AWS Whitepaper

Optimizing Enterprise Economics with Serverless Architectures

Optimizing Enterprise Economics with Serverless Architectures: AWS Whitepaper

Table of Contents

Optimizing Enterprise Economics with Serverless Architectures

Publication date: **September 15, 2021** (*Document history*)

This whitepaper is intended to help Chief Information Officers (CIOs), Chief Technology Officers (CTOs), and senior architects gain insight into serverless architectures and their impact on time to market, team agility, and IT economics. By eliminating idle, underutilized servers at the design level and dramatically simplifying cloud-based software designs, serverless approaches rapidly change the IT landscape.

This whitepaper covers the basics of serverless approaches and the AWS serverless portfolio. It includes several case studies illustrating how existing companies are gaining significant agility and economic benefits from adopting serverless strategies. In addition, it describes how organizations of all sizes can use serverless architectures to architect reactive, event-based systems and quickly deliver cloud-native microservices at a fraction of conventional costs.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Serverless Application Lens, we focus on best practices for architecting your serverless applications on AWS.

In the HPC Lens, we focus on best practices for architecting your High Performance Computing (HPC) workloads on AWS.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Many companies are already gaining benefits from running applications in the public cloud, including cost savings from pay-as-you-go billing and improved agility through the use of on-demand IT resources. Multiple studies across application types and industries have demonstrated that migrating existing application architectures to the cloud lowers the Total Cost of Ownership (TCO) and improves time to market.

Relative to on-premises and private cloud solutions, the public cloud makes it significantly simpler to build, deploy, and manage fleets of servers and the applications that run on them. The public cloud has established itself as the new normal, with double-digit year-over-year growth since its inception.

However, companies today have options beyond classic server or virtual machine (VM) based architectures to take advantage of the public cloud. Although the cloud eliminates the need for companies to purchase and maintain their hardware, *any* server-based architecture still requires them to architect for scalability and reliability. Plus, companies need to own the challenges of patching and deploying to those server fleets as their applications evolve.

Moreover, they must scale their server fleets to account for peak load and then attempt to scale them down when and where possible to lower costs—all while protecting the experience of end-users and the integrity of internal systems. Idle, underutilized servers prove to be costly and wasteful. Researchers calculated the average server utilization to be around only 18 percent for enterprises.

Using serverless services, developers and architects can design and develop complex application architectures, focusing just on business logic without dealing with the complexity of servers.

As a result, product owners can achieve faster time to market with shorter development, deployment, and testing cycles. In addition, the reduction of server management overheads reduces the TCO, which ultimately results in competitive advantages for the companies.

With significantly reduced infrastructure costs, more agile and focused teams, and faster time to market, companies that have already adopted serverless approaches are gaining a key advantage over their competitors.

Understanding serverless architectures

The advantages of the serverless approaches cited above are appealing, but what are the considerations for practical implementation? What separates a serverless application from its conventional server-based counterpart?

Serverless uses managed services where the cloud provider handles infrastructure management tasks like capacity provisioning and patching. This allows your workforce to focus on business logic serving your customers while minimizing infrastructure management, configuration, operations, and idle capacity. In addition, Serverless is a way to describe the services, practices, and strategies that enable you to build more agile applications so you can innovate and respond to change faster.

Serverless applications are designed to run whole or parts of the application in the public cloud using serverless services. AWS offers many serverless services in domains like compute, storage, application integration, orchestration and databases. The serverless model provides the following advantages compared to conventional server-based design:

- There is no need to provision, manage and monitor the underlying infrastructure. All of the actual hardware and platform server software packages are managed by the cloud provider. You need to just deploy your application and its configuration.
- Serverless services have fault tolerance built-in by default. Serverless applications require minimal configuration and management from the user to achieve high availability.
- Reduced operational overhead allows your teams to release quickly, get feedback, and iterate to get to market faster.
- With a pay-for-value billing model, you do not pay for over-provisioning, and your resource utilization is optimized on your behalf
- Serverless applications have built-in service integrations, so you can focus on building your application instead of configuring it.

Is serverless always appropriate?

