Lección 4

Creación de arquitecturas sin servidor con AWS Lambda

AWS Lambda

- Es un servicio de cómputo completamente administrado
- Ejecuta su código según una programación o como respuesta a eventos (por ejemplo, cambios en un bucket de Amazon S3 o en una tabla de Amazon DynamoDB)
- Compatible con Java, Go, PowerShell, Node.js, C#, Python, Ruby y API de tiempo de ejecución
- Puede funcionar en ubicaciones perimetrales más cercanas a sus usuarios

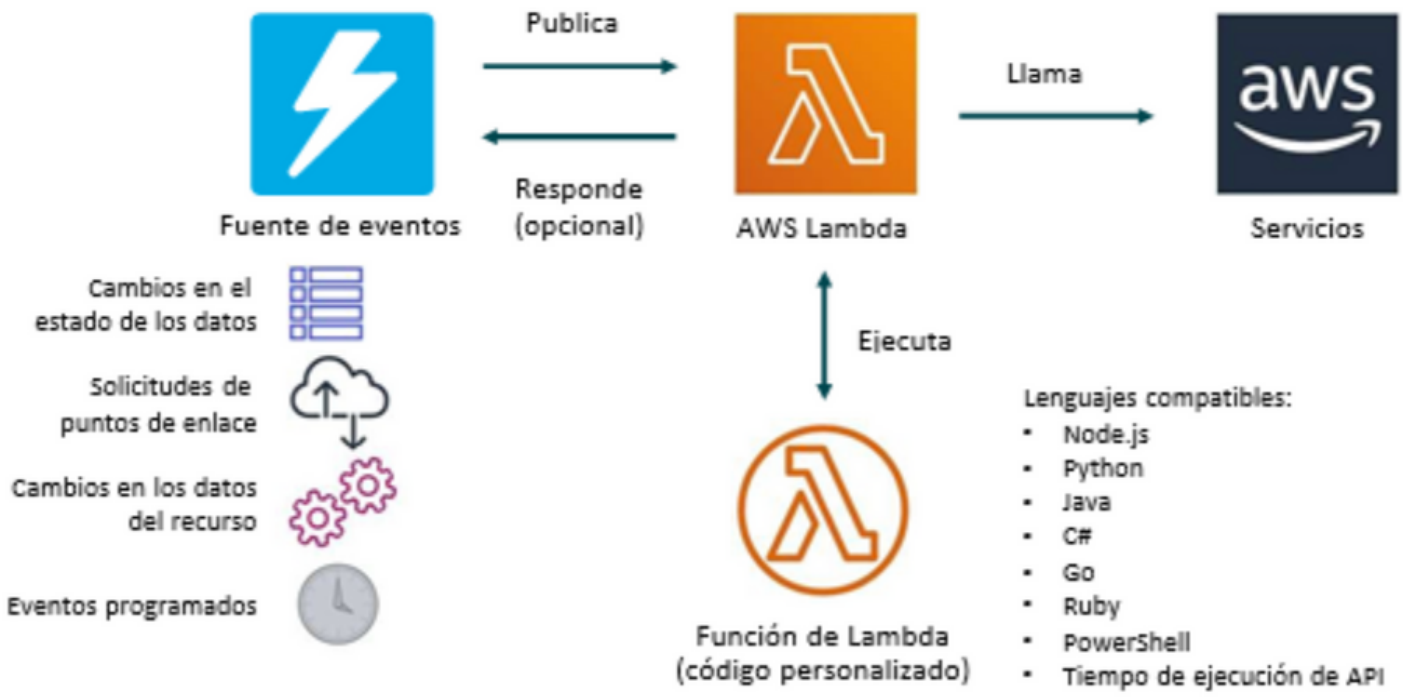
AWS Lambda

AWS Lambda es un servicio de cómputo completamente administrado que ejecuta código en respuesta a eventos y administra automáticamente los recursos de cómputo subyacentes. Lambda ejecuta el código en una infraestructura de cómputo de alta disponibilidad y realiza toda la administración de los recursos de cómputo, incluido el mantenimiento de servidores y sistemas operativos, el aprovisionamiento de la capacidad, el escalado automático, y la supervisión y el registro de códigos.

AWS Lambda admite de forma nativa código Java, Go, PowerShell, Node.js, C#, Python y Ruby, y proporciona una API de tiempo de ejecución que le permite utilizar otros lenguajes de programación para crear sus funciones.

Lambda@Edge es una función de Amazon CloudFront que le permite ejecutar el código más cerca de los usuarios de su aplicación, lo que mejora el rendimiento y reduce la latencia.
Lambda@Edge ejecuta su código como respuesta a eventos generados por la red de entrega de contenido (content delivery network, CDN) de Amazon CloudFront. Lambda@Edge le permite ejecutar funciones de Lambda de Node.js y Python para personalizar el contenido que entrega Amazon CloudFront. Para obtener información sobre cómo agregar encabezados de respuesta de seguridad HTTP, lea esta publicación del blog Redes y entrega de contenido de AWS.

