Optimización de la economía empresarial con arquitecturas sin servidor

Septiembre de 2017

Avisos

Este documento se suministra únicamente con fines informativos. Representa la oferta actual de productos y prácticas de AWS a partir de la fecha de publicación de este documento. Dichas prácticas y productos pueden modificarse sin previo aviso. Los clientes son responsables de realizar sus propias evaluaciones independientes de la información contenida en este documento y de cualquier uso de los productos o servicios de AWS, cada uno de los cuales se ofrece “tal cual”, sin garantía de ningún tipo, ya sea explícita o implícita. Mediante este documento no se genera ninguna garantía, declaración, compromiso contractual, condición ni certeza por parte de AWS, sus filiales, proveedores o licenciantes. Las responsabilidades y obligaciones de AWS con respecto a sus clientes se controlan mediante los acuerdos de AWS y este documento no forma parte ni modifica ningún acuerdo entre AWS y sus clientes.

Contenido

Resumen

Este documento técnico está destinado a ayudar a los directores de información (CIO), directores ejecutivos de tecnología (CTO) y arquitectos sénior que quieren obtener información sobre las arquitecturas sin servidor y su impacto en el tiempo de comercialización, la agilidad de equipo y la economía de TI. Al eliminar los servidores inactivos e infrautilizados durante el diseño y simplificar drásticamente diseños de software basados en la nube, los enfoques sin servidores están cambiando rápidamente el panorama de TI.

Este documento técnico cubre los aspectos básicos de los enfoques sin servidor y la cartera sin servidor de AWS, e incluye una serie de estudios de casos que ilustran cómo las empresas existentes ya están obteniendo una agilidad significativa y beneficios económicos al adoptar enfoques sin servidores. Este documento ilustra cómo las organizaciones de todos los tamaños pueden usar arquitecturas sin servidor para diseñar sistemas reactivos basados en eventos y entregar rápidamente microservicios nativos en la nube a una fracción de los costos convencionales.

Introducción

Muchas compañías ya se ven los beneficios de la ejecución de aplicaciones en la nube pública, que incluyen los ahorros de costos de la facturación de pago por uso y una mayor agilidad a través del uso de recursos de TI bajo demanda. Múltiples estudios en todos los tipos de aplicaciones y sectores han demostrado que la migración de arquitecturas de aplicaciones existentes a la nube disminuye el costo total de propiedad (TCO) y mejora el tiempo de comercialización.[1]

En relación a las soluciones locales y en la nube privada, la nube pública facilita considerablemente la creación, la implementación y la administración de flotas de servidores y de las aplicaciones que se ejecutan en ellos. Sin embargo, las compañías de hoy en día tienen opciones adicionales para aprovechar la nube pública que van más allá del servidor clásico o de las arquitecturas basadas en VM. Aunque la nube elimina la necesidad de que las compañías compren y mantengan su propio hardware, *cualquier* arquitectura basada en servidor aún requiere que integren en ella la escalabilidad y la fiabilidad. Además, las empresas deben enfrentar los desafíos de aplicar parches e implementarlos en esas flotas de servidores a medida que evolucionan sus aplicaciones. Además, deben escalar sus flotas de servidores para tener en cuenta la carga máxima y luego intentar reducirlas cuando sea posible para reducir los costos, todo mientras protegen la experiencia de los usuarios finales y la integridad de los sistemas internos. Los servidores desocupados e infrautilizados resultan ser costosos y antieconómicos. Los analistas estiman que, en la práctica, el 85% de los servidores tienen capacidad subutilizada.[2]

Los servicios de cómputo sin servidor como AWS Lambda están diseñados para abordar estos desafíos ya que ofrecen a las compañías una forma diferente de abordar el diseño de aplicaciones, con costos inherentemente más bajos y menor tiempo de comercialización. *AWS Lambda elimina la complejidad de manejar servidores en todos los niveles de la pila de tecnología e introduce un modelo de facturación de pago por solicitud en el que no hay más costos por la capacidad de cómputo inactiva.* Además, las funciones de Lambda permiten a las organizaciones adoptar fácilmente arquitecturas de microservicios. Eliminar la infraestructura y pasar a un modelo Lambda ofrece ventajas económicas duales:

- Los problemas como servidores inactivos simplemente dejan de existir, junto con las consecuencias económicas que esto acarrea. Un servicio de informática sin servidor como AWS Lambda nunca es “frío”, porque los cargos solo se acumulan cuando se realiza un trabajo útil, con una granularidad de facturación a nivel de milisegundos.
- Ya no es necesaria la administración de la flota (que incluye parches de seguridad, implementaciones y monitoreo de los servidores). Esto significa que no es necesario mantener las herramientas, los procesos y las rotaciones de guardia asociadas que se requieren para dar soporte ininterrumpidamente al tiempo de actividad de la flota de servidores. Usar Lambda para crear microservicios ayuda a las organizaciones a ser más ágiles. Sin la carga de la administración de servidores, las compañías pueden dirigir sus escasos recursos de TI a lo que importa: su negocio.

Con costos de infraestructura muy reducidos, equipos de trabajo más ágiles y enfocados, y un tiempo de comercialización más rápido, las compañías que ya han adoptado enfoques sin servidores están logrando sacar una ventaja clave sobre sus competidores.

Entender las aplicaciones sin servidor

La ventaja del enfoque sin servidor mencionado anteriormente es atractiva, pero ¿cuáles son las consideraciones para la implementación práctica? ¿Qué separa una aplicación sin servidor de su equivalente convencional basada en servidor?

Las aplicaciones sin servidor están diseñadas de manera que los desarrolladores puedan enfocarse en su competencia principal: escribir la lógica real de negocio. Muchos de los componentes estándar de la aplicación, como los servidores web, y todas las tareas pesadas no diferenciadas, como el software para manejar la fiabilidad y el escalado, están completamente alejados del desarrollador. Lo que queda es un enfoque claro y funcional donde la lógica de negocio se activa solo cuando es necesario: un usuario móvil que envía un mensaje, una imagen cargada en la nube, registros que llegan a un streaming, etc. Un enfoque asincrónico basado en eventos para el diseño de aplicaciones, si bien no es obligatorio, es muy común en las aplicaciones sin servidor, porque encaja perfectamente con el concepto de código que se ejecuta (e incurre en costos) solo cuando hay trabajo por hacer.

