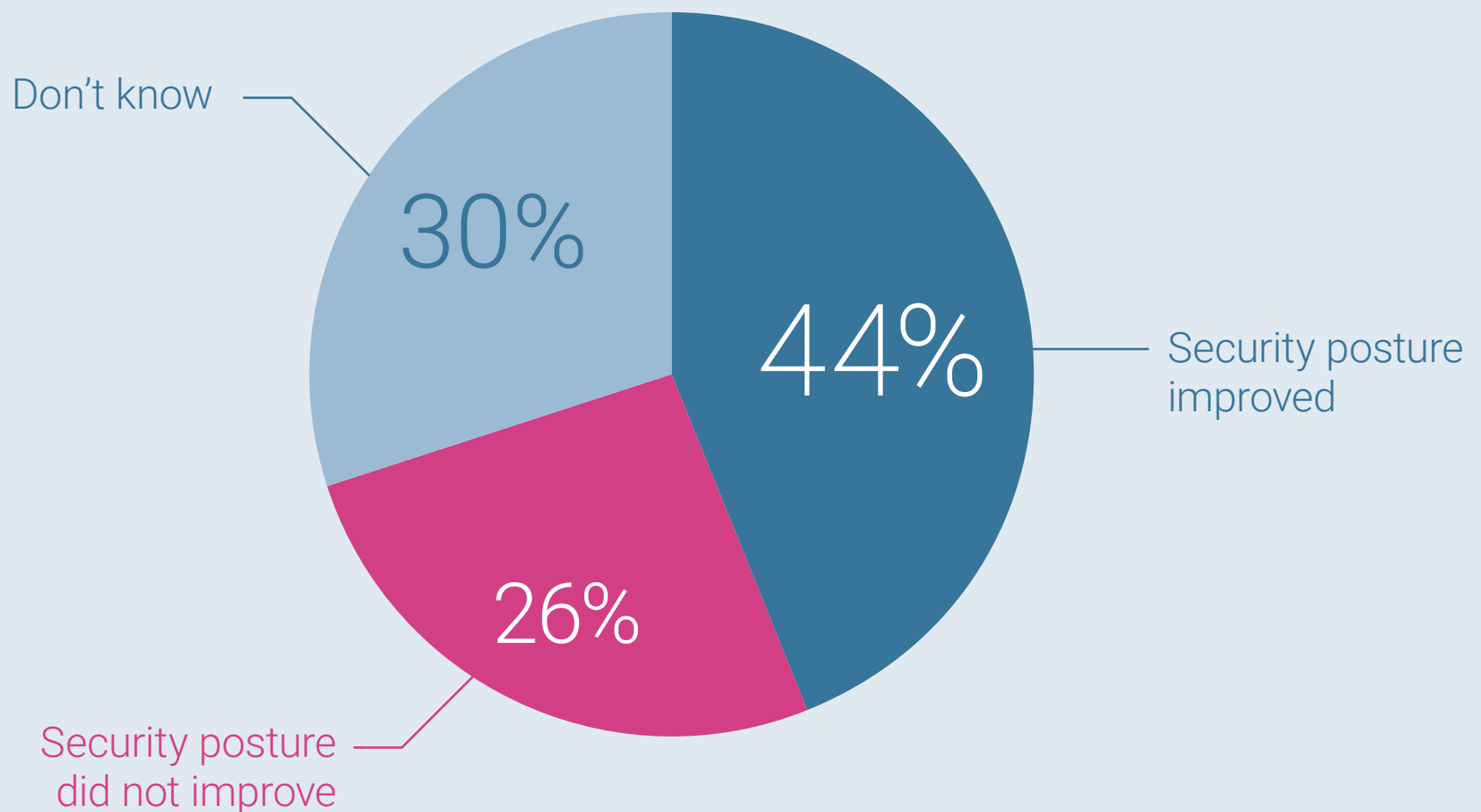
# Serverless Security Strategies

 egions of hackers are doubtless already probing serverless systems for vulnerabilities that could be exploited for fun or profit. But hey, that's nothing new. Every computing platform faces intrusion attempts, and new platforms always create new challenges. Serverless is no different. In fact, serverless even offers some new ways to keep data and applications secure.

The good news is that many of the security headaches associated with traditional technologies, such as installing security updates and maintaining networking hardware, are actually the public cloud provider's problem in a serverless environment. Serverless can be used to abstract yet another layer from the infrastructure, enabling DevOps teams to stop worrying about keeping the compiler and runtime for their language up-to-date and patched against all known security vulnerabilities.<sup>3</sup>

“You could sum up the security advantages of serverless this way: What containers did for ensuring the operating system and hardware layer could be maintained and secured separately, serverless offers the same at the application and web server layer,” writes Vince Power for Twistlock.<sup>4</sup>

## Despite Uncertainty, Positive Impact on Security Posture Among Serverless Users



Source: The New Stack Serverless Survey 2018.  
Q. Do you believe your organization's security posture has been improved by serverless? n=251.

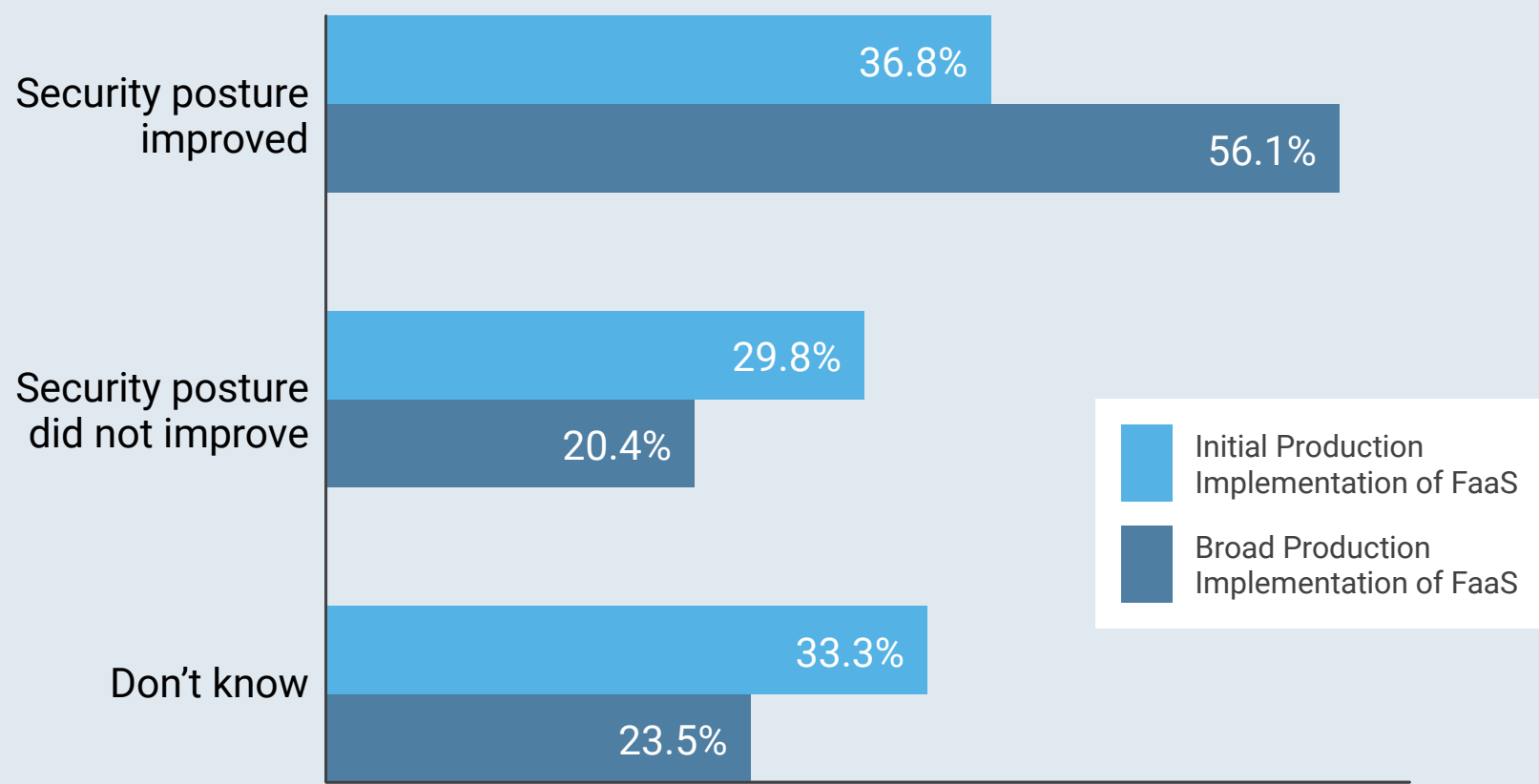
© 2018 THE NEW STACK

**FIG 4.1:** *Although there is an expectation that serverless will improve an organization's security posture, many people (30 percent) still "don't know."*

But that's also the bad news. "Security gets trickier when it comes to cloud and serverless deployments, as organizations have to rely on third parties to provide the security needed for the next-gen applications," [Holger Mueller](#), an analyst for [Constellation Research](#), said. "Each cloud provides different levels, features and capabilities to secure data and applications. Multicloud deployments can quickly become a nightmare."

