[ENGLISH]

Hello cloud gurus and welcome to this lesson,

which is a summary of all my exam tips

for this section of the course,

beginning with SQS, which stands for Simple Queue Service.

And it's a distributed message queueing system

which allows you to decouple

the components of an application

so that they are independent of each other.

So if you see anything in the exam

related to decoupling your application components,

I want you to think of SQS.

Remember that SQS is pull based, not push-based.

And there are two different types

of SQS queue to be aware of.

There's the standard queue and also the FIFO queue

and you'll need to understand the differences.

So with a standard queue,

you get best effort ordering for the messages.

Each message is going to be delivered at least once,

and you can occasionally see duplicate messages,

so your application needs to be able to cope with that.

And a standard queue is the default queue type.

FIFO stands for First In First Out.

And with a FIFO queue,

the message order is strictly preserved.

Messages are delivered only once.

It doesn't introduce duplicates.

And this is great for banking transactions

where duplicate transactions would be a big problem

or for transactions that need to happen in a strict order.

Visibility timeout, is the amount of time

that a message is invisible in the SQS queue

after a reader has picked it up and begun to process it.

The default is 30 seconds.

And if it's gonna take longer than 30 seconds

to process the message,

then you will need to increase the visibility timeout

up to a maximum of 12 hours.

With short polling, a response is returned immediately

even if no messages are in the queue.

And you will pay for every response that you receive

even if the queue is empty.

Whereas with long polling, you periodically poll the queue

and you only get a response

when a message is actually in the queue

or if a timeout has been reached.

And this is the most cost effective option

because you don't end up paying

for loads of empty responses.

And in general, long polling is recommended

because it is the most cost effective option.

Moving on to SQS delay queues.

And a delay queue allows you to postpone

delivery of new messages.

You set a delay period and messages in the delay queue

remain invisible for the duration of the delay period.

And this is great for large distributed applications

that may need to introduce a delay in processing,

for instance, if you need to introduce,

say a three second delay between jobs or messages.

Now when it comes to large SQS messages,

SQS actually handles that by storing

the message payload in S3.

And in order to do that,

you will need to do a couple of things.

So first of all, you will need to use

the SQS Extended Client Library for Java,

which is used to manage the messages.

And then to perform S3 operations,

you will need the AWS SDK for Java

and that provides an API for S3 operations,

like GET and DELETE.

Onto SNS, which is Simple Notification Service.

It supports multiple notification formats

like SMS or text message, SQS, HTTP and email.

It provides push notifications only,

and it doesn't receive notifications.

It uses a publish and subscribe model.

And in order to receive notifications from SNS,

consumers need to subscribe to an SNS topic.

And it also supports fanout,

so it can fanout duplicate messages

to a large number of recipients.

For instance, you can use it to send the same notification

to multiple SQS queues, HTTP end points

and email addresses as well.

Onto SES or Simple Email Service.

And remember, SES is for emails only

and it can be used to send and receive emails.

It's not subscription based.

So all you need to know

in order to start sending emails using SES

is the recipient's email address.

And it's great for sending automated emails

like promotional or marketing emails.

For the exam, you'll need to understand the differences

between SNS and SQS.

So remember, that SNS is a messaging service

that is push based,

so it's pushing notifications out to its subscribers.

And when I think of SNS,

I think about push notifications

that you might receive on your smartphone,

letting you know that your amazon.com,

or UPS delivery is coming this afternoon.

Whereas SQS is also a messaging service,

but this one is pull based,

with consumers polling the queue for messages.

So think about EC2 instances,

polling the queue to get the messages

and get the next task to work on.

Moving on to Kinesis,

and you will need to understand the differences

between the various services in the Kinesis family.

So we have Kinesis Streams,

which includes video streams and data streams,

and it allows you to capture and store

streaming video and data for realtime processing.

We then have Kinesis Data Firehose,

and this allows you to capture, transform

and load data continuously into AWS data stores

or other service providers like Datadog or Splunk.

And existing business intelligence applications and tools

can be used for near real-time analysis on the stored data.

And then finally, there's Kinesis Data Analytics,

and this provides real-time analytics using standard SQL

on data received by Kinesis Data Streams

and Kinesis Data Firehose,

and it then goes on to store the processed data

in an AWS data store, like S3, Redshift or Elasticsearch.

