aws | Deloitte.

Determining the total cost of ownership: comparing serverless and server-based technologies

September 2023

Authors: Gary Arora, Wendy Choi, Nik Jethi, and Bismay Kumar Sahoo

Deloitte Industry Insight

Adoption of serverless strategies is on the rise. In fact, over 75 percent of organizations surveyed report that they have either implemented a serverless strategy or are planning to do so in the next two years (**451 Research**). Current customers are also doing more with serverless technologies—Amazon Web Services (AWS) users ran workloads on **AWS Lambda** 3.5 times more in 2021 than they did in 2019 (**Datadog**).

The popularity of a serverless strategy is growing because it provides the opportunity for faster time to market by dynamically and automatically allocating compute and memory based on user requests. For most use cases, serverless allows customers to focus on business logic and writing code that provides customer value, as opposed to spending time on managing infrastructure, application scaling, building service integrations, and so on. It also provides cost savings through hands-off infrastructure management, which enables organizations to redirect IT budgets and development resources from operations to innovation. The pay-for-use model with serverless technologies leads to a shift from large capital expenditure lockup to flexible, on-demand consumption, allowing users to scale, customize, and provision computing resources dynamically to meet their exact needs. This, in turn, increases business agility.

But predicting costs in a pay-for-use model can be difficult if the inputs are variable. And prospective customers want to optimize costs by comparing server- or virtual machine (VM)–based computing with serverless options. In 2019, we introduced a framework for comparing the total cost of ownership (TCO) for both serverless and server-based applications, factoring in infrastructure, development, and maintenance costs. In that analysis, we concluded that while infrastructure costs, often referred to as the "cost to run" the application workloads, may seem higher with a serverless approach, the TCO is significantly lower due to savings in infrastructure, development, and maintenance costs.

Since 2019, AWS has introduced cost optimizations for both server-based computing with **Amazon Elastic Compute Cloud** (Amazon EC2) and serverless technologies, including Lambda and **Amazon Elastic Container Service** (Amazon ECS) with **AWS Fargate**, the serverless compute engine built on Amazon ECS, and others.

New cost optimizations for Lambda

1. Lambda is now included in the Compute Savings Plans, a flexible pricing model that allows customers to save up to 17 percent in exchange for making a commitment to a consistent amount of compute usage ($10 hour, for example) for a one-year or three-year term.
2. Lambda duration billing changed granularity from 100 ms down to 1 ms, which lowers the cost for most Lambda functions. This is especially pronounced for short-duration Lambda functions, where customers can save up to 70 percent.
3. Lambda now supports function sizes up to 10 gigabytes of RAM and 6 vCPU, allowing customers to reduce costs for resource-constrained, compute-intensive workloads.
4. Lambda AVX2 support allows customers to save up to 30 percent on compute-intensive, vectorizable workloads.
5. Lambda is now included in the **AWS Compute Optimizer**, which allows customers to easily identify and remediate inefficient configurations.
6. Lambda cost can be reduced with event filtering by configuring the event source with a filter criterion that reduces the number of messages that are used to invoke the Lambda function.
7. Finally, with tiered pricing (based on compute duration measured in gigabytes-seconds), customers who run large workloads on Lambda can automatically save on their monthly costs. So customers with an x86 architecture for a 6–15 billion compute duration (gigabytes-seconds) get a 10 percent discount, and for over 15 billion, a 20 percent discount. Similarly, for an arm64 architecture for compute duration (gigabytes-seconds) in the 7.5–18.75 billion range, customers get a 10 percent discount, and for over 18.75 billion, a 20 percent discount.

New cost optimizations for Amazon ECS with Fargate

1. AWS offers Spot Instances for Fargate tasks, which utilize spare compute capacity that is available at a lower price than On-Demand Instances. By using Spot Instances, you can run interruption-tolerant Amazon ECS tasks at up to a 70 percent discount off the Fargate price.
2. AWS introduced Fargate Savings Plans, a discount model that provides you with the same discounts as Reserved Instances in exchange for a commitment to use a specific amount (measured in dollars per hour) of compute power over a one-year or three-year period.
3. **AWS Graviton** processors are designed by AWS to deliver the best price performance for your cloud workloads running on Amazon EC2, AWS managed containers, and other managed services. AWS Graviton provides up to 40 percent better price performance over comparable x86-based instances. AWS Graviton processors are more energy efficient and use up to 60 percent less energy for the same performance than comparable Amazon EC2 instances.
4. Fargate task placement strategies allow you to define rules for how your tasks are placed on the underlying infrastructure. By using these strategies, you can optimize your placement of tasks and reduce costs by avoiding over-provisioning resources.
5. Just like Lambda, Fargate is also now included in the Compute Optimizer, allowing customers to easily identify and remediate inefficient configurations.