Almost all modern applications can be modified to run successfully, and in most cases, in a more economical and scalable fashion, on a serverless platform. The choice between serverless and the alternatives do not need to be an all-or-nothing proposition. Individual components could be run on servers, using containers, or using serverless architectures within an application stack. However, here are a few scenarios when serverless may not be the best choice:

- When the goal is explicitly to avoid making any changes to existing application architecture.
- For the code to run correctly, fine-grained control over the environment is required, such as specifying particular operating system patches or accessing low-level networking operations.
- Applications with ultra low latency requirements for all incoming requests.
- When an on-premises application hasn't been migrated to the public cloud.
- When certain aspects of the application component don't fit within the limits of the serverless services - for example, if a function takes more time to execute than the AWS Lambda function's execution timeout limit, or the backend application takes more time to run than Amazon API Gateway's timeout.

Serverless use cases

The serverless application model is generic and applies to almost any application, from a startup's web app to a Fortune 100 company's stock trade analysis platform. Here are several examples:

- **Data processing** – Developers have discovered that it's much easier to parallelize with a serverless approach, mainly when triggered through events, leading them to increasingly apply serverless techniques to a wide range of big data problems without the need for infrastructure management. (Source: Occupy the Cloud: Eric Jonas et al., *Distributed Computing for the 99%*, https://arxiv.org/abs/1702.04024.) These include map-reduce problems, high-speed video transcoding, stock trade analysis, and compute-intensive Monte Carlo simulations for loan applications.
- **Web applications** – Eliminating servers makes it possible to create web applications that cost almost nothing when there is no traffic while simultaneously scaling to handle peak loads, even unexpected ones.
- **Batch processing** – Serverless architectures can be used in a run multi-step flow-chart like use cases like ETL jobs.
- **IT automation** – Serverless functions can be attached to alarms and monitors to provide customization when required. Cron jobs (used to schedule and automate tasks that need to be carried out periodically) and other IT infrastructure requirements are made substantially simpler to implement by removing the need to own and maintain servers for their use, especially when these jobs and conditions are infrequent or variable in nature.
- **Mobile backends** – Serverless mobile backends offer a way for developers who focus on client development to quickly create secure, highly available, and perfectly-scaled backends without becoming experts in distributed systems design.

- **Media and log processing** – Serverless approaches offer natural parallelism, making it simpler to process compute-heavy workloads without the complexity of building multithreaded systems or manually scaling compute fleets.
- **IoT backends** – The ability to bring any code, including native libraries, simplifies the process of creating cloud-based systems that can implement device-specific algorithms.
- **Chatbots (including voice-enabled assistants) and other webhook-based systems** – Serverless approaches are perfect for any webhook-based system, like a chatbot. In addition, their ability to perform actions (like running code) only when needed (such as when a user requests information from a chatbot) makes them a straightforward and typically lower-cost approach for these architectures. For example, the majority of Alexa Skills for Amazon Echo are implemented using AWS Lambda.
- **Clickstream and other near real-time streaming data processes** – Serverless solutions offer the flexibility to scale up and down with the flow of data, enabling them to match throughput requirements without the complexity of building a scalable compute system for each application. For example, when paired with technology like Amazon Kinesis, AWS Lambda can offer high-speed records processing for clickstream analysis, NoSQL data triggers, stock trade information, and more.
- **Machine learning inference** – Machine learning models can be hosted on serverless functions to support inference requests, eliminating the need for owning or maintaining servers for supporting intermittent inference requests.
- **Content delivery at the edge** –By moving serverless events handing to the edge of the internet, developers can take advantage of lower latency and customize retrievals and content fetches quickly, enabling a new spectrum of use cases that are latency-optimized based on the client's location.
- **IoT at the edge** – Enabling serverless capabilities such as AWS Lambda functions to run inside commercial, residential, and hand-held Internet of Things (IoT) devices enables these devices to respond to events in near real-time. Devices can take actions such as aggregating and filtering data locally, performing machine learning inference, or sending alerts.