Una aplicación sin servidor se ejecuta en la nube pública, en un servicio como AWS Lambda, el cual se encarga de recibir eventos o invocaciones del cliente y luego crea una instancia y ejecuta el código. Este modelo ofrece una serie de ventajas en comparación con el diseño de las aplicaciones convencionales basadas en servidores:

- No es necesario aprovisionar, implementar, actualizar, monitorear ni administrar servidores. Todo el hardware y el software real del servidor lo maneja el proveedor de la nube.
- La aplicación se escala automáticamente cuando se usa realmente. Esto es intrínsecamente diferente de las aplicaciones convencionales, que requieren una flota receptora y una administración explícita de la capacidad para escalar hasta llegar a la carga máxima.
- Además del escalado, se incorpora la disponibilidad y la tolerancia a errores. No se requiere codificación, configuración o administración para obtener el beneficio que ofrecen estas capacidades.
- No hay cargos por capacidad inactiva. No es necesario y, de hecho, no hay posibilidad de preaprovisionar o sobreaprovisionar capacidad. En cambio, la facturación es de pago por solicitud y se basa en lo que dura la ejecución del código.

Casos de uso de aplicaciones sin servidor

El modelo de aplicación sin servidor es genérico y se aplica a casi cualquier tipo de aplicación, desde la aplicación web de una startup hasta la plataforma de análisis bursátiles de una compañía de Fortune 100. Estos son algunos ejemplos:

- **Aplicaciones web y sitios web:** la eliminación de servidores hace posible la creación de aplicaciones web que no cuestan prácticamente nada cuando no hay tráfico, pero simultáneamente se escalan para manejar las cargas máximas, incluso las inesperadas.
- **Backends móviles:** los servidores móviles sin servidor ofrecen una forma para que los desarrolladores que se centran en el desarrollo de clientes creen fácilmente backends seguros, altamente disponibles y perfectamente escalables, sin necesidad de convertirse en expertos en el diseño de sistemas distribuidos.

- **Procesamiento de medios y registros:** los enfoques sin servidor ofrecen un paralelismo natural, simplificando el procesamiento de cargas de trabajo con uso intensivo de cómputos, sin la complejidad de tener que crear sistemas con procesos múltiples o escalar manualmente las flotas.
- **Automatización de TI:** las funciones sin servidor se pueden asociar a alarmas y a monitores para ofrecer personalización cuando sea necesario. La implementación de los trabajos de Cron y otros requisitos de infraestructura de TI se simplifican considerablemente al eliminar el requisito de poseer y mantener servidores para su uso, especialmente cuando estos trabajos y requisitos son poco frecuentes o de naturaleza variable.
- **Backends de IoT:** la capacidad de incorporar cualquier código, incluso las bibliotecas nativas, simplifica el proceso de creación de sistemas basados en la nube que pueden implementar algoritmos específicos de dispositivos.
- **Chatbots (incluso asistentes habilitados para voz) y otros sistemas basados en webhook:** los enfoques sin servidor son perfectos para cualquier sistema basado en webhook, como un chatbot. La capacidad que tienen para realizar acciones (como ejecutar código) solo cuando es necesario (como cuando un usuario solicita información de un chatbot) los convierte en un enfoque directo y de bajo costo para estas arquitecturas. Por ejemplo, la mayoría de las habilidades de Alexa para Amazon Echo se implementan utilizando AWS Lambda.
- **Secuencias de clics y otros procesos de datos de streaming casi en tiempo real:** las soluciones sin servidor ofrecen la flexibilidad para escalar tanto horizontal como verticalmente con el flujo de datos, lo que les permite equiparar los requisitos de rendimiento sin la complejidad de crear un sistema informático escalable para cada aplicación. Cuando se combina con una tecnología como Amazon Kinesis, AWS Lambda puede ofrecer procesamiento de alta velocidad de registros para análisis de secuencias de clics, disparadores de datos NoSQL, información comercial de acciones y más.

Además de los casos de uso altamente adoptados que analizamos anteriormente, las compañías también están aplicando enfoques sin servidor para los siguientes dominios:

- Big data, como problemas de reducción de mapas, transcodificación de video de alta velocidad, análisis bursátiles y simulaciones Monte Carlo de uso intensivo de cómputos para aplicaciones de préstamos. Los desarrolladores han descubierto que es mucho más fácil paralelizar con un enfoque sin servidor,[3] especialmente cuando se activa a través de eventos, lo que los lleva a aplicar cada vez más técnicas sin servidores a una amplia gama de problemas de big data sin la necesidad de administración de infraestructura.
- Procesamiento personalizado y de baja latencia para aplicaciones web y activos entregados a través de redes de entrega de contenido. Al trasladar la entrega de eventos sin servidor al borde de Internet, los desarrolladores pueden aprovechar la latencia más baja y la capacidad de personalizar las recuperaciones y el contenido fácilmente. Esto permite un nuevo espectro de casos de uso que están optimizados para la latencia en función de la ubicación del cliente.
- Los dispositivos conectados que habilitan las funciones sin servidor, como las funciones de AWS Lambda, se ejecutan en dispositivos comerciales, residenciales y del Internet de las cosas (IoT) portátiles. Las soluciones sin servidor, como las funciones de Lambda, ofrecen una abstracción natural del hardware físico subyacente (e incluso virtual), permitiéndoles una transición más fácil del centro de datos al borde, y de una arquitectura de hardware a otra, sin interrumpir el modelo de programación.
- Lógica personalizada y manejo de datos en dispositivos locales como AWS Snowball Edge. Debido a que desasocia la lógica de negocio de los detalles del entorno de ejecución, las aplicaciones sin servidor pueden funcionar fácilmente en una amplia variedad de entornos, incluso en un dispositivo.

