New Relic

For the Love of Serverless

2020 AWS Lambda Benchmark Report for Developers, DevOps, and Decision Makers

Table of Contents

Executive Summary

Turn on, tune in, and stop over-provisioning in 2020

Welcome to the first edition of New Relic's Serverless Technology Semi-annual Report.

A lot of people are talking about serverless trends right now. Still, there's a lack of quantitative data in the market to inform developers, DevOps practitioners, and decision makers of crucial serverless adoption metrics and benchmarks.

New Relic One processes trillions of serverless events every month across the globe. For this report, we've aggregated and analyzed the same sample set over time to call out key trends that help serverless users make knowledge-based decisions about their architecture and performance goals to ultimately build better software.

In addition, we've asked serverless experts throughout the developer community to weigh in on the current state of the industry and the direction it's heading.

If you attended AWS re:Invent in December 2019 or even tangentially followed the event chatter, you most likely observed the conveyor belt of serverless-related announcements—Amazon Virtual Private Cloud (Amazon VPC) cold-start improvements, provisioned concurrency, AWS Lambda Destinations—along with the hundreds of supporting sessions.

Within New Relic One, we've seen serverless adoption significantly increase within enterprise organizations, both among cloud-native innovators and traditional industries that we may not typically view as fast-moving. According to the 2019 Forrester Global Business Technographics Developer Survey highlighted in Forrester's report, Serverless Development Best Practices, 49% of companies are using or plan to use serverless architecture in the next 12 months.

"Development teams use serverless platforms to quickly test out new business concepts or features in hours versus days because the amount of time spent provisioning infrastructure is minimal," noted Forrester.

For the cyclical, real-world workloads of most businesses running in digital environments, it's hard to ignore the opportunity to transition from an over-provisioned infrastructure designed to accommodate the most extreme traffic use cases to flexible programming models like those that run in serverless environments. This capability allows teams to tune their environments to autoscale for busy seasons versus supporting peak traffic 24 hours a day, 365 days a year.

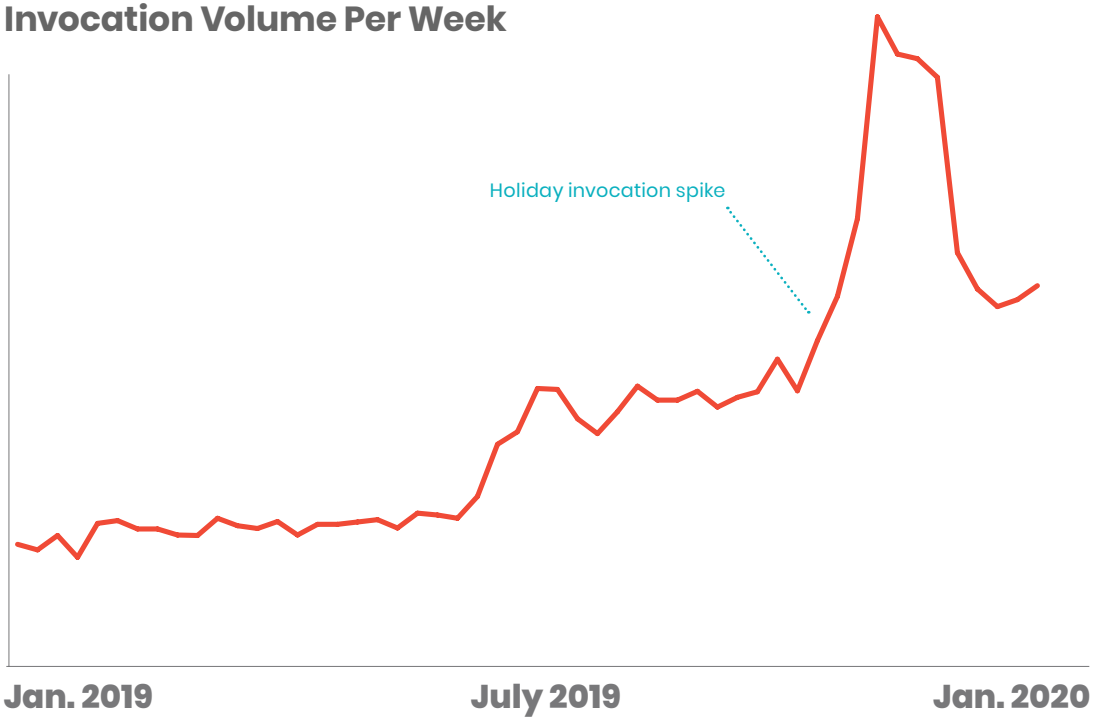
While the term "serverless" includes many services from multiple cloud providers, including Google and Microsoft, we've explored data from Lambda for this report. Based on feedback, we may expand our focus in future editions.

Key Findings

- Serverless adoption among enterprises continues to rise with a 206% increase in average weekly invocations over the last 12 months. The enterprises using serverless in production are expanding their serverless footprint with a 178% increase of functions per account.
- In terms of function volume, developers mostly rely on Node.js and Python for building serverless applications on Lambda, with Java as the third most-used runtime. However, with the AWS launch of **Provisioned Concurrency** mitigating cold start impacts and **VPC improvements**, making Lambda more attractive for enterprises that require isolated environments, we expect the adoption trends for Java to increase in 2020.
- The continued bias toward smaller function code size, due in large part to deployment package size limits from AWS, supports the serverless best practice of creating functions to perform a single, well-defined task with low overall code sizes.
- Developers tend to prolong updates to the latest language version after **deprecation announcements** from AWS. We saw a notable volume of functions still running Node.js. 6.10, Python 2.7, and even older versions. These are likely unmaintained functions inflating error rates and costs.

The Rise in Serverless

Invocation Volume Per Week



Insights Snapshot

- Serverless adoption continues to rise, with a 209% increase in average weekly invocations over the last 12 months.
- Large spikes in invocations during the holiday season reinforce the serverless use case of auto-scaling to support peak holiday workloads across industries like retail, media, and logistics.

Invocation Growth

+ 209%

Average weekly invocation volume within sample set over the past 12 months.

SERVERLESS SPOTLIGHT: **AN INTERVIEW WITH**

FORREST BRAZEAL

SR. MANAGER, A CLOUD GURU

What are the biggest organizational challenges facing serverless adoption in 2020?

Commitment to the cloud or vendor-agnostic architectures as an ideological rather than practical choice continues to be the biggest organizational hurdle facing serverless adoption.