Introduction to the serverless TCO framework

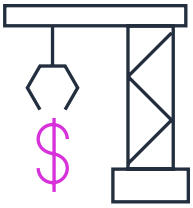
Serverless technologies effectively shift operational responsibilities to a cloud services provider, and organizations are applying this philosophy across the entire application stack, including compute, storage, integration, and network. With a serverless operational model, there are no servers to provision, patch, or manage and there is no software to install, maintain, or operate. In summary, a serverless model enables enhanced scalability, agility, and resiliency and allows developers to focus on core value-added tasks. Many organizations that take advantage of serverless technologies can deploy more frequent releases of their products and services, thereby impacting faster time to market and accelerated revenue growth.

Based on our extensive experience working with Fortune 100 clients across industries, we have developed a serverless TCO framework to evaluate the true cost of running a net-new application using serverless technologies, such as **Lambda**, **Amazon DynamoDB**, or **Amazon ECS with Fargate**, compared to a server-based compute, such as **Amazon EC2**. The serverless TCO framework is composed of three key cost components: infrastructure, development, and maintenance.

1. **Infrastructure cost** is a charge incurred from hosting an application workload on a cloud service provider, in this case, AWS.

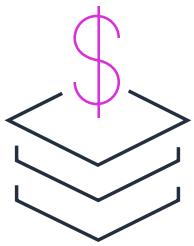
   The detailed section in this paper emphasizes three Deloitte client examples:

   a. Comparing Lambda functions to Amazon EC2 instances for a transportation client
   b. Comparing DynamoDB to running NoSQL on Amazon EC2 for a healthcare client
   c. Comparing Amazon ECS with Fargate to running a server-based application with Amazon EC2 and **Amazon Relational Database Service** (Amazon RDS) for a retirement wealth management client



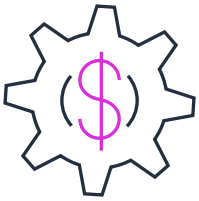
2. **Development cost** is the upfront charge of building and developing a new application on a cloud-based service.

   The detailed section in this paper highlights Deloitte's industry experience estimating development time and the cost of an average development resource.



3. **Maintenance cost** is the expense of the day-to-day operations associated with running and maintaining an application on an Amazon EC2 instance versus serverless architecture.

   The detailed section in this paper shows typical Deloitte benchmarks for maintenance costs across various components, including server-based security, patching, service tickets, and testing teams.



While there are organizational benefits to adopting a serverless strategy, such as increased velocity to address business opportunities, better planning of infrastructure capacity, and so on, this paper focuses solely on the cost elements highlighted above.

Infrastructure

This is the first major cost component, comprising the compute, storage, and network services consumed by host application workloads in the AWS Cloud. Infrastructure costs are often referred to as the "cost to run" the application workloads.

- Compute costs in an Amazon EC2 environment are calculated based on the maximum number of requests an instance can process per second, the number of servers needed to accommodate peak traffic (web, apps, database), and the time period an instance is active.
- In a serverless model, infrastructure cost is based on actual execution time—that is, the application owner is only charged when the code is executed, effectively achieving 100 percent server utilization (Lambda, for example, is charged based on the number of requests and the duration of the execution in milliseconds).
- In addition, leading practices such as high availability and fault tolerance, load balancing, and security services are included in the serverless architecture; however, those services would require additional charges in the server-based cloud execution environment.