Normalmente, las aplicaciones sin servidor se crean utilizando una *arquitectura de microservicios* en la que una aplicación se separa en componentes independientes que realizan trabajos discretos. Estos componentes, que constan de funciones individuales de Lambda junto con API, colas de mensajes, base de datos y otros componentes, se pueden implementar, probar y escalar de forma independiente. De hecho, las aplicaciones sin servidor son una opción natural para los microservicios debido a su modelo basado en funciones. Al evitar arquitecturas y diseños monolíticos, las organizaciones tienen la posibilidad de ser más ágiles porque los desarrolladores pueden implementar de forma incremental y reemplazar o actualizar componentes individuales, como la capa de la base de datos, si es necesario.

En muchos casos, basta con aislar la lógica de negocio de una aplicación para convertirla en una aplicación sin servidor. Los servicios como AWS Lambda admiten lenguajes de programación conocidos y permiten el uso de bibliotecas personalizadas. Las tareas de larga duración se expresan como flujos de trabajo compuestos por funciones individuales que operan dentro de marcos de tiempo razonables, lo que permite que el sistema reinicie o paralelice unidades de cómputo individuales, según sea necesario.

¿La opción sin servidor es adecuada siempre?

Casi todas las aplicaciones modernas se pueden modificar para su correcta ejecución y, en la mayoría de los casos, de una manera más económica y escalable, en una plataforma sin servidor. Sin embargo, hay algunas ocasiones en que la opción sin servidor no es la mejor:

- Cuando el objetivo es explícitamente evitar realizar cambios en una aplicación.
- Cuando se requiere un control preciso del entorno, como la especificación de parches específicos del sistema operativo o el acceso a operaciones de red de bajo nivel, para que el código se ejecute correctamente.
- Cuando una aplicación local no se ha migrado a la nube pública.

Evaluar la plataforma sin servidor de un proveedor de la nube

Al diseñar una aplicación sin servidor, las empresas y las organizaciones deben considerar algo más que la funcionalidad informática sin servidor que ejecuta el código de la aplicación. Las aplicaciones sin servidor completas requieren una amplia gama de servicios, herramientas y capacidades que abarcan almacenamiento, mensajería, diagnósticos y más. Una cartera sin servidor, incompleta o fragmentada de un proveedor de la nube puede ser problemática para los desarrolladores sin servidor, que podrían tener que volver a las arquitecturas basadas en servidor si no pueden codificar correctamente a un nivel constante de abstracción.

Una plataforma sin servidor consta de un conjunto de servicios que comprende la aplicación sin servidor, como los componentes informáticos y de almacenamiento, así como las herramientas necesarias para crear, desarrollar, implementar y diagnosticar aplicaciones sin servidor. Ejecutar una aplicación sin servidor en producción requiere una plataforma de confianza, flexible y confiable que pueda manejar las demandas de pequeñas startups hasta las de corporaciones globales a nivel mundial. La plataforma debe escalar *todos* los elementos de una aplicación y proporcionar fiabilidad de un extremo a otro. Al igual que con las aplicaciones convencionales, ayudar a los desarrolladores a tener éxito en la creación y entrega de soluciones sin servidor es un desafío multidimensional. Para satisfacer las necesidades de las compañías a gran escala en diferentes sectores, una plataforma sin servidor debe ofrecer las siguientes capacidades:

Figura 1: Capacidades de una plataforma sin servidor

- Una *capa lógica en la nube* de alto rendimiento, escalable y de confianza.
- *Fuentes de datos y eventos* propios con capacidad de respuesta y *conectividad simple a sistemas de terceros*.
- *Bibliotecas de integraciones* que permitan a los desarrolladores comenzar fácilmente y agregar nuevos patrones de forma rápida y segura a las soluciones existentes.

- Un *ecosistema de desarrolladores* activo que ayude a los desarrolladores a descubrir y aplicar soluciones en una variedad de dominios y para un amplio conjunto de sistemas y casos de uso de terceros.
- Un conjunto de *marcos de modelado de aplicaciones* adecuados.
- *Organización* que ofrezca administración de estados y de flujos de trabajo.
- *Escala global* y alcance amplio que incluya la certificación del programa de aseguramiento.
- *Fiabilidad incorporada y rendimiento a escala*, sin la necesidad de aprovisionar capacidad a ningún nivel de la escala.
- *Seguridad incorporada* junto con *control de acceso flexible* para recursos y servicios propios y de terceros.

En el núcleo de cualquier plataforma sin servidor se encuentra la *capa de lógica de la nube* responsable de ejecutar las funciones que representan la lógica de negocio. Debido a que estas funciones a menudo se ejecutan en respuesta a eventos, la *integración simple con fuentes de eventos propias y de terceros* es esencial para hacer que las soluciones sean simples de expresar y permitirles escalar automáticamente en respuesta a cargas de trabajo variables. Por ejemplo, las funciones sin servidor pueden necesitar ejecutarse cada vez que se crea un objeto en un almacenamiento de objetos o durante cada actualización realizada en una base de datos NoSQL sin servidor. Las arquitecturas sin servidor eliminan todo el código del escalado y de la administración que normalmente se requiere para integrar dichos sistemas, trasladando esa carga operativa al proveedor de la nube.

Desarrollar con éxito en una plataforma sin servidor requiere que una compañía pueda comenzar fácilmente, incluso que pueda buscar plantillas listas para casos de uso común, ya sea que involucren servicios propios o de terceros. Estas *bibliotecas de integraciones* son esenciales para transmitir patrones exitosos, como el procesamiento de flujos de registros o la implementación de webhooks, especialmente durante el período en que los desarrolladores migran desde arquitecturas basadas en servidores a arquitecturas sin servidores. Una necesidad estrechamente relacionada es un *ecosistema amplio y diverso* que rodea la plataforma central. Un ecosistema grande y dinámico ayuda a los desarrolladores a descubrir y usar soluciones de la comunidad y facilita la contribución de nuevas ideas y enfoques. Dada la variedad de cadenas de herramientas en uso para la administración del ciclo de vida de las aplicaciones, también es necesario un ecosistema en buen estado para garantizar que cada lenguaje, entorno de

desarrollo integrado (IDE) y tecnología de compilación empresarial tenga los tiempos de ejecución, los complementos y las soluciones de código abierto necesarios para integrar la creación y la implementación de aplicaciones sin servidor en los enfoques existentes. También es fundamental que las aplicaciones sin servidor aprovechen las inversiones existentes, incluso el conocimiento de los desarrolladores de marcos como Express y Flask y lenguajes de programación conocidos. Un amplio ecosistema proporciona una aceleración importante entre los dominios y permite a los desarrolladores reutilizar el código existente más fácilmente en una arquitectura sin servidor.