Typically, serverless applications are built using a *microservices architecture* in which an application is separated into independent components that perform discrete jobs. These components, made up of a compute layer and APIs, message queues, database, and other components can be independently deployed, tested, and scaled.

The ability to scale individual components needing additional capacity rather than entire applications can save substantial infrastructure costs. It would allow an application to run lean with minimal idle server capacity without the need for right-sizing activities.

Serverless applications are a natural fit for microservices because of their decoupled nature. Organizations can become more agile by avoiding monolithic designs and architectures because developers can deploy incrementally and replace or upgrade individual components, such as the database tier if needed.

In many cases, not all layers of the architecture need to be moved to serverless services to reap its benefits. For instance, simply isolating the business logic of an application to deploy onto the AWS Lambda, serverless compute service, is all that's required to reduce server management tasks, idle compute capacity and operational overhead immediately.

Serverless architecture also has significant economic advantages over server-based architectures when considering disaster recovery scenarios.

For most serverless architectures, the price for managing a disaster recovery site is near zero, even for warm or hot sites. Serverless architectures only incur a charge when traffic is present and resources are being consumed. Storage cost is one exception, as many applications require readily accessible data.

Nonetheless, serverless architectures truly shine when planning disaster recovery sites, especially when compared to traditional data centers. Running a disaster recovery on-premises often doubles infrastructure costs as many servers are idle waiting for disaster to happen.

Serverless disaster recovery sites can be set up quickly as well. Once serverless architectures have been defined with infrastructure as code using AWS native services such as AWS CloudFormation, an entire architecture can be duplicated in a separate region by running a few commands.

AWS serverless capabilities

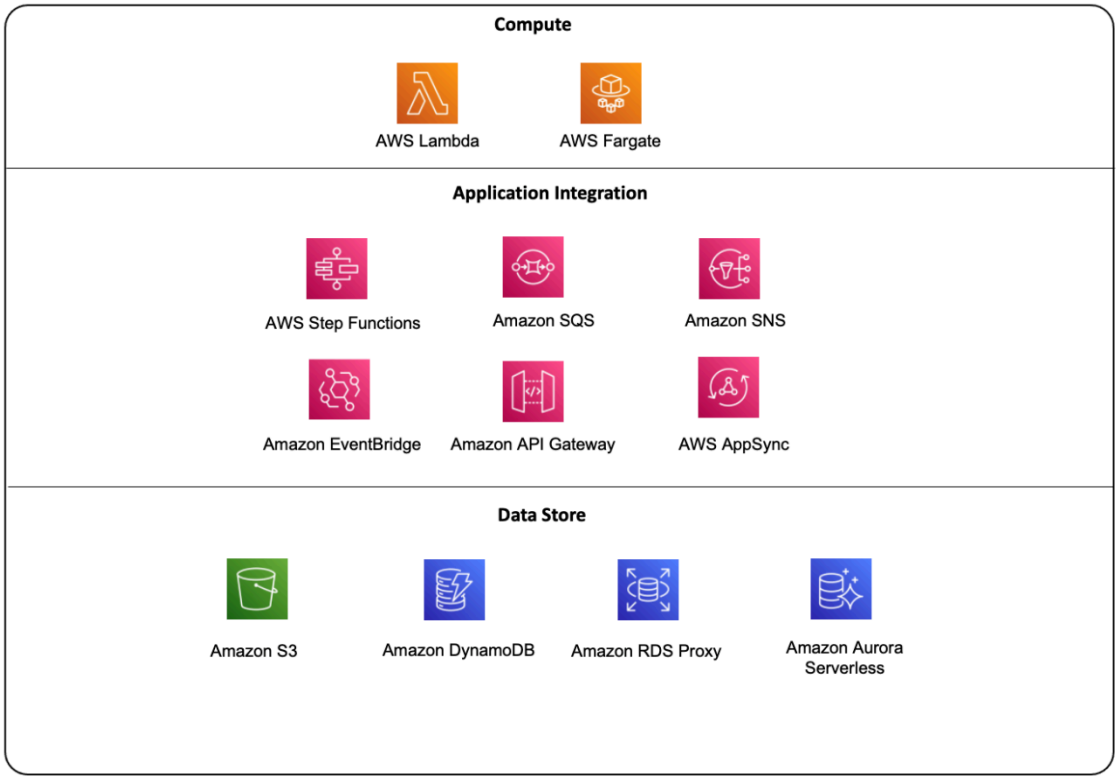
Like any other traditional server and VM-based architecture, serverless provides core capabilities such as compute, storage, messaging and more to its users. However, serverless services are distributed across multiple managed services rather than spread across software-installed virtual machines.