A Kinesis Stream is made up of one or more shards.

Each shard is a sequence of one or more data records

and provides a fixed unit of capacity

within the Kinesis Stream.

So the data capacity of the stream

is determined by the number of shards.

And if your data rate increases,

then you can increase the capacity on your stream

by increasing the number of shards.

When it comes to consuming the Kinesis Stream,

you've got the Kinesis Client Library

running on your consumers, so on your EC2 instances.

And the Kinesis Client Library runs a record processor

for each shard being consumed by your instance.

So in this example, we have four shards

and a record processor for each shard.

And generally, the number of consumer instances

should not exceed the number of shards

except for failover purposes.

If you increase the number of shards,

that doesn't necessarily require

an increase in the number of consumers.

Instead, you should use an auto-scaling group

to drive scaling actions for your EC2 instances

based on their CPU utilization,

not based on the number of shards in your Kinesis Stream.

Well, that is it for this lesson.

If you have any questions, please let me know.

Otherwise, I will see you in the next one.

Thank you.

[SPANISH]

Hola gurús de la nube y bienvenidos a esta lección.

que es un resumen de todos mis consejos para el examen

para esta sección del curso,

comenzando con SQS, que significa Simple Queue Service.

Y es un sistema de cola de mensajes distribuidos

que le permite desacoplar

los componentes de una aplicación

para que sean independientes entre sí.

Así que si ves algo en el examen

relacionados con el desacoplamiento de los componentes de su aplicación,

Quiero que pienses en SQS.

Recuerde que SQS se basa en la extracción, no en la inserción.

Y hay dos tipos diferentes.

de la cola de SQS a tener en cuenta.

Está la cola estándar y también la cola FIFO

y tendrá que entender las diferencias.

Así que con una cola estándar,

obtienes el mejor esfuerzo para ordenar los mensajes.

Cada mensaje se entregará al menos una vez,

y ocasionalmente puedes ver mensajes duplicados,

por lo que su aplicación debe poder hacer frente a eso.

Y una cola estándar es el tipo de cola predeterminado.

FIFO significa Primero en entrar, primero en salir.

Y con una cola FIFO,

el orden de los mensajes se conserva estrictamente.

Los mensajes se entregan una sola vez.

No introduce duplicados.

Y esto es genial para transacciones bancarias.

donde las transacciones duplicadas serían un gran problema

o para transacciones que deben ocurrir en un orden estricto.

Tiempo de espera de visibilidad, es la cantidad de tiempo

que un mensaje es invisible en la cola de SQS

después de que un lector lo haya tomado y comenzado a procesarlo.

El valor predeterminado es 30 segundos.

Y si va a tomar más de 30 segundos

para procesar el mensaje,

entonces deberá aumentar el tiempo de espera de visibilidad

hasta un máximo de 12 horas.

Con un sondeo corto, se devuelve una respuesta inmediatamente.

incluso si no hay mensajes en la cola.

Y pagarás por cada respuesta que recibas

incluso si la cola está vacía.

Mientras que con el sondeo largo, sondea periódicamente la cola

y solo obtienes una respuesta

cuando un mensaje está realmente en la cola

o si se ha alcanzado un tiempo de espera.

Y esta es la opción más rentable.

porque no terminas de pagar

para un montón de respuestas vacías.

Y, en general, se recomienda un sondeo largo

porque es la opción más rentable.

Pasando a las colas de retraso de SQS.

Y una cola de retraso le permite posponer

entrega de nuevos mensajes.

Establece un período de retraso y mensajes en la cola de retraso

permanecer invisible durante la duración del período de retraso.

Y esto es genial para grandes aplicaciones distribuidas

que puede necesitar introducir un retraso en el procesamiento,

por ejemplo, si necesita presentar,

digamos un retraso de tres segundos entre trabajos o mensajes.

Ahora, cuando se trata de mensajes SQS grandes,

SQS en realidad maneja eso almacenando

la carga del mensaje en S3.

Y para hacer eso,

tendrás que hacer un par de cosas.

Entonces, antes que nada, necesitarás usar

la biblioteca de cliente extendida de SQS para Java,

que se utiliza para gestionar los mensajes.