Los *marcos de modelado de aplicaciones*, como AWS Serverless Application Model (AWS SAM) de especificación abierta, permiten a los desarrolladores expresar los componentes que conforman una aplicación sin servidor y habilitar las herramientas y los flujos de trabajo necesarios para crear, implementar y monitorear esas aplicaciones. Otro marco que es clave para el éxito de una plataforma sin servidor es la *organización y la administración del estado*. La naturaleza en gran medida sin estado de la informática sin servidor requiere un mecanismo complementario para habilitar flujos de trabajo de larga ejecución. Las soluciones de organización permiten a los desarrolladores coordinar los múltiples componentes relacionados de las aplicaciones que son típicos en una aplicación sin servidor, al mismo tiempo que permiten que esas aplicaciones se compongan de funciones pequeñas y de corta duración. Los servicios de organización también simplifican el manejo de errores y permiten la integración con sistemas heredados y flujos de trabajo, que incluyen a aquellos que se ejecutan durante más tiempo de lo permitido por las funciones sin servidor en general.

Para brindar soporte a clientes en todo el mundo, incluso a las corporaciones multinacionales con alcance global, una plataforma sin servidor debe ofrecer una *escala global* que incluya los centros de datos y las ubicaciones de borde en todo el mundo. Las ubicaciones de borde son clave para acercar la informática sin servidor de baja latencia a los usuarios finales. Debido a que es la plataforma, en lugar del desarrollador de la aplicación, la responsable de proporcionar la escalabilidad y la alta disponibilidad de las aplicaciones sin servidor, su *fiabilidad* intrínseca es fundamental. Las características como los reintentos incorporados y las colas de mensajes fallidos para eventos no procesados ayudan a los desarrolladores a crear sistemas sólidos con fiabilidad de extremo a extremo utilizando enfoques sin servidores. El rendimiento es igualmente clave, especialmente la baja latencia (carga adicional), dado que se crean instancias bajo demanda en una aplicación sin servidor de los tiempos de ejecución del lenguaje y del código del cliente.

Finalmente, la plataforma debe tener una amplia gama de *controles de seguridad y acceso*, que incluyen soporte para redes privadas virtuales, permisos basados en roles y acceso, integración sólida con autenticación basada en API y mecanismos de control de acceso (que incluyen sistemas heredados y de terceros) y soporte para cifrar elementos de la aplicación, como la configuración de variables del entorno. Los sistemas sin servidor, por su diseño, ofrecen un nivel inherentemente más alto de seguridad y control por los siguientes motivos:

- **Administración de flotas de primera clase, que incluye parches de seguridad:** en un sistema como AWS Lambda, los servidores que ejecutan solicitudes se supervisan, se controlan en ciclos y se escanean por cuestiones de seguridad constantemente. Se pueden aplicar parches pocas horas después de la disponibilidad de las actualizaciones de seguridad clave, a diferencia de muchas flotas informáticas empresariales que pueden tener acuerdos de nivel de servicio mucho más flexibles para los parches y las actualizaciones.
- **Servidores con vida útil limitada:** cada máquina que ejecuta el código del cliente en AWS Lambda se controla en ciclos varias veces al día, lo que limita su exposición a los ataques y garantiza un sistema operativo y parches de seguridad constantemente actualizados.
- **Autenticación por solicitud, control de acceso y auditoría:** cada solicitud de cómputo ejecutada en AWS Lambda, independientemente de su origen, se autentica individualmente, se autoriza para acceder a los recursos especificados y se audita por completo. Las solicitudes que llegan desde fuera de los centros de datos de AWS a través de Amazon API Gateway proporcionan sistemas de defensa adicionales orientados a Internet, que incluye las defensas de ataques de DoS. Las empresas que migran a arquitecturas sin servidor pueden usar AWS CloudTrail para obtener información detallada sobre qué usuarios están accediendo a qué sistemas y con qué privilegios, y pueden usar AWS Lambda para procesar los registros de auditoría mediante programación.

La plataforma AWS sin servidor

Desde la presentación de Lambda en 2014, AWS ha creado una plataforma sin servidor completa. Tiene un amplio conjunto de servicios completamente administrados que permiten a las organizaciones crear aplicaciones sin servidor que se pueden integrar sin problemas con otros servicios de AWS y de terceros. La Figura ilustra un subconjunto de los componentes en la plataforma AWS sin servidor y sus relaciones.