Close behind that, the biggest challenge is simply providing the appropriate training and enablement for teams to feel confident building value in this new paradigm. It's way too easy to get stuck for months (or even years) in "kick the tires" mode, unable or unwilling to pull the trigger.

What are the biggest technical challenges facing serverless adoption in 2020?

Many of the biggest technical challenges facing serverless in 2020 are common to distributed and microservices-based architectures in general. There is a huge backlog of legacy systems and codebases that have to be rewritten, and the tools to do it are not always as good or comprehensive as advertised.

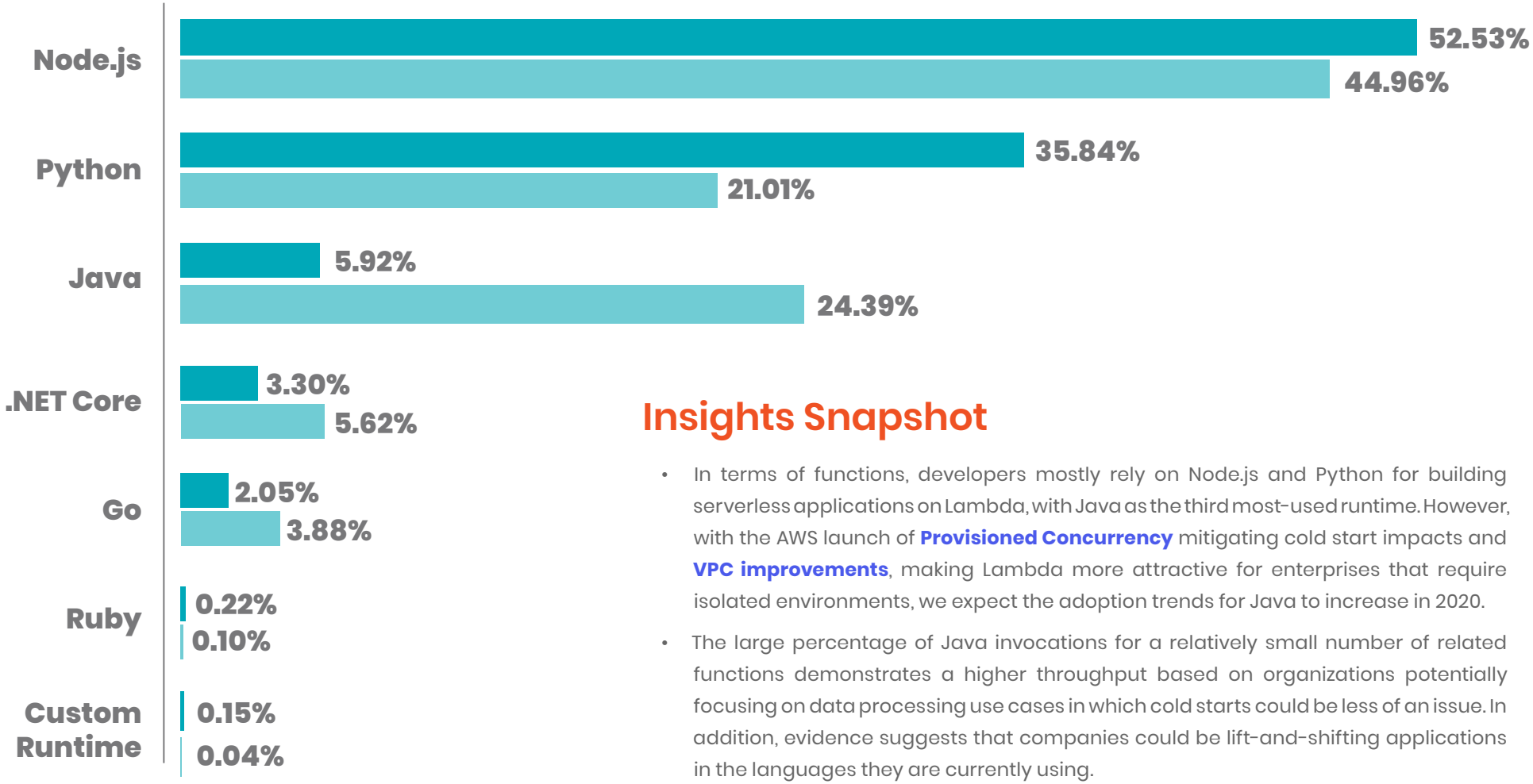
Where do you see serverless heading in 2020?

I believe more "stateful" options will be added to Function-as-a-Service (FaaS) platforms, blurring the line further with managed container offerings like AWS Fargate. I also expect to see the continued abstraction of best practices into developer-friendly services like AWS Amplify. And, opposition to serverless adoption will solidify around business and organizational concerns, such as legacy inertia and the threat of vendor lock-in, rather than strictly technical gaps.

What do you think is the biggest serverless myth that persists?

That "going serverless" is primarily a technical decision. It is not. It is an organizational transformation that requires dev and ops teams to embrace new roles, security to grow comfortable with redistributed threat models, and business leadership to endorse and embrace the value created.

Lambda Adoption by Runtime



Time frame July-December 2019

Insights Snapshot

- In terms of functions, developers mostly rely on Node.js and Python for building serverless applications on Lambda, with Java as the third most-used runtime. However, with the AWS launch of **Provisioned Concurrency** mitigating cold start impacts and **VPC improvements**, making Lambda more attractive for enterprises that require isolated environments, we expect the adoption trends for Java to increase in 2020.
- The large percentage of Java invocations for a relatively small number of related functions demonstrates a higher throughput based on organizations potentially focusing on data processing use cases in which cold starts could be less of an issue. In addition, evidence suggests that companies could be lift-and-shifting applications in the languages they are currently using.

SERVERLESS SPOTLIGHT: **AN INTERVIEW WITH**

DAVE TOWNSEND

PRINCIPAL SOFTWARE ENGINEER, MATSON

What are the biggest organizational challenges facing serverless adoption in 2020?

One of the challenges an organization will face in 2020, as it moves to more serverless workloads, is getting existing staff trained in this new way of doing things. It is a radical shift from how most shops are building applications today. The sooner you can start, the better. I think the best advice is to start small, iterate, and keep everyone involved. But definitely start now. In addition, specifically in larger organizations, getting senior management on board can be an important step. You may need to put on your selling hat and tell the story of why this is a good thing. You'll want to have data to show; otherwise, it's all talk.

What are the biggest technical challenges facing serverless adoption in 2020?