Y luego para realizar operaciones S3,

necesitará el SDK de AWS para Java

y que proporciona una API para las operaciones de S3,

como OBTENER y ELIMINAR.

En SNS, que es el Servicio de notificación simple.

Admite múltiples formatos de notificación.

como SMS o mensaje de texto, SQS, HTTP y correo electrónico.

Solo proporciona notificaciones automáticas,

y no recibe notificaciones.

Utiliza un modelo de publicación y suscripción.

Y para recibir notificaciones de SNS,

los consumidores deben suscribirse a un tema de SNS.

Y también es compatible con fanout ,

para que pueda desplegar mensajes duplicados

a un gran número de destinatarios.

Por ejemplo, puede usarlo para enviar la misma notificación

a múltiples colas SQS, puntos finales HTTP

y direcciones de correo electrónico también.

En SES o Servicio de correo electrónico simple.

Y recuerda, SES es solo para correos electrónicos

y se puede utilizar para enviar y recibir correos electrónicos.

No está basado en suscripción.

Así que todo lo que necesitas saber

para comenzar a enviar correos electrónicos usando SES

es la dirección de correo electrónico del destinatario.

Y es genial para enviar correos electrónicos automatizados.

como correos electrónicos promocionales o de marketing.

Para el examen, deberá comprender las diferencias

entre SNS y SQS.

Así que recuerda, que SNS es un servicio de mensajería.

que está basado en empuje,

por lo que está enviando notificaciones a sus suscriptores.

Y cuando pienso en SNS,

Pienso en las notificaciones push

que puede recibir en su teléfono inteligente,

haciéndole saber que su amazon.com,

o la entrega de UPS llegará esta tarde.

Mientras que SQS es también un servicio de mensajería,

pero este está basado en pull,

con los consumidores sondeando la cola en busca de mensajes.

Así que piense en las instancias EC2,

sondeando la cola para obtener los mensajes

y obtén la siguiente tarea en la que trabajar.

Pasando a Kinesis,

y tendrá que entender las diferencias

entre los diversos servicios de la familia Kinesis.

Así que tenemos Kinesis Streams,

que incluye transmisiones de video y transmisiones de datos,

y te permite capturar y almacenar

transmisión de video y datos para procesamiento en tiempo real .

Luego tenemos Kinesis Data Firehose ,

y esto te permite capturar, transformar

y cargue datos continuamente en almacenes de datos de AWS

u otros proveedores de servicios como Datadog o Splunk .

Y las aplicaciones y herramientas de inteligencia comercial existentes

se puede utilizar para el análisis casi en tiempo real de los datos almacenados.

Y finalmente, está Kinesis Data Analytics,

y esto proporciona análisis en tiempo real usando SQL estándar

sobre los datos recibidos por Kinesis Data Streams

y Kinesis Data Firehose ,

y luego pasa a almacenar los datos procesados

en un almacén de datos de AWS, como S3, Redshift o Elasticsearch .

Un flujo de Kinesis se compone de uno o más fragmentos.

Cada fragmento es una secuencia de uno o más registros de datos

y proporciona una unidad fija de capacidad

dentro de Kinesis Stream.

Así que la capacidad de datos de la corriente

se determina por el número de fragmentos .

Y si tu tarifa de datos aumenta,

entonces puedes aumentar la capacidad en tu transmisión

aumentando el número de fragmentos .

Cuando se trata de consumir Kinesis Stream,

tienes la biblioteca de clientes de Kinesis

ejecutándose en sus consumidores, por lo que en sus instancias EC2.

Y Kinesis Client Library ejecuta un procesador de registros

por cada fragmento consumido por su instancia.

Entonces, en este ejemplo, tenemos cuatro fragmentos

y un procesador de registro para cada fragmento .

Y, en general, la cantidad de instancias de consumidores

debe exceder el número de fragmentos

excepto para fines de conmutación por error.

Si aumenta el número de fragmentos,

eso no necesariamente requiere

un aumento en el número de consumidores.

En su lugar, debe usar un grupo de escalado automático

para impulsar acciones de escalado para sus instancias EC2

en función de su utilización de la CPU,

no se basa en la cantidad de fragmentos en su Kinesis Stream.

Bueno, eso es todo por esta lección.

Si tiene alguna pregunta, por favor hágamelo saber.

De lo contrario, te veré en la próxima.

Gracias.