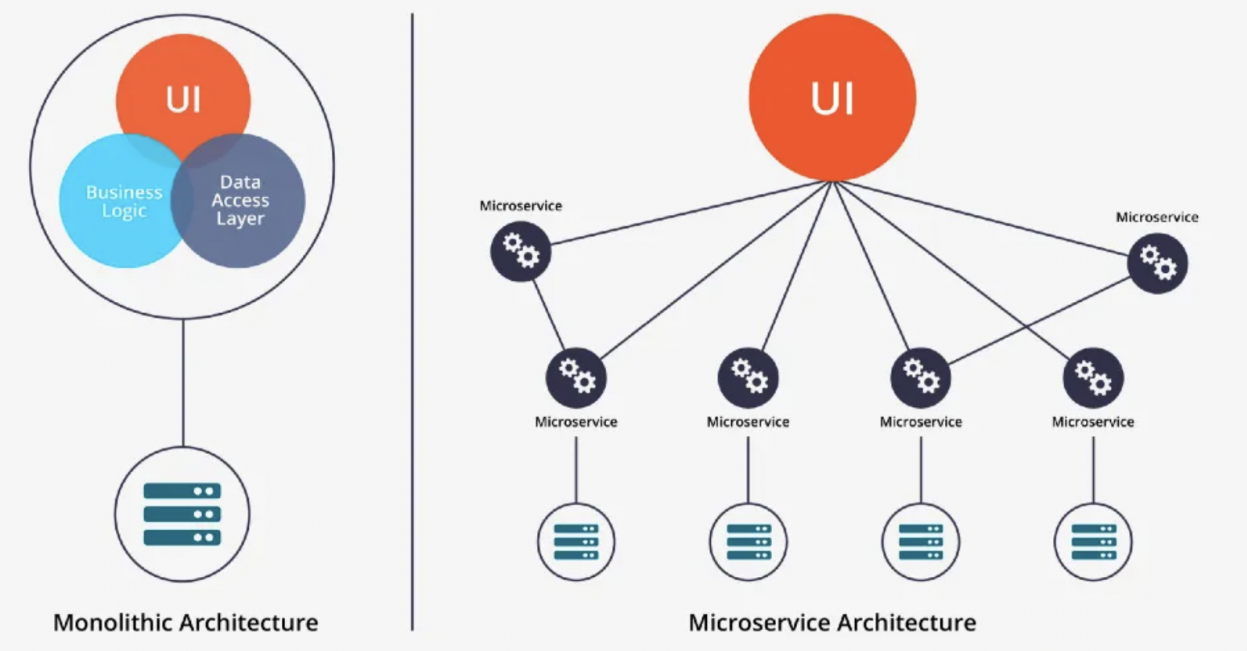
Final Exam

1. **Describe los pros y cons de una arquitectura de software de microservicios.**

Arquitectura de microservicios

Primeramente recordemos lo que es una arquitectura de microservicios. Cuando hacemos referencia a este concepto, se refiere a un software cuya estructura se conforma por un conjunto de multiples servicios pequeños que son ejecutados de manera independiente y representan una funcionalidad específica. Cada uno puede ser ejecutado y modificado sin afectar los otros servicios.



Como funcionan los microservicios?

Los microservicios en este tipo de arquitectura se comunican mediante APIs, estos son aquellos comandos, funciones y protocolos que ya se encuentran definidos. Los desarrolladoras aplican las librerías de las API a su desarrollo para la integración de

servicios. Cada uno de los microserivicios cuenta con su propio sistema de almacenamiento, evitando así la sobrecarga y posible caída de la página/aplicación.

Ventajas

- **Versatilidad:** el uso de microservicios permite utilizar diferentes tecnologías y lenguajes. Esto es altamente benéfico, ya que las funcionalidades se pueden adaptar cada una a las más adecuadas y rentables dependiendo el servicio.
- **Modularidad:** al tratarse de servicios totalmente autónomos, estos se pueden desarrollar y ejecutar de manera independiente, evitando afectar el funcionamiento de los otros servicios.
- **Escalabilidad**: debido a que los servicios se encuentran separados, es decir, es una aplicación modular, se puede escalar de manera horizontal de acuerdo a las necesidades de la empresa.
- **Mantenimiento simple y barato:** al poderse trabajar en un módulo a la vez sin necesidad de intervenir en toda la estructura, el mantenimiento se vuelve mucho mas sencillo y económico en comparación con otras arquitecturas.
- **Agilidad**: los desarrolladores pueden aprovechar funcionalidades que ya han sido desarrolladas por terceros, no se necesita recrearlas.
- Altamente funcional dentro de las metodologías ágiles, debido a que los microservicios se pueden desplegar según sea necesario.