Funcionamiento de AWS Lambda

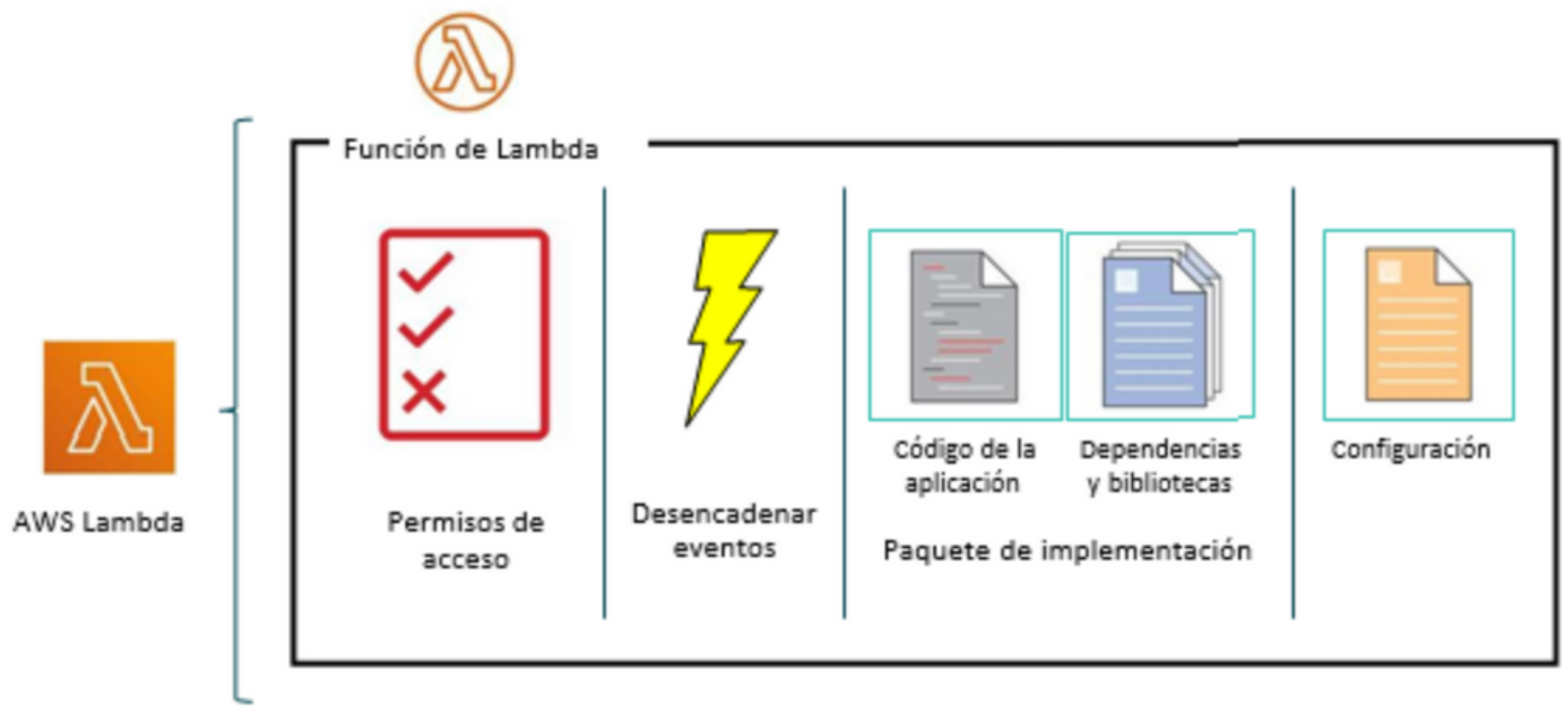


AWS Lambda se integra con otros servicios de AWS para invocar funciones de Lambda. Una función de Lambda es código personalizado que se escribe en uno de los lenguajes compatibles con Lambda. Puede configurar desencadenadores para invocar una función como respuesta a eventos del ciclo de vida de los recursos, responder solicitudes HTTP entrantes, consumir eventos de una cola o ejecutarse según una programación.

Una fuente de eventos es la entidad que publica el evento en Lambda. La función de Lambda procesa el evento y Lambda ejecuta la función de Lambda por usted.

Las funciones de Lambda son sin estado, es decir, no tienen afinidad con la infraestructura subyacente. Lambda puede iniciar rápidamente tantas copias de la función como sea necesario para escalarse a la velocidad de los eventos entrantes.

Funciones de Lambda



Cuando se crea una función de Lambda, se definen los permisos para la función y se especifica qué eventos la desencadenan. También se crea un paquete de implementación que incluye el código de la aplicación y las dependencias y bibliotecas necesarias para ejecutar el código. Por último, configure los parámetros de tiempo de ejecución, como la memoria, el tiempo de espera y la simultaneidad. Cuando se invoca su función, Lambda ejecutará un entorno basado en el tiempo de ejecución y las opciones de configuración que haya seleccionado.