But serverless does provide at least some inherent security benefits. Malicious code, for example, will not run when serverless containers are not in use and the applications are powered down. In The New Stack's 2018 serverless survey, 44 percent of respondents who already use serverless said it improved their risk posture, while only 26 percent said it had not. Organizations that have broadly implemented FaaS are more likely to see a positive impact, with 56 percent saying their security situation improved.

# Security Posture Improves With Broad FaaS Implementations



Source: The New Stack Serverless Survey 2018. Q. Do you believe your organization’s security posture has been improved by serverless? Initial Production Implementation of FaaS, n=114; Broad Production Implementation of FaaS, n=98. © 2018 [THE NEW STACK](#)

**FIG 4.2:** Organizations that have broadly implemented FaaS are more likely to see a positive impact than those with only initial implementations.

“We like the fact that after a function is invoked the environment it executes in is blown away (assuming you have it configured in this fashion),” one respondent told us. “It minimizes the surface area that can be attacked.”

## Serverless Security Challenges

Serverless security still presents challenges. The most obvious lies in the applications themselves.

“The way modern applications are built often include dependencies on other libraries, of which also depend on additional libraries,” [Albert Qian](#), product marketing manager for [Cloud Academy](#), said. “Unknown and known vulnerabilities in server libraries [will also always] attract hackers, who will always seek to inflict damage and cause harm.”

That, of course, will be an issue regardless of whether you run your code on a serverless platform or a bare metal server in your own data center. But the granular access developers have to their code is also paradoxically associated with another serverless security limitation, Qian said. As mentioned in the previous chapter, developers in serverless environments are often tasked with defining their own identity and access management (IAM) policies, but aren't necessarily experts in IAM.

This access control risk occurs when developers take the easy way out and grant the specific application being worked on access to more functions and services than is required for it to work. This practice is common in development tiers as the goal is to just make it work, but in today's model with continuous integration and continuous delivery (CI/CD), it is far too easy to have excess security propagate up to production systems. A simple example would be that in development, the application is granted full access to the data store, but it only really needs the ability to select and insert. Having the extra access to the data store to update or delete may seem harmless until the day the wrong script is run against the wrong environment and you've lost precious customer data, like orders.<sup>5</sup>

The principle of least privilege, a long held security best practice, always applies. In the future, serverless developers should learn to only grant the required access starting at the lowest tiers in development, and have separate users for each serverless function, to make traceability and data integrity much easier to guarantee. Developers and Ops teams should collaborate to define IAM policies and create processes to implement and manage them.

"The traditional way we did security, where developers wrote their code and packaged their workloads, and security operations put security controls around those workloads, just won't work for serverless," writes [Hillel Solow](#), CTO and co-founder of [Protego](#).<sup>6</sup> "Developers can't possibly keep up with the hyper-accelerated velocity they themselves created if they need to wait on security to

open ports, IAM roles or security groups for them.”

Embrace serverless for what it is, Solow said. Find the balance where developers don’t own security, but they aren’t absolved from responsibility either. Redesign how security controls are applied, so that developers have their control, but also are prevented from doing things that create risk. Use tools that let security apply the controls that they need, but in a way that doesn’t interfere with developers’ ability to change their code rapidly. Create processes that make developers aware of the security risks that their code or configuration creates, in a way that helps them resolve those risks at the speed of serverless. This blurring of the lines between DevOps and security teams is an emerging field of practice appropriately named DevSecOps, which we discuss in more detail in a previous ebook, “CI/CD with Kubernetes.” <sup>7</sup>

## Serverless Security Best Practices

DevOps teams will take a serverless platform adoption seriously and the process will invariably involve other ways to mitigate risk. Specific to security, there are numerous boxes to check off.

### ✓ Choose a Serverless Platform You Can Trust

With serverless, it is no longer necessary to maintain patches and secure servers independently. Since the Shellshock vulnerability in 2014, some operating systems introduced automatic patch updates, although many IT system administrators in enterprises are still wary of allowing their operating systems to make those decisions themselves. That may mean that some vulnerabilities go unnoticed. In a serverless environment, that risk is removed.

“The majority of successful exploits are because patches have not been updated, which is a management problem, not a technical problem,” [Guy Podjarny](#), CEO and co-founder of [Snyk](#), pointed out at ServerlessConf last year. <sup>8</sup>

Podjarny says to choose a serverless platform that you can trust to maintain



and secure servers by keeping patches up to date. He also recommends checking whether your serverless platform reuses containers and what security analysis is done on any pre-warmed containers.

Podjarny explained that in the majority of cases, a security attack is not an isolated event. What usually happens, he said, is that an attacker may install an agent on a server and leave it there. Once it has gone unnoticed for a little while, it then introduces more security code to advance another step, and so on, until the organization is compromised.

In serverless, this is less likely to happen as it is a stateless architecture system. After each instance of compute processing, the server is reset so the agent cannot take hold and keep advancing. However, one of the latent risks in serverless is that when a process starts up, containers need to be created and the application started, which introduces a latency known as cold start. To reduce this latency, some serverless services are reusing existing, pre-warmed containers. This sort of pre-warming could allow those agents to keep advancing if compromised containers are re-used.

## ✓ Monitor Code in Production

Monitor potentially vulnerable code from the outset — as soon as the migration to a serverless platform is made, said [Rani Osnat](#), vice president of marketing for [Aqua Security](#).

“The best approach is to prevent vulnerable functions in the first place by scanning them for known vulnerabilities as part of the [CI/CD] pipeline,” Osnat said.

## ✓ Write Secure Code and Keep Libraries Up-to-Date

As with development on any platform, using third-party libraries is a common and useful way to speed up development and delivery of business functionality.<sup>9</sup> But once a new serverless application is deployed, it immediately starts to age, as third-party libraries are still being updated for

new functionality and potential security vulnerabilities. Constant updates are needed to ensure the latest release of every library, in order to prevent a situation similar to the one that Capital One found for itself in 2016, where a third-party library that was out of date allowed for the exposure of millions of customers' private data.

The easiest way to mitigate this stale application problem without having to constantly update all your applications is to scan your code and build artifact repositories to identify at-risk libraries, Powers writes. Rely on a scanning tool to make better use of time and resources and to better target which applications and libraries require an update.

## ✓ Scan for Vulnerabilities

Because new vulnerabilities are constantly discovered, Osnat recommends scanning code on an ongoing basis, perhaps once per day. It's also a good idea to implement pipeline controls that prevent the deployment of vulnerable code and data, or, at the very least, to set up automated alerts sent to the DevOps team, such as via Slack or Jira, when and if vulnerabilities are detected, Osnat said.

He also recommends performance monitoring, because abnormal CPU use or extended activity for a function that normally only runs for a short time may indicate a security issue.

## ✓ Minimize Access Privileges by Default

One especially important checkbox is to remember to minimize when it comes to serverless permissions and functions, writes [John Morello](#), CTO of [Twistlock](#).<sup>10</sup> From a security perspective, it's safest to keep access privileges within your serverless environment to a minimum by default, and increase them manually where needed. (See the previous chapter's discussion on IAM policies.) It will require some work to get this right, and there is always a risk that a lack of permissions could negatively impact functionality. But the extra

effort is well worth it if it means preventing someone from taking control of your serverless functions or the external resources that they can access.

## ✓ Minimize Code

One of the reasons why serverless functions, along with other new infrastructure technologies such as containers, have become so popular is that they make it easy to break complex applications into small, discrete parts. This ability can not only improve agility, but also security.

You want to be sure to take advantage of this feature to maximize serverless security. Each function that you write should contain the minimal amount of code, permissions and external dependencies necessary to achieve its goals. Anything else is just unnecessary potential for attacks.

## Keep Ticking Those Boxes

Keeping serverless platforms secure may sound daunting. But before you have a panic attack picturing some hoodie-clad misanthrope from a bad stock photo trying to steal your organization's data, remember that most of the challenges you'll face are similar to the ones you've always faced. Most of the above checkboxes are best practices even if you're not using serverless.

# Serverless for Teams



A pay-as-you-go pricing model and increased velocity are two well-known effects that serverless technologies have on application delivery. Just as important, but often overlooked, is how serverless affects developer workflows and team dynamics. Rather than focusing entirely on business logic and code, developers take on more responsibility for configuring cloud resources.

“[Serverless] redistributes the responsibility of operations work and in the grand scheme it actually enforces the DevOps model,” Nate Taggart, CEO of Stackery, said. “Serverless is fundamentally DevOps. It’s developers having to iterate over operations work in the same cycle as their development work.”