Desventajas

- **Perfil de desarrolladores:** los microservicios requieren de desarrolladores con mucha experiencia y altos conocimientos sobre solución de problemas, tales como latencia de redes y balanceo de cargas.
- **Uniformidad:** poder disponer de un conjunto de diferentes tecnologías para cada servicio tiene sus ventajas, sin embargo, si no es gestionado de manera correcta, se tiene como resultado un diseño y estructura poco uniforme.
- **Pruebas complicadas:** debido a que se trata de una aplicación con sus funcionalidades distribuidas, las pruebas globales son mucho más complicadas de realizar.

- **Tiempo inicial:** en los comienzos del desarrollo de la arquitectura, se requiere de una mayor inversión de tiempo para fragmentar los multiples microservicios y realizar la integración de la comunicación entre ellos.
- **Gestión**: la gestión e integración de los microservicios se vuelve complicada al trabajar con un gran número de ellos, es fundamental contar con herramientas avanzadas que permitan tener un visión general de todos los microservicios y que dirija el sistema.
- **Consumo de memoria:** debido a que los microservicios cuentas con sus propios recursos y bases de datos, se consume más memoria y CPU.

2. Explica la arquitectura de Publish and Subscribe

¿Qué es?

PubSub es una arquitectura que ofrece un servicio de mensajería asíncrono, que permite comunicarse entre múltiples usuarios en tiempo real.

Existen varios conceptos claves respecto a este servicio:

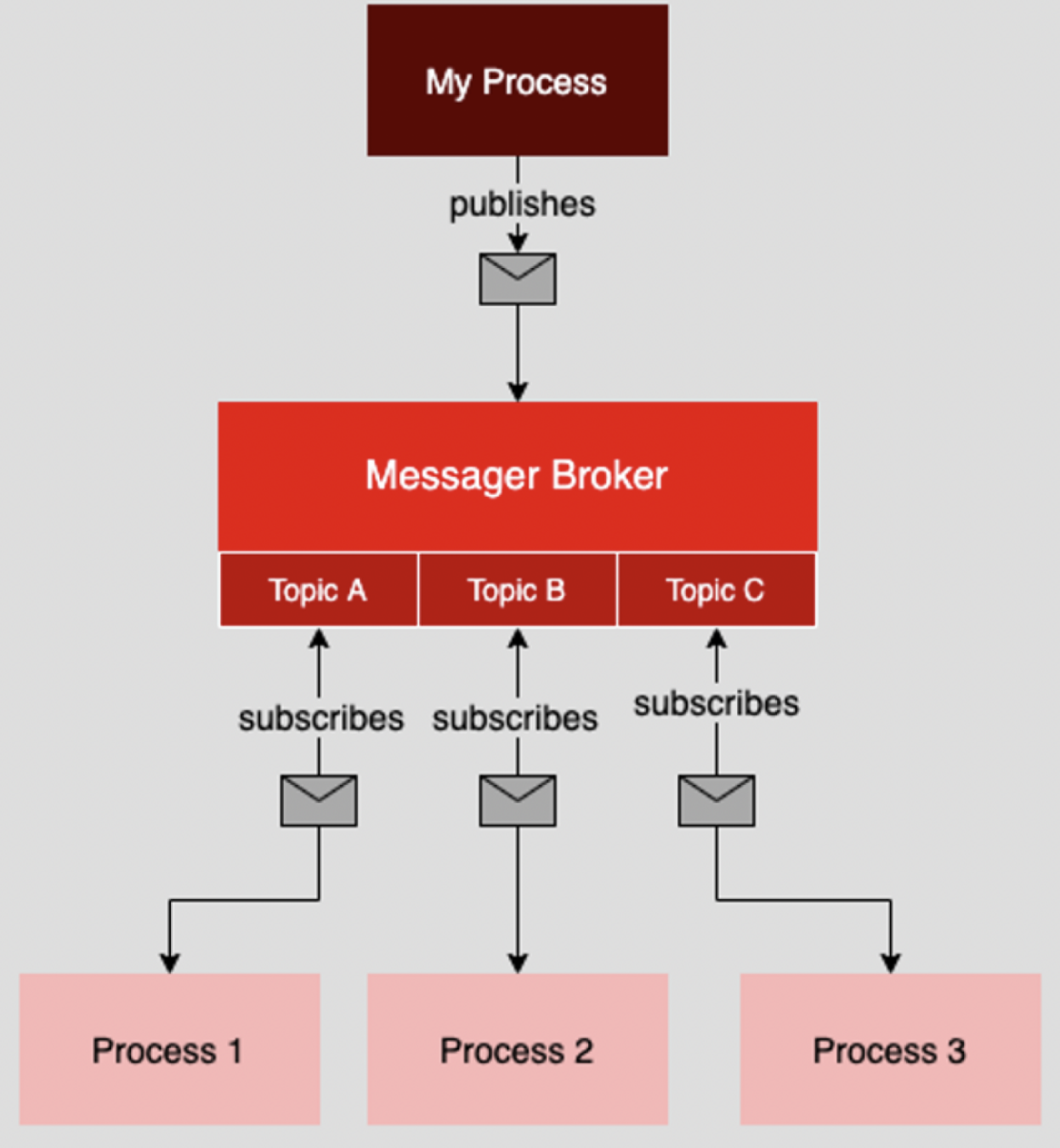
- **Mensaje**: los datos que serán transferidos a través del servicio.
- **Tema**: una entidad que representa un feed de mensajes. (bandeja de entrada)
- **Suscripción**: entidad que representa quien quiere recibir mensajes sobre un tema en específico.
- **Publicador(Productor)**: crea mensajes y los envía/publica al servicio dentro de un tema específico.
- **Suscriptor(Consumidor)**: recibe mensajes de una suscripción específica.