It is still really early in this game, and I think as a community we're evolving. There are certainly some gaps, and the gaps are what people are often the most vocal about pointing out. But, as we saw with some of the releases around serverless at re:Invent 2019, AWS is listening and those gaps are closing. In particular, the enhancements they have made around Lambda startup time in VPCs has definitely helped solve one of the challenges that has been holding enterprises back. People who aren't even doing serverless complained about that one! For the cloud vendors that want to win in this space, we'll no doubt continue to see enhancements to help alleviate a lot of these adoption challenges.

Where do you see serverless heading in 2020?

I'm no Simon Wardley, but from what I can see of the landscape, it looks like a continued steady trajectory upward as more and more companies start to experiment and share success stories. I don't know that serverless will ever hit a crazy spike, mostly because it's not something that you can switch over to with a few tweaks from what you are currently doing. This serverless thing is quite different. And honestly, I don't think it needs to spike. We've all learned from the tortoise and the hare that slow and steady is often the path to winning. Besides, it's like Taylor Otwell said about serverless: "What's the alternative? More complexity, more container orchestration? I don't think that's the future that people are going to naturally gravitate toward."

What are you most excited about in 2020 regarding the state of serverless?

There is a lot of great stuff going on right now, both from the cloud vendors and from the tooling side of things. I love seeing the evolutionary progress in Serverless Framework and SAM, and the progress in the monitoring and observability space around serverless.

Specific to AWS, I like the work going on with AppSync and Amplify. Along with Amplify, frameworks like **Architect** have a huge potential to create a whole new category of full-stack developers out of many existing frontend-only folks. These frameworks are making it super easy for frontend developers without much cloud experience to get a serverless backend up and running very quickly, and that's pretty exciting. Also, I'm super pumped on EventBridge—that service opens the door for some great event-based serverless architecture patterns.

Lambda Adoption by Runtime Version

Top 10 Runtime Versions by Invocation Percentage
July-December 2019

Node.js 8.10	30.58%
Java 8	24.39%
Node.js 10.x	8.99%
Python 3.6	8.38%
Python 3.7	6.67%
Python 2.7	5.96%
.NET Core 2.1	4.27%
Go 1.x	3.88%
Node.js 6.10	3.07%
Node.js 4.3	2.19%

Top 10 Runtime Versions by Function Percentage
July-December 2019

Node.js 8.10	32.42%
Python 3.6	13.45%
Python 2.7	12.32%
Node.js 10.x	11.27%
Python 3.7	9.85%
Java 8	5.89%
Node.js 6.10	3.69%
.NET Core 2.1	2.77%
Go 1.x	2.05%
Node.js 4.3	1.40%

Insights Snapshot

- Node.js 8.10 and Python 2.7 versions are still the most popular versions on Lambda. However, the scheduled deprecation of Node.js 8.10 will shift numbers toward Node.js 10.x. While Python 2.7 has reached the end of life, AWS is continuing to support the language in 2020.
- Developers tend to prolong updates to the latest language version after **deprecation announcements** from AWS. We saw a notable volume of functions still running Node.js 6.10, Python 2.7, and even older versions. These are likely unmaintained functions inflating error rates and costs.

SERVERLESS SPOTLIGHT: **AN INTERVIEW WITH**

FARRAH CAMPBELL

ECOSYSTEMS DIRECTOR, STACKERY

What are the biggest organizational challenges facing serverless adoption in 2020?

One challenge is how organizations adopt serverless as the primary architectural solution for their projects. Many organizations have one or two teams that have achieved great success with serverless but are unsure how to replicate that success on a wider scale. These organizations have talented people who share their knowledge and experience, but they also need tools to help central operations teams support them in the same way they support teams building applications with other architectural patterns.

What are the biggest technical challenges facing serverless adoption in 2020?

In the past year, many serverless platform issues have been addressed, including provisioning of "warm" functions and improved scalability when attached to virtual networks. Now that serverless can handle almost all use cases, the biggest technical challenges will continue to be developing the right workflows and tools for a broader developer base to adopt serverless.

Where do you see serverless heading in 2020?

With larger organizational adoption of serverless, it will start to overcome stereotypes around its limitations and quickly grow in wider adoption. Java and .NET shops, which traditionally have avoided serverless because of cold start issues, will start to adopt and build more tools to operate serverless in enterprise environments. Organizations will question their investment in server operations even more.

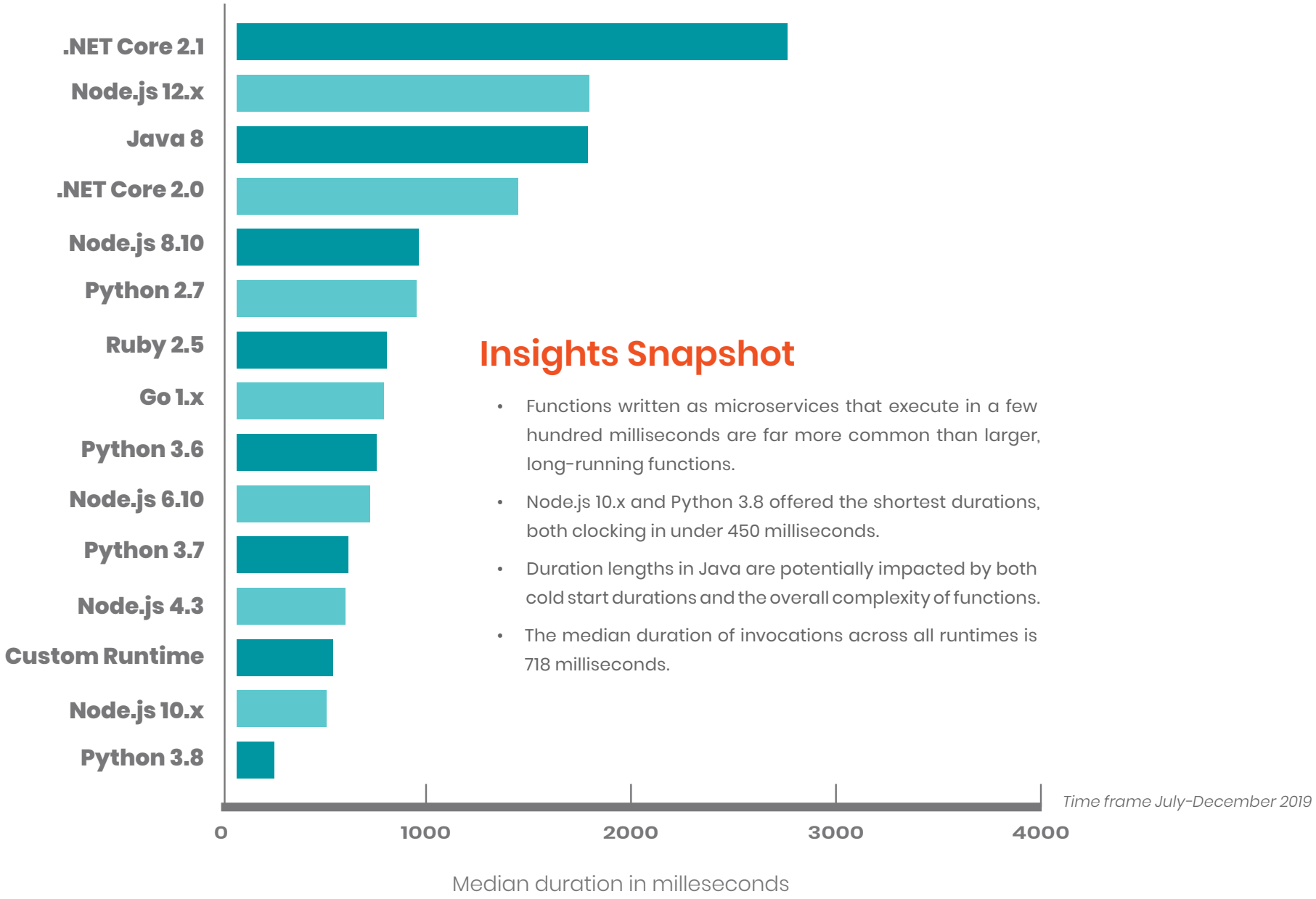
What are you most excited about in 2020 regarding the state of serverless?