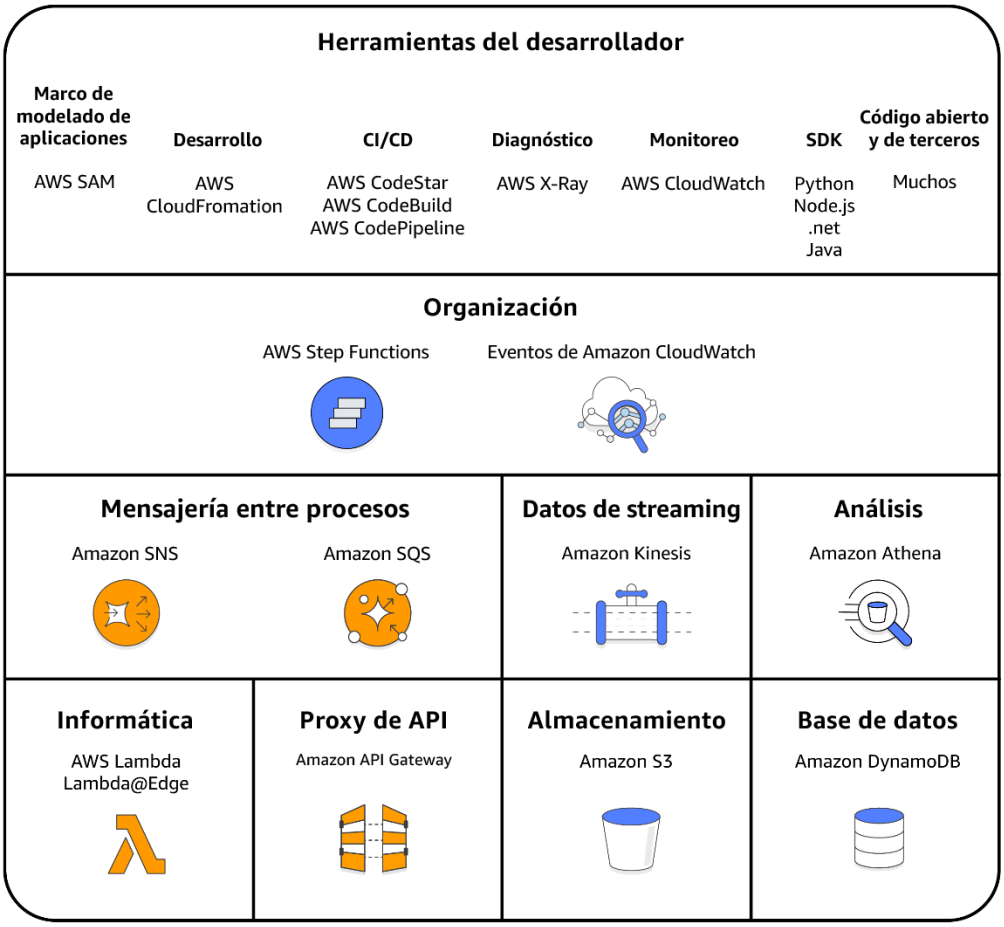

Figura 2: Componentes de la plataforma sin servidor de AWS

Capacidades de la plataforma sin servidor de AWS

AWS proporciona todas las capacidades básicas que se identificaron en la sección anterior como requisitos para una plataforma sin servidor completa. AWS Lambda proporciona la capa lógica de la nube, una oferta de informática sin servidor a gran escala y sin aprovisionamiento basada en funciones. AWS Lambda se complementa con AWS Lambda@Edge, que brinda compatibilidad similar para ejecutar funciones Lambda de latencia extremadamente baja utilizando enrutamiento optimizado en el borde y AWS Greengrass, que permite que las funciones Lambda se ejecuten en dispositivos conectados, incluso dispositivos como AWS Snowball.

Las funciones de Lambda se pueden activar fácilmente mediante una variedad de eventos propios y de terceros, lo que permite a los desarrolladores crear sistemas reactivos basados en eventos (ver Figura 3) sin la molestia habitual de configurar y administrar la infraestructura. Cuando hay múltiples eventos simultáneos, Lambda simplemente ejecuta más copias de la función en paralelo, en respuesta a cada disparador individual. Las funciones de Lambda se escalan con precisión según el tamaño de la carga de trabajo, hasta la solicitud individual. Como resultado, no hay posibilidad de un servidor o contenedor inactivo. El problema de los gastos en infraestructura desperdiciados se elimina *mediante el diseño* en arquitecturas que usan funciones de Lambda.

FaaS, o *Función como servicio*, es un enfoque para crear sistemas de cómputo basados en eventos que se basan en funciones como la unidad de implementación y ejecución. La *FaaS sin servidor* es un tipo de FaaS donde no hay máquinas virtuales ni contenedores presentes en el modelo de programación y donde el proveedor ofrece escalabilidad sin aprovisionamiento y fiabilidad incorporada.

Figura 3: La relación entre la informática basada en eventos, FaaS y la informática sin servidor.

Las capacidades informáticas sin servidor de AWS que proporciona Lambda son un elemento clave de los siguientes servicios administrados que ofrece AWS, y todos se integran perfectamente entre sí:

- Amazon API Gateway: puntos de enlace de HTTP para las funciones de Lambda, que incluyen una gama completa de funciones de administración API y proxy de API.
- Amazon S3: las funciones de Lambda se pueden utilizar como disparadores automáticos de eventos cuando se crea, copia o elimina un objeto.
- Amazon DynamoDB: las funciones de Lambda se pueden usar para procesar cualquiera o todos los cambios realizados en la tabla de una base de datos.
- Amazon SNS: los mensajes se pueden enrutar a las funciones de Lambda para su procesamiento, lo que agrega la capacidad de responder dinámicamente al contenido publicado.
- Amazon SQS: las funciones de Lambda permiten procesar fácilmente los mensajes en las colas.
- Amazon Kinesis Streams: las funciones de Lambda proporcionan el procesamiento de registros en orden del streaming de datos, lo que facilita la creación de motores de análisis casi en tiempo real.
- Amazon Kinesis Firehose: las funciones de Lambda se pueden aplicar automáticamente a los registros que procesa un Firehose, lo que facilita agregar capacidades de transformación, filtrado y análisis a un streaming de datos.
- Amazon Athena: las funciones de Lambda pueden activarse automáticamente para cada objeto en el conjunto de resultados de una consulta.
- AWS Step Functions: se pueden organizar múltiples funciones de Lambda para crear flujos de trabajo de larga ejecución tanto para procesos centrados en el ser humano como automatizados.
- Amazon CloudWatch Events: las funciones de Lambda se pueden usar para responder automáticamente a eventos, incluso de terceros.
- Amazon Aurora: los disparadores de la base de datos se pueden escribir como funciones de Lambda.

Lambda proporciona una biblioteca de integraciones con planos para una amplia variedad de servicios de terceros, Slack, Algorithmia, Twilio, Loggly, Splunk, SumoLogic y Box entre otros, lo que permite a los desarrolladores crear aplicaciones con capacidad de respuesta que incluyen análisis, algoritmos avanzados, comunicaciones, y más con solo unas pocas líneas de código. Se ha mejorado una amplia variedad de marcos de aplicaciones web, entre ellos Express (para aplicaciones NodeJS) y Flask (para aplicaciones de Python) con el fin de que operen bien con las funciones de Lambda. Los proyectos de aplicaciones web de código abierto incluyen Framework Serverless, Sparta, Chalice y muchos otros.

