Arquitecturas multinivel sin servidor de AWS

con Amazon API Gateway y AWS Lambda

Noviembre de 2015

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. Este documento no constituye ninguna garantía, representación, compromiso contractual o condición por parte de AWS, sus filiales, proveedores o licenciantes. Las responsabilidades y obligaciones de AWS con sus clientes se rigen por los acuerdos de AWS, y este documento no forma parte ni supone una modificación de ningún acuerdo entre AWS y sus clientes.

Contenido

Resumen

En este documento técnico, se explica cómo las innovaciones de Amazon Web Services (AWS) pueden cambiar el modo en que se diseñan las arquitecturas multinivel de los modelos tradicionales, como microservicios, back-ends móviles y sitios web públicos. En la actualidad, los arquitectos y los desarrolladores pueden usar un modelo de implementación que incluya Amazon API Gateway y AWS Lambda para reducir los ciclos de operaciones y desarrollo necesarios a la hora de crear aplicaciones multinivel y administrar su funcionamiento.

Introducción

Durante décadas, las aplicaciones multinivel (tres niveles, N niveles, etc.) han constituido uno de los principales modelos arquitectónicos. El modelo multinivel cuenta con unas óptimas directrices que pueden seguirse para obtener componentes de aplicación desacoplados y escalables, cuya administración y mantenimiento puede llevarse a cabo por separado (normalmente, a través de diferentes equipos). Normalmente, las aplicaciones multinivel se crean aplicando un enfoque arquitectónico orientado a servicios (SOA) para usar servicios web. En este enfoque, la red actúa como una línea divisoria entre los diferentes niveles. Sin embargo, cuando se crea un nuevo nivel de servicios web como parte de una aplicación, hay muchos aspectos que no están diferenciados. La mayor parte del código que se escribe en una aplicación web multinivel es el resultado directo del propio modelo. Un ejemplo de ello sería el código que integra un nivel con otro; el código que define una API y el modelo de datos que los niveles usan para entenderse entre sí, y el código relacionado con la seguridad que garantiza que los puntos de integración de los niveles no se vean expuestos de forma no deseada.

Amazon API Gateway[1], un servicio para crear y administrar API, y AWS Lambda[2], un servicio para ejecutar funciones de código arbitrarias, pueden usarse conjuntamente para simplificar la creación de sólidas aplicaciones multinivel.

La integración de Amazon API Gateway con AWS Lambda permite que las funciones de código definidas por el usuario se activen directamente a través de una solicitud HTTPS definida por el usuario. Con independencia del volumen de solicitudes necesario, tanto API Gateway como Lambda se adaptarán de forma automática para ajustarse exactamente a las necesidades de la aplicación. Combinando estos servicios, podrá crear un nivel en la aplicación que le permita escribir el código que es importante para la aplicación, en lugar de centrarse en otros aspectos genéricos de la implementación de una arquitectura multinivel, como la creación de una arquitectura de gran disponibilidad, la elaboración de SDK de clientes, la administración del servidor o del sistema operativo (OS), el escalado y la implementación de un mecanismo de autorización de clientes.

Recientemente, AWS anunció que era capaz de crear funciones Lambda que se ejecutaban en Amazon Virtual Private Cloud (Amazon VPC)[3]. Esta característica amplía los beneficios que ofrece la combinación de API Gateway y Lambda para incluir una variedad de casos de uso en los que se requiere privacidad de red; por ejemplo, si necesita integrar un servicio web con una base de datos relacional que contenga información confidencial. La integración de Lambda y Amazon VPC ha ampliado indirectamente las funcionalidades de Amazon API Gateway, ya que permite que los desarrolladores puedan definir su propio conjunto de API HTTPS accesibles desde Internet delante de un back-end que sigue siendo privado y seguro y que forma parte de Amazon VPC. Los beneficios de este eficaz modelo son evidentes en cada uno de los niveles de una arquitectura multinivel. Este documento técnico se centra en el ejemplo más popular de una arquitectura multinivel: la aplicación web de **tres niveles**. Sin embargo, además de en las aplicaciones web de tres niveles tradicionales, este patrón multinivel puede emplearse en muchas otras situaciones.

Información general sobre la arquitectura de tres niveles

La arquitectura de tres niveles es un modelo frecuente entre las aplicaciones orientadas a los usuarios. Los niveles que conforman esta arquitectura son: **el nivel de presentación**, **el nivel lógico** y **el nivel de datos**. El nivel de presentación es el componente con el que el usuario interactúa directamente (como una página web, la interfaz de usuario de una aplicación móvil, etc.). El nivel lógico contiene el código necesario para traducir las acciones realizadas por el usuario en el nivel de presentación en la funcionalidad que controla el comportamiento de la aplicación. El nivel de datos se compone de los medios de almacenamiento (bases de datos, almacenes de objetos, memorias caché, sistemas de archivos, etc.) que albergan la información que es importante para la aplicación. En la figura 1, se muestra un ejemplo de una sencilla aplicación de tres niveles.