Para obtener más información sobre cómo se ejecuta AWS Lambda, consulte la documentación de AWS.

Anatomía de una función de Lambda

Handler()

Función que se ejecutará después de la invocación

Objeto de evento

Datos enviados durante la invocación de la función de Lambda

Objeto del contexto

Métodos disponibles para interactuar con la información de tiempo de ejecución (ID de solicitud, grupo de registro, más)

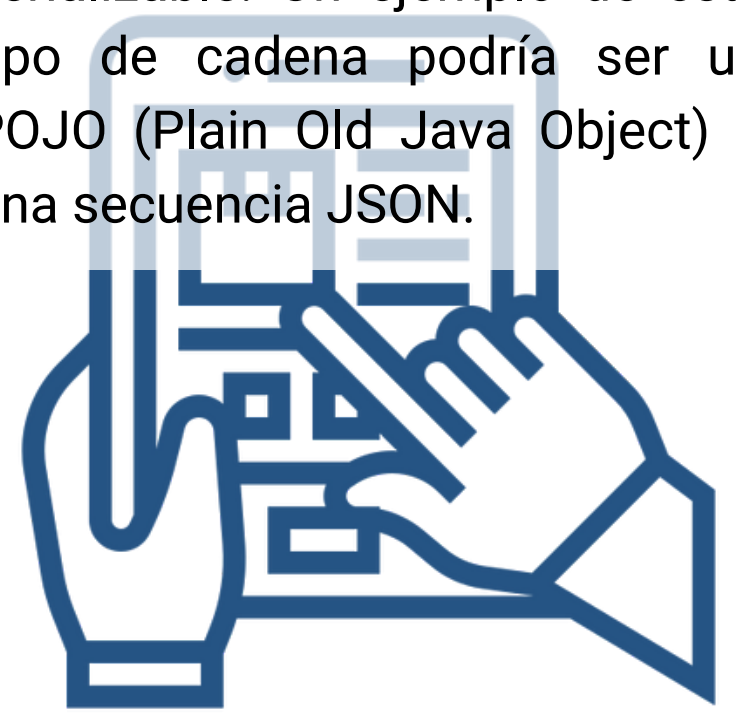
```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World')
    }
```

Cuando se invoca una función de Lambda, el código comienza a ejecutarse en el controlador. El controlador es un método o función de código específico que crea e incluye en su paquete. Se especifica el controlador cuando se crea una función de Lambda. Cada lenguaje compatible tiene sus propios requisitos sobre cómo se puede definir un controlador de funciones y hacer referencia a este dentro del paquete. Después de que se invoque correctamente el controlador en su función de Lambda, el entorno en tiempo de ejecución pertenece al código que escribió.

El controlador siempre toma dos objetos: el objeto de evento y el objeto de contexto.

El objeto de evento proporciona información acerca del evento que desencadenó la función de Lambda. Este evento podría ser un objeto predefinido generado por un servicio de AWS o un objeto personalizado definido por el usuario como una cadena serializable. Un ejemplo de este tipo de cadena podría ser un POJO (Plain Old Java Object) o una secuencia JSON.



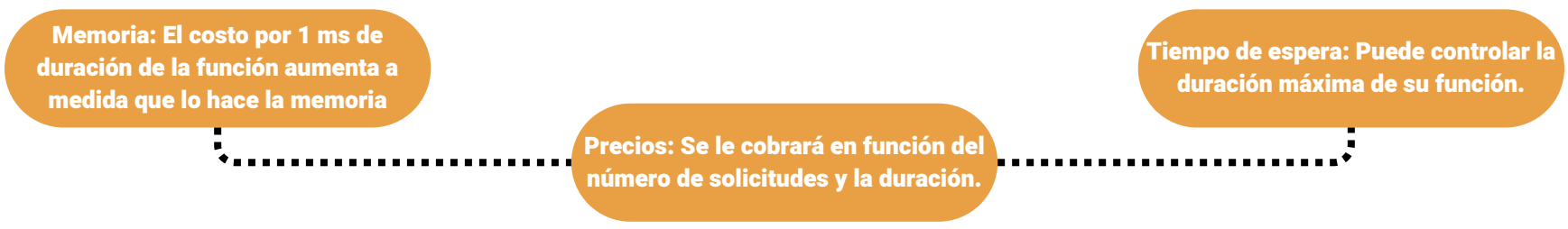
El contenido del objeto de evento incluye todos los datos y metadatos que la función de Lambda necesita para impulsar la lógica. El contenido y la estructura del objeto de evento difieren según la fuente del evento que lo haya creado. Por ejemplo, un evento que crea API Gateway tendrá detalles relacionados con la solicitud HTTPS que realizó el cliente, como la ruta, la cadena de consulta y el cuerpo de la solicitud. Sin embargo, un evento creado por Amazon incluye detalles sobre el bucket y el nuevo objeto.