¿Cómo funciona?

Esta arquitectura es aquella en la que un proceso envía un mensaje a un intermediario (message broker), donde después el mensaje se reenvía a uno o varias partes que se encuentran escuchando los mensajes entrantes dependiendo un tema determinado. Una vez que se vincula un tema y después se escucha por los mensajes que van llegando a la bandeja de entrada, se denomina suscripción.

Se puede definir en tres pasos sencillos:

- El publicador crea un canal(tema).
- Todos los suscriptores interesados en un canal(tema), pueden suscribirse a él.
- El publicador envía mensajes a través del canal(tema) y todos los suscriptores al canal reciben los mensajes, sin que cada entidad tenga conocimiento de la contraparte.



Ventajas

- Al tratarse de un sistema asíncrono, si un proceso queda atrapado en un interacción de intercambio de datos de larga duración, es mínimo el riesgo de la degradación del rendimiento del mismo.

- Alta escalabilidad y flexibilidad, no se quiere programación compleja para añadir o quitar suscriptores a un tema en particular, es manejado por configuración.
- Tanto el publicador como el suscriptor no necesitan conocer la dirección de la otra entidad. Comparando con los HTTP requests de un microservicio, donde si se requiere la dirección IP.

Desventajas

- Realizar pruebas se vuelve un poco complejo, debido a tratarse de interacciones asíncronas, no es simplemente realizar un request y analizarlo, sino que se debe enviar un mensaje y la prueba consiste en observar el comportamiento durante todo el proceso y ver cuando y como el mensaje es manejado. Entre más mensajes se analicen referente a un tema con el paso del tiempo, es más complicado el manejo de las pruebas.
- Aumentos inesperados en la transferencia de mensajes puede causar lo que se conoce como cuello de botella en la red y provocar resultados no favorables para el intermediario de mensajes que los consume, es decir, el suscriptor.
- Se requiere tener bien definidas las políticas referentes al formato de mensajes y el intercambio de los mismos, de no ser así puede ser propenso a errores y alteraciones en su consumo.
- Si se incorporan sistemas externos a la arquitectura, el encriptado de información sensible/privada que es publicada puede verse afectada.
- El almacenamiento que se requiere es alto, especialmente en el caso de existir miles the suscriptores a un tema, algunos son lentos, por lo que se debe conservar la información hasta que el más lento (en espera) sea transferido.

3. **Crea una Lambda + API Gateway que reciba como parámetro un rango de fechas con formato YYYYMMDD y regrese la cantidad de días, horas y minutos entre dichas fechas.**

ARN del lamba

arn:aws:lambda:us-east-1:292274580527:function:datesMichelle

ARN del API

arn:aws:execute-api:us-east-1:292274580527:iul3dat1yg//*GET/range/*

Endpoint

https://iul3dat1yg.execute-api.us-east-1.amazonaws.com/dev/range/{dateRange}

4. **Replica la respuesta anterior, pero en este nuevo end point regresa HTML, creando un dynamic web site.**

ARN del lamba

arn:aws:lambda:us-east-1:292274580527:function:datesMichelleHTML

ARN del API

arn:aws:execute-api:us-east-1:292274580527:oag0d5tcul//*GET/range/*

Endpoint

https://oag0d5tcul.execute-api.us-east-1.amazonaws.com/dev/range/{dateRange}

5. **Describe la diferencia entre los S3 object storage class standard, infrequent access y Glacier.**

S3 Standard

Este servicio de Amazon ofrece almacenamiento de objetos de alta durabilidad, disponibilidad y rendimiento para aquellos datos a los que se requiere acceder frecuentemente. Apropiado para multiples casos de uso, tales como aplicaciones en la nube, páginas web dinámicas, distribución de contenido, análisis de big data, etc.

