Título

FASE 4.1: Mobile Backend Serverless

Estado

Aceptada

Fecha

2023-11-01

Contexto

- Ya hemos finalizado la descomposición de nuestro monolito en funciones lambda, que se conectan a la base de datos Aurora serverless que migramos inicialmente. Con el objetivo inicial en mente de cumplir con todos los requerimientos de nuestro cliente y que nuestra arquitectura fuera totalmente serverless, en esta última fase hemos abordado diversos trade-off respecto a algunas decisiones de la misma como por ejemplo: ¿Por qué usar Api Gateway en lugar de Appsync?, ¿Que justifica el uso api Gateway en lugar de un ALB conectado a nuestras funciones Lambda?, ¿Porque el uso de lambda y no fargate + contenedores?.

Decisión

- **¿Por qué usar Api Gateway en lugar de Appsync?** Asumimos que nuestra aplicación es api restful la cual enrutará solicitudes HTTP/HTTPS a varios servicios Backend, nos decantamos por el uso de Api Gateway - Rest API, siendo además este último algo más barato que appsync.

- **¿Que justifica el uso api Gateway en lugar de un ALB conectado a nuestras funciones Lambda?** En general, API Gateway es una mejor opción si necesita una API RESTful que pueda manejar solicitudes HTTP/HTTPS y enrutarlas a varios servicios de backend. API Gateway también proporciona características de seguridad avanzadas, como la autenticación y la autorización que es uno de los requerimientos de nuestro cliente.

- **¿Porque el uso de lambda y no fargate + contenedores?** Tras analizar el flujo de nuestra aplicación sabemos que el tiempo de ejecución de las diferentes funciones no supera los 55 segundos en el peor de los casos, consideramos por lo tanto que no es necesario el uso de contenedores para ninguna de las funciones de nuestra aplicación.

Consecuencias

Positivas:

- **Escalabilidad**: La arquitectura backend serverless es altamente escalable y puede manejar cargas de trabajo variables sin problemas. Los servicios serverless se escalan automáticamente según la demanda, lo que significa que solo se utilizan los recursos necesarios para manejar la carga de trabajo actual.

- **Costo**: La arquitectura backend serverless puede ser más económica que las arquitecturas tradicionales. Los servicios serverless se facturan en función del número de solicitudes y la duración de la ejecución de la función. Esto significa que solo paga por lo que usa, lo que puede reducir los costos en comparación con las aplicaciones tradicionales que se ejecutan en servidores dedicados.

- **Flexibilidad:** La arquitectura backend serverless es altamente flexible y puede manejar una amplia variedad de cargas de trabajo. Los servicios serverless se pueden utilizar para una amplia variedad de casos de uso, desde aplicaciones web hasta procesamiento de datos y análisis.

- **Facilidad de implementación**: La arquitectura backend serverless es fácil de implementar y administrar. Los servicios serverless no requieren que administre servidores o recursos de infraestructura, lo que puede reducir los costos de administración y mantenimiento.

Negativas:

Limitaciones de tiempo de ejecución: Los servicios serverless tienen un límite de tiempo de ejecución máximo, lo que significa que las funciones deben completarse dentro de un tiempo determinado. Si una función tarda más en ejecutarse que el tiempo de espera máximo, se produce un error.

Limitaciones de memoria: Los servicios serverless tienen un límite de memoria asignada a cada función. Si una función utiliza más memoria de la asignada, se produce un error.

Limitaciones de tamaño de carga útil: Los servicios serverless tienen un límite en el tamaño de la carga útil que se puede enviar a través de ellos. Si una carga útil es demasiado grande, se produce un error.

Complejidad de la implementación: La arquitectura backend serverless puede ser más compleja de implementar que las arquitecturas tradicionales. Los servicios serverless requieren que las funciones se dividan en pequeñas piezas que se ejecutan de forma independiente, lo que puede ser más difícil de implementar que las aplicaciones monolíticas tradicionales. Deben dedicarse tiempo y recursos adicionales a la observabilidad y a la trazabilidad de los microservicios para garantizar el rendimiento, la resiliencia y la detección temprana de posibles errores, ya que cada microservicio al ser componentes independientes necesitan una atención individual.