Traditionally, it has been extremely challenging to build scalable web applications. The breadth of expertise required has limited the pool of people who can participate in this type of software development. Serverless, on the other hand, has lowered barriers to entry and increased diversity among developers wanting to build scalable, resilient applications. It will be exciting to see this impact continue.

What do you think is the biggest serverless myth that persists?

The biggest myth is that serverless == functions. People can get very fixated on functions and hyper-analyze limits and costs. These people are missing the benefits of serverless: simplified infrastructure and operations management. At the very least, serverless should be the preferred solution for most projects, and alternatives should be used only where necessary. Serverless provides such a high ROI that it's crazy to approach development in any other way.

Invocation Duration by Runtime Version



Insights Snapshot

- Functions written as microservices that execute in a few hundred milliseconds are far more common than larger, long-running functions.
- Node.js 10.x and Python 3.8 offered the shortest durations, both clocking in under 450 milliseconds.
- Duration lengths in Java are potentially impacted by both cold start durations and the overall complexity of functions.
- The median duration of invocations across all runtimes is 718 milliseconds.

SERVERLESS SPOTLIGHT: AN INTERVIEW WITH

BRIAN LEROUX

CTO AND CO-FOUNDER, BEGIN

What are the biggest organizational challenges facing serverless adoption in 2020?

Serverless is a movement predicated on the extremely large scale it takes to run a cloud. As cloud vendors grow, they buy and build more and more infrastructure, and in so doing, drive down the total cost of ownership. The services they offer increasingly become commodities. We started out renting cages and racks, moved to virtualization, and finally to renting a function executing in 100-millisecond increments. And while the definition of serverless has expanded to include concepts such as Infra as Code and Observability, the core ethos remains: outsource your undifferentiated, commoditized workloads and focus on core business value.

Of course, so many organizations are evaluating the different cloud vendor serverless offerings. While unfortunate, it is not unexpected that the various capabilities labeled "serverless" are currently very different between vendors. The very phase "serverless" also has enthusiastic advocates that can inflate expectations (myself included). Inflated expectations are great, but the next step in this story is the trough of disillusionment. The challenge I see in 2020 is that we are probably in the trough right now.

All cloud vendors have rough edges (some more than others). This is not to be unkind, but the delta between the leading two cloud vendors alone is significant. There is no doubt the future belongs to serverless, but the past is really clingy, and the larger space is still very early. While possible to port traditional monolithic apps with careful, deliberate, and intentional care (strangling the old monolithic legacy app into submission), it is far easier to start completely serverless. Even then, the path is still pretty rough. Our goal with **Begin.com** is to make AWS serverless frictionless also.

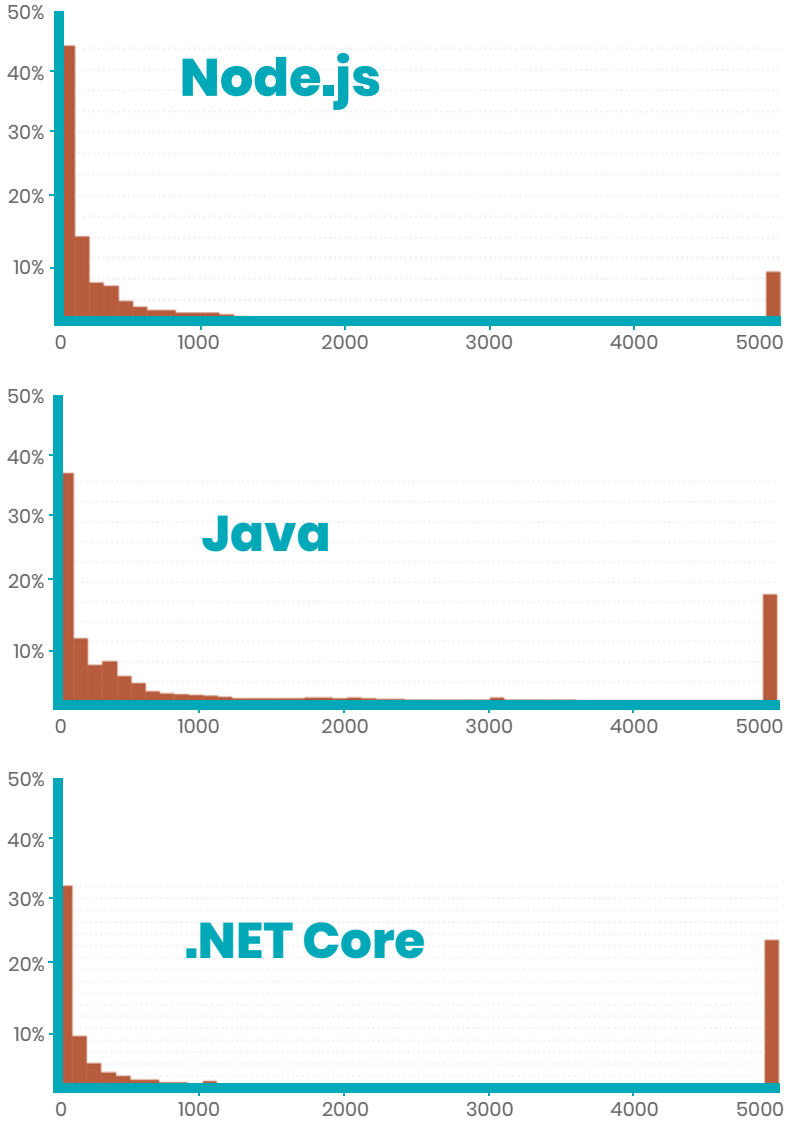
What are you most excited about in 2020 regarding the state of serverless?