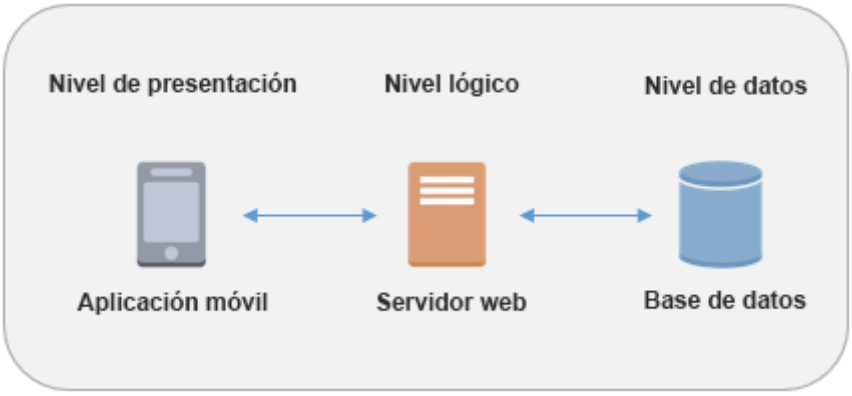

Figura 1: Modelo arquitectónico de una sencilla aplicación de tres niveles

En la Web, encontrará innumerables y excelentes recursos donde podrá obtener más información sobre el modelo *general* de la arquitectura de tres niveles. Este documento técnico se centra en un modelo de implementación específico de esta arquitectura que combina Amazon API Gateway y AWS Lambda.

Nivel lógico sin servidor

El nivel lógico de la arquitectura de tres niveles actúa como el cerebro de la aplicación. Es por ello que la integración de Amazon API Gateway y AWS Lambda para conformar el nivel lógico puede resultar tan revolucionaria. Las características de los dos servicios permiten crear una aplicación de producción sin servidor con gran disponibilidad, escalabilidad y seguridad. Es posible que su aplicación use miles de servidores; sin embargo, con este modelo, no tendrá que administrar ni uno solo de ellos. Además, al usar conjuntamente estos servicios administrados, conseguirá los siguientes beneficios:

- No tendrá que elegir un sistema operativo, ni protegerlo, administrarlo o aplicarle parches.
- No hay servidores, así que no tiene que preocuparse de monitorearlos ni de ajustar su tamaño o sus recursos.
- No correrá el riego de que sus costos se disparen por un aprovisionamiento excesivo.
- No existe peligro de que el desempeño se vea afectado por un aprovisionamiento deficiente.

Además, cada uno de los servicios cuenta con características específicas que favorecen el modelo arquitectónico de varios niveles.

Amazon API Gateway

Amazon API Gateway es un servicio completamente administrado que permite definir, implementar y mantener API. Los clientes se integran con las API a través de solicitudes HTTPS estándar. Su idoneidad para la arquitectura multinivel orientada a servicios resulta evidente. No obstante, cuenta con características y funciones específicas que hacen que esta herramienta resulte extremadamente eficaz para el nivel lógico.

Integración con AWS Lambda

Amazon API Gateway ofrece a las aplicaciones un sencillo mecanismo (solicitudes HTTPS) que permite aprovechar directamente las innovaciones de AWS Lambda. API Gateway es el puente que conecta el nivel de presentación con las funciones escritas en AWS Lambda. Una vez definida la relación cliente-servidor a través de la API, el contenido de las solicitudes HTTPS del cliente se transfieren a una función Lambda para su ejecución. Ese contenido incluye metadatos, así como los encabezados y el cuerpo de las solicitudes.

Desempeño estable de las API en todo el mundo

Cada implementación de Amazon API Gateway incluye una distribución de Amazon CloudFront[4] en segundo plano. Amazon CloudFront es un servicio web de entrega de contenido que usa la red global de ubicaciones de borde de Amazon como puntos de conexión para los clientes que se integran con la API. Esto ayuda a reducir la latencia de tiempo de respuesta total de la API. A través del uso de numerosas ubicaciones de borde en todo el mundo, Amazon CloudFront también le brinda la capacidad de combatir los escenarios de ataque por denegación de servicio distribuido (DDoS). Para obtener más información, consulte el documento técnico acerca de las prácticas recomendadas por AWS para combatir ataques[5] DDoS.

Puede mejorar el desempeño de determinadas solicitudes de API usando Amazon API Gateway para almacenar las respuestas en una caché en memoria opcional. Esto no solo favorece el desempeño de las solicitudes de API repetidas, sino que también reduce las ejecuciones back-end, lo que, a su vez, puede rebajar los costos generales.

Incentivar la innovación

El trabajo de desarrollo necesario para crear una nueva aplicación supone una inversión. Y es necesario que justifique esta inversión para poner en marcha el proyecto. Si reduce el volumen de inversión necesario para las tareas de desarrollo y el tiempo de trabajo, tendrá más libertad para experimentar e innovar.