To analyze the infrastructure costs, we used three real-life client examples:

CASE STUDY 1:

Transportation organization evaluates Lambda compared to Amazon EC2

Overview

The average commuter for this transportation client spends about two hours in transit, during which they can book tickets online, connect to Wi-Fi, and monitor their trip in real time. Now multiply that by millions of passengers annually, hundreds of destinations, and thousands of routes. Transportation organizations supporting these passengers typically struggle with legacy systems that are expensive to support and update. This can lead to increasingly unpredictable and slow response times, causing reports to be delayed and obsolete. These organizations are increasingly moving to a serverless model to reduce the burden of infrastructure management. This is made possible by utilizing a host of microservices that only execute when needed, producing the required data rapidly and seamlessly.

For the purposes of this paper, we compared the costs incurred by the client as they evaluated whether to run its ticket booking system through Lambda functions or on server-based Amazon EC2 instances.

Cost calculations

The transportation organization chose Lambda, among other serverless components, to process bookings and tickets for all its users. With about 1.5 million daily transactions, this application consumed about $1,154 per month in combined infrastructure costs between two Lambda functions, as shown in the table below. Due to architectural requirements and decoupled microservices best practice guidance, two separate Lambda functions were needed—one for ticket processing for the downstream customer and the other for data processing and validation. If the same application were deployed to run on a server-based infrastructure, we assumed the need for three i3en.large Amazon EC2 instances. The cost to run this application amounted to $1,169.

Monthly compute costs for Amazon EC2 stacks compared to Lambda functions are as follows:

Compute costs	Server-based cloud (EC2) 3 x i3en.large, VPC, load balancing	Serverless (Lambda) 512 MB, 1.5M requests/day
Booking and Ticketing App (avg. response time 1 sec)	$584	$390
Data Processing and Validation App (avg. response time 2 secs)	$584	$764
Total Monthly Cost	**$1,169**	**$1,154**
	Monthly Cost Difference	**$15**
		Serverless is 1% cheaper than a server-based solution

CASE STUDY 2:

Healthcare organization evaluates DynamoDB compared to hosting MongoDB on Amazon EC2

Overview

Healthcare organizations store data about millions of consumers and billions of claims. On any given day, millions of claims (new and updates to existing) are entered into the database and made available through APIs to several downstream applications that read data at 500 requests per second throughout the day. The challenge for IT in such organizations is to provide seemingly limitless capacity for dynamic querying capabilities. Those with on-premises or server-based environments lose the ability to scale automatically while maintaining throughput in the event of an unpredictable surge.

For the purposes of this paper, we compared the costs incurred by the client as they evaluated whether to run data queries through a popular NoSQL database—MongoDB—running on Amazon EC2 instances or with DynamoDB.

Cost calculations

The healthcare organization employed DynamoDB, a serverless key-value and document database that can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second. Based on the simplified assumption that the client needs to execute about 500 read and write queries per second, the infrastructure cost of running DynamoDB was about $840 per month. For a similar data platform deployed to run in server-based Amazon EC2, we assumed the need for three d2.xlarge Amazon EC2 instances running NoSQL databases with provisioned IOPS SSD. The table below shows a comparative analysis between the two platforms for this scenario.

Monthly compute costs for NoSQL databases on Amazon EC2 compared to DynamoDB are as follows:

Server-based cloud (MongoDB) d2.xlarge x 3, 2 TB provisioned IOPS SSD	Serverless (DynamoDB) 500 reads/writes per second, 2 TB storage, 5 KB/item
$2,369	$2,417
Monthly Cost Difference	**($49)**
	Serverless is 2% cheaper than a server-based solution

In the first case, the monthly cost to run the transportation workload on server-based and serverless infrastructure is similar. In the second case, the serverless database is clearly a winner in terms of the monthly cost. However, to understand the total cost of running an application, we need to compare not just the infrastructure costs but also the development and maintenance costs, which will be further elaborated on in the “Development” section of this publication.

CASE STUDY 3:

Retirement wealth management organization evaluates Amazon ECS with Fargate compared to Amazon EC2 for its wealth management web application

Overview

Retirement wealth management organizations store data on investors' financial contributions, stock performance, and client account balances. On any given day, investors can utilize wealth management applications to understand their current portfolio value, make withdrawls, and reinvest in different investment funds. As the application scales, IT struggles with constantly re-provisioning Amazon EC2 instances and Amazon RDS databases for an ever-increasing peak capacity. Other IT administrative tasks such as patching, security, and migrating to the latest operating systems can become expensive. IT organizations face challenges with provisioning, scaling, and monitoring underlying instances of their applications, all while dealing with unpredictable demands that can cause usage spikes and make managing for cost difficult.