Serverless adoption doesn’t start with the CTO, Taggart said. Instead, organizations typically adopt serverless because their engineers have started to use AWS Lambda on their own in an effort to ship features faster. The absence of governance can create resiliency and scaling problems for organizations, as individual developers bring their own workflows into a team dynamic. Operations teams quickly lose visibility and control into the application as a whole, and rollbacks and troubleshooting become a concern.

“It’s actually harder to do serverless development with two engineers than it is with one engineer. And [the problem] scales up,” Taggart said.

In order to maintain efficiency and velocity, it becomes important for an organization to set standard practices for serverless architecture. This ensures that individual workflows can scale across the

organization, and provides visibility and consistency in application delivery. For example, teams can adopt a playbook for how to roll back changes or know who shipped a commit.

In this podcast, learn how serverless architecture affects developer workflows and team dynamics, how serverless delivers on the DevOps ideal, and some of the ways that teams can begin to standardize practices around serverless application development with Stackery's serverless toolkit. [Listen on SoundCloud.](#)



**[Nate Taggart](#)** is CEO of Stackery, which offers serverless optimization software for professional serverless development. Previously, he was the second product manager at New Relic and led data science at GitHub.

# Bibliography

## 1. Observability for Better Applications

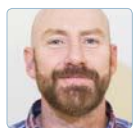
by [Erica Windisch](#), CTO and co-founder of [IOpipe](#), at [ServerlessDays Portland](#), September 17, 2018.



*Windisch explained in this talk how serverless platforms now provide four factors of a twelve-factor application.*

## 2. Why Serverless Requires Rethinking the IT Department

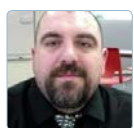
by [Mark Boyd](#), The New Stack, May 15, 2018.



*The team makeup that can work on serverless applications is considerably different from traditional IT hires, so existing job descriptions and selection criteria don't necessarily apply.*

## 3. Serverless Security Risks Laid Bare

by [Vince Power](#), [Twistlock](#), The New Stack, July 13, 2018.



*Serverless environments still need to be designed and managed with security in mind at every moment.*

## 4. Same as #3, above.

### SPONSOR RESOURCE

- CNCF Serverless White Paper

by the [Cloud Native Computing Foundation](#) (CNCF) Serverless Working Group

*This paper describes a new model of cloud native computing enabled by emerging “serverless” architectures and their supporting platforms. It defines what serverless computing is, highlights use cases and successful examples of serverless computing, and shows how serverless computing differs from, and interrelates with, other cloud application development models.*



5. Same as #3, above.

6. **Your Security Just Might Kill Your Serverless**

by [Hillel Solow](#), CTO and co-founder of [Protego](#), on The New Stack, July 27, 2018.



*The traditional way we did security, where developers wrote their code and packaged their workloads, and security operations put security controls around those workloads, just won't work for serverless.*

7. **DevOps Patterns**

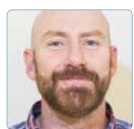
by [Rob Scott](#), site reliability engineer at [ReactiveOps](#), in “CI/CD with Kubernetes,” The New Stack, 2018.



*SecOps bridges the efforts of security and operations teams in the same way that DevOps bridges the efforts of software developers and operations teams.*

8. **Security in Serverless: What Gets Better, What Gets Worse?**

by [Mark Boyd](#), The New Stack, May 16, 2017.



*Highlights of Snyk CEO and co-founder Guy Podjarny's talk at ServerlessConf in Austin.*

9. Same as #3, above.

SPONSOR RESOURCE

• **Rapidly Build Serverless Architectures w/ Stackery**

by [Stackery](#)

*Add speed and scalability to your serverless infrastructure with the Stackery Operations Console. This product brief covers everything you need to accelerate your serverless development time and shorten your release cycles. Learn how to get complete visibility into your serverless applications through Stackery's GUI or command-line interface.*

## 10. [Serverless Security Suggestions: Tips for Securing Functions](#)

by [John Morello](#), CTO of [Twistlock](#), The New Stack, September 26, 2018.



*In some respects, serverless microservices are inherently more secure than other types of production software, Morello writes. Serverless also creates new security challenges. Here are four concerns and how to address them.*

### SECTION 3

# MANAGE SERVERLESS

Serverless, event-driven architecture requires teams to adopt new management tools and practices, due to its complexity and asynchronous nature.

# Contributors




**Mark Boyd** is a freelance writer and analyst for The New Stack and ProgrammableWeb focusing on how APIs, programmable business models, smart cities and civic tech can create a new economy where we are all co-creators in the value we receive.

He works with businesses on creating compelling, actionable content and strategizing on how API-focused businesses can succeed at building new customer and partner relationships and developer communities. He also works on smart cities and civic tech projects aimed at helping grow this sector.

## CHAPTER 05

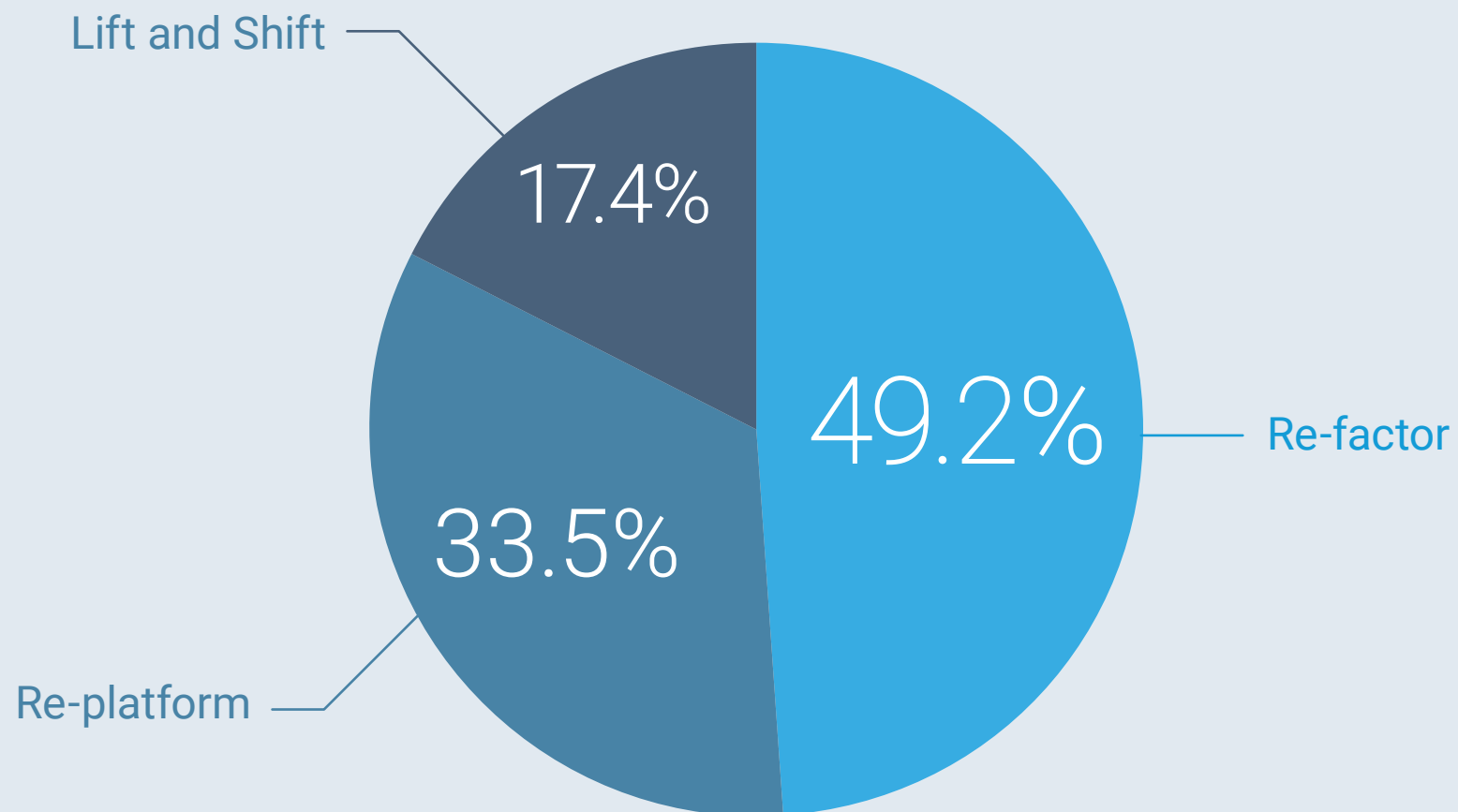
# Migrating to Serverless in a Production Setting

 If you work for a large enterprise, you might be thinking at this point that serverless sounds great, but entirely out of reach. Maybe industry regulations are holding you back, or the need to connect a legacy system, or your company's management culture, or some combination of all three. But even if your organization can't go "all in" on serverless right away, many large enterprises are still finding uses for serverless architectures, even if they can't move data to the public cloud.