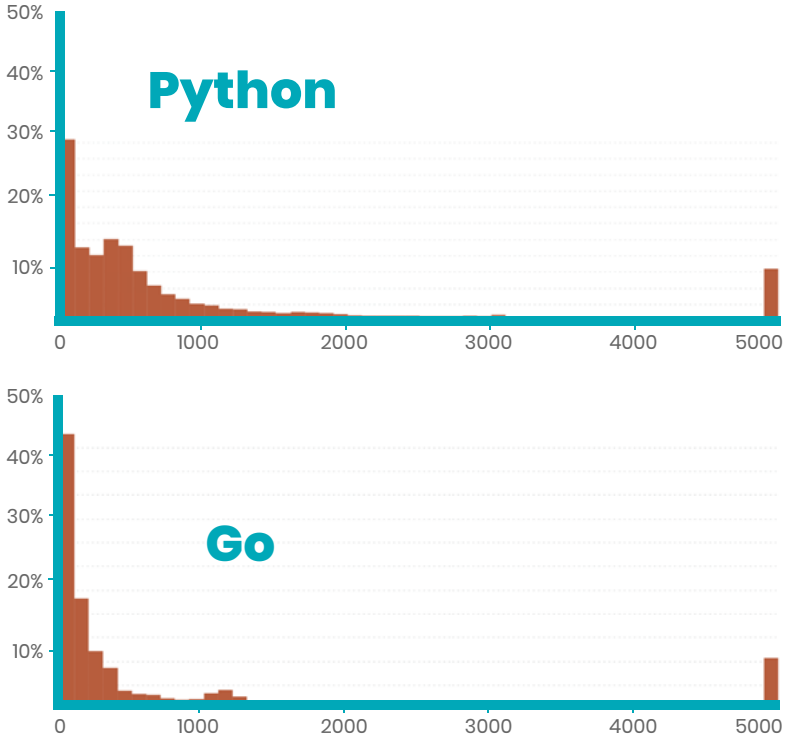
I'm excited by this new experimental JavaScript runtime called Deno. It has a lot of interesting characteristics that make it very desirable for serverless web development. It is sort of a do-over by the creator of Node. It's based on V8, like Node, and has a great cold start. Deno supports ES modules natively, along with the popular TypeScript and JSX dialects. This means serverless-side rendering (SSR), either React (or Preact) with TypeScript, can happen in a Lambda without any third-party build tools. It's pretty slick, and I think it will be a fun way to build web apps "serverlessly."

What do you think is the biggest serverless myth that persists?

Lock in. If you can go all-in with AWS building apps using something like Architect or raw SAM/CloudFormation and only serverless services (like API, Lambda, and DynamoDB), you stand a very good chance of successful execution and an advantage over competitors.

Duration Histogram by Runtime





x-axis – Average Function Duration (ms)
y-axis – Percentage of Functions

Insights Snapshot

- Even in Java, the majority of invocations are under 500 milliseconds. With the availability of the new Provisioned Concurrency feature from AWS to mitigate cold start impacts, Java in Lambda is a viable choice for critical workloads.
- Small spikes in duration at 3 seconds across runtimes demonstrate developers aren't adjusting the default Lambda timeout limit, while the far-right spike in duration at 5 seconds both represents timeout best practices, as well as the longtail of functions running more than 5000 milliseconds.

SERVERLESS SPOTLIGHT: AN INTERVIEW WITH

ALEX CASALBONI

TECHNICAL EVANGELIST, AWS
CO-ORGANIZER,
SERVERLESSDAYS MILAN

What are the biggest organizational challenges facing serverless adoption in 2020?

I think organizational challenges will mainly be related to cultural and technical change. After more than five years, the technology is mature, and I don't think we can blame tooling and UX anymore. If I were to recommend two approaches that might help overcome most organizational challenges for both small and large organizations, I'd choose to invest in training and focusing on smaller teams (who build smaller services). Training doesn't need to involve classrooms. Many developers prefer attending meetups or reading blogs regularly. Smaller teams—or two-pizza teams as we call them at Amazon—are still hard to imagine in some more traditional organizations but have the very high potential to impact both the commitment and productivity of every team member.

What are the biggest technical challenges facing serverless adoption in 2020?

I think most technical challenges have been addressed in the last two to three years. Since I started using serverless in 2016, many blockers have been removed. There are still technical improvements on the roadmap that will unlock even more edge cases and simplify some of the very common ones. But if you look at the last 12 months, you'll find so many innovations in this space (**Amazon EventBridge, Amazon RDS Proxy, Provisioned Concurrency for AWS Lambda, Custom Runtimes, Data API for Amazon Aurora Serverless, AWS CDK**, etc.) that I'm confident 2020 will be the year when all of these new ways of designing architectures come together to simplify developers' lives.

Where do you see serverless heading in 2020?

One of the patterns I see is moving toward writing "less code." Many developers are deeply convinced that their main job is to write code. Forty years ago, "developers" were convinced their job was to punch holes in cards manually. As developers, our primary job is to solve problems and satisfy customers' needs. Now we can achieve the same things much faster by writing less code (even less FaaS code) via managed services integrations and third-party APIs. This pattern started years ago; some even coined the term "serviceful" as an alternative to "serverless," and 2020 might be the year when it becomes the new normal.