En muchas aplicaciones basadas en servicios web multinivel, el nivel de presentación está fragmentado entre los usuarios (es diferente en los dispositivos móviles, los exploradores web, etc.). A menudo, estos usuarios no están limitados a una zona geográfica específica. Sin embargo, los niveles lógicos desacoplados no están fragmentados físicamente entre los usuarios. Todos los usuarios dependen de la misma infraestructura en la que se ejecuta el nivel lógico, lo que magnifica la importancia de dicha infraestructura. A menudo, cuando se implementa por primera vez el nivel lógico (son frecuentes comentarios del tipo “no es necesario instrumentar métricas en el lanzamiento inicial”; “al principio, el uso será lento, pero ya nos ocuparemos de adaptar los recursos más tarde”, etc.), se prefiere obviar algunos aspectos para lanzar una nueva aplicación más rápido. Esto puede generar deficiencias técnicas y riesgos operativos si es necesario implementar estos cambios en una aplicación que ya se esté ejecutando en producción. Amazon API Gateway *le* permite simplificar todos estos aspectos y lanzar sus aplicaciones más rápido porque el servicio ya los tiene implementados.

Es posible que desconozca cuál será la vida útil global de una aplicación o que sepa con certeza que su vida útil va a ser corta. Por ello, crear un caso de negocio para una nueva aplicación multinivel puede ser complicado. Sin embargo, puede resultarle más fácil si el punto de partida incluye las características administradas que Amazon API Gateway proporciona y si solo tiene que hacer frente a los costos estructurales una vez que sus API comienzan a recibir solicitudes. Para obtener más información, consulte Precios de Amazon API Gateway.[6]

Itere rápidamente y mantenga la flexibilidad

En las nuevas aplicaciones, es posible que la base de usuarios aún esté poco definida (tamaño, patrones de uso, etc.). Mientras la base de usuarios toma forma, el nivel lógico debe ser ágil. Su aplicación y su compañía deben ser capaces de adaptarse a las cambiantes expectativas de los primeros usuarios en adoptar la aplicación. Amazon API Gateway reduce el número de ciclos de desarrollo

necesarios para procesar una API desde su creación hasta su implementación. Amazon API Gateway permite crear integraciones ficticias[7] con las que puede generar respuestas de API directamente desde API Gateway a través de las que se pueden desarrollar aplicaciones cliente, mientras que, en paralelo, desarrolla la lógica de todo el back-end. Este beneficio no solo es aplicable a la primera implementación de una API, sino también una vez que la compañía ha decidido que la aplicación (y la API existente) debe ajustarse rápidamente a la respuesta de los usuarios. API Gateway y AWS Lambda permiten el control de versiones, de modo que la funcionalidad y las dependencias del cliente existentes pueden mantenerse sin cambios mientras se pone en marcha una nueva funcionalidad como una versión diferente de la función o de la API.

Seguridad

Implementar el nivel lógico de una aplicación web pública de tres niveles como servicio web aumenta la importancia de la seguridad. La aplicación debe garantizar que solo los clientes autorizados tienen acceso al nivel lógico (expuesto a través de la red). Amazon API Gateway aborda el problema de la seguridad a través de mecanismos que pueden garantizarle que su back-end está protegido. Para controlar el acceso, no debe confiar en proporcionar a las aplicaciones cliente cadenas de claves de API estáticas. Estas cadenas pueden extraerse de los clientes y usarse en otra parte. En su lugar, puede aprovechar los distintos mecanismos mediante los cuales Amazon API Gateway contribuye a proteger el nivel lógico:

- Todas las solicitudes a las API pueden hacerse a través de HTTPS para habilitar el cifrado de los datos en tránsito.
- Las funciones de AWS Lambda pueden restringir el acceso para que se establezca una relación de confianza únicamente entre una determinada API de Amazon API Gateway y una función específica de AWS Lambda. No habrá ninguna otra forma de invocar esa función Lambda a no ser que se use la API a través de la que ha elegido exponer dicha función.
- Amazon API Gateway permite generar SDK de cliente para integrarlos con las API. Esos SDK también se encargan de administrar las firmas de las solicitudes cuando la API requiere autenticación. Las credenciales de API que se usan en el lado cliente para la autenticación se transfieren directamente a la función de AWS Lambda (donde, si es necesario, la autenticación puede llevarse a cabo en el propio código que ha escrito).

- A cada combinación de recurso y método que crea como parte de la API se le asigna el nombre de un recurso de Amazon (ARN) específico, al que se puede hacer referencia en las políticas de AWS Identity y Access Management (IAM)[8].
  - Esto significa que sus API se van a considerar como "ciudadanos de primera clase" junto con las otras API propiedad de AWS. Las políticas de IAM pueden definirse con gran nivel de detalle; pueden hacer referencia a recursos o métodos específicos de una API creada con Amazon API Gateway.
  - El acceso a las API está regido por las políticas de IAM que crea fuera del contexto del código de la aplicación. Esto significa que no tiene que escribir código que administre o exija estos niveles de acceso. Dado que el código no existe, no puede contener errores ni verse comprometido.
  - Al brindar a los clientes acceso a las API a través de la autorización de AWS Signature versión 4 (SigV4)[9] y las políticas de IAM, estas mismas credenciales pueden restringir o permitir el acceso a otros servicios y recursos de AWS en función de las necesidades (por ejemplo, buckets de Amazon S3 o tablas de Amazon DynamoDB).