As a result, AWS provides a complete serverless application that requires a broad array of services, tools, and capabilities spanning storage, messaging diagnostics, and more. Each of these services is available in the developer's toolbox to build a practical application.

Service offerings

Since the introduction of Lambda in 2014, AWS has introduced a wide variety of fully-managed serverless services that enable organizations to create serverless apps that can integrate seamlessly with other AWS services and third-party services.

The launched serverless services include, but are not limited to, Amazon API Gateway (2015), Amazon EventBridge (2019), and Amazon Aurora Serverless v2 (2020). The pace of innovation has not stopped for individual services, as Lambda has had more than 100 major feature releases since its launch. The following figure illustrates a subset of the components in the AWS serverless platform and their relationships.



AWS serverless platform components

Serverless offerings from AWS consist of services that span across all infrastructure layers, including compute, storage, and orchestration. In addition, AWS provides tools needed to author, build, deploy, and diagnose serverless architectures.

Running a serverless application in production requires a reliable, flexible, and trustworthy platform that can handle the demands of small startups to global, worldwide corporations. The platform must scale *all* of an application's elements and provide end-to-end reliability.

Just as with conventional applications, helping developers create and deliver serverless solutions is a multi-dimensional challenge. To meet the needs of large-scale enterprises across various industries, the AWS serverless platform offers the following capabilities through a diverse set of services.

- **A high-performance, scalable, and reliable *serverless compute layer*** - The serverless compute layer is at the core of any serverless architecture, such as AWS Lambda or AWS Fargate, responsible for running the business logic. Because these services are run in response to events, simple integration with both first-party and third-party event sources is essential to making solutions simple to express and enabling them to scale automatically in response to varying

workloads. In addition, serverless architectures eliminate all of the scaling and management code typically required to integrate such systems, shifting that operational burden to AWS.

- **Highly available, durable, and scalable storage layer –** AWS offers fully managed storage layers that offload the overhead of ever-increasing storage requirements to support the serverless compute layer. Instead of manually adding more servers and storage, services such as Amazon Aurora Serverless v2, Amazon DynamoDB, and Amazon Simple Storage Service (Amazon S3) scales based on usage and users are only billed for the consumed resources. In addition, AWS offers purpose-built storage services to meet diverse customer needs, from DynamoDB for key-value storage, Amazon S3 for object storage, and Aurora Serverless v2 for relational data storage.
- **Support for loosely coupled and scalable decoupled serverless workloads –** As applications mature and grow, they become more challenging to maintain or add new features, and some transform into monolithic applications. As a result, they make it challenging to implement changes and slow down the development pace. What is needed is individual components that are decoupled and can scale independently. Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), Amazon EventBridge, and Amazon Kinesis enable developers to decouple individual components, allowing developers to create and innovate without being dependent on one another. In addition, these components all being serverless implies that customers are only being billed for the resources that each component is consuming, eliminating unnecessary resources being wasted.
- ***Orchestration* offering state and workflow management** – Orchestration and state management are also critical to a serverless platform's success. As companies adopt serverless architectures, there is an increased need to orchestrate complex workflows with decoupled components. AWS Step Functions is a visual workflow service that satisfies this need. It is used to orchestrate AWS services, automate business processes, and build serverless applications. Step Functions manage failures, retries, parallelization, service integrations, and observability so developers can focus on higher-value business logic. Building applications from individual components that perform a discrete function lets you scale easily and change applications quickly. Developers can change and add steps without writing code, enabling your team to evolve your application and innovate faster.
- ***Native service integrations* between serverless** services mentioned above, such as Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (Amazon SNS), and Amazon EventBridge, act as application integration services, enabling communication between decoupled components within microservices. Another benefit of these services is that minimal code is needed to allow interoperability between them, so you can focus on building your application instead of configuring it. For instance, integration between Amazon API Gateway -a fully

managed service for hosting APIs - to a Lambda function can be done without writing any code and simply walking through the AWS console.