Las aplicaciones sin servidor suelen estar compuestas por varias piezas: una o más funciones, una base de datos sin servidor como Amazon DynamoDB y una API para que los clientes puedan llamar o una fuente de eventos que activa la aplicación. Para mantener estas piezas organizadas, AWS utiliza SAM, el modelo de aplicación sin servidor de especificación abierta. Con SAM, los desarrolladores pueden describir fácilmente las funciones, API, fuentes de eventos, tablas de bases de datos y otras partes de una aplicación sin servidor. Usar SAM también ayuda a los desarrolladores a administrar todos los pasos en el ciclo de vida del desarrollo de un software y AWS ofrece una variedad de herramientas y servicios para ayudarlo. Esto incluye soporte nativo para depuración y pruebas locales en el IDE que elija (o a través de la línea de comando) a través de SAM local, implementación de aplicaciones SAM usando AWS CloudFormation, soporte para crear aplicaciones SAM en AWS CodeBuild, y soporte para la integración y entrega continuas (CI/CD) basadas en GitHub para Aplicaciones SAM integradas en AWS CodePipeline. Además del soporte propio, una serie de marcos de código abierto, los proveedores de CI/CD y de administración del rendimiento ofrecen soporte para funciones SAM y Lambda, incluso Serverless Framework, Claudia, CloudBees, Datadog y muchos más. Para ver ejemplos adicionales, consulte el artículo Herramientas para desarrolladores de aplicaciones sin servidor.[4]

Después de crear una función Lambda (o en el caso de una aplicación SAM, posiblemente varias funciones de Lambda funcionen juntas), los desarrolladores pueden monitorearla fácilmente utilizando las métricas y los registros creados automáticamente disponibles en Amazon CloudWatch y CloudWatch Logs. AWS también ofrece AWS X-Ray, una solución de análisis de desempeño y seguimiento de solicitudes entre servicios que permite a los desarrolladores hacer seguimiento de la operación y el comportamiento de las funciones individuales y los eventos que procesan.

Las ofertas de plataforma sin servidor de AWS tienen un alcance global, con soporte para AWS Lambda y Amazon API Gateway en prácticamente todas las regiones de AWS en el mundo. Lambda@Edge está disponible en todas las ubicaciones de borde.[5] Lambda ofrece una variedad de características que ayudan a los clientes a mejorar la fiabilidad de sus aplicaciones, que incluyen los reintentos automáticos para eventos ordenados y asincrónicos, y las colas de mensajes no entregados para capturar eventos que no fueron procesados con éxito por la aplicación. La profunda integración con Amazon Virtual Private Cloud (Amazon VPC) y el rango flexible de capacidades de autenticación y control de acceso que proporciona AWS Lambda permiten a las organizaciones crear aplicaciones seguras que cumplen con las mejores prácticas, como el principio de privilegio mínimo. La administración y la seguridad del usuario final es igual de fácil: Amazon Cognito ofrece autorización y autenticación que se pueden combinar fácilmente con Amazon API Gateway y AWS Lambda, permitiendo el registro de usuarios sin servidor y las capacidades de inicio de sesión, incluso la integración con proveedores de redes sociales como Facebook y directorios corporativos.

Casos prácticos

Las compañías han aplicado arquitecturas sin servidor a casos de uso que van desde la validación del comercio bursátil hasta la creación de sitios web de e-commerce y el procesamiento del lenguaje natural. AWS Lambda y el resto de la cartera sin servidor de AWS ofrecen la flexibilidad para crear una amplia gama de aplicaciones, incluso aquellas que requieren programas de aseguramiento como la conformidad con PCI o HIPAA. Las siguientes secciones ilustran algunos de los casos de uso más comunes, pero no son una lista completa. Para obtener una lista completa de referencias de clientes y documentación de casos de uso, consulte Informática sin servidor.[6]

Backends móviles, sitios y aplicaciones web sin servidores

Los enfoques sin servidor son ideales para aplicaciones donde la carga puede variar dinámicamente. El uso de un enfoque sin servidor significa que no se incurre en costos de cómputos cuando no hay tráfico del usuario final, pero se ofrece una escala instantánea para satisfacer la alta demanda, como una venta flash en un sitio de e-commerce o una mención en las redes sociales que impulsa una ola repentina de tráfico. En comparación con los enfoques de infraestructura tradicionales, a menudo es mucho menos costoso desarrollar, entregar y operar un backend web o móvil cuando se ha diseñado sin servidores.

AWS proporciona los servicios que los desarrolladores necesitan para crear rápidamente estas aplicaciones:

- Amazon S3 ofrece una solución de alojamiento simple para contenido estático.
- AWS Lambda, junto con Amazon API Gateway, proporciona soporte para solicitudes de API dinámicas mediante el uso de funciones.
- Amazon DynamoDB ofrece una solución de almacenamiento simple para el estado por usuario y de sesión.
- Amazon Cognito proporciona una forma fácil de administrar el registro, la autenticación y el control de acceso del usuario final a los recursos.
- Los desarrolladores pueden utilizar AWS SAM para describir los diversos elementos de una aplicación.
- AWS CodeStar puede configurar una cadena de herramientas de CI/CD con solo unos pocos clics.

Para obtener más información, consulte el documento técnico Arquitecturas multinivel sin servidor de AWS, que proporciona un examen detallado de los patrones para crear aplicaciones web sin servidor.[7] Para ver las arquitecturas de referencia completas, consulte los artículos Serverless Reference Architecture for creating a Web Application[8] y Serverless Reference Architecture for creating a Mobile Backend[9] en GitHub.