In The New Stack's 2018 serverless survey, 78 percent of respondents expected serverless workloads to be primarily used in new applications, while only 22 percent expected use in an existing application. But even though a majority of applications will be greenfield, 70 percent of respondents will use at least one legacy application in a serverless architecture. Migrating existing monolithic applications requires a lot of development effort, as they can't be easily lifted and shifted to serverless environments. Instead, enterprises are carving out portions of existing applications for microservices or starting from scratch entirely.

We can learn from those who have migrated existing applications what types of applications are a good fit for serverless and the best ways to go about

## How Existing Applications Will be Migrated to Serverless Architecture



Source: The New Stack Serverless Survey 2018. For existing applications that will be used with a serverless architecture, will they primarily be "lift and shifted", "re-factored", or "re-platformed"? n=242.

© 2018 **THE NEW STACK**

**FIG 5.1:** Only 17 percent of respondents say their migrations to serverless architecture will be lift-and-shifted. About half expect to primarily re-factor applications that will then be migrated.

migrating to serverless in an enterprise environment.

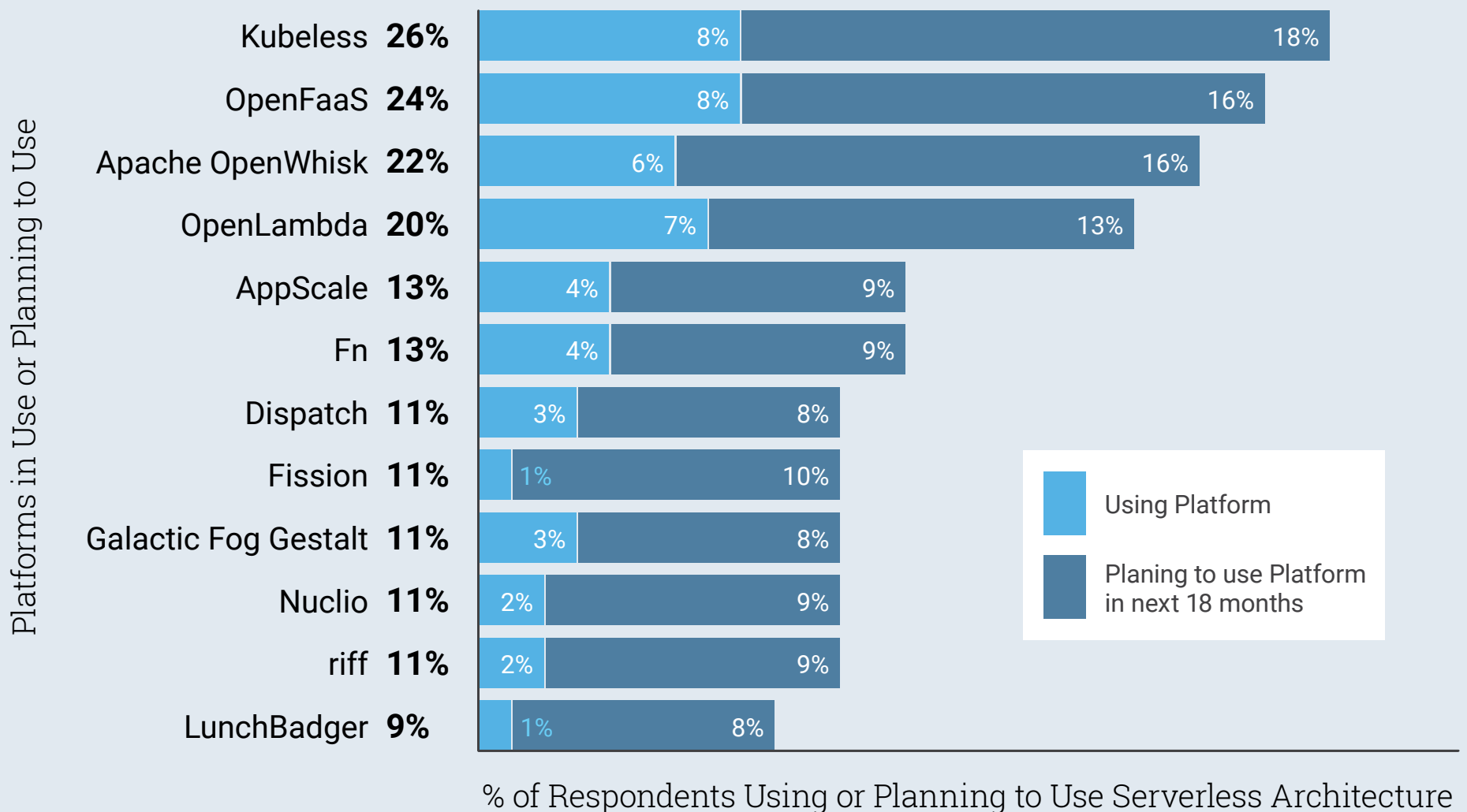
### Staying On Premises

If you're dreaming of the cloud, but stuck on premises, you're not alone. [Mika Borner](#), a data analytics management consultant with the Swiss [LC Systems](#), said large enterprises in heavily regulated industries, like finance, automotive, pharmaceutical and industrial manufacturing, tend to want to stay on-premises with their data centers, but move towards hybrid cloud solutions. That's especially true in countries like Germany and Switzerland, both of which require that companies keep their customers' financial data on servers in the same country.

Those customers are exploring platforms that can be installed either



# Installable Serverless Platforms in Use or Planned for Use



Source: The New Stack Serverless Survey 2018.

Q. Please indicate which of the following your organization is using or planning to use within the next 18 months. n=382.

© 2018 THE NEW STACK

**FIG 5.2:** *Thirty-two percent of those using or planning to use a serverless platform have a Kubernetes-based installable platform in their plans, with Kubeless often being cited alongside other options.*

on-premises or in a cloud. Kubeless, OpenFaaS, OpenLambda and Apache OpenWhisk were the most popular platforms organizations already had in use or were planning to use in the next 18 months, according to The New Stack’s survey. A common theme among many of these platforms is that they are built on top of containers, with several touting Kubernetes as part of their core infrastructure. Kubeless, Fission and Riff are Kubernetes native, as is [Knative](#), a recently announced project that was written-in by several respondents.

## Understanding the Use Cases

As with any new technology, the first step is determining clear use cases.

“Our enterprise customers know their processes better than we do. So we try to understand what they are doing,” said Borner. That means requirements

gathering, setting goals and determining restrictions.

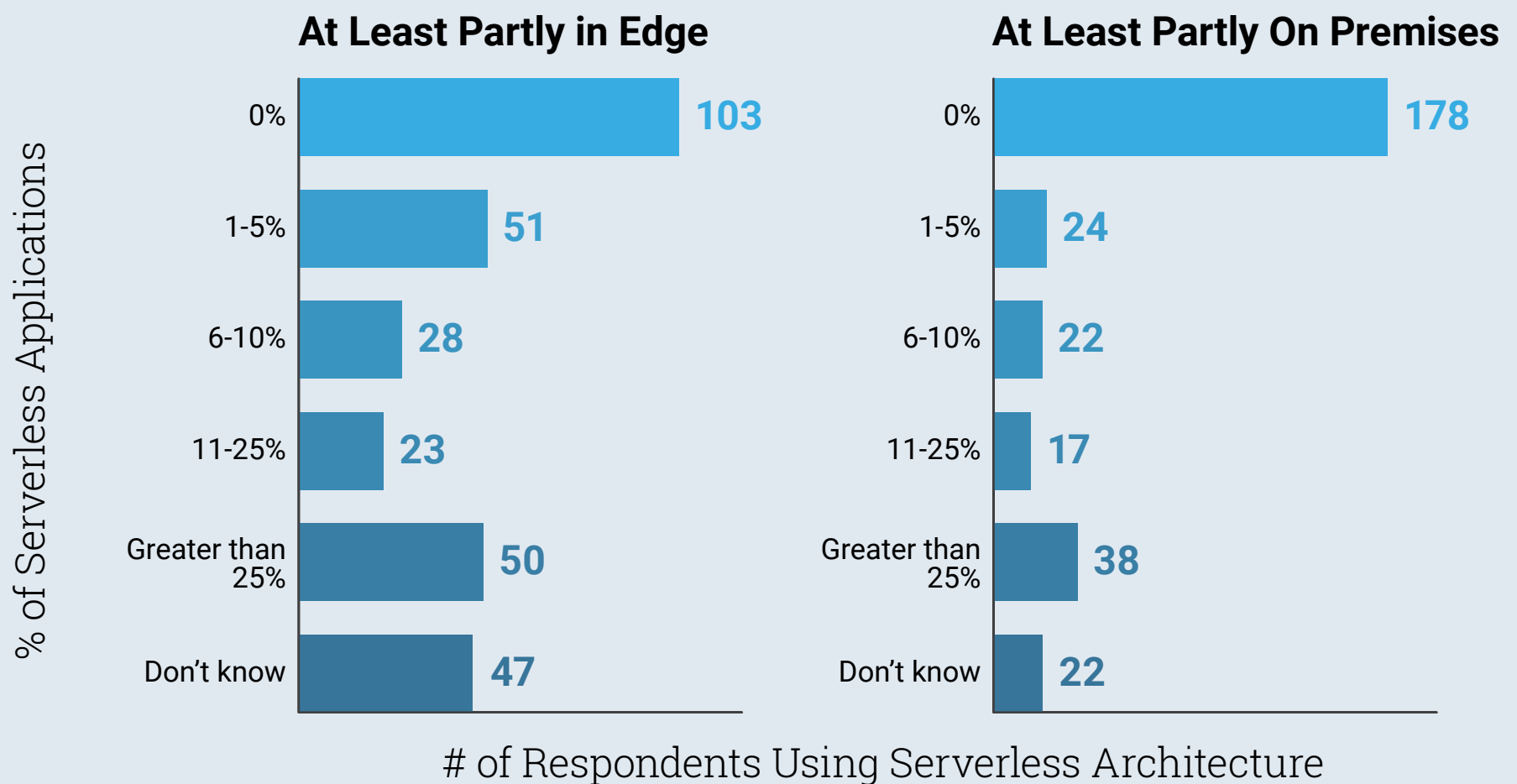
Borner says in general, when a company is talking about measuring the quality of the components, pills or products that can be produced, they listen for whether the company's goal is about numbers being created in one day or components being created in milliseconds. The first goal relates more to batch processing techniques, while the second is about real-time data analytics, and therefore more suited to a serverless model.