Compliance

- Nuestro cliente opera en Europa y Estados Unidos Por lo que ha de cumplir con los estándares GDPR e HIPAA)

Procedimiento - Arquitectura de Soluciones

La arquitectura propuesta aprovecha múltiples servicios de AWS totalmente administrados, incluyendo AWS Lambda, API Gateway, Amazon Cognito, Amazon Aurora Serverless y S3, todo ello para implementar un backend, altamente escalable y robusto para aplicaciones móviles. La arquitectura adopta un enfoque totalmente serverless que elimina la necesidad de preocuparse por el aprovisionamiento o des aprovisionamiento de capacidad de cómputo y de almacenamiento, lo que se traduce en una reducción de la sobrecarga operativa.

Uno de los beneficios clave es su capacidad de escalar de manera óptima a cualquier escala, desde uno a millones de usuarios sin ningún cambio en su implementación. La arquitectura también optimiza los costos porque brinda una alineación completa entre el costo y la escalabilidad, ya que se evitan tener recursos aprovisionados de los que no se va a hacer uso.

La solución utiliza una arquitectura de varios niveles, que incluye un nivel de Presentación (Web Tier) que consta de aplicaciones nativas Android e IOS, un nivel web para aplicaciones web móviles alojadas estáticamente en S3, un nivel lógico impulsado por funciones Lambda expuestas como microservicios y un nivel de datos impulsado por tecnología de almacenamiento escalables como S3, Aurora serverless. Cada nivel de la solución consta de servicios totalmente gestionados que son capaces de escalar de forma independiente.

Niveles

Presentación y nivel web

El nivel de presentación de la solución consta de una aplicación nativa de Android e iOS que encapsula la interfaz de usuario y la lógica de presentación de la aplicación. Para la aplicación web móvil, el nivel de presentación también incluye un nivel web alojado estáticamente en Amazon S3 y distribuido a través de Cloudfront (CDN) y protegido por WAF en capa 7.

Todas las aplicaciones del nivel de presentación (Android, iOS y Web móvil) interactúan con el nivel lógico a través de puntos finales de API Gateway. Las aplicaciones utilizan API Gateway Client SDK generado para Android, iOS y JS para consumir puntos finales de API Gateway. Toda la comunicación entre las aplicaciones móviles y Logic Tier está protegida mediante AWS Cognito y WAF.

Route 53 tiene configurada una public hosted zone con los registros dns de nuestra aplicación y un alias apuntando hacia nuestro cloudfront, además de los Health check para un multiregion active-active del que hablaremos en la siguiente fase, donde aplicamos del DR en 2 regiones cumpliendo con el requerimiento de nuestro cliente de tener desplegada la app en varias regiones.

Nivel lógico

El nivel lógico de la solución encapsula la lógica empresarial y la inteligencia de la solución dentro de funciones Stateless de AWS Lambda. Las funciones Lambda se comunican internamente con el nivel de datos y otras dependencias para ejecutar la

lógica empresarial deseada. La funcionalidad del nivel lógico está expuesta al nivel de presentación a través de API RESTful personalizadas con tecnología de Amazon API Gateway. Estas API actúan como una puerta de entrada para el nivel de presentación para acceder a los datos, la lógica empresarial y la funcionalidad expuesta por los servicios back-end.

Esta Logic Tier proporciona las siguientes características y beneficios a la solución:

- El uso de AWS Lambda proporciona una plataforma para ejecutar la lógica empresarial sin la necesidad de administrar ningún servidor.
- Las funciones Lambda se pueden invocar explícitamente a través de API Gateway endpoints o en respuesta a una variedad de eventos.
- Lambda aumenta o disminuye automáticamente para coincidir con la tasa de eventos/patrones de tráfico. Se activa Provisioned Concurrency para evitar Cold start en nuestras funciones lambda.
- API Gateway está configurado para usar una distribución de CloudFront internamente para reducir la latencia de las API y cachear el contenido.
- Aws Certificate Manager (ACM) aloja nuestro certificado ssl que conectamos a cloudfront.
- La integración con el servicio Amazon SNS permite que Logic Tier envíe notificaciones push entre dispositivos en tiempo real.