Developer support

Providing the right tool and support for developers and architects is essential to boosting productivity. AWS Developer Tools are built to work with AWS, making it easier for teams to set up and be productive.

In addition to popular and well-known developer tools such as AWS Command Line Interface (AWS CLI) and AWS Software Development Kits (AWS SDKs), AWS also provides various AWS, open-source, and third-party web frameworks that simplify serverless application development and deployment.

This includes the AWS Serverless Application Model (AWS SAM) and AWS Cloud Development Kit (AWS CDK) (AWS CDK) that allows customers to onboard faster to serverless architectures, offloading undifferentiated heavy lifting of managing the infrastructure for your applications.

This enables developers to focus on writing code that creates value for their customers. In addition, AWS provides the following support for developers adopting serverless technologies.

- **A collection of fit-for-purpose *application modeling frameworks* –**Application modeling frameworks, such as the open specification AWS SAM or AWS CDK, enable a developer to express the components that make up a serverless application and enable the tools and workflows required to build, deploy, and monitor those applications. Both frameworks work nicely with the AWS SAM Command Line Interface (AWS SAM CLI), making it easy for them to create and manage serverless applications. It also allows developers to build, test locally, and debug serverless applications then deploy them on AWS. It can also create secure continuous integration and deployment (CI/CD) pipelines that follow best practices and integrate with AWS' native and third-party CI/CD systems.
- **A vibrant *developer ecosystem* that helps developers discover and apply solutions in a variety of domains and for a broad set of third-party systems and use cases** - Thriving on a serverless platform requires that a company be able to get started quickly, including finding ready-made templates for everyday use cases, whether they involve first-party or third-party services. These integration libraries are essential to convey successful patterns—such as processing streams of records or implementing webhooks—especially when developers are migrating from server-based to serverless architectures. A closely related need is a broad and diverse ecosystem that surrounds the core platform. A large, vibrant ecosystem helps developers

discover and use solutions from the community and makes it easy to contribute new ideas and approaches. Given the variety of toolchains in use for application lifecycle management, a healthy ecosystem is also necessary to ensure that every language, Integrated Development Environment (IDE), and enterprise build technology has the runtimes, plugins, and open-source solutions essential to integrate the building and to deployment of serverless applications into existing approaches. Finally, a broad ecosystem provides significant acceleration across domains and enables developers to repurpose existing code more readily in a serverless architecture.

Security

All AWS customers benefit from a data center and network architecture built to satisfy the requirements of our most security-sensitive customers. This means that you get a resilient infrastructure designed for high security without a traditional data center's capital outlay and operational overhead. Serverless architecture is no exception.

To accomplish this, AWS' serverless services offer a broad array of *security and access controls*, including support for virtual private networks, role-based and access-based permissions, robust integration with API-based authentication and access control mechanisms and support for encrypting application elements, such as environment variable settings.

These out-of-the-box offered features and services can help developers deploy and publish workloads confidently and reduce time to market. Serverless systems, by their design, also provides an additional level of security and control for the following reasons:

- **First-class fleet management, including security patching** – For managed serverless services such as Lambda, API Gateway, and Amazon SQS, the servers that host the services are constantly monitored, cycled, and security scanned. As a result, they can be patched within hours of essential security update availability instead of many enterprises' compute fleets with much looser service level agreements (SLAs) for patching and updating.
- **Per-request authentication, access control, and auditing** – Every request between natively-integrated services is individually authenticated, authorized to access specified resources, and can be fully audited. Requests arriving from outside of AWS via Amazon API Gateway provide other internet-facing defense systems. For example, AWS Web Application Firewall (AWS WAF) is a web application firewall that integrates natively with Amazon API Gateway. It helps protect hosted APIs against common web exploits and bots that may affect availability, compromise security, or consume excessive resources, including distributed denial-of-service (DDoS) attack defenses. In addition, companies migrating to serverless architectures can use AWS CloudTrail to

gain detailed insight into which users are accessing which systems with what privileges. Finally, they can use AWS tools to process the audit records programmatically.