AWS genera el objeto de contexto y proporciona metadatos sobre el entorno en tiempo de ejecución. El objeto de contexto permite que el código de función interactúe con el entorno en tiempo de ejecución de Lambda. Los contenidos y la estructura del objeto de contexto varían en función del tiempo de ejecución del lenguaje que use la función de Lambda.

Sin embargo, como mínimo, el objeto de contexto contiene:

- awsRequestId: esta propiedad se utiliza para realizar un seguimiento de las invocaciones específicas de una función de Lambda (importante para informar los errores o al contactar con AWS Support)
- logStreamName: indica el flujo de registro de CloudWatch al que se enviarán los enunciados de registro
- getRemainingTimeInMillis(): este método devuelve el número de milisegundos que faltan antes de que se agote el tiempo de ejecución de su función.

Configuración y facturación de las funciones de Lambda



La memoria y el tiempo de espera son configuraciones que determinan el rendimiento de la función de Lambda. Estas configuraciones afectan a su facturación. Con AWS Lambda, se le cobrará en función del número de solicitudes de las funciones (el número total de solicitudes en todas sus funciones) y la duración (el tiempo que tarde en ejecutarse el código). El precio depende de la cantidad de memoria que asigne a su función.

Memoria: especifica la cantidad de memoria que desea asignar a su función de Lambda. Lambda asigna capacidad de procesamiento de CPU proporcional a la memoria. Lambda tiene un precio que hace que el costo por 1 ms de duración de la función aumente a medida que aumenta la configuración de la memoria. Por ejemplo, supongamos que tiene una función de Lambda con 256 MB de memoria y que se ejecuta durante 110 milisegundos. Esta función costará el doble que una función de Lambda con 128 MB de memoria que se ejecute durante el mismo tiempo.

Tiempo de espera: puede controlar la duración máxima de su función configurando el tiempo de espera. Puede establecer el valor de tiempo de espera de una función en cualquier valor hasta un máximo de 15 minutos. Cuando se alcanza el tiempo de espera especificado, AWS Lambda detiene la ejecución de su función de Lambda. Utilizar un tiempo de espera puede evitar mayores costos derivados de funciones de ejecución prolongada. Debe encontrar el equilibrio adecuado entre no prolongar demasiado la función y poder terminarla en circunstancias normales.

Siga estas prácticas recomendadas:

- Pruebe el rendimiento de la función de Lambda para garantizar la elección de la configuración de capacidad de memoria óptima. Puede ver el uso de memoria de su función en los registros de Amazon CloudWatch.
- Pruebe su función de Lambda para analizar el tiempo de ejecución y determinar el mejor valor de tiempo de espera. Esto es especialmente importante cuando la función de Lambda realiza llamadas de red a los recursos que podrían no ser capaces de gestionar el escalado de las funciones de Lambda.

Consulte los siguientes recursos para obtener información sobre:

- Límites de AWS Lambda
- Precios de AWS Lambda

Ahora, el instructor puede hacer una demostración de cómo crear una función de AWS Lambda.

Ejemplo de AWS Lambda:

Juego de navegador de máquina tragamonedas simulada

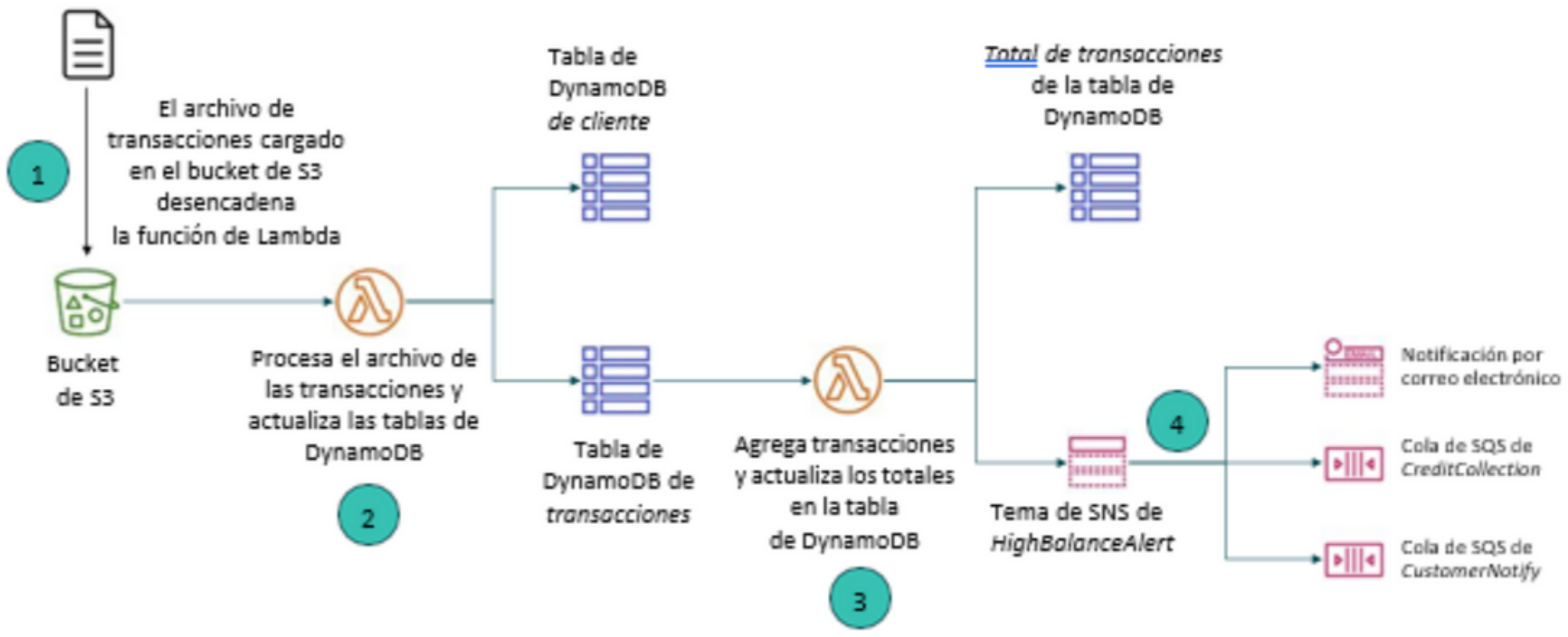

```
lambda.invoke(pullParams, function (error, data) {
 if (error) {
 prompt(error);
 } else {
 pullResults = JSON.parse(data.Payload);
 }
});
```

```
{
 isWinner: false,
 leftWheelImage : {S : 'cherry.png'},
 midWheelImage : {S : 'puppy.png'},
 rightWheelImage : {S : 'robot.png'}
}
```

Puede crear funciones de Lambda para realizar diversas tareas. En este ejemplo se muestra un juego basado en navegador que simula una máquina tragamonedas. El juego invoca una función de Lambda que genera los resultados aleatorios de cada tiro en la máquina tragamonedas. La función devuelve esos resultados como los nombres de archivo de las imágenes utilizadas para mostrar el resultado. Las imágenes se almacenan en un bucket de Amazon S3 que está configurado para funcionar como un host web estático de los activos HTML, CSS y otros que se necesitan para presentar la experiencia de la aplicación.

Ejemplo de una función de Lambda basada en eventos:

Procesamiento de pedidos



En este ejemplo se muestra cómo se puede utilizar Lambda en una solución para el procesamiento de pedidos.

En esta arquitectura:

1. Un cliente carga un archivo de transacciones a un bucket de S3, lo que desencadena la ejecución de una función de Lambda.
2. Una función de Lambda procesa el archivo de transacciones y actualiza las tablas Customer y Transactions de DynamoDB.
3. Los cambios en la tabla Transactions de DynamoDB activan una segunda función de Lambda para agregar las transacciones y actualizar los totales en la tabla Transaction total de DynamoDB. También envía un mensaje HighBalancerAlert al tema de SNS.
4. El tema de SNS HighBalancerAlert envía una notificación por correo electrónico al cliente y actualiza las colas de SQS CreditCollection y CustomerNotify para el procesamiento del pago.

Capas de Lambda

- Habilitan funciones para compartir código fácilmente: puede cargar una capa una vez y utilizarla como referencia en cualquier función
- Promueven la división de responsabilidades: los desarrolladores pueden iterar más rápido en la escritura de la lógica empresarial
- Permiten reducir los paquetes de implementación
- Límites:
  - Hasta cinco capas
  - 250 MB

Cuando se crean aplicaciones sin servidor, es común tener código que se comparte a través de las funciones de Lambda. Puede ser código personalizado que utilizan dos o más funciones, o una biblioteca estándar que se agrega para simplificar la implementación de su lógica empresarial.

Antes, se empaquetaba e implementaba este código compartido junto con todas las funciones que lo utilizaban. Ahora, puede configurar su función de Lambda para incorporar código y contenido adicionales como capas. Una capa es un archivo .zip que contiene bibliotecas, un tiempo de ejecución personalizado u otras dependencias.

Con las capas de Lambda, las funciones pueden compartir código. Los desarrolladores utilizan capas para cargar código una vez y reutilizarlo varias veces. Con las capas, puede utilizar las bibliotecas en la función sin necesidad de incluirlas en el paquete de implementación.

Compartir el código de esta forma puede ayudar a promover la división de responsabilidades. Una persona puede encargarse de gestionar la biblioteca principal. Otra persona puede ocuparse de utilizar y crear sobre el código de la biblioteca para desarrollar la lógica de la aplicación.

