STATE OF SERVERLESS TODAY

Adopting serverless computing relieved many DevOps teams from the responsibilities of purchasing, provisioning, and managing backend servers, instead allowing their teams to focus on code. For many developers, serverless architecture offered increased scalability, flexibility, and faster time to release while reducing costs and operational overhead. These much-touted benefits are the drivers behind the fast adoption of services like AWS Lambda, Microsoft Azure, and Google Cloud Platform.

However, serverless computing is not a magic bullet for all web application developers. This technology has only been widely available since AWS Lambda was introduced in 2014 as the first public cloud infrastructure vendor with an abstract serverless computing offering. Even though in tech years 6 seems like 60, serverless technology is young and comes with its own growing pains like testing and debugging challenges, vendor lock-in risks, performance issues, and security concerns. Since serverless has become a viable option all the way up to the enterprise level, CIOs and DevOps teams have had to weigh these known risks against the benefits of no longer having to manage servers, databases, queues, and even containers as well as the cost savings of the pay-as-you-go economic model.

To better understand where industry leaders, SMEs, and performers see serverless taking us in the future, we have pulled together some of the most interesting and insightful data, reports, and opinions all in one place to keep you informed about one of the biggest topics in the tech world today.

Industry Analysis

In a survey conducted by O'Reilly in June 2019, more than 1,500 respondents across various job titles, industries, and the globe shared their thoughts and experiences on the impacts that serverless architecture is having on their industries.

O'Reilly Key Findings

- 40% of respondents work at organizations that have adopted serverless architecture in some form or another. Reduced operational costs and automatic scaling are the top serverless benefits cited by this group.
- Of the 60% of respondents whose companies haven't adopted serverless, the leading concerns about the paradigm are security and fear of the unknown.
- About 50% of respondents who adopted serverless three-plus years ago consider their implementations successful or extremely successful, a contrast to the 35% of those adopting serverless a year or less ago experiencing a successful or extremely successful implementation—a gap that suggests serverless experience pays off.
- Respondents who have implemented serverless made custom tooling the top tool choice—implying that vendors' tools may not fully address what organizations need to deploy and manage a serverless infrastructure.

The survey demonstrates there is still a lot of market share for serverless vendors to gain, but even respondents who have not taken the plunge into serverless adoption were curious enough to participate in the survey. Respondents also expressed hesitation to adopt because of significant concerns including training/upskilling existing staff, vendor lock-in, and the difficulties of integration and testing as well as security and managing this new type of infrastructure.

In an excellent state of the serverless address from June 2019, UC Berkley Grad Student Chenggang Wu discusses both the benefits and shortcomings of existing serverless offerings including real world examples. He then projects forward to the future and highlights challenges that must be overcome to realize truly general-purpose serverless computing.

Key topics:

- Capabilities and limitations of FaaS (Function-as-a-Service)
- Performance penalties of FaaS
- Poor consistency guarantees
- Lack of inbound network connections
- Stateful serverless computing
- Logical disaggregation with physical co-location
- Casual Consistency
- The future of cloud programming
- Moving forward from FaaS

Andrea Passwater of Serviceless Blog addresses a major drawback of serverless applications and the incompatibility between vendor functions and programming languages. She describes a not too distant multi-cloud future capable of cobbling together your favorite aspects from different severless providers and customizing your own solution, therefore eliminating the common

complaint of vendor lock-in risks.

In Andrea's dreamscape, she suggests serverless multi-cloud could be made easier by implementing three series of things:

- cross-cloud service compatibility
- shims for polyglot language support
- smart data routing

Lastly, don't miss the deserved compliment to Microsoft and Amazon for pushing out serverless features over the past two years faster than a start-up.

While you are over there, Serverless Blog also did a survey of their own to find out more about how their readers are adopting serverless architectures, what problems they're encountering, and how they feel about the future of serverless.

Out of 137 responses, 50% stated that they are using serverless architectures for work, while 21% are using them for a side project, and 22% have experimented with them but are not actually using them on a project yet. As for specific uses of serverless architecture, web server/API comes in at 65%, data processing came in at 34%, while internal tooling, IoT, and chatbots were all marked as use cases by over 20% of respondents while 33% fall into unlisted categories but most commonly, mobile backends.

It is no surprise that when asked which service provider respondents used, AWS Lambda came in far ahead with 96% since it is the most mature service; however, newer players to the market are constantly adding new functionalities and capabilities.

Vendor rankings:

- AWS Lambda - 96%
- Azure Functions - 6%
- Google Cloud Functions - 4%
- Webtask - 2%
- OpenWhisk - 2%

Overall 61% of respondents gave high marks about their optimism in the future of serviceless.

In a very fresh report from February 2020, Datadog does some heavy lifting by examining the serverless usage of thousands of companies to provide a look at how (and how much) serverless is being used in the real world, specifically about AWS Lambda. Click over to the report itself if you are a fan of easy to read graphs and charts you would expect from a name like Datadog.

Key findings:

- Half of AWS users have adopted Lambda
- Lambda is more prevalent in large environments
- Container users have flocked to Lambda
- Amazon SQS and DynamoDB pair well with Lambda
- Node.js and Python dominate among Lambda users
- The median Lambda function runs for 800 milliseconds
- Half of Lambda functions have the minimum memory allocation

- Two-thirds of defined timeouts are under 1 minute
- Only 4 percent of functions have a defined concurrency limit

In New Relic's Serverless Technology Semiannual Report, they have aggregated and analyzed a sample set of data over time, calling out key trends about their AWS Lambda serverless users. The report includes insightful data and every page includes opinion and insight from serverless industry SMEs.

Key Insights:

- 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.