Algunas características:

- Ofrece baja latencia y un alto nivel de procesamiento.
- Diseñado para ofrecer un alto nivel de durabilidad de los objetos en varios zonas de disponibilidad
- Tiene resistencia a los eventos que pueden afectar una zona de disponibilidad completa.
- Diseñado para ofrecer una disponibilidad casi completa (99.99%) por un año completo.
- Admite SSL para datos en tránsito y cifrado de datos en reposo.
- Ofrece la posibilidad de migrar de manera automática de objetos a todas las otras clases de almacenamiento pertenecientes a S3(incluyendo Glacier).

S3 Standard - Infrequent Access

Este servicio se utiliza con datos a los que no se requiere acceder de manera frecuente, en comparación con el anterior, sin embargo, requieren de un acceso rápido cuando se hace una solicitud. Cuenta con las mismas características que S3 Standard, en cuanto a la durabilidad, alto procesamiento y mismo nivel de latencia, pero ahora hay un costo pequeño de almacenamiento por GB y de recuperación por GB.

- Diseñado para ofrecer un alto nivel de durabilidad de los objetos en varios zonas de disponibilidad
- Tiene resistencia a los eventos que pueden afectar una zona de disponibilidad completa.
- Los datos son resilientes en caso de que ocurra una destrucción total de una zona de disponibilidad.
- Diseñado para ofrecer una disponibilidad casi completa (99.99%) por un año completo.
- Admite SSL para datos en tránsito y cifrado de datos en reposo.
- Ofrece la posibilidad de migrar de manera automática de objetos a S3 One Zone-IA o S3 Glacier.

Existe otro servicio muy similar a este servicio, **S3 One Zone - Infrequent Access.** Este tiene la misma aplicación que S3 Standard - IA, sin embargo, en este se almacenan los datos en una sola zona de disponibilidad y cuesta un 20% menos que S3 Standard - IA. S3 One Zone - IA es ideal para los aquellos que desean una opción de menor costo y que no necesitan el mismo nivel de disponibilidad o resiliencia que ofrece S3 Standard o s3 Standard - IA. Un caso de uso muy común, es para el almacenamiento de copias de seguridad secundarias de datos locales o datos que se pueden recrear fácilmente. Al igual que S3 Standard - IA hay un cargo por cada GB de almacenamiento y recuperación.

	S3 Standard	S3 Standard - IA	S3 One Zone - IA
Durablity	99.99999999999%	99.99999999999%	99.99999999999%
Availability	99.99%	99.9%	99.5%
Number of Availability Zones	At least 3	At least 3	Only 1
Minimum capacity charge per object	N/A	128KB	128KB
Minimum storage duration charge	N/A	30 days	30 days
Retrieval fee	N/A	Per GB retrieved	Per GB retrieved
Storage Transition	S3 Standard to all other s3 storage types (including Glacier)	To S3 One Zone - IA or S3 Glacier	To S3 Glacier

S3 Glacier

Esta clase de almacenamiento es para archivos a los que no se necesita acceder frecuentemente y/o inmediatamente. Si bien ofrece el mayor rendimiento, la mayor flexibilidad de recuperación y el menor costo de almacenamiento en la nube, los datos no se pueden recuperar inmediatamente cuando se desee. Con Amazon Glacier la recuperación de archivos puede llegar a tardar varias horas(dependiendo el tipo de servicio), lo que lo hace ideal para archivar.

Algunas de las opciones son:

- **S3 Glacier Instant Retrieval:** ofrece el almacenamiento de menor costo y una recuperación de datos en cuestión de milisegundos.
- **S3 Glacier Flexible Retrieval:** ideal para los datos que no requieren acceso inmediato pero si necesitan flexibilidad de recuperación de grandes cantidades de datos sin costo alguno. Cuenta con una recuperación de datos en minutos o en caso de recuperaciones masivas gratuitas, de 5 a 12 horas.
- **S3 Glacier Deep Archive:** ideal para ahorras lo más posible en el almacenamiento de archivos de larga duración. Ofrece el almacenamiento de menor costo en la nube con una recuperación de datos de entre 12 a 48 horas.