Ejemplo de clientes: Bustle.com

Bustle.com es un sitio web de noticias, entretenimiento, estilo de vida y moda para mujeres. Logró aproximadamente un 84% de ahorro de costos al trasladarse a una arquitectura sin servidor basada en AWS Lambda y Amazon API Gateway. Los ingenieros de Bustle obtuvieron más agilidad, lo que les permitió concentrarse en la creación de nuevas características de productos en lugar de ocuparse de la administración y el escalado de la infraestructura. El equipo de Bustle ahora es más eficiente y usa la mitad de las personas que normalmente se requieren para crear y operar sitios de la escala de Bustle. El backend sin servidor de Bustle también es compatible con las aplicaciones de iOS para dos de sus propiedades web (Bustle y Romper). Para obtener más información, consulte el artículo Bustle case study.[10]

Backends para IoT

Los beneficios que aporta llevar una arquitectura sin servidor a la web y a las aplicaciones móviles también facilitan la creación de backends de IoT y sistemas de procesamiento analítico basados en dispositivos que se adaptan perfectamente a la cantidad de dispositivos. Para ver una arquitectura de referencia de ejemplo, consulte el artículo Serverless Reference Architecture for creating an IoT Backend en GitHub.[11]

Ejemplo de clientes: iRobot

iRobot, que fabrica robots como el de limpieza Roomba, utiliza AWS Lambda junto con el servicio AWS IoT para crear un backend sin servidor para su plataforma IoT. Al utilizar una arquitectura sin servidor, el equipo de ingeniería de iRobot no tiene que preocuparse por la administración de la infraestructura o la escritura manual del código para manejar la disponibilidad y la escala. Esto les permite innovar más rápido y mantenerse enfocados en los clientes. Para obtener más información, consulte las diapositivas de la presentación de AWS en re:Invent 2016 Serverless IoT Back Ends (IOT401)[12] o vea el video.[13]

Procesamiento de datos

Las aplicaciones sin servidor más grandes procesan grandes volúmenes de datos, gran parte en tiempo real. Las arquitecturas típicas de procesamiento de datos sin servidor utilizan una combinación de Amazon Kinesis y AWS Lambda para procesar datos de streaming o combinan Amazon S3 y AWS Lambda para activar el cálculo en respuesta a la creación de objetos o eventos de actualización. Cuando las cargas de trabajo requieren una organización más compleja que un disparador simple, los desarrolladores pueden usar AWS Step Functions para crear flujos de trabajo con estados o de larga ejecución y así invocar una o más funciones de Lambda a medida que avanzan. Para obtener más información sobre las arquitecturas de procesamiento de datos sin servidor, consulte el siguiente artículo en GitHub:

- Serverless Reference Architecture for Real-time Stream Processing[14]
- Serverless Reference Architecture for Real-time File Processing[15]
- Image Recognition and Processing Backend reference architecture[16]

Ejemplo de clientes: FINRA

La Autoridad Reguladora de la Industria Financiera (FINRA) utilizó AWS Lambda para crear una solución de procesamiento de datos sin servidor que les permite realizar medio billón de validaciones de datos en 37 mil millones de eventos bursátiles diariamente. En su charla en AWS re:Invent 2016 titulada The State of Serverless Computing (SVR311),[17] Tim Griesbach, director sénior de FINRA, dijo: "Descubrimos que Lambda nos iba a brindar lo mejor para esta solución de nube sin servidor. Con Lambda, el sistema era más rápido, más barato y más escalable. Entonces, al final del día, hemos reducido nuestros costos en más de un 50%...y podemos hacer un seguimiento a diario, incluso por hora".

Ejemplo de cliente: Thomson Reuters

Thomson Reuters, una empresa de medios de comunicación e información, creó una solución de análisis empresarial sin servidor que permite a sus equipos de productos analizar fácilmente los datos de uso del producto. La solución combina AWS Lambda, Amazon Kinesis Streams y Amazon Kinesis Firehose para recopilar y procesar datos de eventos de streaming para su análisis. El resultado, llamado Product Insight, se lanzó dos meses antes de lo previsto y superó las expectativas técnicas.

Anders Fritz, gerente sénior de innovación de productos en Thomson Reuters, dijo: "Nuestro objetivo inicial era tener capacidad para procesar 2000 eventos por segundo. Nuestras pruebas muestran que Product Insight en AWS puede procesar hasta 4000 eventos por segundo, y en un año esperamos aumentar a más de 10 000 eventos por segundo". Esta cifra representa más de 25 mil millones de eventos por mes. Incluso con este alto rendimiento, el sistema no ha perdido ningún dato desde su concepción. "Debido a la sólida arquitectura de conmutación por error y las capacidades técnicas de AWS, no hemos perdido un solo evento desde que comenzamos a recopilar datos", dice Fritz. Para obtener más información, consulte el artículo Thomson Reuters Case Study[18] o vea la presentación de AWS re:Invent 2016 Real-time Data Processing Using AWS Lambda (SVR301).[19]

Big data

AWS Lambda es una combinación perfecta para muchas cargas de trabajo de procesamiento paralelas de gran volumen. Para obtener un ejemplo de una arquitectura de referencia que usa Mapreduce, consulte el artículo Reference architecture for running serverless MapReduce jobs.[20]

Ejemplo de cliente: Fannie Mae

Fannie Mae, una fuente líder de financiamiento para prestamistas hipotecarios, utiliza AWS Lambda para ejecutar una carga de trabajo "vergonzosamente paralela" para su modelado financiero. Fannie Mae utiliza los procesos de simulación de Monte Carlo para proyectar flujos de efectivo a futuro de hipotecas que lo ayudan a administrar el riesgo hipotecario. La compañía descubrió que sus redes HPC ya no satisfacían sus crecientes necesidades empresariales. Fannie Mae creó su nueva plataforma en Lambda y el sistema logró escalar hasta 15 000 ejecuciones simultáneas de funciones durante las pruebas. El nuevo sistema ejecutó una simulación sobre 20 millones de hipotecas que se completó en 2 horas, lo cual es tres veces más rápido que el sistema anterior. Con una arquitectura sin servidor, Fannie Mae ahora puede ejecutar simulaciones de Monte Carlo a gran escala de manera rentable porque no paga los recursos de cómputo inactivos. También puede acelerar sus cómputos ejecutando múltiples funciones Lambda al mismo tiempo. Fannie Mae también experimentó un tiempo de comercialización más corto que el habitual debido a que pudieron prescindir de la administración y del monitoreo del servidor, junto con la capacidad de eliminar gran parte del complejo código que antes se requería para administrar el escalado y la fiabilidad de las aplicaciones. Para obtener más información, consulte la presentación de Fannie Mae en el AWS Summit 2017 SMC303: Real-time Data Processing Using AWS Lambda.[21]

Automatización de TI

Los enfoques sin servidor eliminan la sobrecarga de la administración de los servidores, lo que hace que la mayoría de las tareas de infraestructura, que incluyen el aprovisionamiento, la configuración, la administración, las alarmas o los monitores y los trabajos cronometrados, sean mucho más fáciles de crear y administrar.