“As soon as we go to a real-time process we go to serverless,” said Borner. From there, LC Systems starts creating a proof of concept (PoC) or pilot example.

Borner cites one common industry example from pharmaceuticals and industrial manufacturing. Both are fitting cameras to their product lines and want to be able to take photos of pills or new components as they are made, compare the photo of the new creation with a quality example of best practice, and split off those that don't make the quality cut.

[Iguazio](#) founder and CEO [Asaf Somekh](#) confirms that real-time analysis is a big part of what Nuclio customers look for. Somekh gives the example of telecommunications companies, which have a need for real-time monitoring of their network infrastructure. “With all those networking monitoring functions, dealing with a DDoS attack, for example, can become very reactive when a network gets overloaded,” Somekh explained. Because of this, a lot of telcos want to introduce artificial intelligence (AI) into their platforms so they can monitor their network infrastructure and apply predictive algorithms. But that requires them to ingest sampling data collected from network devices, compare that data against historical records of network failures and attacks in the past, and then predict risks that will happen within an hour or the next couple of hours. “That allows network operators to prevent the failures by changing their configurations,” he says. “It is predictive maintenance on a network.”

## Where Serverless Workloads Run



Source: The New Stack Serverless Survey 2018. Q. Of the applications or workloads utilizing serverless architecture, what percentage execute compute at least in part in an “edge” environment? n=302. Q. Of the applications or workloads utilizing serverless architecture, what percentage execute at least in part in an on-premises environment? n=301.

© 2018 THE NEW STACK

**FIG 5.3:** Half of serverless organizations utilize edge computing. Stream data processing and IoT sensor input messages are two technical uses cases when an application may operate at least partly at the edge.

Cloud services can be used to train the algorithms, then a serverless workflow can be put in place close to the edge where the data is being collected, and that serverless workflow would apply prediction models as the data is collected. In fact, according to The New Stack’s survey, half of all serverless architectures have at least one application that executes compute at an edge location.

Once this infrastructure is in place, said Somekh, it is easier to then keep collecting data and update the predictive or analytical model every three to six hours. “It depends on the statistical variance, but there is not a lot of overload in uploading an updated model,” he said.

To help enterprises get started, Somekh said the first stage of the serverless onboarding process involves mapping an enterprise’s current infrastructure, including the application programming interfaces (APIs) they’re using. The

idea is to help them use Nuclio without having to move data from one location to another, meaning customers can keep their data on-premises if they want.

Another place to start, writes serverless course instructor [Yan Cui](#), is by identifying discrete services within an existing application.<sup>1</sup> Services, Cui writes, are autonomous, have clear boundaries, own their own data, and are the authoritative source for that data. Services are good candidates to run as functions, whether in a public cloud or on premises.

Cui suggests starting with services that would create real business value if refactored. Which services are real bottlenecks or resource hogs today? Services with unpredictable usage are good candidates as well.<sup>2</sup>

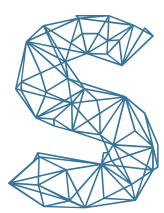
Cui also encourages the use of single purpose functions. That's good from a security perspective, as it reduces the attack surface, but it also keeps things simple.

Much of the advice for migrating to a microservice architecture discussed in The New Stack's "Guide to Cloud Native Microservices" applies to migrating to on-premises serverless architectures as well.

These are just some of the ways enterprises can explore serverless architectures without uploading sensitive data to the public cloud. Just as Amazon EC2 helped spawn an entirely new way of managing and operating infrastructure, expect serverless to do the same with the application life cycle industry.

## CHAPTER 06

# Serverless Testing in Production



Serverless is built on immutable infrastructure that spins up during an event. It's often not even possible to create the same workflow twice, making staging and testing prior to production impossible.

This inability to test in development is not unique to serverless, [Charity Majors](#), co-founder and CEO of monitoring tool [Honeycomb](#), said. Given the nature of building and deploying distributed applications at scale, there is no possible way to test for every eventuality.

This might bring to mind the [popular meme](#) of the “World’s Most Interesting Man” from the Dos Equis commercial: “I don’t always test my code, but when I do, I do it in production.” But Majors is quick to distance what serverless developers need to do from the sort of reckless deploys the meme satirizes.

“When I say ‘test in production,’ I don’t mean not to do best practices first,” explained Majors. “What I mean is, there are unknown unknowns that should be tested by building our systems to be resilient.” Some bugs are only going to show up in production, when an application is running at scale. The key is detecting those bugs and fixing them. While this is true for all applications at scale, serverless also brings its particular context.

Majors points to five emerging techniques for a microservices architecture that can be used to manage serverless testing in production.

## Feature Flags

Feature flags enable developers to stipulate who should be using a new feature, while it remains invisible to the bulk of the rest of the customer base. “Feature flags say ‘Don’t send anyone to this code until I say so,’” Majors said. This allows for developers to test everything on themselves first, prior to more widespread rollouts.

Often, Majors says, feature flags are something an internal team might hack up as they go, but that limits their usability internally, as it is often built without the user interface that makes it accessible beyond the engineering team who built them. New startups, like [LaunchDarkly](#), now offer feature flags as a service that are more accessible to non-developers.

“We use them at Honeycomb, and they are great,” said Majors. “The really cool thing about LaunchDarkly is that the interface is really useful, and easy to use, you can have non-technical people, like marketing, using it.”

This makes it possible for anyone in a business to be able to test their own changes, from a CEO updating the business website to a marketing team member adding API product landing pages. For example, they could use feature flags alongside a tool like LaunchDarkly to test the changes they have made prior to making them live for the majority of the customer base.

## Canaries

Canaries are staged rollouts of new features to specific subsets of users. When Majors was at Facebook, the engineering team often first deployed new features and updates to Brazil, and would then manually check the error rate between a control base and the Brazilian deployment group.



## Staged Rollouts with Auto Promotion to Larger Groups

This is a type of canary with a staged component, instead of deploying to a specific segment. For example, Majors explains, you could set a canary to 10 percent of all of production. If there are no issues, you could then expand that to 25 percent of production. If there are problems at any stage, you can revert back to a previous version. Majors says this is useful particularly for applications at scale, as often there are problems that may not exist at lower levels of deployment in production. “There may be problems you only see when you get to 80 percent, for example, because it requires opening more connections, or using more RAM. There becomes a tipping point, where at aggregate level it takes the server down,” said Majors.

## Rich Instrumentation

“I firmly hope that we are going to look back to today as the bad old days when we shipped code and if we don’t get paged, then everything is okay,” warned Majors. At Honeycomb, her commitment is to building rich instrumentation so that developers can ask whether shipped code behaves as expected. “We should be capturing enough detail so we can see. As an industry, we are still coming up with universal principles for instrumentation. It is an art as well as a science.” Because of the nature of global applications, and the size of serverless and their often large workflows of multiple functions, entire architecture systems can’t fit in the heads of a single architect overseeing the system anymore, requiring a richer set of tooling to assist with managing production.

## Observability-First Development

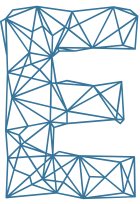
All of this comes to a new paradigm Majors hopes will take the application development industry beyond the current test-driven development principle. With rich instrumentation, it will be possible to monitor systems and understand how they are performing in real time. In an observability-first

development paradigm, an engineering team might map the desired outcome from a workflow, application or feature rollout. But then, before coding for that feature or workflow, the team would create their instrumentation toolkit that first might ask if the solution is even worth building, but then would be able to monitor both development and deployment to ensure alignment with the initial objectives. “It is like wearing a headlamp. Instead of building a feature and monitoring it, which is a very blind approach, with observability-first development you build instrumentation first. It gives more confidence to developers,” Majors said.

