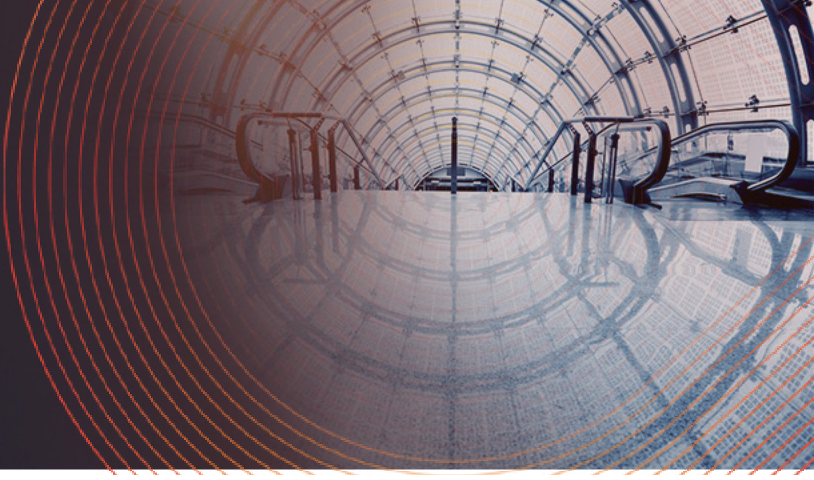
Deloitte.

Determining the Total Cost of Ownership: Comparing Serverless and Server-based Technologies

July 2021

Deloitte Consulting

Authors:
Gary Arora, Akash Tayal, Rakinder Sembhi



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—AWS users run serverless compute service 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. It also provides cost savings through hands-off infrastructure management, which enables organizations to redirect IT budget 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–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 may be higher with a serverless approach, the TCO is significantly lower due to savings in development and maintenance costs.

Since 2019, AWS has introduced cost optimizations for both server-based computing with Amazon EC2 and serverless technologies with AWS Lambda and others. We revisited this analysis to incorporate these optimizations and concluded that AWS Lambda is now more cost-effective than in the original analysis and that serverless technologies deliver lower TCO when compared to a server-based cloud execution model—with savings of 38–57 percent.

New Cost Optimizations for AWS Lambda

1. AWS Lambda is now included in the Compute Savings Plan, 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 (e.g., $10/hour) for a one- or three-year term.
2. AWS Lambda duration billing changed granularity from 100ms down to 1ms, 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. AWS Lambda now supports function sizes up to 10 GB of RAM and 6 vCPU, allowing customers to reduce costs for resource-constrained, compute-intensive workloads.
4. AWS Lambda AVX2 support allows customers to save up to 30 percent on compute-intensive, vectorizable workloads.
5. AWS Lambda is now included in the AWS Compute Optimizer, which allows customers to easily identify and remediate inefficient configurations.

Introduction of the Serverless TCO Framework

Serverless technologies effectively shift operational responsibilities to a cloud service 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 **AWS Lambda** or **Amazon DynamoDB**, 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 Amazon Web Services (AWS).

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

   a. *Comparing AWS Lambda functions versus Amazon EC2 instances for a transportation client*

   b. *Comparing Amazon DynamoDB versus running NoSQL on EC2 for a healthcare 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 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, etc., 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 on the AWS cloud platform. 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 a 100 percent server utilization (AWS 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/fault tolerance, load balancing, and security services are included in the serverless architecture, whereas those services would require additional charges in the server-based cloud execution environment.

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

Case Study 1: Transportation Organization Evaluates AWS Lambda versus 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 their ticket booking system through Lambda functions or on server-based EC2 instances.

Cost Calculations

The transportation organization chose AWS Lambda, among other serverless components, to process bookings and tickets for all its users. With about 1.5 million transactions per day, this application consumed about $1,142 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 EC2s. The cost to run this application amounted to $1,088.

Monthly compute costs for EC2 stacks versus Lambda functions can be compared 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)	$544	$384
Data Processing and Validation App (avg. response time 2 secs)	$544	$758
Total Monthly Cost	**$1,088**	**$1,142**
	Monthly Cost Difference	**$54**
		Serverless is 5% more expensive than a server-based solution

Case Study 2: Healthcare Organization Evaluates Amazon DynamoDB versus 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 EC2 instances or with DynamoDB.