Sin duda la principal diferencia entre estos servicios que ofrece Amazon, es la aplicación de uso, es decir, en qué tipo de casos es viable utilizarlos. S3 Standard para cuando se requiera acceder a datos de manera frecuente, a diferencia de S3 Standard - IA que se utiliza con datos que no se acceden frecuentemente y además ya cuenta con un costo por GB de almacenamiento y recuperación. S3 Glacier es ideal para cuando el acceso a datos no requiere ser frecuente y tampoco inmediato. Debido a que los niveles de frecuencia y respuesta de recuperación son diferentes, los costos son diferentes. De igual manea, otra diferencia es la transición de almacenamiento que pueden realizar los servicios o otros tipos de clases.

6. **Explica cómo hacer route de un subdominio de Route 53 a un static web site en S3.**

Primeramente se tienen que tener las configuraciones de un bucket en S3 para poder tener un static web site. Se comienza por crear un JSON file que contiene los cambios de aws Route53. Este servicio se utiliza para conectar mi página web como un subdominio de una ya existente. A este archivo se le denomina CNAME record. En este caso tomare como ejemplo el que fue creado para hostear mi página web.

El CNAME record se utiliza para asignar nombres a dominios o subdominios. Todos estos registros deben estar apuntando a un dominio, en este caso apunta al url del bucket de la página cetystijuana.com.

En cuanto a los parámetros definidos en el archivo:

- **Action**: se define como CREATE, debido a que estamos creando un DNS record.
- **ResourceRecordSet:** es un arreglo que contiene:
  - **Name**: es el nombre del subdominio, en este caso es michelle.cetystijuana.com
  - **Type**: indica el tipo de record que se esta creando
  - **TTL**: indica por cuanto tiempo el record debe estar almacenado antes de buscarse una actualización
  - **ResourceRecords:** se tiene el **value,** el cual contiene el URL del bucket.

7. Escribe el código en python para leer y borrar un mensaje de SQS

Primeramente, se importan todas las librerías que se requieren para acceder a los servicios de AWS y para el desarrollo del código.

```
import boto3
from datetime import datetime
import time
```

Código que fue realizado en clase. Se tiene una función que se encarga de ambas acciones, tanto la lectura como la eliminación de un mensaje. Primero en la función *process_messages()* se crea el cliente con el servicio de SQS, quien va a proveer una interfaz para acceder a las funcionalidades de aws. Posteriormente se obtiene el URL del queue que se esta especificando, en este caso test_michelle.

Posteriormente se crea el mensaje y se envía con la función *send_message()*, en esta se envía tanto el URL del queue como un mensaje donde indica el número de mensaje que esta siendo enviado y la fecha con hora del envío.

El while True se utiliza para que siempre se este escuchando si viene un mensaje o no, es decir, siempre esta leyendo. De acuerdo al URL del queue se obtiene el mensaje y finalmente, se utiliza la función *delete_message()* para borrar un mensaje del queue y se indica el mensaje que fue eliminado. Para ello se debe indicar tanto el url del queue como el mensaje (mediante ReceiptHandle).

```python
def process_messages():
    sqs_client = boto3.client("sqs")
    queue_url = sqs_client.get_queue_url(QueueName='test_michelle')['QueueUrl']
    print(queue_url)
    date_time = datetime.now()

    message = sqs_client.send_message(
        QueueUrl=queue_url,
        MessageBody=("Mensaje numero {0} enviado a las {1} ".format(1,
                    str(date_time.strftime('%Y-%m-%d %H:%M:%S'))))
    )
    print(message)

    while True:
        print("Queue URL: {0}".format(queue_url))
        message = sqs_client.receive_message(QueueUrl=queue_url)
        print("Message received: {0}".format(message))
        receipt = message["Messages"][0]["ReceiptHandle"]

        time.sleep(100)


    response = sqs_client.delete_message(
        QueueUrl=queue_url,
        ReceiptHandle=receipt
    )
    print("Message deleted: {0}".format(response))


process_messages()
```