These security features of serverless help eliminate additional costs often overlooked when calculating the TCO of one's infrastructure. Such costs include security and monitoring software licenses installed on servers, staffing of information security personnel to ensure that all servers are secure, as well as costs associated with regulatory compliance, and many others.

Serverless architectures also have a smaller blast radius compared to monolithic applications running on virtual machines. As AWS takes responsibility of the security of the servers behind the scenes, customers can focus on implementing least privilege access between the services. Once least privilege access is implemented, the blast radius is dramatically reduced.

The decoupled nature of the architecture will limit the impact to a smaller set of services, compared to a scenario where a malicious actor gains access to an internal server. Considering the significant financial impact of a security breach, this is also an added benefit that help enterprises optimize on infrastructure costs.

Adopting serverless architectures help in reducing or eliminating such expenses that are no longer needed, and capital can be repurposed, and teams are freed to work on higher-value activities.

Partners

AWS has an expansive partner network that assists our customers with building solutions and services on AWS. AWS works closely with validated AWS Lambda Partners for building serverless architectures that help customers develop services and applications without provisioning or managing servers.

Lambda Partners provide developer tooling solutions validated by AWS serverless experts against the AWS Well-Architected Framework. Customers can simplify their technology evaluation process and increase purchasing confidence, knowing these companies' solutions have passed a strict AWS validation of security, performance, and reliability.

Customers can ultimately reduce time to market with the assistance of qualified partners leveraging serverless technologies. For a complete list of AWS Lambda Ready Partners, visit our AWS Partner Network page.

Case studies

Companies have applied serverless architectures to use cases from stock trade validation to e-commerce website construction to natural language processing. AWS serverless portfolio offers the flexibility to create a wide array of applications, including those requiring assurance programs such as PCI or HIPAA compliance.

The following sections illustrate some of the most common use cases but are not a comprehensive list. For a complete list of customer references and use case documentation, see Serverless Computing.

Serverless websites, web Apps, and mobile backends

Serverless approaches are ideal for applications where the load can vary dynamically. Using a serverless approach means no compute costs are incurred when there is no end-user traffic while still offering instant scale to meet high demand, such as a flash sale on an e-commerce site or a social media mention that drives a sudden wave of traffic.

Compared to traditional infrastructure approaches, it is also often significantly less expensive to develop, deliver, and operate a web or mobile backend when architected in a serverless fashion.

AWS provides the services developers need to construct these applications rapidly:

- Amazon Simple Storage Service (Amazon S3) and AWS Amplify offer a simple hosting solution for static content.
- AWS Lambda, in conjunction with Amazon API Gateway, provides support for dynamic API requests using functions.
- Amazon DynamoDB offers a simple storage solution for the session and per-user state.
- Amazon Cognito provides an easy way to handle end-user registration, authentication, and access control to resources.
- Developers can use AWS Serverless Application Model (SAM) to describe the various elements of an application.
- AWS CodeStar can set up a CI/CD toolchain with just a few clicks.

To learn more, see the whitepaper AWS Serverless Multi-Tier Architectures, which provides a detailed examination of patterns for building serverless web applications. For complete reference

architectures, see Serverless Reference Architecture for creating a Web Application and Serverless Reference Architecture for creating a Mobile Backend on GitHub.

Customer example – Neiman Marcus

A luxury household name, Neiman Marcus has a reputation for delivering a first-class, personalized customer service experience. To modernize and enhance that experience, the company wanted to develop Connect, an omnichannel digital selling application that would empower associates to view rich, personalized customer information with the goal of making each customer interaction unforgettable.

Choosing a serverless architecture with mobile development solutions on Amazon Web Services (AWS) enabled the development team to launch the app much faster than in the 4 months it had originally planned. “Using AWS cloud-native and serverless technologies, we increased our speed to market by at least 50 percent and were able to accelerate the launch of Connect,” says Sriram Vaidyanathan, senior director of omni engineering at Neiman Marcus.