Ejemplo de clientes: Autodesk

Autodesk, que fabrica el software de diseño e ingeniería 3D, utiliza AWS Lambda para automatizar los procesos de creación y administración de cuentas de AWS en toda su organización de ingeniería. Autodesk estima que realizó un ahorro de costos del 98% (teniendo en cuenta los ahorros estimados en horas de trabajo empleadas en el aprovisionamiento de cuentas). Ahora puede aprovisionar cuentas en solo 10 minutos en lugar de las 10 horas que tardó en aprovisionarse con el proceso anterior basado en infraestructura. La solución sin servidor permite a Autodesk aprovisionar automáticamente cuentas, configurar y aplicar estándares, y ejecutar auditorías con mayor automatización y menos puntos de contacto

manuales. Para obtener más información, consulte la presentación del AWS Summit 2017 SMC301: The State of Serverless Computing.[22] Visite GitHub para ver el servicio de Autodesk Tailor.

Casos de uso adicionales

Los casos de uso descritos en la sección anterior solo rozan la superficie de las posibilidades que implica trabajar con Lambda y las otras ofertas sin servidor de AWS. Otros casos de uso incluyen una fuerte comprensión del lenguaje humano a través de chatbots creados con Amazon Lex y AWS Lambda, Edge Computing global de baja latencia utilizando Lambda@Edge con Amazon CloudFront y un poderoso procesamiento local de archivos con funciones Lambda dentro de AWS Snowball. Estas son solo algunas de las magníficas capacidades de este enfoque versátil. Para obtener más información, consulte AWS Lambda.[23]

Conclusión

Los enfoques sin servidor están diseñados para abordar dos problemas clásicos de la administración de TI: servidores inactivos que agotan el balance de una compañía sin ofrecer valor y el costo de crear y operar flotas de servidores y software de servidor que distraen y desvirtúan el negocio de crear valor diferenciado para los clientes. AWS Lambda y las otras ofertas sin servidor de AWS resuelven estos problemas antiguos eliminando los servidores, los contenedores, los discos y otros recursos a nivel de la infraestructura del modelo de programación y facturación. Como resultado, los desarrolladores pueden trabajar con un modelo de aplicaciones simple que les ayuda a entregar servicios más rápido y las organizaciones solo pagan por el trabajo útil. La forma más fácil y rápida de diseñar sistemas reactivos basados en eventos y entregar microservicios nativos en la nube es mediante el uso de arquitecturas sin servidor. Para obtener más información y leer documentos técnicos sobre temas relacionados, consulte el artículo Aplicaciones y capacidad de computación sin servidor.[24]

Colaboradores

Las siguientes personas y organizaciones hicieron contribuciones a este documento:

- Tim Wagner, gerente general de aplicaciones sin servidor de AWS, Amazon Web Services

Lectura adicional

Para obtener información adicional, consulte:

- Serverless Reference Architectures with AWS Lambda de Werner Vogels, director de tecnología en Amazon.com
- AWS re:Invent 2016: The State of Serverless Computing [presentación] de Tim Wagner, gerente general de aplicaciones sin servidor de AWS
- The economics of serverless cloud computing de Owen Rogers, director de investigaciones en 451 Research

Arquitecturas de referencia

- Aplicaciones web
- Backends móviles
- Backends para IoT
- Procesamiento de archivos
- Procesamiento de streaming
- Procesamiento de reconocimiento de imagen
- MapReduce

Revisiones del documento

Fecha	Descripción
Septiembre de 2017	Primera publicación

Notes

[1] https://www.forbes.com/sites/moorinsights/2016/04/11/tco-analysis-demonstrates-how-moving-to-the-cloud-can-save-your-company-money/#537e2bd07c4e
http://www.cloudstrategymag.com/articles/86033-understanding-tco-cloud-economics

[2] En 2012, Gartner hizo una estimación de la utilización del centro de datos que iba del 7 al 12% (http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html). Un estudio de McKinsey de 2008 colocó este dato en el 6% (https://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf). Un documento de Accenture que analiza un conjunto de aplicaciones basadas en EC2 encontró aproximadamente un 7% de utilización (http://ieeexplore.ieee.org/document/6118751/). Un estudio de 2014 de NRDC y Anthesis descubrió que, en 2013, más del 30% de los servidores estaban completamente "comatosos" (enchufados, pero sin agregar nada de valor) (http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf).

[3] Occupy the Cloud: Eric Jonas et al., *Distributed Computing for the 99%*, https://arxiv.org/abs/1702.04024.

[4] https://aws.amazon.com/serverless/developer-tools

[5] https://aws.amazon.com/lambda/edge/

[6] https://aws.amazon.com/serverless/

[7] https://d0.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Architectures.pdf

[8] https://github.com/awslabs/lambda-refarch-webapp

[9] https://github.com/awslabs/lambda-refarch-mobilebackend

[10] https://aws.amazon.com/solutions/case-studies/bustle/

[11] https://github.com/awslabs/lambda-refarch-iotbackend

[12] https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-serverless-iot-back-ends-iot401

[13] https://www.youtube.com/watch?v=gKMaf5E-z7Q

[14] https://github.com/awslabs/lambda-refarch-streamprocessing

[15] https://github.com/awslabs/lambda-refarch-fileprocessing

[16] https://github.com/awslabs/lambda-refarch-imagerecognition

[17] https://www.youtube.com/watch?v=AcGv3qUrRC4&feature=youtu.be&t=1153

[18] https://aws.amazon.com/solutions/case-studies/thomson-reuters/

[19] https://www.youtube.com/watch?v=VFLKOy4GKXQ&feature=youtu.be&t=1449

[20] https://github.com/awslabs/lambda-refarch-mapreduce

[21] https://www.slideshare.net/AmazonWebServices/smc303-realtime-data-processing-using-aws-lambda/28

[22] https://www.slideshare.net/AmazonWebServices/smc301-the-state-of-serverless-computing-75290821/22

[23] https://aws.amazon.com/lambda/

[24] https://aws.amazon.com/serverless/