8. Explica las diferencias entre SNS y SQS

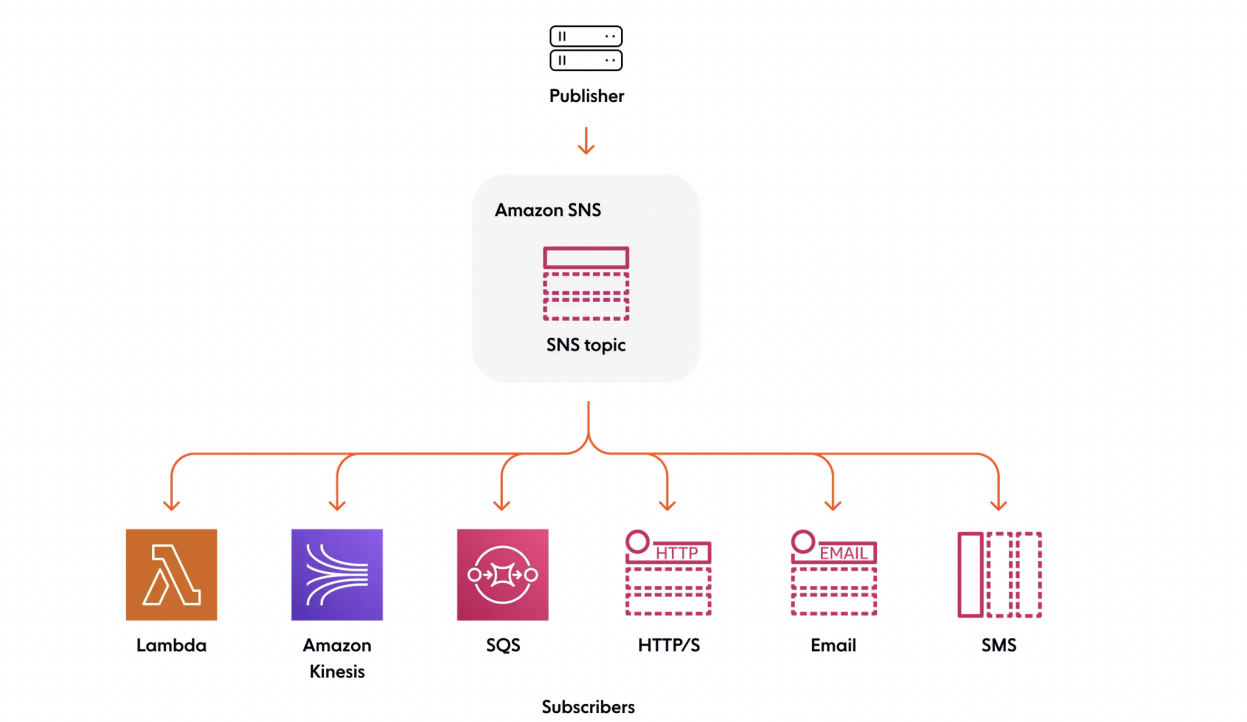
SNS (Simple Notification Service)

Este fue uno de los primeros servicios lanzados en Amazon Web Services (AWS), es una solución distribuida de PubSub que se utiliza para la comunicación de lo que se conoce como application-to-application (A2A) y application-to-person (A2P).

Recordando un poquito:

- A2A: comunicación entre dos aplicaciones a través de un intercambio de mensajes.
- A2P: proceso en el que se envía un mensaje desde una aplicación de software a un dispositivo móvil.

SNS topic es utilizado para permitir la comunicación entre los productores(publishers) y los consumidores(consumers). Los productores publican mensajes en los topic y los consumidores se suscriben a estos temas para recibir los mensajes. Se pueden entregar mensajes a varios tipos de suscriptores, tales como AWS SQS, AWS Lambda y HTTP endpoints. De igual manera, se puede utilizar SNS para enviar mensajes SMS, correo electrónico y notificaciones automáticas a los dispositivos de un usuario final.



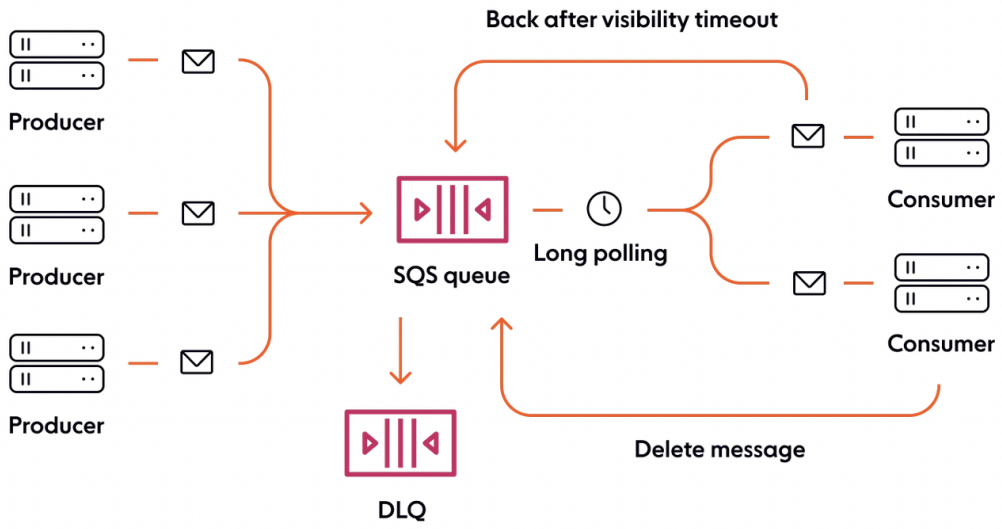
Existen dos tipos de topics:

- **Standard topics:** ofrecen el máximo rendimiento, pedidos de mejor esfuerzo y deduplicación de mejor esfuerzo(quiere decir que los mensajes pueden ser entregados más de una vez).
- **FIFO topics:** provee ordenamiento estricto y deduplicación. Sin embargo, únicamente se puede usar colas de SQS para recibir los eventos proporcionados por los temas FIFO, no se permite otro tipo de consumidor.

SQS (Simple Queue Service)

Este es un servicio de amazon de cola administrado y distribuido utilizado para manejar la comunicación entre aplicaciones, microservicios y sistemas distribuidos. Al igual que con otros softwares intermediarios de mensajería, SQS esta compuesto por 3 partes fundamentales:

- **Producers:** aquellos que envían los mensajes a una cola.
- **Queue:** cola que almacena el mensaje.
- **Consumers:** quienes reciben los mensajes de la cola.



Existen dos tipos de cola:

- **Standard queues:** ofrecen el máximo rendimiento, ordenamiento de mejor esfuerzo y entrega al menos una vez.
- **FIFO queues:** diseñado para garantizar que los mensajes se procesen únicamente una vez, de acuerdo al orden en el que son enviados.

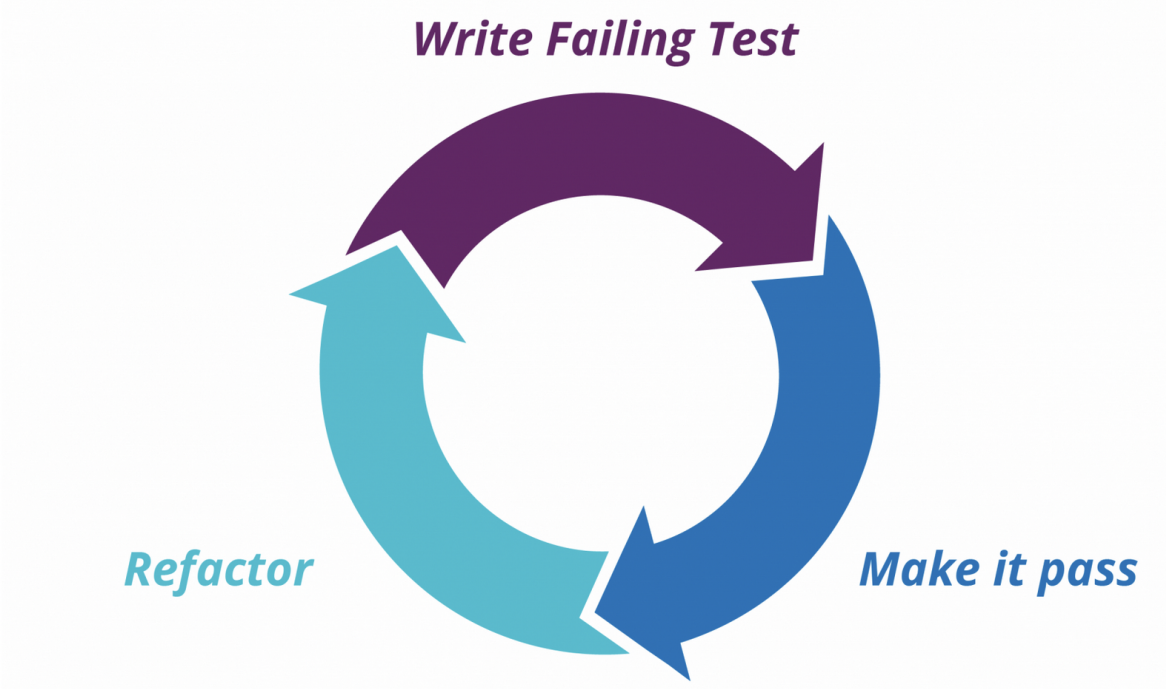
En cuanto a las diferencias entre estos servicios, primeramente son las comunicaciones que manejan, SNS soporta tanto A2A como A2P, mientras que SQS solo soporta A2A. Otra principal característica es el tipo de sistema de cada uno, SNS siendo un sistema PubSub y SQS un sistema de colas.