AWS Lambda

En esencia, AWS Lambda permite activar cualquier código arbitrario escrito en alguno de los idiomas admitidos (Node, JVM based y Python a partir de noviembre de 2015) en respuesta a un evento. Ese evento puede ser uno de los diferentes disparadores programáticos que AWS pone a disposición de los usuarios, lo que se denomina **origen del evento** (consulte aquí los orígenes de eventos admitidos actualmente[10]). Muchos casos de uso frecuentes de AWS Lambda se resuelven en torno a los flujos de trabajo de procesamiento de datos controlados por eventos, como el procesamiento de archivos almacenados en Amazon Simple Storage Service (Amazon S3)[11] o el streaming de registros de datos desde Amazon Kinesis[12].

Cuando AWS Lambda se utiliza junto con Amazon API Gateway, puede haber una función de AWS Lambda en el contexto de un servicio web tradicional, y esta función puede activarse directamente mediante una solicitud HTTPS. Amazon API Gateway actúa como la puerta principal del nivel lógico, pero ahora no es necesario ejecutar la lógica que subyace a estas API. Aquí es donde AWS Lambda entra en acción.

La lógica de negocio va aquí

AWS Lambda le permite escribir funciones de código (denominadas **controladores**), que se ejecutarán cuando las active un evento. Por ejemplo, puede escribir un controlador que se active cuando se produzca un evento, como una solicitud HTTPS dirigida a la API. Lambda le permite crear controladores modulares con el nivel de detalle que elija (un controlador por API o un controlador por cada método de API). Estos controladores pueden actualizarse, invocarse y modificarse por separado. A continuación, el controlador se libera para llegar a otras dependencias que tenga (por ejemplo, otras funciones que haya desarrollado con su código, bibliotecas, binarios nativos o incluso servicios web externos). Lambda le permite empaquetar todas las dependencias necesarias en la definición de la función durante su creación. Cuando cree una función, deberá especificar qué método interno del paquete de implementación actuará como controlador de la solicitud. Si lo desea, puede volver a usar el mismo paquete de implementación con otras definiciones de funciones Lambda, donde cada función Lambda puede tener un controlador único en el mismo paquete de implementación. En el modelo arquitectónico multinivel sin servidores, cada una de las API creadas en Amazon API Gateway se integrará con una función Lambda (y el controlador que contiene), que se encargará de ejecutar la lógica de negocio necesaria.

Integración de Amazon VPC

AWS Lambda, la piedra angular del nivel lógico, será el componente que se encargue de la integración directa con el nivel de datos. Como normalmente el nivel de datos contiene información confidencial del usuario o de la compañía, debe estar fuertemente protegido. En el caso de los servicios de AWS que pueden integrarse desde una función Lambda, el control de acceso puede administrarse a través de las políticas de IAM. Estos servicios incluyen Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), otras funciones de AWS Lambda, etc. Sin embargo, es posible que tenga un componente que dicte su propio control de acceso, como una base de datos relacional. En el caso de este tipo de componentes, podría mejorar la seguridad si los implementa en un entorno de red privado: Amazon Virtual Private Cloud (Amazon VPC)[13].

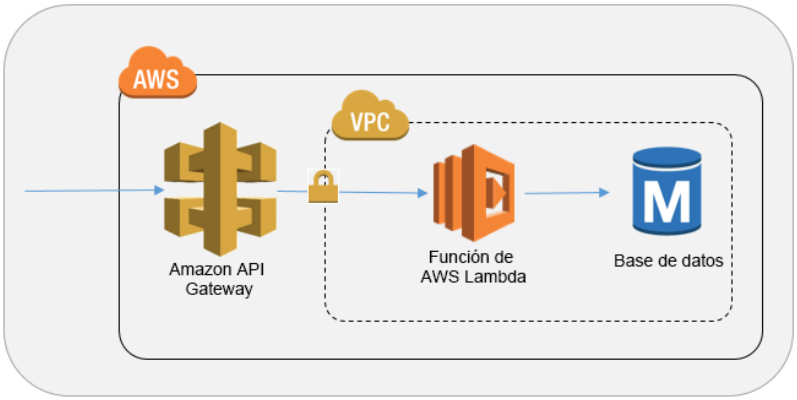

Figura 2: Modelo arquitectónico en el que se usa un VPC

El uso de un VPC hace que las bases de datos y otros medios de almacenamiento de los que depende la lógica de negocio puedan resultar inaccesibles a través de Internet. El VPC también garantiza que la *única* manera de interactuar con los datos desde Internet sea a través de las API que ha definido y de las funciones de código Lambda que ha escrito.

Seguridad

Para ejecutar una función Lambda, esta debe activarse mediante un evento o servicio al que se le hayan asignado permisos a través de una política de IAM. Es posible crear una función Lambda que no pueda ejecutarse *de ninguna de las maneras* a no ser que se invoque mediante una solicitud de API Gateway que usted haya definido. El código solo se procesará como parte del caso de uso válido, definido por la API que ha creado.