The reality is that “testing in production” doesn’t necessarily mean pushing untested code to customers. Developers can test code that is technically in production, but not yet being used by actual customers, then gradually increase the number of real customers actually using the code, not unlike a beta release. Far from being a reckless practice, it roots out issues that couldn’t be found any other way.

## CHAPTER 07

# Serverless Pricing

 Even if cost savings aren't the main reason for using serverless, the prospect of not paying for unused resources is often what first attracts many a manager to serverless. And yes, there are savings to be had by offloading a function or a chunk of data to some pay-as-you-go service instead of renting dedicated resources. But you must be mindful lest these services rack up bills that are actually higher than you'd be paying for a traditional server, whether in the cloud or on premises.

“The cost of serverless systems is something we are finding that people are worried about more and more,” said [Nitzan Shapira](#), CEO and co-founder of application performance monitoring company [Epsagon](#). “More adopters are now asking: What are we paying, and what for? Is it because the code is slow, or because of an API we are using? They need to be able to analyze costs in a very business [friendly] and logical way.”

## Finding the Sweet Spot in Function Runtime

Pricing of FaaS platforms is structured in similar ways by AWS, Azure Functions, Google Cloud Functions, and IBM Cloud Functions, which all charge for both compute time and number of invocations of a function. Users can then

choose the amount of memory size to allocate to compute time.

On the face of it, it would appear that choosing the lowest memory allocation for compute would be the cheapest. But that's not always the case. Execution time may lengthen with smaller memory size allocations, thus raising the overall cost, [Jeremy Daly](#), chief technology officer (CTO) of connected home company AlertMe, explained in a [blog post](#) earlier this year.<sup>3</sup> Daly shared data presented by [Chris Munns](#), principal developer advocate for serverless at AWS, that demonstrated that allocating 128 MB for memory size could cost \$0.024628 after 1,000 invocations, and take 11.72 seconds. The price then rose slightly for memory size allocations of 256 MB, and 512 MB, but what was most interesting was when a user allocated 1024 MB RAM to compute, the execution time dropped to 1.47 seconds, but the cost was only \$0.00001 more at \$0.024638. At Serverlessconf in New York last year, several speakers all shared similar findings in regards to finding the sweet spot with serverless pricing, regardless of the cloud vendor.

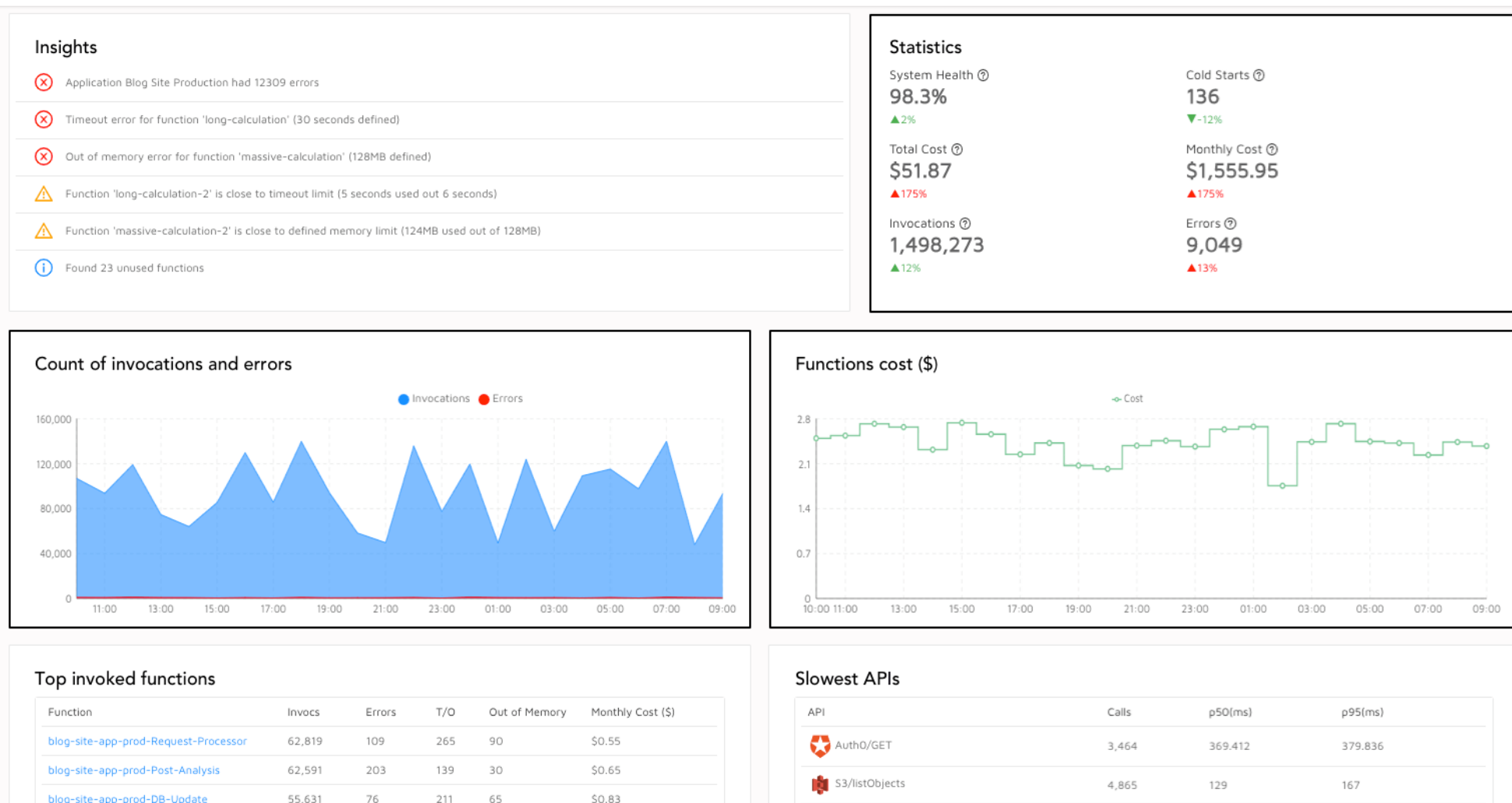
## Pricing is More Than Just Invocations and Compute

A serverless workflow generates costs besides just function invocations and memory allocation. For example, the company [Ipdata](#) migrated its application for looking up geographic information associated with particular IP addresses to AWS Lambda, founder Jonathan Kosgei, [wrote](#) this year. Lambda pricing was great, Kosgei wrote, but the company also used Amazon's CloudWatch tools, which turned out to be surprisingly costly.

**“Every API you call, you're not just paying for the service, you also pay for the response time.”**

— [Nitzan Shapira](#), CEO and co-founder of [Epsagon](#).<sup>4</sup>

But perhaps the biggest hidden cost comes from API requests. “Since many serverless apps are heavy on API calls, this can get quite pricey at roughly



**FIG 7.1:** Epsagon dashboard shows link between performance and costs.

\$3.50 per 1M executions,” [Amiram Shachar](#), CEO and founder of Spotinst, [wrote at the start of the year](#).

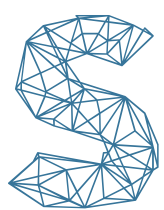
That makes monitoring the performance of these APIs with a tool such as Epsagon crucial, because if even one third-party API is slow, it can slow down any function that depends on it, thus driving up the total compute time, and therefore the bill as well.

As a 100 percent serverless company, Epsagon experienced this cost firsthand early on in its company history when a bug in its CloudWatch function prevented the service from scaling well. They discovered that 5,000 functions were repeatedly getting timeouts that amounted to a cost of \$12,000 per month — a fortune for a small startup.

Cloud providers do, thankfully, offer the ability to set rate limits and spending thresholds on service accounts and will issue automatic billing alerts if you reach your limits. But closely monitoring risks associated with spiking and ensuring that estimated costs will not jump from poor performance of serverless workflows and systems will be a focus in the coming year.

## CHAPTER 08

# Serverless Analytics



erverless architecture, like other distributed systems, pose challenges for those looking to collect and analyze data generated by their own software. In the past, a VP of engineering may have been able to go to their operations team and ask for data on why an application is underperforming or broken, or why infrastructure costs were higher than budgeted. Now the Ops team has been replaced by serverless and cloud engineering, which pushes the onus onto the developer teams more generally. So instead of 10 Ops specialists, there are now 200 developers working under a VP of engineering and each one is responsible for collecting the data on the elements of the system they are working on.