For the purpose of this paper, we compared the cost incurred by a retirement wealth management client as its financial operations team evaluated whether to continue manually scaling its existing server-based application or to rearchitect the application using containers running on Amazon ECS with Fargate.

Cost calculations

The retirement wealth management organization selected Amazon ECS with Fargate, among other serverless options for their application. For the server-based application, they assumed nine i3en.large Amazon EC2 instances and three db.x2g.large Amazon RDS databases. This application is expected to handle 10,000 users daily, averaging about 20-minute visits each. Over one month of time, the serverless application consumed about $7,389 in cost. For the serverless version, they assumed a static webpage hosted in an **Amazon Simple Storage Service** (Amazon S3) bucket distributed via an **Amazon CloudFront** distribution network, supported by Fargate cluster middleware. They assume these Fargate pods will last 20 minutes on average across 25 pods per hour with 20 gigabytes of ephemeral storage. Fargate clusters will be programmed to manage the read and write of data to and from an **Amazon Aurora Serverless** 2 terabyte database. Because this application tends to experience bursts of traffic during the daytime and on weekends, the utilization of vCPU and memory is far higher with Fargate than with the server-based alternative.

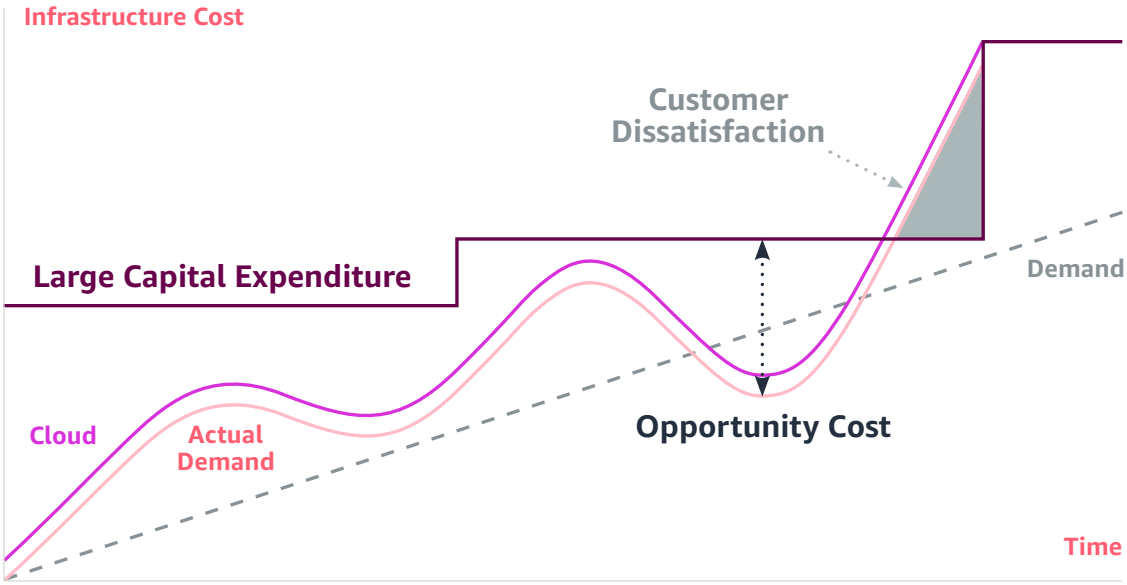
Monthly compute costs for a server-based application (Amazon EC2, Amazon RDS) compared to a serverless configuration (Fargate, Aurora Serverless) are as follows:

Compute and transfer costs	**Server-based** UI and middleware: 9x i3.enlarge, API gateway backend: 3x db.x2g.large, VPCs transfer cost included	**Fargate serverless** UI: Static webpage stored in S3, CloudFront CDN middleware: API gateway, VPC, Fargate backend: Aurora Serverless transfer cost included
Retirement Wealth Management Application	$7,389	$ 3,800
Monthly Cost Difference		**($3,589)**
		Amazon ECS with Fargate is 49% lower in cost than server-based Amazon EC2 and Amazon RDS

Development

The second major cost component, development, is a one-time upfront cost that can be quantified by the time and effort required for preplanning the application build. This is often referred to as the "cost to achieve" migration to the cloud. Using Amazon EC2 instances, one must determine how the architecture should scale to support the application over time. However, in a serverless environment, the capacity is automatically scaled so that fluctuations in demand are accommodated. As expected, if we underscale an EC2 instance, we will be faced with challenges surrounding our ability to provide enough capacity when we experience peak activity. However, if we overscale an EC2 instance, we will be spending more money than required, as we have underutilized capacity.