This approach also greatly reduced app-building costs and provided developers with more agility for the development and rapid deployment of updates. The app elastically scales to support traffic at any volume for greater cost efficiency, and it has increased associate productivity. For more information, see the Neiman Marcus case study.

IoT backends

The benefits that a serverless architecture brings to web and mobile apps make it easy to construct IoT backends and device-based analytic processing systems that seamlessly scale with the number of devices.

For an example reference architecture, see Serverless Reference Architecture for creating an IoT Backend on GitHub.

Customer example – iRobot

iRobot, which makes robots such as the Roomba cleaning robot, uses AWS Lambda in conjunction with the AWS IoT service to create a serverless backend for its IoT platform. As a popular gift on any holiday, iRobot experiences increased traffic on these days.

While huge traffic spikes could also mean huge headaches for the company and its customers alike, iRobot’s engineering team doesn’t have to worry about managing infrastructure or manually

writing code to handle availability and scaling by running on serverless. This enables them to innovate faster and stay focused on customers. Watch the AWS re:Invent 2020 video Building the next generation of residential robots for more information.

Data processing

The largest serverless applications process massive volumes of data, much of it in real-time. Typical serverless data processing architectures use a combination of Amazon Kinesis and AWS Lambda to process streaming data, or they combine Amazon S3 and AWS Lambda to trigger computation in response to object creation or update events.

When workloads require more complex orchestration than a simple trigger, developers can use AWS Step Functions to create stateful or long-running workflows that invoke one or more Lambda functions as they progress. To learn more about serverless data processing architectures, see the following on GitHub:

- Serverless Reference Architecture for Real-time Stream Processing
- Serverless Reference Architecture for Real-time File Processing
- Image Recognition and Processing Backend reference architecture

Customer example – FINRA

The Financial Industry Regulatory Authority (FINRA) used AWS Lambda to build a serverless data processing solution that enables them to perform half a trillion data validations on 37 billion stock market events daily.

In his talk at AWS re:Invent 2016 entitled The State of Serverless Computing (SVR311), Tim Griesbach, Senior Director at FINRA, said, “We found that Lambda was going to provide us with the best solution for this serverless cloud solution. With Lambda, the system was faster, cheaper, and more scalable. So at the end of the day, we’ve reduced our costs by over 50 percent, and we can track it daily, even hourly.”

Customer example – Toyota Connected

Toyota Connected is a subsidiary of Toyota and a technology company offering connected platforms, big data, mobility services and other automotive-related services.

Toyota Connected chose serverless computing architecture to build its Toyota Mobility Services Platform, leveraging AWS Lambda, Amazon Kinesis Data Streams (Amazon KDS), and Amazon S3 to offer personalized, localized, and predictive data to enhance the driving experience.

With its serverless architecture, Toyota Connected seamlessly scaled to 18 times its usual traffic volume, with 18 billion transactions per month running through the platform, reducing aggregation job times from 15+ hours to 1/40th of the time while reducing operational burden. Additionally, serverless enabled Toyota Connected to deploy the same pipeline in other geographies with smaller volumes and only pay for the resources consumed.

For more information, read our Big Data Blog on Toyota Connected or watch the re:Invent 2020 video Reimagining mobility with Toyota Connected (AUT303).

Big data

AWS Lambda is a perfect match for many high-volume, parallel processing workloads. For an example of a reference architecture using MapReduce, see Reference Architecture for running serverless MapReduce jobs.

Customer example – Fannie Mae

Fannie Mae, a leading source of financing for mortgage lenders, uses AWS Lambda to run an "embarrassingly parallel" workload for its financial modeling. Fannie Mae uses Monte Carlo simulation processes to project future cash flows of mortgages that help manage mortgage risk.