Nivel de datos

El nivel de datos de la solución consta de servicios totalmente administrados, escalables y de alta disponibilidad como Aurora Serverless y Amazon S3 y RDS proxy. Aurora proporciona un almacenamiento escalable (cloud native) para almacenar datos relacionales y Amazon S3 proporciona un almacenamiento de objetos muy duradero e infinitamente escalable para almacenar fotos, vídeos, datos binarios y otros .

Cuando se usan funciones lambda con una base de datos RDS, estas abren y mantienen la conexión a la base de datos pudiendo resultar en un error de “TooManyConnectiones”, para evitarlo usamos RDS proxy que mejorará la eficiencia de nuestra base de datos reduciendo el estrés sobre la misma.

Secrets Manager se encarga de rotar el password de conexión a nuestro cluster de Aurora Serverless como buena práctica y es usado por RDS proxy y lambda para obtener esa credencial de conexión a la base de datos.

Usaremos S3 para almacenar/consultar recursos de nuestra app que sean de mayor tamaño alojados como objetos.

Los datos almacenados en Amazon S3 Standard por defecto, se archivan en el servicio S3 IA tras pasar 30 días desde su último acceso, se archivarán en Amazon Glacier Instant retrieval tras 90 desde su último acceso y en Glacier Deep Archive después de 180 días desde su último acceso, aplicando una Lifecycle rule.

Consideraciones clave

Autenticación de usuarios, gestión de identidades y sincronización de datos entre dispositivos

La arquitectura de referencia también brinda la capacidad de administrar identidades de usuarios y sincronizar datos específicos del usuario, como las preferencias del usuario, en múltiples dispositivos. Ambos requisitos se cumplen mediante el uso de Amazon Cognito

Amazon Cognito ofrece administración de identidades móviles y sincronización de datos entre dispositivos de forma inmediata sin la necesidad de implementar una lógica de backend personalizada. Para la migración de los usuarios existentes hemos configurado un desencadenador Lambda, donde Amazon Cognito detecta que es el primer inicio de sesión de ese usuario y llama al desencadenador Lambda, el cual recupera los parámetros (username, password, validationData y clientmetadata), este desencadenador lambda, autentica al usuario con el sistema de autenticación existente en el monolito, recupera los atributos del usuario relevantes (nombre, correo ...) y por último crea un nuevo usuario en Cognito con los atributos recuperados

(Username, correo..). De esta forma los usuarios pueden migrar sin experimentar interrupciones.

La solución se puede integrar con proveedores de identidades públicas como Facebook y Google para generar identidades de usuario únicas que se pueden almacenar en un grupo de identidades administrado por Amazon Cognito junto con datos específicos del usuario en el almacén de sincronización de Cognito.

Operational Excellence

Para lograr la excelencia operacional hemos automatizado la implementación de nuestra infraestructura mediante el despliegue CI/CD anteriormente comentado.

Se ha creado un sistema resiliente gracias a las características que aporta una arquitectura serverless a nivel de región en caso de fallas.

El aplicativo está en constante monitorización mediante cloudwatch (Metrics, alarms, Insights) y cloudtrail, creando alarmas por ejemplo para posible problemas de concurrencias con las lambda, Throttles con lambda o api gateway etc.., notificando mediante SNS. Se configura Eventbridge + CloudTrail para por ejemplo identificar, si alguna llamada a API intenta modificar el Security group de la base datos Aurora o del RDS proxy.

Se hará uso de X-ray, lo que nos proporcionará una visualización extremo a extremo de las solitudes realizadas a la aplicación, pudiendo identificar cuellos de botella y problemas de rendimiento.