The diagram below highlights the benefit of serverless in dynamic scaling to track the actual usage requirements.



An Amazon EC2 environment secures some fixed capacity based on projected forecasts and does not scale as dynamically as serverless applications do. This can lead to waste in spend when capacity is over-provisioned (opportunity costs) and customer dissatisfaction when capacity does not meet usage requirements.

- Developers using Amazon EC2 instances need to spend considerable time evaluating challenges that the IT architecture could face at scale and determine what trade-offs need to be made upfront.
  - › The cost incurred for this preplanning includes the number of resources involved and the cost of both resources and time.
- Additional costs are incurred due to developer time spent setting up network and load balancers, provisioning automatic scaling, planning for availability (selecting the right number of availability zones), and purchasing licenses and software.

A serverless application leverages an event-based architecture, thereby allowing development teams to start developing the application rather than planning a robust deployment architecture. The table below summarizes the typical savings we have seen from the reduction of time required to provision applications on serverless as compared to server-based Amazon EC2 instances. On average, a serverless environment takes 68 percent less time to provision as compared to a server-based environment, which can equate to hundreds of dollars in savings per month per application.[1]

One-time development cost for server-based (Amazon EC2) compared with serverless (Lambda)

Development	Server based	Serverless	Difference
Days to deploy	~25 days	~8 days	~17 days
One-time upfront	$38,300	$12,300	$26,000
Monthly cost	$640	$205	$(435)
			Serverless is 68% cheaper than server-based application

Monthly cost calculated from annual FTE cost per month amortized over 5 years

- Days to deploy new compute/storage
- Traditional: Three developers took 4–5 weeks
- Serverless: Three developers took 8–9 days
- FTE Rate $120k/year, eight hours/workday
- One-time fee is for three developers, eight-hour days
- Monthly cost is assumed to amortize over five years
- Net-new application—no application migration costs
- Assuming the right talent exists, no additional cost to hire/train developers
- Building a stateless application
- Not a consistently high memory usage application
- Not a consistently high CPU application
- Not a near real-time application—for example, running in stock exchanges
- One can deploy the app on Amazon EC2 or Lambda

[1] Carvalho, L., et al., "Generating Value Through IT Agility and Business Scalability with AWS Serverless Platform," IDC, November 2018

Maintenance

The third major cost component, maintenance, considers the time and resources spent on ongoing tasks once the application is deployed in production, which is commonly referred to as the "cost to support" an application. Maintenance cost can be categorized as time spent by a developer in four areas:

1. Provisioning and scaling of applications
2. Security implementation (hardening of AMIs)
3. Patching and operating system (OS) updates
4. Ongoing application operations, such as delivering/adding new features, monitoring, logging, verifying, and testing

Although numbers vary based on the client organization and the nature of the application, Deloitte estimates that an application developer spends on average 8–10 hours a month on application provisioning, security implementation, and patching and OS updates. An additional 40 hours a month is spent on application monitoring, logging, verifying, and testing when running Amazon EC2 services.

With serverless services, such as Lambda, Amazon ECS with Fargate, and DynamoDB, most of these maintenance tasks are no longer required because these services are fully managed by the cloud provider. This allows developers to focus their time and resources on developing the core capability to create or build their business versus focusing on reboots and reconfigurations on the servers themselves.

- In a server-based Amazon EC2 model, teams would need to open service tickets and patching teams would reach out to developers to patch the environment, both of which could delay development activities.
- In a serverless model utilizing Lambda, these types of patches and other related activities happen behind the scenes and so do not impact core development.

Additionally, serverless implementations can digitize many security rules, thus making them more secure and eliminating the need for human intervention and resource requirements, such as housing a dedicated team to handle special provisioning of firewall licenses and host scanning. The table below summarizes the additional time (in hours per month) it may take an app developer to perform application maintenance.