Cada función Lambda asume un rol de IAM, una función que debe concederse a través de una relación de confianza de IAM. El rol de IAM define los demás servicios o recursos de AWS con los que podrá interactuar la función Lambda (por ejemplo, una tabla de Amazon DynamoDB o un bucket de Amazon S3). Los servicios a los que tiene acceso la función se definirán y controlarán desde fuera de la propia función. Tal vez parezca poca cosa, pero resulta muy eficaz. De este modo, el código que escribe no necesita almacenar ni recuperar credenciales de AWS. Esto significa que no tiene que codificar de forma rígida las claves de las API ni escribir código para recuperarlas y almacenarlas en memoria. La capacidad para que sea la función Lambda la que llame a los servicios permitidos, según lo establecido en el rol de IAM que tiene asignado, está administrada por el propio servicio.

El nivel de datos

Al usar AWS Lambda como nivel lógico, dispone de multitud de opciones para almacenar la información en el nivel de datos. Estas opciones se dividen en dos amplias categorías: almacenes de datos hospedados por Amazon VPC y almacenes de datos habilitados para IAM. AWS Lambda tiene la capacidad de integrarse de forma segura con ambos.

Almacenes de datos hospedados por Amazon VPC

La integración de AWS Lambda con Amazon VPC permite que las funciones se integren con un gran número de tecnologías de almacenamiento de datos de forma privada y segura.

- Amazon RDS[14]

  Use uno de los motores disponibles a través de Amazon Relational Database Service (Amazon RDS). Conéctese a Amazon RDS directamente desde el código que ha escrito en Lambda igual que lo haría desde fuera de Lambda, pero con la ventaja de disfrutar de una integración sencilla con AWS Key Management Service (AWS KMS) para el cifrado de las credenciales de bases de datos.

- Amazon ElastiCache[15]

  Integre las funciones Lambda con una caché en memoria administrada para potenciar el desempeño de la aplicación.

- Amazon Redshift[16]

  Puede crear funciones que consulten de forma segura un almacén de datos corporativo para crear informes, paneles o recuperar consultas ad-hoc.

- Servicio web privado hospedado por Amazon Elastic Compute Cloud (Amazon EC2)[17]

  Es posible que tenga ya aplicaciones que se ejecutan de forma privada como un servicio web en un VPC. Ejecute las solicitudes HTTP sobre la red lógica y privada de VPC desde una función Lambda.

Almacenes de datos habilitados para IAM

Como AWS Lambda se integra con IAM, puede usar IAM para proteger la integración con cualquiera de los servicios de AWS, que pueden usarse directamente a través de las API de AWS.

- Amazon DynamoDB[18]

  Amazon DynamoDB es la base de datos NoSQL de AWS, y puede escalarse sin límites. Considere la posibilidad de usar Amazon DynamoDB si desea recuperar registros de datos (400 KB o más pequeños) con un desempeño en milisegundos de un solo dígito, a cualquier escala. Con el detallado control de acceso de Amazon DynamoDB, las funciones Lambda pueden aplicar la estrategia de conceder los mínimos privilegios al consultar datos específicos en DynamoDB

- Amazon S3[19]

  Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos a escala de Internet. Amazon S3 está diseñado para ofrecer una durabilidad del 99,999999999 % de los objetos; por ello, puede considerar la posibilidad de usar este servicio si la aplicación requiere un almacenamiento de gran durabilidad que sea barato. Además, Amazon S3 está diseñado para ofrecer una disponibilidad de los objetos del 99,99 % durante un año específico, por lo que puede constituir una buena opción si la aplicación requiere un almacenamiento de gran disponibilidad. El acceso a los objetos almacenados en Amazon S3 (archivos, imágenes, logs o datos binarios) se puede obtener directamente a través de HTTP. Las funciones Lambda pueden comunicarse de forma segura con Amazon S3 a través de puntos de enlace privados virtuales, mientras que los datos de S3 pueden restringirse exclusivamente a la política de IAM vinculada a la función Lambda.

- Amazon Elasticsearch Service[20]

  Amazon Elasticsearch Service (Amazon ES) es una versión administrada del conocido motor de búsqueda y análisis Elasticsearch. Amazon ES permite aprovisionar los clústeres de forma administrada, detectar errores y sustituir los nodos. Puede restringir el acceso a la API de Amazon ES usando políticas de IAM.

El nivel de presentación

Amazon API Gateway ofrece una gran variedad de posibilidades para el nivel de presentación. Cualquier cliente capaz de establecer comunicaciones HTTPS puede usar una API HTTPS accesible desde Internet. La lista que se incluye a continuación contiene ejemplos que puede usar en el nivel de presentación de la aplicación:

- Aplicación móvil: además de integrar la lógica de negocio personalizada a través de Amazon API Gateway y AWS Lambda, puede usar Amazon Cognito[21] como mecanismo para crear y administrar identidades de usuario.
- Contenido de sitios web estáticos (como archivos hospedados en Amazon S3): puede hacer que las API de Amazon API Gateway sean compatibles con el uso compartido de recursos entre orígenes (CORS). De este modo, los exploradores web podrán invocar directamente las API desde páginas web estáticas.
- Cualquier otro dispositivo cliente habilitado para HTTPS: muchos dispositivos conectados pueden comunicarse a través de HTTPS. No hay nada especial ni exclusivo en el modo en el que los clientes se comunican con las API que ha creado mediante Amazon API Gateway; es el HTTPS de siempre. No se necesitan licencias ni software de cliente específicos.