The company found that its existing HPC grids were no longer meeting its growing business needs. So Fannie Mae built its new platform on Lambda, and the system successfully scaled up to 15,000 concurrent function executions during testing. The new system ran one simulation on 20 million mortgages completed in 2 hours, which is three times faster than the old system. Using a serverless architecture, Fannie Mae can run large-scale Monte Carlo simulations effectively because it doesn't pay for idle compute resources. It can also speed up its computations by running multiple Lambda functions concurrently.

Fannie Mae also experienced shorter than typical time-to-market because they were able to dispense with server management and monitoring, along with the ability to eliminate much of the complex code previously required to manage application scaling and reliability. See the Fannie Mae AWS Summit 2017 presentation SMC303: Real-time Data Processing Using AWS Lambda for more information.

IT automation

Serverless approaches eliminate the overhead of managing servers, making most infrastructure tasks, including provisioning, configuration, management, alarms/monitors, and timed cron jobs, easier to create and manage.

Customer example – Autodesk

Autodesk, which makes 3D design and engineering software, uses AWS Lambda to automate its AWS account creation and management processes across its engineering organization.

Autodesk estimates that it realized cost savings of 98 percent (factoring in estimated savings in labor hours spent provisioning accounts). It can now provision accounts in just 10 minutes instead of the 10 hours it took to provision with the previous, infrastructure-based process.

The serverless solution enables Autodesk to automatically provision accounts, configure and enforce standards, and run audits with increased automation and fewer manual touchpoints. For more information, see the Autodesk AWS Summit 2017 presentation SMC301: The State of Serverless Computing. Visit GitHub to see the Autodesk Tailor service.

Machine learning

You can use serverless services to capture, store, and preprocess data before feeding it to your machine learning model. After training the model, you can also serve the model for prediction at scale for inference without providing or managing any infrastructure.

Customer example – Genworth

Genworth Mortgage Insurance Australia Limited is a leading provider of lenders' mortgage insurance in Australia. Genworth has more than 50 years of experience and data in this industry and wanted to use this historical information to train predictive analytics for loss mitigation machine learning models.

To achieve this task, Genworth built a serverless machine learning pipeline at scale using services like AWS Glue, a serverless managed ETL processing service to ingest and transform data, and Amazon SageMaker AI to batch transform jobs and, perform ML inference, and process and publish the results of the analysis.

With the ML models, Genworth could analyze recent repayment patterns for each insurance policy to prioritize them in likelihood and impact for each claim. This process was automated end-to-end

to help the business make data-driven decisions and simplify high-value manual work performed by the Loss Mitigation team. Read the Machine Learning blog How Genworth built a serverless ML pipeline on AWS using Amazon SageMaker AI and AWS Glue for more information.

Conclusion

Serverless approaches are designed to tackle two classic IT management problems: idle servers, and operating fleets of servers that distract and detract from the business of creating differentiated customer value.

AWS serverless offerings solve these longstanding problems with a pay-for-value billing model, and by eliminating the need to manage the underlying infrastructure. AWS constantly scans, patches and monitors the underlying infrastructure making these applications more secure, and provides built-in fault tolerance with minimal configuration needed for high availability. As a result, developers can focus on writing business logic rather than managing infrastructure, allowing enterprises to reduce time to market while paying for only the resources consumed.

Existing companies are gaining significant agility and economic benefits from adopting serverless architectures, and enterprises should consider serverless first strategy for building cloud-native microservices. To learn more and read whitepapers on related topics, see Serverless Computing and Applications.

Contributors

The following individuals and organizations contributed to this document:

- Tim Wagner, General Manager of AWS Serverless Applications, Amazon Web Services
- Paras Jain, Technical Account Manager, Amazon Web Services
- John Lee, Solutions Architect, Amazon Web Services
- Diego Magalhães, Principal Solutions Architect, Amazon Web Services

Further reading

For additional information, see the following:

- Architecture Best Practices for Serverless
- AWS Architecture Center
- AWS Ramp-Up Guide: Serverless

Reference architectures

- Web Applications
- Mobile Backends
- IoT Backends
- File Processing
- Stream Processing
- Image Recognition Processing
- MapReduce

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Content refreshed.	September 15, 2021
Initial publication	Whitepaper first published.	October 2, 2017

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.