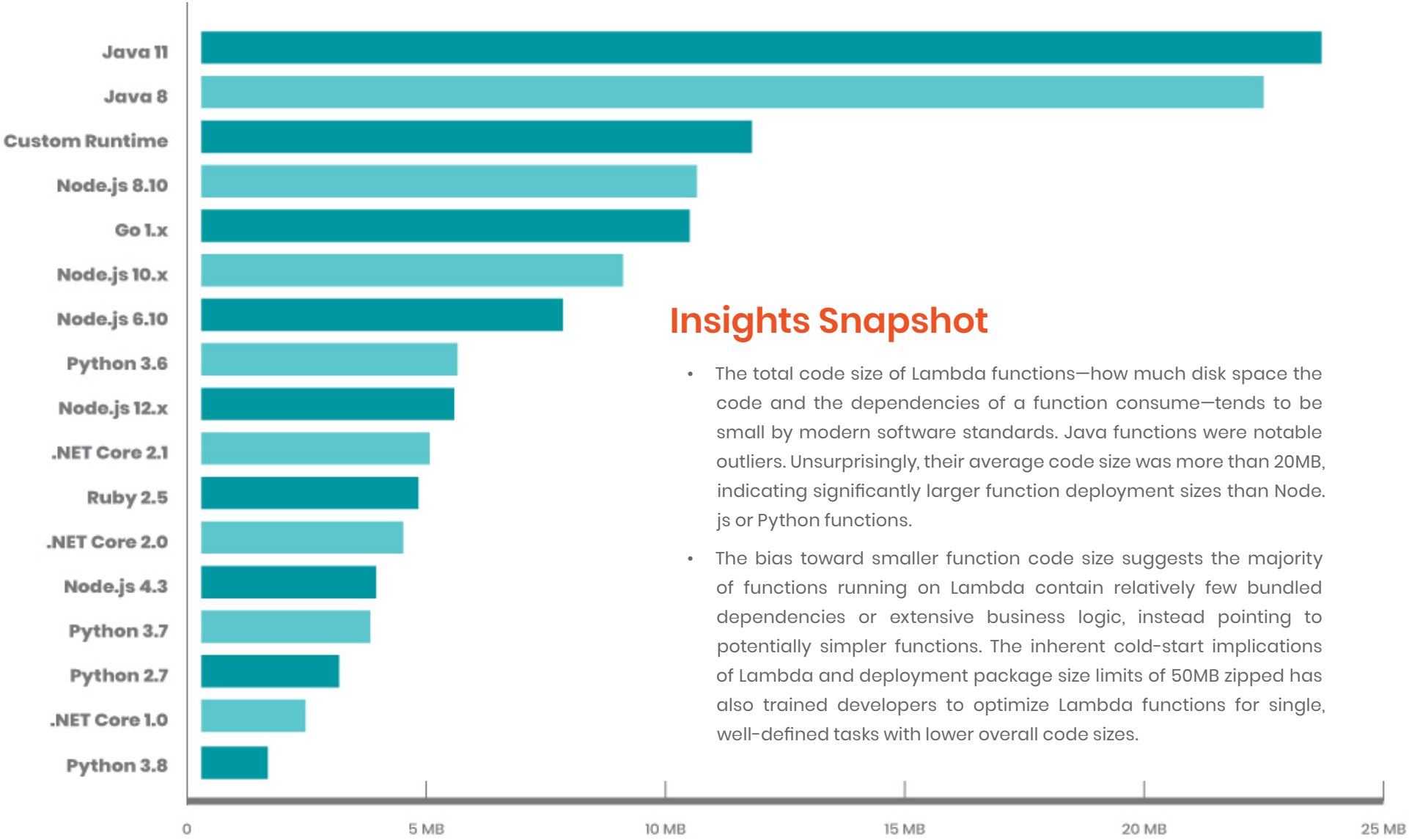
What are you most excited about in 2020 regarding the state of serverless?

I haven't heard "it doesn't run on Lambda" for at least six months. The opportunities to "lift and shift" to serverless have increased, as a first step to get started quickly and experience the benefits.

What do you think is the biggest serverless myth that persists?

Performance. Because of the increasing number of built-in integrations (rather than custom polling or workarounds) and the continuous performance improvements under the hood, the "serverless is slow" myth should be reconsidered. Especially taking into account two of the major improvements of 2019: no more **VPC** cold starts and provisioned concurrency. These two combined will solve most concerns related to latency-sensitive applications at scale.

Average Code Size by Runtime



Time frame July-December 2019

Insights Snapshot

- The total code size of Lambda functions—how much disk space the code and the dependencies of a function consume—tends to be small by modern software standards. Java functions were notable outliers. Unsurprisingly, their average code size was more than 20MB, indicating significantly larger function deployment sizes than Node.js or Python functions.
- The bias toward smaller function code size suggests the majority of functions running on Lambda contain relatively few bundled dependencies or extensive business logic, instead pointing to potentially simpler functions. The inherent cold-start implications of Lambda and deployment package size limits of 50MB zipped has also trained developers to optimize Lambda functions for single, well-defined tasks with lower overall code sizes.

SERVERLESS SPOTLIGHT: AN INTERVIEW WITH

ERICA WINDISCH

PRINCIPAL ENGINEER, NEW RELIC

What are the biggest organizational challenges facing serverless adoption in 2020?

Those who are just getting started may wrangle with CI/CD and other release processes, including understanding their serverless application inventory and spend. More seasoned serverless organizations will be seeing their developers building more complex and powerful applications, and facing challenges of operating that workload at scale.

Where do you see serverless heading in 2020?

A few years ago, serverless compute was all AWS, but CloudFlare Workers and Twilio Functions seem to be gaining traction. Rather than services trying to be better at Lambda than AWS, I expect more companies to expand the serverless market with differentiated products targeting IoT, mobile, telecom, etc.

What do you think is the biggest serverless myth that persists?

I believe that estimating costs for operating serverless projects remains difficult and has resulted in a myth that it is too expensive. Tools such as servers.lol can help dispel this myth and build more accurate projections when pitching projects.