Because the infrastructure is fully abstracted, the health of the system is no longer a useful predictor of application health. And some would argue system metrics, such as uptime, have always been meaningless to developers. There is no single, standard metric that developers can rely upon to tell them whether their code is working or not. Observability gives developers the ability to collect data from their application and trace problems to the root cause, in order to debug code and relieve issues affecting the application in production. If monitoring is about watching the state of the system over time, then



observability is more broadly about gaining insight into why a system behaves in a certain way.<sup>5</sup> Each issue is unique and has its own set of metrics that combine to give the complete picture. It's the nature of complex, distributed systems that there are no standard set of issues that cause failures and thus no obvious fix.

“Every time you're getting paged, it will be a problem you haven't seen before and won't see again.”

— [Charity Majors](#), co-founder and CEO at [Honeycomb](#).<sup>6</sup>

This shift toward observability —the ability to understand how a system is behaving by looking at the parameters it exposes through metrics and logs — is upending standard monitoring practices and creating challenges for organizations that adopt microservice and serverless architectures. In The New Stack's 2018 serverless survey, 60 percent of respondents who already use serverless said debugging and tracing is a major monitoring pain point.

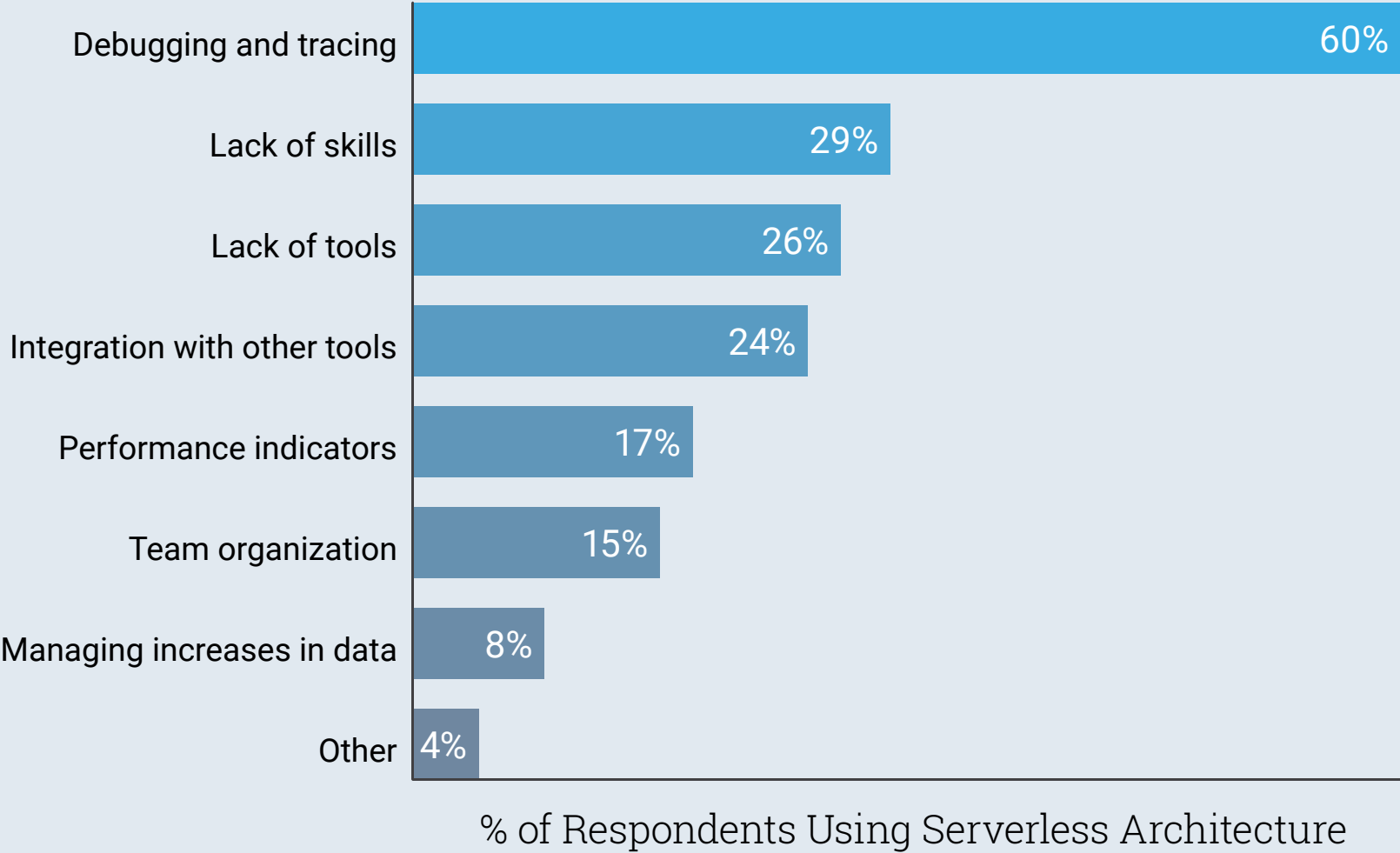
A cloud native system is composed of applications assembled as microservices. These loosely coupled services run across cloud providers in data centers around the globe. There may be millions of these services running at the same time, producing a hive of interactions. At such scale it becomes impossible to monitor each one individually, let alone their interdependencies and communications.

“With a monolithic app, you had one set of logs,” says [Nate Taggart](#), CEO and co-founder of the serverless management software [Stackery](#). “In serverless, you have distributed apps, with hundreds of functions each creating their own log systems. How do you correlate that across your whole architecture?”

## Observability Starts in the Code

This challenge of monitoring complex systems, and tasking developers with

# Serverless Monitoring Pain Points



Source: The New Stack Serverless Survey 2018. Q. What are the top two pain points for monitoring? n=254. © 2018 THE NEW STACK

**FIG 8.1:** *Debugging is by far the top monitoring pain point among those using serverless.*

debugging code in production, also surfaces a more fundamental issue: the organizational structure itself. “You need some kind of governance,” Taggart says. “There is a reason why we have standards in our legacy infrastructure, and the idea that we don’t need that same level of governance in serverless is not true. If anything we need more, because we are opening the doors of our systems to more engineers to manage.”

Serverless demands that metric collection is planned at the development cycle, Taggart said. While this may often be an aspirational best practice, with serverless it becomes a required best practice. Other serverless industry leaders, like Charity Majors and Kelsey Hightower, also advocate for instrumentation design during the development cycle, in order to be able to continue testing serverless applications in production environments.

The idea is to move away from the old real world where an engineer builds a function and releases a Lambda in the enterprise production account and on the day that it fails, as is so often the case, that engineer is out of the office. If the organization has introduced standardization, if every resource has shipping logs and is collecting metrics, instead of tracking down which engineer built the function and understanding what they did, the organization can move directly into resolving the problem by addressing the reason why failure, or underperformance, or cost blowout, occurred.

Taggart says at the heart of improving analytics is the need for standardization and automation. He advocates for using serverless as an opportunity to build self-healing applications, which means having policies and standardization processes in place so that instrumentation is automatically added during the build process, and that's a big part of what he's trying to do at Stackery.

“When building self-healing apps, you design resilience into the app. While developing the app you are thinking ahead to when it fails. How will you auto-recover and deal with failure?” Taggart said.

## Let Business Goals Guide Metrics

Monitoring in serverless systems is not as simple as just setting up alerts for performance, speed or time-outs, but, instead, requires a higher level of thinking that relates back to the intended goals of the application. This involves concepts such as infrastructure-as-code. An application holds all of the code, not just for performing its functions and compute processes, but because it is being run on serverless systems, also for initiating instances and scaling up when needed, automatically.

“Your software creates a certain scale: It creates new clusters, it meets new use cases. But as developers, our current processes often fail to meet this new layer of speed,” said [Or Weis](#), CEO and co-founder of serverless monitoring tool

[Rookout](#). “We have to speed up our own monitoring processes and how we observe and iterate.”

Weis says that there are no must-have metrics today when monitoring in serverless. Instead, monitoring and analytics need to relate back to the application goals: What are your use cases? Are they going to change over time? What are your goals? What are your next features? How will customers be using your product? What will you need in terms of observability? How much time will you have and how much time will you need? Those are the questions you need to ask before you start monitoring, said Weis.

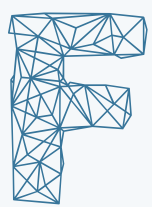
“[Serverless monitoring] is not about application performance, but behavior,” Erica Windisch, CTO and co-founder of IOpipe, said.<sup>7</sup> “It’s not about servers’ uptime, but about whether your application is working. I want to monitor: Did my sales go down, and [answer] why did they go down?”

Fortunately, there are a growing number of tools available to make all of this a little easier. Performance is crucially important in serverless environments, because slow response times could indicate security problems or lead to higher costs. For those using AWS, CloudWatch provides a number of helpful tools, but Taggart points to third-party companies like [IOpipe](#) and [Epsagon](#) as the state of the art for serverless application performance monitoring.

Watch for serverless monitoring and analytics tooling to keep evolving.

## CHAPTER 09

# Serverless Case Study of Success



Fabric is an insurance startup designed for new parents. The company offers accident and life insurance policies, as well as a free service for making wills. Both products are completely provided using serverless technologies.