Cost Calculations

The healthcare organization employed Amazon 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 $657 per month. For a similar data platform deployed to run in server-based Amazon EC2, we assumed the need for three d2.xlarge EC2s running NoSQL DB with provisioned IOPS SSD. The table below shows a comparative analysis between the two platforms for this scenario.

Monthly compute costs for NoSQL DB on EC2 versus DynamoDB can be compared as follows:

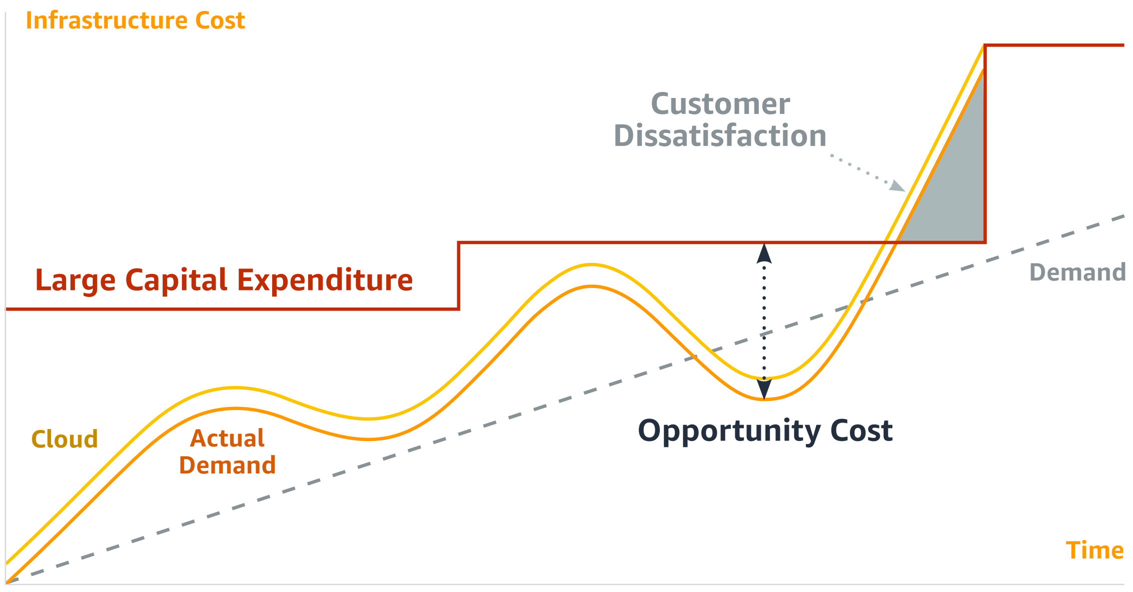
Server-based Cloud (MongoDB) d2.xlarge x 3, 2 TB Provisioned IOPS SSD	Serverless (DynamoDB) 500 reads/writes per second, 2 TB storage, 5 KB/item
$2,475	$657
Monthly Cost Difference	**($1,818)**
	Serverless is 73% cheaper than server-based solution

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, as shown below.

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 overprovisioned (opportunity costs) and customer dissatisfaction when capacity does not meet usage requirements.

- Developers using 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 versus 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 (EC2) Compared with Serverless (AWS 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

Monthly Cost Calculated from Annual FTE Cost per Month Amortized over Five Years

- Days to deploy new compute/storage
- Traditional: 3 developers took 4–5 weeks
- Serverless: 3 developers took 8–9 days
- FTE Rate $120K/year, 8 hours/workday
- One-time fee is for 3 developers, 8-hour days
- Monthly cost is assumed to amortize over 5 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—e.g., running in stock exchanges
- One can deploy the app on EC2 or Lambda

[1] IDC: Generating Value Through IT Agility & Business Scalability with AWS Serverless Platform

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 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 AWS Lambda and Amazon 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, all of which could delay development activities.

- In a serverless model utilizing AWS 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 (EC2) versus 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 EC2 Instance Savings Plans and Compute Savings Plans, and hardware, such as new instance types tuned for machine learning, graphics-intensive workloads, high memory for organization COTS, and efficient networking. While some of these pricing efficiencies have also been introduced for AWS Lambda, most enhancements for serverless services have been around launching new capabilities, such as provisioned concurrency in AWS Lambda, Edge Computing with Amazon 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, machine learning-inference workloads, and more.