Ejemplos de modelos arquitectónicos

Puede implementar los siguientes modelos arquitectónicos usando Amazon API Gateway y AWS Lambda como el adhesivo que conforma el nivel lógico. En cada uno de los ejemplos, usaremos únicamente los servicios de AWS que no requieren que los usuarios administren su propia infraestructura.

Back-end móvil

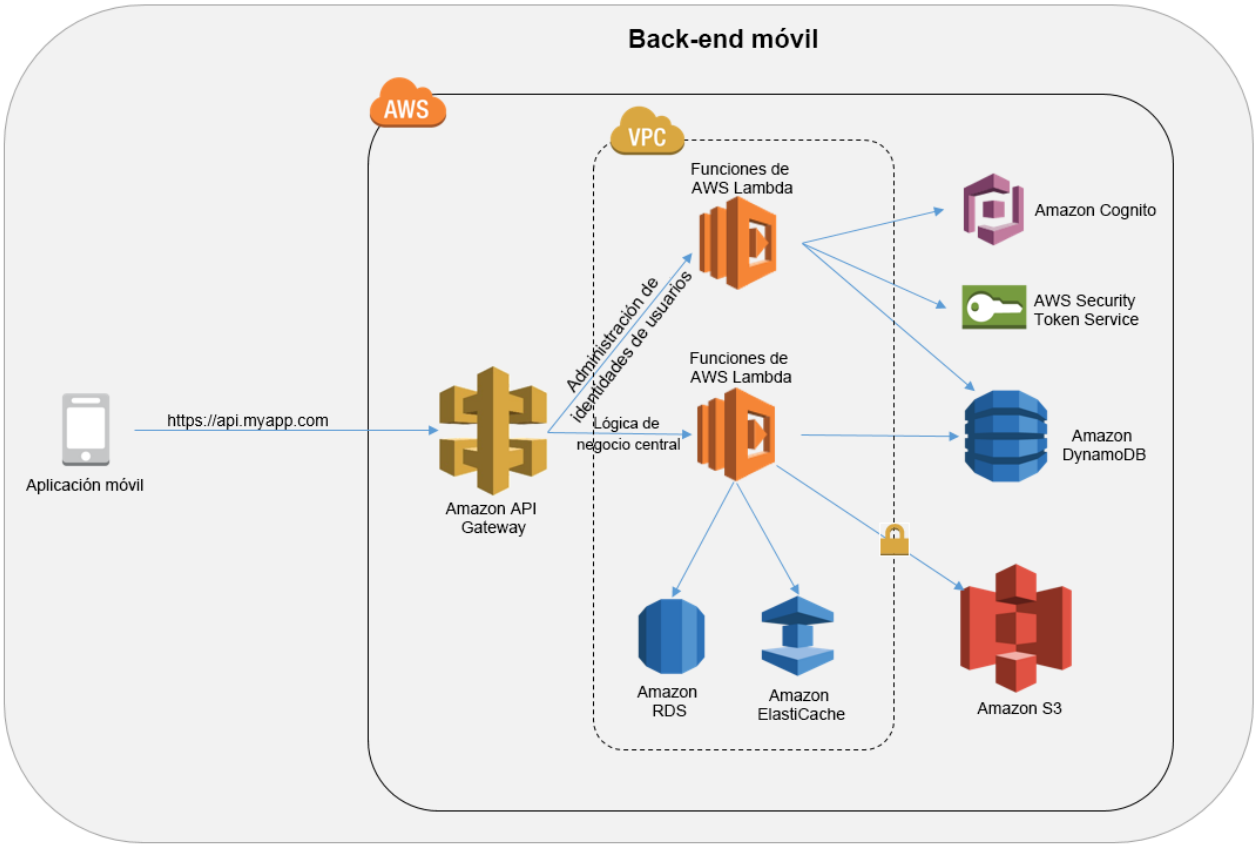

Figura 3: Modelo arquitectónico de un back-end móvil

- **Nivel de presentación:** aplicación móvil que se ejecuta en el smartphone de cada usuario.
- **Nivel lógico:** Amazon API Gateway y AWS Lambda. El nivel lógico está globalmente distribuido mediante la distribución de Amazon CloudFront creada con cada API de Amazon API Gateway. Se puede especificar un conjunto de funciones Lambda para la administración y autenticación de identidades de usuario que se administre a través de Amazon Cognito, que proporciona integración con IAM para credenciales de acceso de usuarios temporales así como con proveedores de identidades de otros fabricantes. Otras funciones Lambda pueden definir la lógica de negocio central del back-end móvil.