Like so many other companies, [Fabric](#) was attracted to Serverless by the ability to execute quickly and with minimal complexity. “Honestly, I haven’t been too price sensitive,” [Steven Surgnier](#), co-founder and CTO, said. “We need to decrease the time from the initial idea to getting the end product in the customer’s hands. The cost savings of Lambda are a nice-to-have.”

As an early adopter of AWS Lambda, starting over four years ago, Fabric engineers began with a minimal serverless tool palette to paint with, including CloudFormation, API Gateway, and DynamoDB.

Relying on serverless, especially prior to the more recent generation of tooling that has become available, gave Fabric’s engineers a chance to build robust, standardized processes. “We are a small team, so we don’t have a dedicated DevOps team, we just have full stack engineers,” said Surgnier.

Like many other serverless pioneers, Surgnier points to one big way the team

works differently from traditional software design: They need to focus on the access to cloud services and the data flow through identity access management (IAM).

“In serverless, there is no server that is sitting in front of a database and is the guard to that database,” said Surgnier. “You need to define that access to data up front. At Fabric, it’s each individual engineer’s responsibility to set the appropriate responsibility levels for IAM.”

Surgnier also plans to continue building on their identity access management processes. As engineering grows large enough to warrant a dedicated security team, Surgnier sees a point where that team would own the IAM permissions of a CloudFormation template. “All of this points out that Ops has not gone away, it is just that there is a different class of problems now,” said Surgnier.

Overall, Surgnier sees serverless as the best decision for Fabric as a startup entering an established market and revitalizing the product offerings available. The company is focusing on listening to a particular customer market — young families — and meeting their needs, which have been sidelined in a larger, homogenized industry approach.

“We want that experience to be reflected in the brand,” he says.

“So we focus on being better at listening to customers. Every great company knows how to deal with that well. That just takes compassion and energy, it doesn’t take servers.”



# Bibliography

## 1. [How to Migrate Existing Monoliths to Serverless](#)

by [Yan Cui](#), principal engineer at [DAZN](#), writer and technology instructor, [Binaris blog](#), May 13, 2018.



*Cui details the experience of migrating a social network, Yubl, to serverless.*

## 2. [Guide to Cloud Native Microservices](#)

by The New Stack, 2018.

*This ebook provides a high-level overview of what organizations should consider as they create, deploy and manage microservices for cloud native applications and lays the foundation for understanding serverless development and operations.*

## 3. [15 Key Takeaways from the Serverless Talk at AWS Startup Day](#)

by [Jeremy Daly](#), CTO of AlertMe, [jeremydaly.com](#), July 11, 2018.



*Daly distills what he learned from a talk that Chris Munns gave at AWS Startup Day on “The Best Practices and Hard Lessons Learned of Serverless Applications.”*

### SPONSOR RESOURCE

- [CNCF Serverless White Paper](#)

by the [Cloud Native Computing Foundation](#) (CNCF) Serverless Working Group

*This paper describes a new model of cloud native computing enabled by emerging “serverless” architectures and their supporting platforms. It defines what serverless computing is, highlights use cases and successful examples of serverless computing, and shows how serverless computing differs from, and interrelates with, other cloud application development models.*

#### 4. [Nitzan Shapira](#), CEO and co-founder of [Epsagon](#)

in “What We Should All Worry About When Monitoring Serverless Applications” at [ServerlessDays Portland](#), September 17, 2018.



*Serverless systems are more than just functions, but also APIs and transactions and should be monitored as such, Shapira discussed in this talk.*

#### 5. [Cloud Native Monitoring](#)

by [Ian Crosby](#), [Maarten Hoogendoorn](#), [Thijs Schnitger](#) and [Etienne Tremel](#), of [Container Solutions](#) in The New Stack’s “CI/CD with Kubernetes” ebook, 2018.



*In this new cloud native era, continuous understanding about an infrastructure’s state of health defines how applications are*



*built, deployed and managed.*

#### 6. [Charity Majors](#), CEO of [Honeycomb](#)

in “Observability (and Responsibility) for Serverless Systems” at [ServerlessDays Portland](#), September 17, 2018.



*Majors explained observability in this talk and how troubleshooting becomes more complex with microservice and serverless architectures.*

#### SPONSOR RESOURCE

- [Rapidly Build Serverless Architectures w/ Stackery](#)

by [Stackery](#)

*Add speed and scalability to your serverless infrastructure with the Stackery Operations Console. This product brief covers everything you need to accelerate your serverless development time and shorten your release cycles. Learn how to get complete visibility into your serverless applications through Stackery’s GUI or command-line interface.*

## 7. [Observability To Better Serverless Apps](#)

by [Erica Windisch](#), CTO and co-founder of [IOpipe](#), at [ServerlessDays Portland](#), September 17, 2018.



*Windisch explained in this talk how serverless platforms now provide four factors of a twelve-factor application.*

# Closing

Serverless architecture reflects how the orchestrated, declarative use of event-based functions makes infrastructure more deeply accessible and efficient for scale-out purposes. But abstractions don't come overnight. They come through incremental changes. The more boring the infrastructure, the better the developer experience. Ease-of-use defines how well a developer may build out services. The core strength for serverless technologies comes in leveraging cloud services that test and monitor the distributed architecture that runs the application platforms and services. It allows for a more iterative approach that takes a fraction of the time compared to more traditional IT practices. It allows DevOps teams to focus on higher value tasks: extensive testing for every new function, every serverless architecture, every application a developer team is building, deploying and managing.

What matters first are the people who are building application services. Their workflows must fit with practices that operations teams follow to make sure the most can be gained from the infrastructure and platforms. It's a shared responsibility among Dev and Ops teams that has to be holistic in every respect. Shared ownership over application security is of particular importance, which again shows the deep need for an effective approach to monitor services.

The path for adopting a serverless architecture reflects the maturity of the organization. Serverless users often have existing developer teams that are connecting services and need an effective way to work across team boundaries to do so. They work independently in a DevOps-based, microservices approach.

Serverless architecture could arguably be what allows for a far more cost efficient development process. But it's time efficiency that matters most. That's the promise of serverless. It's not a magic potion that the gods use to perfume

the clouds and attract developers. It's an approach to make application development faster and more effective by following business objectives and evolving through organizational structure and practices.

This ebook is the second to follow The New Stack's new approach to developing ebooks. We believe it is people and their workflows that matter. These shape organizational structure, which in turn dictate technology adoption.

Thanks and see you again soon.

**Alex Williams**

Founder and Editor-in-Chief, The New Stack

# Appendix

The New Stack conducted a survey in August 2018. Four hundred seventy-nine people completed the survey, and another 129 respondents' incomplete data was included in the analysis. Demographic information on the respondents is presented below. The full data set for the survey results is [available online](#).

Company Size	
Self-employed or not working	5.9%
2-50	30.4%
51-250	17.3%
251-1,000	12.6%
1,001-10,000	14.6%
Greater than 10,000	17.1%
Don't know	2.0%

Source: The New Stack Serverless Survey 2018.  
Q. How many people work for your company or organization? n=748.

Job Role	
Developer / software engineer	37.8%
Architect	26.7%
IT management, including CIO / CISO / CTO	18.9%
Administrator / operations	8.8%
Marketing / PR	2.5%
Community manager / developer advocate	1.3%
C-level (non-technology) management	1.1%
Other	2.9%

Source: The New Stack Serverless Survey 2018.  
Q. Which category most closely defines your role? n=445.



Industry Vertical	
Information technology: cloud native services or serverless vendor	20.1%
Information technology but not cloud native services or serverless vendor	18.1%
Financial services	11.3%
Professional services	10.8%
Telecommunications or media	7.7%
Consumer goods or retail	7.0%
Government or nonprofit	5.6%
Industrial or manufacturing	4.7%
Healthcare or pharmaceuticals	3.4%
Energy or utilities	1.8%
Other	9.5%

Source: The New Stack Serverless Survey 2018.  
Q. Which industry vertical does your enterprise or organization belong to? n=443.

Geographic Region	
North America	38.3%
Europe	32.0%
South Asia	8.6%
East or Southeast Asia	7.4%
South America	5.1%
Australia, Oceania or Pacific Islands	5.0%
Africa or Middle East	3.6%

Source: The New Stack Serverless Survey 2018.  
Q. What geographic region are you located in? n=444.

# Disclosure

The following companies are sponsors of The New Stack:

Alcide, AppDynamics, Aqua Security, Aspen Mesh, Blue Medora, Buoyant, CA Technologies, Chef, CircleCI, CloudBees, Cloud Foundry Foundation, Cloud Native Computing Foundation, Dynatrace, InfluxData, LaunchDarkly, MemSQL, Navops, New Relic, OpenStack, PagerDuty, Pivotal, Portworx, Pulumi, Raygun, Red Hat, Rollbar, SaltStack, Semaphore, The Linux Foundation, Tigera, Twistlock, Univa, VMware, Wercker and WSO2.