Conclusion

In the above sections, we compared the TCO framework across two 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 AWS Lambda or Amazon DynamoDB. In both use cases, the client decided to build with serverless architectures to realize these cost savings. The table below summarizes the total combined costs across both uses cases:

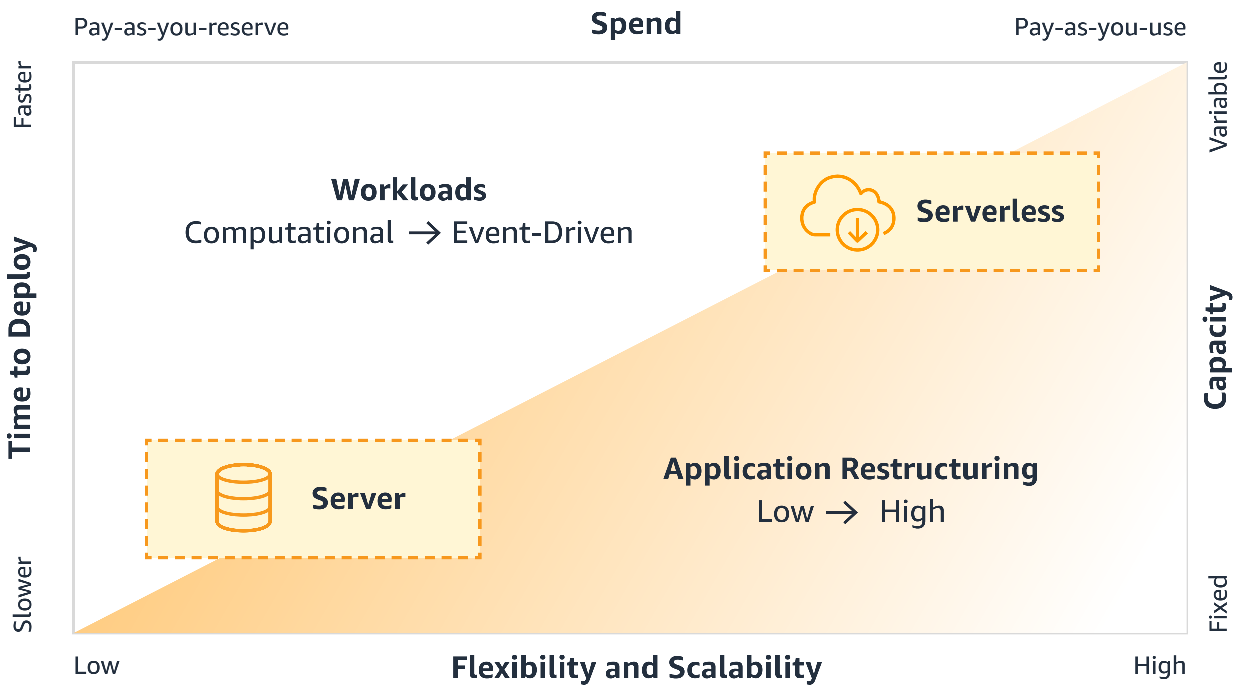
	Transportation Organization		Healthcare Organization	
	EC2	**Lambda**	**EC2**	**Serverless**
Infrastructure Cost ($/month)	$1,088	$1,142	$2,457	$657
	Difference	**$54**	Difference	**$(1,818)**
Development Cost ($/month)	$640	$205	$640	$205
	Difference	$(435)	Difference	$(435)
Maintenance Cost ($/month)	$4,096	$2,240	$4,096	$2,240
	Difference	$(1,856)	Difference	$(1,856)
Total Cost ($/month)	$5,824	$3,587	$7,193	$3,102
	Difference	$(2,237)	Difference	$(4,091)
	Serverless is 38% cheaper than server-based solution		**Serverless is 57% cheaper than 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 AWS Lambda or Amazon DynamoDB, 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 total cost of ownership.

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 below illustrates:

- Applications with variable capacity and high scalability requirements are good candidates for a serverless strategy.
- Applications that spend large amounts of time waiting for long operations outside of Lambda will continue paying for their API and other service calls, and as such are not ideal candidates for a serverless strategy.
- A serverless strategy is best suited for web, mobile, and 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.

Copyright © 2021 Deloitte Development LLC. All rights reserved.

Deloitte.