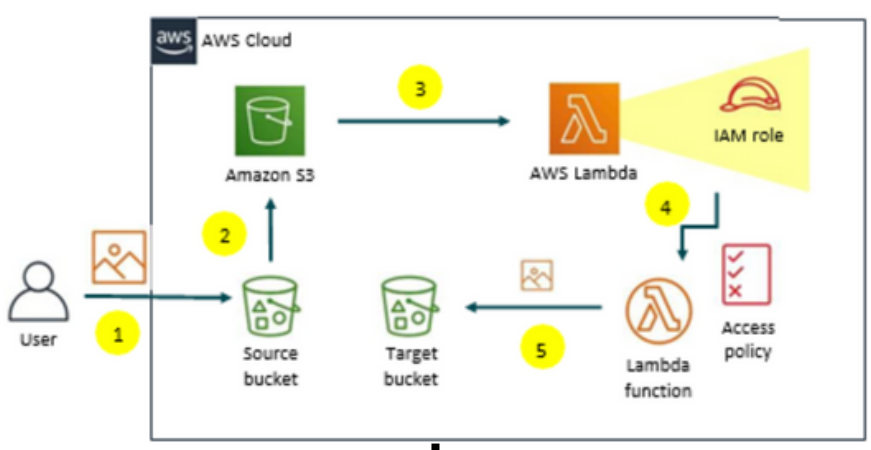
Las capas permiten reducir el tamaño del paquete de implementación, lo que facilita el desarrollo.

Una función puede utilizar hasta cinco capas a la vez. El tamaño total descomprimido de la función y todas las capas no puede superar el límite del tamaño del paquete de implementación sin comprimir, 250 MB.

Para obtener más información sobre las capas, consulte Capas de AWS Lambda.

Ahora, el instructor puede hacer una demostración de cómo configurar un evento de Amazon S3 para desencadenar una función de Lambda.

Diagrama de demostración



Compartir el código de esta forma puede ayudar a promover la división de responsabilidades. Una persona puede encargarse de gestionar la biblioteca principal. Otra persona puede ocuparse de utilizar y crear sobre el código de la biblioteca para desarrollar la lógica de la aplicación.

Esta demostración cubre los siguientes pasos: Un usuario carga una imagen en un depósito de origen S3. Amazon S3 detecta el evento de creación de objetos. Amazon S3 publica el evento a Lambda invocando la función Lambda y pasando los datos del evento como parámetro de función. Lambda asume un rol IAM que le permite ejecutar la función, y luego ejecuta la función Lambda. A partir de los datos del evento que recibe, la función Lambda conoce el depósito de origen y el nombre clave del objeto. La función Lambda lee el objeto, crea una miniatura de la imagen original y guarda la miniatura en el depósito de destino S3.

```
import boto3
import os
import sys
import uuid
from PIL import Image
import PIL.Image

s3_client = boto3.client('s3')

def resize_image(image_path, resized_path):
    with Image.open(image_path) as image:
        image.thumbnail((128, 128))
        image.save(resized_path)

def handler(event, context):
    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        key = record['s3']['object']['key']
        download_path = '/tmp/{}{}'.format(uuid.uuid4(), key)
        upload_path = '/tmp/resized-{}'.format(key)

        s3_client.download_file(bucket, key, download_path)
        resize_image(download_path, upload_path)
        s3_client.upload_file(upload_path, '{}-resized'.format(bucket), key)
```