Security

La solución implementa el Least privilege principle para proteger datos y recursos. El uso de roles de IAM proporciona acceso controlado a diversos recursos y servicios de AWS. Y se han implementado mecanismos que mejoran los aspectos de seguridad, asegurando que solo usuario y apps autorizadas pueden utilizar el servicio.

Una vez que un usuario final se autentica con un proveedor de identidad pública, Amazon Cognito genera credenciales de temporales con privilegios limitados para el usuario. Estas credenciales temporales están asociadas con un role de IAM específico que define los permisos para acceder a los recursos de AWS. Las aplicaciones del nivel de presentación pueden usar las credenciales temporales para llamar a la API expuesta por el nivel lógico. Esto elimina la necesidad de codificar las credenciales en la aplicación. Todas las solicitudes a las API de nivel lógico se realizan a través de HTTPS para permitir el cifrado en tránsito. Api Gateway está configurado junto con waf y cloudfront para solo aceptar solicitudes desde este último mediante CloudFront Custom header (X-Origin Verify) generada en Secrets Manager con rotación cada cierto tiempo como buena práctica.

KMS encripta nuestro secrets manager, y además permite al role que asume nuestra cuenta de desarrollo(mediante la configuración de la pertinente resource policy en Secrets manager y una KMS policy) desencriptar y acceder a nuestra base de datos Aurora.

Performance Efficiency

La solución aprovecha múltiples servicios totalmente administrados de AWS que garantizan automáticamente alta disponibilidad, confiabilidad y escalamiento automático para satisfacer los picos y valles en la curva de demanda, reduciendo así significativamente los gastos generales de administración. Por ejemplo, AWS Lambda ejecuta código en una infraestructura informática de alta disponibilidad y realiza toda la administración de los recursos informáticos, incluido el mantenimiento del servidor y del sistema operativo, el aprovisionamiento de capacidad y el escalado automático, la implementación de códigos y parches de seguridad, y el monitoreo y registro de códigos.

Reliability

Los servicios subyacentes de AWS que consume la solución ofrecen tolerancia a fallas integrada y garantizan una alta disponibilidad mediante el uso de múltiples zonas de disponibilidad en cada región para ayudar a proteger contra fallas de máquinas individuales o centros de datos.

Mediante el uso de plantillas de AWS CloudFormation, la solución se puede empaquetar y replicar fácilmente en múltiples entornos de implementación dentro de AWS.

Cost Optimization

La solución adopta todas las virtudes de una arquitectura serverless en cuento a la optimización de costos, como son el pago por uso que reduce el gasto en comparación con aplicaciones alojadas en instancias ec2, el escalado automático donde la aplicación escala en función de la demanda minimizando el gasto, sin costo de infraestructura ya que no se requiere administrar servidores o recursos de infraestructura logrando reducir los costos de administración y mantenimiento.

Independientemente del ahorro asociado al tipo de arquitectura desplegada proponemos:

- Cloudfront: Seleccionaremos solo Price Class 100, que todos los usuarios de nuestra infrastructura están repartidos entre las regiones de Us-west-1 y eu-east-1 permitiendo disminuir los costes de nuestro caching.

- Compute Savings plan 3 años : Mediante Organizations y su billing consolidado podremos aplicar un compute savings plans a nuestras funciones Lambda en todas nuestras cuentas, lo que nos permitirá ahorrar hasta un 66% del coste respecto al uso bajo demanda.

- Se usará Compute optimizer para analizar de forma periódica nuestras funciones lambda y obtener recomendaciones para ajustar el tamaño de memoria acorde a su uso, ayudando a reducir costos y mejorando el rendimiento.

Sustainability

AWS implementa medidas como eficiencia energética, reducción de residuos, seguimiento de carbono e innovación sostenible, haciendo que al migrar nuestro cliente desde on-premise al cloud se beneficie de todas ellas.

Notas

- Author: Fran Díaz
- Version: 0.1
- Changelog:
    - 0.1: Versión Inicial propuesta.