Ongoing maintenance efforts for server-based (Amazon EC2) compared to serverless (Lambda) for an entire application portfolio

Maintenance costs	Server-based cloud (hours)	Serverless (hours)
Provisioning and scaling	8	1
Security implementation	8	1
Patching and OS updates	8	1
Ongoing application operations	40	8–32
Monthly development cost in app maintenance	$4,096	$704–$2,240
	Monthly Cost Difference	$(3,392)–$(1,856)
	% Savings (moving to serverless from EC2)	45%–80%

Assumptions:

Monthly Cost Calculated from Annual FTE Cost per Month

- FTE rate $120k/yr, $64/hr

Enhancements

Both serverless and server-based services have evolved since the first iteration of this paper. Most enhancements for server-based services have been around efficient pricing models, such as Amazon EC2 Instance Savings Plans and Compute Savings Plans, and hardware, such as new instance types tuned for machine learning (ML), graphics-intensive workloads, high memory for organization COTS, and efficient networking. While some of these pricing efficiencies have also been introduced for Lambda, most enhancements for serverless services have been around launching new capabilities, such as provisioned concurrency in Lambda, edge computing with CloudFront, the introduction of an event bus with **Amazon EventBridge**, and container image support, among many others—all of which make serverless computing more accessible for new use cases, including event-driven applications (EDAs), ML-inference workloads, and more.

Conclusion

In the previous sections, we compared the TCO framework across three Deloitte client use cases to demonstrate how we evaluate the total cost of a net-new application on Amazon EC2 instances versus on serverless services. When considering only the infrastructure cost, running the application on EC2 instances is the more cost-effective choice. However, when we account for development and maintenance costs, it becomes significantly cheaper to run the application through serverless services, such as Lambda, DynamoDB, or Amazon ECS with Fargate. In all use cases, the client decided to build with serverless architectures to realize these cost savings. The table below summarizes the total combined costs across all three uses cases:

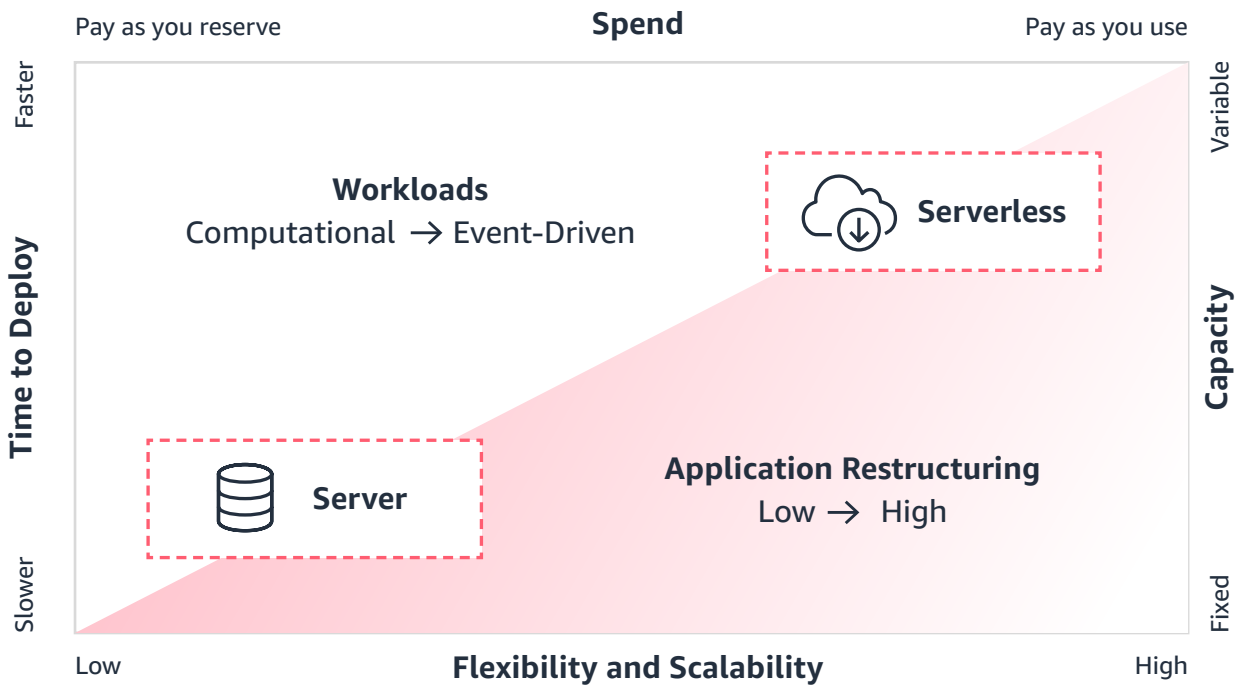
	Transportation organization		Healthcare organization		Retirement wealth management organization	
	EC2	Lambda	EC2	Serverless	EC2	ECS with Fargate
Infrastructure Cost ($/month)	$1,169	$1,154	$2,369	$2,417	$7,389	$3,800
	Difference	**($15)**	Difference	**$49**	Difference	**($3,589)**
Development Cost ($/month)	$640	$205	$640	$205	$640	$205
	Difference	**($435)**	Difference	**($435)**	Difference	**($435)**
Maintenance Cost ($/month)	$4,096	$2,240	$4,096	$2,240	$4,096	$2,240
	Difference	**($1,856)**	Difference	**($1,856)**	Difference	**($1,856)**
Total Cost ($/month)	$5,905	$3,599	$7,105	$4,862	$12,125	$6,245
	Difference	**($2,306)**	Difference	**($2,242)**	Difference	**($5,880)**
	Serverless is 39% cheaper than a server-based solution		**Serverless is 32% cheaper than a server-based solution**		**Serverless is 48% cheaper than a server-based solution**	