- **Nivel de datos:** los diferentes servicios de almacenamiento de datos pueden usarse en función de las necesidades (las opciones disponibles se describen más arriba en este documento).

Sitio web hospedado en Amazon S3

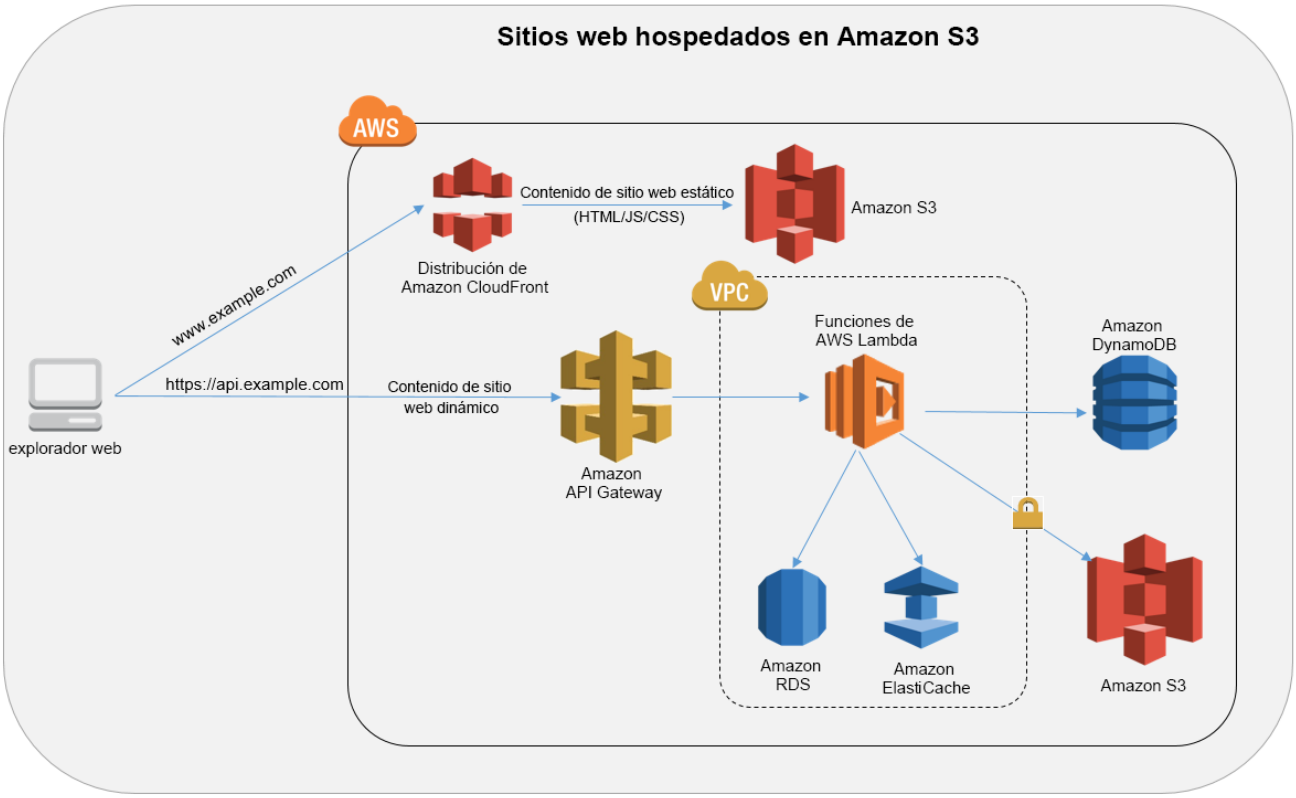

Figura 4: Modelo arquitectónico de un sitio web estático hospedado en Amazon S3

- **Nivel de presentación:** contenido del sitio web estático hospedado en Amazon S3 y distribuido por Amazon CloudFront. El hospedaje del contenido de sitios web estáticos en Amazon S3 es una alternativa económica para hospedar contenido en una infraestructura basada en servidores. Sin embargo, si el sitio web cuenta con características avanzadas, el contenido estático normalmente debe integrarse con un back-end dinámico.
- **Nivel lógico:** Amazon API Gateway y AWS Lambda. El contenido web estático hospedado en Amazon S3 se puede integrar directamente con Amazon API Gateway, lo que puede ser compatible con CORS.

- **Nivel de datos:** los diferentes servicios de almacenamiento de datos pueden usarse en función de las necesidades (las opciones disponibles se describen más arriba en este documento).