Receives S3 event and downloads the image to local storage

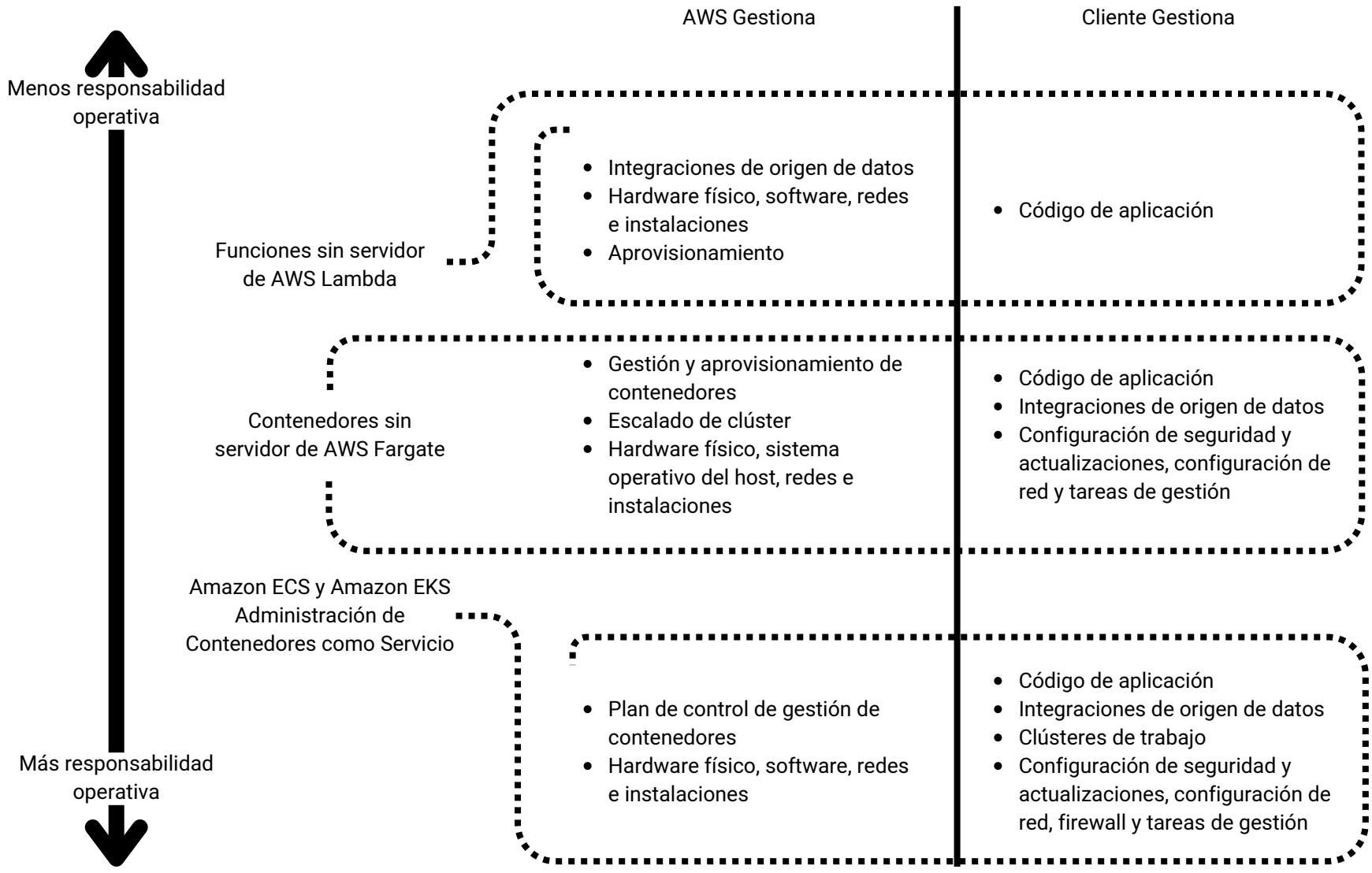
Resizes the image

Uploads resized image to the –*resized* bucket

El código de la función Lambda realiza los siguientes pasos:

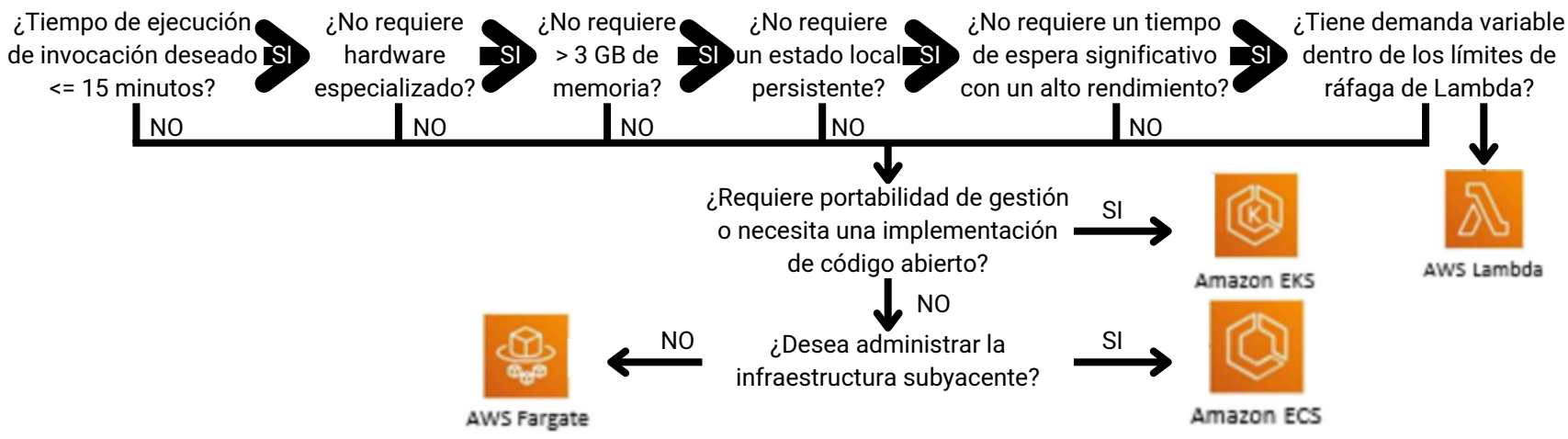
- Recibe un evento de S3, que contiene el nombre del objeto entrante (Bucket, Key).
- Descarga la imagen al almacenamiento local.
- Redimensiona la imagen utilizando la biblioteca Pillow.
- Sube la imagen redimensionada al bucket S3 especificado para imágenes redimensionadas.