SERVERLESS SPOTLIGHT: **AN INTERVIEW WITH**

JEREMY DALY

HOST, SERVERLESS CHATS

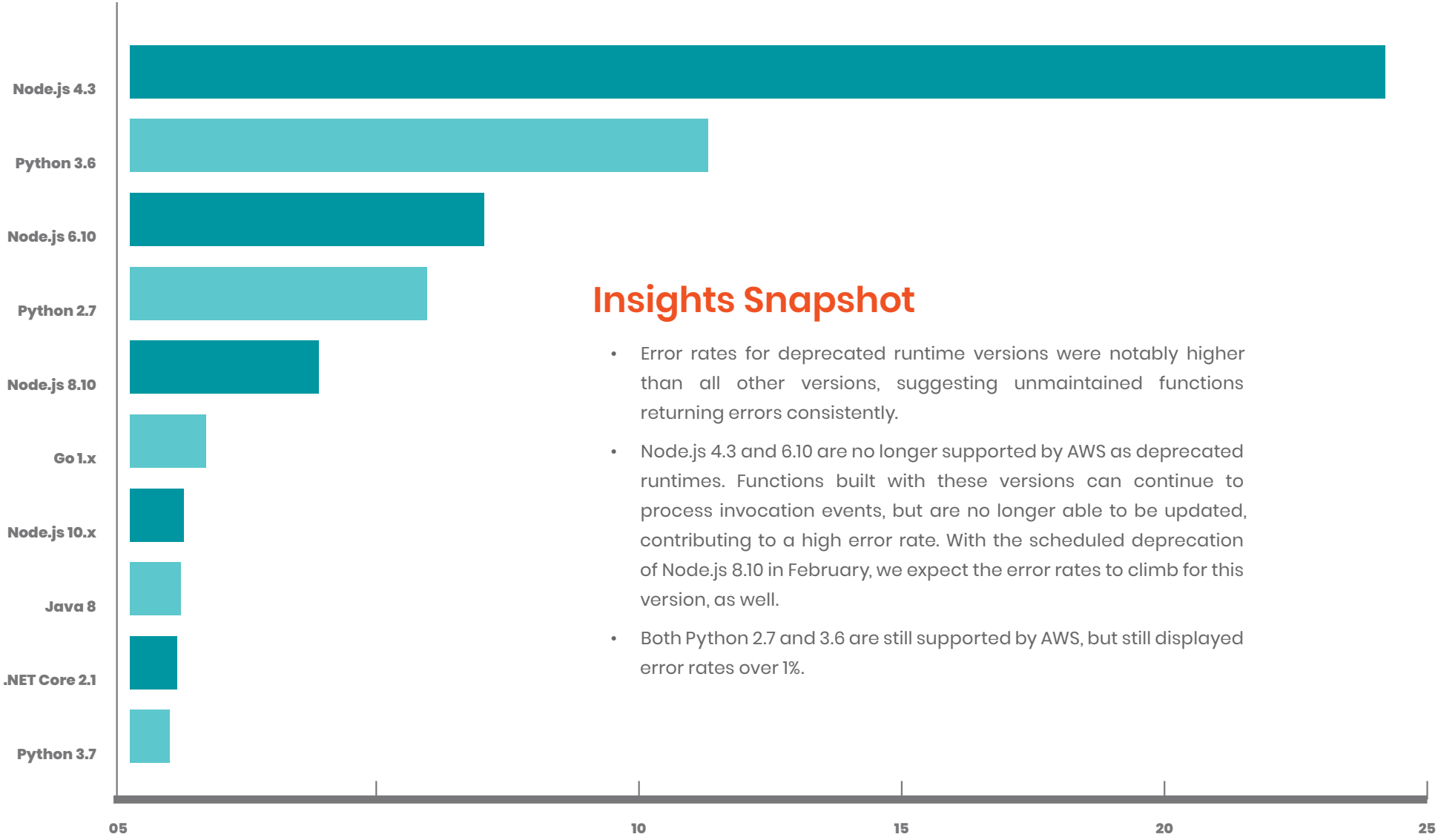
Where do you see serverless heading in 2020?

I think we're going to see Serverless become more complicated before it becomes easier. While the basics of getting a simple function up and running is quite simple, creating complex applications that communicate with multiple managed services is a much different story. Abstracting away the management of the infrastructure does not remove the need for developers to understand its proper use cases and limitations. This often requires a deep understanding of how specific managed services work, when and how to tweak certain knobs to optimize performance, and most importantly, understanding how to deal with the inevitable failures of distributed systems. These complexities are likely necessary, but I think we'll see more companies working on what I call "abstraction as a service," that will help to encapsulate best practices and compliance requirements into higher level components that can be reused by community and team members.

What are you most excited about in 2020 regarding the state of serverless?

I think Serverless has evolved to the point where it covers an incredibly large number of use case. The recent addition of tools like the AWS Data Proxy, Provisioned Concurrency, and Lambda Destinations, have helped to reduce several objections from holdouts in the space. I have spoken with many organizations who are starting to look at "serverless first" solutions to build new features, as well as migrate existing functionality.

Error Rates by Runtime



Time frame July-December 2019

Insights Snapshot

- Error rates for deprecated runtime versions were notably higher than all other versions, suggesting unmaintained functions returning errors consistently.
- Node.js 4.3 and 6.10 are no longer supported by AWS as deprecated runtimes. Functions built with these versions can continue to process invocation events, but are no longer able to be updated, contributing to a high error rate. With the scheduled deprecation of Node.js 8.10 in February, we expect the error rates to climb for this version, as well.
- Both Python 2.7 and 3.6 are still supported by AWS, but still displayed error rates over 1%.

Timeouts by Runtime

Python 3.8	5.40%
Ruby 2.5	3.36%
Node.js 12.x	3.09%
Python 2.7	2.96%
Custom Runtime	2.95%
Node.js 6.10	2.32%
.NET Core 1.0	2.00%
Python 3.6	1.93%
Node.js 4.3	1.44%
Python 3.7	1.41%
Node.js 8.10	0.94%
.NET Core 2.1	0.85%
Go 1.x	0.78%
Node.js 10.x	0.77%
.NET Core 2.0	0.42%
Java 8	0.36%

Time frame July-December 2019

Insights Snapshot

- Java is used heavily for large volume data process workloads which don't often incurs cold starts, meaning these are likely highly tuned and don't run into timeouts often.
- Node and Python languages could run into timeouts more often since they're more likely to be CRUD operations like APIs, which can equal more connections to third-party services and databases that are more likely to cause timeouts.
- Timeout errors can be attributed to a number of different factors, including not adjusting default timeout limits. While the default execution time limit of 3 seconds can be ideal for customer-facing, event-driven functions, more-complex functions require longer to execute. There are far too many user variables and requirements to make meaningful correlations between timeouts and runtime versions.

SERVERLESS SPOTLIGHT: **AN INTERVIEW WITH**

SHEEN BRISALS

SENIOR ENGINEERING MANAGER, THE LEGO GROUP

What are the biggest organizational challenges facing serverless adoption in 2020?

Developing the organizational mind-shift to see serverless as an ecosystem of managed services as opposed to seeing serverless as just a bunch of Lambda functions.

It is often challenging (or even impossible) to show upper management a convincing like-for-like cost comparison of an on-prem or hosted environment versus a serverless estate. Without this direct comparison, it often delays the migration to serverless or management buy-in into serverless.