Entorno de microservicios

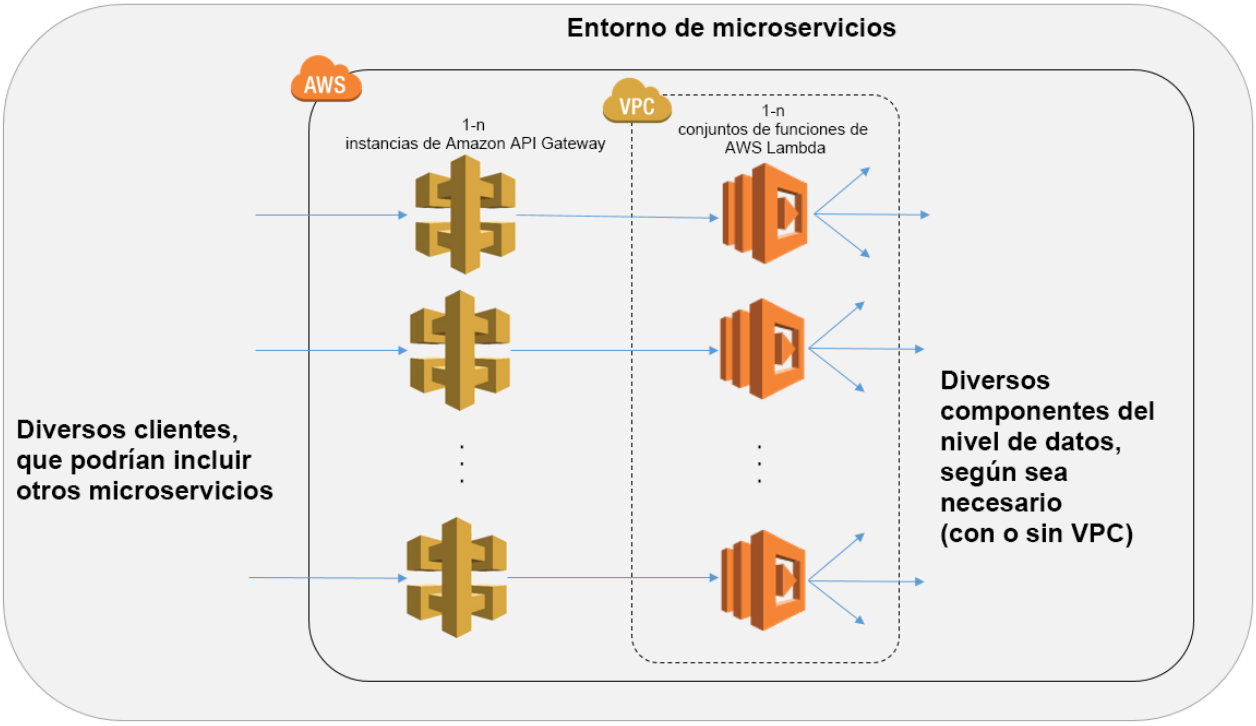

Figura 5: Modelo arquitectónico de un entorno de microservicios

El modelo arquitectónico de **microservicios** no está asociado con la arquitectura de tres niveles típica que hemos visto en este documento técnico. En una arquitectura de microservicios, existe una desconexión masiva de los componentes de software, lo que amplía los beneficios de una arquitectura multinivel. Todo lo que necesita para generar un microservicio es crear una API con Amazon API Gateway y unas funciones que se ejecuten posteriormente a través de AWS Lambda. Su equipo puede usar libremente estos servicios para desvincular y fragmentar el entorno con el nivel de detalle que prefiera.

En general, un entorno de microservicios puede entrañar las siguientes dificultades: sobrecarga reiterada al crear cada nuevo microservicio; problemas con la optimización de la densidad y el uso de servidores; complejidad para ejecutar simultáneamente varias versiones de diferentes microservicios, y proliferación de los requisitos de código del lado cliente que se va a integrar con diversos servicios independientes.

Sin embargo, si crea los microservicios usando el modelo sin servidores de AWS, estos problemas resultan más fáciles de resolver en algunos casos, mientras que, en otros, simplemente desaparecen por completo. El modelo de microservicios de AWS reduce los obstáculos de crear cada uno de los microservicios subsiguientes (Amazon API Gateway incluso permite clonar las API existentes). Con este patrón, ya no es importarte optimizar el uso de los servidores. Tanto API Gateway como Lambda cuentan con sencillas funciones para el control de versiones. Por último, Amazon API Gateway proporciona SDK de cliente generados mediante programación en diferentes lenguajes para reducir la sobrecarga de integración.

Conclusión

El modelo arquitectónico multinivel favorece la práctica recomendada de crear componentes de aplicaciones que resulten fáciles de mantener, que estén desacoplados y que sean escalables. Cuando crea un nivel lógico en el que la integración tiene lugar a través de Amazon API Gateway y la computación se realiza a través de AWS Lambda, está en el camino correcto para alcanzar todos esos objetivos, a la vez que reduce el esfuerzo necesario para conseguirlos. Juntos, estos servicios proporcionan un front-end de API HTTPS para sus clientes y un entorno seguro en el VPC para ejecutar la lógica de negocio. De este modo, puede sacar provecho de muchos escenarios frecuentes en los que puede usar estos servicios administrados, en lugar de ocuparse por sí mismo de una infraestructura tradicional basada en servidores.

Colaboradores

En este documento han participado las siguientes personas y organizaciones:

Andrew Baird, arquitecto de soluciones de AWS

Stefano Buliani, director senior de producto, Tech, AWS Mobile

Vyom Nagrani, director senior de producto, AWS Mobile

Ajay Nair, director senior de producto, AWS Mobile

Notas

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/