When considering the cost of running an application in a server-based cloud environment like Amazon EC2 over serverless services such as Lambda or DynamoDB or Amazon ECS with Fargate, it is important to consider the total cost of running the application, including the infrastructure, development, and maintenance costs, also referred to as the cost to run, cost to achieve, and cost to support the application. In isolation, each one of these cost components may provide an incomplete picture of the total cost, so a comprehensive comparison across all three cost components is needed to arrive at an accurate TCO.

Exceptions to the rule

Although there are many benefits to moving to serverless technologies, not all applications are the right fit for a serverless architecture. It is important to consciously select your technology stack and configure it in a cost-effective manner for serverless functions to yield the fully intended benefits. As the diagram on the following page illustrates:

- Applications with variable capacity and high scalability requirements are good candidates for a serverless strategy.
- A serverless strategy is best suited for web, mobile, and Internet of Things (IoT) apps, real-time analytics, and data processing.
- A serverless strategy may be least suited for long-running computational tasks, data migrations from relational to NoSQL, applications requiring significant disc space or RAM, and applications requiring SSH server access.



In summary, time spent on maintenance and ongoing operations is diminished in serverless applications, as this is fully managed by the cloud provider. As such, the roles of a dedicated operations team will need to evolve. Serverless architectures also provide infinite scalability and built-in high availability, which are additional efforts and costs in a server-based environment. When we compare only infrastructure costs across the platforms, we may determine that a server-based model is cost-effective. However, when we layer on the additional benefits and cost savings of a serverless model, we see that organizations can save significantly by constructing applications and overall organizational structures to effectively leverage a serverless architecture.

Produced in partnership with:

aws

This publication contains general information only, and Deloitte is not, by means of this publication, rendering accounting, business, financial, investment, legal, tax, or other professional advice or services. This publication is not a substitute for such professional advice or services, nor should it be used as a basis for any decision or action that may affect your business. Before making any decision or taking any action that may affect your business, you should consult a qualified professional advisor. Deloitte shall not be responsible for any loss sustained by any person who relies on this publication.

About Deloitte

Deloitte refers to one or more of Deloitte Touche Tohmatsu Limited, a UK private enterprise limited by guarantee ("DTTL"), its network of member firms, and their related entities. DTTL and each of its member firms are legally separate and independent entities. DTTL (also referred to as "Deloitte Global") does not provide services to clients. In the United States, Deloitte refers to one or more of the US member firms of DTTL, their related entities that operate using the "Deloitte" name in the United States and their respective affiliates. Certain services may not be available to attest clients under the rules and regulations of public accounting. Please see **www.deloitte.com/about** to learn more about our global network of member firms.

aws | Deloitte.