Indirect expenditure—serverless services may reduce the cost in most cases. However, adding support solutions such as monitoring and observability, may bring in more cost complexities. For example, the pricing model of the observability tools (based on events and data points) worries me as this can shoot up pretty fast as the serverless ecosystem grows. Often the cost-saving gained by moving to serverless is quashed by paying for these solutions. This may trigger an anti-serverless sentiment.

Lastly, over-glorifying the term "vendor lock-in."

What are the biggest technical challenges facing serverless adoption in 2020?

Tooling—This is getting better, but there is still quite a lot of confusion and inconsistency that leaves teams in limbo.

Observability—Specialist vendor solutions are becoming active with many offerings, but from an average team or small startup point of view, these solutions are rather expensive (in my opinion).

Confusion—The blessing of having many similar services capable of doing the same things creates confusion among engineers and early adopters. This sometimes slows down the momentum.

Also, typically, teams moving from a traditional development setup to cloud and serverless look for a perfect solution, platform, and ops environment to launch their serverless adoption. This causes unintended delay and friction.

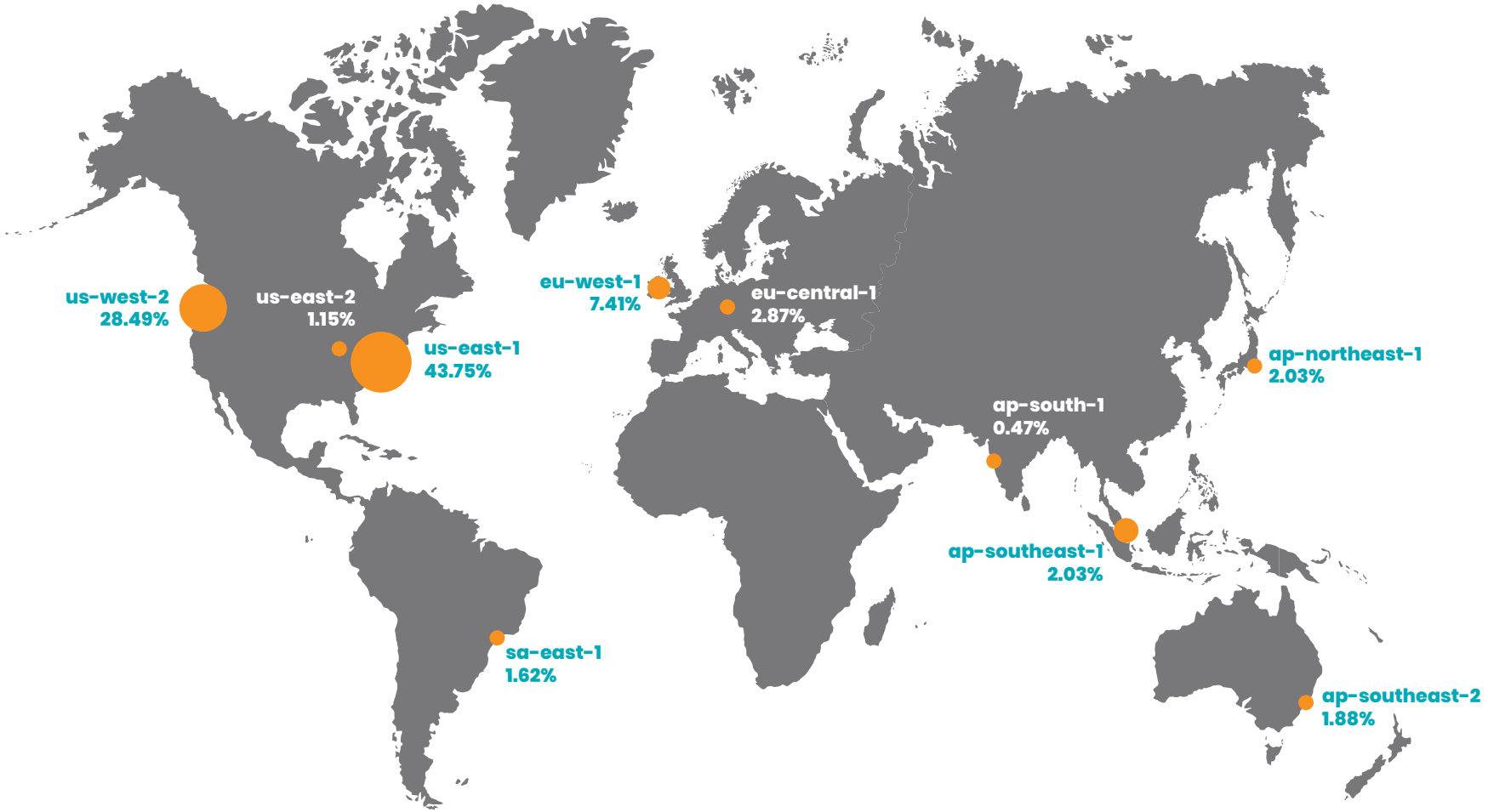
Where do you see serverless heading in 2020?

For starters, functionless prominence, i.e., fewer functions or more codeless opportunities. Becoming more and more event-driven and reducing the need to add functions for everything.

Also, [I expect to see] more industry adoption for all the right reasons. AWS is gradually bringing down the barrier between the accessibility of traditional server resources and services from the serverless side of the fence. Meaning, you don't need to opt for a container-based solution simply to access non-serverless services. They're not fully there yet, but they are well on the path.

And I'm excited about the prospect of AWS EventBridge taking center stage and orchestrating the functionless, event-driven culture among custom and AWS services and also with approved partner solutions. I am hoping to see more features added to EventBridge.

Invocation Percentages by Region



Insights Snapshot

- With Lambda now available in almost every AWS region around the globe, we were not surprised to see functions running in Europe, North America, and Asia.
- Regions in the United States accounted for the majority of invocations in our sample data set. US West (Oregon) and US East (N. Virginia) made up more than 70% of monitored invocations.

Time frame July-December 2019

Error Rates by Region

Insights Snapshot

- The Europe (Ireland) Availability Region showed the highest error rate at 6.81%, which could be attributed to a higher percentage of Lamba functions in staging, particularly with a smaller percentage of invocations versus the other regions.
- Asia Pacific (Sydney) and Asia Pacific (Singapore) regions displayed the next highest error rates at over 5%.

Time frame July-December 2019

Function Faster on AWS Lambda

As serverless adoption rates and complexity of use cases continue to rise, observability is a crucial component for developers building applications at scale on serverless architectures.

Learn more about how to monitor, visualize, troubleshoot, and alert across all your Lambda functions in a single experience with **New Relic Serverless** and get started for free.

New Relic