Comparación de la responsabilidad operativa para arquitecturas de contenedores y sin servidor



Qué servicio de cómputo utilizas para construir tu aplicación depende del nivel de control que desees. Compartes un espectro de responsabilidad operativa con AWS sobre tus opciones de cómputo para arquitecturas de contenedores y sin servidor.

Elección de una plataforma de cómputo: Contenedores vs AWS Lambda



Esta es una muestra una guía aproximada para seleccionar tu plataforma de cómputo. La recomendación se basa en los límites de AWS Lambda (limitación de memoria, limitación de tiempo).

Entre los puntos clave de esta Lección del unidad, se incluyen los siguientes:

- Lambda es un servicio de cómputo sin servidor que proporciona tolerancia a errores integrada y escalado automático.
- Una función de Lambda es un código personalizado que se escribe para procesar eventos.
- Un controlador invoca una función de Lambda, que toma como parámetros un objeto de evento y un objeto de contexto.
- Un origen de eventos es un servicio de AWS o una aplicación creada por el desarrollador que desencadena una función de Lambda para que se ejecute.
- Las capas de Lambda permiten que las funciones compartan código y limitan el tamaño de los paquetes de implementación.

Laboratorio guiado 2:

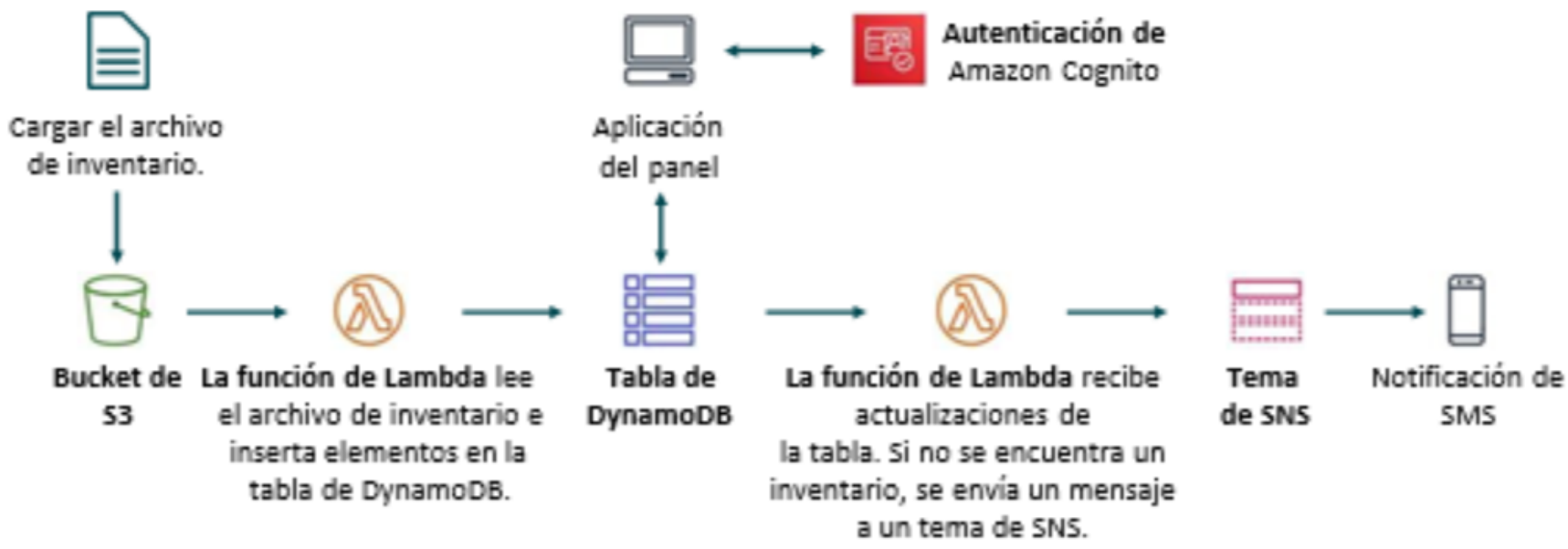
Implementación de una arquitectura sin servidor en AWS

En este laboratorio guiado, realizará las siguientes tareas:

1. Crear una función de Lambda para la carga de datos
2. Configurar un evento de Amazon S3
3. Probar el proceso de carga
4. Configurar las notificaciones
5. Crear una función de Lambda para enviar notificaciones
6. Probar el sistema

Laboratorio guiado 2:

Producto final



El diagrama resume lo que creará después de completar el laboratorio.