Otras diferencias son:

- En cuanto a la persistencia, SN no cuenta con ello ya que entrega los mensajes a los suscriptores presentes y después son eliminados. En comparación, SQS si persiste los mensajes de entre 1 minuto a 14 días.
- SNS es comúnmente utilizado para aplicaciones que requieren de notificaciones en tiempo real, mientras que SQS es más adecuado para casos de uso de procesamiento de mensajes.
- Para la entrega de mensajes SQS utiliza el mecanismo *long polling* y SNS usa un mecanismo push para entregar de manera inmediata los mensajes a los suscriptores.
    - **Long polling:** mecanismo en la que un cliente solicita información del servidor sin esperar una respuesta inmediata, que básicamente puede solicitar una solicitud HTTP a un servidor y mantener la conexión abierta para permitir una respuesta más tarde.

9. **Explica el ciclo de TDD**

TDD por sus siglas en ingles significa Test Driven Development, este es un proceso en el que se escriben las pruebas de un software antes de escribir el código. Esta práctica no solo mejora la calidad del software final, sino que ayuda a reducir los costes de mantenimiento, ya que primero se escriben las pruebas, después se escribe el código fuente, una vez que se pasen correctamente las pruebas y se termina con lo que es la refactorización del código escrito.



- **Etapa morada:** Esta etapa es el comienzo del ciclo TDD y se crea la prueba de validación, la cual es un test que debe fallar. En esta parte del proceso, el desarrollador debe tener conocimiento total de los requerimientos del software, todas las condiciones y los casos bordes.
- **Etapa azul:** En esta etapa se lleva acabo el desarrollo del código con el cual se realizan las validaciones de las diferentes pruebas. Se debe conseguir que todas las pruebas establecidas pasen. Una vez que sucede esto, se puede decir que el software es válido y cumple con los requerimiento establecidos.
- **Etapa azul claro:** Esta es la etapa final, en la que se refactoriza el código, la implementación de la prueba creada se limpia y se reconducen las necesidades que han ido surgiendo a lo largo del proceso. Terminando esto, el proceso se repite.

10. **Explica si el amor lo puede todo y por qué.**

Sin duda, considero que esta es una frase que genera mucho debate y depende del punto de vista del que se vea. Hablando desde mi experiencia y los momentos difíciles que he vivido, puedo decir que estoy de acuerdo con que el amor lo puede todo. Por

ejemplo, a veces se tiene la mentalidad de cuando alguien cercano a ti te falla, le deseas lo peor y crees que nunca lo vas a poder superar, pero realmente no es así. Si realmente hubo cierto nivel de amor hacía la persona o puede ser alguna cosa que te diera mucha felicidad, cuando empiezan a surgir problemas se buscan maneras para solucionarlo. No es que el amor lo solucione, sino que es lo que te ayuda a seguir adelante y luchar por cambiar las cosas. En el caso de que ya existan muchos problemas y no se vea algún cambio o las cosas van empeorados en lugar de mejorar, el mismo amor propio y el que se tiene a la otra parte nos da las fuerzas para tomar la decisión difícil de tomar caminos separados. Claro que es difícil, pero con el tiempo las cosas mejoran.

Digo esto no necesariamente en contexto de pareja, sino puede ser un gran amigo, un familiar o en mi caso, una de las situaciones que más marcaron mi vida y como soy como persona ahora, fue mi relación con el flag football. Muchos podrían decir “que tontería”, pero el irme dando cuenta que lo que más me apasionaba en la vida y a lo que en un momento me quería dedicar, ya no lo disfrutaba como antes fue muy doloroso. Por el amor que le tenía, luche y luche varias veces para volver a sentir lo mismo de antes, pero me di cuenta que mi salud mental estaba siendo muy afectada por ello y fue cuando tome una de las decisiones más difíciles en mi vida y deje de jugar. Reflexionando, el amor que le tenía al deporte, por más doloroso que fuera me permitió alejarme antes de tener una relación negativa con el flag; al igual que el amor que me tengo yo y evitar dañarme más.

Como este, creo que hay muchos casos en los que no diría que el amor es la solución, pero el amor te da las fuerzas para superar las adversidades que se presentan en la vida. Además, cuando se hacen las cosas con amor las cosas son mucho más bonitas y se disfruta de la vida.